Combining Existential Rules and Transitivity: Next Steps

Jean-François Baget
Inria, CNRS,
Univ. Montpellier
Montpellier, France

Meghyn Bienvenu
CNRS,
Univ. Paris-Sud
Orsay, France

Marie-Laure Mugnier
Univ. Montpellier,
Inria, CNRS
Montpellier, France

Swan Rocher
Univ. Montpellier,
Inria, CNRS
Montpellier, France

Abstract

We consider existential rules (aka Datalog$\pm$) as a formalism for specifying ontologies. In recent years, many classes of existential rules have been exhibited for which conjunctive query (CQ) entailment is decidable. However, most of these classes cannot express transitivity of binary relations, a frequently used modelling construct. In this paper, we address the issue of whether transitivity can be safely combined with decidable classes of existential rules. First, we prove that transitivity is incompatible with one of the simplest decidable classes, namely aGRD (acyclic graph of rule dependencies), which clarifies the landscape of 'finite expansion sets' of rules. Second, we show that transitivity can be safely added to linear rules (a subclass of guarded rules, which generalizes the description logic DL-Lite$_R$) in the case of atomic CQs, and also for general CQs if we place a minor syntactic restriction on the rule set. This is shown by means of a novel query rewriting algorithm that is specially tailored to handle transitivity rules. Third, for the identified decidable cases, we pinpoint the combined and data complexities of query entailment.

1 Introduction

Ontology-based data access (OBDA) is a new paradigm in data management, which exploits the semantic information provided by ontologies when querying data. Briefly, the notion of a database is replaced by that of a knowledge base (KB), composed of a dataset and an ontology. *Existential rules*, aka Datalog$\pm$, have been proposed to represent ontological knowledge in this context [Calì *et al.*, 2009; Baget *et al.*, 2009; 2011b; Krötzsch and Rudolph, 2011]. These rules are an extension of function-free first-order Horn rules (aka Datalog), that allows for existentially quantified variables in rule heads. The addition of existential quantification allows one to assert the existence of yet unknown entities and to reason about them, an essential feature of ontological languages, which is also at the core of description logics (DLs). Existential rules generalize the DLs most often considered in the OBDA setting, like the DL-Lite and $\mathcal{EL}$ families [Calvanese *et al.*, 2007; Baader, 2003; Lutz *et al.*, 2009] and Horn DLs [Krötzsch *et al.*, 2007].

The fundamental decision problem related to OBDA is the following: is a Boolean conjunctive query (CQ) entailed from a KB? This problem has long been known to be undecidable for general existential rules (this follows e.g., from [Beeri and Vardi, 1981]). Consequently, a significant amount of research has been devoted to the issue of finding decidable subclasses with a good expressivity / tractability tradeoff. It has been observed that most exhibited decidable classes fulfill one of the three following properties [Baget *et al.*, 2011a]: finiteness of a forward chaining mechanism known as the chase, which allows inferences to be materialized in the data (we call such rule sets *finite expansion sets*, *fes*); finiteness of query rewriting into a union of CQs, which allows to the rules to be compiled into the query (*finite unification sets, fus*); treelike shape of the possibly infinite chase, which allows one to finitely encode the result (*bounded-treewidth sets, bts)*. The class of *guarded* rules [Calì *et al.*, 2008] is a well-known class satisfying the latter property.

Known decidable classes are able to express many useful properties of binary relations (e.g., inverses / symmetry) but most of them lack the ability to define a frequently required property, namely *transitivity*. This limits their applicability in key application areas like biology and medicine, for which transitivity of binary relations (especially the ubiquitous 'part of' relation) is an essential modelling construct. The importance of transitivity has long been acknowledged in the DL community [Horrocks and Sattler, 1999; Sattler, 2000], and many DLs support transitive binary relations. While adding transitivity to a DL often does not increase the complexity of CQ entailment (see [Eiter *et al.*, 2009] for some exceptions), it is known to complicate the design of query answering procedures [Glimm *et al.*, 2008; Eiter *et al.*, 2012], due to the fact that it destroys the tree structure of the chase upon which DL reasoning algorithms typically rely. In contrast to the extensive literature on transitivity in DLs, rather little is known about the compatibility of transitivity with decidable classes of existential rules. A notable exception is the recent result of [Gottlob *et al.*, 2013] on the incompatibility of transitivity with guarded rules, which holds even under strong syntactic restrictions (see Section 3).

In this paper, we investigate the issue of whether transitivity can be safely added to some well-known rule classes

and provide three main contributions. First, we show that adding transitivity to one of the simplest *fes* and *fus* classes (namely *aGRD*) makes atomic CQ entailment undecidable, which clarifies the issue for *fes* classes (Theorem 1). Second, we investigate the impact of adding transitivity to *linear* rules, a natural subclass of guarded rules which generalizes the well-known description logic DL-Lite$_R$. We introduce a query rewriting procedure that is sound and complete for all rule sets consisting of linear and transitivity rules (Theorem 2), and which is guaranteed to terminate for atomic CQs, and for arbitrary CQs if the rule set contains only unary and binary predicates or satisfies a certain safety condition, yielding decidability for these cases (Theorem 3). Third, based on a careful analysis of our algorithm, we establish tight bounds on the combined and data complexities of query entailment for the identified decidable cases. The obtained complexities are the lowest that could be expected, namely, NL in data complexity and PSPACE in combined.

Detailed proofs can be found in the accompanying technical report [Baget *et al.*, 2015].

2 Preliminaries

A *term* is a variable or a constant. An *atom* is of the form $p(t_1, \ldots, t_k)$ where p is a predicate of arity k, and the t_i are terms. We consider *(unions of) Boolean conjunctive queries ((U)CQs)*, which are (disjunctions of) existentially closed conjunctions of atoms. Note however that all results can be extended to non-Boolean queries. A CQ is often viewed as the *set* of atoms. An *atomic CQ* is a CQ consisting of a single atom. A *fact* is an atom without variables. A *fact base* is a finite set of facts.

