Open World and Closed World

Three examples for relating the Open World semantic and the Closed World semantic

Sven J. Jäger

August 7, 2015

Contents

1 Introduction

In relational databases it is assumed that the information contained are complete, what is called the *Closed World Assumption*. This assumption is not made for incomplete databases

or knowledge bases based on description logics (DL) as for example OWL. There it is assumed that the available information are incomplete (*Open World Assumption*). Whereas at a query with a negated condition, a relational database returns all tuples for which it is not stored in the database that they satisfy the condition, the answer under the Open World Assumption contains only such tuples for which it is guaranteed that they do not satisfy the condition.

In the article *Obtaining Complete Answers from Incomplete Databases* by Alon Y. Levy [Lev96] the relational data model well-known at that time is used for partially incomplete databases. The authors investigate the question when the completeness of an answer to a query can be ensured, which was already considered in [Mot89]. The Closed World Assumption inhering in the relational model involves that the information that tuple does not satisfy a certain condition cannot be expressed. Later, with RDF and OWL, a formalism was developed in which the Open World Assumption is inherent. Besides the possibility to store positive atoms (atomic facts), therein exists the possibility to store further axioms (i. e. concept inclusions), that are taken into account in query answering and which can for example imply that a tuple cannot satisfy a certain condition. Such axioms concerning only concepts and not concrete instances form the *TBox* (*Terminological Formalism*), the schema of a knowledge base. They are formulated in the language of description logic and can be translated to predicate logic formulas.

The schema of a relational database contains as well conditions that can be represented by predicate logic formulas, namely the so-called *integrity constraints*. However, these are not taken into consideration in query answering but used to ensure the integrity of the database state after a data update. Considering them for query answering is unnecessary because owing to the underlying Closed World Assumption, all true facts have to be contained in the database. At best, it yields a performance gain.

The different behaviour of relational databases and knowledge bases at query answering can be described formally by means of the underlying semantics. In relational databases, every predicate symbol (relation schema) is interpreted as the relation given by the current database state $\mathcal{S}$. A tuple $(a_1, \ldots, a_n)$ thereby occurs in the answer to a query $\varphi(x_1, \ldots, x_n)$ if and only if $\mathcal{S} \vDash \varphi[a_1/x_1, \ldots, a_n/x_n]$. In knowledge bases, the stored information are not treated as a complete specification but merely as conditions that a potential interpretation must fulfill. Every such interpretation of the predicate symbols is called a *model* of the specification $\mathcal{K}$. A tuple $(a_1, \ldots, a_n)$ then appears in the answer to a query $\varphi(x_1, \ldots, x_n)$ if and only if $\mathcal{S} \vDash \varphi[a_1/x_1, \ldots, a_n/x_n]$ for every interpretation $\mathcal{S}$ with $\mathcal{S} \vDash \mathcal{K}$ (then one says $\mathcal{K}$ *entails* $\varphi[a_1/x_1, \ldots, a_n/x_n]$ and writes $\mathcal{K} \vDash \varphi[a_1/x_1, \ldots, a_n/x_n]$). If $\mathcal{K}$ only contains atoms, then the interpretation used for relational databases is exactly the *minimal model* of $\mathcal{K}$.

The different handling of the database schema in relational databases and DL knowledge bases causes according to [MHS07] misunderstandings and problems in practice. Each of the two behaviours described – on the one hand treating the TBox axioms as background knowledge, from which conclusions can be drawn (*reasoning*) in order to answer a query, on the other hand the strict enforcement of conditions to ensure the integrity of the database state – are in certain situations preferable. In the article *Bridging the Gap Between OWL and Relational Databases* [MHS07], Boris Motik, Ian Horrocks and Ulrike Sattler describe how the semantic of DL knowledge bases can be extended by integrity constraints.

Owing to the high expressiveness of OWL 2, the complexity of query answering over OWL 2 ontologies is high. A common approach to obtain reasonable performance is to restrict the ontology language to a more tractable fragment (*OWL profile*). The answer of a reasoner for such a profile is always correct, but it is complete only if the ontology falls inside the profile. Hence, the answer computed by such a system provides a lower bound for the actual answer. However the incompleteness is unsatisfactory which is why the optimization of fully fledged OWL 2 reasoners is an ongoing field of research. Such an approach is presented by Yujiao Zhou, Yavor Nenov, Bernardo Cuenca Grau and Ian Horrocks in the paper *Pay-as-you-go Ontology Query Answering Using a Datalog Reasoner* [ZNGH14] for which the prototypical system PAGOdA was developed and the associated technical report [ZCGN$^+$15]. The idea is based on the calculation of a lower and an upper bound for the answer and on checking the answers in the gap with respect to a relevant fragment of the ontology. Thereto a Datalog reasoner is used. Like the semantic of the relational model, the semantic of Datalog is based on the Closed World Assumption. Unfortunately, the system PAGOdA did not answer correctly typical test queries.

2 Completeness of answers in incomplete databases

Let $\mathbf{R} = \{R_1(\overline{X_1}), \ldots, R_n(\overline{X_n})\}$ be a relational signature. Below it is assumed that all tuples in the database also hold in reality but that not all tuples holding in reality are contained in the database. So two different relational structures over $\mathbf{R}$ are considered, on the one hand the "real state of the world" $\mathcal{S}$ and on the other hand the database state $\mathcal{S}'$, for which holds that $\mathcal{S}'(R_i) \subset \mathcal{S}(R_i)$ for all $i \in \{1, \ldots, n\}$.[1] Thereto the notation $\mathcal{S}' \sqsubseteq \mathcal{S}$ is used.

The answer set $\mathcal{S}'(Q)$ to a query Q is called *correct* if $\mathcal{S}'(Q) \subset \mathcal{S}(Q)$ and *complete* if $\mathcal{S}(Q) \subset \mathcal{S}'(Q)$. Obviously, under the above assumption, the answer to every query without negation is correct. In what follows, it is investigated when such an answer is also complete. In order to make a statement about the completeness (without knowing $\mathcal{S}$), it must be known that parts of the database are complete. What is meant by this, is illustrated by the following example and subsequently defined formally.

Example 2.1. Consider a movie database with the following relational signature $\mathbf{R}$

$$\begin{aligned}&\mathsf{Movie}(\underline{\mathsf{title}}, \mathsf{director}, \underline{\mathsf{year}})\\ &\mathsf{Oscar}(\mathsf{title}, \mathsf{year})\\ &\mathsf{Showing}(\mathsf{title}, \mathsf{year}, \mathsf{cinema}, \mathsf{hour})\\ &\mathsf{Director}(\underline{\mathsf{name}}, \mathsf{dateOfBirth}, \mathsf{nationality}),\end{aligned}$$

that contains besides the information about the movies also the showings in some cinema in Göttingen. Assume that the relation Movie is complete from the year 1960 on, but that older movies can be missing, and that the relations Oscar, Showing and Director are complete.

[1] The notation of [Lev96] was changed to the notation of the database lecture, which was also used by [Mot89].

1. The answer to the query

$$(Q_1) \qquad \begin{array}{l} \texttt{SELECT m.director} \\ \texttt{FROM Movie m, Oscar o} \\ \texttt{WHERE m.title} = \texttt{o.title} \\ \quad \texttt{AND m.year} = \texttt{o.year} \\ \quad \texttt{AND m.year} \geq 1965 \end{array}$$

   for the directors of all movies that won an Oscar since 1965 is under the above assumptions complete.

2. For the answer to the query

$$(Q_2) \qquad \begin{array}{l} \texttt{SELECT m.title, m.director} \\ \texttt{FROM Movie m, Showing s} \\ \texttt{WHERE m.title} = \texttt{s.title} \\ \quad \texttt{AND m.year} = \texttt{s.year} \end{array}$$

   for the names and directors of all movies that are currently shown in Göttingen, the completeness cannot be guaranteed. However if

$$\begin{aligned} \mathcal{S}'(\mathsf{Showing}) := \{&(\textit{Jurassic World}, 2015, \textit{CinemaxX}, 20{:}00), \\ &(\textit{Elser – Er hätte die Welt verändert}, 2015, \textit{Lumière}, 17{:}00), \\ &(\textit{Die Entdeckung der Unendlichkeit}, 2014, \textit{Unikino}, 20{:}00)\}, \end{aligned}$$

   then the answer to the query must be complete because all movies that are currently played in Göttingen must be contained in the relation Movie.

The example shows that it does not suffice to specify a set of names of relations that are complete in $\mathcal{S}'$, since single relations can be "partially complete", i. e. complete for all tuples satisfying a certain condition. This will be conceptualized by the following definitions.

Definition 2.1. Let $R(X_1, \ldots, X_n) \in \mathbf{R}$. A *constraint* C on R is a conjunction of atoms that can include variables from $X_1, \ldots, X_n$ as well as further variables and that does not use the relation name R. A tuple $(a_1, \ldots, a_n)$ *satisfies* C w. r. t. the database state $\mathcal{T}$ if $\mathcal{T} \models C[a_1/X_1, \ldots, a_n/X_n]$.

Example 2.2.

1. In the above example, year ≥ 1960 is a constraint on Movie. It is satisfied by all movies that were shot since 1960.

2. Showing(title, year, cinema, hour) $\wedge$ year ≥ 1960 is a constraint on Movie as well. It is satisfied by all movies shot since 1960 and currently shown in a cinema in Göttingen.

3. One has to be careful with the naming of the variables. For instance the constraint Showing(title2, year, cinema, hour) $\wedge$ year ≥ 1960 is satisfied by all movies shot in a year since 1960 in which also a film currently shown in Göttingen was shot.

Definition 2.2. Let $\mathcal{S}' \sqsubseteq \mathcal{S}$ and C a constraint on the relation name R. The database state $\mathcal{S}'$ is *locally complete* at R under the constraint C for $\mathcal{S}$ if $\mathcal{S}'(R)$ contains all tuples from $\mathcal{S}(R)$ that satisfy C. Then, one says $\mathcal{S}'$ fulfills the *local completeness statement* $\mathrm{LC}(R, C)$ for $\mathcal{S}$ and writes $(\mathcal{S}, \mathcal{S}') \models \mathrm{LC}(R, C)$.

Example 2.3. The statement

$$(\mathcal{S}, \mathcal{S}') \models \mathrm{LC}(\mathsf{Movie}, \mathsf{Showing}(\mathsf{title}, \mathsf{year}, \mathsf{cinema1}, \mathsf{hour1}) \\ \wedge \mathsf{Showing}(\mathsf{title}, \mathsf{year}, \mathsf{cinema2}, \mathsf{hour2}) \wedge \mathsf{cinema1} \neq \mathsf{cinema2}).$$

means that all movies from $\mathcal{S}$ that are shown in at least two cinemas in Göttingen are contained in $\mathcal{S}'$.

For a set Γ of local completeness statements, one writes $(\mathcal{S}, \mathcal{S}') \models \Gamma$ if $(\mathcal{S}, \mathcal{S}') \models \mathrm{LC}(R, C)$ for all $\mathrm{LC}(R, C) \in \Gamma$.

