An Introduction to Description Logics and Query Rewriting

Roman Kontchakov and Michael Zakharyaschev

Department of Computer Science and Information Systems,
Birkbeck, University of London, U.K.

Abstract. This chapter gives an overview of the description logics underlying the OWL 2 Web Ontology Language and its three tractable profiles, OWL 2 RL, OWL 2 EL and OWL 2 QL. We consider the syntax and semantics of these description logics as well as main reasoning tasks and their computational complexity. We also discuss the semantical foundations for first-order and datalog rewritings of conjunctive queries over knowledge bases given in the OWL 2 profiles, and outline the architecture of the ontology-based data access system Ontop.

1 Introduction

The first aim of this chapter is to introduce and discuss logic-based formalisms that underpin the OWL 2 Web Ontology Language and its three tractable profiles: RL, EL and QL. OWL 2 is a rather involved language that was designed to represent knowledge about various domains of interest in a machine-accessible form. The diagram in Fig. 1, taken from the official W3C document,[1] shows the general structure of OWL 2. As follows from the diagram, there are (at least) five syntaxes for OWL 2 (various tools can use their own versions). In this chapter, we consider a sixth one, the language of Description Logic (DL). Although not covering all the bells and whistles of the full OWL 2, it will allow the reader to quickly grasp the meaning of the main modelling constructs that OWL 2 provides. The language of DL is elegant and concise because it stems from the formalisms that have been developed in mathematical logic since the middle of the 19th century. It is underpinned by the crystal-clear model-theoretic semantics developed by A. Tarski since the mid 1930s (as shown in the diagram in Fig. 1, OWL 2 has two semantics: RDF-based[2] and Direct Semantics[3]; the latter is based on the model-theoretic semantics of DLs).

We will introduce, in Section 2, most important modelling constructs of OWL 2 and their semantics in terms of the description logics $\mathcal{ALCHI}$ and $\mathcal{SROIQ}$ and the model-theoretic semantics. We then explain fundamental reasoning tasks such as checking consistency (satisfiability), concept and role subsumption, instance checking and conjunctive query answering, and discuss their

[1] www.w3.org/TR/owl2-overview
[2] www.w3.org/TR/owl2-rdf-based-semantics
[3] www.w3.org/TR/owl2-direct-semantics

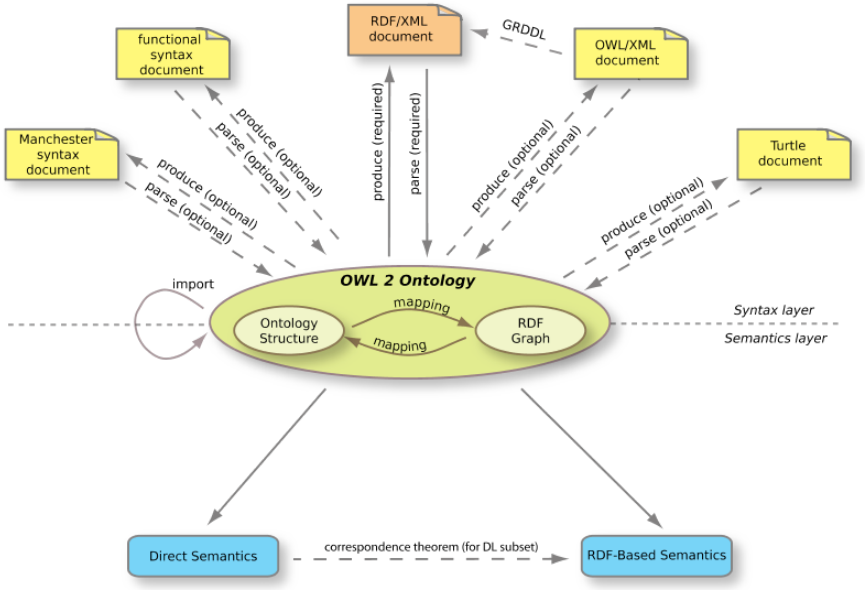

Fig. 1. General structure of OWL 2.

computational complexity. Our focus in Section 3 is on the DLs underlying the three profiles (or fragments) of OWL 2 that were identified by the OWL 2 working group to ensure tractability of reasoning at the expense of the expressive power. The Euler diagram in Fig. 2 gives a general overview of the DLs considered in this chapter (RDFS denotes the RDFS fragment of OWL 2 DL under the Direct Semantics).

The second aim of the chapter is to explain, in Section 4, the semantical foundations for first-order and datalog rewritings of conjunctive queries over knowledge bases given in the OWL 2 profiles. Query rewriting is a fundamental technique underlying ontology-based data access: it reduces answering queries over knowledge bases to answering first-order or datalog queries over plain data, which can be done using conventional database management systems or, respectively, datalog engines. For more expressive languages, ontology-based data access will be discussed in Chapter 6.

Finally, in Section 5, we give an overview of the architecture of the ontology-based data access system *Ontop*[4].

2 Description Logics

Description Logic is an area of knowledge representation and reasoning in Artificial Intelligence and the Semantic Web that studies logic-based formalisms

[4] `ontop.inf.unibz.it`

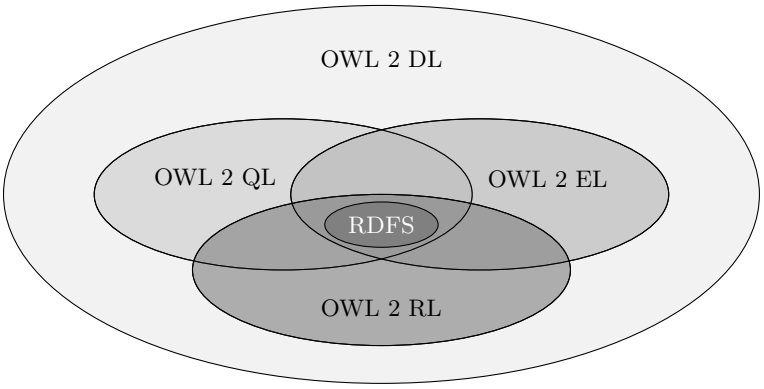

Fig. 2. Relationships between the DL fragments of OWL 2.

whose languages operate with *concepts* to represent classes of individuals in an application domain, and *roles* to represent binary relations between the individuals. Each concrete formalism, called a *description logic* (DL, for short), is characterised by its set of constructs that can be used to build complex concepts and roles from primitive ones.

The zoo of DLs is very big. We begin by defining one particular representative, which goes in the zoo under the moniker of $\mathcal{ALCHI}$. The example below shows a simple knowledge base (or an ontology), which is given in the OWL functional-style syntax (FSS) and the more terse syntax of $\mathcal{ALCHI}$. The reader is invited to decipher the meaning of the knowledge base before consulting the formal definitions.

Example 1. The following three statements are written in the FSS:

$$\begin{aligned}
&\mathsf{SubClassOf}(\mathsf{ObjectIntersectionOf}(\mathsf{Person}, && (1)\\
&\qquad \mathsf{ObjectSomeValuesFrom}(\mathsf{takesCourse},\ \mathsf{Course})),\ \mathsf{Student}),\\
&\mathsf{SubObjectPropertyOf}(\mathsf{mastersDegreeFrom},\ \mathsf{degreeFrom}), && (2)\\
&\mathsf{SubClassOf}(\mathsf{ObjectSomeValuesFrom}(&& (3)\\
&\qquad \mathsf{ObjectInverseOf}(\mathsf{takesCourse}),\ \mathsf{owl{:}Thing}),\ \mathsf{Course}),\\
&\mathsf{ClassAssertion}(\mathsf{Student},\ \mathsf{john}), && (4)\\
&\mathsf{ObjectPropertyAssertion}(\mathsf{takesCourse},\ \mathsf{john},\ \mathsf{sw}). && (5)
\end{aligned}$$

Using the syntax of $\mathcal{ALCHI}$, the same statements can be expressed as follows:

$$\begin{aligned}
&\mathsf{Person} \sqcap \exists \mathsf{takesCourse}.\mathsf{Course} \sqsubseteq \mathsf{Student}, && (1')\\
&\mathsf{mastersDegreeFrom} \sqsubseteq \mathsf{degreeFrom}, && (2')\\
&\mathsf{takesCourse}^{-}.\top \sqsubseteq \mathsf{Course}, && (3')\\
&\mathsf{Student}(\mathsf{john}), && (4')\\
&\mathsf{takesCourse}(\mathsf{john},\ \mathsf{sw}). && (5')
\end{aligned}$$

2.1 Syntax

The alphabet of $\mathcal{ALCHI}$ contains three pairwise disjoint and countably infinite sets: *concept names* $A_1, A_2, \ldots$, *role names* $P_1, P_2, \ldots$ and *individual names* $a_1, a_2, \ldots$. An $\mathcal{ALCHI}$ *role*, R, is either a role name P_i or its *inverse* P_i^-. $\mathcal{ALCHI}$ *concepts*, C, are constructed from two special primitive concepts, $\top$ ('top') and $\bot$ ('bottom'), concept names and roles using the following grammar:

$$C \quad ::= \quad A_i \quad | \quad \top \quad | \quad \bot \quad | \quad \neg C \quad | \quad C_1 \sqcap C_2 \quad | \quad C_1 \sqcup C_2 \quad | \quad \exists R.C \quad | \quad \forall R.C.$$

An $\mathcal{ALCHI}$ *terminological box* (or TBox), $\mathcal{T}$, is a finite set of *concept* and *role inclusion axioms* of the form

$$C_1 \sqsubseteq C_2 \qquad \text{and} \qquad R_1 \sqsubseteq R_2,$$

where C_1, C_2 are concepts and R_1, R_2 roles. An $\mathcal{ALCHI}$ *assertion box* (or ABox), $\mathcal{A}$, is a finite set of *concept* and *role assertions* of the form

$$C(a) \qquad \text{and} \qquad R(a, b),$$

where C is a concept, R a role and a, b are individual names. Given an ABox $\mathcal{A}$, we denote by $\mathsf{ind}(\mathcal{A})$ the set of individual names that occur in $\mathcal{A}$. Taken together, $\mathcal{T}$ and $\mathcal{A}$ comprise an $\mathcal{ALCHI}$ *knowledge base* (or KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$.

Example 1 shows that, in many respects, the DL syntax is simply a less verbose form of the FSS. Thus, $\sqsubseteq$ stands for SubClassOf and SubObjectPropertyOf, and P^- for $\mathsf{ObjectInverseOf}(P)$; the correspondences for concept constructs are listed below:

DL	FSS
$\bot$	owl:Nothing
$\top$	owl:Thing
$\neg C$	$\mathsf{ObjectComplementOf}(C)$
$C_1 \sqcap C_2$	$\mathsf{ObjectIntersectionOf}(C_1, C_2)$
$C_1 \sqcup C_2$	$\mathsf{ObjectUnionOf}(C_1, C_2)$
$\exists R.C$	$\mathsf{ObjectSomeValuesFrom}(R, C)$
$\forall R.C$	$\mathsf{ObjectAllValuesFrom}(R, C)$

In Section 2.4, we shall see how most of the OWL constructs can be mapped into the DL syntax. But before that, we need to define the meaning of DL constructs.

2.2 Semantics

As well as all DLs, $\mathcal{ALCHI}$ is equipped with a Tarski-style semantics defined in terms of interpretations (which are a simplified form of interpretations in the Direct Semantics of OWL). An *interpretation* $\mathcal{I}$ is a pair $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ that consists of a non-empty *domain of interpretation* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$. The latter assigns

– an element $a_i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ to each individual name a_i;
– a subset $A_i^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ to each concept name A_i;
– a binary relation $P_i^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to each role name P_i.

(In $\mathcal{ALCHI}$, distinct individuals are usually assumed to be interpreted by distinct domain elements—this is called the *unique name assumption*, or UNA. In this chapter we follow the convention and assume the UNA for all of our DLs, which cannot distinguish between models with and without UNA. Note, however, that OWL 2 and its three profiles *do not* adopt the UNA and use constructs like SameIndividual and DifferentIndividuals instead.) We extend inductively the interpretation function $\cdot^{\mathcal{I}}$ to complex roles and concepts by taking

$$\begin{aligned}
(P^-)^{\mathcal{I}} &= \{\, (v,u) \mid (u,v) \in P^{\mathcal{I}} \,\},\\
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}},\\
\bot^{\mathcal{I}} &= \emptyset,\\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}},\\
(C_1 \sqcap C_2)^{\mathcal{I}} &= C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}},\\
(C_1 \sqcup C_2)^{\mathcal{I}} &= C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}},\\
(\exists R.C)^{\mathcal{I}} &= \big\{\, u \mid \text{ there is } v \in C^{\mathcal{I}} \text{ such that } (u,v) \in R^{\mathcal{I}} \,\big\},\\
(\forall R.C)^{\mathcal{I}} &= \big\{\, u \mid v \in C^{\mathcal{I}}, \text{ for all } v \text{ with } (u,v) \in R^{\mathcal{I}} \,\big\}.
\end{aligned}$$

Having fixed the interpretation of individual names, concepts and roles, we now define the *satisfaction relation* $\models$ for inclusions and assertions:

$$\begin{aligned}
\mathcal{I} \models C_1 \sqsubseteq C_2 \quad &\text{if and only if} \quad C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}},\\
\mathcal{I} \models R_1 \sqsubseteq R_2 \quad &\text{if and only if} \quad R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}},\\
\mathcal{I} \models C(a) \quad &\text{if and only if} \quad a^{\mathcal{I}} \in C^{\mathcal{I}},\\
\mathcal{I} \models R(a,b) \quad &\text{if and only if} \quad (a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}.
\end{aligned}$$

We say that an interpretation $\mathcal{I}$ is a *model* of a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if it satisfies all concept and roles inclusions of $\mathcal{T}$ and all concept and role assertions of $\mathcal{A}$. In this case we write $\mathcal{I} \models \mathcal{K}$ (and also $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$).

It is to be remembered that (unlike databases) the choice of domains and interpretation functions is not fixed in the DL semantics, so that every knowledge base can have many models. This reflects the *open world assumption*, or OWA, adopted in DL (and OWL), according to which no single agent can possess complete knowledge. Thus, we have to consider all possible assignments of truth-values to assertions—as long as they do not contradict the given knowledge base. (Databases adopt the *closed world assumption*, or CWA, that defines everything unknown as false.)

Example 2. Consider the following knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$:

$$\begin{aligned} \mathcal{T} &= \{\ \mathsf{GraduateStudent} \sqsubseteq \mathsf{Student}, \\ &\qquad \mathsf{GraduateStudent} \sqsubseteq \exists \mathsf{takesCourse}.\mathsf{GraduateCourse}\ \}, \\ \mathcal{A} &= \{\ \mathsf{GraduateStudent}(\mathsf{john})\ \}. \end{aligned}$$

Denote by $\mathcal{I}_1$ an interpretation with domain $\Delta^{\mathcal{I}_1} = \{john, sw\}$ such that

$$\begin{aligned} \mathsf{john}^{\mathcal{I}_1} &= john, \\ \mathsf{GraduateStudent}^{\mathcal{I}_1} &= \{john\}, & \mathsf{Student}^{\mathcal{I}_1} &= \{john\}, \\ \mathsf{GraduateCourse}^{\mathcal{I}_1} &= \{sw\}, & \mathsf{takesCourse}^{\mathcal{I}_1} &= \{(john, sw)\}. \end{aligned}$$

The reader can readily check that $\mathcal{I}_1$ is a model of $\mathcal{K}$; that is, the 'world' described by $\mathcal{I}_1$ satisfies the knowledge and data given in $\mathcal{K}$. Now, take another interpretation $\mathcal{I}_2$ with domain $\Delta^{\mathcal{I}_2} = \{a\}$ in which

$$\begin{aligned} \mathsf{john}^{\mathcal{I}_2} &= a, \\ \mathsf{GraduateStudent}^{\mathcal{I}_2} &= \{a\}, & \mathsf{Student}^{\mathcal{I}_2} &= \{a\}, \\ \mathsf{GraduateCourse}^{\mathcal{I}_2} &= \{a\}, & \mathsf{takesCourse}^{\mathcal{I}_2} &= \{(a, a)\}. \end{aligned}$$

This interpretation does not make much sense from the modelling point of view (because a takes course a in the world described by $\mathcal{I}_2$). Nevertheless, $\mathcal{I}_2$ satisfies all of the inclusions in $\mathcal{T}$ and assertions in $\mathcal{A}$, and so is a model of $\mathcal{K}$. Yet another interpretation, $\mathcal{I}_3$, with domain $\Delta^{\mathcal{I}_3} = \{john\}$ and

$$\begin{aligned} \mathsf{john}^{\mathcal{I}_3} &= john, \\ \mathsf{GraduateStudent}^{\mathcal{I}_3} &= \{john\}, & \mathsf{Student}^{\mathcal{I}_3} &= \{john\}, \\ \mathsf{GraduateCourse}^{\mathcal{I}_3} &= \emptyset, & \mathsf{takesCourse}^{\mathcal{I}_3} &= \emptyset \end{aligned}$$

satisfies the assertions in $\mathcal{A}$ and the first concept inclusion in $\mathcal{T}$ but fails to satisfy the second concept inclusion, and so is not a model of $\mathcal{K}$.

Intuitively, everything that takes place in *each and every* model of a knowledge base is a logical consequence of the KB, not necessarily explicitly presented in it. Finding logical consequences is usually referred to as *reasoning*.

2.3 Reasoning Problems

A very basic reasoning problem is to decide whether a given knowledge base is *consistent* in the sense that it does not imply mutually contradicting statements. Formally, we can call a knowledge base $\mathcal{K}$ *satisfiable* (or *consistent*) if there exists at least one model of $\mathcal{K}$ (which is obviously enough to guarantee that $\mathcal{K}$ contains no contradictions).

Example 3. Let $\mathcal{T}$ be a TBox containing the following concept inclusions:

$$\mathsf{UndergraduateStudent} \sqsubseteq \forall \mathsf{takesCourse}.\mathsf{UndergraduateCourse},$$
$$\mathsf{UndergraduateCourse} \sqcap \mathsf{GraduateCourse} \sqsubseteq \bot,$$

and $\mathcal{A}$ an ABox with the following assertions:

$$\mathsf{UndergraduateStudent}(\mathsf{john}),$$
$$\mathsf{takesCourse}(\mathsf{john}, \mathsf{sw}),$$
$$\mathsf{GraduateCourse}(\mathsf{sw}).$$

If we assume that $(\mathcal{T}, \mathcal{A})$ has a model, then the undergraduate student John in it will only be able to take undergraduate courses that are not graduate courses such as SW. However, according to the ABox, John takes SW, which is a contradiction. Thus, $(\mathcal{T}, \mathcal{A})$ is inconsistent.

Another important reasoning problem is entailment. We say that a concept inclusion $C_1 \sqsubseteq C_2$ is *entailed by* a knowledge base $\mathcal{K}$ and write $\mathcal{K} \models C_1 \sqsubseteq C_2$ if $\mathcal{I} \models C_1 \sqsubseteq C_2$ for all models $\mathcal{I}$ of $\mathcal{K}$ (entailment for role inclusions and concept and role assertions is defined similarly).

Example 4. Consider a TBox $\mathcal{T}$ with the following two concept inclusions:

$$\forall \mathsf{takesCourse}.\mathsf{UndergraduateCourse} \sqsubseteq \mathsf{UndergraduateStudent},$$
$$\mathsf{FirstYearStudent} \sqsubseteq \exists \mathsf{takesCourse}.\mathsf{UndergraduateCourse}.$$

In the model $\mathcal{I}_1$ of $\mathcal{T}$ given below, $\mathsf{FirstYearStudent} \sqsubseteq \mathsf{UndergraduateStudent}$ holds true. However, $\mathcal{T}$ *does not entail* this concept inclusion because there is another model, $\mathcal{I}_2$, where it is not satisfied:

$$\begin{aligned}
\Delta^{\mathcal{I}_1} &= \{j, s\}, & \Delta^{\mathcal{I}_2} &= \{j, s, \ell\},\\
\mathsf{takesCourse}^{\mathcal{I}_1} &= \{(j, s)\}, & \mathsf{takesCourse}^{\mathcal{I}_2} &= \{(j, s), (j, \ell)\},\\
\mathsf{FirstYearStudent}^{\mathcal{I}_1} &= \{j\}, & \mathsf{FirstYearStudent}^{\mathcal{I}_2} &= \{j\},\\
\mathsf{UndergraduateStudent}^{\mathcal{I}_1} &= \{j\}, & \mathsf{UndergraduateStudent}^{\mathcal{I}_2} &= \emptyset,\\
\mathsf{UndergraduateCourse}^{\mathcal{I}_1} &= \{s\}, & \mathsf{UndergraduateCourse}^{\mathcal{I}_2} &= \{s\}.
\end{aligned}$$

Intuitively, in $\mathcal{I}_1$, the individual j is a first-year student who takes only one undergraduate course, and so must be an undergraduate student (to satisfy the first concept inclusion). In contrast, in $\mathcal{I}_2$, the individual j also takes another course, ℓ, which is not an undergraduate course, and therefore, j does not have to be an undergraduate student (still satisfying the first concept inclusion); see Fig. 3.

If $C_1 \sqsubseteq C_2$ is entailed by $\mathcal{K}$, then we also say that C_2 *subsumes* C_1 with respect to $\mathcal{K}$ (or that C_1 is *subsumed* by C_2 with respect to $\mathcal{K}$). We note in passing that if $\mathcal{K}$ is inconsistent then any concept is subsumed by any other

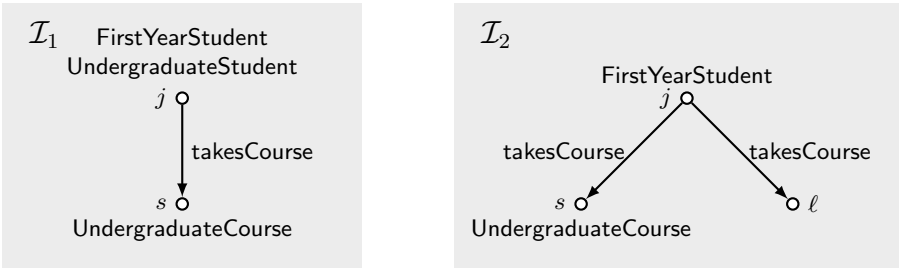

Fig. 3. Interpretations for $\mathcal{T}$ in Example 4.

concept: indeed, an arbitrary inclusion $C_1 \sqsubseteq C_2$ is trivially true in every model of $\mathcal{K}$ simply because $\mathcal{K}$ has no models. The following proposition, the proof of which is left to the reader as a simple exercise, shows that concept subsumption is in fact reducible to consistency.

Proposition 5. $(\mathcal{T}, \mathcal{A}) \models C_1 \sqsubseteq C_2$ *if and only if* $(\mathcal{T}, \mathcal{A} \cup \{C_1(a), \neg C_2(a)\})$ *is not satisfiable, for a fresh individual name* a *(not occurring in* $\mathcal{A}$*).*

(Note as a warning that not every DL allows negation and complex concepts in the ABoxes, in which case the reduction may be not so trivial.)

