Conjunctive Query Answering for Directional Rules

Technical Report 3026
AIFB, Kalrsruhe Institute of Technology

Markus Krötzsch[1] and Sebastian Rudolph[2]

[1] Dep. of Computer Science, University of Oxford, UK, markus.kroetzsch@cs.ox.ac.uk
[2] Institute AIFB, Karlsruhe Institute of Technology, DE, sebastian.rudolph@kit.edu

Abstract. This paper introduces Directional Rules, a new extension of Datalog with existential quantifiers in rule heads in the spirit of formalisms like tuple-generating dependencies, Datalog+/− and ∀∃-rules that attracted new interest recently. As opposed to known decidable classes of such existential rules, Directional Rules support complex join conditions as required for expressing transitivity. Nonetheless, the new language suggests surprisingly simple algorithms for answering a broad class of conjunctive queries in polynomial time, regarding both data and program complexity. In contrast, answering unrestricted conjunctive queries is undecidable, and we propose further restrictions and more complex algorithms for recovering decidability in the general case. Besides their immediate use for data integration and data exchange, Directional Rules are of particular interest since they can capture large real-world ontologies that could hitherto be modelled in description logics only, even though they are mostly used in database-driven applications.

1 Introduction

Datalog – the logical language of function-free first-order Horn clauses – has been widely studied and applied in both in the field of deductive databases and that of knowledge representation and reasoning. However, pure Datalog can only be used to make statements about the active domain (i.e., the set of constants or database individuals that are given *a priori*), whereas the capability to derive the existence of new individuals – a feature commonly called *value invention* [10, 2] – is considered a necessary prerequisite for the deployment of rule based paradigms for ontological knowledge representation [31].

Therefore, extensions of Datalog have been introduced which feature value invention. In the database community, the according logical fragment is usually referred to as *tuple generating dependencies* (TGDs, see, e.g., [1]) and has been employed in the area of data exchange and data integration [21]. The *Datalog+/–* formalism [12] which is based on TGDs has been shown to be able to capture lightweight ontological languages such as the DL-Lite family [15].

Coming from a parallel strand of research concerned with graph-based knowledge representation [17, 33] $\forall\exists$-rules have been suggested for knowledge representation and reasoning on (hyper)graph data repositories [7], a notion which coincides with TGDs.

In spite of these efforts of establishing a rule-based foundation for expressive knowledge representation and reasoning, ontological modelling applications today are more often based on description logics (DLs), a family of knowledge representation languages that has gained prominence as the semantic foundation of the W3C Web Ontology Language OWL (see [5, 22] for textbook introductions). This might be surprising given that ontologies are increasingly used in large scale and data-centric scenarios, where a close integration with database technology is desirable.

A related example is the medical ontology SNOMED Clinical Terms that includes about 300,000 concepts [24]. While DLs are used for developing SNOMED, the ontology is actually deployed as a materialised database that is shipped to customers in CSV file format. But the development version of SNOMED uses expressive means that are not available in Datalog, e.g., to say that a particular class of pharmaceutical products has an active ingredient from a particular class of chemical compounds. Since this statement refers to *some* (not fully specified) element from a class of compounds, the according axioms in SNOMED CT merely require the existence of a suitable element.

Besides this, SNOMED CT also uses a few simple rules, such as the one that states that, if a procedure acts directly on some substance which in turn has a particular active ingredient, then the procedure acts on the active ingredient:

$$\mathsf{direct\text{-}subst(x,y)}, \mathsf{active\text{-}ingredient(y,z)} \rightarrow \mathsf{direct\text{-}subst(x,z)} \tag{1}$$

While this might suggest an approach based on Datalog and TGDs, this direct combination turns out to be too expressive for efficient use in practice: even simple query answering problems are undecidable for this language. In contrast, SNOMED CT is actually covered by the light-weight ontology language OWL EL for which simple queries can be answered in polynomial time. Yet, existing proposals for decidable fragments of TGDs typically exclude rules like (1) and many other join-expressions such as transitivity.

These observations motivate the present paper. Drawing inspiration from the description logic $\mathcal{EL}^{++}$ that is underlying OWL EL, we define a new decidable TGD language of *Directional Rules*. Argument positions in predicates are partitioned in inputs and outputs to enforce a certain direction of dependencies between existing values and newly invented ones. This form of directedness is indirectly enforced in some DLs, but specifying it as an explicit and general paradigm allows Directional Rules to be significantly more expressive than DLs. In particular, unlike DLs, Directional Rules are not restricted to relations of one or two parameters, and can easily accommodate arbitrary Datalog rules if the interaction with TGDs is suitably restricted.

Answering simple queries for Directional Rules is surprisingly simple: we show that, in essence, one can safely replace existentially quantified variables by fresh constants. This reduces inferencing to classical Datalog and allows existing tools and algorithms to be used. It also allows us to obtain various complexity results, especially polynomial data complexity. This simplicity is in stark contrast to the case of general

conjunctive queries that do not respect the requirements of directionality: checking their entailment is undecidable.

To address this issue, we further restrict our language to Simple Directional Rules, and adopt existing methods from description logics to decide conjunctive query entailment. Simple Directional Rules are indeed closely related to the description logic $\mathcal{EL}$. We provide various undecidability results for conjunctive querying in the general case to motivate this strong restriction. While complexities for query answering are adequately – NP data complexity and PSpace combined complexity given reasonable assumptions – the query answering algorithm that we use is of no practical utility due to a high degree of non-determinism. Our results are still interesting since they show query decidability for a TGD language that is outside any of the known decidable fragments.

We begin by providing some basic notation and a short survey of related approaches in Section 2. Directional Rules are defined in Section 3, and a method for answering atomic and directional queries for them is presented in Section 4. General queries are discussed in Section 5, where we introduce Simple Directional Rules, and in Section 6, where a query answering algorithm is presented. We finish with some further related work and some outlook in Section 7.

2 Existentional Rules and Their Decidable Fragments

We now provide the basic notions of the logical framework we consider, followed by an overview of a number of important approaches in this area.

Definition 1. *Consider a* signature $\langle \mathbf{I}, \mathbf{R}, \mathbf{V} \rangle$ *consisting of a finite set of* constant symbols $\mathbf{I}$*, a finite set of* relation symbols $\mathbf{R}$*, and an infinite set of* variable names $\mathbf{V}$*, all of which are mutually disjoint. A function* $\mathsf{ar} : \mathbf{R} \to \mathbb{N}$ *associates a natural number* $\mathsf{ar}(s)$ *with each relation symbol* $r \in \mathbf{R}$ *that defines the (unique)* arity *of* r*. The set of* positions *of a relation symbol* r *is the set* $\Pi_r = \{1, \ldots, \mathsf{ar}(r)\}$.

A term *is a variable* $x \in \mathbf{V}$ *or a constant* $c \in \mathbf{I}$*. An* atom *is a formula of the form* $r(t_1, \ldots, t_n)$ *if* $t_1, \ldots, t_n$ *are terms, and* $r \in \mathbf{R}$ *is a relation symbol with* $\mathsf{ar}(r) = n$*. A* tuple generating dependency *(TGD) is a formula of the form*

$$\forall \boldsymbol{x}.(B_1 \wedge \ldots \wedge B_k \to \exists \boldsymbol{y}.H_1 \wedge \ldots \wedge H_l),$$

where $B_1, \ldots, B_k, H_1, \ldots, H_l$ *are atoms all of whose variables are in the scope of some quantifier, and where no variable occurs more than once in* $\boldsymbol{x}, \boldsymbol{y}$*. We use sets of atoms as a convenient notation for conjunctions of atoms. A* Datalog rule *is a TGD that has no existential quantifiers. A TGD with* $k = 0$ *is called a* fact*, and a TGD with* $l = 0$ *is called a* constraint*. The premise of a TGD – which we will sometimes simply refer to as rule if there is no danger of confusion – is called the* rule body *while the conclusion is called the* rule head*. Since all variables in TGDs are quantified, we will often omit the explicit preceding universal quantifier.*

Clearly the rule language hereby introduced is a syntactic fragment of first-order predicate logic and we also consider it under the according semantics.