An *existential rule* (hereafter abbreviated to *rule*) R is a formula $\forall \vec{x} \forall \vec{y}(B[\vec{x}, \vec{y}] \rightarrow \exists \vec{z}\ H[\vec{x}, \vec{z}])$ where B and H are conjunctions of atoms, resp. called the *body* and the *head* of R. The variables $\vec{z}$ (resp. $\vec{x}$), which occur only in H (resp. in B and in H) are called *existential* variables (resp. *frontier* variables). Hereafter, we omit quantifiers in rules and simply denote a rule by $B \rightarrow H$. For example, $p(x, y) \rightarrow p(x, z)$ stands for $\forall x \forall y(p(x, y) \rightarrow \exists z(p(x, z)))$. A *knowledge base* (KB) $\mathcal{K} = (\mathcal{F}, \mathcal{R})$ consists of a fact base $\mathcal{F}$ and a finite set of rules $\mathcal{R}$. The *(atomic) CQ entailment* problem consists in deciding whether $\mathcal{K} \models Q$, where $\mathcal{K}$ is a KB viewed as a first-order theory, Q is an (atomic) CQ, and $\models$ denotes standard logical entailment.

Query rewriting relies on a unification operation between the query and a rule head. Care must be taken when handling existential variables: when a term t of the query is unified with an existential variable in a rule head, all atoms in which t occurs must also be part of the unification, otherwise the result is unsound. Thus, instead of unifying one query atom at a time, we have to unify subsets ("pieces") of the query, hence the notion of a piece-unifier defined next. A partition P of a set of terms is said to be *admissible* if no class of P contains two constants; a substitution σ can be obtained from P by selecting an element e_i in each class C_i of P, with priority given to constants, and setting $\sigma(t) = e_i$ for all $t \in C_i$. A *piece-unifier* of a CQ Q with a rule $R = B \rightarrow H$ is a triple $\mu = (Q', H', P_\mu)$, where $Q' \subseteq Q$, $H' \subseteq H$ and P_μ is an admissible partition on the terms of $Q' \cup H'$ such that:

1. $\sigma(H') = \sigma(Q')$, where σ is any substitution obtained from P_μ;
2. if a class C_i in P_μ contains an existential variable, then the other terms in C_i are variables from Q' that do not occur in $(Q \setminus Q')$.

We say that Q' is a *piece* (and μ is a single-piece unifier) if there is no non-empty subset Q'' of Q' such that P_μ restricted to Q'' satisfies Condition 2. From now on, we consider only single-piece unifiers, which we simply call *unifiers*. The *(direct) rewriting* of Q with R w.r.t. μ is $\sigma(Q \setminus Q') \cup \sigma(B)$ where σ is a substitution obtained from P_μ. A *rewriting* of Q w.r.t. a set of rules $\mathcal{R}$ is a CQ obtained by a sequence $Q = Q_0, \ldots, Q_n$ ($n \geq 0$) where for all $i > 0$, Q_i is a direct rewriting of Q_{i-1} with a rule from $\mathcal{R}$. For any fact base $\mathcal{F}$, we have that $\mathcal{F}, \mathcal{R} \models Q$ iff there is a rewriting Q_n of Q w.r.t. $\mathcal{R}$ such that $\mathcal{F} \models Q_n$ [König *et al.*, 2013].

Example 1 *Consider the rule $R = h(x) \rightarrow p(x, y)$ and CQ $Q = q(u) \wedge p(u, v) \wedge p(w, v) \wedge r(w)$. If $p(u, v)$ is unified with $p(x, y)$, then v is unified with the existential variable y, hence $p(w, v)$ has to be part of the unifier. The triple $\mu = (\{p(u, v), p(w, v)\}, \{p(x, y)\}, \{\{x, u, w\}\{v, y\}\}$ is a unifier. The direct rewriting of Q associated with the substitution $\sigma = \{x \mapsto u, w \mapsto u, y \mapsto v\}$ is $h(u) \wedge q(u) \wedge r(u)$.*

We now define some important kinds of rule sets (see e.g., [Mugnier, 2011] for an overview). A model M of a KB $\mathcal{K}$ is called *universal* if for any CQ Q, M is a model of Q iff $\mathcal{K} \models Q$. A rule set $\mathcal{R}$ is a *finite expansion set (fes)* if any KB $(\mathcal{F}, \mathcal{R})$ has a finite universal model. It is a *bounded-treewidth set (bts)* if any KB $(\mathcal{F}, \mathcal{R})$ has a (possibly infinite) universal model of bounded treewidth. It is a *finite unification set (fus)* if, for any CQ Q, there is a finite set S of rewritings of Q w.r.t. $\mathcal{R}$ such that for any fact base $\mathcal{F}$, we have $\mathcal{F}, \mathcal{R} \models Q$ iff there is $Q' \in S$ such that $\mathcal{F} \models Q'$.

A *Datalog* rule has no existential variables, hence Datalog rule sets are *fes*. Other kinds of *fes* rules are considered in the next section. A rule $B \rightarrow H$ is *guarded* if there is an atom in B that contains all the variables occurring in B. Guarded rules are *bts*. A *linear* rule has a body composed of a single atom and does not contain any constant. Linear rules are guarded, hence *bts*, moreover they are *fus*.

As a special case of Datalog rules, we have *transitivity rules*, of the form $p(x, y) \wedge p(y, z) \rightarrow p(x, z)$, which are not *fus*. A predicate is called *transitive* if it appears in a transitivity rule. If $\mathcal{C}$ is a class of rule sets, $\mathcal{C}$*+trans* denotes the class obtained by adding transitivity rules to rule sets from $\mathcal{C}$.

3 Combining *fes* / *fus* and Transitivity

A large hierarchy of *fes* classes is known (see e.g., [Cuenca Grau *et al.*, 2013] for an overview). Beside Datalog, the simplest classes are *weakly-acyclic (wa)* sets, which prevent cyclic propagation of existential variables along predicate positions, and *aGRD* (acyclic Graph of Rule Dependencies) sets, which prevent cyclic dependencies between rules. Datalog is generalized by *wa*, while *wa* and *aGRD* are incomparable. Some classes generalize *wa* by a finer analysis of

variable propagation (up to *super-weakly acyclic (swa)* sets). Most other *fes* classes generalize both *wa* and *aGRD*.

We show that *aGRD+trans* is undecidable even for atomic CQs. Since *aGRD* is both *fes* and *fus*, this negative result also transfers to *fes+trans* and *fus+trans*.