If $C(a)$ is entailed by $\mathcal{K}$ (that is, $C(a)$ holds in every model of $\mathcal{K}$), then we also say that a is an *instance* of C with respect to $\mathcal{K}$. The problem of checking whether a is an instance of a given C with respect to $\mathcal{K}$ is called *instance checking*. This problem is also reducible to knowledge base consistency (provided that complex concepts are allowed in the ABoxes):

Proposition 6. $(\mathcal{T}, \mathcal{A}) \models C(a)$ *if and only if* $(\mathcal{T}, \mathcal{A} \cup \{\neg C(a)\})$ *is not satisfiable.*

A more general reasoning task is answering conjunctive queries over knowledge bases. A *conjunctive query* (CQ for short) $\boldsymbol{q}(\boldsymbol{x})$ is an expression of the form $\exists \boldsymbol{y}\, \varphi(\boldsymbol{x}, \boldsymbol{y})$, where $\varphi(\boldsymbol{x}, \boldsymbol{y})$ is a conjunction of atoms such as $A(z)$ and $P(z_1, z_2)$, for a concept name A, role name P and terms z, z_1 and z_2, which are individual names or variables from $\boldsymbol{x}$ and $\boldsymbol{y}$. The variables x_i in $\boldsymbol{x} = (x_1, \dots, x_n)$ are called *answer variables* and the variables in $\boldsymbol{y}$ *existentially quantified variables*. Given a tuple $\boldsymbol{a} = (a_1, \dots, a_n)$ of individual names from $\mathcal{A}$, we denote by $\boldsymbol{q}(\boldsymbol{a})$ the result of replacing each answer variable x_i in $\exists \boldsymbol{y}\, \varphi(\boldsymbol{x}, \boldsymbol{y})$ with the respective a_i from $\boldsymbol{a}$. A tuple $\boldsymbol{a}$ of individual names from $\mathcal{A}$ is a *certain answer* to $\boldsymbol{q}(\boldsymbol{x})$ over $(\mathcal{T}, \mathcal{A})$ if, for any model $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A})$, the sentence $\boldsymbol{q}(\boldsymbol{a})$ is true in $\mathcal{I}$ ($\mathcal{I} \models \boldsymbol{q}(\boldsymbol{a})$, in symbols). We write $(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q}(\boldsymbol{a})$ to indicate that $\boldsymbol{a}$ is a certain answer to $\boldsymbol{q}(\boldsymbol{x})$ over $(\mathcal{T}, \mathcal{A})$. A CQ $\boldsymbol{q}$ without answer variables is called *Boolean*, in which case a certain answer to $\boldsymbol{q}$ over $(\mathcal{T}, \mathcal{A})$ is 'yes' if $(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q}$ and 'no' otherwise. The problem of answering Boolean CQs is known as *CQ entailment*.

Example 7. (Andrea's example (Schaerf, 1993)) Suppose a TBox $\mathcal{T}$ contains the inclusions

$$\top \sqsubseteq \mathsf{Male} \sqcup \mathsf{Female}, \qquad \mathsf{Male} \sqcap \mathsf{Female} \sqsubseteq \bot$$

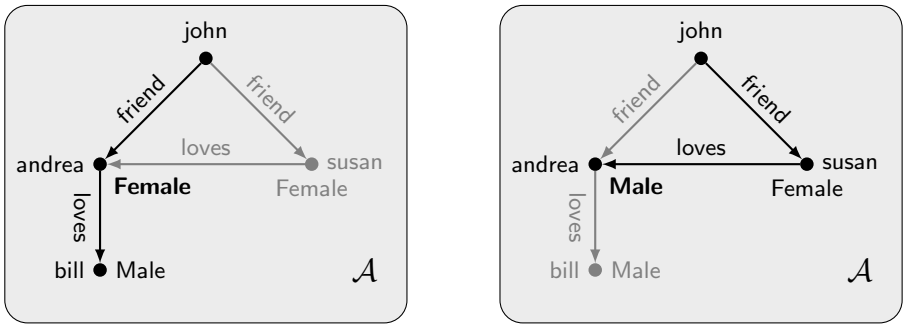

Fig. 4. Two representative models of $(\mathcal{T}, \mathcal{A})$ in Example 7.

and an ABox $\mathcal{A}$ contains the assertions

$$\begin{array}{ll} \mathsf{friend}(\mathsf{john}, \mathsf{susan}), & \mathsf{friend}(\mathsf{john}, \mathsf{andrea}), \\ \mathsf{loves}(\mathsf{susan}, \mathsf{andrea}), & \mathsf{loves}(\mathsf{andrea}, \mathsf{bill}), \\ \mathsf{Female}(\mathsf{susan}), & \mathsf{Male}(\mathsf{bill}). \end{array}$$

Consider the CQ

$$\boldsymbol{q}(x) \;=\; \exists y, z \, \big(\mathsf{friend}(x, y) \land \mathsf{Female}(y) \land \mathsf{loves}(y, z) \land \mathsf{Male}(z)\big),$$

which asks to find every individual (in the ABox) with a female friend who is in love with a male. Note that the same CQ can be expressed in the query language SPARQL[5] as follows:

```
SELECT ?x
WHERE {
    ?x :friend ?y.
    ?y a :Female.
    ?y :loves ?z.
    ?z a :Male.
}
```

(Here a is an abbreviation of rdf:type and can be read as 'is a.') We invite the reader to check that the only certain answer to $\boldsymbol{q}(x)$ over $(\mathcal{T}, \mathcal{A})$ is $x \mapsto \mathsf{john}$. (Hint: in every model $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A})$, either $\mathsf{andrea}^{\mathcal{I}} \in \mathsf{Female}^{\mathcal{I}}$ or $\mathsf{andrea}^{\mathcal{I}} \in \mathsf{Male}^{\mathcal{I}}$; see Fig. 4.)

Observe that this particular CQ $\boldsymbol{q}(x)$ can also be represented as an *instance query* $C(x)$, where C is the complex concept

$$\exists \mathsf{friend}.(\mathsf{Female} \sqcap \exists \mathsf{loves}.\mathsf{Male}).$$

[5] www.w3.org/TR/sparql11-query

It is readily seen that, for any individual name a, we have $(\mathcal{T}, \mathcal{A}) \models C(a)$ if and only if $(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q}(a)$. However, in general, CQs are not necessarily tree-shaped, can contain more than one answer variable, and so are more expressive than instance queries.

2.4 From OWL to DL

The language of OWL 2 contains more constructs than any of the DLs, with many of these constructs being just shortcuts for certain DL expressions. For example,

$$\mathsf{ObjectPropertyDomain}(\mathsf{takesCourse}, \mathsf{Student})$$

can be represented as the concept inclusion

$$\exists \mathsf{takesCourse}.\top \sqsubseteq \mathsf{Student}.$$

Similarly,

$$\mathsf{ObjectPropertyRange}(\mathsf{takesCourse}, \mathsf{Course})$$

can be represented as

$$\exists \mathsf{takesCourse}^{-}.\top \sqsubseteq \mathsf{Course}.$$

Note that the latter concept inclusion can only be written in the DLs with *inverse roles* (such as the RL or QL profiles of OWL 2 to be discussed below). If, however, inverse roles are not available in a DL (for instance, $\mathcal{ALC}$), then one can use a universal restriction:

$$\top \sqsubseteq \forall \mathsf{takesCourse}.\mathsf{Course}.$$

One can verify that the two concept inclusions are equivalent in the sense that they are satisfied by precisely the same interpretations. We have collected standard equivalencies of this sort in the following proposition:

Proposition 8. *The following pairs of (sets of) concept inclusions have the same models*:

$$C_1 \sqsubseteq \forall P.C_2 \quad \textit{and} \quad \exists P^{-}.C_1 \sqsubseteq C_2, \tag{6}$$

$$C_1 \sqcup C_2 \sqsubseteq C \quad \textit{and} \quad \{\, C_i \sqsubseteq C \mid i = 1, 2\}, \tag{7}$$

$$C \sqsubseteq C_1 \sqcap C_2 \quad \textit{and} \quad \{\, C \sqsubseteq C_i \mid i = 1, 2\}, \tag{8}$$

$$C_1 \sqsubseteq \neg C_2 \quad \textit{and} \quad C_1 \sqcap C_2 \sqsubseteq \bot. \tag{9}$$

OWL 2 also has a shortcut for equivalent classes and properties: EquivalentClasses, EquivalentObjectProperties and EquivalentDataProperties. These can be represented using $\sqsubseteq$: for example,

$$\mathsf{EquivalentClasses}(C_1, C_2, \ldots, C_n)$$

can be written in the DL parlance as n concept inclusions

$$C_1 \sqsubseteq C_2, \quad C_2 \sqsubseteq C_3, \quad \ldots, \quad C_{n-1} \sqsubseteq C_n, \quad C_n \sqsubseteq C_1.$$

Another way is to use a common DL abbreviation $\equiv$, which is defined by taking $A \equiv B$ if and only if $A \sqsubseteq B$ and $B \sqsubseteq A$.

Equivalence (9) provides two alternative ways of expressing *disjointness* of concepts C_1 and C_2 in DL. OWL 2 offers a shortcut for pairwise disjointness of n classes:

$$\mathsf{DisjointClasses}(C_1, C_2, \ldots, C_n),$$

which can be represented in DLs as $n(n-1)/2$ concept inclusions

$$C_i \sqcap C_j \sqsubseteq \bot, \quad \text{for all } i, j \text{ with } 1 \leq i < j \leq n.$$

Yet another shortcut in OWL 2 allows one to say that class C is the disjoint union of $C_1, \ldots, C_n$:

$$\mathsf{DisjointUnion}(C, C_1, C_2, \ldots, C_n),$$

which combines pairwise disjointness of $C_1, \ldots, C_n$ (see above) with

$$C \equiv C_1 \sqcup \cdots \sqcup C_n.$$

Object properties can be declared symmetric in OWL 2 by using axioms of the form $\mathsf{SymmetricObjectProperty}(P)$. The same effect can be achieved in DLs with the help of the role inclusion

$$P^- \sqsubseteq P.$$

More expressive description logics contain additional constructs such as number restrictions, $\exists R.\mathsf{Self}$, transitive roles, role chains, etc. In particular, the DL subset of OWL 2, known as OWL 2 DL, is based on the description logic called $\mathcal{SROIQ}$ (Horrocks et al., 2006).

2.5 Complexity of Reasoning

Having formulated the reasoning problems, we are facing the following fundamental questions:

- Are these problems *decidable* in the sense that there exist algorithms which always halt and return correct answers?
- How *complex* are such algorithms in terms of the time or memory space they require?
- Are these algorithms (reasonably) *efficient* in real-world applications?

The next theorem, which is a compilation of various results from (Tobies, 2001; Eiter et al., 2009; Horrocks et al., 2006),[6] gives answers to the first two questions:

Theorem 9. (*i*) *The satisfiability problem is* ExpTime*-complete for* $\mathcal{ALCHI}$ *KBs and* N2ExpTime*-complete for* $\mathcal{SROIQ}$ *KBs.*

(*ii*) *Concept and role subsumption and instance checking are* ExpTime*- and* coN2ExpTime*-complete for, respectively,* $\mathcal{ALCHI}$ *and* $\mathcal{SROIQ}$ *KBs.*

(*iii*) *CQ entailment over* $\mathcal{ALCHI}$ *KBs is* 2ExpTime*-complete.*

[6] See also the DL Complexity Navigator at `www.cs.man.ac.uk/~ezolin/dl`.

Note that full OWL 2 under the RDF-based semantics is undecidable (Motik, 2007), while OWL 2 DL under the direct (model-theoretic) semantics is decidable. There is, however, a price to pay for the additional expressive power of $\mathcal{SROIQ}$ underlying OWL 2 DL: satisfiability is harder than in $\mathcal{ALCHI}$ and CQ entailment is not even known to be decidable.

ExpTime-completeness of satisfiability in $\mathcal{ALCHI}$ means, in particular, two things: first, there exists a satisfiability-checking algorithm that runs in at most exponential time in the size of the input knowledge base and, second, *no* algorithm can check satisfiability of *any* given knowledge base in polynomial time. This complexity-theoretic result does not suggest that reasoning algorithms can be efficient in practice. Fortunately, practical reasoners for OWL 2 DL have been implemented: FaCT++ (Tsarkov and Horrocks, 2006), HermiT (Horrocks et al., 2012), Pellet (Sirin et al., 2007), Konclude (Steigmiller et al., 2014). Their efficiency in typical real-world applications (rather than worst-case scenarios) relies upon sophisticated optimisation techniques and the empirical fact that ontologies designed by humans for such applications are often simple enough for basic reasoning techniques. On the other hand, answering instance queries and, more generally, conjunctive queries over knowledge bases in expressive languages has not become practical so far, especially for large data sets.

3 Description Logics for the OWL Profiles

The current W3C recommendation OWL 2 of the Web Ontology Language identifies three *profiles* (fragments or sub-languages) specifically designed to ensure efficiency (tractability) of reasoning at the expense of expressive power. In this section, we introduce these profiles in the form of DLs (sacrificing some features for the clarity of presentation).

We begin with an observation that in order to ensure tractability of reasoning (that is, the existence of a *deterministic polynomial-time* algorithm for checking consistency) one has to avoid using $\sqcup$ on the right-hand side of concept inclusions. Intuitively, this construct requires (non-deterministic) reasoning about possible cases such as in Example 7 (case 1: Andrea is a female; case 2: Andrea is a male), which can be NP-hard; for an introduction on the computational complexity, consult classical (Garey and Johnson, 1979) or more recent (Kozen, 2006; Arora and Barak, 2009).

We illustrate non-deterministic reasoning by encoding the NP-complete *graph 3-colourability* problem: given an (undirected) graph $G = (V, E)$, decide whether each of its vertices can be painted in one of the given three colours in such a way that no pair of adjacent vertices has the same colour. We represent the input graph G by means of an ABox $\mathcal{A}_G$ comprising assertions

$$\mathsf{edge}(v_1, v_2), \quad \text{for each } \{v_1, v_2\} \in E.$$

Consider a TBox $\mathcal{T}$ containing the following concept inclusions:

$$\begin{aligned} &\top \sqsubseteq C_1 \sqcup C_2 \sqcup C_3, && \\ &C_i \sqcap C_j \sqsubseteq \bot, && 1 \leq i < j \leq 3, \\ &C_i \sqcap \exists \mathsf{edge}.C_i \sqsubseteq \bot, && 1 \leq i \leq 3, \end{aligned}$$

where C_1, C_2 and C_3 are concept names representing the given three colours. It is not hard to see that each model of $(\mathcal{T}, \mathcal{A}_G)$ gives rise to a 3-colouring of G and, conversely, each 3-colouring of G can be encoded in a model of $(\mathcal{T}, \mathcal{A}_G)$. In other words, $(\mathcal{T}, \mathcal{A}_G)$ is satisfiable if and only if G is 3-colourable. This means that the satisfiability problem for knowledge bases in any DL able to express this TBox is NP-hard (that is, not tractable).

3.1 OWL 2 RL

The OWL 2 RL profile[7] is aimed at applications requiring scalable reasoning that can be done by rule-based implementations (such as datalog engines). Its design was inspired by the so-called Description Logic Programs (Grosof et al., 2003) and pD* (ter Horst, 2005). OWL 2 RL is supported by Oracle Database 11g, OWLIM, BaseVISor, ELLY, Jena and RDFox (for details and references, see `www.w3.org/2001/sw/wiki/OWL/Implementations`).

A key feature of OWL 2 RL is that it does not allow existential quantifiers on the right-hand side of concept inclusions. Therefore, when reasoning with an OWL 2 RL knowledge base, we do not have to deal with individuals that are not explicitly present in the knowledge base ABox.

In this section, we consider a somewhat simplified version of OWL 2 RL, which will be called *RL*. Concept and role inclusions in *RL* take the form

$$B \sqsubseteq A, \qquad R_1 \sqsubseteq R_2 \quad \text{and} \quad B \sqsubseteq \bot,$$

where R_1 and R_2 are roles (role names or their inverses), A is a concept name and B a concept defined by the following grammar:

$$B \;::=\; A \;\mid\; \exists R.\top \;\mid\; \exists R.B \;\mid\; B_1 \sqcap B_2.$$

Observe that there is no $\sqcup$ in the syntax, and the existential quantifiers occur only on the left-hand side of concept inclusions. On the other hand, by Proposition 8, universal restrictions, intersection and complement on the right-hand side of concept inclusions and union on the left-hand side of concept inclusions are simply syntactic sugar.

An *RL knowledge base* $(\mathcal{T}, \mathcal{A})$ comprises a finite set $\mathcal{T}$ of inclusions introduced above and a *simple* ABox $\mathcal{A}$, which contains only assertions of the form $A(a)$ and $P(a, b)$, for a concept name A and a role name P.

To understand reasoning in *RL* (and the other two profiles of OWL 2), it is useful to represent its concept and role inclusions as first-order sentences.

[7] `www.w3.org/TR/owl2-profiles/#OWL_2_RL`

Example 10. The *RL* concept inclusions

$$\exists \mathsf{takesCourse}.\mathsf{UndergraduateCourse} \sqsubseteq \mathsf{UndergraduateStudent},$$
$$\mathsf{UndergraduateStudent} \sqsubseteq \mathsf{Student}$$

are equivalent (have the same models) as the first-order sentences

$$\forall x \forall y \big(\mathsf{takesCourse}(x, y) \land \mathsf{UndergraduateCourse}(y) \to \mathsf{UndergraduateStudent}(x)\big),$$
$$\forall x \big(\mathsf{UndergraduateStudent}(x) \to \mathsf{Student}(x)\big).$$

More formally, we define a *standard translation* ST of concepts and roles by induction on their structure. First, for any concept name A and any role name P, we set

$$ST_x(A) = A(x),$$
$$ST_{x,y}(P) = P(x, y),$$

where the subscript specifies the variables used as arguments of the predicates. After that, we extend the translation ST to complex roles and concepts by taking

$$ST_{x,y}(P^-) = P(y, x),$$
$$ST_x(\exists R.\top) = \exists y\, ST_{x,y}(R),$$
$$ST_x(\exists R.B) = \exists y \big(ST_{x,y}(R) \land ST_y(B)\big),$$
$$ST_x(B_1 \sqcap B_2) = ST_x(B_1) \land ST_x(B_2).$$

Finally, we translate concept and role inclusions into universally quantified implications:

$$(B \sqsubseteq A)^* = \forall x \big(ST_x(B) \to ST_x(A)\big),$$
$$(R_1 \sqsubseteq R_2)^* = \forall x \forall y \big(ST_{x,y}(R_1) \to ST_{x,y}(R_2)\big),$$
$$(B \sqsubseteq \bot)^* = \forall x \big(ST_x(B) \to \bot\big).$$

As $\exists R.\top$ and $\exists R.B$ can occur only on the left-hand side of concept inclusions, the translation $\cdot^*$ of any *RL* TBox contains only sentences of the form

$$\forall \boldsymbol{y} \big(\gamma_1(\boldsymbol{y}) \land \cdots \land \gamma_k(\boldsymbol{y}) \to \gamma_0(\boldsymbol{y})\big),$$
$$\forall \boldsymbol{y} \big(\gamma_1(\boldsymbol{y}) \land \cdots \land \gamma_k(\boldsymbol{y}) \to \bot\big),$$

where $\gamma_0(\boldsymbol{y}), \dots, \gamma_k(\boldsymbol{y})$ are unary or binary predicates with variables in $\boldsymbol{y}$. Such sentences are examples of *Horn clauses* (sets of sentences of the first kind are also called *datalog programs*; see e.g., Ceri et al. (1989)). A very important property of Horn clauses known from logic and databases is that everything we may want to know about a knowledge base, whose TBox axioms are Horn clauses, can be found in the canonical model (or chase), which is constructed by 'applying' the clauses to the ABox. We first illustrate the construction by a simple example.

Example 11. Consider the TBox $\mathcal{T}$ from Example 10 and the following ABox:

$$\mathcal{A} \;=\; \{\, \mathsf{takesCourse}(\mathsf{john}, \mathsf{sp1}),\ \mathsf{UndergraduateCourse}(\mathsf{sp1}) \,\}.$$

We begin the construction of the canonical model by representing the ABox as an interpretation $\mathcal{I}_0$ with the domain $\Delta^{\mathcal{I}_0} = \{\mathsf{john}, \mathsf{sp1}\}$ and the interpretation function given by

$$\begin{aligned} \mathsf{takesCourse}^{\mathcal{I}_0} &= \{(\mathsf{john}, \mathsf{sp1})\}, \\ \mathsf{UndergraduateCourse}^{\mathcal{I}_0} &= \{\mathsf{sp1}\}, \\ \mathsf{UndergraduateStudent}^{\mathcal{I}_0} &= \emptyset, \\ \mathsf{Student}^{\mathcal{I}_0} &= \emptyset. \end{aligned}$$

The interpretation $\mathcal{I}_0$ does not satisfy the first concept inclusion in $\mathcal{T}$ because john is related by takesCourse to sp1, which is an UndergraduateCourse, but john is not an instance of UndergraduateStudent. To repair this 'defect', we apply (as a rule) the first concept inclusion to $\mathcal{I}_0$ and obtain an interpretation $\mathcal{I}_1$ with the same domain, $\Delta^{\mathcal{I}_1} = \Delta^{\mathcal{I}_0}$, and the interpretation function $\cdot^{\mathcal{I}_1}$ that expands $\cdot^{\mathcal{I}_0}$ with

$$\mathsf{UndergraduateStudent}^{\mathcal{I}_1} = \{\mathsf{john}\}$$

(all other symbols have the same interpretation as in $\mathcal{I}_0$). Now, $\mathcal{I}_1$ satisfies the first concept inclusion of $\mathcal{T}$ but fails to satisfy the second one. We repair this defect by 'applying' the second concept inclusion to $\mathcal{I}_1$ and obtaining an interpretation $\mathcal{I}_2$ with the same domain and the interpretation function expanding $\cdot^{\mathcal{I}_1}$ with

$$\mathsf{Student}^{\mathcal{I}_2} = \{\mathsf{john}\}$$

(all other symbols keep their interpretation). It is readily seen that now $\mathcal{I}_2$ is a model of $(\mathcal{T}, \mathcal{A})$. Note that we constructed $\mathcal{I}_2$ by adding to the given ABox $\mathcal{A}$ only those assertions—UndergraduateStudent(john) and Student(john)—that were required by the TBox $\mathcal{T}$. That is why $\mathcal{I}_2$ is referred to as the *minimal model* or *canonical model* of $(\mathcal{T}, \mathcal{A})$.

The procedure used in the example above is known as *forward chaining*. Formally, it can be described as follows. First, a simple ABox can be regarded as an interpretation:

Definition 12. The *standard model* $\mathcal{I}_\mathcal{A}$ of a simple ABox $\mathcal{A}$ is defined by taking

$$\begin{aligned} \Delta^{\mathcal{I}_\mathcal{A}} &= \mathsf{ind}(\mathcal{A}), \\ a^{\mathcal{I}_\mathcal{A}} &= a, && \text{for } a \in \mathsf{ind}(\mathcal{A}), \\ A^{\mathcal{I}_\mathcal{A}} &= \{a \mid A(a) \in \mathcal{A}\}, && \text{for concept name } A, \\ P^{\mathcal{I}_\mathcal{A}} &= \{(a, b) \mid P(a, b) \in \mathcal{A}\}, && \text{for role name } P. \end{aligned}$$

Thus, the domain of $\mathcal{I}_\mathcal{A}$ is the set of individual names in $\mathcal{A}$, and each individual name is interpreted in $\mathcal{I}_\mathcal{A}$ by itself, which is often referred to as the *standard name assumption*.

Next, given an *RL* knowledge base $(\mathcal{T}, \mathcal{A})$, we construct a sequence of interpretations $\mathcal{I}_0, \mathcal{I}_1, \dots, \mathcal{I}_n$ by setting $\mathcal{I}_0 = \mathcal{I}_\mathcal{A}$ and then applying the following rules to each $\mathcal{I}_k$ to obtain $\mathcal{I}_{k+1}$:

(c) if $a \in B^{\mathcal{I}_k}$, $B \sqsubseteq A \in \mathcal{T}$ but $a \notin A^{\mathcal{I}_k}$, then we add a to $A^{\mathcal{I}_{k+1}}$;
(r) if $(a, b) \in R_1^{\mathcal{I}_k}$, $R_1 \sqsubseteq R_2 \in \mathcal{T}$ but $(a, b) \notin R_2^{\mathcal{I}_k}$, then we add (a, b) to $R_2^{\mathcal{I}_{k+1}}$;
(b) if $a \in B^{\mathcal{I}_k}$ and $B \sqsubseteq \bot \in \mathcal{T}$, then the process terminates.

