Second-Order Queries for Rule-Based Data Access

Technical Report 3019
Institute AIFB, KIT
November 2011

Markus Krötzsch
Department of Computer Science
University of Oxford, UK
markus.kroetzsch@cs.ox.ac.uk

Sebastian Rudolph
Institute AIFB
Karlsruhe Institute of Technology, DE
sebastian.rudolph@kit.edu

ABSTRACT

Rules and ontologies can be used to enrich a database system with advanced data access capabilities. The success of this paradigm has led to a number of languages such as DL-Lite, Datalog+/- and OWL RL. The two major approaches to answering queries under constraints expressed in such languages are forward-chaining (materialization) and backward-chaining (query rewriting). The latter is typically focused on first-order queries that have only limited expressivity. We propose a querying formalism based on monadic second-order logic which subsumes and goes beyond conjunctive queries and regular path queries, but still has a decidable query subsumption problem. We devise methods for rewriting rule sets to queries in this new formalism and we show that query entailment in most of the established rule-based approaches can be decided by combining two methods: (i) bottom-up forward-chaining computation w.r.t. a rule set with the bounded treewidth model property and (ii) top-down second-order query rewriting w.r.t. a rewritable rule set.

1. INTRODUCTION

Expressive querying capabilities are a crucial requirement in intelligent database systems. One of the major approaches to extend the classical framework of evaluating conjunctive queries against relational databases is to introduce an inference layer on the schema level. Then, query answering would take not only the relation instances of the database into account, but also those which can be inferred.

Two main paradigms for specifying the inference layer can be distinguished.

Rule-based approaches are rooted in deductive databases and logic programming. Recent approaches accommodate the capability of *value invention*, that is, ways to assert the existence of domain entities which are not in the active domain [13]. Thereby, they have also become very similar to the framework of tuple-generating-dependencies (TGDs), initially introduced for information exchange and information integration.

AIFB Technical Report 3019, Institut für Angewandte Informatik und Formale Beschreibungsverfahren (AIFB), Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
.

EXAMPLE 1. *Consider a database with the relation instances*

$$\textit{hasAuthor(a,c)} \quad \textit{and} \quad \textit{cites(a,b)}$$

as well as the inference rules

$$\begin{aligned} \textit{hasAuthor}(x,y) &\rightarrow \textit{publication}(x) \\ \textit{cites}(x,y) &\rightarrow \textit{publication}(x) \wedge \textit{publication}(y) \\ \textit{publication}(x) &\rightarrow \exists y.\textit{hasAuthor}(x,y) \end{aligned}$$

Then a rule-enhanced database would entail the conjunctive query $\exists z.(\textit{hasAuthor}(a,z))$ *but also the query* $\exists z.(\textit{hasAuthor}(b,z))$.

Mainstream ontological approaches rest on the logical framework of description logics (DLs, [4]). Ontologies, initially developed in the field of the Semantic Web, are currently gaining influence in the database areas of data integration and modeling. This trend is supported by the current concentration of DL research on so-called tractable fragments – light-weight logics which allow for quick inferencing and query answering on large data sets (cf. the approaches to *ontology-based data access* based on DL-Lite, [17]). Thereby it becomes apparent that most of the logics thus considered are structurally close to Horn-style rule formalisms (in particular they allow for universal models), and can partially be recast into a rule-based representation [13].

EXAMPLE 2. *The set of inference rules in Example 1 can equivalently be expressed by the DL-Lite ontology*

$$\begin{aligned} \exists\textit{hasAuthor} &\sqsubseteq \textit{publication} \\ \exists\textit{cites} &\sqsubseteq \textit{publication} \\ \exists\textit{cites}^- &\sqsubseteq \textit{publication} \\ \textit{publication} &\sqsubseteq \exists\textit{hasAuthor}. \end{aligned}$$

Since query answering in databases extended by an expressive inference layer can easily lead to undecidable problems [20, 11], much work has been spent on identifying restrictions on the structure of the inference layer that ensure decidability and even allow for efficient processing. Next to bottom-up materialization under inference rules (commonly referred to as the *chase* in databases), the central technique to answer conjunctive queries in an rule- or ontology-enhanced database is *query rewriting*, that is, to find "substitute queries" that can be evaluated against the databases alone, ignoring the inference layer, and yet deliver the same answer. In fact, for certain kinds of rule sets and ontologies, it is possible to establish the property of *first-order rewritability* [1]: in this case one can – given the inference rules and an arbitrary conjunctive query – compute a first-order formula which has this property irrespective of the underlying database.