Definition 2. *Let Σ be a set of rules. We call Σ* satisfiable *if it has a model according to the standard semantics of first-order logic. Two rule sets Σ and Σ' are* equisatisfiable *if either both or none of them is satisfiable. A* boolean conjunctive query *(BCQ) is a set of atoms. We say that a BCQ Q is entailed by Σ, if $\exists \boldsymbol{x}.Q$ (with $\boldsymbol{x}$ containing all variables occurring in Q) is a logical consequence of Σ according to standard FOL semantics. In particular, we refer to the case $|Q| = 1$ as* simple *or* fact entailment*.*

It has long been known that checking satisfiability and BCQ entailment for unrestricted TGDs is undecidable [16, 9] even with very strong restrictions on the vocabulary or the number of rules [7]. Therefore, a large body of work has been devoted to the identification specific types of TGDs which are decidable and still allow for sufficient expressiveness. A generic tool for establishing decidability results is the *chase* introduced in [29] and extended to query containment in [25]. Intuitively the chase procedure starts with a given set of factual data (ground facts) and "applies" TGDs in a production rule style by introducing new domain elements whenever required by an existentially quantified variable in a rule head. In general, termination of this procedure cannot be guaranteed and an infinite set of new domain elements and facts may be created.

Many of the decidable TGD classes come about by establishing properties about the chase they create. Finiteness of the chase is one possible straightforward criterion for decidability. In [7], a TGD set with this property is referred to as *finite extension set*, but it is also shown that this criterion is undecidable in general. However, several sufficient conditions on TGD sets for chase-finiteness have been identified: Pure Datalog (also referred to as *full implicational dependencies* [16] or *total TGDs* [9]) is an immediate case, as no new domain elements are created at all. Weakly acyclic TGDs [20, 21] constitute a more elaborate way by – roughly speaking – allowing for bounded value generation sequences by taking record of predicate positions. Another way of ensuring finiteness of the chase is to require acyclicity of the *graph of rule dependencies* as introduced in [6].

An even more relaxed condition than finiteness of the chase is that the (possibly infinite) chase enjoys a variant of the bounded treewidth property. TGDs of that kind are referred to as *bounded treewidth sets* in [7] and their decidability is an immediate consequence of the decidability of classes of first order logic with the bounded treewidth model property [18]. Again TGDs with this property are not recognizable in general, but a variety of sufficient conditions has been established. The definition of *guarded TGDs* – which enjoy this property – has been inspired by the guarded fragment of first-order logic [3]. It has been generalized to *weakly guarded TGDs* [11] and to *frontier-guarded rules* [7], both being subsumed by *weakly frontier-guarded TGDs* [7]. The most expressive currently known bts fragments are that of *greedy bts TGDs* [8] and *glut-guarded TGDs* [27].

Independently of the chase, other decidability criteria can be established by considering rewritings of the query in a backward-chaining manner. In analogy to the finite chase condition, one can define *finite unification sets* [7] as TGDs where this subsequent rewriting procedure terminates resulting in a finite set of rewritten queries. Again, recognizing whether a given TGD set has this property is undecidable, yet the decidability of several TGD fragments such as *atomic-hypothesis rules* and *domain restricted rules*

[7] as well as *linear Datalog+/–* [12] and *sticky sets of TGDs* and *sticky-join sets of TGDs* [13, 14]. Note that being a finite unification set implies first-order rewritability and in turn AC_0 data complexity of BCQ entailment checking.

3 Directional Rules

In this section we introduce a class of rule sets, called Directional Rules. Intuitively, this definition captures the idea of directedness by stipulating for every predicate symbol which positions are considered as input (or source) and which ones as output (or target) positions. As an illustrative example, one may think of directed graphs that are encoded using a binary predicate *edge* where the first position is defined to be input and the second as output.

Definition 3. *Given a relation symbol $r \in \mathbf{R}$, a* mode *for r is a pair $\langle \Pi_{in}, \Pi_{out} \rangle$ of disjoint sets of positions such that $\Pi_{in} \cup \Pi_{out} = \Pi_r$, where Π_{in} is the set of* input positions *and Π_{out} is the set of* output positions*. Accordingly, a term t may be said to appear in an output position (input position) of an atom $r(\boldsymbol{t})$. Irrespective of the arity or order of arguments of r, the notation $r(\boldsymbol{in}|\boldsymbol{out})$ is used to denote the terms in input and output positions of $r(\boldsymbol{t})$. When discussing the extension of r, a tuple $\langle \delta_1, \ldots, \delta_{\mathsf{ar}(n)} \rangle$ of domain elements will be written as $\langle \boldsymbol{\delta_i} | \boldsymbol{\delta_o} \rangle$ where $\boldsymbol{\delta_i}$ ($\boldsymbol{\delta_o}$) is the tuple of all elements of $\langle \delta_1, \ldots, \delta_{\mathsf{ar}(n)} \rangle$ at input (output) positions of r.*

We first consider sets of rules with the following properties:

Definition 4. *Consider a signature $\langle \mathbf{I}, \mathbf{R}, \mathbf{V} \rangle$ for which a unique mode has been assigned to each relation symbol. A set of rules Σ over $\langle \mathbf{I}, \mathbf{R}, \mathbf{V} \rangle$ is* directional *if the following hold for each rule $\forall \boldsymbol{x}.B[\boldsymbol{x}] \rightarrow \exists \boldsymbol{y}.H[\boldsymbol{x}, \boldsymbol{y}]$:*

(1) Variables in input positions of H do not occur in output positions in B.
(2) If an existentially quantified variable occurs in an input position of an atom in H, then it is the only input position of this atom and all other variables in that atom are existentially quantified as well.
(3) No variable in B occurs in more than one output position.
(4) There is a strict order $\prec$ on variables in B such that $x \prec y$ holds whenever x occurs on an input position of an atom where y occurs on an output position.

As an example for a directional rule set, consider Fig. 1. Mode assignments to the used predicates are indicated as described in Definition 3. It can be readily checked that the given set of TGDs is indeed directional. It becomes immediately clear that the chase will become infinite in general, as the procedure will generate parents of persons which are again persons. On the other hand, joins of many kinds occur in the program, as well as projections etc. A closer inspection of the rule set reveals that it is not a *bounded treewidth set* (cf. [7]) since – given an initial individual of type *Person* in the database – the chase contains a substructure isomorphic to $(\mathbb{N}, <)$, w.r.t. the predicate *Ancstr*, which does not allow for a tree decomposition of bounded tree width; consequently it lies outside a number of concrete decidable classes of TGDs as discussed in Section 2.

$$\forall x.Person(x|) \rightarrow \exists y,z.Parents(x|y,z) \wedge Woman(y|) \wedge Man(z|)$$
$$\forall x.Man(x|) \rightarrow Person(x|)$$
$$\forall x.Woman(x|) \rightarrow Person(x|)$$
$$\forall x,y,z.Parents(x|y,z) \wedge Married(y,z|) \rightarrow LegitimateChild(x|)$$
$$\forall x,y.Married(x,y|) \rightarrow Married(y,x|)$$
$$\forall x,y,z,w.Twin(x,y|) \wedge Parents(x|z,w) \rightarrow Parents(y|z,w)$$
$$\forall x,y,z.Parents(x|y,z) \rightarrow Parent(x|y) \wedge Parent(x|z)$$
$$\forall x,y.Parent(x|y) \rightarrow Ancstr(x|y)$$
$$\forall x,y.Ancstr(x|y) \wedge Ancstr(y|z) \rightarrow Ancstr(x|z)$$
$$\forall x,y,z,w.Ancstr(x|y) \wedge Ancstr(z|w) \wedge Enemy(y,w|) \rightarrow Enemy(x,z|)$$

Fig. 1. Example directional rule set using | to denote modes as in Definition 3

Likewise the rules do not constitute a *finite unification set* as already witnessed by the single rule stating transitivity of *Ancstr*.

We next review the question how hard it is to find an appropriate assignment of modes for a given set of TGDs.

Theorem 1. *Deciding whether modes can be assigned to relation symbols in such a way that a given set of rules Σ is directional is NP-complete w.r.t. the number of predicates in Σ.*

Proof. Membership in NP is straightforward, as for a given assignment of modes to predicates the criteria for directionality can be checked in polynomial time.