Now the problem to decide whether the answer to a query Q is complete may be formalized. If $\mathcal{S}$ and $\mathcal{S}'$ are known, this can trivially be decided by calculating $\mathcal{S}(Q)$ and $\mathcal{S}'(Q)$ and comparing them. In practice, this is of course not applicable since $\mathcal{S}$ is not available. Instead, the completeness is to be derived from the local completeness statements and, when indicated, from the current database state $\mathcal{S}'$.

Definition 2.3. Let $\mathbf{R}$ be a relational signature and Γ a set of local completeness statements of the form $\mathrm{LC}(R, C)$ for $R \in \mathbf{R}$. A query Q over $\mathbf{R}$ is called *answer-complete w. r. t.* Γ if for all structures $\mathcal{S}$ and $\mathcal{S}'$ over $\mathbf{R}$ with $(\mathcal{S}, \mathcal{S}') \models \Gamma$ holds that $\mathcal{S}'(Q) = \mathcal{S}(Q)$.

Definition 2.4. Let $\mathbf{R}$ be a relational signature, Γ a set of local completeness statements of the form $\mathrm{LC}(R, C)$ for $R \in \mathbf{R}$ and $\mathcal{S}'$ a database state over $\mathbf{R}$. A query Q over $\mathbf{R}$ is called *instance answer-complete w. r. t.* Γ *and* $\mathcal{S}'$ if for all structures $\mathcal{S}$ over $\mathbf{R}$ with $(\mathcal{S}, \mathcal{S}') \models \Gamma$ holds: $\mathcal{S}'(Q) = \mathcal{S}(Q)$.

Clearly, if a query is answer complete, then it is instance answer complete for every database instance $\mathcal{S}'$. The converse does not hold: The query Q_2 from example 2.1 is not answer-complete but instance answer-complete w. r. t. the given state $\mathcal{S}'$.

Remark 2.1. From a theoretical point of view, one could define a query Q to be reality answer-complete w. r. t. Γ and $\mathcal{S}$ if for all $\mathcal{S}'$ with $(\mathcal{S}, \mathcal{S}') \models \Gamma$ holds that $\mathcal{S}'(Q) = \mathcal{S}(Q)$. Of course, this cannot be decided only on the basis of the database state; however it would be conceivable that the reality answer-completeness of a query can be deduced from additional knowledge about the reality (ontology). If $\mathcal{S}$ and $\mathcal{S}'$ are two structures over $\mathbf{R}$ with $(\mathcal{S}, \mathcal{S}') \models \Gamma$ and if Q is a positive query, then neither the instance answer-completeness of Q w. r. t. $\mathcal{S}'$ implies the reality answer-completeness w. r. t. $\mathcal{S}$ nor vice versa.

Example 2.4.

1. Let $\Gamma = \{\mathrm{LC}(\mathsf{Director}, \top)\}$ and

$$\mathcal{S}'(\mathsf{Director}) = \{(\mathit{Woody\ Allen}, \mathit{December}\ 1,\ 1935, \mathit{USA}), \\ (\mathit{Steven\ Spielberg}, \mathit{December}\ 18, 1946,\ \mathit{USA})\}$$

and

$$\mathcal{S}'(\mathsf{Movie}) = \{(Midnight\ in\ Paris, Woody\ Allen, 2011),\\ (Minority\ Report, Steven\ Spielberg, 2002)\}.$$

Then the query

$$(Q_1) \qquad \begin{array}{l}\texttt{SELECT}\ \texttt{d.name},\ \texttt{d.dateOfBirth}\\ \texttt{FROM}\ \mathsf{Director}\ \texttt{d},\ \mathsf{Movie}\ \texttt{m}\\ \texttt{WHERE}\ \texttt{d.name} = \texttt{m.director}\\ \quad \texttt{AND}\ \texttt{m.year} \geq 2000,\end{array}$$

that asks for all directors with their dates of birth that directed a film since 2000, is instance answer-complete w. r. t. $\mathcal{S}'$. Let $\mathcal{S} := \mathcal{S}'$. Then Q_1 is not reality answer-complete w. r. t. $\mathcal{S}$ because for $\mathcal{S}''$ with $\mathcal{S}''(\mathsf{Movie}) = \emptyset$, $\mathcal{S}''(Q_1) \neq \mathcal{S}(Q_1)$.

2. Let $\Gamma = \{\mathrm{LC}(\mathsf{Movie}, \mathsf{year} \geq 1960)\}$ and

$$\mathcal{S}(\mathsf{Movie}) = \{(Midnight\ in\ Paris, Woody\ Allen, 2011),\\ (Minority\ Report, Steven\ Spielberg, 2002)\}.$$

Then the query

$$(Q_2) \qquad \begin{array}{l}\texttt{SELECT}\ \texttt{m.title}\\ \texttt{FROM}\ \mathsf{Movie}\ \texttt{m}\end{array}$$

is reality answer-complete w. r. t. $\mathcal{S}$. However, for $\mathcal{S}' := \mathcal{S}$, Q_2 is not instance answer-complete w. r. t. $\mathcal{S}'$

In the following, we will focus on answer-completeness and instance answer-completeness.

2.1 Answer-completeness

In [Lev96], the problem of checking answer-completeness is transformed to the problem of detecting update independence for which was already known ([LS93]) that it can be reduced to the problem of checking query equivalence. In what follows, we will omit the intermediate step and directly reduce the answer-completeness problem to the query equivalence problem.

Theorem 2.1. *Let Q be a union of positive conjunctive queries over $\mathbf{R} = \{R_1(\overline{X}_1), \ldots, R_n(\overline{X}_n)\}$ and comparison predicates, and let $\Gamma = \{\mathrm{LC}(R_i, C_{ij}) \mid 1 \leq j \leq m_j,\ 1 \leq i \leq n\}$. Let $E_1, \ldots, E_n$ be new relation symbols and for $1 \leq i \leq n$ let V_i be the view defined by*

$$V_i(\overline{X}_i) :- R_i(\overline{X}_i) \vee (E_i(\overline{X}_i) \wedge \neg C_{i1} \wedge \cdots \wedge \neg C_{im_i}).$$

Then the query Q is answer-complete w. r. t. Γ if and only if Q is equivalent to the query Q' obtained by replacing every occurrence of R_i by V_i for $1 \leq i \leq n$.

Remark 2.2. In [Lev96], the case that there is more than one local completeness statement for one relation, is not properly considered. Instead, in [Lev96, Theorem 3.1], it is claimed that Q is answer complete if and only if for every local completeness statement $\mathrm{LC}(R, C_i)$, Q is independent of the insertion of tuples not satisfying C_i. This is wrong in the case of multiple local completeness statements for one relation as shows the following example.

Example 2.5. Consider

$$\Gamma = \{\mathrm{LC}(\mathsf{Movie},\ \mathsf{year} \geq 1960),\ \mathrm{LC}(\mathsf{Movie},\ \mathsf{year} \geq 1970)\}$$

and the query

$$\begin{array}{ll} (Q) & \texttt{SELECT}\ \texttt{m.title} \\ & \texttt{FROM}\ \mathsf{Movie}\ \texttt{m} \\ & \texttt{WHERE}\ \texttt{m.}\mathsf{year} \geq 1965 \end{array}$$

Then Q is answer-complete. However, Q is not independent of the insertion of tuples with $\mathsf{year} < 1970$.

The query equivalence problem is of high complexity. In many cases, the answer-completeness problem can however be reduced to checking query satisfiability, what can be done in polynomial time. In the general case of a conjunctive query, we only get a sufficient condition for answer-completeness, but under certain additional assumptions, this condition is also necessary. We call a conjunctive query $Q = c_1 \wedge \cdots \wedge c_k$ *minimal* if there is no $i \in \{1, \ldots, k\}$ such that Q is equivalent to $c_1 \wedge \cdots \wedge c_{i-1} \wedge c_{i+1} \wedge \cdots \wedge c_k$.

Theorem 2.2. *Let Q be a positive conjunctive query over $\mathbf{R} = \{R_1, \ldots, R_n\}$, and let $\Gamma = \{\mathrm{LC}(R_i, C_{ij}) \mid 1 \leq j \leq m_i,\ 1 \leq i \leq n\}$. For $i \in \{1, \ldots, n\}$ let R_i have relation schema $R_i(X_1^i, \ldots, X_{l_i}^i)$. If for all $i \in \{1, \ldots, n\}$, for every occurrence $R_i(Y_1, \ldots, Y_{l_i})$ of the relation name R_i in the query Q, the query $Q \wedge \sigma(\neg C_{i1} \wedge \cdots \wedge \neg C_{im_i})$, where σ is the substitution that replaces X_k^i by Y_k, is unsatisfiable, then Q is answer-complete w. r. t. Γ.*

Conjecture. *Let Q be a* minimal *positive conjunctive query over $\mathbf{R} = \{R_1, \ldots, R_n\}$, and let $\Gamma = \{\mathrm{LC}(R_i, C_{ij}) \mid 1 \leq j \leq m_i,\ 1 \leq i \leq n\}$. For $i \in \{1, \ldots, n\}$ let R_i have relation schema $R_i(X_1^i, \ldots, X_{l_i}^i)$. If there is a $i \in \{1, \ldots, n\}$ and an occurrence $R_i(Y_1, \ldots, Y_{l_i})$ of the relation name R_i in the query Q such that the query $Q \wedge \sigma(\neg C_{i1} \wedge \cdots \wedge \neg C_{im_i})$ is satisfiable, where σ substitutes X_k^i by Y_k, then Q is not answer-complete w. r. t. Γ.*

The reason for the first statement to be true is that if all the queries obtained by restricting an occurrence of a relation R_i to the potentially incomplete part are unsatisfiable, then these parts cannot contribute to the answer to the positive conjunctive query. We give two examples where this is not the case, i. e. there is an occurrence of R_i such that the restriction to the incomplete part is still satisfiable.

Example 2.6. We consider again the signature from example 2.1. Let

$$\begin{aligned} \Gamma := \{&\mathrm{LC}(\mathsf{Movie},\ \mathsf{year} \geq 1960), \\ &\mathrm{LC}(\mathsf{Oscar},\ \top), \\ &\mathrm{LC}(\mathsf{Showing},\ \top), \\ &\mathrm{LC}(\mathsf{Director}, \top)\}. \end{aligned}$$

Let

$$(Q_1) \qquad \begin{array}{l} \texttt{SELECT m1.title} \\ \texttt{FROM Movie m1, Movie m2} \\ \texttt{WHERE m1.year} \geq 1960 \\ \texttt{AND m.director} = \texttt{m2.director} \\ \texttt{AND m2.year} < 1960 \end{array}$$

be a query asking for all films since 1960 whose director also directed a film before 1960. This query is minimal. In relational calculus form, it is

$$Q_1(\mathsf{title}) = \mathsf{Movie}(\mathsf{title}, \mathsf{director}, \mathsf{year1}) \wedge \mathsf{year1} \geq 1960 \wedge \mathsf{Movie}(\_, \mathsf{director}, \mathsf{year2}) \wedge \mathsf{year2} < 1960,$$

where variables occurring only once are written as underscores (*anonymous variables*). The only relation name in the query is Movie, which occurs twice. Let C be the corresponding constraint, i. e. $C = \mathsf{year} \geq 1960$.