EXAMPLE 3. *Given the rule set from Example 1 and the conjunctive query $\exists v, w.(hasAuthor(v, w))$, an appropriate first-order rewriting would be*

$$\exists v, w.(hasAuthor(v, w)) \lor \exists v, w.(cites(v, w)) \lor \exists v.(publication(v))$$

As evaluating first-order formulae against pure databases can be performed via standard SQL querying and hence make use of all optimization techniques developed for it, first-order rewritability is a very desirable property. Unfortunately it turns out that many practically useful modeling features destroy first-order rewritability.

EXAMPLE 4. *As a straightforward consequence of the fact that first-order logic cannot express the transitive closure, there cannot be a first-order rewriting for the conjunctive query $\exists v, w.(s(v) \land r(v, w) \land s(w))$ given the rule $r(x, y) \land r(y, z) \to r(x, z)$.*

This directly leads us to the central question of this paper:

> *How can the idea of query rewriting be extended to a more expressive formalism that allows rewriting a significantly larger amount of practically relevant TGDs, and still provides essential computational properties?*

One solution which immediately comes to mind is to add recursion to the querying language and one prominent way to do so is via Datalog queries. Of course, the Datalog query containing the two rules $r(x, y) \land r(y, z) \to r(x, z)$ and $s(v) \land r(v, w) \land s(w) \to match$ can serve as representation of the rewriting (note that here, the rules are considered as part of the query and not of the inference layer). However, Datalog turns out to be too expressive to allow for even the most basic tasks of query management. Most notably, answering Datalog queries on DL-Lite ontologies is already undecidable in general. Moreover, checking subsumption of Datalog queries is undecidable as well [33].

Consequently, it seems that the rewriting target formalism should allow for recursion, yet only in a restricted, regular way. In fact, the above problem can be satisfactorily solved by using *conjunctive regular path queries* [28, 19] which provide tamed recursiveness by allowing for regular expressions on binary predicates.

EXAMPLE 5. *Without going into syntactic and semantic formalities yet, we note that the conjunctive regular path query*

$$\exists v, w.(s(v) \land r^*(v, w) \land s(w)),$$

where $r^*(v, w)$ matches any individual pair connected by a path of r-relations, can serve as a suitable rewriting for Example 4.

To the best of our knowledge, rewriting of conjunctive queries into conjunctive regular path queries has been addressed only very implicitly by now [32]. Moreover, regular path queries are still rather constrained: besides further structural restrictions, they only allow for recursion over binary predicates.

Hence it seems that a suitable notion of expressive yet computationally manageable queries by means of which query rewriting can be applied to a wider range of cases is yet to be identified. The contribution of this paper can be summarized as follows:

- We introduce *positive monadic second-order queries* (POMSOQs) as a suitable target formalism for query rewriting that subsumes unions of conjunctive queries as well as conjunctive 2-way regular path queries. We discuss the expressivity of this new notion of queries which we deem interesting and practically relevant in their own right.
- We show that POMSOQs are equivalent to a certain well-behaved fragment of Datalog queries and establish complexity bounds for POMSOQ answering.
- Exploiting correspondences to monadic second-order logic, we prove that the subsumption problem for POMSOQs is decidable.
- We introduce the notion of POMSOQ-rewritability for which we identify sufficient conditions and show how the rewriting can be obtained if these are met. Additionally, we provide a technique to transform suitable sets of inference rules into a logically equivalent form that satisfies this condition.
- We show that POMSOQ entailment is satisfiable in the presence of rules that satisfy the bounded tree-width model property (which is the case for a plethora of popular TGD fragments like Datalog, acyclic TGDs, guarded TGDs and generalizations thereof). Additionally, we show that conjunctive query answering is decidable for any rule set that can be decomposed into one that is POMSOQ-rewritable and one with the bounded-treewidth-model property.