We show NP-hardness by a reduction from 3-SAT to the directionality decision problem. Consider a finite set **Prop** of propositional atoms and a propositional formula $\varphi = \varphi_1 \wedge \ldots \wedge \varphi_n$ with $\varphi_i = a_{i1} \vee a_{i2} \vee a_{i3}$ such that $a_{ij} \in \mathbf{Prop} \cup \{\neg p \mid p \in \mathbf{Prop}\}$. We use binary predicates r_p, $r_{\neg p}$, and s_p for each $p \in \mathbf{Prop}$, and a unary predicate aux. The rule set Σ_φ is defined as follows:

For every propositional atom p contained in φ we let Σ_φ contain the rules

$$r_p(x,y) \wedge r_p(x,z) \rightarrow \tag{2}$$
$$r_{\neg p}(x,y) \wedge r_{\neg p}(x,z) \rightarrow \tag{3}$$
$$\rightarrow \exists x,y.s_p(x,y) \tag{4}$$
$$\mathsf{aux}(z) \wedge s_p(x,y) \rightarrow r_p(z,x) \tag{5}$$
$$\mathsf{aux}(z) \wedge s_p(x,y) \rightarrow r_{\neg p}(z,y) \tag{6}$$
$$r_p(x,y) \wedge r_{\neg p}(y,x) \rightarrow \tag{7}$$

This ensures that every assignment of modes making Σ_φ directional assigns input to the first position of predicates r_p and $r_{\neg p}$ as a consequence of Rules (2) and (3) together with criterion (3). Second, Rule (4) ensures that at least one position of s_p must be assigned output according to criterion (2). Consequently, Rules (5) and (6) together with criterion (1) require that at least one of r_p, $r_{\neg p}$ must have output assigned to its second position. Finally, Rule (7) and criterion (4) rule out the case that both r_p, $r_{\neg p}$ have second position

output, whence we conclude that exactly one of r_p, $r_{\neg p}$ has second position output. So the possible mode combinations are $r_p(i,i), r_{\neg p}(i,o)$ and $r_p(i,o), r_{\neg p}(i,i)$. We will use $r_p(i,i)$ to encode that p is evaluated to true, and $r_{\neg p}(i,i)$ to encode that $\neg p$ is evaluated to true.

Now we create for every $\varphi_i = a_{i1} \vee a_{i2} \vee a_{i3}$ the rule

$$r_{a_{i1}}(x,y) \wedge r_{a_{i2}}(y,z) \wedge r_{a_{i2}}(z,x) \rightarrow \tag{8}$$

Due to criterion (4) and the above established restriction on assignments, rule (8) ensures that at least one of $r_{a_{i1}}$, $r_{a_{i2}}$, $r_{a_{i3}}$ must have two input positions. Hence φ is satisfiable if there is an assignment of modes to the predicates of Σ_φ that makes Σ_φ directional. We can obtain an according truth value assignment as follows: a propositional atom p is assigned *true* exactly if all positions of r_p are assigned input. □

4 Directional Query Answering

We now turn to the problem of answering atomic queries, i.e. conjunctive queries with only one atom. Clearly, any procedure for answering such queries can be applied to all conjunctive queries that are legal as rule bodies of a set of Directional Rules, since one can extend the set with a suitable rule to obtain query results.

Definition 5. *Given a signature with associated modes, a BCQ Q is called* directional *if $\forall \boldsymbol{x}.Q \rightarrow$ is a Directional Rule.*

We will answer atomic queries using the following easy transformation of rule sets:

Definition 6. *For a set of rules Σ, the set of rules $\Gamma(\Sigma)$ contains, for every rule ρ in Σ, a rule ρ' obtained by uniformly replacing each existentially quantified variable that occurs in an output position of a head atom of ρ by a fresh constant symbol.*

For the example of Fig. 1, $\Gamma(\Sigma)$ contains all the TGDs of the original rule set except for the first one, which is replaced by

$$\forall x.Person(x|) \rightarrow Parents(x|w,m) \wedge Woman(w|) \wedge Man(m|),$$

where w, m are fresh constants which have not previously occurred in the rules nor the data. This translation alters the semantics of the theory as it introduces generic parents for all individuals (e.g., $Parents(w,w,m)$ and $Parents(m,w,m)$ are consequences of the newly generated Datalog program), however we will show that it still faithfully reflects certain aspects of the original TGD set and can be used for satisfiability and certain entailment checks.

The main result of this section is as follows:

Theorem 2. *Deciding satisfiability, fact entailment, and body-shaped conjunctive query entailment for Directional Rules is NP-complete for bounded arity and* ExpTime*-complete for unbounded arity. Data complexity is P-complete in both cases.*

To prove this result, we verify the following essential statement.

Lemma 1. *$\Gamma(\Sigma)$ and Σ are equisatisfiable.*

Proof. It is easy to see that $\Gamma(\Sigma) \models \Sigma$, so every model of $\Gamma(\Sigma)$ is also a model of Σ as required.

For the other direction, consider a model $\mathcal{I}$ of Σ, and let D denote the set of all constants that have been introduced when constructing $\Gamma(\Sigma)$ from Σ. A model $\mathcal{J}$ of $\Gamma(\Sigma)$ is constructed as follows. The domain of $\mathcal{J}$ is $\Delta^{\mathcal{J}} := \Delta^{\mathcal{I}} \cup D$ where we assume this to be a disjoint union. We first introduce some auxiliary notions to define the interpretation function.

For a constant $d \in D$ that was introduced in $\Gamma(\Sigma)$ when replacing an existentially quantified variable z in a rule $\forall \boldsymbol{x}.B \to \exists \boldsymbol{y}.H$, we define φ_d to be the formula $\exists \boldsymbol{x}.\exists \boldsymbol{y'}.B \wedge H$ where $\boldsymbol{y'}$ consists of all variables in $\boldsymbol{y}$ other than z. Thus φ_d is a formula with one free variable z, and we may write $\varphi_d[z]$ to emphasise the variable name. We say that an element $\epsilon \in \Delta^{\mathcal{I}}$ is an *$\mathcal{I}$-instance* of an element $\delta \in \Delta^{\mathcal{J}}$ if either $\epsilon = \delta \in \Delta^{\mathcal{I}}$, or $\delta \in D$ and $\mathcal{I}, \{x \mapsto \epsilon\} \models \varphi_\delta[x]$. Analogously, a tuple $\langle \epsilon_1, \dots, \epsilon_n \rangle$ is an *$\mathcal{I}$-instance* of a tuple $\langle \delta_1, \dots, \delta_n \rangle$ if ϵ_i is an $\mathcal{I}$-instance of δ_i for all $i = 1, \dots, n$.

Now the interpretation function of $\mathcal{J}$ can be defined as follows:

- $c^{\mathcal{J}} = c$ for all $c \in D$, and $c^{\mathcal{J}} = c^{\mathcal{I}}$ for all other constants,
- $\langle \boldsymbol{\delta_i} | \boldsymbol{\delta_o} \rangle \in r^{\mathcal{J}}$ if, for all $\mathcal{I}$-instances $\boldsymbol{\epsilon_i}$ of $\boldsymbol{\delta_i}$, there is an $\mathcal{I}$-instance $\boldsymbol{\epsilon_o}$ of $\boldsymbol{\delta_o}$ such that $\langle \boldsymbol{\epsilon_i} | \boldsymbol{\epsilon_o} \rangle \in r^{\mathcal{I}}$, and there is at least one $\mathcal{I}$-instance of $\boldsymbol{\delta_i}$.

To show that $\mathcal{J}$ is a model of $\Gamma(\Sigma)$, consider any rule $\forall \boldsymbol{x}.B[\boldsymbol{x}] \to H'[\boldsymbol{x}] \in \Gamma(\Sigma)$ that was obtained from a rule $\forall \boldsymbol{x}.B[\boldsymbol{x}] \to \exists \boldsymbol{y}.H[\boldsymbol{x}, \boldsymbol{y}] \in \Sigma$. Also, let $\boldsymbol{x_i}$ denote the variables in $\boldsymbol{x}$ that occur in input positions of atoms in B only. Assume that there is a variable assignment $\mathcal{Z}$ for $\mathcal{J}$ such that $\mathcal{J}, \mathcal{Z} \models B$. We iteratively construct a variable assignment $\tilde{\mathcal{Z}}$ for $\mathcal{I}$ by repeating the following step until $\tilde{\mathcal{Z}}$ is defined for all variables in B:

- Select a variable x in B such that, if x occurs in the output position of any atom in B, then $\tilde{\mathcal{Z}}$ has been defined for all variables in input positions of this atom.
- Select a value $\tilde{\mathcal{Z}}(x)$ as follows:
  (1) There is some atom $r[\boldsymbol{t_i}|\boldsymbol{t_o}] \in B$ where x occurs in $\boldsymbol{t_o}$. By the choice of x, ${\boldsymbol{t_i}}^{\mathcal{I},\tilde{\mathcal{Z}}}$ is a tuple of elements in $\Delta^{\mathcal{I}}$. Let $\boldsymbol{\epsilon_o}$ be an $\mathcal{I}$-instance of ${\boldsymbol{t_o}}^{\mathcal{J},\mathcal{Z}}$ such that $\langle {\boldsymbol{t_i}}^{\mathcal{I},\tilde{\mathcal{Z}}}, \boldsymbol{\epsilon_o} \rangle \in r^{\mathcal{I}}$. Then, for all variables y at a position p in $\boldsymbol{t_o}$, set $\tilde{\mathcal{Z}}(y) := \boldsymbol{\epsilon_o}_p$.
  (2) If the above is not the case, set $\tilde{\mathcal{Z}}(x)$ to an arbitrary $\mathcal{I}$-instance of $\mathcal{Z}(x)$.