- Let $\sigma_1(\mathsf{year}) = \mathsf{year1}$. Then

  $$\begin{aligned} Q_1 \wedge \sigma_1(\neg C) = \mathsf{Movie}(\mathsf{title}, \mathsf{director}, \mathsf{year1}) \wedge \mathsf{year1} \geq 1960 \\ \wedge \mathsf{Movie}(\_, \mathsf{director}, \mathsf{year2}) \wedge \mathsf{year2} < 1960 \wedge \mathsf{year1} < 1960 \end{aligned}$$

  which is obviously unsatisfiable.

- Let $\sigma_2(\mathsf{year}) = \mathsf{year2}$. Then

  $$\begin{aligned} Q_1 \wedge \sigma_2(\neg C) = \mathsf{Movie}(\mathsf{title}, \mathsf{director}, \mathsf{year1}) \wedge \mathsf{year1} \geq 1960 \\ \wedge \mathsf{Movie}(\_, \mathsf{director}, \mathsf{year2}) \wedge \mathsf{year2} < 1960 \wedge \mathsf{year2} < 1960 \end{aligned}$$

  which is satisfiable. In accord with the conjecture, Q_1 is not answer-complete. The example illustrates that it is necessary to consider every occurrence of the relation names.

The next example shows that the additional assumption of minimality in the second statement is necessary.

Example 2.7. We consider the same signature and the same set Γ as in the last example. Let

$$(Q_2) \qquad \begin{array}{l} \texttt{SELECT m1.title} \\ \texttt{FROM Movie m1, Movie m2} \\ \texttt{WHERE m1.year} \geq 1960 \\ \texttt{AND m1.director} = \texttt{m2.director}. \end{array}$$

In relational calculus form, this is

$$Q_2(\mathsf{title}) = \mathsf{Movie}(\mathsf{title}, \mathsf{director}, \mathsf{year1}) \wedge \mathsf{year1} \geq 1960 \wedge \mathsf{Movie}(\_, \mathsf{director}, \mathsf{year2})$$

Again, restricting $\mathsf{year2} < 1960$ preserves satisfiability. However, the query is answer-complete since m2 can always be chosen equal to m1. This does not disprove the conjecture because the query is not minimal.

Remark 2.3. [Lev96] claims a weaker version of the conjecture in which an additional precondition on the shape of C is made and gives a proof sketch for it.

2.2 Instance answer-completeness

The idea is to check whether the answer to one part of the query contains the answer to another part of the query which is known to be answer-complete and which determines functionally all variables of the query.

Example 2.8. Consider again the query Q_2 from example 2.1. In relational calculus form, it is

$$Q_2(\mathsf{title}, \mathsf{director}) = \mathsf{Movie}(\mathsf{title}, \mathsf{director}, \mathsf{year}) \wedge \mathsf{Showing}(\mathsf{title}, \mathsf{year}, \mathsf{cinema}, \mathsf{hour}).$$

Then the subquery

$$\underline{Q_2}(\mathsf{title}, \mathsf{year}) = \mathsf{Showing}(\mathsf{title}, \mathsf{year}, \mathsf{cinema}, \mathsf{hour})$$

is known to be answer-complete and the variables of this query functionally determine all variables of Q_2. Furthermore for the database state $\mathcal{S}'$ of Example 2.1, the answer to $\underline{Q_2}$ is contained in the answer to $\overline{Q_2}(\mathsf{title}, \mathsf{year}) = \mathsf{Movie}(\mathsf{title}, \mathsf{director}, \mathsf{year})$. This is why the answer to Q_2 must be complete.

The example can easily be generalized leading to the following theorem.

Theorem 2.3. *Let $Q(\overline{X}) = R_1(\overline{X}_1) \wedge \cdots \wedge R_n(\overline{X}_n) \wedge C$ be a conjunctive query, where $R_1, \ldots, R_n \in \mathbf{R}$ and C is a conjunction of comparison atoms, let Γ be a set of local completeness statements of the form $\mathrm{LC}(R, C)$ with $R \in \mathbf{R}$ and let $\mathcal{S}'$ be a database state. Let $\{1, \ldots, n\} = I \dot\cup J = \{i_1, \ldots, i_k\} \dot\cup \{j_1, \ldots, j_{n-k}\}$ be a partition of the relational atoms, $\overline{X}_I := \bigcup_{i \in I} \overline{X}_i$, $\overline{X}_J := \bigcup_{j \in J} \overline{X}_j$ and $\overline{Y} := \overline{X} \cap \overline{X}_I \cap \overline{X}_J$. Let furthermore C_I be the set of comparison atoms of C containing only variables from $\overline{X}_I$ and C_J be the set of comparison atoms containing only variables from $\overline{X}_J$. Let $Q_I(\overline{Y}) = R_{i_1}(\overline{X}_{i_1}) \wedge \cdots \wedge R_{i_k}(\overline{X}_{i_k}) \wedge C_I$ and $Q_J(\overline{Y}) = R_{j_1}(\overline{X}_{j_1}) \wedge \cdots \wedge R_{j_{n-k}}(\overline{X}_{j_{n-k}}) \wedge C_J$.*

If $\overline{X}_I$ functionally determines every variable of Q, Q_I is answer complete, and $\mathcal{S}'(Q_I) \subset \mathcal{S}'(Q_J)$, then Q is instance answer-complete w. r. t. Γ and $\mathcal{S}'$.

If $\overline{X}_I$ functionally determines the other variables, then for every answer to Q_I, there exists exactly one answer to Q in $\mathcal{S}$. If this answer is already in $\mathcal{S}'(Q_J)$, then it is also in $\mathcal{S}'(Q)$. So the answer is complete.

2.3 Correctness

The results of the last sections for incomplete databases can easily be transferred to incorrect databases. There, $\mathcal{S} \sqsubseteq \mathcal{S}'$ is assumed. All results only relied on identifying if a query is independent of updates. This is why the same algorithms can be applied even if the database is both incomplete and incorrect.

3 Integrity Constraints in DL-Knowledge Bases

In [MHS07], the authors discuss different ideas and on the basis of examples, the related problems and draw conclusions, that finally result in the semantic proposed. In the following subsections, the ideas and the resulting problems are summarized.

3.1 Idea 1: Consistency check

In relational databases, after changing the database state, it is checked whether the schema constraints are compatible with the database state, i. e. whether the model given by the database state satisfies the schema constraints. Something similar is done at a consistency check of a knowledge bases. There it is checked whether the schema axioms are consistent with the atomic facts *ABox* (*assertional formalism*), that means, if there is a model that satisfies the statements of the ABox and the axioms of the TBox. As owing to the Open World Assumption, among others, models in which also not explicitly given statements hold are considered, the completeness of the data cannot be verified. In case of an insufficient specification of the ontology, some database states felt to be contradictory do not result in a logic inconsistency.

Example 3.1. The objective is to ensure that for every country at least one language spoken therein is listed. For this purpose, the TBox axiom

$$\mathsf{Country} \sqsubseteq \exists \mathsf{language}.\mathsf{Language}$$

is set up. However, the ABox

$$\mathsf{Country}(\mathsf{Germany}),\ \mathsf{Country}(\mathsf{Bahamas}),\ \mathsf{Language}(\mathsf{German}),\ \mathsf{language}(\mathsf{Germany},\ \mathsf{German})$$

is not inconsistent because a model $\mathcal{S}$ with $\mathcal{S}(\mathsf{Country}) = \{\mathsf{Germany},\ \mathsf{Bahamas}\}$ and $\mathcal{S}(\mathsf{language}) = \{(\mathsf{Germany},\ \mathsf{German}),\ (\mathsf{Bahamas},\ \mathsf{German})\}$ satisfies for example the TBox axiom.

Example 3.2. It is to be ensured that the capital of every country is a city. Thereto the TBox axiom

$$\top \sqsubseteq \forall \mathsf{capital}.\mathsf{City}$$

is introduced. This time the ABox

$$\mathsf{City}(\mathsf{Berlin}),\ \mathsf{Person}(\mathsf{Angela\ Merkel}),\ \mathsf{capital}(\mathsf{Germany},\ \mathsf{Angela\ Merkel})$$

again does not cause an inconsistency, although Angela Merkel is indicated in the database to be a person, because it is not stated in the TBox that the concepts City and Person are disjoint. The reasoner would merely conclude from the preceding axiom that Angela Merkel must be city and person at the same time.

If the objective is to guarantee that a certain part of the database is complete (Closed World), then the treatment of the TBox axioms as integrity constraints is intended; but if the database is incomplete in a certain part (Open World), then the TBox axiom should, as customary for

DL knowledge bases, be used to derive further facts. The goal is therefore that a user can explicitly indicate, which TBox axioms should be treated as integrity constraints. To this end, the set of TBox axioms is partitioned in a set $\mathcal{T}_S$ of standard TBox axioms and a set $\mathcal{T}_C$ of TBox axioms treated as integrity constraints.

3.2 Idea 2: The ABox must satisfy the integrity constraints

The question arises how the satisfaction of the integrity constraints can be checked. Since they are supposed to ensure the correctness of the data and not of the schema, it is a likely idea to consider, while checking the integrity, only the ABox $\mathcal{A}$, and to regard the standard TBox axioms only while answering queries. For the integrity check, the ABox could be treated like a relational data base taking the Closed World Assumption as a basis. This approach would have the desired effect for the two preceding examples.

In terms of model theory, the integrity constraints would be applied to the model of the ABox in which every predicate symbol is interpreted by the set of explicitly asserted tuples.

It is not clear at first sight why the rules for drawing conclusions should be taken into consideration in the check of the integrity constraints. The following example is supposed to show that this can indeed be appropriate and that the nonconsiederation can have unwanted consequences.

Example 3.3. The objective is to ensure that only an animal can be the pet of somebody. For this purpose, one sets

$$\mathcal{T}_C := \{\top \sqsubseteq \forall \mathsf{hasPet}.\mathsf{Animal}\}.$$

Let further

$$\mathcal{T}_S := \{\mathsf{Cat} \sqsubseteq \mathsf{Animal},\ \mathsf{Dog} \sqsubseteq \mathsf{Animal}\}$$

and the ABox be

$$\mathcal{A} := \{\mathsf{Dog}(\mathsf{Struppi}),\ \mathsf{Cat}(\mathsf{Garfield}),\ \mathsf{hasPet}(\mathsf{John}, \mathsf{Garfield}),\ \mathsf{hasPet}(\mathsf{Alice}, \mathsf{Struppi})\}.$$

Then the ABox does not satisfy the integrity constraint, as in the ABox, it is not asserted explicitly that Garfield and Struppi are animals. This only follows using the axiom from $\mathcal{T}_S$.

3.3 Idea 3: Every model of $\mathcal{A} \cup \mathcal{T}_S$ must satisfy the integrity constraints

At first sight, the demand that every model of $\mathcal{A} \cup I_S$ has to satisfy the integrity constraints would solve the problem of the last example, because in every model of $\mathcal{A} \cup \mathcal{T}_S$ hold $\mathsf{Animal}(\mathsf{Garfield})$ and $\mathsf{Animal}(\mathsf{Struppi})$. However, there are quite a lot of models of $\mathcal{A} \cup \mathcal{T}_S$, many of which do not satisfy the integrity constraints, as shows the following example.