Theorem 1 *Atomic CQ entailment over* aGRD+*trans KBs is undecidable, even with a single transitivity rule.*

Proof: The proof is by reduction from atomic CQ entailment with general existential rules (which is known to be undecidable). Let $\mathcal{R}$ be a set of rules. We first translate $\mathcal{R}$ into an *aGRD* set of rules $\mathcal{R}^a$. We consider the following new predicates: p (which will be the transitive predicate) and, for each rule $R_i \in \mathcal{R}$, predicates a_i and b_i. Each rule $R_i = B_i \rightarrow H_i$ is translated into the two following rules:

- $R_i^1 = B_i \rightarrow a_i(\vec{x}, z_1) \wedge p(z_1, z_2) \wedge p(z_2, z_3) \wedge b_i(z_3)$
- $R_i^2 = a_i(\vec{x}, z_1) \wedge p(z_1, z_2) \wedge b_i(z_2) \rightarrow H_i$

where z_1,z_2 and z_3 are existential variables and $\vec{x}$ are the variables in B_i.

Let $\mathcal{R}^a = \{R_i^1, R_i^2 \mid R_i \in \mathcal{R}\}$, and let $GRD(\mathcal{R}^a)$ be the graph of rule dependencies of $\mathcal{R}^a$, defined as follows: the nodes of $GRD(\mathcal{R}^a)$ are in bijection with $\mathcal{R}^a$, and there is an edge from a node R_1 to a node R_2 if the rule R_2 depends on the rule R_1, i.e., if there is a piece-unifier of the body of R_2 (seen as a CQ) with the head of R_1.

We check that for any $R_i \in \mathcal{R}$, R_i^1 has no outgoing edge and R_i^2 has no incoming edge (indeed the z_j are existential variables). Hence, in $GRD(\mathcal{R}^a)$ all (directed) paths are of length less or equal to one. It follows that $GRD(\mathcal{R}^a)$ has no cycle, i.e., $\mathcal{R}^a$ is aGRD.

Let R^t be the rule stating that p is transitive. Let $\mathcal{R}' = \mathcal{R}^a \cup \{R^t\}$. The idea is that R^t allows to "connect" rules in $\mathcal{R}^a$ that correspond to the same rule in $\mathcal{R}$. For any fact base $\mathcal{F}$ (on the original vocabulary), for any sequence of rule applications from $\mathcal{F}$ using rules in $\mathcal{R}$, one can build a sequence of rule applications from $\mathcal{F}$ using rules from $\mathcal{R}'$, and reciprocally, such that both sequences produce the same fact base (restricted to atoms on the original vocabulary). Hence, for any $\mathcal{F}$ and Q (on the original vocabulary), we have that $\mathcal{F}, \mathcal{R} \models Q$ iff $\mathcal{F}, \mathcal{R}' \models Q$. □

Corollary 1 *Atomic CQ entailment over* fus+*trans or* fes+*trans KBs is undecidable.*

Most known *fes* classes that do not generalize *aGRD* range between Datalog and *swa* (inclusive). It can be easily checked that any *swa* set of rules remains *swa* when transitivity rules are added (and this is actually true for all known classes between Datalog and *swa*).

Proposition 1 *The classes* swa *and* swa+*trans coincide. Hence,* swa+*trans is decidable.*

Proof: It suffices to note that the addition of transitivity rules does not create new edges in the 'SWA position graph' from [Cuenca Grau *et al.*, 2013]. □

It follows that the effect of transitivity on the currently known *fes* landscape is now quite clear, which is not the case for *fus* classes. In the following, we focus on a well-known *fus* class, namely *linear* rules. We show by means of a query rewriting procedure that query entailment over *linear+trans* KBs is decidable in the case of atomic CQs, as well as for general CQs if we place a minor safety condition on the rule set. Such an outcome was not obvious in the light of existing results. Indeed, atomic CQ entailment over *guarded+trans* rules was recently shown undecidable, even when restricted to rule sets that belong to the two-variable fragment, use only unary and binary predicates, and contain only two transitive predicates [Gottlob *et al.*, 2013]. Moreover, inclusion dependencies (a subclass of linear rules) and functional dependencies (a kind of rule known to destroy tree structures, as do transitivity rules) are known to be incompatible [Chandra and Vardi, 1985].

4 Linear Rules and Transitivity

To obtain finite representations of sets of rewritings involving transitive predicates, we define a framework based on the notion of *pattern*.

4.1 Framework

To each transitive predicate we assign a *pattern name*. Each pattern name has an associated *pattern definition* $P := a_1 | \dots | a_k$, where each a_i is an atom that contains the special variables $\#1$ and $\#2$. A *pattern* is either a *standard pattern* $P[t_1, t_2]$ or a *repeatable pattern* $P^+[t_1, t_2]$, where P is a pattern name and t_1 and t_2 are terms. A *union of patterned conjunctive queries* (UPCQ) is a pair $(\mathbb{Q}, \mathbb{P})$, where $\mathbb{Q}$ is a disjunction of conjunctions of atoms and patterns, and $\mathbb{P}$ is a set of pattern definitions that gives a unique definition to each pattern name occurring in $\mathbb{Q}$. A *patterned conjunctive query* (PCQ) $\mathcal{Q}$ is a UPCQ without disjunction. For the sake of simplicity, we will often denote a (U)PCQ by its first component $\mathbb{Q}$, leaving the pattern definitions implicit.

An *instantiation* T of a UPCQ $(\mathbb{Q}, \mathbb{P})$ is a node-labelled tree that satisfies the following conditions:

- the root of T is labelled by $\mathcal{Q} \in \mathbb{Q}$;
- the children of the root are labelled by the patterns and atoms occurring in $\mathcal{Q}$;
- each node that is labelled by a repeatable pattern $P^+[t_1, t_2]$ may be expanded into $k \geq 1$ children labelled respectively by $P[t_1, x_1]$, $P[x_1, x_2]$, …, $P[x_{k-1}, t_2]$, where the x_i are fresh variables;
- each node labelled by a standard pattern $P[t_1, t_2]$ may be expanded into a single child whose label is obtained from an atom a in the pattern definition of P in $\mathbb{P}$ by substituting $\#1$ (resp. $\#2$) by t_1 (resp. t_2), and freshly renaming the other variables.

For brevity, we will often refer to nodes in an instantiation using their labels.