Longer proofs are omitted from the main paper, especially if they do not contribute interesting conceptual points. They can be found in the Appendix.

2. DATABASES, RULES AND QUERIES

In this section, we introduce our notation for databases, conjunctive queries, Datalog and tuple-generating dependencies (TGDs).

We consider a standard language of predicate logic, based on a finite set of *constant symbols* $\mathbf{C}$, a finite set of *predicate symbols* $\mathbf{P}$, and an infinite set of *(object) variables* $\mathbf{V}$. Each predicate $p \in \mathbf{P}$ is associated with a natural number $\mathsf{ar}(p)$ called the *arity* of p. We often assume that some such signature has been fixed and do not refer to it explicitly.

A *term* is a variable $x \in \mathbf{V}$ or a constant $c \in \mathbf{C}$. We use symbols s, t to denote terms, x, y, z to denote variables, a, b, c to denote constants. Expressions like $\mathbf{t}$, $\mathbf{x}$, $\mathbf{c}$ denote finite lists of such entities. An *atom* is a formula of the form $r(t_1, \ldots, t_n)$ where $t_1, \ldots, t_n$ are terms and $r \in \mathbf{P}$ is a predicate symbol with $\mathsf{ar}(r) = n$. We write $\varphi[\mathbf{x}]$ to emphasize that a formula φ has free variables $\mathbf{x}$; we write $\varphi[\mathbf{c}/\mathbf{x}]$ for the formula obtained from φ by replacing each variable in $\mathbf{x}$ by the respective constant in $\mathbf{c}$ (both lists must have the same length).

A *conjunctive query* (CQ) is a formula $Q[\mathbf{x}] = \exists \mathbf{y}.\psi[\mathbf{x}, \mathbf{y}]$ where $\psi[\mathbf{x}, \mathbf{y}]$ is a conjunction of atoms. A *tuple generating dependency* (TGD) is a formula of the form $\forall \mathbf{x}, \mathbf{y}.\varphi[\mathbf{x}, \mathbf{y}] \to \exists \mathbf{z}.\psi[\mathbf{x}, \mathbf{z}]$ where φ and ψ are conjunctions of atoms, called the *body* and *head* of the TGD, respectively. TGDs never have free variables, so we usually omit the universal quantifier when writing them. A *Datalog rule* is a TGD without existentially quantified variables. By convention, we consider empty bodies to be true and empty heads to be false, i.e., a TGD with empty body is a *fact* (something that is unconditionally true), and a TGD with empty head is a *constraint* (something that must never be true). A formula is *ground* if it contains no variables, and it is a *sentence* if it contains no free variables. A *database* is a finite set of ground facts.

We consider formulae under the standard semantics of first-order logic. An *interpretation* $\mathcal{I}$ consists of a (possibly infinite) *domain* $\Delta^{\mathcal{I}}$ and a function $\cdot^{\mathcal{I}}$ that maps constants c to domain elements $c^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ and predicate symbols p to relations $p^{\mathcal{I}} \in (\Delta^{\mathcal{I}})^{\mathsf{ar}(p)}$. A *variable assignment* for $\mathcal{I}$ is a function $\mathcal{Z} : \mathbf{V} \to \mathbf{C}$. Conditions for $\mathcal{I}$ and $\mathcal{Z}$ to satisfy a first-order formula φ (i.e., to be a *model* of φ, written

$\mathcal{I}, \mathcal{Z} \models \varphi$) are defined as usual. If φ has no free variables, then $\mathcal{Z}$ is irrelevant for satisfaction and can be omitted. Given interpretations $\mathcal{I}$, $\mathcal{J}$, a *homomorphism* π from $\mathcal{I}$ to $\mathcal{J}$ is a function $\pi : \Delta^{\mathcal{I}} \to \Delta^{\mathcal{J}}$ such that: (i) for all constants c, we have $\pi(c^{\mathcal{I}}) = c^{\mathcal{J}}$, and (ii) for all predicate symbols p and list of domain elements $\boldsymbol{\delta}$, we have $\boldsymbol{\delta} \in p^{\mathcal{I}}$ implies $\pi(\boldsymbol{\delta}) \in p^{\mathcal{J}}$.