Example 3.4. For $\mathcal{T}_S$ and $\mathcal{A}$ from example 3.3, $\mathcal{S}$ with $\mathcal{S}(\mathsf{Animal}) = \{\mathsf{Struppi},\ \mathsf{Garfield}\}$ and $\mathcal{S}(\mathsf{hasPet}) = \{(\mathsf{John}, \mathsf{Garfield}),\ (\mathsf{Alice}, \mathsf{Struppi}),\ (\mathit{Ash}, \mathit{Pikachu})\}$ is a feasible model which obviously does not satisfy the integrity constraint $\top \sqsubseteq \forall \mathsf{hasPet}.\mathsf{Animal}$.

One can limit oneself to Herbrand models over the active domain ADOM of the database, i. e. such models whose universe consists only of the occurring resources and literal values. But of course, it is easy to find an example of such a Herbrand model, in which facts hold that can not at all be deduced from the knowledge base, and which does not satisfy the integrity constraints.

Example 3.5. For the same $\mathcal{T}_S$ and $\mathcal{A}$ as in the last two examples, $\mathcal{S}$ with $\mathcal{S}(\mathsf{Animal}) = \{\mathsf{Struppi},\ \mathsf{Garfield}\}$ and $\mathcal{S}(\mathsf{hasPet}) = \{(\mathsf{John},\ \mathsf{Garfield}),\ (\mathsf{Alice},\ \mathsf{Struppi}),\ (\mathsf{Garfield},\ \mathsf{John})\}$ is a possible Herbrand model over the active domain that does not satisfy the integrity constraint.

In fact, this approach is equivalent to checking whether $\mathcal{A} \cup \mathcal{T}_S$ entails the integrity constraint.

3.4 Idea 4: The consequence set of $\mathcal{A} \cup \mathcal{T}_S$ must satisfy the integrity constraints

The idea to consider all possible models resulted in problems in the last example since there are models in which some facts are true that do not follow from the ABox and the TBox. Therefore it seams sensible to regard instead the set of all positive atoms following from the ABox-atoms and TBox-axioms. So it is checked whether the integrity constraints are satisfied in the model of $\mathcal{A}$ in which every n-ary predicate symbol R is interpreted by the relation $\{(a_1, \ldots, a_n) \mid \mathcal{T}_S \cup \mathcal{A} \vDash R(a_1, \ldots, a_n)\}$. This approach corresponds to the semantic of positive query answering. In all preceding examples, this would have the desired effect (if the condition in the first two examples is considered as integrity constraint). Problems arise however for disjunctive rules as the subsequent example shows.

Example 3.6. The purpose is to ensure that every student has an entry as bachelor or master student in the database and that every staff member is known to be a technical staff member or a scientific staff member. To this end, the integrity constraints

$\mathcal{T}_C := \{\mathsf{Student} \sqsubseteq \mathsf{BachelorStudent} \sqcup \mathsf{MasterStudent},\ \mathsf{StaffMember} \sqsubseteq \mathsf{TechicalStaff} \sqcup \mathsf{ScientificStaff}\}$

are used. Let now

$$\mathcal{T}_S := \{\mathsf{FacultyMember} \sqsubseteq \mathsf{Student} \sqcup \mathsf{StaffMember}\},$$
$$\mathcal{A} := \{\mathsf{FacultyMember}(\mathsf{Fritze\ Bolte})\}.$$

Then neither the statement $\mathsf{Student}(\mathsf{Fritze\ Bolte})$ nor the statement $\mathsf{StaffMember}(\mathsf{Fritze\ Bolte})$ is a consequence of $\mathcal{T}_S \cup \mathcal{A}$, so the two predicate symbols Student and StaffMember are interpreted by the empty set and thus, the integrity constraint is satisfied. So Fritze Bolte being listed neither as bachelor student nor as master student nor as technical staff member nor as scientific staff member does not cause a violation of the integrity constraint.

This behaviour is even more unpleasant if, as in the following example, the same constraint has to be satisfied for both possible superclasses.

Example 3.7. As a variant of the preceding example, now, it is to be guaranteed that every student and every staff member be a person, i. e.

$$\mathcal{T}_C := \{\mathsf{Student} \sqsubseteq \mathsf{Person},\ \mathsf{StaffMember} \sqsubseteq \mathsf{Person}\}.$$

Still, for $\mathcal{T}_S$ and $\mathcal{A}$ as above, the integrity constraint is satisfied for the consequence set although it can not be derived that Fritze Bolte is a person.

3.5 Idea 5: Minimal Herbrand models must satisfy the integrity constraints

As the last examples show, the consequence set of $\mathcal{A} \cup \mathcal{T}_S$ is too small as a base of the integrity check. It is indeed a model of $\mathcal{A}$ but in general not a model of $\mathcal{A} \cup \mathcal{T}_S$, whence it is possible that integrity constraints are satisfied that are violated by every model of $\mathcal{A} \cup \mathcal{T}_S$ and vice versa. The problem is that the disjunctive rule heads do not find their way into the consequence set though they confine the set of feasible models. In example 3.7, this problem could be solved by introducing a new predicate StudentOrStaffMember and adapting the other constraints accordingly. In example 3.6 however, this would not work.

Instead of considering the consequence set, A better idea is to require minimal Herbrand models of $\mathcal{A} \cup \mathcal{T}_S$ to satisfy the integrity constraints. For this, a Herbrand model $\mathcal{S}$ is identified with the subset of the Herbrand base consisting of all atoms that are true in $\mathcal{S}$. Then, $\mathcal{S}$ is called *minimal* if there is no Herbrand model $\mathcal{S}'$ with $\mathcal{S}' \subsetneq \mathcal{S}$. Unlike to the second idea, the integrity constraints are not just checked under the assumption that the ABox contains a closed description of the world. Nevertheless, the restriction to minimal models is predicated on the Closed World Assumption. It corresponds to the semantic of Datalog.

In example 3.6, there are two minimal models, namely

$$\mathcal{S}_1 = \{\mathsf{FacultyMember}(\mathsf{Fritze\ Bolte}), \mathsf{Student}(\mathsf{Fritze\ Bolte})\},$$
$$\mathcal{S}_2 = \{\mathsf{FacultyMember}(\mathsf{Fritze\ Bolte}), \mathsf{StaffMember}(\mathsf{Fritze\ Bolte})\},$$

both obviously not satisfying the integrity constraint. Thus, the knowledge base is rightly identified as incorrect. The same applies for example 3.7.

Also in the examples before these two, the approach results in the intended behaviour: in each of the examples 3.1 and 3.2, there is only one minimal model, which violates the integrity constraint as requested, and in example 3.3, the unique minimal model satisfies the integrity constraint.

3.6 Further decisions and formal definition

The following two questions are to be settled:

1. How to proceed if some of the minimal models satisfy an integrity constraint and some do not?
2. How to deal with axioms in $\mathcal{T}_S$ containing quantifiers?

Example 3.8. In the ontology, it is specified that every student has a matriculation number even if the latter is not explicitly given, i. e.

$$\mathcal{T}_S := \{\mathsf{Student} \sqsubseteq \exists \mathsf{matrNo}.\top\}.$$

Let furthermore

$$\mathcal{A} := \{\mathsf{Student}(\mathsf{Fritze\ Bolte})\}.$$

Now it depends on what is used as (algebraic) signature of the Herbrand universe (datatype literals, active domain, natural numbers, Skolem function symbols, ...). The active domain would not be a good idea in this example; it only consists of Fritze Bolte, and since there are no function symbols, the associated Herbrand universe is $U = \{\mathsf{Fritze\ Bolte}\}$. Consequently, the only Herbrand model of $\mathcal{A} \cup \mathcal{T}_S$ is given by

$$\{\mathsf{Student}(\mathsf{Fritze\ Bolte}), \mathsf{matrNo.}(\mathsf{Fritze\ Bolte}, \mathsf{Fritze\ Bolte})\}.$$

To consider a separate minimal model for every datatype value is of course not a good idea as well. The only practical method is to introduce new objects for the existential quantifiers, similar to Blank Nodes in RDF. Below, alternative approaches are compared.

For answering the first question, it has to be decided

a) whether the integrity constraints from $\mathcal{T}_C$ must be satisfied in *every* minimal model,

b) or whether it suffices if *there is* a minimal model satisfying them.

This question is not treated explicitly in [MHS07]. The following two examples are supposed to illustrate the effects of the decision on practical applications and to clarify which variant better matches the intuition.

Example 3.9. Let

$$\begin{aligned}\mathcal{T}_S &:= \{\mathsf{Person} \sqsubseteq \mathsf{Student} \sqcup \mathsf{Employee} \sqcup \mathsf{Unemployed} \sqcup \mathsf{Freelancer} \sqcup \mathsf{Pensioner} \sqcup \mathsf{Prisoner}\},\\ \mathcal{A} &:= \{\mathsf{Person}(\mathsf{John})\},\\ \mathcal{T}_C &:= \{\mathsf{Prisoner} \sqsubseteq \mathsf{Murderer} \sqcup \mathsf{Thief}\}\end{aligned}$$

Under the first alternative, the database state would be rejected, but under the second it would be accepted.

Example 3.10. Let $\mathcal{T}_S$ and $\mathcal{A}$ be as above and

$$\begin{aligned}\mathcal{T}_C := \{&\mathsf{Student} \sqsubseteq \mathsf{Lazy} \sqcup \mathsf{Diligent},\ \mathsf{Employee} \sqsubseteq \mathsf{PublicServant} \sqcup \mathsf{IndustrialEmployee},\\ &\mathsf{Unemployed} \sqsubseteq \mathsf{ShortTerm} \sqcup \mathsf{LongTerm},\ \mathsf{Pensioner} \sqsubseteq \mathsf{Under65} \sqcup \mathsf{Over65},\\ &\mathsf{Prisoner} \sqsubseteq \mathsf{Murderer} \sqcup \mathsf{Thief}\}\end{aligned}$$

Just as in the last example, the database state would be rejected under the first alternative and accepted under the second.

In the first example, the rejection of the database state seems quite strict in view of the fact that apart from disjunctions, only facts that follow from the database are taken into consideration. Meanwhile, the acceptance of the database state in the second example seams quite weak.

In [MHS07], the authors choose the first, stricter condition for the acceptance of the integrity constraints. Thereby, possible modelling errors are detected with a higher probability.

In contrast to the first question, the authors of [MHS07] devote a comprehensive discussion to the handling of quantifiers. Universal quantifiers in the head or existential quantifiers in the body of a rule do not pose any problem. Every rule with a universal quantifier in the body

can be transformed to a rule with an existential quantifier in the head. In the following, three alternatives for handling existential quantifiers in rule heads are discussed.

Beforehand we notice that by the confinement to Herbrand models, the Unique Name Assumption is implicitly made. In order to be compatible with OWL, an equality relation $\approx$ can be taken into account. Then the semantic has to be modified in such a way that only models $\mathcal{S}$ are allowed that fulfill the following equality axioms are allowed [ZCGN$^+$15]. For each n-ary predicate symbol R and every $1 \leq i \leq n$:

$$\forall x_1, \ldots, x_n \ (P(x_1, \ldots, x_n) \rightarrow x_i \approx x_i) \tag{EQ1}$$

$$\forall x, y \ (x \approx y \rightarrow y \approx x) \tag{EQ2}$$

$$\forall x, y, z \ (x \approx y \wedge y \approx z \rightarrow x \approx z) \tag{EQ3}$$

$$\forall x_1, \ldots, x_n, y \ (P(x_1, \ldots, x_n) \wedge x_i \approx y \rightarrow P(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n) \tag{EQ4}$$

Then, the assumption that different constants denote by default different resources becomes manifest in the fact that minimal (also w. r. t. the equality relation) models are looked for.