The *instance* associated with an instantiation is the PCQ obtained by taking the conjunction of the labels of its leaves. An instance of a UPCQ is an instance associated with one of its instantiations. An instance is called *full* if it does not contain any pattern, and we denote by $full(\mathbb{Q}, \mathbb{P})$ the set of full instances of $(\mathbb{Q}, \mathbb{P})$.

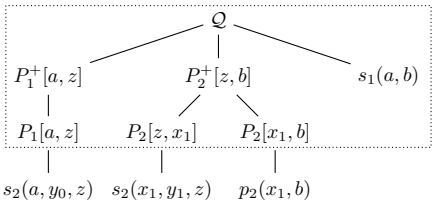

Figure 1: Instantiations of a PCQ

Example 2 *Let $(\mathcal{Q},\mathbb{P})$ be a PCQ, where $\mathcal{Q} = P_1^+[a,z] \wedge P_2^+[z,b] \wedge s_1(a,b)$ and $\mathbb{P}$ contains the pattern definitions: $P_1 := p_1(\#1,\#2)|s_2(\#1,y,\#2)$ and $P_2 := p_2(\#1,\#2)|s_2(\#2,y,\#1)$.*

Two instantiations of $\mathcal{Q}$ are displayed in Figure 1. The smaller instantiation (within the dotted lines) gives rise to the (non-full) instance $Q_1 = P_1[a,z] \wedge P_2[z,x_1] \wedge P_2[x_1,b] \wedge s_1(a,b)$. By expanding the three nodes labelled by patterns according to the definitions in $\mathbb{P}$, we may obtain the larger instantation (occupying the entire figure), whose associated instance $Q_2 = s_2(a,y_0,z) \wedge s_2(x_1,y_1,z) \wedge p_2(x_1,b) \wedge s_1(a,b)$ is a full instance for $(\mathcal{Q},\mathbb{P})$.

A UPCQ $(\mathbb{Q},\mathbb{P})$ can be translated into a set of Datalog rules $\Pi_{\mathbb{P}}$ and a UCQ $Q_{\mathbb{Q}}$ as follows. For each definition $P := a_1(\vec{t_1})|\ldots|a_k(\vec{t_k})$ in $\mathbb{P}$, we create the transitivity rule $p^+(x,y) \wedge p^+(y,z) \rightarrow p^+(x,z)$ and the rules $a_i(\vec{t_i}) \rightarrow p^+(\#1,\#2)$ $(1 \leq i \leq k)$. The UCQ $Q_{\mathbb{Q}}$ is obtained from $\mathbb{Q}$ by replacing each repeatable pattern $P^+[t_1,t_2]$ by the atom $p^+(t_1,t_2)$. Observe that $\Pi_{\mathbb{P}}$ is non-recursive except for the transitivity rules. The next proposition states that $(\Pi_{\mathbb{P}}, Q_{\mathbb{Q}})$ can be seen as a finite representation of the set of full instances of $(\mathbb{Q},\mathbb{P})$.

Proposition 2 *Let $\mathcal{F}$ be a fact base and $(\mathbb{Q},\mathbb{P})$ be a UPCQ. Then $\mathcal{F},\Pi_{\mathbb{P}} \models Q_{\mathbb{Q}}$ iff $\mathcal{F} \models Q$ for some $Q \in full(\mathbb{Q},\mathbb{P})$.*

A *unifier* $\mu = (Q',H,P_u)$ of a *PCQ* is a unifier of one of its (possibly non-full) instances such that Q' is a set of (usual) atoms. We distinguish two types of unifiers (internal and external), defined next.

Let T be an instantiation, Q be its associated instance, and $\mu = (Q',H,P_u)$ be a unifier of Q. Assume T contains a repeatable pattern $P^+[t_1,t_2]$ that is expanded into $P[u_0,u_1]$, $\ldots$, $P[u_k,u_{k+1}]$, where $u_0 = t_1$ and $u_{k+1} = t_2$. We call $P[u_i,u_{i+1}]$ *relevant for* μ if it is expanded into an atom from Q'. Because we consider only single-piece unifiers (cf. Sec. 2), it follows that if such relevant patterns exist, they form a sequence $P[u_i,u_{i+1}]$,$P[u_{i+1},u_{i+2}]$,$\ldots$, $P[u_{j-1},u_j]$. Terms u_i and u_j are called *external* to $P^+[t_1,t_2]$ w.r.t. μ; the other terms occurring in the sequence are called *internal*. The unifier μ is said to be *internal* if all atoms from Q' are expanded from a single repeatable pattern, and no external terms are unified together or with an existential variable; otherwise μ is called *external*.

Example 3 *Consider Q_2 from Example 2 and the rules $R_1 = s_1(x',y') \rightarrow p_2(x',y')$ and $R_2 = s_1(x',y') \rightarrow s_2(x',y',z')$. The unifier of Q_2 with R_1 that unifies $p_2(x_1,b)$ with $p_2(x',y')$ is internal. The unifier of Q_2 with R_2 that unifies $\{s_2(a,y_0,z), s_2(x_1,y_1,z)\}$ with $s_2(x',y',z')$ is external because it involves two repeatable patterns.*

4.2 Overview of the Algorithm

Our query rewriting algorithm takes as input a CQ Q and a set of rules $\mathcal{R} = \mathcal{R}_L \cup \mathcal{R}_T$, with $\mathcal{R}_L$ a set of linear rules and $\mathcal{R}_T$ a set of transitivity rules, and produces a finite set of Datalog rules and a (possibly infinite) set of CQs. The main steps of the algorithm are outlined below.

Step 1 For each predicate p that appears in $\mathcal{R}_T$, create a pattern definition $P := p(\#1,\#2)$, where P is a fresh pattern name. Call the resulting set of definitions $\mathbb{P}_0$.

Step 2 Let $\mathcal{R}_L^+$ be the result of considering all of the rule bodies in $\mathcal{R}_L$ and replacing every body atom $p(t_1,t_2)$ such that p is a transitive predicate by the repeatable pattern $P^+[t_1,t_2]$.

Step 3 (Internal rewriting) Initialize $\mathbb{P}$ to $\mathbb{P}_0$ and repeat the following operation until fixpoint: select a pattern definition $P \in \mathbb{P}$ and a rule $R \in \mathcal{R}_L^+$ and compute the direct rewriting of $\mathbb{P}$ w.r.t. P and R.

Step 4 Replace in Q all atoms $p(t_1,t_2)$ such that p is a transitive predicate by the repeatable pattern $P^+[t_1,t_2]$, and denote the result by $\mathcal{Q}^+$.