We need to check that this is a valid definition. First note that $\tilde{\mathcal{Z}}(x)$ is guaranteed to be an $\mathcal{I}$-instance of $\mathcal{Z}(x)$ in all cases of the definition. A suitable variable x can be found in each iteration step due to the forest structure of B. For case (1), the forest structure of B also implies that the atom $r[\boldsymbol{t_i}|\boldsymbol{t_o}] \in B$ is unique, so this part of the definition is deterministic. Moreover, since we have $\mathcal{J}, \mathcal{Z} \models r[\boldsymbol{t_i}|\boldsymbol{t_o}]$, the definition of $\mathcal{J}$ implies that, for all $\mathcal{I}$-instances of ${\boldsymbol{t_i}}^{\mathcal{J},\mathcal{Z}}$, there is an $\mathcal{I}$-instance $\boldsymbol{\epsilon_o}$ of ${\boldsymbol{t_o}}^{\mathcal{J},\mathcal{Z}}$ as required in (1). Hence such an instance can always be found since ${\boldsymbol{t_i}}^{\mathcal{I},\tilde{\mathcal{Z}}}$ is indeed an $\mathcal{I}$-instance of ${\boldsymbol{t_i}}^{\mathcal{J},\mathcal{Z}}$. The variable assignments constructed in this case are consistent since the variables in $\boldsymbol{t_o}$ occur at most once in $\langle \boldsymbol{t_i}|\boldsymbol{t_o} \rangle$. Together with the forest shape of B, this also shows that $\tilde{\mathcal{Z}}$ has not been defined for any of the variables in $\boldsymbol{t_o}$ before. Finally, for item (2) the

required assignment is possible since x must occur in some input position of an atom $A \in B$ with $\mathcal{J}, \mathcal{Z} \models A$, and the definition of $\mathcal{J}$ implies that there is an $\mathcal{I}$-instance for all terms in A. Note that this part of the definition is not deterministic: we can freely chose any $\mathcal{I}$-instance as a value. This is exploited below.

It is easy to see that $\mathcal{I}, \tilde{\mathcal{Z}} \models B$. This is a direct consequence of $\mathcal{J}, \mathcal{Z} \models B$ and the definition of $\tilde{\mathcal{Z}}$.

We can now apply this construction to see that all head atoms in H' are satisfied by $\mathcal{J}$ and $\mathcal{Z}$. Consider any atom $s[\boldsymbol{t_i}|\boldsymbol{t_o}] \in H$ with corresponding atom $s[\boldsymbol{t'_i}|\boldsymbol{t'_o}] \in H'$. Since we observed that $\tilde{\mathcal{Z}}$ can always be constructed, it follows that $\mathcal{I}, \tilde{\mathcal{Z}} \models H$ since $\mathcal{I}$ is a model of Σ. By definition of $\mathcal{I}$-instances, $\langle \boldsymbol{t_i}^{\mathcal{I},\tilde{\mathcal{Z}}} | \boldsymbol{t_o}^{\mathcal{I},\tilde{\mathcal{Z}}} \rangle$ is an $\mathcal{I}$-instance of $\langle \boldsymbol{t'_i}^{\mathcal{J},\mathcal{Z}} | \boldsymbol{t'_o}^{\mathcal{J},\mathcal{Z}} \rangle$, establishing that some such instance exists. This is easy to check using the definition of φ_d that is underlying the notion of $\mathcal{I}$-instances. We show that the remaining conditions in the definition of $s^{\mathcal{J}}$ are also satisfied. Definition 4 allows two cases: either (a) $\boldsymbol{t_i} = \langle z \rangle$ and z as well as all variables from $\boldsymbol{t_o}$ are existentially quantified, or (b) $\boldsymbol{t_i}$ contains no existentially quantified variables at all.

For (a), $\boldsymbol{t'_i} = \langle d \rangle$ and the $\mathcal{I}$-instances ϵ of $d^{\mathcal{J},\mathcal{Z}}$ are such that $\mathcal{I}, \{z \mapsto \epsilon\} \models \varphi_d[z]$, where $\varphi_d[z]$ is of the form $\exists \boldsymbol{x}.\exists \boldsymbol{y'}.B \wedge H$. This requires that for all $\mathcal{I}$-instances ϵ, there is a variable assignment $\mathcal{Z}'$ for the variables in $\boldsymbol{x}$ and $\boldsymbol{y}$ such that $\mathcal{I}, \mathcal{Z}' \models B \wedge H$ and $\mathcal{Z}'(z) = \epsilon$. In particular $\mathcal{I}, \mathcal{Z}' \models s[\langle z \rangle | \boldsymbol{t_o}]$. Thus, for any $\mathcal{I}$-instance ϵ of d, we find an $\mathcal{I}$-instance $\boldsymbol{t_o}^{\mathcal{I},\mathcal{Z}'}$ with $\langle \langle \epsilon \rangle | \boldsymbol{t_o}^{\mathcal{I},\mathcal{Z}'} \rangle\} \in s^{\mathcal{I}}$ (note that the satisfaction of the respective characteristic formulae is straightforward), and thus $\langle \langle d \rangle | \boldsymbol{t'_o}^{\mathcal{J},\mathcal{Z}} \rangle\} \in s^{\mathcal{J}}$ as required. Note that this argument builds on the above result that some $\mathcal{I}$-instance exists.

For (b), Definition 4 ensures that all variables in $\boldsymbol{t_i}$ are universally quantified, and no such variable occurs in output positions of B. Thus item (2) of the above definition of $\tilde{\mathcal{Z}}$ applies for any such variable. Thus, for an arbitrary $\mathcal{I}$-instance $\boldsymbol{\epsilon_i}$ of $\boldsymbol{t'_i}^{\mathcal{J},\mathcal{Z}}$, we can construct $\tilde{\mathcal{Z}}$ such that $\boldsymbol{t_i}^{\mathcal{I},\tilde{\mathcal{Z}}} = \boldsymbol{\epsilon_i}$. Since $\mathcal{I}, \tilde{\mathcal{Z}} \models B$, we find that $\mathcal{I}, \tilde{\mathcal{Z}} \models \exists \boldsymbol{y}.H$. So there is an extension $\tilde{\mathcal{Z}}'$ of $\tilde{\mathcal{Z}}$ to variables in $\boldsymbol{y}$ such that $\mathcal{I}, \tilde{\mathcal{Z}}' \models s[\boldsymbol{t_i}|\boldsymbol{t_o}]$. The existence of this assignment also shows that $\boldsymbol{t_o}^{\mathcal{I},\tilde{\mathcal{Z}}'}$ is an $\mathcal{I}$-instance of $\boldsymbol{t'_o}^{\mathcal{J},\mathcal{Z}}$. Since $\boldsymbol{t_i}^{\mathcal{I},\tilde{\mathcal{Z}}'} = \boldsymbol{\epsilon_i}$ can be an arbitrary $\mathcal{I}$-instance of $\boldsymbol{t'_i}^{\mathcal{J},\mathcal{Z}}$, this establishes the conditions for $\langle \boldsymbol{t'_i}^{\mathcal{J},\mathcal{Z}} | \boldsymbol{t'_o}^{\mathcal{J},\mathcal{Z}} \rangle \in s^{\mathcal{J}}$.

We have thus shown that, for all variable assignments $\mathcal{Z}$ under which $\mathcal{J}$ satisfies the body of a rule in $\Gamma(\Sigma)$, it also satisfies each atom in the respective rule head. □

Proof (of Theorem 2). First note that directional query entailment and fact entailment can be reduced to checking satisfiability by adding a rule $\forall \boldsymbol{x}.Q \rightarrow$ to Σ. In particular, this implies that facts over constants of Σ are entailed by Σ if and only if they are entailed by $\Gamma(\Sigma)$.