By the axioms, it is possible to conclude the existence of resources that are not explicitly mentioned in the ABox (*Blank Nodes*). It has to be decided whether

a) only "really minimal" models are considered, i. e. existentially quantified variables are interpreted by new individual if no other individual satisfies the condition,

b) all (minimal) models are considered in which existentially quantified variables represent already known or new individuals,

c) or only such minimal models are considered in which it is assumed that all existentially quantified variables denote different, not already known things (provided the contrary cannot be deduced).

Example 3.11. Let

$$\begin{aligned}
\mathcal{T}_S &:= \{\mathsf{TwoChildrenParent} \sqsubseteq (\exists \mathsf{hasChild}.\top)\},\\
\mathcal{A} &:= \{\mathsf{TwoChildrenParent}(\mathsf{John}),\ \mathsf{hasChild}(\mathsf{John}, \mathsf{Bob})\}\\
\mathcal{T}_C &:= \{\mathsf{TwoChildrenParent} \sqsubseteq\ (\geq 2\, \mathsf{hasChild}.\top)\}.
\end{aligned}$$

According to alternatives a) and b), the database state would be rejected because it is possible that John has only one child, namely Bob. In contrast, it would be accepted according to alternative c) because it is assumed that the existentially quantified child is different from Bob.

Example 3.12. Let now

$$\begin{aligned}
\mathcal{T}_S &:= \{\mathsf{OneChildParent} \sqsubseteq (\exists \mathsf{hasChild}.\top)\},\\
\mathcal{A} &:= \{\mathsf{OneChildParent}(\mathsf{John}),\ \mathsf{hasChild}(\mathsf{John}, \mathsf{Bob})\}\\
\mathcal{T}_C &:= \{\mathsf{OneChildParent} \sqsubseteq\ (\leq 1\, \mathsf{hasChild}.\top)\}.
\end{aligned}$$

This time, the database state would be accepted according to alternative a) since in it is assumed that the existentially quantified child coincides with Bob. According to the alternatives

b) and c), it would however be rejected because (also) the model is considered in which the existentially quantified child is a new unknown child.

The acceptance of the integrity constraint in the first example according to alternative c) is counter-intuitive. This variant has moreover the disadvantage that semantically equivalent axioms can lead to different behaviour. Its advantage is that it is easy to implement: It is obtained by replacing every existential quantifier in a rule head by an own Skolem function symbol f_i and considering minimal Herbrand models for $\mathcal{A} \cup \mathcal{T}_S$ over the signature $(\mathrm{ADOM}, (f_i)_{i \in I}, \mathbf{R})$ where $\mathbf{R}$ is the set of all occurring predicate symbols. Since in minimal Herbrand models (with equality), all terms are (by default) interpreted as different, this produces the behaviour described in alternative c). For this reason, the authors opt for this variant. We will now give the formal definition.

Definition 3.1. An *extended DL-knowledge base* is a triple $(\mathcal{T}_S, \mathcal{T}_C, \mathcal{A})$ consisting of a set $\mathcal{T}_S$ of DL-axioms called the *standard TBox*, a further set $\mathcal{T}_C$ of DL-axioms called the *integrity constraints*, and a set $\mathcal{A}$ of atomic facts called the *ABox*.

For a set $\mathcal{K}$ of DL-axioms, let $\pi(\mathcal{K})$ be an equivalent first-order logic formula. Such a translation is explained in detail in [SCM03].

Definition 3.2. For a set $\mathcal{K}$ of DL-axioms, let $\mathsf{sk}(\mathcal{K})$ be the formula obtained by Skolemizing $\pi(\mathcal{K})$. We write

$$\mathsf{sk}(\mathcal{K}) \models_{\mathsf{MM}} \psi$$

for a formula ψ if $\mathcal{S} \vDash \psi$ for every minimal Herbrand model $\mathcal{S}$ of $\mathsf{sk}(\mathcal{K})$ (together with the equality axioms) over the signature of $\mathsf{sk}(\mathcal{K})$ (and the symbol $\approx$).

Definition 3.3. Let $\mathcal{K} := (\mathcal{T}_S, \mathcal{T}_C, \mathcal{A})$ be an extended DL-knowledge base. The integrity constraints $\mathcal{T}_C$ are *satisfied* in $\mathcal{K}$ if

$$\mathsf{sk}(\mathcal{T}_S \cup A) \models_{\mathsf{MM}} \pi(\mathcal{T}_C).$$

This definition is not without formal elegance, however, example 3.11 shows that minimal cardinality constraints cannot be ensured therewith. This is why in my opinion, it is worthy to consider alternative a) in more detail than it is done in [MHS07]. The idea to instantiate existential rules only if the rule head is not already satisfied is known as *restricted chase* and described in [CGK13, Section 2.5]. The applicability of this idea to integrity constraints could be the object of future research.

3.7 Remarks

We conclude this section with some remarks on the approach of Motik, Horrocks and Sattler presented in [MHS07].

Remark 3.1. A DL-knowledge base $(\mathcal{T}, \mathcal{A})$ according to the usual definition can be seen as special extended DL-knowledge base, namely as $(\mathcal{T}, \emptyset, \mathcal{A})$.

Remark 3.2. If $\mathcal{T}_C \subset \mathcal{T}_S$ and $\mathcal{T}_S \cup \mathcal{A}$ is consistent, then the integrity constraints $\mathcal{T}_C$ are satisfied. This is a trivial consequence of the definition, however it is essential for the understanding of the role of integrity constraints. It does not make sense to use a condition at the same time as integrity constraint and as a usual TBox-axiom.

Remark 3.3. If the knowledge base satisfies the integrity constraints, it makes no difference if $\mathcal{T}_S$ or $\mathcal{T}_S \cup \mathcal{T}_C$ is used as TBox for answering queries without negated conditions.

Proof. Let U be the Herbrand universe, $\varphi(x_1, \ldots, x_n)$ a query without negated conditions and $(a_1, \ldots, a_n) \in U^n$. Owing to the monotonicity of reasoning under the Open World Assumption, it is clear that

$$((\mathcal{T}_S \cup \mathcal{A}) \models \varphi[a_1/x_1, \ldots, a_n/x_n]) \quad \Longrightarrow \quad ((\mathcal{T}_S \cup \mathcal{T}_C \cup \mathcal{A}) \models \varphi[a_1/x_1, \ldots, a_n/x_n]) .$$

For the converse, we assume that $(\mathcal{T}_S \cup \mathcal{A}) \not\models \varphi[a_1/x_1, \ldots, a_n/x_n]$, i. e. that there is a model of $\mathcal{T}_S \cup \mathcal{A}$ in which $\varphi[a_1/x_1, \ldots, a_n/x_n]$ does not hold. So by Herbrand's theorem, there is also such an Herbrand model $\mathcal{S}$. Let $\mathcal{S}_{\min}$ be a minimal Herbrand model with $\mathcal{S}_{\min} \subset \mathcal{S}$. Then, $\mathcal{S}_{\min} \not\models \varphi[a_1/x_1, \ldots, a_n/x_n]$ as well because every positive statement in $\mathcal{S}_{\min}$ also holds in $\mathcal{S}$. This minimal model satisfies by assumption the integrity constraints, i. e. $\mathcal{S}_{\min} \models \mathcal{T}_C$. Thus, $\mathcal{S}_{\min}$ is a model of $\mathcal{T}_S \cup \mathcal{T}_C \cup \mathcal{A}$ which does not satisfy $\varphi[a_1/x_1, \ldots, a_n/x_n]$. □

Remark 3.4. By taking into consideration the constraints from $\mathcal{T}_C$, the performance of the reasoner can be influenced. The authors of [MHS07] give an example in which the constraint from $\mathcal{T}_C$ is badly manageable so that it yields a performance gain to omit it. But surely there are also examples in which the axioms in $\mathcal{T}_C$ shorten the reasoning process.

4 Answering OWL-queries with the aid of a Datalog reasoner

The aim of PAGOdA is to speed up the calculation of a correct and complete answer for queries to an OWL 2 knowledge base. Thereto the system resorts to an existing fully-fledged OWL 2 reasoner (HermiT) and a Datalog reasoner (RDFox). The main idea is to use the efficient Datalog reasoner as much as possible and to fall back to the OWL reasoner only for a small relevant fragment of the ontology. More precisely, for a given knowledge base $\mathcal{K}$, two knowledge bases $\mathcal{L}(\mathcal{K})$ and $\mathcal{U}(\mathcal{K})$ that fall into the Datalog fragment of OWL 2 are computed, such that for every query Q the answer set for $\mathcal{L}(\mathcal{K})$ provides a lower bound and the answer set of $\mathcal{U}(\mathcal{K})$ provides an upper bound to the answer set for $\mathcal{K}$. After calculating these bounds for every query Q, the tuples in the gap are checked w. r. t. a relevant fragment $\mathcal{K}^Q_{\text{rel}}$ by the fully-fledged OWL 2 reasoner. This is concretized by the following algorithm.

1. Check consistence of the knowledge base.
2. Use a Datalog reasoner to compute the answers L^Q to the query w. r. t. $\mathcal{L}(\mathcal{K})$ and U^Q w. r. t. $\mathcal{U}(\mathcal{K})$ providing a lower and an upper bound respectively. If $L^Q = U^Q$, return L^Q. Otherwise, let $G^Q = U^Q \setminus L^Q$.
3. Compute a relevant fragment $\mathcal{K}^Q_{\text{rel}}$.

4. For each tuple $\mathbf{a} \in G^Q$, check with the aid of the fully fledged OWL 2 reasoner whether $\mathcal{K}^Q_{\text{rel}} \models Q(\mathbf{a})$. If not, set $G^Q := G^Q \setminus \{\mathbf{a}\}$.
5. Return $L^Q \cup G^Q$.

4.1 Semantics of Datalog and OWL

As already described in the introduction, the semantic of OWL is based on the well-defined notion of entailment. A tuple $\mathbf{a}$ is in the answer to a query if the knowledge base entails that $\mathbf{a}$ satisfies the condition of the query. Contrary to this, the semantic of Datalog is based on minimal Herbrand models. However, the minimal model of a knowledge base need not be unique if there are rules with negated atoms in the body or disjunction in the head. This is the reason why the definition of the semantic has to be stated more precisely in the case of negative atoms in the body. A restriction to stable models reduces the set of possible models but still does not guarantee uniqueness. The definition of well-founded semantics circumvents this problem by allowing a 3-valued model that only specifies what is definitely true and what is false in every *minimal* model. Yet, if there is a unique stable model, then the well-founded model coincides therewith. In order to compute the well-founded model, one alternately computes an upper bound and a lower bound to all minimal models. In the first step, one assumes all negative atoms to hold and derives the corresponding minimal model M_1 that overestimates every minimal model of $\mathcal{K}$. In the second step, one assumes only the negative atoms to hold whose corresponding positive atoms do not belong to M_1 and calculates the minimal model M_2 for this. Then M_2 underestimates all minimal models of $\mathcal{K}$. Overestimating the conclusion in one step can thus lead to an underestimation in a later step. Then the odd sequence elements form a falling sequence of upper bounds and the even sequence elements form a raising sequence of lower bounds, so both converge, yielding the well-founded model. It is possible that for two logically equivalent knowledge bases, the stable models or the well-founded model differ.