Step 5 (External rewriting) Initialize $\mathbb{Q}$ to $\{\mathcal{Q}^+\}$ and repeat the following operation until fixpoint: choose $\mathcal{Q}_i \in \mathbb{Q}$, compute a direct rewriting of $\mathcal{Q}_i$ w.r.t. $\mathbb{P}$ and a rule from $\mathcal{R}_L^+$, and add the result to $\mathbb{Q}$ (except if it is isomorphic to some $\mathcal{Q}_j \in \mathbb{Q}$).

Step 6 Let $\Pi_{\mathbb{P}}$ be the Datalog translation of $\mathbb{P}$, and let $Q_{\mathbb{Q}}$ be the (possibly infinite) set of CQs obtained by replacing each repeatable pattern $P^+[t_1,t_2]$ in $\mathbb{Q}$ by $p^+(t_1,t_2)$.

The rewriting process in Step 3 is always guaranteed to terminate, and in Section 6, we propose a modification to Step 5 that ensures termination and formulate sufficient conditions that preserve completeness. When $\mathcal{Q}_{\mathbb{Q}}$ is finite (i.e., it is a UCQ), it can be evaluated over the fact base saturated by $\Pi_{\mathbb{P}}$, or alternatively, translated into a set of Datalog rules, which can be combined with $\Pi_{\mathbb{P}}$ and passed to a Datalog engine for evaluation. Observe that the construction of $\Pi_{\mathbb{P}}$ is query-independent and can be executed as a preprocessing step.

5 Rewriting Steps in Detail

A PCQ that contains a repeatable pattern has an infinite number of instances. Instead of considering all instances of a PCQ, we consider a finite set of 'instances of interest' for a given rule. Such instances will be used for both the internal and external rewriting steps.

Instances of interest Consider a PCQ $(\mathcal{Q},\mathbb{P})$ and a rule $R \in \mathcal{R}_L^+$ with head predicate p. The *instantiations of interest* of $(\mathcal{Q},\mathbb{P})$ w.r.t. R are constructed as follows. For each repeatable pattern $P_i^+[t_1,t_2]$ in $\mathcal{Q}$, let $a_1^i,\ldots,a_{n_i}^i$ be the atoms in the definition of P_i with predicate p. If $n_i > 0$, then expand $P_i^+[t_1,t_2]$ into k standard patterns, where $0 < k \leq min(arity(p),n_i) + 2$, and expand each of these standard patterns in turn into some a_ℓ^i. An *instance of interest* is the instance associated with an instantiation of interest.

Example 4 *Reconsider $\mathcal{Q}$, Q_2 and R_2 from Examples 2 and 3. Q_2 is not an instance of interest of $\mathcal{Q}$ w.r.t. R_2 since $P_2[x_1, b]$ is expanded into $p(\#1, \#2)$ whereas the head predicate of R_2 is s_2. If we expand $P_2[x_1, b]$ with $s_2(\#2, y, \#1)$ instead, we obtain the instance of interest $Q_3 = s_2(a, y_0, z) \wedge s_2(x_1, y_1, z) \wedge s_2(b, y_2, x_1) \wedge s_1(a, b)$.*

We next show that the set of unifiers computed on the instances of interest of a PCQ 'captures' the set of unifiers computed on all of its instances.

Proposition 3 *Let $(\mathcal{Q}, \mathbb{P})$ be a PCQ and $R \in \mathcal{R}_L^+$. For every instance Q of $(\mathcal{Q}, \mathbb{P})$ and unifier μ of Q with R, there exist an instance of interest Q' of $(\mathcal{Q}, \mathbb{P})$ w.r.t. R and a unifier μ' of Q' with R such that μ' is more general [1] than μ.*

5.1 Internal Rewriting

Rewriting w.r.t. internal unifiers is performed 'inside' a repeatable pattern, independently of the other patterns and atoms in the query. We will therefore handle this kind of rewriting in a query-independent manner by updating the pattern definitions.

To find all internal unifiers between instances under a repeatable pattern $P^+[t_1, t_2]$ and a rule head $H = p(\ldots)$, one may think that it is sufficient to consider each atom a_i in P's definition and check if there is an internal unifier of a_i with H. Indeed, this suffices when predicates are binary: in an internal unifier, t_1 and t_2 are unified with distinct variables, which cannot be existential; thus, the terms in H are frontier variables, and a piece must consist of a single atom. If the arity of p is greater than 2, the other variables can be existential, so it may be possible to unify a path of atoms from P's definition, but not a single such atom (see next example).

Example 5 *Let $R = s(x, y) \rightarrow r(z_1, x, z_2, y)$ and $P := r(\#2, \#1, x_0, x_1) \mid\ r(\#1, x_2, \#2, x_3) \mid\ r(x_4, x_5, \#1, \#2)$. There is no internal unifier of an atom in P's definition with $H = r(z_1, x, z_2, y)$. However, if we expand $P^+[t_1, t_2]$ into a path $P[t_1, y_0]P[y_0, y_1]P[y_1, t_2]$, then expand the ith pattern of this path into the ith atom in P's definition, the resulting instance can be unified with H by an internal unifier (with the partition $\{\{z_1, y_0, x_4\}, \{x, t_1, x_2, x_5\}, \{z_2, x_0, y_1\}, \{y, x_1, x_3, t_2\}\}$).*

Fortunately, we can bound the length of paths to be considered using both the arity of p and the number of atoms with predicate p in P's definition, allowing us to use instances of interest introduced earlier.

A *direct rewriting* $\mathbb{P}'$ of a set of pattern definitions $\mathbb{P}$ w.r.t. a pattern name P and a rule $R = B \rightarrow H \in \mathcal{R}_L^+$ is the set of pattern definitions obtained from $\mathbb{P}$ by updating P's definition as follows. We consider the PCQ $(\mathcal{Q} = P^+[x, y], \mathbb{P})$. We select an instance of interest Q of $\mathcal{Q}$ w.r.t. R, an internal unifier μ of Q with H, and a substitution σ associated with μ that preserves the external terms. Let B' be obtained from $\sigma(B)$ by substituting the first (resp. second) external term by $\#1$ (resp. $\#2$). If B' is an atom, we add it to P's definition. Otherwise, B' is a repeatable pattern of the form $S^+[\#1, \#2]$ or $S^+[\#2, \#1]$. Let f be a bijection on $\{\#1, \#2\}$: if B' is of the form $S^+[\#1, \#2]$, f is the identity, otherwise f permutes $\#1$ and $\#2$. For all s_i in the definition of S, we add $f(s_i)$ to P's definition.