Then the data complexity as well as upper complexity bounds for the bounded and unbounded arity case are immediate consequences from respective results for pure Datalog [19] via Lemma 1. Lower complexity bounds arise from the same results by the observation that every Datalog program is a set of directional rules if we define all positions to be input positions.

5 Conjunctive Query Answering

Conjunctive queries that are not restricted as in Section 4 are significantly harder to evaluate against Directional Rules: the problem turns out to be undecidable. We now explain various reasons for this undecidability, motivating the introduction of a restricted fragment of *Simple Directional Rules* for which decidability is regained. In spite of its limitations, the latter set can still not be captured by previous approaches for defining decidable TGD fragments.

The following two theorems illustrate how various features of Directional Rules contribute to undecidability of conjunctive querying.

Theorem 3. *Checking entailment of conjunctive queries for Directional Rules is undecidable, even when restricting to binary predicates with exactly one input and one output position.*

Proof. Strings of a formal language can be represented as linear relational structures with labelled edges. We use this to reduce the (known undecidable) problem of checking the emptiness of the intersection of two context-free languages to query entailment. Consider context-free grammars G_1 and G_2 over an alphabet $\mathcal{A}$. For each alphabet or non-terminal symbol α, let r_α be a binary predicate with first position as its input and second position as its output. For each $\alpha \in \mathcal{A}$, create a rule $\forall x.\exists y.r_\alpha(x, y)$ (ensuring that arbitrary chains of alphabet-encoding relations exist in any model). For each grammar production rule (in either grammar) $T := S\,1 \ldots S\,n$ with T a non-terminal, and $S\,i$ alphabet symbols or non-terminals, create a rule $r_{S\,1}(x_1, x_2), \ldots, r_{S\,n}(x_n, x_{n+1}) \rightarrow r_T(x_1, x_{n+1})$. All of these rules are directional. It is not hard to see that the languages described by G_1 and G_2 share a common word if and only if the query $S_1(x, y), S_2(x, y)$ is entailed, where S_i is the start non-terminal of G_i. □

Similar problems are encountered when studying (conjunctive query answering for) description logics, which also support existential quantifiers and means of encoding context-free grammars [28]. The solution that has been proposed there is to restrict theories in such a way that only regular languages can be encoded. Transitivity axioms and many other expressions are then still allowed. For predicates with more than two arguments, however, it is not clear how regularity is to be formulated, and the following result shows that even very basic recursive axioms are problematic in this case.

Theorem 4. *Checking entailment of conjunctive queries for Directional Rules is undecidable, even with only a single recursive rule that uses a single predicate only.*

Proof. The undecidable *Post Correspondence Problem* (PCP) is as follows: Given two lists of words $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$ over some alphabet $\mathcal{A}$, is there a sequence of numbers $i_1, \ldots, i_k$ $(1 \leq i_j \leq n)$ such that $u_{i_1} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}$? We encode words using predicates r_α as in the proof of Theorem 3. We also use a 4-ary predicate pair with the first two positions as its input. For each $\alpha \in \mathcal{A}$, create a rule $\forall x.\exists y.r_\alpha(x, y)$ (cf. Theorem 3). Now for each pair of words $u_i = U1 \ldots Un$ and $v_i = V1 \ldots Vm$, create a rule $r_{U1}(x_1, x_2), \ldots, r_{Un}(x_n, x_{n+1}), r_{V1}(y_1, y_2), \ldots, r_{Vm}(y_m, y_{m+1}) \rightarrow \mathsf{pair}(x_1, y_1, x_{n+1}, y_{m+1})$. Finally, add the rule $\mathsf{pair}(x_1, y_1, x_2, y_2), \mathsf{pair}(x_2, y_2, x_3, y_3) \rightarrow \mathsf{pair}(x_1, y_1, x_3, y_3)$, the only recursive rule in the rule set. All of the rules are clearly directional. This rule set entails the query $\mathsf{pair}(x, x, y, y)$ if and only if the underlying PCP has a solution. □

The previous proof could be modified to combine the recursive rule with the rules for recognising matching pairs, making them recursive instead. This allows for combining the first two (or last two) arguments of pair into one, showing that basic head (or tail) recursion with ternary predicates suffices for undecidability. To recover decidability, we combine conditions that prevent the encodings of both Theorem 3 and 4.

Definition 7. *A set Σ of Directed Rules is* simple *if it satisfies the following conditions:*

(1) All predicates in Σ are either unary or binary, and their first position is their one and only input position.
(2) Rule bodies are connected, i.e. the graph obtained from a rule body by taking variables as nodes and atoms as (undirected) edges is connected.
(3) Binary head atoms contain at most one existentially quantified variable.
(4) There is a strict partial order $\prec$ between binary predicate symbols with the following property: If a rule head contains a binary atom $r(x, y)$ with universally quantified variables x and y, then either $x = y$ or the body contains a (necessarily unique) path of binary predicates from x to y which must have one of the following forms:
   – $r(x, z), r(z, y)$,
   – $r(x, z_1), s_1(z_1, z_2), \ldots, s_n(z_n, y)$,
   – $s_1(x, z_2), \ldots, s_n(z_n, z_{n+1}), r(z_{n+1}, y)$,
   – $s_1(x, z_2), \ldots, s_n(z_n, y)$,

   where $s_i \prec r$ for all $i = 1, \ldots, n$.
(5) Constant symbols only occur in rules that are facts.

Note that these conditions are easy to verify. With the modes predefined, one can check the requirements of Definition 4 in polynomial time. Condition (4) in Definition 7 is not hard to check: each path to which the condition applies can have at most one of the normal forms, and this can be checked effectively. To show the existence of the order $\prec$, it suffices to verify pairwise if the $\prec$ relationships that are required for the normal forms are free of cycles.

Definition 7 (4) captures a sufficient condition for preventing the encoding of context-free languages as in the proof of Theorem 3. Lemma 2 below states, essentially, that the relevant chains of binary relations form a regular language due to this. The complexity of the required constructions relates to the maximal length l of chains $s_1 \prec \ldots \prec s_l$ for the order $\prec$ used in Definition 7 (4), so we refer to l as the *depth* of $\prec$.

We can restrict attention to the following special case of Simple Directional Rules.

Definition 8. *A Simple Directional Rule is in* normal form *if it is either a fact, or in one of the following forms*

$$r(x, y), p(y) \to q(x) \qquad r(x, y) \to \exists z.t(x, z), q(z) \qquad r(x, y), s(y, z) \to t(x, z)$$
$$p_1(x), p_2(x) \to q(x) \qquad p(x) \to t(x, x) \qquad r(x, y) \to t(x, y)$$

where r, s, t are binary predicates, and p, q are unary predicates, $\top$, or $\bot$.

Proposition 1. *For every set of Simple Directional Rules Σ, there is a set of Simple Directional Rules in normal form Σ' that entails the same consequences over the signature of Σ. Moreover, Σ' can be computed in polynomial time.*

Proof. Essentially, two types of basic propositional transformations are used: (a) the transformation of an implication $p \to q \wedge r$ into two implications $p \to q$ and $p \to r$; (b) the transformation of an implication $p \wedge q \to r$ into two implications $p \wedge a \to r$ and $q \to a$ where a is a new symbol (or, in our case, formula). Transformations of type (b) introduce new auxiliary signature symbols, hence the normalisation does not preserve semantic equivalence. But since auxiliary symbols are fresh, it is clear that every model of the original formula can be extended to the enlarged signature to obtain a corresponding model of the transformed formulae. Thus a formula over the original signature is a logical consequences of the original rule set precisely if it is a consequence of the transformed rule set.

The normalisation proceeds in several stages. Initialise $\Sigma' := \Sigma$. First, consider every rule head as a conjunction $\bigwedge_{i=1}^{n} \varphi_i$ formulae φ_i of one the forms $t(x, y)$, $q(x)$, $\exists z.t(x, z), q(z)$, and $\exists z.q(z)$. This is possible by applying each existential quantifier in the head to only those atoms that contain the quantified variable. Due to Definition 7 (3) and Definition 4 (2) this must result in the required form (where q can be $\top$). Now each rule of the form $B \to \bigwedge_{i=1}^{n} \varphi_i$ with $n > 1$ is replaced by n rules $B \to \varphi_i$ for each $i = 1, \ldots, n$, thus introducing a linear number of new rules; a transformation of type (a). Moreover, all rule heads $\exists z.q(z)$ are replaced by $\exists z.s(x, z), q(z)$ for a fresh auxiliary predicate s and with x being the (unique) variable of the rule body that does not occur in output positions. After this step, every rule head in Σ' is in a form as in one of the normal forms.