The two issues do not occur in the semantic of OWL.

1. In OWL, the reasoning is monotonic, so overestimating the conclusion in one step always leads to an overestimation (of the known facts) in later steps. This is because negated preconditions are not assumed to hold by default.
2. Logical equivalence transformations of the knowledge base do not change the answers to a query.

The second point is the reason why every OWL rule can be translated to a set of predicate logic formulas of the form

$$\forall \mathbf{x} \left(\bigwedge_{j=1}^{n} B_j(\mathbf{x}) \rightarrow \bigvee_{i=1}^{m} \exists \mathbf{y}_i \varphi_i(\mathbf{x}, \mathbf{y}_i) \right) \tag{1}$$

where $\mathbf{x}, \mathbf{y}_i$ are vectors of variables, $B_j(\mathbf{x})$ are positive atoms and $\varphi_j(\mathbf{x}, \mathbf{y}_j)$ are conjunctions of positive atoms. The universal quantifier is usually omitted. *Positive Datalog rules* are exactly the rules with $m = 1$ that do not contain an existential quantifier. A knowledge base is called a *Datalog knowledge base* if all rules are positive Datalog rules.

In this chapter, only positive conjunctive queries are considered, i. e. queries given by a formula $Q(\mathbf{x}) = \exists \mathbf{y} \varphi(\mathbf{x}, \mathbf{y})$ where φ is a conjunction of atoms. The set of all answers to Q w. r. t. the knowledge base $\mathcal{K}$ is denoted by $\mathsf{Ans}(Q, \mathcal{K})$.

In the case of a Datalog knowledge base, the minimal model is unique. The reason why Datalog can be used to answer queries in OWL is that for a conjunctive query over a Datalog knowledge base, the answer w. r. t. the minimal model semantic coincides with the answer w. r. t. the OWL semantic.

During the whole section we consider the following university example.

Example 4.1. Let the knowledge base $\mathcal{K}$ consist of the following axioms

$$\mathsf{Professor}(x) \to \mathsf{StaffMember}(x) \tag{R_1}$$
$$\mathsf{Assistant}(x) \to \mathsf{StaffMember}(x) \tag{R_2}$$
$$\mathsf{Assistant}(x) \wedge \mathsf{teaches}(x, y) \to \mathsf{Student}(y) \tag{R_3}$$
$$\mathsf{Professor}(x) \wedge \mathsf{teaches}(x, y) \wedge \mathsf{Professor}(y) \to \bot \tag{R_4}$$
$$\mathsf{StaffMember}(x) \to \exists y(\mathsf{teaches}(x, y)) \tag{R_5}$$
$$\mathsf{StaffMember}(x) \to \mathsf{Professor}(x) \vee \mathsf{Assistant}(x) \tag{R_6}$$

and the following facts

Professor(Schöbel)	Assistant(Behrends)	teaches(Schöbel, Kaufmann)
Professor(Seppänen)	StaffMember(Merz)	teaches(Seppänen, Merz)
	Student(Kaufmann)	teaches(Merz, Kaesberg)

and let the query be

$$Q(x) = \exists y(\mathsf{teaches}(x, y) \wedge \mathsf{Student}(y))$$

whose answer should be $\{\mathsf{Schöbel}, \mathsf{Behrends}, \mathsf{Merz}\}$

4.2 Computing the lower bound

A direct way to generate a Datalog knowledge base $\mathcal{L}(\mathcal{K})$ providing a lower bound, is to omit all non-Datalog rules from $\mathcal{K}$.

Example 4.2. In the university example, $\mathcal{L}(\mathcal{K})$ contains the rules (R_1)-(R_4) and all atoms. The answer L^Q to this database is then $\{\mathsf{Schöbel}\}$.

Some improvements are possible:

Program Shifting

Example 4.3. From the facts Professor(Seppänen) and teaches(Seppänen, Merz) and rule (R_4), one can deduce that Merz is not a professor. From this conclusion, the fact StaffMember(Merz) and rule (R_6), one can deduce that Merz is an assistant. Because of the fact teaches(Merz, Kaesberg) and rule (R_3), it is derivable that Kaesberg is a student so that Merz is supposed to be an answer to the query.

For the reasoning in the last example, one needs the disjunctive rule (R_6). However, there is no reasoning by case involved. The necessary rules can be expressed in Datalog by dint of a new predicate: One can add the predicate $\overline{\mathsf{Professor}}$, meaning that somebody is not a professor, and the rules $\mathsf{Professor}(x) \wedge \mathsf{teaches}(x, y) \rightarrow \overline{\mathsf{Professor}}(y)$ and $\mathsf{StaffMember}(x) \wedge \overline{\mathsf{Professor}}(x) \rightarrow \mathsf{Assistant}(x)$. Using this extended Datalog fragment, one obtains the lower bound $\{\mathsf{Schöbel}, \mathsf{Merz}\}$.

With this approach, one obtains a Datalog program in polynomial time, called the *shift* of $\mathcal{K}$, that provides a stronger lower bound. The general transformation is described in [ZCGN$^+$15, Section 4.1]. It only takes into account the rules without existential quantifier. But even for the purely disjunctive rules, the transformed program does not allow for conclusions that can only be drawn by reasoning by case, so it is not an equivalent formulation.

The combined approach for $\mathcal{ELHO}^r_\bot$

The combined approach was originally developed in [SMH13] to compute the answer to a query w. r. t. a $\mathcal{ELHO}^r_\bot$-knowledge base with the help of a Datalog reasoner. The given knowledge base $\mathcal{K}$ need not fall inside $\mathcal{ELHO}^r_\bot$. However, by applying the approach to the $\mathcal{ELHO}^r_\bot$-fragment of $\mathcal{K}$, it provides a lower bound.

The algorithm for calculating the answer w. r. t. an $\mathcal{ELHO}^r_\bot$-knowledge base $\mathcal{K}'$ in turn first transforms the knowledge base to a Datalog program $\tilde{\mathcal{U}}(\mathcal{K}')$ giving an upper bound and then uses a filtration algorithm to discard spurious answers.

Combining Program Shifting with the combined approach

The two approaches can be combined as follows:

1. Construct the shift of $\mathcal{K}$ and compute its materialization M_1.
2. Let $\mathcal{K}'$ be the $\mathcal{ELHO}^r_\bot$-fragment of $\mathcal{K}$. Compute $\tilde{\mathcal{U}}(\mathcal{K}')$.
3. Compute the materialization M_2 of $\tilde{\mathcal{U}}(\mathcal{K}') \cup M_1$ providing an upper bound for L^Q.
4. Compute the answers to Q w. r. t. M_2 and filter out the sound answers w. r. t. $\mathcal{K}' \cup M_1$. The result is L^Q.

Since the facts of M_1 have to hold in any model, one can consider them as given atoms in the subsequent calculation of the lower bound. This is done by calculating the complete and correct answer to the $\mathcal{ELHO}^r_\bot$-fragment for $\mathcal{K} \cup M_1$. According to [ZCGN$^+$15], it is important to take into account M_1 in the calculation of M_2. This is because not all the facts from M_1 can be derived from the $\mathcal{ELHO}^r_\bot$-fragment, but together with the $\mathcal{K}'$-rules they can imply further facts. An interesting question not treated in [ZCGN$^+$15] is whether one would profit from applying again step 1 to $\mathcal{K} \cup M_2$ and obtaining M_3, then using $\tilde{U}(\mathcal{K}') \cup M_3$ in step 3 to obtain M_4 and so on.

4.3 Computing the upper bound

The main idea of computing the upper bound $\mathcal{U}(\mathcal{K})$ is to replace a rule with a disjunction $\bigvee_{i=1}^{m} \exists \mathbf{y}_i \varphi_i(\mathbf{x}, \mathbf{y}_i)$ in the head by m rules with heads $\exists \mathbf{y}_1 \varphi_1(\mathbf{x}, \mathbf{y}_1), \ldots, \exists \mathbf{y}_m \varphi_m(\mathbf{x}, \mathbf{y}_m)$ and to replace existential quantifiers by Skolem constants.

Obviously, the knowledge base $\mathcal{U}'(\mathcal{K})$ obtained by applying these two steps to $\mathcal{K}$ entails $\mathcal{K}$, whence by monotonicity of OWL-reasoning it provides an upper bound to the answer to any query. It can however happen that $\mathcal{U}'(\mathcal{K})$ is inconsistent while $\mathcal{K}$ is consistent. This is not a contradiction to the previous observation because according to the semantic of OWL, in an inconsistent knowledge base, everything is an answer to every query, so it still provides an upper bound. Yet this is unpleasant because the upper bound is trivial and does not yield any useful information. The following example illustrates that in the common case of a disjoint union, $\mathcal{K}$ is consistent while $\mathcal{U}'(\mathcal{K})$ is inconsistent.

Example 4.4. The OWL 2 rule

$$\mathsf{StaffMember}\ \texttt{rdfs:subClassOf}\ \texttt{owl:disjointUnionOf}(\mathsf{Professor}\ \mathsf{Assistant}).$$

is translated to the two DL-axioms

$$\begin{aligned} \mathsf{StaffMember} &\sqsubseteq (\mathsf{Professor} \sqcup \mathsf{Assistant}) \\ \mathsf{Professor} \sqcap \mathsf{Assistant} &\sqsubseteq \bot \end{aligned}$$

which in turn correspond to the predicate logic rules

$$\begin{aligned} \mathsf{StaffMember}(x) &\to \mathsf{Professor}(x) \vee \mathsf{Assistant}(x) \\ \mathsf{Professor}(x) \wedge \mathsf{Assistant}(x) &\to \bot. \end{aligned}$$

By applying the replacement described above, one obtains the three rules

$$\begin{aligned} \mathsf{StaffMember}(x) &\to \mathsf{Professor}(x) \\ \mathsf{StaffMember}(x) &\to \mathsf{Assistant}(x) \\ \mathsf{Professor}(x) \wedge \mathsf{Assistant}(x) &\to \bot. \end{aligned}$$

If there is any atom ensuring the existence of a staffMember, the transformed knowledge base is inconsistent.

For this reason, before applying the two transformation steps described above, every $\bot$ in a rule head is replaced by a new nullary predicate symbol $\bot_s$ with no predefined meaning. We now come to the formal definition. Let as in section 3 $\mathcal{T}$ denote the TBox and $\mathcal{A}$ denote the ABox of $\mathcal{K}$.

Definition 4.1. The *splitting* $\mathsf{split}(r)$ of a rule r of the form (1) (p. 18) is the following set of rules:

- if the head of r is $\bot$, then $\mathsf{split}(r) := \{B_1(\mathbf{x}) \wedge \cdots \wedge B_n(\mathbf{x}) \to \bot_s\}$;
- otherwise, $\mathsf{split}(r) := \{B_1(\mathbf{x}) \wedge \cdots \wedge B_n(\mathbf{x}) \to \exists \mathbf{y}_i \varphi_i(\mathbf{x}, \mathbf{y}_i) \mid 1 \leq i \leq m\}$.