Since the domains of the $\mathcal{I}_k$ are finite and all coincide with the set of individuals in $\mathcal{A}$, the process terminates after a finite number of steps either because neither **(c)** nor **(r)** is applicable or because **(b)** applies. In the former case the resulting interpretation satisfies all the inclusions in $\mathcal{T}$ and all the assertions in $\mathcal{A}$; it is called the *canonical model of* $(\mathcal{T}, A)$ and denoted by $\mathcal{C}_{\mathcal{T},\mathcal{A}}$. In the latter case, $(\mathcal{T}, \mathcal{A})$ is inconsistent.

Let $\mathcal{T}_+$ be the *positive* part of an *RL* TBox $\mathcal{T}$ that consists of all role inclusions in $\mathcal{T}$ and all concept inclusions of the form $B \sqsubseteq A$ in $\mathcal{T}$. By definition, $(\mathcal{T}_+, \mathcal{A})$ is consistent for any $\mathcal{A}$, and so its canonical model $\mathcal{C}_{\mathcal{T}_+,\mathcal{A}}$ is defined and can be used to check consistency of the full knowledge base (see, e.g., Calì et al. (2012)):

Theorem 13. *An RL knowledge base $(\mathcal{T}, \mathcal{A})$ is consistent if and only if we have $\mathcal{C}_{\mathcal{T}_+,\mathcal{A}} \models B \sqsubseteq \bot$, for all $B \sqsubseteq \bot$ in $\mathcal{T}$.*

Similar results hold for the other two profiles of OWL 2 (or indeed for any Horn DL), and therefore in the following we do not consider negative concept inclusions (although they are part of both OWL 2 EL and OWL 2 QL).

The canonical model $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ is *universal* in the sense that it can be homomorphically mapped into any other model of $(\mathcal{T}, \mathcal{A})$ (this notion formalises the minimality mentioned above).

Definition 14. A *homomorphism* h from an interpretation $\mathcal{I}_1$ to an interpretation $\mathcal{I}_2$ is a map from $\Delta^{\mathcal{I}_1}$ to $\Delta^{\mathcal{I}_2}$ such that

- $h(a^{\mathcal{I}_1}) = a^{\mathcal{I}_2}$, for each individual name a,
- $h(u) \in A^{\mathcal{I}_2}$, for any $u \in A^{\mathcal{I}_1}$ and any concept name A,
- $(h(u), h(v)) \in P^{\mathcal{I}_2}$, for any $(u, v) \in P^{\mathcal{I}_1}$ and any role name P.

The universality of the canonical models means, in particular, that checking subsumption and answering CQs over *RL* knowledge bases can be done by analysing their canonical models:

Theorem 15. *Let $(\mathcal{T}, \mathcal{A})$ be a consistent RL knowledge base. Then $\mathcal{C}_{\mathcal{T},\mathcal{A}} \models (\mathcal{T}, \mathcal{A})$. In addition, we have the following*:

(i) $(\mathcal{T}, \mathcal{A}) \models B \sqsubseteq A$ *if and only if* $\mathcal{C}_{\mathcal{T},\mathcal{A}} \models B \sqsubseteq A$;
(ii) $(\mathcal{T}, \mathcal{A}) \models R_1 \sqsubseteq R_2$ *if and only if* $\mathcal{C}_{\mathcal{T},\mathcal{A}} \models R_1 \sqsubseteq R_2$;
(iii) $(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q}(\boldsymbol{a})$ *if and only if* $\mathcal{C}_{\mathcal{T},\mathcal{A}} \models \boldsymbol{q}(\boldsymbol{a})$.

Observe that the forward chaining procedure only requires a polynomial number of steps to construct the canonical model (more precisely, the number of steps is bounded by $O(m^2 \times s)$, where m is the number of individual names in the ABox and s is the number of concept and role names). Therefore, knowledge base consistency, subsumption and instance checking are tractable in *RL* (the matching lower bound will be discussed in Section 4):

Theorem 16. *The problems of knowledge base consistency, concept and role subsumption and instance checking are* P*-complete in RL. The problem of CQ entailment in RL is* NP*-complete.*

Forward chaining can be regarded as a bottom-up saturation procedure. In datalog, however, more common is the top-down procedure, where a proof is constructed for a given statement; see e.g., (Ceri et al., 1989). We shall return to this topic in Section 4.3.

It is important to note that, since the clauses in the standard translation of *RL* inclusions contain only universally quantified variables, the canonical model of any *RL* knowledge base is *finite* (forward chaining does not add any new elements to the individuals in the ABox). We now turn to the profiles where this is not necessarily the case.

3.2 OWL 2 EL

The design of the OWL 2 EL profile[8] of OWL 2 (Baader et al., 2008, 2005) was based on the observation that biomedical ontologies such as SNOMED CT,[9] NCI[10] and GO[11] essentially use only conjunctions and existential quantifiers.

In this section, we consider a somewhat simplified version of OWL 2 EL, which will be called *EL*. Concept and role inclusions in *EL* look as follows:

$$C_1 \sqsubseteq C_2 \qquad \text{and} \qquad P_1 \sqsubseteq P_2,$$

where P_1 and P_2 are role names (role inverses are not allowed in *EL*) and C_1, C_2 are concepts defined by the following grammar:

$$C \quad ::= \quad A \quad | \quad \exists P.\top \quad | \quad \exists P.C \quad | \quad C_1 \sqcap C_2$$

(again, existential restrictions contain only role names). The following are typical concept inclusions from the SNOMED CT ontology:

$\mathsf{Pericardium} \sqsubseteq \mathsf{Tissue} \sqcap \mathsf{containedIn.Heart}$,

$\mathsf{Pericarditis} \sqsubseteq \mathsf{Inflammation} \sqcap \mathsf{hasLocation.Pericardium}$,

$\mathsf{Inflammation} \sqsubseteq \mathsf{Disease} \sqcap \mathsf{actsOn.Tissue}$,

$\mathsf{Disease} \sqcap \mathsf{hasLocation.containedIn.Heart} \sqsubseteq \mathsf{HeartDisease} \sqcap \mathsf{NeedsTreatment}$.

[8] www.w3.org/TR/owl2-profiles/#OWL_2_EL

[9] www.ihtsdo.org/snomed-ct

[10] www.obofoundry.org/cgi-bin/detail.cgi?id=ncithesaurus

[11] www.geneontology.org

ABoxes in *EL* are simple; see Section 3.1. The language we defined above is almost identical to $\mathcal{ELH}$: the only difference is that we do not allow $\top$ to occur outside the scope of existential restrictions. Thus, the *EL* concepts here are exactly those *RL* concepts from Section 3.1 that do not contain inverse roles. Note that *EL* extended with inverse roles is not tractable; moreover reasoning becomes as complex as in $\mathcal{ALCHI}$, that is, ExpTime-complete (Baader et al., 2008).

Since the existential restrictions can also occur on the right-hand side of concept inclusions, the result of translating a given *EL* TBox into first-order logic (see Section 3.1) is no longer a datalog program. Instead, it belongs to an extension of datalog called datalog$^{\pm}$ (Calì et al., 2012) or existential rules (Baget et al., 2011)[12]; in database theory, this language has been known since the early 1980s under the name of tuple-generating dependencies (Abiteboul et al., 1995). More precisely, *EL* TBoxes can be translated into sets of sentences of the form

$$\forall \boldsymbol{y} \big(\gamma_1(\boldsymbol{y}) \land \cdots \land \gamma_k(\boldsymbol{y}) \to \exists \boldsymbol{x}\, \gamma_0(\boldsymbol{x}, \boldsymbol{y})\big),$$

where $\gamma_1(\boldsymbol{y}), \dots, \gamma_k(\boldsymbol{y})$ contain only universally quantified variables $\boldsymbol{y}$ whereas $\gamma_0(\boldsymbol{x}, \boldsymbol{y})$ contains both universally quantified variables $\boldsymbol{y}$ and existentially quantified variables $\boldsymbol{x}$ (note that the standard translations of *EL* TBoxes are in fact more restricted than this general form, but it suffices for our explanations). Since these sentences are Horn clauses, we can again apply the forward chaining procedure (chase). However, in the case of *EL* TBoxes the chase has to 'invent' new domain elements for the existential quantifiers. Following the terminology of database theory, these fresh (previously not existing) domain elements will be called *labelled nulls*. In general, the forward chaining procedure may require infinitely many fresh labelled nulls resulting in a possibly infinite canonical model.

The forward chaining construction of the canonical model for an *EL* knowledge base $(\mathcal{T}, \mathcal{A})$ can be defined by taking the standard model $\mathcal{I}_\mathcal{A}$ of the ABox as $\mathcal{I}_0$ (see Definition 12) and applying inductively the following rules to obtain $\mathcal{I}_{k+1}$ from $\mathcal{I}_k$:

(c′) if $d \in C^{\mathcal{I}_k}$ and $C \sqsubseteq A \in \mathcal{T}$, then we add d to $A^{\mathcal{I}_{k+1}}$;
(r′) if $(d, d') \in P_1^{\mathcal{I}_k}$ and $P_1 \sqsubseteq P_2 \in \mathcal{T}$, then we add (d, d') to $P_2^{\mathcal{I}_{k+1}}$;
(e) if $d \in C^{\mathcal{I}_k}$ and $C \sqsubseteq \exists P.D \in \mathcal{T}$, where D is a concept name or $\top$, then we take a *fresh* labelled null, d', and add d' to $D^{\mathcal{I}_{k+1}}$ and (d, d') to $P^{\mathcal{I}_{k+1}}$

(here, we assume that only A, $\exists P.A$ and $\exists P.\top$ can occur on the right-hand side of concept inclusions—this restriction is inessential, as we shall see in Theorem 20). Note that rules **(c′)** and **(r′)** are similar to the rules from Section 3.1 except that they are applied without checking whether d and (d, d') are already present in the interpretation of the concept and role, respectively. This modification of the chase procedure is usually called the *oblivious chase* (Johnson and Klug, 1984); see also (Calì et al., 2013).

[12] See also Chapter 6 in this volume.

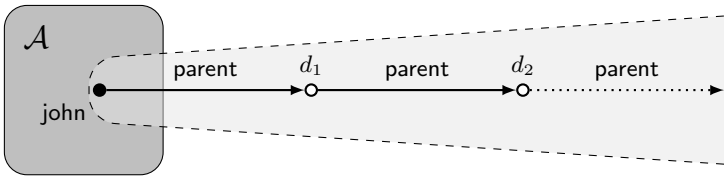

Fig. 5. The infinite canonical model for the KB from Example 17.

Example 17. Consider an *EL* TBox $\mathcal{T}$ with the following concept inclusion:

$$\mathsf{Person} \sqsubseteq \exists \mathsf{parent}.\mathsf{Person}.$$

Let us apply the forward chaining procedure to the ABox $\mathcal{A} = \{\mathsf{Person}(\mathsf{john})\}$. We begin by setting $\mathcal{I}_0 = \mathcal{I}_\mathcal{A}$:

$$\Delta^{\mathcal{I}_0} = \{\mathsf{john}\}, \quad \mathsf{Person}^{\mathcal{I}_0} = \{\mathsf{john}\} \quad \text{and} \quad \mathsf{parent}^{\mathcal{I}_0} = \emptyset.$$

By the concept inclusion in the TBox, there must be some d_1 that is parent-related to john. So, we expand $\mathcal{I}_0$ to $\mathcal{I}_1$ by taking

$$\Delta^{\mathcal{I}_1} = \{\mathsf{john}, d_1\}, \quad \mathsf{Person}^{\mathcal{I}_1} = \{\mathsf{john}, d_1\} \quad \text{and} \quad \mathsf{parent}^{\mathcal{I}_1} = \{(\mathsf{john}, d_1)\}.$$

Now, the concept inclusion is satisfied for john but fails for d_1, since there must be some d_2 that is parent-related to d_1. So, we expand $\mathcal{I}_1$ to $\mathcal{I}_2$ by taking

$$\Delta^{\mathcal{I}_2} = \{\mathsf{john}, d_1, d_2\}, \ \mathsf{Person}^{\mathcal{I}_2} = \{\mathsf{john}, d_1, d_2\}, \ \mathsf{parent}^{\mathcal{I}_2} = \{(\mathsf{john}, d_1), (d_1, d_2)\}.$$

If we take all the newly introduced labelled nulls d_i to be distinct then, clearly, this process will continue *ad infinitum*; see Fig. 5. (A possibility of making some of the d_i identical will be discussed later on in this section.)

Since our simplified definition of *EL* does not involve $\bot$ and $\neg$, every *EL* knowledge base $(\mathcal{T}, \mathcal{A})$ is consistent and the (possibly infinite) forward chaining procedure constructs the canonical model $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ of $(\mathcal{T}, \mathcal{A})$. As in the case of *RL*, the resulting canonical model is universal: it can be homomorphically mapped into any other model of the knowledge base. Note, however, that universal models are not uniquely defined (for instance, one can take two identical fresh labelled nulls instead of one) and, in contrast to *RL*, some knowledge bases may only have *infinite* universal models. For instance, the knowledge base $(\mathcal{T}, \mathcal{A})$ from Example 17 has no finite universal model. Indeed, suppose, for the sake of contradiction, that there is a finite universal model $\mathcal{U}$ of $(\mathcal{T}, \mathcal{A})$. Then $\mathcal{U}$ must contain a sequence of domain elements connected by parent into a cycle. However, the (canonical) model constructed in Example 17 does not contain a homomorphic image of such a cycle, contrary to our assumption.

On the other hand, if we only want to check concept and role subsumption or find answers to instance queries, then we do not have to consider infinite models. The 'folding' construction we are going describe below re-uses the labelled nulls and reminds of the filtration technique known from modal logic; see, e.g., (Chagrov and Zakharyaschev, 1997).

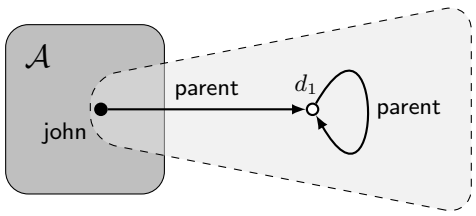

Fig. 6. The small canonical model for the KB from Example 17.

Example 18. Consider $(\mathcal{T}, \mathcal{A})$ from Example 17 and suppose that, on step 2 of forward chaining, instead of introducing a fresh labelled null, d_2, we take d_1 instead. This will result in the following interpretation $\mathcal{I}_*$ (see Fig. 6):

$$\begin{aligned} \Delta^{\mathcal{I}_*} &= \{\mathsf{john}, d_1\}, \\ \mathsf{Person}^{\mathcal{I}_*} &= \{\mathsf{john}, d_1\}, \\ \mathsf{parent}^{\mathcal{I}_*} &= \{(\mathsf{john}, d_1), (d_1, d_1)\}. \end{aligned}$$

Although the interpretation $\mathcal{I}_*$ makes little sense from the modelling point of view (it states that d_1 is its own parent), it is clearly a model of $(\mathcal{T}, \mathcal{A})$. Moreover, as we shall see below, this small model is good enough for checking subsumption and answering instance queries.

To define such small canonical models (also known as generating models) formally, we first convert *EL* TBoxes to a normal form.

Definition 19. An *EL* TBox is said to be in *normal form* if any of its concept inclusions looks as follows:

$$A_1 \sqcap A_2 \sqsubseteq A, \qquad \exists P.D \sqsubseteq A \quad \text{or} \quad A \sqsubseteq \exists P.D,$$

where P is a role name, A, A_1 and A_2 are concept names and D is either a concept name or $\top$.

By introducing abbreviations for complex concepts, one can transform any *EL* TBox to an equivalent one in normal form.

Theorem 20 (Baader et al. (2005)). *Every EL TBox $\mathcal{T}$ can be transformed into a TBox $\mathcal{T}'$ in normal form such that the size of $\mathcal{T}'$ is linear in the size of $\mathcal{T}$, and $\mathcal{T}$ and $\mathcal{T}'$ are equivalent in the following sense*:

- *every model of $\mathcal{T}$ can be extended to a model of $\mathcal{T}'$ by defining interpretations of the fresh concept names,*
- *every model of $\mathcal{T}'$ is a model of $\mathcal{T}$.*

Example 21. Given a concept inclusion $\exists P.A \sqcap B \sqsubseteq \exists R.\exists P.A$, we first take a fresh concept name C that will stand for $\exists P.A$ and obtain

$$C \sqcap B \sqsubseteq \exists R.C, \qquad \underbrace{\exists P.A \sqsubseteq C, \qquad C \sqsubseteq \exists P.A.}_{\text{`}C\text{ is equivalent to }\exists P.A\text{'}}$$

Next, we take another fresh concept name D to replace $\exists R.C$, which results in the following TBox in normal form:

$$C \sqcap B \sqsubseteq D, \qquad \exists P.A \sqsubseteq C, \qquad C \sqsubseteq \exists P.A, \qquad D \sqsubseteq \exists R.C, \qquad \exists R.C \sqsubseteq D.$$

We are now in a position to define generating models formally. Let $(\mathcal{T}, \mathcal{A})$ be an $\mathcal{EL}$ knowledge base in normal form. We say that a concept *occurs positively in* $\mathcal{T}$ if it occurs on the right-hand side of a concept inclusion in $\mathcal{T}$. For each concept $\exists S.D$ occurring positively in $\mathcal{T}$ (where D is either a concept name or $\top$), we introduce a *witness* $w_{\exists S.D}$ and define a *generating relation* $\rightsquigarrow_{\mathcal{T},\mathcal{A}}$ on the set of these witnesses together with $\mathsf{ind}(\mathcal{A})$ by taking:

$$\begin{aligned} a &\rightsquigarrow_{\mathcal{T},\mathcal{A}} w_{\exists S.D} && \text{if} && a \in \mathsf{ind}(\mathcal{A}) \text{ and } (\mathcal{T}, \mathcal{A}) \models (\exists S.D)(a), \\ w_{\exists P.A} &\rightsquigarrow_{\mathcal{T},\mathcal{A}} w_{\exists S.D} && \text{if} && \mathcal{T} \models A \sqsubseteq \exists S.D. \end{aligned}$$

The *generating model* $\mathcal{G}_{\mathcal{T},\mathcal{A}}$ for $(\mathcal{T}, \mathcal{A})$ is defined as follows:

$$\begin{aligned} \Delta^{\mathcal{G}_{\mathcal{T},\mathcal{A}}} &= \mathsf{ind}(\mathcal{A}) \ \cup \ \{\, w_{\exists S.D} \mid \exists S.D \text{ occurs positively in } \mathcal{T} \,\}, \\ a^{\mathcal{G}_{\mathcal{T},\mathcal{A}}} &= a, \qquad \text{for } a \in \mathsf{ind}(\mathcal{A}), \\ A^{\mathcal{G}_{\mathcal{T},\mathcal{A}}} &= \{\, a \in \mathsf{ind}(\mathcal{A}) \mid (\mathcal{T}, \mathcal{A}) \models A(a) \,\} \ \cup \\ &\quad \{\, w_{\exists S.D} \mid \mathcal{T} \models D \sqsubseteq A \,\}, \qquad \text{for a concept name } A, \\ P^{\mathcal{G}_{\mathcal{T},\mathcal{A}}} &= \{(a, b) \mid S(a, b) \in \mathcal{A},\ \mathcal{T} \models S \sqsubseteq P\} \ \cup \\ &\quad \{(w, w_{\exists S.D}) \mid w \rightsquigarrow_{\mathcal{T},\mathcal{A}} w_{\exists S.D},\ \mathcal{T} \models S \sqsubseteq P\}, \quad \text{for a role name } P. \end{aligned}$$

It should be clear that $\mathcal{G}_{\mathcal{T},\mathcal{A}}$ can be constructed in polynomial number of steps by the modified forward chaining procedure that does not invent fresh labelled nulls for $\exists S.D$ but instead re-uses the existing element $w_{\exists S.D}$ in the domain. The following theorem shows that $\mathcal{G}_{\mathcal{T},\mathcal{A}}$ is indeed a model of $(\mathcal{T}, \mathcal{A})$ and it provides enough information about all concept and role subsumptions and about instance queries.

Theorem 22 (Baader et al. (2005)). *Let $(\mathcal{T}, \mathcal{A})$ be an $\mathcal{EL}$ knowledge base. Then $\mathcal{G}_{\mathcal{T},\mathcal{A}} \models (\mathcal{T}, \mathcal{A})$. In addition, we have the following*:

(i) $(\mathcal{T}, \mathcal{A}) \models C_1 \sqsubseteq C_2$ *if and only if* $\mathcal{G}_{\mathcal{T},\mathcal{A}} \models C_1 \sqsubseteq C_2$;
(ii) $(\mathcal{T}, \mathcal{A}) \models P_1 \sqsubseteq P_2$ *if and only if* $\mathcal{G}_{\mathcal{T},\mathcal{A}} \models P_1 \sqsubseteq P_2$;
(iii) $(\mathcal{T}, \mathcal{A}) \models C(a)$ *if and only if* $\mathcal{G}_{\mathcal{T},\mathcal{A}} \models C(a)$;
(iv) $(\mathcal{T}, \mathcal{A}) \models P(a, b)$ *if and only if* $\mathcal{G}_{\mathcal{T},\mathcal{A}} \models P(a, b)$.

Since the generating model $\mathcal{G}_{\mathcal{T},\mathcal{A}}$ can be constructed in polynomial time in the size of $(\mathcal{T}, \mathcal{A})$, concept and role subsumption are tractable; the same concerns instance checking (matching lower bounds will be discussed in Section 4).

Theorem 23. *The problems of concept and role subsumption and instance checking are* P*-complete in EL.*

However, the following example shows that $\mathcal{G}_{\mathcal{T},\mathcal{A}}$ cannot be directly used to compute answers to conjunctive queries.[13]

Example 24. Consider $(\mathcal{T},\mathcal{A})$ from Example 17 and the CQ

$$\boldsymbol{q} \;=\; \exists x\, \mathsf{parent}(x,x).$$

It should be clear that the answer to $\boldsymbol{q}$ over $(\mathcal{T},\mathcal{A})$ is 'no' because $\mathcal{C}_{\mathcal{T},\mathcal{A}} \not\models \boldsymbol{q}$; see Fig. 5. On the other hand, since the generating model $\mathcal{G}_{\mathcal{T},\mathcal{A}}$ contains a loop (see Fig. 6), we have $\mathcal{G}_{\mathcal{T},\mathcal{A}} \models \boldsymbol{q}$.

We call $\mathcal{G}_{\mathcal{T},\mathcal{A}}$ generating because the standard canonical model $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ of $(\mathcal{T},\mathcal{A})$ can be generated by unravelling cycles of the generating relation $\rightsquigarrow_{\mathcal{T},\mathcal{A}}$ into infinite trees: for example, $\mathcal{G}_{\mathcal{T},\mathcal{A}}$ in Fig. 6 can be unravelled into $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ in Fig. 5. We remark that the generating model defined here was initially represented as a pair functions by Brandt (2004) and later called the canonical model; see e.g., (Lutz and Wolter, 2007). We prefer the term generating model to avoid confusion with the (possibly infinite) canonical model (the chase).

In Section 4, we shall return to the problem of answering CQs over *EL* knowledge bases and revisit the canonical model construction. In the meantime, we consider the third profile of OWL 2.

3.3 OWL 2 QL

The OWL 2 QL profile[14] of OWL 2 was designed for *ontology-based data access* via query rewriting, where answering CQs over a knowledge base is reduced to answering first-order queries over a database storing the ABox of the KB (this will be discussed in Section 4). OWL 2 QL is based on the logics of the DL-Lite family (Calvanese et al., 2007; Artale et al., 2009).