Second, repeat the following "rolling-up" reduction as often as possible: Select a body atom $r(x, y)$ in a rule that does not have y in its rule head or in any other binary body atom, but for which x occurs in some other body atom. Let $\psi(y)$ denote the (possibly empty) conjunction of all unary body atoms $p(y)$. Now replace $r(x, y) \wedge \psi(y)$ with an atom $a(x)$ for some fresh unary predicate a, and add a new rule $r(x, y) \wedge \psi(y) \to a(x)$ (a type (b) transformation). After this step, each rule is either of the form $r(x, y) \wedge \psi(y) \to a(x)$ or has a body of the form $\psi_1(x_1), s_1(x_1, x_2), \psi_2(x_2), \ldots, \psi_n(x_n), s_n(x_n, x_{n+1})$ where x_1 and x_{n+1} occur in the head (a special case of this is $x_1 = x_{n+1}$).

Third, repeat the following reduction as often as possible: Select two body atoms $p(x), q(x)$ in some rule that has more than two atoms in its body, replace this occurrence with $a(x)$ for a fresh unary predicate a, and add a new rule $p(x), q(x) \to a(x)$. This is a transformation of type (b). At most linearly many such reductions are possible. After completing this step, all rules with body atoms of the form $p(x), q(x)$ are in normal form.

Fourth, for each unary predicate p in Σ', introduce a fresh binary predicate r_p, and add a rule $p(x) \to r_p(x, x)$. Then replace all occurrences of $p(x)$ in bodies of rules that are not in normal form yet by $r_p(x, x')$ for a fresh variable x', and replace x by x' in all input positions of other body atoms and in all output positions of head atoms. Again, only a linear number of replacements can happen. It is easy to see that this transformation preserves the semantics of the rule set, even though we use $r_p(x, x')$ instead of $r_p(x, x)$ in rule bodies. After this step, rules that are not in normal form contain only a single chain of binary body atoms.

Fifth, for each rule that contains more than two binary body atoms, select body atoms $s(x, y), r(y, z)$. For a fresh binary predicate t, replace $s(x, y), r(y, z)$ in the body

Table 1. Correspondence of Simple Directional Rules and Description Logic axioms

$p_1(x), p_2(x) \rightarrow q(x)$	$p_1 \sqcap p_2 \sqsubseteq q$	$r(x,y) \rightarrow \exists z.t(x,z), q(z)$	$\exists r.\top \sqsubseteq \exists t.q$
$r(x,y), p(y) \rightarrow q(x)$	$\exists r.p \sqsubseteq q$	$p(x) \rightarrow t(x,x)$	$p \sqsubseteq \exists t.\mathsf{Self}$
$r(x,y), s(y,z) \rightarrow t(x,z)$	$r \circ s \sqsubseteq t$	$r(x,y) \rightarrow t(x,y)$	$r \sqsubseteq t$

with $t(x,z)$ and add a rule $s(x,y), r(y,z) \rightarrow t(x,y)$. Again, only a linear number of these type (b) steps is possible. After this, all rules are in normal form.

It is easy to see that this transformation preserves Definition 7 (4), as all of the additional binary predicates can be integrated into the order $\prec$. In particular, the auxiliary predicates r_p for encoding unary predicates can be assumed to be $\prec$-minimal as they do not occur in the head of any rule which includes binary body atoms. The auxiliary binary predicates t introduced in the fifth step can be assumed to be $\prec$-larger than the predicates s, r that they replace, but $\prec$-smaller than all predicates that are $\prec$-larger than s and r.

Based on this normal form, it is easy to see that the semantics of many Simple Directional Rules can be expressed in the syntax of description logics as shown in Table 1. The DL axioms on the right have the same first-order semantics as the rules on the left. The required expressive features are already found in the tractable ontology language OWL EL, which can be viewed as a DL [26]. However, there are no results about the complexity of conjunctive query answering for this logic yet. The DL Horn-$\mathcal{SROIQ}$ covers all features of Table 1 and is known to admit query answering in 2ExpTime (ExpTime if the depth of $\prec$ is bounded) [30]. A logic that is closer to OWL EL is $\mathcal{EL}^{++}$, for which query answering is possible in exponential time (PSpace if the depth of $\prec$ is bounded) [28]. We will show below how this result can be extended to Simple Directional Rules.

6 A Query Entailment Algorithm

The only feature in Table 1 that is missing in $\mathcal{EL}^{++}$ are axioms of the form $p \sqsubseteq \exists t.\mathsf{Self}$. In this section, we show how the $\mathcal{EL}^{++}$ query entailment procedure from [28] can be extended to cover this case. We point out that our techniques do not solve the problem for arbitrary uses of Self in OWL EL. Our main result is as follows.

Theorem 5. *Checking conjunctive query entailment for Simple Directional Rules is decidable. The problem is NP-complete in the size of the query and P-complete in the size of the data. For rule sets for which the depth of the order $\prec$ in Definition 7 is bounded, the combined complexity of query entailment is* PSpace*-complete.*

The conjunctive query entailment procedure in [28] is highly non-deterministic since it aims at obtaining optimal complexity bounds rather than at providing a basis for implementations. The procedure translates inferencing tasks into word recognition problems for regular languages by associating words $w = s_1 \dots s_n$ over an alphabet $\mathbf{R} \cup \mathbf{I}$ of unary and binary relation symbols and constants to formulae φ_w. A

word w encodes a chain of elements connected by the given predicates, where constant symbols in w state that the current element in the chain is denoted by this constant. So words w encode paths in a model. For example, the word $prqcsp$ encodes the formula $p(x_0) \wedge r(x_0, c) \wedge q(c) \wedge s(c, x_2) \wedge p(x_2)$. To formalise this, we write w as $w = u_{01}u_{02} \dots u_{0m_0}\, r_1\, u_{11}u_{12} \dots u_{1m_1} \dots u_{(n-1)1} \dots u_{(n-1)m_{n-1}}\, r_n\, u_{n1} \dots u_{nm_n}$, where each $u_i = u_{i1} \dots u_{im_i}$ is a (possibly empty) word of unary predicate symbols and constants, and r_i is binary predicate symbol. For each word u_i, define a term $t(u_i) \coloneqq c$ if u_i contains a constant $c \in \mathbf{I}$, and $t(u_i) \coloneqq x_i \in \mathbf{V}$ otherwise (in particular if u_i is empty). We only consider words for which $t(u_i)$ is well-defined, i.e., where u_i contains at most one constant symbol.[3] Now define a formula $\varphi(u_i) \coloneqq \bigwedge\{u_{ij}(t(u_i)) \mid 1 \leq j \leq m_i \text{ and } u_{ij} \notin \mathbf{I}\}$, and set $\varphi_w \coloneqq \bigwedge_{i=0}^{n} \varphi(u_i) \wedge \bigwedge_{i=1}^{n} r_i(t(u_{i-1}), t(u_i))$.

We now give a standard result from description logics, and a less standard result that has been established for $\mathcal{EL}^{++}$:

Lemma 2 ([23]). *Consider a set Σ of Simple Directional Rules of form $r(x, y) \to t(x, y)$ and $r(x, y), s(y, z) \to t(x, z)$. For every binary predicate r, there is a non-deterministic finite automaton (NFA) $\mathcal{A}_r$ that accepts a word $w = s_1 \dots s_n$ iff $\Sigma \models \varphi_w \to r(x_0, x_n)$.*

Let $\prec$ be the order used in Definition 7. The size of $\mathcal{A}_r$ is exponential in the depth of $\prec$ and polynomial in the size of Σ for fixed depth of $\prec$. $\mathcal{A}_r$ can be constructed in time polynomial to its size.

Lemma 3 ([28] Theorem 2). *Consider a set of Simple Directional Rules Σ. For every unary predicate p, there is an NFA $\mathcal{A}_p$ such that, for every word w over predicates and constants, Σ entails $p(x_0) \to \exists\, (\mathsf{var}(\varphi_w) \setminus \{x_0\}).\varphi_w$ iff one of the following holds:*

- *$\mathcal{A}_p$ accepts the word w, or*
- *Σ is inconsistent, or Σ implies that p is empty.*

Similarly, there are NFA $\mathcal{A}_c$ for all constants c, such that the above conditions hold iff Σ entails the formula $\exists\, (\mathsf{Var}(\varphi_w) \setminus \{x_0\}).(\varphi_w\{x_0 \mapsto c\})$ where x_0 has been replaced by c.