The splitting of the knowledge base $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ is defined as $\mathsf{split}(\mathcal{K}) := (\bigcup_{r \in \mathcal{T}} \mathsf{split}(r)) \cup \mathcal{A}$.

Definition 4.2. The *Datalog strengthening* $\mathcal{U}(\mathcal{K})$ is defined as the *c-Skolemization* of $\mathsf{split}(\mathcal{K})$, i. e. every rule r of the form $B_1(\mathbf{x}) \wedge \cdots \wedge B_n(\mathbf{x}) \to \exists \mathbf{y} \varphi(\mathbf{x}, \mathbf{y})$ is replaced by the rule

$$B_1(\mathbf{x}) \wedge \cdots \wedge B_n(\mathbf{x}) \to \varphi(\mathbf{x}, \mathbf{c}_r)$$

where $\mathbf{c}_r$ is a vector of globally unique fresh constant symbols.

We show the transformation for the example 4.1. The first three rules stay unchanged. Rule (R_4) is transformed in the splitting step to

$$\mathsf{Professor}(x) \wedge \mathsf{teaches}(x, y) \wedge \mathsf{Professor}(y) \to \bot_s. \tag{R_4'}$$

and is not changed by the c-Skolimzation. (R_5) equals is splitting and is altered to

$$\mathsf{StaffMember}(x) \to \mathsf{teaches}(x, c_{R_5}) \tag{R_5'}$$

by the c-Skolemization step, where c_{R_5} is a new constant symbol. Finally (R_6) is split to

$$\mathsf{StaffMember}(x) \to \mathsf{Professor}(x) \tag{R_6^1}$$
$$\mathsf{StaffMember}(x) \to \mathsf{Assistant}(x). \tag{R_6^2}$$

We already observed that splitting up rules with disjunctive heads and c-Skolemizing both strengthen the knowledge base $\mathcal{K}$, i. e. transform $\mathcal{K}$ to a knowledge base entailing $\mathcal{K}$. Thus, these two operations provide an upper bound to any positive query. What is however the effect of replacing $\bot$ by $\bot_s$? In this case, the opposite is true: the replacement results in a weakening of the knowledge base. Since $\{\bot\} \models \{\bot_s\}$, every model satisfying $\mathcal{K}$ gives a model satisfying the transformed knowledge base (by interpreting $\bot_s$ arbitrarily). We will now give a contrived example of a query for which $\mathcal{U}(\mathcal{K})$ does not provide an upper bound.

Example 4.5. The answer to the query $Q() = \bot$ is either the empty set (if the knowledge base is consistent) or the set containing only the empty tuple (if the knowledge base is inconsistent). Consider now the following autodidact knowledge base $\mathcal{K}_{\text{auto}}$ consisting only of the rule (R_4) and the facts $\mathsf{Professor}(\mathsf{Bessel})$ and $\mathsf{teaches}(\mathsf{Bessel}, \mathsf{Bessel})$. Then $\mathcal{U}(\mathcal{K}_{\text{auto}})$ consists of the rule (R_4') and the two given facts. Furthermore $\mathsf{Ans}(Q, \mathcal{K}_{\text{auto}}) = \{()\}$, whereas $\mathsf{Ans}(Q, \mathcal{U}(\mathcal{K}_{\text{auto}})) = \emptyset$ because $\mathcal{U}(\mathcal{K}_{\text{auto}})$ is consistent, having a model $\mathcal{S}$ with $\mathcal{S}(\bot_s) = \top$. So the answer w. r. t. $\mathcal{U}(\mathcal{K}_{\text{auto}})$ is not an upper bound to the answer w. r. t. $\mathcal{K}_{\text{auto}}$.

The example relies on the fact that $\mathcal{K}$ is inconsistent. It turns out that this is the only problematic case: in [ZCGN$^+$15], it is proved that whenever $\mathcal{K}$ is consistent, then for every query Q holds: $\mathsf{Ans}(Q, \mathcal{K}) \subseteq \mathsf{Ans}(Q, \mathcal{U}(\mathcal{K}))$. The reason for this is that the information from the disjointness axioms (rules with head $\bot$) used to derive an answer w. r. t. $\mathcal{K}$ is not needed in $\mathcal{U}(\mathcal{K})$ because of the splitting of the disjunctive rule heads. This is illustrated by the following example.

Example 4.6. In Example 4.1, in order to derive the Answer Merz to the query Q, rule (R_4) is needed to exclude that Merz is a Professor from which can be concluded by rule (R_6) that Merz must be an Assistant. However, in $\mathcal{U}(\mathcal{K})$, it is not necessary to exclude Merz to be a Professor in order to deduce that he is an Assistant because the latter follows directly from rule (R_6^2).

The relation of the answer sets w. r. t. the original knowledge base $\mathcal{K}$, the knowledge base $\mathcal{U}'(\mathcal{K})$ obtained by only splitting up disjunctions and c-Skolemizing, and the Datalog strengthening $\mathcal{U}(\mathcal{K})$ according do definition 4.2 is illustrated in figure 1.

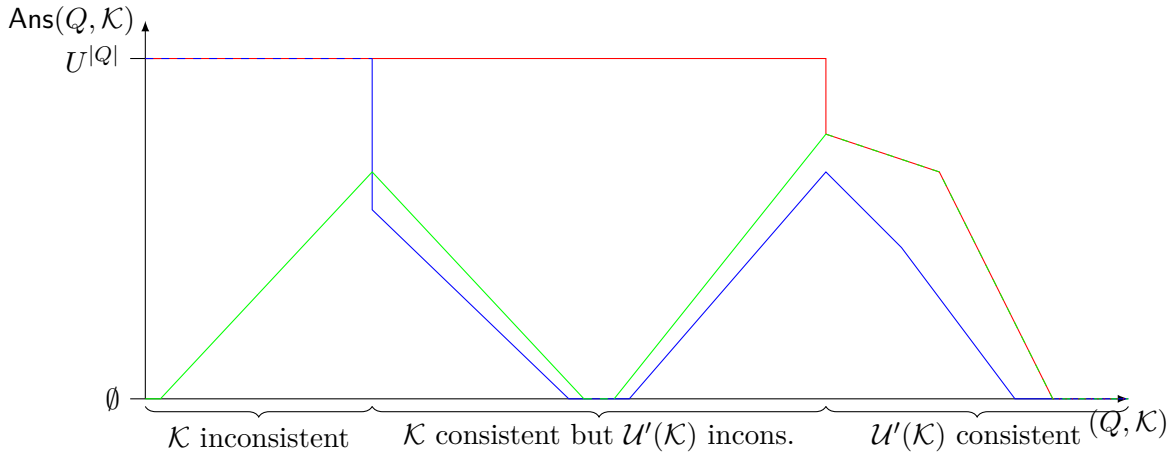

Figure 1: Qualitative relation between $\mathcal{K}$ (blue), $\mathcal{U}'(\mathcal{K})$ (red) and $\mathcal{U}(\mathcal{K})$ (green)

Example 4.7. In example 4.1, the upper bound would lead to the answer $\mathsf{Ans}(Q,\mathcal{U}(\mathcal{K})) = \{\mathsf{Schöbel}, \mathsf{Behrends}, \mathsf{Merz}, \mathsf{Seppänen}\}$, where Seppänen is derived with the aid of rule (R_1) followed by (R_6^2) and (R_3).

Two improvements of the Datalog strengthening are suggested in [ZCGN$^+$15].

1. Replace the c-Skolemization by the so-called *c-chase* similar to the restricted chase [CGK13, Section 2.5] already mentioned in section 3.6.
2. Replace a disjunctive head only by one of its disjuncts, which is selected by a *choice function* taking as input all rules computed so far.

4.4 Computing Relevant Fragments

If, as in the above example, the lower bound and upper bound do not coincide, then the gap tuples have to be checked separately. Instead of checking them w. r. t. the whole knowledge base $\mathcal{K}$, a relevant fragment $\mathcal{K}_{\text{rel}}^Q$ is extracted. For this, different approaches are presented in [ZNGH14] and [ZCGN$^+$15].

- In [ZNGH14], all statements $\alpha \in \mathcal{K}$ are included that are used in an SLD-resolution proof of either $\bot_s$ or $Q(\mathbf{a})$ for some $\mathbf{a} \in G^Q$ w. r. t. the Datalog strengthening $\mathcal{U}(\mathcal{K})$.

- In [ZCGN$^+$15], an algorithm is given that directly determines a set of statements from $\mathcal{K}$ that could occur in a hyperresolution proof of $\bot$ or $Q(\mathbf{a})$ for some $\mathbf{a} \in G^Q$, without using the Datalog strengthening $\mathcal{U}(\mathcal{K})$.

Here, a short summary of the first approach is given.

Definition 4.3. The *relevant fragment* $\mathcal{K}^Q_{\text{rel}}$ of $\mathcal{K}$ to the query Q consists of all statements $\alpha \in \mathcal{K}$ for which exists a $\beta \in \mathcal{U}(\alpha)$ involved in a resolution proof in $\mathcal{U}(\mathcal{K})$ of either $\bot_s$ or of $Q(\mathbf{a})$ for some $\mathbf{a} \in G^Q$.

The following theorem ensures the correctness of the algorithm using this fragment.

Theorem 4.1.

1. *$\mathcal{K}$ is inconsistent if and only if $\mathcal{K}^{\bot_s}_{\text{rel}}$ is inconsistent.*
2. *If $\mathcal{K}$ is consistent, then for every query Q and every $\mathbf{a} \in G^Q$ holds:*

$$\mathcal{K} \vDash Q(\mathbf{a}) \text{ if and only if } \mathcal{K}^q_{\text{rel}} \vDash Q(\mathbf{a}).$$

For both claims the "if"-direction is clear by monotonicity of first-order logic. The other direction is based on the following lemma.

Lemma 1. *Let Q be a query and $\mathbf{a} \in G^Q$. For every statement $\alpha \in \mathcal{K}$ occurring in a resolution proof of $Q(\mathbf{a})$ w. r. t. $\mathcal{K}$, there is a statement $\beta \in \mathcal{U}(\alpha)$ occurring either in a resolution proof of $Q(\mathbf{a})$ or of $\bot_s$ w. r. t. $\mathcal{U}(\mathcal{K})$.*

From this lemma follows that every atom $Q(\mathbf{a})$ with $\mathbf{a} \in G^Q$ that can be derived by resolution from $\mathcal{K}$, can already be derived from $\mathcal{K}^Q_{\text{rel}}$. The theorem ensues thus from the (refutation) completeness and correctness of resolution calculus.

Example 4.8. Consider again example 4.6. In order to derive the answer $Q(\mathsf{Merz})$ w. r. t. $\mathcal{K}$, we need the facts $\mathsf{StaffMember}(\mathsf{Merz}), \mathsf{Professor}(\mathsf{Seppänen})$ and $\mathsf{teaches}(\mathsf{Seppänen}, \mathsf{Merz})$ and the rules (R_5), (R_3), (R_6) and (R_4).