A list of constants $\mathbf{c}$ is an *answer* to a CQ $Q[\mathbf{x}]$ over a database D and set Σ of TGDs if $D, \Sigma \models Q[\mathbf{c}/\mathbf{x}]$.[1] Query answering is facilitated in practice since one may focus on universal models. A *universal model* of a set of sentences S is an interpretation $\mathcal{I}$ such that (i) $\mathcal{I} \models S$, (ii) $c^{\mathcal{I}} = c$, and (iii) for every interpretation $\mathcal{J}$ with $\mathcal{J} \models S$, there is a homomorphism from $\mathcal{I}$ to $\mathcal{J}$. The two main facts for query answering are:

- For every database D and set Σ of tgds, if $D \cup \Sigma$ is satisfiable then $D \cup \Sigma$ has some universal model.
- Given a CQ $Q[\mathbf{x}]$ and constants $\mathbf{c}$, the models of $Q[\mathbf{c}/\mathbf{x}]$ are closed under homomorphic images.

Therefore, we have that $D \cup \Sigma \models Q[\mathbf{c}/\mathbf{x}]$ if and only if $\mathcal{I} \models Q[\mathbf{c}/\mathbf{x}]$ for some universal model $\mathcal{I}$. Universal models can be defined by a (possibly infinite) construction process called the *chase*. We let $\mathcal{I}(D \cup \Sigma)$ denote an arbitrarily chosen but fixed universal model of $D \cup \Sigma$.

3. POSITIVE MSO QUERIES

In this section, we introduce an expressive query language based on monadic second-order logic (MSO). With this we mean the extension of first-order logic by *set variables* that are used like predicates of arity 1. To distinguish them from object variables x, y, z, we denote set variables with uppercase letters $\mathtt{U}, \mathtt{V}$, possibly with subscripts. We adhere to the standard semantics of MSO that we will not repeat here.

As discussed above, answering CQs is facilitated by the possibility to restrict attention to universal models, which is due to the fact that the models of CQs are closed under homomorphisms. According to the Łos-Tarski-Lyndon Theorem, every first-order formula that is preserved under homomorphisms is equivalent to a *positive* existential formula. The latter can be easily further normalized into a shape widely known as union of conjunctive queries. We are not aware of a version of the Łos-Tarski-Lyndon Theorem for MSO which could provide a similar characterization, but in what follows, we will identify a fragment of MSO that is preserved under homomorphisms and sufficiently expressive for our subsequent considerations.

Definition 1. *The set of* positive monadic second-order queries *(*POMSOQs*) is defined inductively. A POMSOQ of degree* 0 *is an atomic formula. A POMSOQ of degree* $d + 1$ *is an MSO formula*

$$\forall \mathtt{U}_1, \ldots, \mathtt{U}_m \neg \forall \mathbf{y}. \bigwedge_{R \in \mathcal{R}} R$$

where $\mathtt{U}_1, \ldots, \mathtt{U}_m$ *are monadic second-order variables,* $\mathbf{y}$ *is a list of first-order variables, and every* $R \in \mathcal{R}$ *is an implication of the form* $B_1 \wedge \ldots \wedge B_\ell \to H$ *such that*

- B_i *are POMSOQs of degree at most* d *or of the form* $\mathtt{U}_j(x)$*, and*
- H *is of the form* $\mathtt{U}_j(x)$ *or empty.*

We write $\mathfrak{Q}[\mathbf{x}]$ *to emphasize that a POMSOQ* $\mathfrak{Q}$ *has free variables* $\mathbf{x}$*. The* arity *of* $\mathfrak{Q}[\mathbf{x}]$ *is the number of variables in* $\mathbf{x}$*. The* subqueries *of* $\mathfrak{Q}$ *are the POMSOQs that are subformulae of* $\mathfrak{Q}$*.*