$\mathcal{A}_p$ and $\mathcal{A}_c$ can be constructed in time polynomial in the size of Σ, and it has linearly many states in the size of Σ.

Proof. The construction for $\mathcal{EL}^{++}$ as given in [28] proceeds incrementally by adding transitions to the constructed automaton (see Table 1 cit.loc.). For example, translated to Datalog syntax, the construction rule (CR1) states that, whenever there is a transition $C \xrightarrow{p} C$ and a rule $p(x) \to q(x)$, a transition $C \xrightarrow{q} C$ is to be added (if not done already). It is easy to see that a similar construction rule can be given to cover axioms $p(x) \to t(x, x)$, where we require that $C \xrightarrow{p} C$ and $p(x) \to t(x, x) \in \Sigma$ imply $C \xrightarrow{t} C$. A binary relation t in an accepted word corresponds to a path segment $t(x, x')$, i.e., identity of x is not encoded unless the current state is a constant. It is easy to see that this does not affect the applicability of other Simple Directional Rules which cannot contain $t(x, x)$ due to the syntactic restrictions. □

[3] This is sufficient since co-occurrence of two constants would encode their equality which cannot be derived in our language.

The construction of $\mathcal{A}_p$ in [28] is such that the states of the NFA are exactly the unary relation symbols and constants, with the intuition that each state represents potential domain elements of which we know nothing else but that they belong to this relation, or that they are identified by this constant. This is used in the proofs of [28] that use a *universal model* to establish the correctness of the procedure. These arguments work exactly as in [28], so we omit repeating the details here. For completing our rule-based exposition of the algorithm, we record the following definition:

Definition 9. *Given an NFA $\mathcal{A}_p$ and a unary relation or constant symbol q, the NFA $\mathcal{A}(p, q)$ is obtained from $\mathcal{A}_p$ by removing all transitions that are labelled with unary predicates or constants, and by changing the set of final states to $\{q\}$ if $\Sigma \models \exists x.p(x)$, and to $\emptyset$ otherwise.*

The NFA $\mathcal{A}(p, q)$ encodes the paths of binary relations between the types of elements characterised by p and q. This completes the NFA constructions that are the basis of our query entailment algorithm.

The algorithm constructs a *proof graph* which establishes, for all models $\mathcal{I}$ of Σ, the existence of a suitable variable assignment that shows query entailment. Intuitively, the nodes of the proof graph are abstract representations of domain elements, and the proof graph encodes a fragment of an arbitrary model of Σ. Formally, let $\mathbf{C}$ be the set of all unary relation symbols and constant names. A proof graph is a tuple (N, L, E) consisting of a set of nodes N, a labelling function $L : N \to \mathbf{C} \cup \{\top\}$, and a *partial* transition function $E : N \times N \to \mathbf{A}$, where $\mathbf{A}$ is the set of all NFA over the alphabet $\mathbf{R} \cup \mathbf{I}$. A node $m \in N$ is *reachable* if there is some node $n \in N$ such that $E(n, m)$ is defined, and *unreachable* otherwise.

The algorithm for deciding conjunctive query entailment is given in Table 2. Any occurrence of the word "select" in the description indicates a non-deterministic choice of the algorithm. Step A factorises the query to allow multiple variables to refer to the same element or to a constant. Guessing this initially is not practical but convenient for an algorithm that aims at establishing complexity bounds. Step B initiates the proof graph and ensures that all nodes are reachable. Variable nodes eventually are reachable through exactly one predecessor node.

Steps C and D verify that the chosen proof graph implies the existence of all required elements (C) and the entailment of the query's unary atoms (D). Step D has been extended to also verify binary atoms of the form $t(x, x)$. This is not necessary for $\mathcal{EL}^{++}$: since it cannot express statements of the form $p(x) \to t(x, x)$, query atoms $t(x, x)$ can only match if x takes the value of a constant (checked in Step F). In contrast, Simple Directional Rules allow $t(x, x)$ to be derived for other elements. Our extension covers all additional cases: rules $p(x) \to t(x, x)$ can largely be eliminated from the rule set by replacing occurrence of $t(x, y)$ in rule bodies by $p(x)$ and renaming y to x. The correctness of our modified algorithm then follows from the correctness of the algorithm in [28], and the fact that the inferencing procedure of Section 4 can be used to check the entailments in lines 18 and 20.

Step E computes the automata $\mathcal{A}(R)$ of Lemma 2 and applies a non-deterministic *splitting* operation (see [28] for a formal definition). Splitting an NFA into k parts results in k copies of the NFA that only differ in their start and end states. The first part begins

Table 2. A non-deterministic algorithm for deciding conjunctive query entailment for a set *rb* of Simple Directional Rules in normal form

	A. Factorise query
1	Select a (possibly empty) set $X \subseteq \mathsf{Var}(q)$
2	For each $x \in X$
3	Select some $e \in \mathsf{Var}(q) \cup \mathbf{I}$ and
4	replace all occurrences of x in q with e
	B. Initialise proof graph (N, L, E)
5	$N \coloneqq \mathbf{I} \cup \mathsf{Var}(q)$, let E be undefined for all arguments
6	For each $a \in \mathbf{I}$, $L(a) \coloneqq a$
7	For each $x \in \mathsf{Var}(q)$, select $L(x) \in (\mathbf{C} \setminus \mathbf{I}) \cup \{\top\}$
8	For each $n \in N$, $a \in \mathbf{I}$, $E(n, a) \coloneqq \mathcal{A}(L(n), L(a))$
9	While there is an unreachable node
10	Select some unreachable $x \in \mathsf{Var}(q)$,
11	select some reachable $n \in N$
12	$E(n, x) \coloneqq \mathcal{A}(L(n), L(x))$
	C. Check proof graph
13	For each $n \in N$, $m \in \mathsf{Var}(q)$
14	If $E(n, m)$ is defined and accepts no word,
15	terminate with failure
	D. Check entailment of unary and reflexive atoms
16	For each query atom $\psi[n] \in q$ of form $p(n)$ or $t(n, n)$
17	If $L(n) \in \mathbf{R}$ (and thus $n \in \mathsf{Var}(q)$)
18	if $\Sigma \not\models L(n)(n) \rightarrow \psi$, terminate with failure
19	If $L(n) \in \mathbf{I}$ (and thus $n \in \mathbf{I}$)
20	if $\Sigma \not\models \psi$, terminate with failure
	E. Split inferencing automata
21	For each binary atom $r(n, m) \in q$
22	Compute shortest path $n = n_0, \ldots, n_k = m$ from n to m
23	Split $\mathcal{A}_r$ into k automata
24	$\mathcal{A}(r(n, m), n_0, n_1), \ldots, \mathcal{A}(r(n, m), n_{k-1}, n_k)$
25	For each $\mathcal{A}(r(n, m), n_{i-1}, n_i)$
26	If $\mathcal{A}(r(n, m), n_{i-1}, n_i)$ accepts no word,
27	terminate with failure
	F. Check entailment of binary atoms
28	$acc \coloneqq$ `true`
29	For each $n, m \in N$ with $E(n, m)$ defined
30	If $m \in \mathbf{I}$
31	For each split automaton $\mathcal{A}(F, n, m)$
32	If $\mathcal{A}(F, n, m)$ and $E(n, m)$ do not accept a common word
33	$acc \coloneqq$ `false`
34	Else if $m \in \mathsf{Var}(q)$
35	If no there is no word that is accepted by
36	$E(n, m)$ and by all split automata $\mathcal{A}(F, n, m)$
37	$acc \coloneqq$ `false`
38	If acc is `false`, then terminate with failure
39	Else accept the query

with the original initial state and terminates at some intermediate state that the next automaton starts at. The last part of the split again terminates in the original final state. The intuition underlying this operation is that each NFA $\mathcal{A}_r$ encodes possible chains of relations that suffice to derive r. One such chain must be found for every query atom $r(n, m)$. The structure of the proof graph defines how elements n and m can be connected, so any match with r must be distributed along the paths of the proof graph. This is implemented by the split.