Note that the addition of an atom to a pattern definition is up to *isomorphism* (with $\#1$ and $\#2$ treated as distinguished variables, i.e., $\#1$ and $\#2$ are mapped to themselves).

Example 6 *Reconsider R, μ, and the definition of P from Example 5. Performing a direct rewriting w.r.t. P using R and μ results in adding the atom $s(\#1, \#2)$ to P's definition.*

Proposition 4 *Let $(\mathcal{Q}, \mathbb{P})$ be a PCQ where $P^+[t_1, t_2]$ occurs and $R \in \mathcal{R}_L^+$. For any instance Q of $(\mathcal{Q}, \mathbb{P})$, any classical direct rewriting $\mathcal{Q}'$ of Q with R w.r.t. to a unifier internal to $P^+[t_1, t_2]$, and any $Q' \in full(\mathcal{Q}', \mathbb{P})$, there exists a direct rewriting $\mathbb{P}'$ of $\mathbb{P}$ w.r.t. P and R such that $(\mathcal{Q}, \mathbb{P}')$ has a full instance that is isomorphic to Q'.*

5.2 External Rewriting

Let $(\mathcal{Q}, \mathbb{P})$ be a PCQ, $R \in \mathcal{R}_L^+$, T be an instantiation of interest of $(\mathcal{Q}, \mathbb{P})$ w.r.t. R, Q be the instance associated with T, and $\mu = (Q', H, P)$ be an external unifier of Q with R. From this, several direct rewritings of $\mathcal{Q}$ w.r.t. $\mathbb{P}$ and R can be built. First, we mark all leaves in T that either have the root as parent or are labelled by an atom in Q', and we restrict T to branches leading to a marked leaf. Then, we consider each instantiation T_i that can be obtained from $\mathcal{Q}$ as follows. Replace each repeatable pattern $P^+[t_1, t_2]$ that has $k > 0$ children in T by one of the following:

(i) $P^+[t_1, x_1] \wedge X[x_1, x_2] \wedge P^+[x_2, t_2]$,

(ii) $P^+[t_1, x_1] \wedge X[x_1, t_2]$,

(iii) $X[t_1, x_2] \wedge P^+[x_2, t_2]$,

(iv) $X[t_1, t_2]$,

where $X[v_1, v_2]$ is a sequence $P[v_1, y_1], P[y_1, y_2], \ldots, P[y_{k-1}, v_2]$. Let $\mathcal{Q}_i$ be the instance associated with T_i.

If $P[x', y']$ in T has child $a(\vec{t})$, expand in T_i the corresponding $P[x, y]$ into $a(\rho(\vec{t}))$ where $\rho = \{x' \mapsto x, y' \mapsto y\}$. If $\mu' = (\rho(Q'), H, \rho(P))$ is still a unifier of $\mathcal{Q}_i$ with H, we say that $\mathcal{Q}_i$ is a *minimally-unifiable instance* of $\mathcal{Q}$ w.r.t. μ. In this case, $\mathcal{Q}'_i = \mu'(Q_i) \setminus \mu'(H) \cup \mu'(B)$ is a *direct rewriting* of $\mathcal{Q}$ w.r.t. $\mathbb{P}$ and R.

Example 7 *Reconsider Q_3 and R_2, and let $\mu = (\{s_2(a, y_0, z), s_2(x_1, y_1, z)\}, H_2, \{\{a, x_1, x'\}, \{y_0, y_1, y'\}, \{z, z'\}\})$. First, we consider the instantiation that generated Q_3, and we remove the node labelled $P_2[x_1, b]$ and its child $s_2(b, y_2, x_1)$, since the latter atom is not involved in μ. Next will replace the repeatable pattern $P_1^+[a, z]$ (resp. $P_2^+[z, b]$) using one of the four cases detailed above, and we check whether μ' (obtained from μ) is still a unifier. We obtain in this manner the following minimally-unifiable instances: $\mathcal{Q}_1 = P_1^+[a, x_2] \wedge s_2(x_2, y_0, z) \wedge s_2(x_1, y_1, z) \wedge P_2^+[x_1, b] \wedge s_1(a, b)$, and $\mathcal{Q}_2 = s_2(a, y_0, z) \wedge s_2(x_1, y_1, z) \wedge P_2^+[x_1, b] \wedge s_1(a, b)$. Finally, we rewrite $\mathcal{Q}_1$ and $\mathcal{Q}_2$*

[1] Consider unifiers $\mu = (Q, H, P_\mu)$ and $\mu' = (Q', H, P_{\mu'})$, and let σ (resp. σ') be a substitution associated with P_μ (resp. $P_{\mu'}$). We say that μ' is *more general* than μ if there is a substitution h from $\sigma'(Q')$ to $\sigma(Q)$ such that $h(\sigma'(Q')) \subseteq \sigma(Q)$ (i.e., h is a homomorphism from $\sigma'(Q')$ to $\sigma(Q)$), and for all terms x and y in $Q' \cup H$, if $\sigma'(x) = \sigma'(y)$ then $\sigma(h(x)) = \sigma(h(y))$.

into: $\mathcal{Q}'_1 = P_1^+[a, x'] \wedge s_1(x', y') \wedge P_2^+[x', b] \wedge s_1(a, b)$ *and* $\mathcal{Q}'_2 = s_1(a, y') \wedge P_2^+[a, b] \wedge s_1(a, b)$.

Proposition 5 *Let* $(\mathcal{Q}, \mathbb{P})$ *be a PCQ and* $R \in \mathcal{R}_L^+$. *For every* $Q \in full(\mathcal{Q}, \mathbb{P})$ *and every classical direct rewriting* Q' *of* Q *with* R *w.r.t. an external unifier, there is a direct rewriting* $\mathcal{Q}'$ *of* $\mathcal{Q}$ *w.r.t.* $\mathbb{P}$ *and* R *that has an instance isomorphic to* Q'.

6 Termination and Correctness

To establish the correctness of the query rewriting algorithm, we utilize Propositions 2, 4 and 5.