At a first glance, the shape of POMSOQs may appear to be rather specific. But our choice of this formulation is well motivated by the aim to express a particular "computation scheme" in monadic second-order logic. The intuition for degree 1 POMSOQs is that, given a database and a tuple $\boldsymbol{\delta} = \langle \delta_1, \ldots, \delta_n \rangle$ of domain elements, we determine whether $\boldsymbol{\delta}$ is an answer by an iterative deterministic coloring procedure. "Coloring rules" specify how colors are assigned to domain elements – depending on (i) the chosen $\boldsymbol{\delta}$, (ii) the instances of predicates from $\mathbf{P}$ and (iii) colors that were already assigned. The monadic predicate variables $\mathtt{U}_1, \ldots, \mathtt{U}_m$ in the above formula encode the colors, and rules of the shape $B_1 \wedge \ldots \wedge B_\ell \to \mathtt{U}_i(x)$ can be directly read as declarative descriptions on how to initialize and propagate colors. The actual "success criterion" for $\boldsymbol{\delta}$ being an answer set is, whether one of possibly several certain configurations can be found in the color-saturated database. These configurations can be expressed by conjunctive queries accessing both database relation instances and colors. In order to obtain a uniform representation as rule set without introducing auxiliary predicates of arity > 1 (which would force us to go beyond MSO logic), we encode these "success criteria" as integrity constraints, that is, rules with an empty head having the desired configuration as body. This way, we obtain a coloring scheme which "results in" an inconsistency exactly if the test tuple $\boldsymbol{\delta}$ is an answer. This is the reason why we have to invert the whole criterion by putting a negation in front of the whole rule set. The universal quantification over the monadic predicates is necessary to minimize their extension and to ensure that, roughly speaking, just those database elements are colored which have to, i.e., no spurious colors are introduced. Finally, we obtain an MSO formula that is true for all bindings of its free variables to domain element tuples for which the described coloring technique succeeds. Furthermore, the specific form of the formula (containing free variables) allows us to conceive it as a definition of a new predicate and to use it inside another POMSOQ. This leads to the general nested structure of POMSOQs described in our above definition.

Example 6. *To start with an easy example along the lines of the introduction, assume we are interested in certification chains, more precisely, we want to query for all pairs* x, y *where a chain of certifiedBy relations from* x *to* y *exists. This information need can be expressed by the POMSOQ* $\mathfrak{Q}_1[x, y] = \forall \mathtt{U}_1 \neg \forall v, v' \bigwedge_{R \in \mathcal{R}} R$ *with* $\mathcal{R}$ *consisting of the rules*

$$\begin{aligned} \mathit{certifiedBy}(x, v) &\to \mathtt{U}_1(v) \\ \mathtt{U}_1(v) \wedge \mathit{certifiedBy}(v, v') &\to \mathtt{U}_1(v') \\ \mathtt{U}_1(y) &\to \end{aligned}$$

Figure 1 (left) displays the class of structures recognized by this and the following POMSOQs graphically. As a next example, assume we have a routing problem where a message has to be securely passed through a network from Alice to Bob. Assume we have entities certifying the security of the message handling in certain nodes. We are interested in which entities x *are able to (directly) certify secure treatment of the message on all intermediate nodes on some path from Alice to Bob, as illustrated in Fig. 1 (top). This can be expressed by the degree 1 POMSOQ* $\mathfrak{Q}_2[y] = \forall \mathtt{U}_2, \mathtt{U}_3 \neg \forall x, x' \bigwedge_{R \in \mathcal{R}'} R$*, where* $\mathcal{R}'$ *consists of the following rules:*

$$\begin{aligned} \mathit{certifiedBy}(x, y) &\to \mathtt{U}_3(x) \qquad (\ddagger) \\ \mathit{linkedTo}(\mathit{alice}, x) &\to \mathtt{U}_2(x) \\ \mathtt{U}_2(x) \wedge \mathtt{U}_3(x) \wedge \mathit{linkedTo}(x, x') &\to \mathtt{U}_2(x') \\ \mathtt{U}_2(\mathit{bob}) &\to \end{aligned}$$

[1] This semantics does not make the unique name assumption which can be axiomatised in well-known ways if equality is used. If equality is not used, the unique name assumption has no effect on query answers.