For deriving the same fact w. r. t. $\mathcal{U}(\mathcal{K})$, one only needs the fact $\mathsf{StaffMember}(\mathsf{Merz})$ and the rules (R'_5), (R^2_6) and (R_3). However, to derive $\bot_s$, a possible resolution proof uses the facts $\mathsf{Professor}(\mathsf{Seppänen})$, $\mathsf{StaffMember}(\mathsf{Merz})$ and $\mathsf{teaches}(\mathsf{Seppänen}, \mathsf{Merz})$ together with the rules (R^1_6) and (R'_4).

Remark 4.1. In [ZNGH14], a stronger version of Lemma 1 is provided, namely that for every statement $\alpha \in \mathcal{K}$ occurring in a resolution proof of $Q(\mathbf{a})$, *every* statement $\beta \in \mathcal{U}(\alpha)$ either occurs in a resolution proof in $\mathcal{U}(\mathcal{K})$ of $Q(\mathbf{a})$ or of $\bot_s$. This is however not needed to prove the theorem.

It remains the question how to compute $\mathcal{K}^Q_{\text{rel}}$. This can again be done with the aid of a Datalog reasoner. The idea is to track the resolution proofs in $\mathcal{U}(\mathcal{K})$ with the help of a

- a new predicate $\overline{P}$ for each predicate P occurring in $\mathcal{K}$ which will give the relevant facts,

- a fresh constant c_r for every rule r,
- a fresh unary predicate S which will keep track of the rules involved in resolution proofs.

We assume that the query Q is atomic (or equivalently that Q is added as a rule to $\mathcal{K}$). The tracking knowledge base $\mathsf{track}(\mathcal{U}(\mathcal{K}), Q)$ for $\mathcal{U}(\mathcal{K})$ contains

- each rule and fact in $\mathcal{U}(\mathcal{K})$,
- a fact $\overline{Q}(\mathbf{a})$ for each $\mathbf{a} \in G^Q$ as well as a fact $\overline{\bot}_s$,
- and, for each rule r in $\mathcal{U}(\mathcal{K})$ of the form $B_1(\mathbf{x}) \wedge \cdots \wedge B_n(\mathbf{x}) \to C(\mathbf{x})$, the following new rules:

$$\begin{aligned} &\overline{C}(\mathbf{x}) \wedge B_1(\mathbf{x}) \wedge \cdots \wedge B_n(\mathbf{x}) \to S(c_r) \\ &\overline{C}(\mathbf{x}) \wedge B_1(\mathbf{x}) \wedge \cdots \wedge B_n(\mathbf{x}) \to \overline{B}_i(\mathbf{x}) \qquad\qquad \forall i \in \{1, \ldots, n\}. \end{aligned}$$

The facts $\overline{Q}(\mathbf{a})$ mean that the facts $Q(\mathbf{a})$ occur in their own resolution proofs. The new rules referring to a rule $r \in \mathcal{U}(\mathcal{K})$ mean that, for any predicate symbol C and any $\mathbf{b}$, if $C(\mathbf{b})$ can occur in a resolution proof, and $B_1(\mathbf{b}), \ldots, B_n(\mathbf{b})$ are all true, then these premises and the rule r can also occur in a resolution proof.

4.5 Summarization and answer dependencies

Summarization and the consideration of answer dependencies are exploited to accelerate the examination of the gap tuples.

Summarization

Summarization is described in [FKM$^+$06]. The idea is to identify constants instantiating the same unary predicates (or equivalently belonging to the same classes). This process again yields an upper bound to the answer to the query and can hence be used to exclude candidate answers from G^Q. This idea will now be formalized.

Definition 4.4. Let $\mathcal{K}$ be a knowledge base over the relational signature $\mathbf{R}$ and the constant symbols $\mathbf{C}$. The *type* of $a \in \mathbf{C}$ is the set $T(a) = \{A \in \mathbf{R} \mid A(a) \in \mathcal{K}\}$. For each type T let c_T be a globally unique fresh constant, and let σ be the substitution replacing each $a \in \mathbf{C}$ by $c_{T(a)}$. The knowledge base $\sigma(\mathcal{K})$ obtained by applying σ to every atom and every rule in $\mathcal{K}$ is called the *summary* of $\mathcal{K}$.

Proposition 4.1. *Summarizing provides an upper bound: for every query Q we have* $\sigma(\mathsf{Ans}(Q, \mathcal{K})) = \mathsf{Ans}(\sigma(Q), \sigma(\mathcal{K}))$.

Remark 4.2. Here, the idea of summarization is described in terms of a transformation of the knowledge base. Equivalently, it is possible to describe the idea as a change of the model theory, namely by restricting the set of considered models to those models $\mathcal{S}$ with $\mathcal{S}(a) = \mathcal{S}(b)$ whenever $T(a) = T(b)$. This is a similar idea to the restriction to minimal Herbrand models in definition 3.2.

Answer dependencies

If a tuple $\mathbf{b} = (b_1, \ldots, b_n)$ fulfills every statement of $\mathcal{K}$ that another tuple $\mathbf{a} = (a_1, \ldots, a_n)$ fulfills and if $\mathbf{a}$ is an answer to the positive conjunctive query Q, then $\mathbf{b}$ must also be an answer. To formalize this, let σ be the map that substitutes every a_i by b_i. If then $\sigma(\alpha) \in \mathcal{K}$ for all $\alpha \in \mathcal{K}$, then $\mathcal{K} \models Q(\mathbf{a})$ implies $\mathcal{K} \models Q(\mathbf{b})$.

According to [ZCGN$^+$15, Proposition 7.5], this can be generalized by allowing, instead of the map σ that only substitutes the a_i by b_i, a so-called *endomorphism*, i. e. a mapping h from the set of all constants into itself mapping a_i to b_i and satisfying $\sigma(\alpha) \in \mathcal{K}$ for all $\alpha \in \mathcal{K}$. However, this is not true if the query contains constants as the following example shows.

Example 4.9. Consider the knowledge base

Male(John)	Female(Alice)	hasChild(John, Alice),
Male(Oscar)	Female(Carol)	hasChild(John, Bob)
Male(Bob)		hasChild(Oscar, Carol)

Then the map h: {John, Oscar, Alice, Bob, Carol} $\to$ {John, Oscar, Alice, Bob, Carol} defined by

$$\text{John} \mapsto \text{John} \qquad \text{Oscar} \mapsto \text{John} \qquad \text{Alice} \mapsto \text{Alice} \qquad \text{Bob} \mapsto \text{Bob} \qquad \text{Carol} \mapsto \text{Alice}$$

is an endomorphism. Consider the query

$$Q(x) = \text{hasChild}(x, \text{Carol}).$$

Then Oscar is an answer to the query and h is an endomorphism mapping Oscar to John. Nonetheless, John is not an answer to Q.

4.6 The tool PAGOdA

The authors claim in their paper to have implemented the described techniques in the prototypical system PAGOdA. By consulting the web page `https://www.cs.ox.ac.uk/isg/tools/PAGOdA/` one really finds a downloadable jar archive with this name and a very short user guide. After downloading it and installing the required Java 8, the jar archive still did not run on the test system (Intel® Core™ i3-2328M, neither under Ubuntu 14.04 64 Bit nor under Windows 8.1 64 Bit). The webpage informed us that in this case, we have to download and compile RDFox and insert it into the archive. The attempt to do this failed because the version of RDFox published in the web was incorrect. After contacting the authors of the paper, we received by e-mail an other version of RDFox that could be compiled. However the version published in the web is still erroneous (5.7.).

After inserting the self-compiled JRDFox into PAGOdA, we still had to make some preparations none of which was mentioned in the user guide.

1. Create a folder with the name config, therein create a file called uobm.conf. In this file, some standard parameter values can be specified. With ANSWER= one can specify the standard output file.

2. Remove comments from the query files. PAGOdA cannot handle comments because newline characters are deleted before parsing comments so that the end of a comment is not found.

3. Properties must at least contain a "`:/`".

Although the source code of PAGOdA is available on the web, it is so difficult to compile it that we did not add the ability to process comments to PAGOdA itself but wrote a bash script that uses sed to generate a temporary copy of the query file without comments.

After all these preparations, we had to discover that PAGOdA did not answer correctly any of our test queries for which a rule with disjunction in the head or a rule with an existential quantifier in the head has to be applied. Even for ontologies that fall into the Datalog fragment, some queries were not correctly answered. For example, unqualified cardinality of roles (`owl:minCardinality`) was not taken into account by the reasoner. What works and what does not work with PAGOdA is compiled in detail in the beamer presentation accompanying this seminar paper.

References

[CGK13] Andrea Calì, Georg Gottlob, and Michael Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res. (JAIR)*, 48:115–174, 2013.

[FKM+06] Achille Fokoue, Aaron Kershenbaum, Li Ma, Edith Schonberg, and Kavitha Srinivas. The summary abox: Cutting ontologies down to size. In Isabel F. Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Michael Uschold, and Lora Aroyo, editors, *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006, Proceedings*, volume 4273 of *Lecture Notes in Computer Science*, pages 343–356. Springer, 2006.

[Lev96] Alon Y. Levy. Obtaining complete answers from incomplete databases. In T. M. Vijayaraman, Alejandro P. Buchmann, C. Mohan, and Nandlal L. Sarda, editors, *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India*, pages 402–412. Morgan Kaufmann, 1996.

[LS93] Alon Y. Levy and Yehoshua Sagiv. Queries independent of updates. In Rakesh Agrawal, Seán Baker, and David A. Bell, editors, *19th International Conference on Very Large Data Bases, August 24-27, 1993, Dublin, Ireland, Proceedings.*, pages 171–181. Morgan Kaufmann, 1993.

[MHS07] Boris Motik, Ian Horrocks, and Ulrike Sattler. Bridging the gap between OWL and relational databases. In Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy, editors, *Proceedings of the 16th International*

Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007, pages 807–816. ACM, 2007.

[Mot89] Amihai Motro. Integrity = validity + completeness. *ACM Trans. Database Syst.*, 14(4):480–502, 1989.

[SCM03] Ulrike Sattler, Diego Calvanese, and Ralf Molitor. Relationships with other formalisms. In Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 137–177. Cambridge University Press, 2003.

[SMH13] Giorgio Stefanoni, Boris Motik, and Ian Horrocks. Introducing nominals to the combined query answering approaches for EL. In Marie desJardins and Michael L. Littman, editors, *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*. AAAI Press, 2013.

[ZCGN+15] Yujiao Zhou, Bernardo Cuenca Grau, Yavor Nenov, Mark Kaminski, and Ian Horrocks. PAGOdA: Pay-as-you-go ontology query answering using a datalog reasoner. Technical Report 2244, Department of Computer Science, University of Oxford, March 2015.

[ZNGH14] Yujiao Zhou, Yavor Nenov, Bernardo Cuenca Grau, and Ian Horrocks. Pay-as-you-go ontology query answering using a datalog reasoner. In Meghyn Bienvenu, Magdalena Ortiz, Riccardo Rosati, and Mantas Simkus, editors, *Informal Proceedings of the 27th International Workshop on Description Logics, Vienna, Austria, July 17-20, 2014.*, volume 1193 of *CEUR Workshop Proceedings*, pages 352–364. CEUR-WS.org, 2014.