Theorem 2 *Let* Q *be a CQ,* $(\mathcal{F}, \mathcal{R})$ *be a linear+trans KB, and* $(\Pi_{\mathbb{P}}, Q_{\mathbb{Q}})$ *be the (possibly infinite) output of the algorithm. Then:* $\mathcal{F}, \mathcal{R} \models Q$ *iff* $\mathcal{F}, \Pi_{\mathbb{P}} \models Q'$ *for some* $Q' \in Q_{\mathbb{Q}}$.

Regarding termination, we observe that Step 3 (internal rewriting) must halt since every direct rewriting step adds a new atom (using a predicate from $\mathcal{R}_L^+$) to a pattern definition, and there are finitely many such atoms, up to isomorphism.

By contrast, Step 5 (external rewriting) need not halt, as the rewritings may grow unboundedly in size. Thus, to ensure termination, we will modify Step 5 to exclude direct rewritings that increase rewriting size. Specifically, we identify the following 'problematic' minimally-unifiable instances:

- Q' is composed of atoms expanded from a single pattern $P^+[t_1, t_2]$, $\mu'(t_1) = \mu'(t_2)$, and $P^+[t_1, t_2]$ is replaced as in case (i), (ii) or (iii).
- Q' is obtained from the expansion of repeatable patterns, a term t of Q is unified with an existential variable of the head of the rule, t appears only in repeatable patterns of form $P_i^+[t_i, t]$ (resp. $P_i^+[t, t_i]$), and all these repeatable patterns are rewritten as in case *(ii)* $P_i^+[t_i, t'_i] \wedge X[t'_i, t]$ (resp. as in case *(iii)* $X[t, t'_i] \wedge P_i^+[t'_i, t_i]$).

We will call a direct rewriting *excluded* if it is based on such a minimally-unifiable instance; otherwise, it is *non-excluded*.

Example 8 *The rewriting* $\mathcal{Q}'_1$ *from Example 7 is excluded because it is obtained from the minimally-unifiable instance* Q_1 *in which the repeatable patterns* $P_1^+[a, z]$ *is expanded as in case* (ii) *and* $P_2^+[z, b]$ *as in case* (iii), *and* z *is unified with the existential variable* z'.

Proposition 6 *Let* $(\mathcal{Q}, \mathbb{P})$ *be a PCQ and* $R \in \mathcal{R}_L^+$. *If* $\mathcal{Q}'$ *is a non-excluded direct rewriting of* $\mathcal{Q}$ *with* R, *then* $|\mathcal{Q}'| \leq |\mathcal{Q}|$.

Let us consider the 'modified query rewriting algorithm' that is obtained by only performing non-excluded direct rewritings in Step 5. This modification ensures termination but may comprise completeness. However, we can show that the modified algorithm is complete in the following key cases: when the CQ is atomic, when there is no specialization of a transitive predicate, or when all predicates have arity at most two. By further analyzing the latter case, we can formulate a safety condition, defined next, that guarantees completeness for a much wider class of rule sets.

Safe rule sets We begin by defining a specialization relationship between predicates. A predicate q is a *direct specialization* of a binary predicate p on positions $\{\vec{i}, \vec{j}\}$ ($\vec{i} \neq \emptyset, \vec{j} \neq \emptyset$) if there is a rule of the form $q(\vec{u}) \rightarrow p(x, y)$ such that $\vec{i}$ (resp. $\vec{j}$) contains those positions of $\vec{u}$ that contain the term x (resp. y). It is a *specialization* of p on positions $\{\vec{i}, \vec{j}\}$ if (a) it is a direct specialization of p on positions $\{\vec{i}, \vec{j}\}$, or (b) there is a rule of the form $q(\vec{u}) \rightarrow r(\vec{v})$ such that $r(\vec{v})$ is a specialization of p on positions $\{\vec{k}, \vec{l}\}$ and the terms occurring in positions $\{\vec{k}, \vec{l}\}$ of $\vec{v}$ occur in positions $\{\vec{i}, \vec{j}\}$ of $\vec{u}$ with $\vec{i} \neq \emptyset$ and $\vec{j} \neq \emptyset$. We say that q is a *pseudo-transitive predicate* if it is a specialization of at least one transitive predicate.

We call a linear+*trans* rule set *safe* if it satisfies the following *safety condition*: for every pseudo-transitive predicate q, there exists a pair of positions $\{i, j\}$ with $i \neq j$ such that for all transitive predicates p of which q is a specialization on positions $\{\vec{i}, \vec{j}\}$, either $i \in \vec{i}$ and $j \in \vec{j}$, or $i \in \vec{j}$ and $j \in \vec{i}$.

Note that if we consider binary predicates, the safety condition is always fulfilled. Then, specializations correspond exactly to the subroles considered in DLs.

Example 9 *Let* $R_1 = s_1(x, x, y) \rightarrow p_1(x, y)$, $R_2 = s_2(x, y, z) \rightarrow p_2(x, y)$, $R_3 = s_1(x, y, z) \rightarrow s_2(z, x, y)$, *and* p_1 *and* p_2 *be two transitive predicates.*

The following specializations have to be considered: s_1 *is a direct specialization of* p_1 *on positions* $\{\{1, 2\}, \{3\}\}$, s_2 *is a direct specialization of* p_2 *on positions* $\{\{1\}, \{2\}\}$, s_1 *is a specialization of* p_2 *on positions* $\{\{3\}, \{1\}\}$. *We then have two pseudo-transitive predicates:* s_1 *and* s_2. *By choosing the pair* $\{1, 3\}$ *for* s_1 *and* $\{1, 2\}$ *for* s_2, *we observe that* $\{R_1, R_2, R_3\}$ *satisfies the safety condition.*

If we replace R_3 *by* $R_4 = s_1(x, y, z) \rightarrow s_2(x, y, z)$, s_1 *is a specialization of* p_2 *on positions* $\{\{1\}, \{2\}\}$, *and* $\{R_1, R_2, R_4\}$ *is not safe.*

Theorem 3 *The modified query rewriting algorithm halts. Moreover, Theorem 2 (soundness and completeness) holds for the modified algorithm if either the input CQ is atomic, or the input rule set is safe.*

7 Complexity

A careful analysis of our query rewriting algorithm allows us to pinpoint the worst-case complexity of atomic CQ entailment over linear+*trans* KBs, and general CQ entailment over safe linear+*trans* KBs. As usual, we consider two complexity measures: *combined complexity* (measured in terms of the size of the whole input), and *data complexity* (measured in terms of the size of the fact base). The latter is often considered more relevant since the fact base is typically significantly larger than the rest of the input.