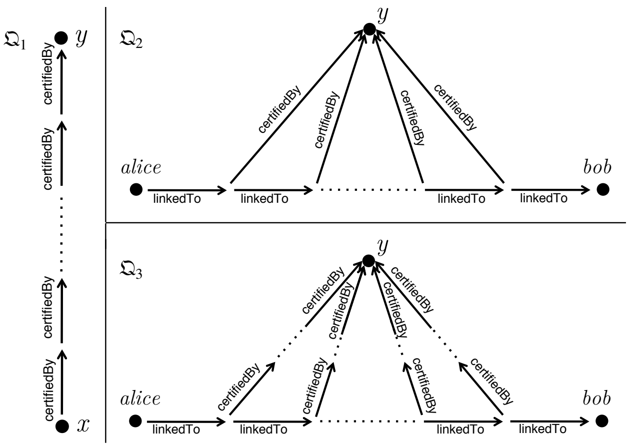

Figure 1: Sketches of the structures recognized by the POMSOQs in Example 6.

Finally, assume that we are not only interested in the case where the nodes are certified directly by an entity, but that there may be a certification chain from the node to the searched guarantee-providing entity, see Fig. 1 (bottom). This request can be expressed by the degree 2 POMSOQ $\mathfrak{Q}_3$ that coincides with $\mathfrak{Q}_2$ except that Rule (‡) is substituted by $\mathfrak{Q}_1[x, y] \to \mathtt{U}_3(x)$.

This new query notion is rather powerful. It is easy to see that it subsumes conjunctive queries (CQs) as well as unions of CQs. Indeed, given k CQs $\exists \mathbf{y_i}.B_i[\mathbf{x_i}, \mathbf{y_i}]$ with $i \in \{1, \ldots, k\}$, their union is expressed as the degree 1 POMSOQ $\neg\forall \mathbf{y_1}, \ldots, \mathbf{y_k}. \bigwedge_{i=1}^{k} .B_i[\mathbf{x_i}, \mathbf{y_i}] \to .$

Before providing further examples, we note that POMSOQs share a property with CQs which is typical for positive queries which are supposed to detect "structural configurations": their model classes are closed under structure-preserving mappings.

Theorem 1. *For a POMSOQ $\mathfrak{Q}[\mathbf{x}]$, the set of models of $\exists\mathbf{x}.\mathfrak{Q}$ is closed under homomorphisms.*

Note that the property established above also holds for formulas of the shape $\mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ obtained by replacing the free variables $\mathbf{x}$ of a POMSOQ by an answer $\mathbf{c}$, since they themselves can be interpreted as POMSOQs with no free variables. By virtue of the above theorem we are able to reduce entailment checking to model checking, if the premise contains only ground facts.

Corollary 2. *Let $\mathfrak{Q}[\mathbf{x}]$ be a POMSOQ, let D be a database, and let $\mathbf{c}$ be a list of constants. Then $D \models \mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ iff $\mathcal{I}(D) \models \mathfrak{Q}[\mathbf{c}/\mathbf{x}]$.*

Proof. The "only if" direction is a straightforward consequence from the fact that $\mathcal{I}(D) \models D$. For the "if" direction, suppose toward a contradiction that $\mathcal{I}(D) \models \mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ but $D \not\models \mathfrak{Q}[\mathbf{c}/\mathbf{x}]$, i.e., there is an interpretation $\mathcal{J}$ with $\mathcal{J} \models D$ but $\mathcal{I} \not\models \mathfrak{Q}[\mathbf{c}/\mathbf{x}]$. Exploiting the universality of $\mathcal{I}(D)$ we find a homomorphism from $\mathcal{I}(D)$ to $\mathcal{J}$. Since models of $\mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ are closed under homomorphisms, we conclude that $\mathcal{I} \models \mathfrak{Q}$, a contradiction. □

4. REGULAR PATH QUERIES

We now show that POMSOQs subsume not only standard conjunctive queries, but also the more powerful notion of *conjunctive 2-way regular path queries* [28, 19]. This observation further deepens our understanding of the expressive power of POMSOQs.