Finally, Step F again verifies the earlier choices of the algorithm by comparing the (logically deducible) chains of relations that provided by the generating edge NFA with the chains that the split NFA require to exist for establishing a match. The case distinction reflects the different intention of edges leading to individual or variable nodes. For edges leading to a variable node, only a single generating role path exists in the canonical model, and *all* split automata must match one such path (line 35/36). For edges leading to constant nodes, all accepted paths exist in every model. Hence line 32 implements pairwise comparisons of each split NFA with the edge NFA.

There are two approaches for implementing the checks in lines 32 and 35/36, used for establishing different complexity bounds [28]. Computing the intersection automaton of two NFA can be done in quadratic time w.r.t. the size of the NFA. If the number of required intersections is bounded, this leads to a polynomial procedure, provided that the size of the NFA is polynomially bounded as in Lemma 2. The number of intersections is bounded if either the query or the size of Σ is bounded. If this is not the

case, the intersection NFA can be exponentially large and a more efficient approach is to run all (still linearly many) NFA in parallel, choosing transitions non-deterministically for each. If this procedure concurrently reaches a final state in all NFA, then they accept a common word. This can be done in polynomial space, and Savitch's Theorem (NPSPACE = PSPACE) yields the desired complexity. All remaining steps of the algorithm can be performed in NP for all cases considered in Theorem 5.

7 Conclusions

We have proposed novel fragments of TGDs for which simple and conjunctive query answering is decidable. While these approaches are not subsumed by any of the TGD fragments that have been discussed in the literature before, they cover important real-world models such as SNOMED CT.

In spite of these encouraging results, we believe that much more research is needed to properly understand the complex mechanisms that are encountered when combining value invention with complex rule languages. Description logics have long dealt with related issues but do not typically provide the complex dependency structures that rules can offer. Our results suggest that some ideas of DL show new ways for extending TGDs, whereas other DL-specific restrictions are actually unnecessary. Moreover, it is far from clear how to further extend our results on deciding general conjunctive queries – classical languages and automata are hardly applicable in the general n-ary case.

Thus further sources need to be tapped to advance this growing field. For example, the idea of assigning modes to predicates in order to establish a form of directedness has already been considered for Prolog more than a decade ago [4, 32]. This strand of work, which is also the source of our terminology, is not asking for decidability of logic programming but strives to improve the efficiency of unification. Yet, the underlying mechanisms of both approaches are clearly related.

Finally, the increasing amount of ideas on TGD formalisms needs to be further consolidated. Previous work has shown that it is often possible to combine and generalise different approaches, and it seems likely that Directional Rules can also be extended by incorporating some of the other ideas in the field. In addition to the work on such theoretical questions, increased development effort will be required to supply the software tools needed to help Datalog-based formalisms to become as widespread as some other modelling formalisms are today.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley (1994)
2. Abiteboul, S., Vianu, V.: Datalog extensions for database queries and updates. J. of Computer and System Sciences 43, 62–124 (1991)
3. Andréka, H., Németi, I., van Benthem, J.: Modal languages and bounded fragments of predicate logic. J. of Philosophical Logic 27(3), 217–274 (1998)
4. Apt, K., Etalle, S.: On the unification free Prolog programs. In: Mathematical Foundations of Computer Science (MFCS-93). LNCS, vol. 711, pp. 1–19. Springer (1993)

5. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, second edn. (2007)
6. Baget, J.F., Leclère, M., Mugnier, M.L., Salvat, E.: Extending decidable cases for rules with existential variables. In: Boutilier, C. (ed.) Proc. 21st Int. Joint Conf. on Artificial Intelligence (IJCAI'09). pp. 677–682. IJCAI (2009)
7. Baget, J.F., Leclère, M., Mugnier, M.L., Salvat, E.: On rules with existential variables: Walking the decidability line. Artificial Intelligence 175(9–10), 1620–1654 (2011)
8. Baget, J.F., Mugnier, M.L., Rudolph, S., Thomazo, M.: Walking the complexity lines for generalized guarded existential rules. In: Walsh [34], pp. 712–717
9. Beeri, C., Vardi, M.Y.: The implication problem for data dependencies. In: Even, S., Kariv, O. (eds.) Proc. 8th Colloquium on Automata, Languages and Programming (ICALP'81). LNCS, vol. 115, pp. 73–85. Springer (1981)
10. Cabibbo, L.: The expressive power of stratified logic programs with value invention. Information and Computation 147(1), 22–56 (1998)
11. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: Brewka, G., Lang, J. (eds.) Proc. 11th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'08). pp. 70–80. AAAI Press (2008)
12. Calì, A., Gottlob, G., Lukasiewicz, T.: A general datalog-based framework for tractable query answering over ontologies. In: Paredaens, J., Su, J. (eds.) Proc. 28th Symposium on Principles of Database Systems (PODS'09). pp. 77–86. ACM (2009)
13. Calì, A., Gottlob, G., Pieris, A.: Advanced processing for ontological queries. Proceedings of VLDB 2010 3(1), 554–565 (2010)
14. Calì, A., Gottlob, G., Pieris, A.: Query answering under non-guarded rules in Datalog+/-. In: Hitzler, P., Lukasiewicz, T. (eds.) Proc. 4th Int. Conf. on Web Reasoning and Rule Systems (RR 2010). LNCS, vol. 6333, pp. 1–17. Springer (2010)
15. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. of Automated Reasoning 39(3), 385–429 (2007)
16. Chandra, A.K., Lewis, H.R., Makowsky, J.A.: Embedded implicational dependencies and their inference problem. In: Proc. 13th Annual ACM Symposium on Theory of Computation (STOC'81). pp. 342–354. ACM (1981)
17. Chein, M., Mugnier, M.L.: Graph-Based Knowledge Representation: Computational Foundations of Conceptual Graphs. Springer (2008)
18. Courcelle, B.: The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. Information and Computation 85(1), 12–75 (1990)
19. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A.: Complexity and expressive power of logic programming. ACM Computing Surveys 33(3), 374–425 (2001)
20. Deutsch, A., Tannen, V.: Reformulation of XML queries and constraints. In: Calvanese, D., Lenzerini, M., Motwani, R. (eds.) Proc. 9th Int. Conf. on Database Theory (ICDT'03). LNCS, vol. 2572, pp. 225–241. Springer (2003)
21. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. Theoretical Computer Science 336(1), 89–124 (2005)
22. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)
23. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Proc. 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'06). pp. 57–67. AAAI Press (2006)
24. James, A.G., Spackman, K.A.: Representation of disorders of the newborn infant by SNOMED CT. In: Andersen, S.K., Klein, G.O., Schulz, S., Aarts, J. (eds.) Proc. 21st Int. Congress of the European Federation for Medical Informatics (MIE'08). pp. 833–838 (2008)

25. Johnson, D.S., Klug, A.: Testing containment of conjunctive queries under functional and inclusion dependencies. In: Proc. 1st Symposium on Principles of Database Systems (PODS'82). pp. 164–169. ACM (1982)
26. Krötzsch, M.: Efficient rule-based inferencing for OWL EL. In: Walsh [34], pp. 2668–2673
27. Krötzsch, M., Rudolph, S.: Extending decidable existential rules by joining acyclicity and guardedness. In: Walsh [34], pp. 963–968
28. Krötzsch, M., Rudolph, S., Hitzler, P.: Conjunctive queries for a tractable fragment of OWL 1.1. In: Aberer, K., Choi, K.S., Noy, N., Allemang, D., Lee, K.I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) Proc. 6th Int. Semantic Web Conf. (ISWC'07). LNCS, vol. 4825, pp. 310–323. Springer (2007)
29. Maier, D., Mendelzon, A.O., Sagiv, Y.: Testing implications of data dependencies. ACM Transactions on Database Systems 4, 455–469 (1979)
30. Ortiz, M., Rudolph, S., Simkus, M.: Worst-case optimal reasoning for the Horn-DL fragments of OWL 1 and 2. In: Lin, F., Sattler, U., Truszczynski, M. (eds.) Proc. 12th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'10). pp. 269–279. AAAI Press (2010)
31. Patel-Schneider, P.F., Horrocks, I.: A comparison of two modelling paradigms in the Semantic Web. J. of Web Semantics 5, 240–250 (2007)
32. Rao, M., Shyamasundar, R.: Unification-free execution of well-moded and well-typed Prolog programs. In: Mycroft, A. (ed.) Static Analysis. LNCS, vol. 983, pp. 243–260. Springer (1995)
33. Sowa, J.F.: Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley (1984)
34. Walsh, T. (ed.): Proc. 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11). AAAI Press/IJCAI (2011)