With regards to data complexity, we show completeness for NL (non-deterministic logarithmic space), which is the same complexity as in the presence of transitivity rules alone.

Theorem 4 *Both (i) atomic CQ entailment over linear+trans KBs and (ii) CQ entailment over safe linear+trans KBs are* NL*-complete in data complexity.*

Regarding combined complexity, we show that the addition of transitivity rules does not increase the complexity of query entailment for the two considered cases.

Theorem 5 *Both (i) atomic CQ entailment over linear+trans KBs and (ii) CQ entailment over safe linear+trans KBs are* PSPACE*-complete in combined complexity.*

8 Conclusion

In this paper, we made some steps towards a better understanding of the interaction between transitivity and decidable classes of existential rules. We obtained an undecidability result for aGRD+*trans*, hence for *fes*+*trans* and *fus*+*trans*. More positively, we established decidability (with the lowest possible complexity) of atomic CQ entailment over linear+*trans* KBs and general CQ entailment for safe linear+*trans* rule sets. The safety condition was introduced to ensure termination of the rewriting mechanism when predicates of arity more than two are considered (rule sets which use only unary and binary predicates are trivially safe). We believe the condition can be removed with a much more involved termination proof.

In future work, we plan to explore the effect of transitivity on decidable rule classes that are incomparable with linear rules, namely frontier-one rules, a *bts* class that has close connections to Horn DLs, and two *fus* classes: domain-restricted and sticky rule sets [Baget *et al.*, 2011a; Calì *et al.*, 2010].

Acknowledgements

This work was supported by ANR project PAGODA (contract ANR 12 JS02 007 01).

References

[Baader, 2003] F. Baader. Terminological cycles in a description logic with existential restrictions. In *Proc. of IJCAI*, pages 325–330, 2003.

[Baget *et al.*, 2009] J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat. Extending decidable cases for rules with existential variables. In *Proc. of IJCAI*, pages 677–682, 2009.

[Baget *et al.*, 2011a] J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat. On rules with existential variables: Walking the decidability line. *Art. Intell. (AIJ)*, 175(9-10):1620–1654, 2011.

[Baget *et al.*, 2011b] J.-F. Baget, M.-L. Mugnier, S. Rudolph, and M. Thomazo. Walking the complexity lines for generalized guarded existential rules. In *Proc. of IJCAI*, pages 712–717, 2011.

[Baget *et al.*, 2015] J.-F. Baget, M. Bienvenu, M.-L. Mugnier, and S. Rocher. Combining existential rules and transitivity: Next steps. abs/1504.07443, 2015.

[Beeri and Vardi, 1981] C. Beeri and M. Y. Vardi. The implication problem for data dependencies. In *Proc. of ICALP*, volume 115 of *LNCS*, pages 73–85. Springer, 1981.

[Calì *et al.*, 2008] A. Calì, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. In *Proc. of KR*, pages 70–80, 2008.

[Calì *et al.*, 2009] A. Calì, G. Gottlob, and T. Lukasiewicz. A general Datalog-based framework for tractable query answering over ontologies. In *Proc. of PODS*, pages 77–86, 2009.

[Calì *et al.*, 2010] A. Calì, G. Gottlob, and A. Pieris. Query answering under non-guarded rules in Datalog+/-. In *Proc. of RR*, pages 1–17, 2010.

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

[Chandra and Vardi, 1985] A. K. Chandra and M. Y. Vardi. The implication problem for functional and inclusion dependencies is undecidable. *SIAM J. Comput.*, 14(3):671–677, 1985.

[Cuenca Grau *et al.*, 2013] B. Cuenca Grau, I. Horrocks, M. Krötzsch, C. Kupke, D. Magka, B. Motik, and Z. Wang. Acyclicity notions for existential rules and their application to query answering in ontologies. *J. Art. Intell. Res. (JAIR)*, 47:741–808, 2013.

[Eiter *et al.*, 2009] T. Eiter, C. Lutz, M. Ortiz, and M. Simkus. Query answering in description logics with transitive roles. In *Proc. of IJCAI*, pages 759–764, 2009.

[Eiter *et al.*, 2012] T. Eiter, M. Ortiz, M. Simkus, T.-K. Tran, and G. Xiao. Query rewriting for Horn-$\mathcal{SHIQ}$ plus rules. In *Proc. of AAAI*, 2012.

[Glimm *et al.*, 2008] B. Glimm, C. Lutz, I. Horrocks, and U. Sattler. Conjunctive query answering for the description logic $\mathcal{SHIQ}$. *J. Artif. Intell. Res. (JAIR)*, 31:157–204, 2008.

[Gottlob *et al.*, 2013] G. Gottlob, A. Pieris, and L. Tendera. Querying the guarded fragment with transitivity. In *Proc. of ICALP*, volume 7966 of *LNCS*, pages 287–298. Springer, 2013.

[Horrocks and Sattler, 1999] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *J. Log. Comput.*, 9(3):385–410, 1999.

[König *et al.*, 2013] M. König, M. Leclère, M.-L. Mugnier, and M. Thomazo. On the exploration of the query rewriting space with existential rules. In *Proc. of RR*, pages 123–137, 2013.

[Krötzsch and Rudolph, 2011] M. Krötzsch and S. Rudolph. Extending decidable existential rules by joining acyclicity and guardedness. In *Proc. of IJCAI*, pages 963–968, 2011.

[Krötzsch *et al.*, 2007] M. Krötzsch, S. Rudolph, and P. Hitzler. Complexity boundaries for Horn description logics. In *Proc. of AAAI*, pages 452–457. AAAI Press, 2007.

[Lutz *et al.*, 2009] C. Lutz, D. Toman, and F. Wolter. Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In *Proc. of IJCAI*, pages 2070–2075, 2009.

[Mugnier, 2011] M.-L. Mugnier. Ontological query answering with existential rules. In *Proc. of RR*, pages 2–23, 2011.

[Sattler, 2000] U. Sattler. Description logics for the representation of aggregated objects. In *Proc. of ECAI*, pages 239–243, 2000.