Intuitively, conjunctive 2-way regular path queries allow arbitrary regular expressions over binary predicates and inverted binary predicates to be used in place of binary atoms.

Definition 2. *A conjunctive 2-way regular path query (C2RPQ) over a signature $\langle \mathbf{C}, \mathbf{P}, \mathbf{V}\rangle$ is a first-order conjunctive query over the signature $\langle \mathbf{C}, \mathbf{P} \cup \mathbf{P}^{\mathrm{reg}}, \mathbf{V}\rangle$ with $\mathbf{P}^{\mathrm{reg}}$ containing all regular expressions over the alphabet $\Gamma = \{p \mid \mathsf{ar}(p) = 2\} \cup \{p^- \mid \mathsf{ar}(p) = 2\}$ and setting $\mathsf{ar}(ex) := 2$ for all $ex \in \mathbf{P}^{\mathrm{reg}}$. Given an interpretation $\mathcal{I}$ and a C2RPQ Q over $\langle \mathbf{C}, \mathbf{P}, \mathbf{V}\rangle$, we let $\mathcal{I} \models Q$ if $\mathcal{I}^r \models Q$ where $\mathcal{I}^r$ is an interpretation over $\langle \mathbf{C}, \mathbf{P} \cup \mathbf{P}^{\mathrm{reg}}, \mathbf{V}\rangle$ that coincides with $\mathcal{I}$ for all elements from $\langle \mathbf{C}, \mathbf{P}, \mathbf{V}\rangle$ and lets $\langle \delta, \delta'\rangle \in ex^{\mathcal{I}^{\mathrm{reg}}}$ exactly if there is a word $\gamma_1 \ldots \gamma_n$ matching the regular expression ex and a sequence $\delta = \delta_0 \ldots \delta_n = \delta'$ of domain elements such that for every $i \in \{0, \ldots, n-1\}$ one of the two is the case*

- *$\gamma_i = r \in \mathbf{P}$ and $\langle \delta_i, \delta_i + 1\rangle \in r^{\mathcal{I}}$ or*
- *$\gamma_i = r^-$ with $r \in \mathbf{P}$ and $\langle \delta_i + 1, \delta_i\rangle \in r^{\mathcal{I}}$.*

The following definition provides a way to translate C2RPQs into POMSOQs.

Definition 3. *Given a C2RPQ $Q = \exists\mathbf{x}.pr_1(\mathbf{x}_1) \wedge \ldots \wedge pr_k(\mathbf{x}_k)$, we define the POMSOQ $\mathfrak{Q}_Q$ as follows. Given an atom $ex(x, y)$ of Q with $ex \in \mathbf{P}^r$, let $\mathcal{A}_{ex} = (\Gamma, S, I, F, T)$ be the finite automaton corresponding to ex. Then, we let $\mathfrak{Q}_{ex(x,y)}$ denote the POMSOQ*

$$\forall(\mathtt{U}_s)_{s\in S}\, \neg\forall z, z'. \bigwedge_{R\in\mathcal{R}} R$$

with $\mathcal{R}$ containing the rules

- *$\to \mathtt{U}_s(x)$ for every initial state $s \in I$,*
- *$\mathtt{U}_s(y) \to$ for every final state $s \in F$,*
- *$(U)_s(z) \wedge r(z, z') \to \mathtt{U}_{s'}(z')$ for every transition $\langle s, r, s'\rangle \in T$, and*
- *$(U)_s(z) \wedge r(z', z) \to \mathtt{U}_{s'}(z')$ for every transition $\langle s, r^-, s'\rangle \in T$.*

Finally, we define $\mathfrak{Q}_Q$ as the formula

$$\neg\forall\mathbf{x}.(\tau(pr_1(\mathbf{x}_1)) \wedge \ldots \wedge \tau(pr_k(\mathbf{x}_k))) \to$$

$$\text{with} \qquad \tau(pr_i(\mathbf{x}_i)) := \begin{cases} pr_i(\mathbf{x}_i) & \text{if } pr_i \in \mathbf{P} \\ \mathfrak{Q}_{ex(\mathbf{x})} & \text{if } pr_i \in \mathbf{P}^{\mathrm{reg}}. \end{cases}$$

The intuition behind the translation of C2RPQs to POMSOQs is to find the possible bindings to x and y in $ex(x, y)$ by simulating all possible runs of the automaton corresponding to a regular expression predicate and see whether a run starting in x in the initial state and reaching y in the final state can be found. Colors are associated to states of the automaton and used to keep track of the information which domain elements can be reached in which state if one starts at x in an initial state. Consequently, the success criterion is satisfied if y is colored by the final state. This way, we can establish the following proposition.