In this section, we consider a somewhat simplified version of OWL 2 QL, which will be called *QL*. (It is almost identical to what is known as *DL-Lite*$_{\mathcal{R}}$ (Calvanese et al., 2007) or *DL-Lite*$^{\mathcal{H}}_{core}$ (Artale et al., 2009).) Concept and role inclusions in *QL* are of the form

$$B \sqsubseteq C \qquad \text{and} \qquad R_1 \sqsubseteq R_2,$$

where R_1 and R_2 are roles (role names or their inverses) and B and C are concepts defined by the following grammar:

$$\begin{array}{lcllll} B & ::= & A & | & \exists R.\top, & \\ C & ::= & A & | & \exists R.\top & | \quad \exists R.C. \end{array}$$

[13] The generating model can be used the for answering CQs but the given query has to be modified to take account of identifications of the labelled nulls. This is known as the *combined approach* to query answering, see (Lutz et al., 2009) for the case of *EL* and (Kontchakov et al., 2011) for the case of *QL*. Alternatively, a special procedure has to filter out spurious answers resulting from identification (Lutz et al., 2013).

[14] `www.w3.org/TR/owl2-profiles/#OWL_2_QL`

ABoxes in QL are simple; see Section 3.1. Note that the universal restrictions are not allowed at all and the existential restrictions on the left-hand side of concept inclusions must have $\top$ as their filler (such existential restrictions are called *unqualified* and the $\top$ symbol is often omitted, making $\exists R$ out of $\exists R.\top$); however, the existential restrictions on the right-hand side of concept inclusions can be qualified. Similarly to the case of EL, the standard first-order translations of QL TBoxes require existential quantification, which causes the canonical models to be infinite.

Analogously to EL TBoxes, one can transform any QL TBox to an equivalent one in normal form (see Theorem 20).

Definition 25. A QL TBox is said to be in *normal form* if any concept inclusion in it looks as follows:

$$A' \sqsubseteq A, \qquad \exists R \sqsubseteq A \quad \text{or} \quad A \sqsubseteq \exists R.D,$$

where R is a role, A and A' are concept names and D is either a concept name or $\top$.

Given a QL knowledge base $(\mathcal{T}, \mathcal{A})$ with $\mathcal{T}$ in normal form, we can find all answers to a CQ $\boldsymbol{q}$ over this KB by evaluating $\boldsymbol{q}$ over the (possibly infinite) canonical model $\mathcal{C}_{\mathcal{T},\mathcal{A}}$, which can be constructed using forward chaining (cf. Section 3.2). We begin by taking the standard model $\mathcal{I}_\mathcal{A}$ of the ABox as $\mathcal{I}_0$ (see Definition 12) and apply inductively the following rules to obtain $\mathcal{I}_{k+1}$ from $\mathcal{I}_k$:

(c′) if $d \in B^{\mathcal{I}_k}$ and $B \sqsubseteq A \in \mathcal{T}$, then we add d to $A^{\mathcal{I}_{k+1}}$;
(r′) if $(d, d') \in R_1^{\mathcal{I}_k}$ and $R_1 \sqsubseteq R_2 \in \mathcal{T}$, then we add (d, d') to $R_2^{\mathcal{I}_{k+1}}$;
(e) if $d \in B^{\mathcal{I}_k}$ and $B \sqsubseteq \exists R.D \in \mathcal{T}$, where D is a concept name or $\top$, then we take a *fresh* labelled null, d', and add d' to $D^{\mathcal{I}_{k+1}}$ and (d, d') to $R^{\mathcal{I}_{k+1}}$.

(These rules are similar to the rules from Section 3.2 except that they refer to roles with inverses and QL concepts).

The canonical model $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ constructed using rules **(c′)**, **(r′)** and **(e)** in a bottom-up fashion can alternatively be defined by unravelling the generating structure, which is closer to the top-down approach and will be required for query rewriting in Section 4. There are two key observations that lead us to the alternative definition: first, fresh labelled nulls can only be added by applying **(e)**, and, second, if two labelled nulls, d_1 and d_2, are introduced by applying **(e)** with the same concept inclusion $B \sqsubseteq \exists R.D$ then the same rules will be applicable to d_1 and d_2 in the continuation of the forward chaining procedure. So, each labelled null d' resulting from applying **(e)** to some $B \sqsubseteq \exists R.D$ on a domain element d can be identified with a pair of the form $(d, \exists R.D)$. More formally, for each concept $\exists R.D$ that occurs positively in $\mathcal{T}$, we introduce a *witness* $w_{\exists R.D}$ and define a *generating relation* $\rightsquigarrow_{\mathcal{T},\mathcal{A}}$ on the set of these witnesses together with $\mathsf{ind}(\mathcal{A})$ by taking:

$$\begin{aligned} a &\rightsquigarrow_{\mathcal{T},\mathcal{A}} w_{\exists R.D} \quad \text{if} \quad a \in \mathsf{ind}(\mathcal{A}),\ \mathcal{I}_\mathcal{A} \models B(a) \text{ and } \mathcal{T} \models B \sqsubseteq \exists R.D, \\ w_{\exists S.B} &\rightsquigarrow_{\mathcal{T},\mathcal{A}} w_{\exists R.D} \quad \text{if} \quad \mathcal{T} \models \exists S^- \sqsubseteq \exists R.D \quad \text{or} \quad \mathcal{T} \models B \sqsubseteq \exists R.D. \end{aligned}$$

A $\rightsquigarrow_{\mathcal{T},\mathcal{A}}$-*path* σ is a finite sequence $aw_{\exists R_1.D_1}\cdots w_{\exists R_n.D_n}$, $n \geq 0$, such that $a \in \mathsf{ind}(\mathcal{A})$ and, if $n > 0$, then

$$a \rightsquigarrow_{\mathcal{T},\mathcal{A}} w_{\exists R_1.D_1} \quad \text{and} \quad w_{\exists R_i.D_i} \rightsquigarrow_{\mathcal{T},\mathcal{A}} w_{\exists R_{i+1}.D_{i+1}}, \text{ for } i < n.$$

Thus, a path of the form $\sigma w_{\exists R.D}$ represents the fresh labelled null introduced by applying **(e)** to some $B \sqsubseteq \exists R.D$ on the domain element σ (and which corresponds to the pair $(\sigma, \exists R.D)$ mentioned above). Denote by $\mathsf{tail}(\sigma)$ the last element in σ; as we noted above, the last element in σ uniquely determines all the subsequent rule applications. The *canonical model* $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ is defined by taking $\Delta^{\mathcal{C}_{\mathcal{T},\mathcal{A}}}$ to be the set of all $\rightsquigarrow_{\mathcal{T},\mathcal{A}}$-paths and setting

$$\begin{aligned}
a^{\mathcal{C}_{\mathcal{T},\mathcal{A}}} &= a, && \text{for } a \in \mathsf{ind}(\mathcal{A}),\\
A^{\mathcal{C}_{\mathcal{T},\mathcal{A}}} &= \{a \in \mathsf{ind}(\mathcal{A}) \mid \mathcal{I}_{\mathcal{A}} \models B(a) \text{ and } \mathcal{T} \models B \sqsubseteq A\} \ \cup\\
&\quad \{\sigma w_{\exists R.D} \mid \mathcal{T} \models \exists R^- \sqsubseteq A \text{ or } \mathcal{T} \models D \sqsubseteq A\}, && \text{for a concept name } A,\\
P^{\mathcal{C}_{\mathcal{T},\mathcal{A}}} &= \{(a,b) \mid \mathcal{I}_{\mathcal{A}} \models R(a,b) \text{ and } \mathcal{T} \models R \sqsubseteq P\} \ \cup\\
&\quad \{(\sigma w_{\exists R.D}, \sigma) \mid \mathsf{tail}(\sigma) \rightsquigarrow_{\mathcal{T},\mathcal{A}} w_{\exists R.D},\ \mathcal{T} \models R \sqsubseteq P^-\} \ \cup\\
&\quad \{(\sigma, \sigma w_{\exists R.D}) \mid \mathsf{tail}(\sigma) \rightsquigarrow_{\mathcal{T},\mathcal{A}} w_{\exists R.D},\ \mathcal{T} \models R \sqsubseteq P\}, && \text{for a role name } P.
\end{aligned}$$

Intuitively, by the definition of rule **(c′)**, an ABox individual a belongs to $A^{\mathcal{C}_{\mathcal{T},\mathcal{A}}}$ just in case there is a sequence of concepts $B_0, B_1, \ldots, B_n$ such that $\mathcal{I}_{\mathcal{A}} \models B_0(a)$, the $B_{i-1} \sqsubseteq B_i$ are in $\mathcal{T}$, for $1 \leq i \leq n$, and $B_n = A$; in other words, if $\mathcal{I}_{\mathcal{A}} \models B_0(a)$ and $\mathcal{T} \models B_0 \sqsubseteq A$, for some concept B_0. Similarly, by the definition of rules **(c′)** and **(e)**, a labelled null of the form $\sigma w_{\exists R.D}$ belongs to $A^{\mathcal{C}_{\mathcal{T},\mathcal{A}}}$ just in case $\mathcal{T} \models \exists R^- \sqsubseteq A$ or $\mathcal{T} \models D \sqsubseteq A$. For a role name P, rules **(r′)** and **(e)** provide an analogous argument. More precisely, by the definition of rule **(r′)**, a pair (d, d') of domain elements belongs to $P^{\mathcal{C}_{\mathcal{T},\mathcal{A}}}$ just in case there is a sequence of roles $R_0, \ldots, R_n$ such that $(d, d') \in R_0^{\mathcal{C}_{\mathcal{T},\mathcal{A}}}$, the $R_{i-1} \sqsubseteq R_i$ are in $\mathcal{T}$, for $1 \leq i \leq n$, and $R_n = P$; in other words, $(d, d') \in R_0^{\mathcal{C}_{\mathcal{T},\mathcal{A}}}$ and $\mathcal{T} \models R_0 \sqsubseteq P$, for some role R_0. It then follows from the definition of rule **(e)** that a pair (d, d') belongs to $P^{\mathcal{C}_{\mathcal{T},\mathcal{A}}}$ just in three cases: (i) both elements of the pair are ABox individuals with $\mathcal{I}_{\mathcal{A}} \models R(d, d')$ and $\mathcal{T} \models R \sqsubseteq P$ (ii) the first component of the pair is created by an application of **(e)** to the second component of the pair: $d = \sigma w_{\exists R.D}$, $d' = \sigma$ and $\mathcal{T} \models R \sqsubseteq P^-$ or (iii) the second component of the pair is created by an application of **(e)** to the first component: $d = \sigma$, $d' = \sigma w_{\exists R.D}$ and $\mathcal{T} \models R \sqsubseteq P$. These three cases are reflected in the three sets in the union in the definition of $P^{\mathcal{C}_{\mathcal{T},\mathcal{A}}}$.

Example 26. Consider a *QL* TBox $\mathcal{T}$ with the following concept and role inclusions:

$$\begin{aligned}
&\mathsf{RA} \sqsubseteq \exists \mathsf{worksOn}.\mathsf{Project},\\
&\mathsf{Project} \sqsubseteq \exists \mathsf{isManagedBy}.\mathsf{Prof},\\
&\mathsf{worksOn}^- \sqsubseteq \mathsf{involves},\\
&\mathsf{isManagedBy} \sqsubseteq \mathsf{involves}
\end{aligned}$$

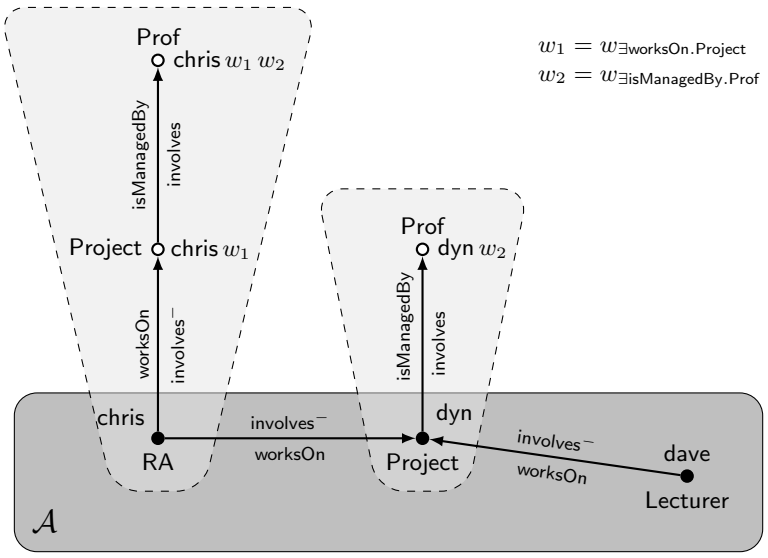

Fig. 7. The canonical model for the knowledge base in Example 26.

and an ABox $\mathcal{A}$ comprising the following assertions:

$$\mathsf{RA}(\mathsf{chris}), \quad \mathsf{worksOn}(\mathsf{chris}, \mathsf{dyn}), \quad \mathsf{Project}(\mathsf{dyn}), \quad \mathsf{Lecturer}(\mathsf{dave}), \quad \mathsf{worksOn}(\mathsf{dave}, \mathsf{dyn}).$$

Two concepts occur positively in $\mathcal{T}$: $\exists\mathsf{worksOn.Project}$ and $\exists\mathsf{isManagedBy.Prof}$. For brevity, the witnesses for them will be denoted by w_1 and w_2, respectively. The generating relation $\rightsquigarrow_{\mathcal{T},\mathcal{A}}$ is then defined by taking

$$\mathsf{chris} \rightsquigarrow_{\mathcal{T},\mathcal{A}} w_1, \qquad \mathsf{dyn} \rightsquigarrow_{\mathcal{T},\mathcal{A}} w_2, \qquad w_1 \rightsquigarrow_{\mathcal{T},\mathcal{A}} w_2.$$

This gives the following $\rightsquigarrow_{\mathcal{T},\mathcal{A}}$-paths:

$$\mathsf{chris}, \qquad \mathsf{chris}\, w_1, \qquad \mathsf{chris}\, w_1\, w_2, \qquad \mathsf{dyn}, \qquad \mathsf{dyn}\, w_2 \quad \text{and} \quad \mathsf{dave}$$

(note that if the graph of $\rightsquigarrow_{\mathcal{T},\mathcal{A}}$ contains a cycle then the set of $\rightsquigarrow_{\mathcal{T},\mathcal{A}}$-paths is infinite; cf. Example 17). The resulting canonical model $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ is depicted in Fig. 7.

For any *QL* knowledge base, the defined canonical model is universal, and thus contains all the necessary information for checking concept and role subsumptions, answering instance queries and, more generally, for answering CQs:

Theorem 27. *Let $(\mathcal{T},\mathcal{A})$ be a QL knowledge base. Then $\mathcal{C}_{\mathcal{T},\mathcal{A}} \models (\mathcal{T},\mathcal{A})$ and, for any CQ $\boldsymbol{q}(\boldsymbol{x})$ and any tuple $\boldsymbol{a}$ in $\mathsf{ind}(\mathcal{A})$, we have*

$$(\mathcal{T},\mathcal{A}) \models \boldsymbol{q}(\boldsymbol{a}) \quad \text{if and only if} \quad \mathcal{C}_{\mathcal{T},\mathcal{A}} \models \boldsymbol{q}(\boldsymbol{a}).$$

Although the canonical models in *EL* and *QL* contain all the information required to compute answers to conjunctive queries, these canonical models are not necessarily finite (in contrast to *RL*), and therefore cannot be simply materialised, say, by a datalog engine, a triple store or a database management system. In the next section, we analyse the problem of answering CQs and develop practical techniques for dealing with infinite canonical models.

4 Conjunctive Query Answering via Query Rewriting

Database management systems implement sophisticated algorithms for evaluating SQL queries over relational data instances. The (theoretically and empirically supported) fact that databases have been very efficient in practice suggests the following approach to answering queries over knowledge bases. We store the assertions of a given ABox $\mathcal{A}$ in a relational database. Given a CQ $\boldsymbol{q}(\boldsymbol{x})$, we use the inclusions of the TBox $\mathcal{T}$ to 'rewrite' $\boldsymbol{q}(\boldsymbol{x})$ into another query $\boldsymbol{q}'(\boldsymbol{x})$ that would return, when evaluated over the data instance $\mathcal{A}$, all the certain answers to $\boldsymbol{q}(\boldsymbol{x})$ over $(\mathcal{T}, \mathcal{A})$. It is important to emphasise here that the rewriting $\boldsymbol{q}'(\boldsymbol{x})$ must only depend on the TBox $\mathcal{T}$ and the given query $\boldsymbol{q}(\boldsymbol{x})$, and so should work for *all possible* ABoxes $\mathcal{A}$. Thus, we arrive to the following definition.

We call a CQ $\boldsymbol{q}(\boldsymbol{x})$ and a TBox $\mathcal{T}$ *first-order rewritable* (FO-rewritable, for short) if there exists a first-order formula $\boldsymbol{q}'(\boldsymbol{x})$ such that, for any ABox $\mathcal{A}$ and any tuple $\boldsymbol{a}$ of individuals in $\mathcal{A}$, we have

$$(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q}(\boldsymbol{a}) \qquad \text{if and only if} \qquad \mathcal{I}_{\mathcal{A}} \models \boldsymbol{q}'(\boldsymbol{a}), \tag{10}$$

where $\mathcal{I}_{\mathcal{A}}$ is the standard model of $\mathcal{A}$ (see Definition 12). The formula $\boldsymbol{q}'(\boldsymbol{x})$ is called an *FO-rewriting of* $\boldsymbol{q}$ *and* $\mathcal{T}$.

This idea of reducing CQ answering over knowledge bases to database query answering, first formulated by Calvanese et al. (2005), may sound too good to be applicable in all cases. In fact, there are a few issues in realising this idea.

A minor one is that, from a practical point of view, if an FO-rewriting $\boldsymbol{q}'(\boldsymbol{x})$ is to be executed by a relational database engine then $\boldsymbol{q}'(\boldsymbol{x})$ must be a domain-independent query (Abiteboul et al., 1995). This is the case, in particular, for FO-rewritings that contain only conjunction, disjunction and existential quantifiers. Such rewritings (and formulas) are called *positive existential rewritings* (PE-rewritings, for short).

A more serious issue is that not all DL constructs can guarantee FO-rewritability of all CQs. To understand why, let us recall (Libkin, 2004) that the problem of evaluating a first-order formula in a given interpretation belongs the class AC^0 for *data complexity*. Data complexity is a complexity measure that only takes account of the size of the data (the interpretation in this case) and regards the query to be fixed. This measure was suggested by Vardi (1982) who tried to find a theoretical explanation of the practical efficiency of database management systems. It is also known from the complexity theory that AC^0 is a *proper* subclass of LogSpace and that $\text{LogSpace} \subseteq \text{NLogSpace} \subseteq \text{P}$ (Papadimitriou, 1994; Arora and Barak, 2009). It follows that if the problem '$(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q}$?' is

NLogSpace- or P-hard, for some fixed $\boldsymbol{q}$ and $\mathcal{T}$, then these $\boldsymbol{q}$ and $\mathcal{T}$ cannot be FO-rewritable. This observation allows us to identify the DL constructs that can ruin FO-rewritability. Here we give two simple examples illustrating this technique; for more consult, e.g., (Calvanese et al., 2006; Artale et al., 2009).

Example 28. A typical example of an NLogSpace-complete problem is the *reachability problem* for directed graphs: given a directed graph $G = (V, E)$ with vertices V and arcs E and two distinguished vertices $s, t \in V$, decide whether there is a directed path from s to t in G. We represent the graph and the target vertex by means of an ABox $\mathcal{A}_{G,t}$ comprising

$$\begin{aligned} &\mathsf{edge}(v_1, v_2), \quad \text{for } (v_1, v_2) \in E, \\ &\mathsf{ReachableFromTarget}(t). \end{aligned}$$

Consider now a TBox containing concept inclusion

$$\exists \mathsf{edge}.\mathsf{ReachableFromTarget} \sqsubseteq \mathsf{ReachableFromTarget}$$

and the Boolean CQ

$$\boldsymbol{q} \;= \mathsf{ReachableFromTarget}(s).$$

It is readily seen that $(\mathcal{T}, \mathcal{A}_{G,t}) \models \boldsymbol{q}$ if and only if there is a directed path from s to t in G. Therefore, the problem '$(\mathcal{T}, \mathcal{A}_{G,t}) \models \boldsymbol{q}$?' is NLogSpace-hard. Since $\boldsymbol{q}$ and $\mathcal{T}$ are fixed (and do not depend on G or t), $\boldsymbol{q}$ and $\mathcal{T}$ cannot be FO-rewritable. In other words, TBoxes capable of computing the transitive closure of some relations in ABoxes do not allow FO-rewritability.

Example 29. Next, we consider an example of a P-complete problem. Let (V, E) be a pair that consists of a finite set of vertices V and a relation $E \subseteq V \times V \times V$. A vertex $v \in V$ is said to be *accessible* from a set $S \subseteq V$ of source vertices in (V, E) if either $v \in S$ or $(v_1, v_2, v) \in E$, for some v_1 and v_2 that are accessible from S in (V, E) (v_1 and v_2 are called inputs and v the output; such a triple can also be thought of as an implication of the form $v_1 \wedge v_2 \to v$). The *path system accessibility* problem is defined as follows: given (V, E) as above, source vertices $S \subseteq V$ and a terminal vertex $t \in V$, decide whether t is accessible from S in (V, E). This problem is known to be P-complete (Garey and Johnson, 1979). The path system (V, E) and source vertices S can be encoded by an ABox $\mathcal{A}_{V,E,S}$ in the following way:

$$\begin{aligned} &\mathsf{Accessible}(v), && \text{for } v \in S, \\ &\mathsf{input}_1(e, v_1),\ \mathsf{input}_2(e, v_2) \text{ and } \mathsf{output}(v, e), && \text{for } e = (v_1, v_2, v) \in E. \end{aligned}$$

Consider now a TBox $\mathcal{T}$ containing

$$\begin{aligned} \exists \mathsf{input}_1.\mathsf{Accessible} \sqcap \exists \mathsf{input}_2.\mathsf{Accessible} &\sqsubseteq \mathsf{BothInputsAccessible}, \\ \exists \mathsf{output}.\mathsf{BothInputsAccessible} &\sqsubseteq \mathsf{Accessible} \end{aligned}$$

and the Boolean CQ

$$\boldsymbol{q} = \mathsf{Accessible}(t).$$

It should be clear that $(\mathcal{T}, \mathcal{A}_{V,E,S}) \models \mathsf{Accessible}(v)$ if and only if v is accessible from S in (V, E); and $(\mathcal{T}, \mathcal{A}) \models \mathsf{BothInputsAccessible}(e)$ if and only if *both* inputs of e are accessible, that is, both are instances of $\mathsf{Accessible}$. Therefore, the answer to $\boldsymbol{q}$ is 'yes' if and only if t is accessible from S in (V, E). Thus, the problem '$(\mathcal{T}, \mathcal{A}_{V,E,S}) \models \boldsymbol{q}$' is P-hard, and so these fixed $\boldsymbol{q}$ and $\mathcal{T}$ cannot be FO-rewritable.

The OWL 2 QL profile of OWL 2 was designed so that problems such as graph reachability and path system accessibility above could not be expressed in it (on the other hand, observe that both of the TBoxes above belong to *RL* and *EL*, which proves the lower complexity bounds in Theorems 16 and 23). A number of various rewriting techniques have been proposed and implemented for OWL 2 QL: PerfectRef (Poggi et al., 2008), Presto / Prexto (Rosati and Almatelli, 2010; Rosati, 2012), as well as for extensions of OWL 2 QL to datalog$^{\pm}$ and existential rules: Nyaya (Gottlob et al., 2011) and PURE (König et al., 2012) and more expressive DLs: Requiem / Blackout (Pérez-Urbina et al., 2009, 2012), Rapid (Chortaras et al., 2011) and Clipper (Eiter et al., 2012), which go beyond FO-rewritability.

In this section, we discuss the tree-witness rewriting (Kikot et al., 2012b). We require the following definitions in the sequel. Whenever convenient, we write $S(\boldsymbol{z})$ for either a unary atom $A(z_1)$ or a binary atom $P(z_1, z_2)$; we also identify $P^-(z_2, z_1)$ with $P(z_1, z_2)$. Any CQ $\boldsymbol{q}(\boldsymbol{x}) = \exists \boldsymbol{y}\, \varphi(\boldsymbol{x}, \boldsymbol{y})$ is regarded as the set of atoms in φ, so we can write $S(\boldsymbol{z}) \in \boldsymbol{q}$ (when referring to the query as a set of atoms, we often omit the answer variables). Any set $\boldsymbol{q}$ of atoms can also be viewed as an interpretation over the domain of its terms (variables and individual names) such that an atom $S(\boldsymbol{z})$ is true in this interpretation just in case $S(\boldsymbol{z}) \in \boldsymbol{q}$. We slightly abuse notation and denote by $\boldsymbol{q}$ both the set of atoms and the corresponding interpretation. The reason behind these definitions and notations is as follows: it is not hard to see that $\mathcal{I} \models \boldsymbol{q}(\boldsymbol{a})$ if and only if there is a homomorphism from $\boldsymbol{q}(\boldsymbol{a})$ to $\mathcal{I}$ (see Definition 14).

4.1 PE-Rewriting for Flat *QL* (and RDFS)

We first consider an important special case of *flat* *QL* TBoxes that do not contain existential quantifiers on the right-hand side of concept inclusions. In other words, flat *QL* TBoxes in normal form can only contain concept and role inclusions of the form

$$A \sqsubseteq A', \qquad \exists R \sqsubseteq A \quad \text{and} \quad R \sqsubseteq R',$$

for concept names A and A' and roles R and R'. Note that the language of flat *QL* TBoxes differs from the language of RDFS[15] only in that *QL* allows inverse roles in role inclusions whereas RDFS restricts role inclusions to just role names.

[15] `www.w3.org/TR/rdf-schema`

Let $\mathcal{T}$ be a flat QL TBox and $\boldsymbol{q}(\boldsymbol{x})$ a conjunctive query. By Theorem 27, $(\mathcal{T},\mathcal{A}) \models \boldsymbol{q}(\boldsymbol{a})$ if and only if $\boldsymbol{q}(\boldsymbol{a})$ is true in the canonical model $\mathcal{C}_{\mathcal{T},\mathcal{A}}$. Since the TBox is flat, the generating relation $\rightsquigarrow_{\mathcal{T},\mathcal{A}}$ is empty, the canonical model $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ contains no labelled nulls, and so, by the definition of $\mathcal{C}_{\mathcal{T},\mathcal{A}}$, we have

$$\begin{array}{rll} \mathcal{C}_{\mathcal{T},\mathcal{A}} \models A(a) & \text{if and only if} & \mathcal{I}_{\mathcal{A}} \models B(a) \text{ and } \mathcal{T} \models B \sqsubseteq A, \text{ for some } B, \\ \mathcal{C}_{\mathcal{T},\mathcal{A}} \models P(a,b) & \text{if and only if} & \mathcal{I}_{\mathcal{A}} \models R(a,b) \text{ and } \mathcal{T} \models R \sqsubseteq P, \text{ for some } R. \end{array}$$

Define a PE-formula $\boldsymbol{q}_{\mathsf{ext}}(\boldsymbol{x})$ to be the result of replacing every atom $A(z)$ in $\boldsymbol{q}(\boldsymbol{x})$ with $A_{\mathsf{ext}}(z)$ and every atom $P(z_1,z_2)$ in $\boldsymbol{q}(\boldsymbol{x})$ with $P_{\mathsf{ext}}(z_1,z_2)$, where

$$A_{\mathsf{ext}}(u) = \bigvee_{\mathcal{T} \models B \sqsubseteq A} ST_u(B), \qquad P_{\mathsf{ext}}(u,v) = \bigvee_{\mathcal{T} \models R \sqsubseteq P} ST_{u,v}(R),$$

and ST is the standard translation defined in Section 3.1. It is not hard to see that, for any ABox $\mathcal{A}$ and any tuple $\boldsymbol{a}$ from $\mathsf{ind}(\mathcal{A})$, we have $\mathcal{C}_{\mathcal{T},\mathcal{A}} \models \boldsymbol{q}(\boldsymbol{a})$ if and only if $\mathcal{I}_{\mathcal{A}} \models \boldsymbol{q}_{\mathsf{ext}}(\boldsymbol{a})$.

Proposition 30. *For any CQ $\boldsymbol{q}(\boldsymbol{x})$ and any flat QL TBox $\mathcal{T}$, $\boldsymbol{q}_{\mathsf{ext}}(\boldsymbol{x})$ is a PE-rewriting of $\boldsymbol{q}(\boldsymbol{x})$ and $\mathcal{T}$.*

Thus, in the case of flat QL TBoxes, it is really easy to construct PE-rewritings.

Example 31. Consider the CQ

$$\boldsymbol{q}(x,y) \;\; = \;\; \mathsf{Student}(x) \land \mathsf{takesCourse}(x,y) \land \mathsf{teacherOf}(\mathsf{p0},y)$$

and a flat QL TBox $\mathcal{T}$ with the following concept and role inclusions:

$$\begin{array}{l} \mathsf{UndergraduateStudent} \sqsubseteq \mathsf{Student}, \\ \mathsf{enrolledAt} \sqsubseteq \mathsf{Student}, \\ \mathsf{teaches}^{-} \sqsubseteq \mathsf{teacherOf}. \end{array}$$

We then define the following formulas for concept and role names from the query:

$$\begin{array}{rcl} \mathsf{Student}_{\mathsf{ext}}(u) & = & \mathsf{Student}(u) \;\lor\; \mathsf{UndergraduateStudent}(u) \;\lor\; \exists v\, \mathsf{enrolledAt}(u,v), \\ \mathsf{takesCourse}_{\mathsf{ext}}(u,v) & = & \mathsf{takesCourse}(u,v), \\ \mathsf{teacherOf}_{\mathsf{ext}}(u,v) & = & \mathsf{teacherOf}(u,v) \;\lor\; \mathsf{teaches}(v,u), \end{array}$$

and we obtain the following PE-rewriting of $\boldsymbol{q}(x,y)$ and $\mathcal{T}$:

$$\begin{array}{rcl} \boldsymbol{q}_{\mathsf{ext}}(x,y) & = & \big(\mathsf{Student}(x) \lor \mathsf{UndergraduateStudent}(x) \lor \exists v\, \mathsf{enrolledAt}(x,v)\big) \;\land \\ & & \mathsf{takesCourse}(x,y) \;\land\; \big(\mathsf{teacherOf}(\mathsf{p0},y) \lor \mathsf{teaches}(y,\mathsf{p0})\big). \end{array}$$

Next, we turn to the case of general QL TBoxes.

4.2 Tree-Witness PE-Rewriting for Full *QL*

Suppose $\mathcal{T}$ is a *QL* TBox in normal form. By Theorem 27, to compute certain answers to $\boldsymbol{q}(\boldsymbol{x})$ over $(\mathcal{T}, \mathcal{A})$, for some $\mathcal{A}$, it is enough to find answers to $\boldsymbol{q}(\boldsymbol{x})$ in the canonical model $\mathcal{C}_{\mathcal{T},\mathcal{A}}$. To do this, we have to check, for every tuple $\boldsymbol{a}$ of elements in $\mathsf{ind}(\mathcal{A})$, whether there exists a homomorphism from $\boldsymbol{q}(\boldsymbol{a})$ to $\mathcal{C}_{\mathcal{T},\mathcal{A}}$. Thus, as in the case of flat TBoxes, the answer variables take values from $\mathsf{ind}(\mathcal{A})$. However, the existentially quantified variables in $\boldsymbol{q}(\boldsymbol{x})$ can be mapped both to $\mathsf{ind}(\mathcal{A})$ and to the labelled nulls in $\mathcal{C}_{\mathcal{T},\mathcal{A}}$. In order to define the rewriting that does not depend on a particular ABox, we need to have a closer look at the structure of the canonical models $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ with fixed $\mathcal{T}$ and varying $\mathcal{A}$.

Let $\exists R.D$ be a concept occurring positively in $\mathcal{T}$ (recall that D is either a concept name or $\top$). For an individual name a, we define the *$\exists R.D$-generated $\mathcal{T}$-tree on a* as the restriction of the canonical model of the KB $(\mathcal{T}, \{(\exists R.D)(a)\})$ to the domain that consists of a and the labelled nulls with the prefix $aw_{\exists R.D}$. We denote this tree by $\mathcal{C}_{\mathcal{T}}^{\exists R.D}(a)$. (Note that $\mathcal{C}_{\mathcal{T}}^{\exists R.D}(a)$ is not necessarily a model of $\mathcal{T}$.) Take now any ABox $\mathcal{A}$ and any $a \in \mathsf{ind}(\mathcal{A})$. By the definition of the canonical model, if $a \rightsquigarrow_{\mathcal{T},\mathcal{A}} w_{\exists R.D}$ then $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ contains a sub-tree that is isomorphic to the $\exists R.D$-generated $\mathcal{T}$-tree on a, excluding possibly the root. Moreover, such sub-trees may intersect only on their common root a. For instance, the canonical model in Fig. 7 contains the $\exists \mathsf{worksOn}.\mathsf{Project}$-generated $\mathcal{T}$-tree on chris and the $\exists \mathsf{isManagedBy}.\mathsf{Prof}$-generated $\mathcal{T}$-tree on dyn. The following example illustrates a more complex configuration.

Example 32. Consider a TBox $\mathcal{T}$ with the following concept inclusions:

$$A \sqsubseteq \exists R.D, \qquad D \sqsubseteq \exists P_1, \qquad D \sqsubseteq \exists P_2, \qquad B \sqsubseteq \exists S.C$$

and suppose that an ABox $\mathcal{A}$ contains $A(a), P_1(a,b), A(b), B(b), P_2(b,c)$. The canonical model $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ is depicted in Fig. 8. The individual a in this canonical model has a single $\mathcal{T}$-tree generated by $\exists R.D$. The individual b has two $\mathcal{T}$-trees, one generated by $\exists R.D$ and another by $\exists S.C$. These two $\mathcal{T}$-trees intersect only on their common root b.

For a more formal treatment, we require the following opertation. Given interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ (under the standard name assumption), we define their *join*, $\mathcal{I}_1 \oplus \mathcal{I}_2$, by taking

$$\begin{aligned}
&\Delta^{\mathcal{I}_1 \oplus \mathcal{I}_2} = \Delta^{\mathcal{I}_1} \cup \Delta^{\mathcal{I}_2}, \\
&a^{\mathcal{I}_1 \oplus \mathcal{I}_2} = a, && \text{for an individual } a \text{ in } \mathcal{I}_1 \text{ or } \mathcal{I}_2, \\
&A^{\mathcal{I}_1 \oplus \mathcal{I}_2} = A^{\mathcal{I}_1} \cup A^{\mathcal{I}_2}, && \text{for a concept name } A, \\
&P^{\mathcal{I}_1 \oplus \mathcal{I}_2} = P^{\mathcal{I}_1} \cup P^{\mathcal{I}_2}, && \text{for a role name } P.
\end{aligned}$$

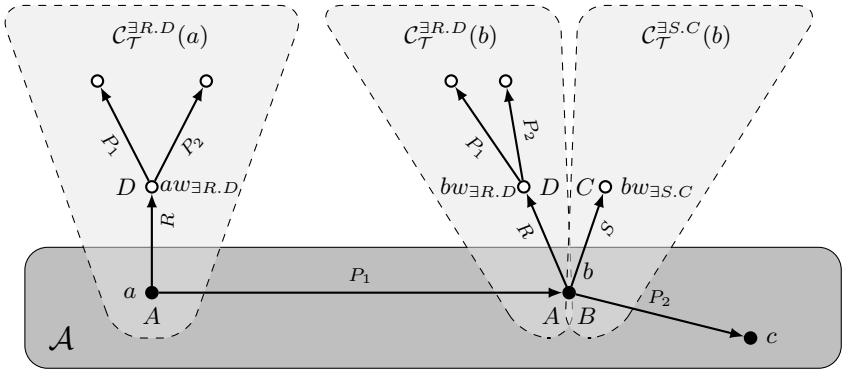

Fig. 8. The canonical model $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ from Example 32.

Then the canonical model $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ of any *QL* knowledge base $(\mathcal{T},\mathcal{A})$, with $\mathcal{T}$ in normal form, can be represented as the following join:

$$\mathcal{C}_{\mathcal{T},\mathcal{A}} \;=\; \mathcal{I}^*_{\mathcal{A}} \;\oplus \bigoplus_{\substack{a\in\mathsf{ind}(\mathcal{A})\\ B\sqsubseteq\exists R.D\in\mathcal{T} \text{ with } \mathcal{I}^*_{\mathcal{A}}\models B(a)}} \mathcal{C}_{\mathcal{T}}^{\exists R.D}(a), \tag{11}$$

where $\mathcal{I}^*_{\mathcal{A}}$ is a model of $\mathcal{A}$ with domain $\mathsf{ind}(\mathcal{A})$, which will be called the *ABox part* of $\mathcal{C}_{\mathcal{T},\mathcal{A}}$; the join of the $\mathcal{C}_{\mathcal{T}}^{\exists R.D}(a)$ will be called the *anonymous part of* $\mathcal{C}_{\mathcal{T},\mathcal{A}}$. We are now fully equipped to present the tree-witness rewriting of a CQ $\boldsymbol{q}(\boldsymbol{x})$ and a *QL* TBox $\mathcal{T}$.

Following the divide and conquer strategy, we show how the process of constructing FO-rewritings can be split into two steps: the first step considers only the flat part of the TBox and uses the formulas $A_{\mathsf{ext}}(u)$ and $P_{\mathsf{ext}}(u,v)$ defined in Section 4.1; the second step (to be described below) takes account of the remaining part of the TBox, that is, inclusions of the form $B \sqsubseteq \exists R.D$.

H-completeness Let $\mathcal{T}$ be a (not necessarily flat) *QL* TBox. A simple ABox $\mathcal{A}$ is said to be *H-complete with respect to* $\mathcal{T}$ if, for all concept names A and role names P, we have

$$\begin{aligned} A(a) \in \mathcal{A} \quad &\text{if} \quad \mathcal{I}_{\mathcal{A}} \models B(a) \text{ and } \mathcal{T} \models B \sqsubseteq A, \text{ for some } B = A' \text{ or } B = \exists R,\\ P(a,b) \in \mathcal{A} \quad &\text{if} \quad \mathcal{I}_{\mathcal{A}} \models R(a,b) \text{ and } \mathcal{T} \models R \sqsubseteq P, \text{ for some } R. \end{aligned}$$

We say that a first-order formula $\boldsymbol{q}'(\boldsymbol{x})$ is an *FO-rewriting of* $\boldsymbol{q}(\boldsymbol{x})$ *and* $\mathcal{T}$ *over H-complete ABoxes* if (10) holds for any ABox $\mathcal{A}$ that is H-complete with respect to $\mathcal{T}$ and any tuple $\boldsymbol{a}$ from $\mathsf{ind}(\mathcal{A})$; as before, a PE-rewriting is an FO-rewriting that contains only conjunction, disjunction and existential quantification.

Observe that if an ABox $\mathcal{A}$ is H-complete with respect to $\mathcal{T}$ then the ABox part of $\mathcal{C}_{\mathcal{T},\mathcal{A}}$, that is, $\mathcal{I}^*_{\mathcal{A}}$ in (11), coincides with $\mathcal{I}_{\mathcal{A}}$. Thus, if $\mathcal{T}$ is flat then $\boldsymbol{q}(\boldsymbol{x})$ itself is clearly a PE- and FO-rewriting of $\boldsymbol{q}(\boldsymbol{x})$ and $\mathcal{T}$ over H-complete ABoxes.

More generally, we can easily obtain rewritings (over arbitrary ABoxes) from rewritings over H-complete ABoxes:

Proposition 33. *If $\boldsymbol{q}'(\boldsymbol{x})$ is an FO-rewriting (PE-rewriting) of $\boldsymbol{q}(\boldsymbol{x})$ and $\mathcal{T}$ over H-complete ABoxes, then $\boldsymbol{q}'_{\mathsf{ext}}(\boldsymbol{x})$ is an FO-rewriting (respectively, PE-rewriting) of $\boldsymbol{q}(\boldsymbol{x})$ and $\mathcal{T}$.*

Thus, we can only focus on constructing PE-rewritings over H-complete ABoxes.

Tree Witnesses Consider a CQ $\boldsymbol{q}(\boldsymbol{x})$ and a knowledge base $(\mathcal{T}, \mathcal{A})$. Suppose that, for some tuple $\boldsymbol{a}$ in $\mathsf{ind}(\mathcal{A})$, there is a homomorphism h from $\boldsymbol{q}(\boldsymbol{a})$ to $\mathcal{C}_{\mathcal{T},\mathcal{A}}$. Then h partitions $\boldsymbol{q}(\boldsymbol{a})$ into the atoms mapped by h to the ABox part and atoms mapped by h to the $\exists R.D$-generated $\mathcal{T}$-trees of the anonymous part of $\mathcal{C}_{\mathcal{T},\mathcal{A}}$. The tree-witness rewriting of $\boldsymbol{q}(\boldsymbol{x})$ and $\mathcal{T}$ we are going to present now lists all possible partitions of the atoms of $\boldsymbol{q}(\boldsymbol{x})$ into such subsets. We begin with an example illustrating this idea.

Example 34. Consider the *QL* TBox $\mathcal{T}$ from Example 26 with the concept and role inclusions

$$\mathsf{RA} \sqsubseteq \exists \mathsf{worksOn}.\mathsf{Project}, \tag{12}$$

$$\mathsf{Project} \sqsubseteq \exists \mathsf{isManagedBy}.\mathsf{Prof}, \tag{13}$$

$$\mathsf{worksOn}^{-} \sqsubseteq \mathsf{involves}, \tag{14}$$

$$\mathsf{isManagedBy} \sqsubseteq \mathsf{involves} \tag{15}$$

and the CQ asking to find those who work with professors:

$$\boldsymbol{q}(x) \;=\; \exists y, z \, \big(\mathsf{worksOn}(x, y) \land \mathsf{involves}(y, z) \land \mathsf{Prof}(z)\big).$$

Recall that if the canonical model $\mathcal{C}_{\mathcal{T},\mathcal{A}}$, for some ABox $\mathcal{A}$, contains some individuals $a \in \mathsf{RA}^{\mathcal{C}_{\mathcal{T},\mathcal{A}}}$ and $b \in \mathsf{Project}^{\mathcal{C}_{\mathcal{T},\mathcal{A}}}$, then $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ must also contain the $\exists\mathsf{worksOn}.\mathsf{Project}$-generated $\mathcal{T}$-tree on a and the $\exists\mathsf{isManagedBy}.\mathsf{Prof}$-generated $\mathcal{T}$-tree on b; see Fig. 7, where such an a is chris and such a b is dyn. Let us consider all possible ways of obtaining certain answers to the query, that is, all possible homomorphisms from atoms of $\boldsymbol{q}(x)$ to $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ such that the answer variable x is mapped to $\mathsf{ind}(\mathcal{A})$. First, x, y and z can be mapped to ABox individuals. Alternatively, x and y can be mapped to ABox individuals, a and b, and if b belongs to $\mathsf{Project}^{\mathcal{C}_{\mathcal{T},\mathcal{A}}}$, then there is a homomorphism h_1 from the last two atoms of $\boldsymbol{q}(a)$ to the anonymous part; see Fig. 9. Another option is to map only x to an ABox individual, a, and if a belongs to $\mathsf{RA}^{\mathcal{C}_{\mathcal{T},\mathcal{A}}}$ then the whole $\boldsymbol{q}(a)$ can be homomorphically mapped to the anonymous part; see h_2 in Fig. 9. Finally, another homomorphism, h_3 in Fig. 9, maps both x and z to a provided that a is in $\mathsf{RA}^{\mathcal{C}_{\mathcal{T},\mathcal{A}}}$ and $\mathsf{Prof}^{\mathcal{C}_{\mathcal{T},\mathcal{A}}}$ at the same time. The possible ways of mapping subsets of a query to the anonymous part of the canonical model are called *tree*

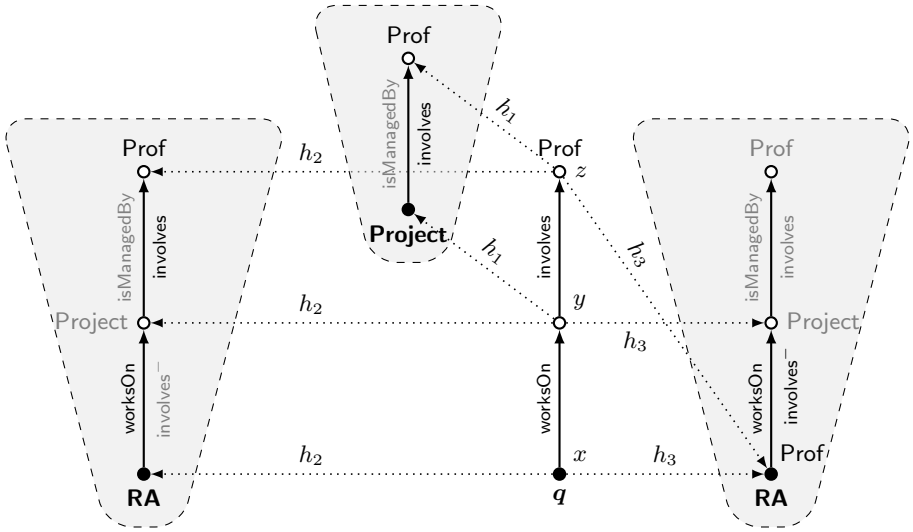

Fig. 9. Three homomorphisms from subsets of $\boldsymbol{q}$ to $\mathcal{T}$-trees.

witnesses. The three tree witnesses for $\boldsymbol{q}(x)$ and $\mathcal{T}$ found above give rise to the following PE-rewriting $\boldsymbol{q}_{\mathsf{tw}}(x)$ of $\boldsymbol{q}(x)$ and $\mathcal{T}$ over H-complete ABoxes:

$$\begin{aligned}\boldsymbol{q}_{\mathsf{tw}}(x) \;=\; \exists y, z \Big[&\big(\mathsf{worksOn}(x,y) \land \mathsf{involves}(y,z) \land \mathsf{Prof}(z)\big) \lor \\ &\big(\mathsf{worksOn}(x,y) \land \mathsf{Project}(y)\big) \lor \\ &\mathsf{RA}(x) \lor \big(\mathsf{RA}(x) \land \mathsf{Prof}(z) \land (x = z)\big)\Big].\end{aligned}$$

We now give a general definition of the tree-witness rewriting over H-complete ABoxes. Let $\mathcal{T}$ be a *QL* TBox in normal form and $\boldsymbol{q}(\boldsymbol{x})$ a CQ. Consider a pair $\mathfrak{t} = (\mathfrak{t}_{\mathsf{r}}, \mathfrak{t}_{\mathsf{i}})$ of disjoint sets of terms in $\boldsymbol{q}(\boldsymbol{x})$, where $\mathfrak{t}_{\mathsf{i}}$ is non-empty and contains only existentially quantified variables ($\mathfrak{t}_{\mathsf{r}}$, on the other hand, can be empty and can contain answer variables and individual names). Set

$$\boldsymbol{q}_{\mathfrak{t}} \;=\; \big\{\, S(\boldsymbol{z}) \in \boldsymbol{q} \mid \boldsymbol{z} \subseteq \mathfrak{t}_{\mathsf{r}} \cup \mathfrak{t}_{\mathsf{i}} \text{ and } \boldsymbol{z} \not\subseteq \mathfrak{t}_{\mathsf{r}} \,\big\}.$$

We say that $\mathfrak{t}$ is a *tree witness for* $\boldsymbol{q}(\boldsymbol{x})$ *and* $\mathcal{T}$ *generated by* $\exists R.D$ if the following two conditions are satisfied:

(tw1) there exists a homomorphism h from $\boldsymbol{q}_{\mathfrak{t}}$ to $\mathcal{C}_{\mathcal{T}}^{\exists R.D}(a)$, for some a, such that $\mathfrak{t}_{\mathsf{r}} = \{z \mid h(z) = a\}$ and $\mathfrak{t}_{\mathsf{i}}$ contains the remaining variables in $\boldsymbol{q}_{\mathfrak{t}}$,

(tw2) $\boldsymbol{q}_{\mathfrak{t}}$ is a *minimal* subset of $\boldsymbol{q}$ such that, for any $y \in \mathfrak{t}_{\mathsf{i}}$, every atom in $\boldsymbol{q}$ containing y belongs to $\boldsymbol{q}_{\mathfrak{t}}$.

Note that unary atoms with arguments in $\mathfrak{t}_{\mathsf{r}}$ or binary atoms with *both* arguments in $\mathfrak{t}_{\mathsf{r}}$ do not belong to $\boldsymbol{q}_{\mathfrak{t}}$ and, therefore, condition **(tw1)** does not require them

to be homomorphically mapped into $\mathcal{C}_{\mathcal{T}}^{\exists R.D}(a)$. The terms in $\mathfrak{t}_\mathsf{r}$ (if any) are called the *roots* of $\mathfrak{t}$ and the (existentially quantified) variables in $\mathfrak{t}_\mathsf{i}$ the *interior* of $\mathfrak{t}$. The homomorphism h in condition **(tw1)** is not necessarily unique; however, it is important that all roots are mapped to a and all variables of the interior are *not* mapped to a. Thus, $\boldsymbol{q}_\mathfrak{t}$ can contain at most one individual name, a; if $\boldsymbol{q}_\mathfrak{t}$ does not contain an individual name then the choice of a is irrelevant. Condition **(tw2)** reflects the fact that if a homomorphism from $\boldsymbol{q}$ to the canonical model of $(\mathcal{T},\mathcal{A})$, for some $\mathcal{A}$, maps a variable y of an atom $R(y,z)$ to a non-root of a tree $\mathcal{C}_{\mathcal{T}}^{\exists R.D}(a)$ then the other variable of the atom must be mapped to the same $\exists R.D$-generated $\mathcal{T}$-tree on a.)

Let $\mathfrak{t} = (\mathfrak{t}_\mathsf{r}, \mathfrak{t}_\mathsf{i})$ be a tree witness for $\boldsymbol{q}(\boldsymbol{x})$ and $\mathcal{T}$. Consider the following formula

$$\mathsf{tw}_\mathfrak{t} \;=\; \exists u \Big[\bigwedge_{x \in \mathfrak{t}_\mathsf{r}} (x = u) \quad \wedge \bigvee_{\substack{B \sqsubseteq \exists R.D \in \mathcal{T} \\ \mathfrak{t} \text{ generated by } \exists R.D}} ST_u(B) \Big], \tag{16}$$

whose free variables are the roots, $\mathfrak{t}_\mathsf{r}$, of $\mathfrak{t}$. The formula $\mathsf{tw}_\mathfrak{t}$ describes the ABox individuals that root the trees in the anonymous part of $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ into which the atoms $\boldsymbol{q}_\mathfrak{t}$ of the tree witness $\mathfrak{t}$ can be homomorphically mapped. More formally, if $\mathcal{I}_\mathcal{A} \models \mathsf{tw}_\mathfrak{t}(a,\dots,a)$, for some $a \in \mathsf{ind}(\mathcal{A})$, then $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ contains the $\exists R.D$-generated $\mathcal{T}$-tree on a, and so there is a homomorphism from $\boldsymbol{q}_\mathfrak{t}$ to $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ that maps all the roots of $\mathfrak{t}$ to a. Conversely, if there is a homomorphism from $\boldsymbol{q}_\mathfrak{t}$ to $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ such that all the roots of $\mathfrak{t}$ are mapped to a (but all the variables from the interior, $\mathfrak{t}_\mathsf{i}$, of $\mathfrak{t}$ are mapped to labelled nulls) then $\mathcal{I}_\mathcal{A} \models \mathsf{tw}_\mathfrak{t}(a,\dots,a)$.

Let $\Theta_\mathcal{T}^{\boldsymbol{q}}$ be the set of tree witnesses for $\boldsymbol{q}(\boldsymbol{x})$ and $\mathcal{T}$. Tree witnesses $\mathfrak{t}$ and $\mathfrak{t}'$ are said to be *conflicting* if $\boldsymbol{q}_\mathfrak{t} \cap \boldsymbol{q}_{\mathfrak{t}'} \neq \emptyset$ (in other words, the interior of one tree witness, say, $\mathfrak{t}$, contains a root or an interior variable of the other, $\mathfrak{t}'$, or the other way round, which makes it impossible to have both tree witnesses mapped into the anonymous part of $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ at the *same time*). A set $\Theta \subseteq \Theta_\mathcal{T}^{\boldsymbol{q}}$ of tree witnesses is said to be *independent* if any two distinct tree witnesses in Θ are non-conflicting. If Θ is independent then we can 'cut' the query $\boldsymbol{q}(\boldsymbol{x})$ into independent subqueries in the following way. Consider a homomorphism that, for each $\mathfrak{t} \in \Theta$, maps the subset $\boldsymbol{q}_\mathfrak{t}$ of $\boldsymbol{q}$ to the $\exists R.D$-generated $\mathcal{T}$-tree on some a (provided that $\mathfrak{t}$ is generated by $\exists R.D$) and maps the remaining atoms in $\boldsymbol{q}$ to the ABox part of $\mathcal{C}_{\mathcal{T},\mathcal{A}}$. By (11), such a homomorphism is possible if there is a tuple $\boldsymbol{a}$ in $\mathsf{ind}(\mathcal{A})$ such that the formula

$$\boldsymbol{q}_{\mathsf{cut}}^{\Theta}(\boldsymbol{x}) \;=\; \exists \boldsymbol{y} \Big((\boldsymbol{q} \setminus \boldsymbol{q}_\Theta) \;\wedge \bigwedge_{\mathfrak{t} \in \Theta} \mathsf{tw}_\mathfrak{t} \Big) \tag{17}$$

holds in $\mathcal{I}_\mathcal{A}$ on $\boldsymbol{a}$, where $\boldsymbol{q} \setminus \boldsymbol{q}_\Theta$ is the conjunction of all the atoms in $\boldsymbol{q}$ that do not belong to $\boldsymbol{q}_\mathfrak{t}$, for any $\mathfrak{t} \in \Theta$. (Recall that due to H-completeness of the ABox, $\mathcal{I}_\mathcal{A}^* = \mathcal{I}_\mathcal{A}$.) Conversely, if there is a homomorphism from $\boldsymbol{q}(\boldsymbol{a})$ to $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ then there exists an independent set Θ of tree witnesses such that $\mathcal{I}_\mathcal{A} \models \boldsymbol{q}_{\mathsf{cut}}^{\Theta}(\boldsymbol{a})$.

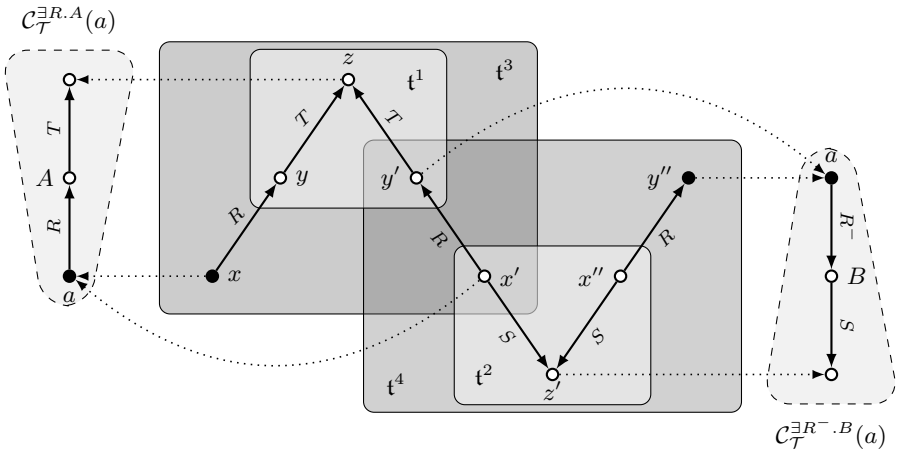

Fig. 10. Tree witnesses in Example 35.

The following PE-formula $\boldsymbol{q}_{\mathsf{tw}}(\boldsymbol{x})$ is called the *tree-witness rewriting of* $\boldsymbol{q}(\boldsymbol{x})$ *and* $\mathcal{T}$ *over H-complete ABoxes*:

$$\boldsymbol{q}_{\mathsf{tw}}(\boldsymbol{x}) \quad = \bigvee_{\Theta \subseteq \Theta_{\mathcal{T}}^{\boldsymbol{q}} \text{ independent}} \boldsymbol{q}_{\mathsf{cut}}^{\Theta}(\boldsymbol{x}). \tag{18}$$

Example 35. Consider a TBox with the following concept inclusions:

$$A_0 \sqsubseteq \exists R.A, \qquad A \sqsubseteq \exists T,$$
$$B_0 \sqsubseteq \exists R^-.B, \qquad B \sqsubseteq \exists S$$

and the following CQ

$$\boldsymbol{q}(x, y'') = \exists y, z, y', x', z', x'' \big(R(x,y) \land T(y,z) \land T(y',z) \land R(x',y') \land S(x',z') \land S(x'',z') \land R(x'',y'')\big)$$

shown in Fig. 10 alongside the $\exists R.A$-generated $\mathcal{T}$-tree and the $\exists R^-.B$-generated $\mathcal{T}$-tree. There are four tree witnesses for $\boldsymbol{q}(x, y'')$ and $\mathcal{T}$:

- $\mathfrak{t}^1 = (\mathfrak{t}^1_{\mathsf{r}}, \mathfrak{t}^1_{\mathsf{i}})$ generated by $\exists T$ with $\mathfrak{t}^1_{\mathsf{r}} = \{y, y'\}$ and $\mathfrak{t}^1_{\mathsf{i}} = \{z\}$ and
$$\boldsymbol{q}_{\mathfrak{t}^1} = \{\, T(y,z),\ T(y',z) \,\};$$
- $\mathfrak{t}^2 = (\mathfrak{t}^2_{\mathsf{r}}, \mathfrak{t}^2_{\mathsf{i}})$ generated by $\exists S$ with $\mathfrak{t}^2_{\mathsf{r}} = \{x', x''\}$ and $\mathfrak{t}^2_{\mathsf{i}} = \{z'\}$ and
$$\boldsymbol{q}_{\mathfrak{t}^2} = \{\, S(x',z'),\ S(x'',z') \,\};$$
- $\mathfrak{t}^3 = (\mathfrak{t}^3_{\mathsf{r}}, \mathfrak{t}^3_{\mathsf{i}})$ generated by $\exists R.A$ with $\mathfrak{t}^3_{\mathsf{r}} = \{x, x'\}$ and $\mathfrak{t}^3_{\mathsf{i}} = \{y, y', z\}$ and
$$\boldsymbol{q}_{\mathfrak{t}^3} = \{\, R(x,y),\ T(y,z),\ T(y',z),\ R(x',y') \,\};$$

– $\mathfrak{t}^4 = (\mathfrak{t}^4_{\mathsf{r}}, \mathfrak{t}^4_{\mathsf{i}})$ generated by $\exists R^-.B$ with $\mathfrak{t}^4_{\mathsf{r}} = \{y', y''\}$ and $\mathfrak{t}^4_{\mathsf{i}} = \{x', x'', z'\}$ and

$$\boldsymbol{q}_{\mathfrak{t}^4} = \{\, R(x', y'),\ S(x', z'),\ S(x'', z'),\ R(x'', y'')\,\}.$$

Clearly, $\mathfrak{t}^3$ and $\mathfrak{t}^4$ are conflicting; $\mathfrak{t}^3$ is also conflicting with $\mathfrak{t}^1$ but not with $\mathfrak{t}^2$ despite sharing a common root; symmetrically, $\mathfrak{t}^4$ is conflicting with $\mathfrak{t}^2$. Thus, we have the following 8 independent sets of tree witnesses:

$$\emptyset, \quad \{\mathfrak{t}^1\}, \quad \{\mathfrak{t}^2\}, \quad \{\mathfrak{t}^3\}, \quad \{\mathfrak{t}^4\}, \quad \{\mathfrak{t}^1, \mathfrak{t}^2\}, \quad \{\mathfrak{t}^1, \mathfrak{t}^4\}, \quad \{\mathfrak{t}^2, \mathfrak{t}^3\},$$

which result in a tree-witness rewriting of 8 subqueries with the following tree-witness formulas:

$$\begin{aligned}
\mathsf{tw}_{\mathfrak{t}^1}(y, y') &= \exists u \big((u = y) \land (u = y') \land A(u)\big),\\
\mathsf{tw}_{\mathfrak{t}^2}(x', x'') &= \exists u \big((u = x') \land (u = x'') \land B(u)\big),\\
\mathsf{tw}_{\mathfrak{t}^3}(x, x') &= \exists u \big((u = x) \land (u = x') \land A_0(u)\big),\\
\mathsf{tw}_{\mathfrak{t}^4}(y', y'') &= \exists u \big((u = y') \land (u = y'') \land B_0(u)\big).
\end{aligned}$$

Theorem 36 (Kikot et al. (2012b)). *Let $\mathcal{T}$ be a QL TBox and $\boldsymbol{q}(\boldsymbol{x})$ a CQ. For any ABox $\mathcal{A}$ that is H-complete with respect to $\mathcal{T}$ and any tuple $\boldsymbol{a}$ in $\mathsf{ind}(\mathcal{A})$, we have*

$$\mathcal{C}_{\mathcal{T},\mathcal{A}} \models \boldsymbol{q}(\boldsymbol{a}) \quad \text{if and only if} \quad \mathcal{I}_{\mathcal{A}} \models \boldsymbol{q}_{\mathsf{tw}}(\boldsymbol{a}).$$

By Proposition 33, to obtain a PE-rewriting of $\boldsymbol{q}(\boldsymbol{x})$ and $\mathcal{T}$ over *arbitrary* ABoxes, it is enough to take the tree-witness rewriting $\boldsymbol{q}_{\mathsf{tw}}(\boldsymbol{x})$ over H-complete ABoxes and replace every atom $S(\boldsymbol{z})$ in it with $S_{\mathsf{ext}}(\boldsymbol{z})$.

The number of tree witnesses, $|\Theta^{\boldsymbol{q}}_{\mathcal{T}}|$, is bounded by $3^{|\boldsymbol{q}|}$. On the other hand, there is a sequence of queries $\boldsymbol{q}_n$ and ontologies $\mathcal{T}_n$ with exponentially many (in $|\boldsymbol{q}_n|$) tree witnesses (Kikot et al., 2012b). The length of $\boldsymbol{q}_{\mathsf{tw}}$ is $O(2^{|\Theta^{\boldsymbol{q}}_{\mathcal{T}}|} \cdot |\boldsymbol{q}| \cdot |\mathcal{T}|)$. It is to be noted that there exist CQs $\boldsymbol{q}$ and *QL* TBoxes $\mathcal{T}$ any PE-rewritings of which are of exponential size in $|\boldsymbol{q}|$ provided that the rewritings use the same symbols as in $\boldsymbol{q}$ and $\mathcal{T}$ (Kikot et al., 2012a). One can always reduce the size of PE-rewritings to polynomial by employing two new constants that do not occur in $\boldsymbol{q}$; for details and further references, consult (Gottlob et al., 2014).

If any two tree-witnesses for $\boldsymbol{q}(\boldsymbol{x})$ and $\mathcal{T}$ are *compatible*—that is, they are either non-conflicting or one is included in the other—then $\boldsymbol{q}_{\mathsf{tw}}(\boldsymbol{x})$ can be equivalently transformed into the PE-rewriting

$$\boldsymbol{q}'_{\mathsf{tw}}(\boldsymbol{x}) \;=\; \exists \boldsymbol{y} \bigwedge_{S(\boldsymbol{z}) \in \boldsymbol{q}} \Big(S(\boldsymbol{z}) \;\lor \bigvee_{\mathfrak{t} \in \Theta^{\boldsymbol{q}}_{\mathcal{T}} \text{ with } S(\boldsymbol{z}) \in \boldsymbol{q}_{\mathfrak{t}}} \mathsf{tw}_{\mathfrak{t}} \Big)$$

of size $O(|\Theta^{\boldsymbol{q}}_{\mathcal{T}}| \cdot |\boldsymbol{q}|^2 \cdot |\mathcal{T}|)$; for details we refer the reader to (Kikot et al., 2012b).

As we saw in Example 29, CQ entailment over *RL* and *EL* knowledge bases is P-hard for data complexity, and so some CQs over such knowledge bases do not have first-order rewritings. In the next two sections, we show that one can always rewrite CQs over *RL* or *EL* TBoxes into datalog queries of polynomial size.

4.3 Datalog Rewriting for *RL*

Recall from Section 3.1 that a *datalog program* is a set of Horn clauses with one positive literal, that is, universally quantified sentences of the form

$$\gamma_0(\boldsymbol{x}) \leftarrow \gamma_1(\boldsymbol{x}) \land \dots \land \gamma_n(\boldsymbol{x}),$$

where each variable of the head, $\gamma_0(\boldsymbol{x})$, occurs in at least one of the atoms in the body, $\gamma_1(\boldsymbol{x}), \dots, \gamma_n(\boldsymbol{x})$. (Following the datalog tradition, we omit the universal quantifiers and write the implication from right to left.) Given a datalog program Π, a set of ground atoms D and a ground atom $Q(\boldsymbol{a})$, we write $(\Pi, D) \models Q(\boldsymbol{a})$ if $Q(\boldsymbol{a})$ is true in every interpretation satisfying D and all clauses of Π, or, equivalently, if $Q(\boldsymbol{a})$ is true in the minimal (or canonical) model of (Π, D), which is constructed in the same way as the canonical models of *RL* knowledge bases (in general, however, the arity of predicates is not bounded by 2).

We say that a CQ $\boldsymbol{q}(\boldsymbol{x})$ and a TBox $\mathcal{T}$ are *datalog-rewritable* if there exist a datalog program Π and a predicate $Q(\boldsymbol{x})$ such that, for any ABox $\mathcal{A}$ and any tuple $\boldsymbol{a}$ of individuals in $\mathcal{A}$, we have

$$(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q}(\boldsymbol{a}) \qquad \text{if and only if} \qquad (\Pi, \mathcal{A}) \models Q(\boldsymbol{a}). \tag{19}$$

(In the following two sections we view any ABox as a set of ground atoms: each concept assertion is a unary ground atom and each role assertion is a binary ground atom.) In this case, the pair $(\Pi, Q(\boldsymbol{x}))$ is called a *datalog rewriting of* $\boldsymbol{q}(\boldsymbol{x})$ *and* $\mathcal{T}$.

Let $\mathcal{T}$ be a *positive RL* TBox, that is, an *RL* TBox without negative concept inclusions of the form $B \sqsubseteq \bot$ (see Section 3.1). Denote by $\Pi_{\mathcal{T}}$ the datalog program that contains the standard translations of all concept and role inclusions in $\mathcal{T}$:

$$\begin{aligned} ST_u(A) &\leftarrow ST_u(B), && \text{for each } B \sqsubseteq A \text{ in } \mathcal{T}, \\ ST_{u,v}(R_2) &\leftarrow ST_{u,v}(R_1), && \text{for each } R_1 \sqsubseteq R_2 \text{ in } \mathcal{T}. \end{aligned}$$

(Note that B can be a complex *RL* concept, constructed using $\sqcap$ and $\exists R.C$ with possibly inverse roles.) By Theorem 15, we then obtain the following result:

Corollary 37. *For any positive RL TBox* $\mathcal{T}$ *and any CQ* $\boldsymbol{q}(\boldsymbol{x}) = \exists \boldsymbol{y}\, \varphi(\boldsymbol{x}, \boldsymbol{y})$,

$$(\Pi_{\mathcal{T}} \cup \{Q(\boldsymbol{x}) \leftarrow \varphi(\boldsymbol{x}, \boldsymbol{y})\},\ Q(\boldsymbol{x}))$$

is a datalog rewriting of $\boldsymbol{q}(\boldsymbol{x})$ *and* $\mathcal{T}$*, where* Q *is a fresh predicate symbol.*

Note that the size of this datalog rewriting is the sum of the sizes of the TBox and the query (thus, it is linear in both).

4.4 Tree-Witness Datalog Rewriting for *EL*

Let $\mathcal{T}$ be an *EL* TBox in normal form (Definition 19). As we saw in Section 3.2, the canonical model of $(\mathcal{T}, \mathcal{A})$ can be defined by unravelling the cycles in the

generating model $\mathcal{G}_{\mathcal{T},\mathcal{A}}$. So, the domain of the canonical model consists of $\rightsquigarrow_{\mathcal{T},\mathcal{A}}$-paths of the form $aw_{\exists P_1.D_1} \cdots w_{\exists P_n.D_n}$, $n \geq 0$, such that $a \in \mathsf{ind}(\mathcal{A})$ and, if $n > 0$, then

$$a \rightsquigarrow_{\mathcal{T},\mathcal{A}} w_{\exists P_1.D_1} \quad \text{and} \quad w_{\exists P_i.D_i} \rightsquigarrow_{\mathcal{T},\mathcal{A}} w_{\exists P_{i+1}.D_{i+1}}, \text{ for } i < n.$$

Similarly to the case of QL, we can represent the canonical model as the join of its ABox and anonymous parts. The latter consists of the trees $\mathcal{C}_{\mathcal{T}}^{\exists P.D}(a)$ defined in precisely the same way as in Section 4.2: $\mathcal{C}_{\mathcal{T}}^{\exists P.D}(a)$ is the restriction of the canonical model of $(\mathcal{T}, \{(\exists P.D)(a)\})$ to a and the labelled nulls with the prefix $aw_{\exists P.D}$. More formally, $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ is represented as

$$\mathcal{C}_{\mathcal{T},\mathcal{A}} \quad = \quad \mathcal{I}_{\mathcal{A}}^{*} \quad \oplus \quad \bigoplus_{\substack{a \in \mathsf{ind}(\mathcal{A}) \\ A \sqsubseteq \exists P.D \in \mathcal{T} \text{ with } \mathcal{I}_{\mathcal{A}}^{*} \models A(a)}} \mathcal{C}_{\mathcal{T}}^{\exists P.D}(a), \tag{20}$$

where $\mathcal{I}_{\mathcal{A}}^{*}$ is a model of $\mathcal{A}$ with domain $\mathsf{ind}(\mathcal{A})$. We again follow the divide and conquer strategy and split the process of query rewiring in two steps: rewriting over H-complete ABoxes and tree witnesses.

H-completeness Let $\mathcal{T}$ be an EL TBox. A simple ABox $\mathcal{A}$ is said to be *H-complete with respect to* $\mathcal{T}$ if, for all concept names A and role names P, we have

$$\begin{aligned} A(a) \in \mathcal{A} \quad &\text{if} \quad \mathcal{I}_{\mathcal{A}} \models C(a) \text{ and } \mathcal{T} \models C \sqsubseteq A, \text{ for some } EL \text{ concept } C, \\ P(a,b) \in \mathcal{A} \quad &\text{if} \quad \mathcal{I}_{\mathcal{A}} \models R(a,b) \text{ and } \mathcal{T} \models R \sqsubseteq P, \text{ for some role name } R. \end{aligned}$$

(Note that the definition is the same as for QL except that now C can be an arbitrary EL concept, not only A' or $\exists R$.) We say that $(\Pi, Q(\boldsymbol{x}))$, for a datalog program Π and an atom $Q(\boldsymbol{x})$, is a *datalog rewriting of* $\boldsymbol{q}(\boldsymbol{x})$ *and* $\mathcal{T}$ *over H-complete ABoxes* if (19) holds for any ABox $\mathcal{A}$ that is H-complete with respect to $\mathcal{T}$ and any tuple $\boldsymbol{a}$ from $\mathsf{ind}(\mathcal{A})$.

Observe that if an ABox $\mathcal{A}$ is H-complete with respect to $\mathcal{T}$ then the ABox part of $\mathcal{C}_{\mathcal{T},\mathcal{A}}$, that is, $\mathcal{I}_{\mathcal{A}}^{*}$ in (20), coincides with $\mathcal{I}_{\mathcal{A}}$. Thus, if $\mathcal{T}$ is a flat EL TBox in normal form (which does not contain concept inclusions of the form $A \sqsubseteq \exists P.D$) then

$$(\{Q(\boldsymbol{x}) \leftarrow \varphi(\boldsymbol{x}, \boldsymbol{y})\},\ Q(\boldsymbol{x}))$$

is clearly a datalog rewriting of $\boldsymbol{q}(\boldsymbol{x}) = \exists \boldsymbol{y}\, \varphi(\boldsymbol{x}, \boldsymbol{y})$ and $\mathcal{T}$ over H-complete ABoxes. Now, any such TBox $\mathcal{T}$ can also be seen as a positive RL TBox, and so, by Corollary 37, the datalog program $\Pi_{\mathcal{T}}$ defined in Section 4.3 describes precisely the conditions of H-completeness of $\mathcal{A}$:

$$\begin{aligned} A(a) \in \mathcal{A} \quad &\text{if and only if} \quad (\Pi_{\mathcal{T}}, \mathcal{A}) \models A(a), \\ P(a,b) \in \mathcal{A} \quad &\text{if and only if} \quad (\Pi_{\mathcal{T}}, \mathcal{A}) \models P(a,b). \end{aligned}$$

It follows then that we can easily obtain datalog rewritings (over arbitrary ABoxes) from datalog rewritings over H-complete ABoxes:

Proposition 38. *If $(\Pi, Q(\boldsymbol{x}))$ is a datalog rewriting of $\boldsymbol{q}(\boldsymbol{x})$ and $\mathcal{T}$ over H-complete ABoxes, then $(\Pi \cup \Pi_{\mathcal{T}}, Q(\boldsymbol{x}))$ is a datalog rewriting of $\boldsymbol{q}(\boldsymbol{x})$ and $\mathcal{T}$ over arbitrary ABoxes.*

Thus, we can only concentrate on constructing datalog rewritings over H-complete ABoxes.

Tree Witnesses Let $\mathcal{T}$ be an *EL* TBox in normal form and let $\boldsymbol{q}(\boldsymbol{x})$ be a CQ. Similarly to the case of *QL*, to construct a rewriting of $\boldsymbol{q}(\boldsymbol{x})$ and $\mathcal{T}$, we need to consider all possible subsets of $\boldsymbol{q}$, the atoms of the query, that can be homomorphically mapped to the ABox part of the canonical model $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ and subsets of $\boldsymbol{q}$ that can be homomorphically mapped to the anonymous part of $\mathcal{C}_{\mathcal{T},\mathcal{A}}$. Tree witnesses for $\boldsymbol{q}(\boldsymbol{x})$ and $\mathcal{T}$ are defined in absolutely the same way as in Section 4.2. A major difference, however, is that *EL* does not contain inverse roles, and so each $\mathcal{T}$-tree in the anonymous part is generated by a concept of the form $\exists P.D$, where P is a role name. It follows that if a variable, say y, of any CQ belongs to the interior of some tree witness and the CQ contains an atom of the form $P(y, y')$, then y' must also be in the interior of the same tree witness. In particular, we have the following:

Proposition 39. *Let $\mathcal{T}$ be an EL TBox and $\boldsymbol{q}(\boldsymbol{x})$ a CQ. Then, for any binary atom $P(z_1, z_2)$ in $\boldsymbol{q}$, where P is a role name,*

- *there is no tree witness for $\boldsymbol{q}(\boldsymbol{x})$ and $\mathcal{T}$ with $P(z_1, z_2) \in \boldsymbol{q}_{\mathfrak{t}}$, $z_2 \in \mathfrak{t}_{\mathsf{r}}$, $z_1 \in \mathfrak{t}_{\mathsf{i}}$;*
- *there is at most one tree witness $\mathfrak{t}$ for $\boldsymbol{q}(\boldsymbol{x})$ and $\mathcal{T}$ such that $P(z_1, z_2) \in \boldsymbol{q}_{\mathfrak{t}}$, $z_1 \in \mathfrak{t}_{\mathsf{r}}$ and $z_2 \in \mathfrak{t}_{\mathsf{i}}$.*

These observations suggest a simple algorithm for constructing all tree witnesses for any given CQ and *EL* TBox, which we first illustrate by a concrete example.

Example 40. Let $\mathcal{T}$ be an *EL* TBox containing the following concept and role inclusions:

$$A \sqsubseteq \exists S.B, \qquad B \sqsubseteq \exists R.C, \qquad R \sqsubseteq T.$$

Consider the CQ

$$\boldsymbol{q}(x) = \exists y, z, u, v \, \big(S(x, y) \land T(y, z) \land R(u, z) \land T(u, v) \land C(v)\big).$$

The algorithms begins with the smallest tree witnesses. By **(tw2)**, each tree witness $\mathfrak{t}$ is uniquely defined by its interior (a non-empty set of existentially quantified variables): the roots are all the terms that do not belong to $\mathfrak{t}_{\mathsf{i}}$ but occur in atoms with an argument in $\mathfrak{t}_{\mathsf{i}}$. The smallest tree witnesses are thus induced by singleton sets $\mathfrak{t}_{\mathsf{i}}$. For $\mathfrak{t}_{\mathsf{i}} = \{v\}$, we have $\mathfrak{t}_{\mathsf{r}} = \{u\}$, which gives the tree witness $\mathfrak{t}^1$ with

$$\boldsymbol{q}_{\mathfrak{t}^1} = \{\, T(u, v),\ C(v) \,\},$$

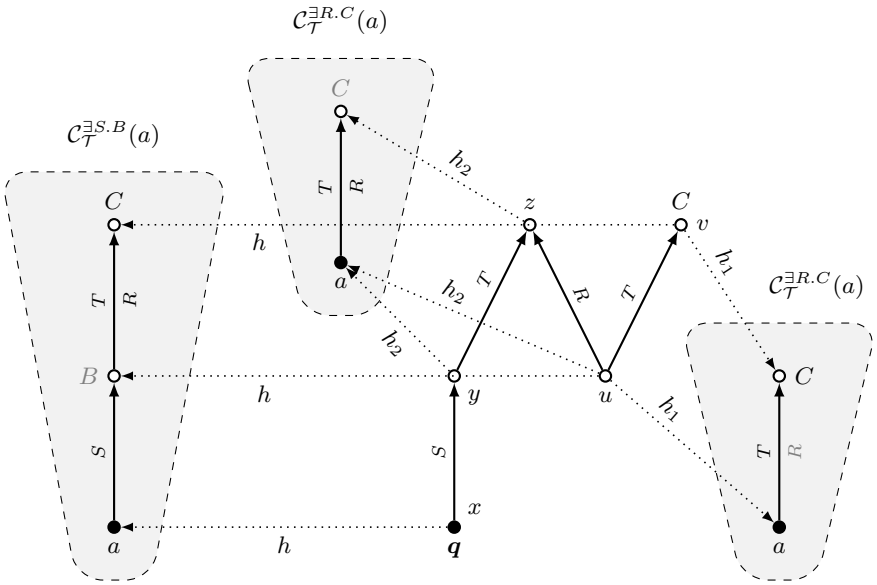

Fig. 11. Tree witnesses in Example 40.

which is generated by $\exists R.C$; see h_1 in Fig. 11. For $\mathfrak{t}_\mathsf{i} = \{u\}$, we have $\mathfrak{t}_\mathsf{r} = \{v, z\}$ but, since *EL* contains no inverse roles, there can be no homomorphism h that maps the set of atoms $\{\, T(y,z),\ R(u,z)\,\}$ into a $\mathcal{T}$-tree in such a way that h takes $\mathfrak{t}_\mathsf{r}$ to an ABox individual and $\mathfrak{t}_\mathsf{i}$ to a labelled null. The same argument applies to $\mathfrak{t}_\mathsf{i} = \{y\}$. However, $\mathfrak{t}_\mathsf{i}^2 = \{z\}$ gives rise to another tree witness, $\mathfrak{t}^2$, with

$$\boldsymbol{q}_{\mathfrak{t}^2} = \{\, T(y,z),\ R(u,z)\,\},$$

which is again generated by $\exists R.C$; see h_2 in Fig. 11 (note that both variables in $\mathfrak{t}_\mathsf{r}^2 = \{y, u\}$ are mapped to a, the root of the $\exists R.C$-generated $\mathcal{T}$-tree $\mathcal{C}_{\mathcal{T}}^{\exists R.C}(a)$). Thus, we have considered all singleton subsets of the existentially quantified variables and constructed all possible tree witnesses of *depth 1*. Next, we observe that larger tree witnesses must contain smaller tree witnesses in their interior. So, suppose that $\mathfrak{t}_\mathsf{i}$ contains both the roots and the interior of tree witness $\mathfrak{t}^2$, that is, $\{y, z, u\} \subseteq \mathfrak{t}_\mathsf{i}$. Then, since *EL* has no inverse roles, $\mathfrak{t}_\mathsf{i}$ must also contain v (and thus, the whole of tree witness $\mathfrak{t}^1$). In this way, we obtain a tree witness $\mathfrak{t}$ of *depth 2* with $\mathfrak{t}_\mathsf{i} = \{y, u, z, v\}$, $\mathfrak{t}_\mathsf{r} = \{x\}$ and

$$\boldsymbol{q}_{\mathfrak{t}} = \{\, S(x,y),\ T(y,z),\ R(u,z),\ T(u,v),\ C(v)\,\},$$

which is generated by $\exists S.B$; see h in Fig. 11. We have covered all possible subsets of the existentially quantified variables and in this way obtained all tree witnesses for $\boldsymbol{q}(x)$ and $\mathcal{T}$.

A general algorithm constructing all tree witnesses for any given CQ $\boldsymbol{q}(\boldsymbol{x})$ and *EL* TBox $\mathcal{T}$ in normal form works as follows. It begins by identifying tree

witnesses that have a single interior variable (these are tree witnesses of depth 1). The algorithm then considers each set of tree witnesses $\mathfrak{t}^1, \dots, \mathfrak{t}^k$ sharing common roots (provided that all their roots are existentially quantified variables)—the roots and interiors will become the interior of a potential new, larger, tree witness. In order to satisfy **(tw2)**, the algorithm extends the set of atoms in $\boldsymbol{q}_0 = \boldsymbol{q}_{\mathfrak{t}^1} \cup \dots \cup \boldsymbol{q}_{\mathfrak{t}^k}$ by all atoms incident on the variables in $\boldsymbol{q}_0$ and then, to satisfy **(tw1)**, checks whether this extended set of atoms can be homomorphically mapped to a $\mathcal{T}$-tree. Since EL contains no inverse roles, any two tree witnesses $\mathfrak{t}^i$ and $\mathfrak{t}^j$ that share a common root must become part of the *interior* of a larger tree witness (if it exists at all) and therefore, the number of sets of tree witnesses that need consideration is bounded by the number of (existentially quantified) variables in $\boldsymbol{q}$. Therefore, this algorithm constructs all tree witnesses and runs in polynomial time in the size of $\boldsymbol{q}$ and $\mathcal{T}$. (We note in passing that the presented algorithm resembles the construction of the equivalence relation $\sim_q$ defined by Lutz et al. (2009) except that the equivalence relation does not take account of the TBox and the distinction between answer variables and existentially quantified variables.)

It follows that the number of tree witnesses for any $\boldsymbol{q}(\boldsymbol{x})$ and $\mathcal{T}$ does not exceed the number of atoms in $\boldsymbol{q}$. Moreover, each pair, $\mathfrak{t}$ and $\mathfrak{t}'$, of tree witnesses for $\boldsymbol{q}(\boldsymbol{x})$ and $\mathcal{T}$ is compatible, that is,

$$\text{either} \quad \boldsymbol{q}_{\mathfrak{t}} \cap \boldsymbol{q}_{\mathfrak{t}'} = \emptyset \quad \text{or} \quad \boldsymbol{q}_{\mathfrak{t}} \subseteq \boldsymbol{q}_{\mathfrak{t}'} \quad \text{or} \quad \boldsymbol{q}_{\mathfrak{t}} \supseteq \boldsymbol{q}_{\mathfrak{t}'}.$$

(In other words, the tree witnesses are partially ordered by the $\subseteq$ relation on their sets of atoms.) It follows that we can use the 'modified' tree-witness rewriting $\boldsymbol{q}'_{\mathsf{tw}}(\boldsymbol{x})$ over H-complete ABoxes defined at the end of Section 4.2, or rather its datalog representation. Let $Q_{S(\boldsymbol{z})}$, for each atom $S(\boldsymbol{z})$ in $\boldsymbol{q}$, and D be fresh k-ary predicate symbols, for $k = |\boldsymbol{x}| + |\boldsymbol{y}|$, and let D_1 be a fresh unary predicate symbol. Intuitively, the $Q_{S(\boldsymbol{z})}$ are the rewritings of individual atoms of the query, the interpretation of D_1 consists of individuals from the ABox that are relevant to the query and the interpretation of D of all tuples of such individuals. Formally, let $\Omega^{\boldsymbol{q}}_{\mathcal{T}}$ comprise the rule

$$Q(\boldsymbol{x}) \leftarrow \bigwedge_{S(\boldsymbol{z}) \in \boldsymbol{q}} Q_{S(\boldsymbol{z})}(\boldsymbol{x}, \boldsymbol{y}),$$

the following rules, for each atom $S(\boldsymbol{z})$ in $\boldsymbol{q}$:

$$\begin{aligned} &Q_{S(\boldsymbol{z})}(\boldsymbol{x}, \boldsymbol{y}) \leftarrow D(\boldsymbol{x}, \boldsymbol{y}) \wedge S(\boldsymbol{z}), \\ &Q_{S(\boldsymbol{z})}(\boldsymbol{x}, \boldsymbol{y}) \leftarrow D(\boldsymbol{x}, \boldsymbol{y}) \wedge \mathsf{tw}_{\mathfrak{t}}(\mathfrak{t}_{\mathsf{r}}), \qquad && \text{for } \mathfrak{t} \in \Theta^{\boldsymbol{q}}_{\mathcal{T}} \text{ with } S(\boldsymbol{z}) \in \boldsymbol{q}_{\mathfrak{t}}, \end{aligned}$$

and the following rules defining D and D_1:

$$\begin{aligned} &D(z_1, \dots, z_k) \leftarrow D_1(z_1) \wedge \dots D_1(z_k), \\ &\qquad D_1(z) \leftarrow A(z), && \text{for a concept name } A \text{ in } \boldsymbol{q} \text{ or } \mathcal{T}, \\ &\qquad D_1(z) \leftarrow P(z, y), && \text{for a role name } P \text{ in } \boldsymbol{q} \text{ or } \mathcal{T}, \\ &\qquad D_1(z) \leftarrow P(y, z), && \text{for a role name } P \text{ in } \boldsymbol{q} \text{ or } \mathcal{T}. \end{aligned}$$

(Strictly speaking, D_1 is not the same as the set of individuals in the ABox because concept and role names that occur in the ABox but do not occur in the query or the TBox are not included in the definition above: any individual that belongs only to such a concept or role is simply not visible to the query.) Thus we obtain a datalog rewriting $(\Omega^{\boldsymbol{q}}_{\mathcal{T}}, Q(\boldsymbol{x}))$ of $\boldsymbol{q}(\boldsymbol{x})$ and $\mathcal{T}$ over H-complete ABoxes:

Theorem 41. *Let $\mathcal{T}$ be an EL TBox and $\boldsymbol{q}(\boldsymbol{x})$ a CQ. For any ABox $\mathcal{A}$ that is H-complete with respect to $\mathcal{T}$ and any tuple $\boldsymbol{a}$ in $\mathsf{ind}(\mathcal{A})$, we have*

$$\mathcal{C}_{\mathcal{T},\mathcal{A}} \models \boldsymbol{q}(\boldsymbol{a}) \quad \text{if and only if} \quad (\Omega^{\boldsymbol{q}}_{\mathcal{T}}, \mathcal{I}_{\mathcal{A}}) \models Q(\boldsymbol{a}).$$

Finally, since the number of tree witnesses is linear in the size of $\boldsymbol{q}$, this datalog rewriting is of polynomial size in the size of $\boldsymbol{q}$ and $\mathcal{T}$. We also note in passing that the datalog rewriting over H-complete ABoxes is nonrecursive (none of the predicates is defined, even indirectly, in terms of itself) and the only recursive component of the rewriting over arbitrary ABoxes is the rules ensuring H-completeness of the ABox.

5 OBDA with Ontop and Databases

In the final section of this chapter, we briefly describe the ontology-based data access (OBDA) system *Ontop*[16] (Rodríguez-Muro et al., 2013) implemented at the Free University of Bozen-Bolzano and available as a plugin for the ontology editor Protégé 4, a SPARQL end-point and OWLAPI and Sesame libraries. *Ontop* is the first system to support the W3C recommendations OWL 2 QL, R2RML, SPARQL and the OWL 2 QL direct semantics entailment regime.

We illustrate how *Ontop* works using an example from (Rodríguez-Muro et al., 2013), which involves a (simplified) database IMDb[17] (in a typical OBDA scenario data comes from a relational database rather than an ABox). From a logical point of view, a *database schema* (Abiteboul et al., 1995) contains predicate symbols (with their arity) for both stored database relations (also known as tables) and views (with their definitions in terms of stored relations) as well as a set Σ of *integrity constraints* (in the form of functional and inclusion dependencies; for example, primary and foreign keys): any instance $\boldsymbol{I}$ of the database schema must satisfy its integrity constraints Σ.

The schema of IMDb contains relations $\mathsf{title}[m, t, y]$ with information about movies (ID, title, production year), and $\mathsf{castinfo}[p, m, r]$ with information about movies' cast (person ID, movie ID, person role). Thus, a data instance $\boldsymbol{I}_{\text{IMDb}}$ of this schema may contain the tables

title

m	t	y
728	'Django Unchained'	2012

castinfo

p	m	r
n37	728	1
n38	728	1

[16] `ontop.inf.unibz.it`

[17] `www.imdb.com/interfaces`

The integrity constraints Σ_{IMDb} of IMDb include the following foreign key (an inclusion dependency):

$$\forall m \left(\exists p, r\, \mathsf{castinfo}(p, m, r) \to \exists t, y\, \mathsf{title}(m, t, y)\right) \tag{21}$$

('each tuple in castinfo must refer to an existing title') and the following primary key (a functional dependency with determinant m):

$$\forall m \forall t_1 \forall t_2 \left(\exists y\, \mathsf{title}(m, t_1, y) \land \exists y\, \mathsf{title}(m, t_2, y) \to (t_1 = t_2)\right), \tag{22}$$

$$\forall m \forall y_1 \forall y_2 \left(\exists t\, \mathsf{title}(m, t, y_1) \land \exists t\, \mathsf{title}(m, t, y_2) \to (y_1 = y_2)\right) \tag{23}$$

('each title is uniquely determined by its ID m').

In the framework of OBDA, the users are not supposed to know the structure of the database. Instead, they are given an ontology, e.g., MO[18], which describes the application domain in terms of concepts and roles. In our example we have concepts mo:Movie and mo:Person, and roles mo:cast, mo:title and mo:year related by OWL 2 QL TBox $\mathcal{T}_{\text{MO}}$ containing, in particular, the following inclusions:

$$\begin{aligned} \mathsf{mo{:}Movie} &\sqsubseteq \exists \mathsf{mo{:}title}, & \mathsf{mo{:}Movie} &\sqsubseteq \exists \mathsf{mo{:}cast}, \\ \mathsf{mo{:}Movie} &\sqsubseteq \exists \mathsf{mo{:}year}, & \exists \mathsf{mo{:}title} &\sqsubseteq \mathsf{mo{:}Movie}, \\ \exists \mathsf{mo{:}cast} &\sqsubseteq \mathsf{mo{:}Movie}, & \exists \mathsf{mo{:}cast}^{-} &\sqsubseteq \mathsf{mo{:}Person}. \end{aligned}$$

The vocabularies of the database schema and the given OWL 2 QL TBox are linked together by means of mappings produced by a domain expert or extracted (semi)automatically. There are different known types of mappings: LAV (local-as-views), GAV (global-as-views), GLAV, etc.; consult, e.g., (Lenzerini, 2002) for an overview. Here we concentrate on GAV mappings because they guarantee low complexity of query answering (in what follows we call them simply mappings)—*Ontop* uses R2RML[19] to specify them. A *mapping*, $\mathcal{M}$, is a set of rules of the form

$$S(\boldsymbol{x}) \leftarrow \varphi(\boldsymbol{x}, \boldsymbol{z}),$$

where S is a concept or role name in the ontology and $\varphi(\boldsymbol{x}, \boldsymbol{z})$ is a conjunction of atoms with database relations (both stored and views) and a *filter*, that is, a Boolean combination of built-in predicates such as $=$ and $<$. (Note that, by including views in the schema, we can express any SQL query in mappings, which is important from the practical point of view.) In our example, a mapping $\mathcal{M}_{\text{MO}}$ that relates the terms of MO to the schema of IMDb contains the following rules:

$$\mathsf{mo{:}Movie}(m) \leftarrow \mathsf{title}(m, t, y), \tag{24}$$

$$\mathsf{mo{:}title}(m, t) \leftarrow \mathsf{title}(m, t, y), \tag{25}$$

$$\mathsf{mo{:}year}(m, y) \leftarrow \mathsf{title}(m, t, y), \tag{26}$$

$$\mathsf{mo{:}cast}(m, p) \leftarrow \mathsf{castinfo}(p, m, r), \tag{27}$$

$$\mathsf{mo{:}Person}(p) \leftarrow \mathsf{castinfo}(p, m, r). \tag{28}$$

[18] www.movieontology.org

[19] http://www.w3.org/TR/r2rml/

Given a mapping $\mathcal{M}$ from a database schema to an OWL 2 QL TBox $\mathcal{T}$ and an instance $\boldsymbol{I}$ of this schema, the ground atoms

$$S(\boldsymbol{a}), \quad \text{for } S(\boldsymbol{x}) \leftarrow \varphi(\boldsymbol{x}, \boldsymbol{z}) \text{ in } \mathcal{M} \text{ and } \boldsymbol{I} \models \exists \boldsymbol{z}\, \varphi(\boldsymbol{a}, \boldsymbol{z}),$$

comprise the ABox denoted by $\mathcal{A}_{\boldsymbol{I},\mathcal{M}}$ and called the *virtual ABox* for $\mathcal{M}$ over $\boldsymbol{I}$ (Rodríguez-Muro and Calvanese, 2011). This ABox is just a convenient presentational tool and does not have to be materialised by the system. Then the virtual ABox for $\mathcal{M}_{\text{MO}}$ over $\boldsymbol{I}_{\text{IMDb}}$ consists of the ground atoms

$$\mathsf{mo{:}Movie}(\mathsf{728}),\ \mathsf{mo{:}title}(\mathsf{728},\ \text{'Django Unchained'}),\ \mathsf{mo{:}year}(\mathsf{728},\ \mathsf{2012}),$$
$$\mathsf{mo{:}Person}(\mathsf{n37}),\ \mathsf{mo{:}cast}(\mathsf{728},\ \mathsf{n37}),$$
$$\mathsf{mo{:}Person}(\mathsf{n38}),\ \mathsf{mo{:}cast}(\mathsf{728},\ \mathsf{n38}).$$

The *certain answers* to a CQ $\boldsymbol{q}(\boldsymbol{x})$ over $\mathcal{T}$ linked by $\mathcal{M}$ to $\boldsymbol{I}$ are defined as the certain answers to $\boldsymbol{q}(\boldsymbol{x})$ over $(\mathcal{T}, \mathcal{A}_{\boldsymbol{I},\mathcal{M}})$. In order to find certain answers, one could first construct a PE-rewriting $\boldsymbol{q}'(\boldsymbol{x})$ of $\boldsymbol{q}(\boldsymbol{x})$ and $\mathcal{T}$ over *arbitrary* ABoxes (in the sequel, it will convenient to represent such a rewriting as a non-recursive datalog program). The rewriting $\boldsymbol{q}'(\boldsymbol{x})$ could then be *unfolded* into an SQL query using the so-called *partial evaluation* (Lloyd and Shepherdson, 1991), which exhaustively applies SLD-resolution to $\boldsymbol{q}'(\boldsymbol{x})$ and the mapping $\mathcal{M}$ and returns those rules whose bodies contain only database atoms. Consider, for example, CQ $\boldsymbol{q}(x) = \mathsf{mo{:}Movie}(x)$. An obvious rewiring of $\boldsymbol{q}(x)$ and the TBox $\mathcal{T}_{\text{MO}}$ (over arbitrary ABoxes) contains the following three rules:

$$\boldsymbol{q}'(x) \leftarrow \mathsf{mo{:}Movie}(x), \tag{29}$$
$$\boldsymbol{q}'(x) \leftarrow \mathsf{mo{:}title}(x, y), \tag{30}$$
$$\boldsymbol{q}'(x) \leftarrow \mathsf{mo{:}cast}(x, y). \tag{31}$$

The unfolding applies the SLD-resolution procedure to these three rules and the mapping $\mathcal{M}_{\text{MO}}$ and produces the rules

$$\boldsymbol{q}'(x) \leftarrow \mathsf{title}(x, t, y), \tag{29+24}$$
$$\boldsymbol{q}'(x) \leftarrow \mathsf{title}(x, t, y), \tag{30+25}$$
$$\boldsymbol{q}'(x) \leftarrow \mathsf{castinfo}(p, x, r). \tag{31+27}$$

The resulting union of Select-Project-Join queries could then be forwarded for execution to a relational database management system (RDBMS).

The same result can be achieved by using the tree-witness rewriting $\boldsymbol{q}_{\mathsf{tw}}(\boldsymbol{x})$ of $\boldsymbol{q}(\boldsymbol{x})$ and $\mathcal{T}$ over H-complete ABoxes introduced in Section 4.2. An obvious way to construct H-complete ABoxes is to take the *composition* of $\mathcal{M}$ and the inclusions in $\mathcal{T}$, that is, a mapping $\mathcal{M}^{\mathcal{T}}$ given by

$$\begin{aligned}
A(x) &\leftarrow \varphi(x, \boldsymbol{z}), && \text{if } A'(x) \leftarrow \varphi(x, \boldsymbol{z}) \in \mathcal{M} \text{ and } \mathcal{T} \models A' \sqsubseteq A,\\
A(x) &\leftarrow \varphi(x, y, \boldsymbol{z}), && \text{if } R(x, y) \leftarrow \varphi(x, y, \boldsymbol{z}) \in \mathcal{M} \text{ and } \mathcal{T} \models \exists R \sqsubseteq A,\\
P(x, y) &\leftarrow \varphi(x, y, \boldsymbol{z}), && \text{if } R(x, y) \leftarrow \varphi(x, y, \boldsymbol{z}) \in \mathcal{M} \text{ and } \mathcal{T} \models R \sqsubseteq P.
\end{aligned}$$

(Recall that we do not distinguish between $P^-(y,x)$ and $P(x,y)$.) Thus, for any $\boldsymbol{I}$ and any tuple $\boldsymbol{a}$ of individuals in $\mathcal{A}_{\boldsymbol{I},\mathcal{M}}$, we have:

$$(\mathcal{T}, \mathcal{A}_{\boldsymbol{I},\mathcal{M}}) \models \boldsymbol{q}(\boldsymbol{a}) \quad \text{if and only if} \quad \mathcal{A}_{\boldsymbol{I},\mathcal{M}^\mathcal{T}} \models \boldsymbol{q}_{\mathsf{tw}}(\boldsymbol{a}). \tag{32}$$

So, to compute the answers to $\boldsymbol{q}(\boldsymbol{x})$ over $\mathcal{T}$ linked by $\mathcal{M}$ to $\boldsymbol{I}$, one can unfold the tree-witness rewriting $\boldsymbol{q}_{\mathsf{tw}}(\boldsymbol{x})$ over H-complete ABoxes with the help of the composition $\mathcal{M}^\mathcal{T}$. However, the resulting query will produce duplicating answers if the ontology axioms express the same properties of the application domain as the integrity constraints of the database. For example, the IMDb schema Σ_{IMDb} contains foreign key (21): movie ID in castinfo references movie ID in title, and therefore the unfolded rewriting above will return the same movie many times—once from title and once for each of the cast members of the movie in castinfo. Such a duplication is clearly an undesirable feature of this straightforward approach.

For this reason, before applying $\mathcal{M}^\mathcal{T}$ to unfold the tree-witness rewriting, *Ontop* optimises the mapping using the database integrity constraints Σ. This allows *Ontop* to reduce redundancy in answers and substantially shorten the SQL queries, which makes the OBDA system more efficient.

5.1 $\mathcal{T}$-mappings

We say that a mapping $\mathcal{M}$ is a *$\mathcal{T}$-mapping over dependencies Σ* if the ABox $\mathcal{A}_{\boldsymbol{I},\mathcal{M}}$ is H-complete with respect to $\mathcal{T}$, for any data instance $\boldsymbol{I}$ satisfying Σ. The composition $\mathcal{M}^\mathcal{T}$ defined above is trivially a $\mathcal{T}$-mapping over any Σ. *Ontop* starts with $\mathcal{M}^\mathcal{T}$ and then applies a series of optimisations to construct a simpler $\mathcal{T}$-mapping.

Inclusion Dependencies. Suppose $\mathcal{M} \cup \{S(\boldsymbol{x}) \leftarrow \psi_1(\boldsymbol{x},\boldsymbol{z})\}$ is a $\mathcal{T}$-mapping over Σ. If there is a more specific rule than $S(\boldsymbol{x}) \leftarrow \psi_1(\boldsymbol{x},\boldsymbol{z})$ in $\mathcal{M}$, then $\mathcal{M}$ itself will also be a $\mathcal{T}$-mapping. To discover such 'more specific' rules, we run the standard query containment check (Abiteboul et al., 1995) taking account of the inclusion dependencies. For example, since $\mathcal{T}_{\text{MO}} \models \exists \mathsf{mo{:}cast} \sqsubseteq \mathsf{mo{:}Movie}$ in our running example, the composition of mapping $\mathcal{M}_{\text{MO}}$ and ontology $\mathcal{T}_{\text{MO}}$ contains the following rules for mo:Movie:

$$\begin{aligned} \mathsf{mo{:}Movie}(m) &\leftarrow \mathsf{title}(m,t,y), \\ \mathsf{mo{:}Movie}(m) &\leftarrow \mathsf{castinfo}(p,m,r). \end{aligned}$$

As we observed above, the latter rule is redundant because Σ_{IMDb} contains inclusion dependency (21), which is repeated here for reference:

$$\forall m \left(\exists p, r\, \mathsf{castinfo}(p,m,r) \rightarrow \exists t, y\, \mathsf{title}(m,t,y)\right).$$

concept	index	interval
mo:Actor	1	[1,1]
mo:Artist	2	[1,2]
mo:Director	3	[3,3]
mo:Person	4	[1,4]

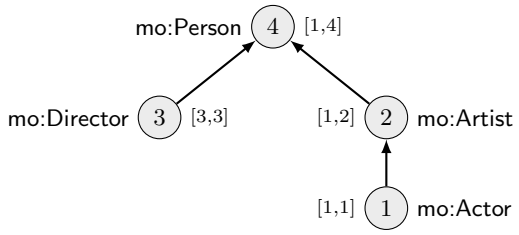

Fig. 12. Semantic Index example.

Disjunctions in SQL. Another way to reduce the size of a $\mathcal{T}$-mapping is to identify pairs of rules whose bodies are equivalent up to *filters with respect to constant values*. This optimisation deals with the rules introduced due to the so-called type (discriminating) attributes (Elmasri and Navathe, 2010) in database schemas. For example, the mapping $\mathcal{M}_{\text{MO}}$ contains six rules for sub-concepts of mo:Person:

$$\begin{aligned} \mathsf{mo{:}Actor}(p) &\leftarrow \mathsf{castinfo}(c,p,m,r),(r=1),\\ \mathsf{mo{:}Actress}(p) &\leftarrow \mathsf{castinfo}(c,p,m,r),(v=2),\\ &\cdots\\ \mathsf{mo{:}Editor}(p) &\leftarrow \mathsf{castinfo}(c,p,m,r),(r=6). \end{aligned}$$

Thus, the composition of $\mathcal{M}_{\text{MO}}$ and $\mathcal{T}_{\text{MO}}$ contains six rules for mo:Person that differ only in the last condition $(r = k)$, $1 \leq k \leq 6$. These can be reduced to a single rule:

$$\mathsf{mo{:}Person}(p) \leftarrow \mathsf{castinfo}(c,p,m,r),\ \big((r=1)\vee\cdots\vee(r=6)\big).$$

Note that such disjunctions lend themselves to efficient evaluation by RDBMSs.

Materialised ABoxes and Semantic Index. In addition to working with proper relational data sources, *Ontop* also supports ABox storage in the form of structureless *universal tables*: a binary relation $\mathsf{CA}[id, \textit{concept-id}]$ and a ternary relation $\mathsf{RA}[id_1, id_2, \textit{role-id}]$ represent concept and role assertions. The universal tables give rise to trivial mappings, and *Ontop* implements a technique, the *semantic index* (Rodríguez-Muro and Calvanese, 2011), that takes advantage of SQL features in $\mathcal{T}$-mappings for this scenario. The key observation is that, since the IDs in the universal tables CA and RA can be chosen by the system, each concept and role in the TBox $\mathcal{T}$ can be assigned a numeric *index* and a set of numerical *intervals* in such a way that the resulting $\mathcal{T}$-mapping contains simple SQL queries with interval filter conditions. For example, in $\mathcal{T}_{\text{MO}}$, we have

$$\begin{aligned} \mathsf{mo{:}Actor} &\sqsubseteq \mathsf{mo{:}Artist},\\ \mathsf{mo{:}Artist} &\sqsubseteq \mathsf{mo{:}Person},\\ \mathsf{mo{:}Director} &\sqsubseteq \mathsf{mo{:}Person}, \end{aligned}$$

so we can choose indexes and intervals for these concepts as in Fig. 12. It can be seen that these intervals respect the concept inclusions of the TBox: for instance, [1,1] for mo:Actor is a subset of [1,2] for mo:Artist. This will generate a $\mathcal{T}$-mapping with

$$\begin{aligned}
\mathsf{mo{:}Actor}(p) &\leftarrow \mathsf{CA}(p, \textit{concept-id}), \quad (\textit{concept-id} = 1),\\
\mathsf{mo{:}Artist}(p) &\leftarrow \mathsf{CA}(p, \textit{concept-id}), \quad (1 \leq \textit{concept-id} \leq 2),\\
\mathsf{mo{:}Director}(p) &\leftarrow \mathsf{CA}(p, \textit{concept-id}), \quad (\textit{concept-id} = 3),\\
\mathsf{mo{:}Person}(p) &\leftarrow \mathsf{CA}(p, \textit{concept-id}), \quad (1 \leq \textit{concept-id} \leq 4).
\end{aligned}$$

Thus, by choosing appropriate concept and role IDs, we effectively construct H-complete ABoxes *without* the expensive forward chaining procedure (and the need to store large amounts of the derived assertions and the extra complications with updating the data). On the other hand, the semantic index $\mathcal{T}$-mappings are based on range expressions that can be evaluated efficiently by RDBMSs using standard B-tree indexes (Elmasri and Navathe, 2010).

5.2 Unfolding with Semantic Query Optimisation (SQO)

Ontop applies the Semantic Query Optimisation (Chakravarthy et al., 1986) to rules obtained at the intermediate steps of unfolding. In particular, this eliminates redundant JOIN operations caused by reification of database relations by means of concepts and roles. Consider, for example, the CQ

$$\boldsymbol{q}(t, y) \leftarrow \mathsf{mo{:}Movie}(m),\ \mathsf{mo{:}title}(m, t),\ \mathsf{mo{:}year}(m, y),\ (y > 2010).$$

It has no tree witnesses, and so $\boldsymbol{q}_{\mathsf{tw}}(t, y) = \boldsymbol{q}(t, y)$. By straightforwardly applying the unfolding to $\boldsymbol{q}_{\mathsf{tw}}(t, y)$ and the $\mathcal{T}$-mapping $\mathcal{M}_{\mathrm{MO}}$, we obtain the query

$$\boldsymbol{q}'_{\mathsf{tw}}(t, y) \leftarrow \mathsf{title}(m, t_0, y_0),\ \mathsf{title}(m, t, y_1),\ \mathsf{title}(m, t_2, y),\ (y > 2010),$$

which requires two (potentially) expensive JOIN operations. However, by using functional dependencies (22) and (23) for the primary key m of title, which are repeated below:

$$\begin{aligned}
&\forall m \forall t_1 \forall t_2 \left(\exists y\, \mathsf{title}(m, t_1, y) \land \exists y\, \mathsf{title}(m, t_2, y) \rightarrow (t_1 = t_2)\right),\\
&\forall m \forall y_1 \forall y_2 \left(\exists t\, \mathsf{title}(m, t, y_1) \land \exists t\, \mathsf{title}(m, t, y_2) \rightarrow (y_1 = y_2)\right),
\end{aligned}$$

we can reduce two JOIN operations in the first three atoms of $\boldsymbol{q}'_{\mathsf{tw}}(t, y)$ to a single atom $\mathsf{title}(m, t, y)$:

$$\boldsymbol{q}''_{\mathsf{tw}}(t, y) \leftarrow \mathsf{title}(m, t, y),\ (y > 2010).$$

Note that these two JOIN operations were introduced to reconstruct the ternary relation from its reification by means of the roles mo:title and mo:year.

The role of SQO in OBDA systems appears to be much more prominent than in conventional RDBMSs, where it was initially proposed to optimise SQL

queries. While some of SQO techniques reached industrial RDBMSs, it never had a strong impact on the database community because it is costly compared to statistics- and heuristics-based methods, and because most SQL queries are written by highly-skilled experts (and so are nearly optimal anyway). In OBDA scenarios, in contrast, SQL queries are generated automatically, and so SQO becomes the only tool to avoid redundant and expensive JOIN operations.

References

Abiteboul, S., Hull, R., and Vianu, V. (1995). *Foundations of Databases*. Addison-Wesley.

Arora, S. and Barak, B. (2009). *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st edition.

Artale, A., Calvanese, D., Kontchakov, R., and Zakharyaschev, M. (2009). The DL-Lite family and relations. *Journal of Artificial Intelligence Research (JAIR)*, 36:1–69.

Baader, F., Brandt, S., and Lutz, C. (2005). Pushing the EL envelope. In Kaelbling, L. P. and Saffiotti, A., editors, *Proceedings of the 19th Int. Joint Conf. on Artificial Intelligence, IJCAI-05*, pages 364–369. Professional Book Center.

Baader, F., Brandt, S., and Lutz, C. (2008). Pushing the EL envelope further. In Clark, K. and Patel-Schneider, P. F., editors, *Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*.

Baget, J.-F., Leclère, M., Mugnier, M.-L., and Salvat, E. (2011). On rules with existential variables: Walking the decidability line. *Artificial Intelligence*, 175(9–10):1620–1654.

Brandt, S. (2004). Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In *Proc. of the 16th European Conf. on Artificial Intelligence (ECAI-2004)*, pages 298–302. IOS Press.

Calì, A., Gottlob, G., and Kifer, M. (2013). Taming the infinite chase: Query answering under expressive relational constraints. *J. of Artificial Intelligence Research*, 48:115–174.

Calì, A., Gottlob, G., and Lukasiewicz, T. (2012). A general datalog-based framework for tractable query answering over ontologies. *J. of Web Semantics*, 14:57–83.

Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., and Rosati, R. (2005). Dl-lite: Tractable description logics for ontologies. In *Proc. of AAAI*, pages 602–607. AAAI Press / The MIT Press.

Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., and Rosati, R. (2006). Data complexity of query answering in description logics. In *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 260–270.

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

Ceri, S., Gottlob, G., and Tanca, L. (1989). What you always wanted to know about datalog (and never dared to ask). *IEEE Trans. Knowl. Data Eng.*, 1(1):146–166.

Chagrov, A. and Zakharyaschev, M. (1997). *Modal Logic*, volume 35 of *Oxford Logic Guides*. Clarendon Press, Oxford.

Chakravarthy, U. S., Fishman, D. H., and Minker, J. (1986). *Semantic query optimization in expert systems and database systems*. Benjamin-Cummings Publishing Co., Inc.

Chortaras, A., Trivela, D., and Stamou, G. (2011). Optimized query rewriting for OWL 2 QL. In *Proc. of CADE-23*, volume 6803 of *LNCS*, pages 192–206. Springer.

Eiter, T., Lutz, C., Ortiz, M., and Simkus, M. (2009). Query answering in description logics: the knots approach. In *Proc. of WOLLIC*, volume 5514 of *Lecture Notes in Computer Science*, pages 26–36. Springer.

Eiter, T., Ortiz, M., Šimkus, M., Tran, T.-K., and Xiao, G. (2012). Query rewriting for Horn-SHIQ plus rules. In *Proc. of AAAI 2012*. AAAI Press.

Elmasri, R. and Navathe, S. (2010). *Fundamentals of Database Systems*. Addison-Wesley, 6th edition.

Garey, M. and Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

Gottlob, G., Kikot, S., Kontchakov, R., Podolskii, V. V., Schwentick, T., and Zakharyaschev, M. (2014). The price of query rewriting in ontology-based data access. *Artif. Intell.*, 213:42–59.

Gottlob, G., Orsi, G., and Pieris, A. (2011). Ontological queries: Rewriting and optimization. In *Proc. of ICDE 2011*, pages 2–13. IEEE Computer Society.

Grosof, B. N., Horrocks, I., Volz, R., and Decker, S. (2003). Description logic programs: Combining logic programs with description logic. In *Proc. of the 12th Int. World Wide Web Conference (WWW 2003)*, pages 48–57.

Horrocks, I., Kutz, O., and Sattler, U. (2006). The even more irresistible SROIQ. In *Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 57–67. AAAI Press.

Horrocks, I., Motik, B., and Wang, Z. (2012). The HermiT OWL reasoner. In *Proc. of ORE*, volume 858 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Johnson, D. S. and Klug, A. C. (1984). Testing containment of conjunctive queries under functional and inclusion dependencies. *J. Comput. Syst. Sci.*, 28(1):167–189.

Kikot, S., Kontchakov, R., Podolskii, V., and Zakharyaschev, M. (2012a). Exponential lower bounds and separation for query rewriting. In *Proc. of ICALP 2012, Part II*, volume 7392 of *LNCS*, pages 263–274. Springer.

Kikot, S., Kontchakov, R., and Zakharyaschev, M. (2012b). Conjunctive query answering with OWL 2 QL. In *Proc. of KR 2012*. AAAI Press.

König, M., Leclère, M., Mugnier, M.-L., and Thomazo, M. (2012). A sound and complete backward chaining algorithm for existential rules. In *Proc. of RR 2012*, volume 7497 of *LNCS*, pages 122–138. Springer.

Kontchakov, R., Lutz, C., Toman, D., Wolter, F., and Zakharyaschev, M. (2011). The combined approach to ontology-based data access. In *Proceedings of the 20th Int. Joint Conf. on Artificial Intelligence, IJCAI-2011*, pages 2656–2661. AAAI Press.

Kozen, D. (2006). *Theory of Computation*. Springer.

Lenzerini, M. (2002). Data integration: A theoretical perspective. In *Proc. of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'02)*, pages 233–246. ACM.

Libkin, L. (2004). *Elements Of Finite Model Theory*. Springer.

Lloyd, J. and Shepherdson, J. (1991). Partial Evaluation in Logic Programming. *The Journal of Logic Programming*, 11(3-4):217–242.

Lutz, C., Seylan, I., Toman, D., and Wolter, F. (2013). The combined approach to obda: Taming role hierarchies using filters. In *Proc. of the 12th International Semantic Web Conference (ISWC 2013)*, volume 8218 of *Lecture Notes in Computer Science*, pages 314–330. Springer.

Lutz, C., Toman, D., and Wolter, F. (2009). Conjunctive query answering in the description logic EL using a relational database system. In *Proceedings of the 21st Int. Joint Conf. on Artificial Intelligence, IJCAI 2009*, pages 2070–2075.

Lutz, C. and Wolter, F. (2007). Conservative extensions in the lightweight description logic EL. In *Proc. of CADE*, volume 4603 of *Lecture Notes in Computer Science*, pages 84–99. Springer.

Motik, B. (2007). On the properties of metamodeling in owl. *J. Log. Comput.*, 17(4):617–637.

Papadimitriou, C. (1994). *Computational Complexity*. Addison-Wesley.

Pérez-Urbina, H., Motik, B., and Horrocks, I. (2009). A comparison of query rewriting techniques for DL-lite. In *Proc. of DL 2009*, volume 477 of *CEUR-WS*.

Pérez-Urbina, H., Rodríguez-Díaz, E., Grove, M., Konstantinidis, G., and Sirin, E. (2012). Evaluation of query rewriting approaches for OWL 2. In *Proc. of SSWS+HPCSW 2012*, volume 943 of *CEUR-WS*.

Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., and Rosati, R. (2008). Linking data to ontologies. *J. on Data Semantics*, X:133–173.

Rodríguez-Muro, M. and Calvanese, D. (2011). Dependencies: Making ontology based data access work. In *Proc. of AMW 2011*, volume 749. CEUR-WS.org.

Rodríguez-Muro, M., Kontchakov, R., and Zakharyaschev, M. (2013). Ontology-Based Data Access: Ontop of databases. In *Proc. of the 12th International Semantic Web Conference (ISWC 2013)*, volume 8218 of *Lecture Notes in Computer Science*, pages 558–573. Springer.

Rosati, R. (2012). Prexto: Query rewriting under extensional constraints in DL-Lite. In *Proc. of EWSC 2012*, volume 7295 of *LNCS*, pages 360–374. Springer.

Rosati, R. and Almatelli, A. (2010). Improving query answering over DL-Lite ontologies. In *Proc. of KR 2010*. AAAI Press.

Schaerf, A. (1993). On the complexity of the instance checking problem in concept languages with existential quantification. *J. of Intelligent Information Systems*, 2:265–278.

Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., and Katz, Y. (2007). Pellet: A practical OWL-DL reasoner. *J. of Web Semantics*, 5(2):51–53.

Steigmiller, A., Liebig, T., and Glimm, B. (2014). Konclude: System description. *J. of Web Semantics*.

ter Horst, H. J. (2005). Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. *J. of Web Semantics*, 3(2–3):79–115.

Tobies, S. (2001). *Complexity results and practical algorithms for logics in Knowledge Representation*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany.

Tsarkov, D. and Horrocks, I. (2006). FaCT++ description logic reasoner: System description. In *Proc. of IJCAR*, volume 4130 of *Lecture Notes in Computer Science*, pages 292–297. Springer.

Vardi, M. (1982). The complexity of relational query languages (extended abstract). In *Proc. of the 14th ACM SIGACT Symp. on Theory of Computing (STOC'82)*, pages 137–146.