Proposition 3. *For any C2RPQ Q, the answer sets for Q and $\mathfrak{Q}_Q$ coincide.*

Example 7. *Consider the regular path query*

$$mountain(x) \wedge continent(y) \wedge (locatedIn|hasPart^-)^*(x, y).$$

The corresponding POMSOQ looks as follows

$$\begin{array}{l} \neg(mountain(x) \wedge continent(y) \wedge \\ \quad \forall\mathtt{U}.\neg\forall z, z'.(\to \mathtt{U}(x), \mathtt{U}(y) \to, \\ \qquad\qquad \mathtt{U}(z) \wedge locatedIn(z, z') \to \mathtt{U}(z') \\ \qquad\qquad \mathtt{U}(z) \wedge hasPart(z', z) \to \mathtt{U}(z')) \\ \quad\quad) \to \end{array}$$

which in this case can be simplified to

$$\begin{aligned}&\textit{mountain}(x)\wedge\textit{continent}(y)\wedge\\&\quad\forall \mathtt{U}.(\mathtt{U}(x)\\&\qquad\wedge\forall z,z'.(\mathtt{U}(z)\wedge \textit{locatedIn}(z,z')\rightarrow \mathtt{U}(z'))\wedge\\&\qquad\wedge\forall z,z'.(\mathtt{U}(z)\wedge \textit{hasPart}(z',z)\rightarrow \mathtt{U}(z'))\\&\qquad\rightarrow \mathtt{U}(y)\end{aligned}$$

Arguably, the latter formula illustrates the underlying coloring idea in the most graspable way. Here, we need just one color which is initialized at x and then propagated over locatedIn and inversely over hasPart relationships with the success criterion being that y is finally colored.

However, the expressivity of POMSOQs goes well beyond that of C2RPQs even if we restrict to at most binary predicates and to POMSOQs of degree 1. Informally, this follows from the easy observation that for every C2RPQ Q there is an integer n, such that whenever Q matches into a graph G, it also matches into a graph G' where all vertices have degree $\leq n$ and from which there is a homomorphism into G. On the other hand, is easy to see that the POMSOQ $\mathfrak{Q}_2[y]$ from Example 6 does not have this property.

5. FROM POMSOQS TO DATALOG

In the following, we will show that POMSOQs of arbitrary degree can be expressed as Datalog queries. Thereby the monadic predicate variables have to be "contextualized" which increases their arity. Hence in general, the Datalog queries obtained by translating POMSOQs will not be monadic Datalog.

DEFINITION 4. *Given a POMSOQ $\mathfrak{Q}[\mathbf{x}]$, a set $\Sigma(\mathfrak{Q})$ of Datalog rules over an extended signature is defined inductively as follows. Let $p_{\mathfrak{Q}}$ be a fresh predicate symbol of arity n where n is the number of query variables $\mathbf{x}$. If $\mathfrak{Q}$ is of degree 0, then $\Sigma(\mathfrak{Q}) := \{\forall \mathbf{x}.\mathfrak{Q}[\mathbf{x}] \rightarrow p_{\mathfrak{Q}}(\mathbf{x})\}$. If $\mathfrak{Q}$ is of the form $\forall \mathtt{U}_1,\ldots,\mathtt{U}_m \neg\forall \mathbf{y}. \bigwedge_{R\in\mathcal{R}} R$ with degree $d>0$, then $\Sigma(\mathfrak{Q})$ consists of the following rules:*

- *for every $R\in\mathcal{R}$, a rule $\hat{R}$ is obtained by replacing each occurrence of a second-order atom $\mathtt{U}_i(z)$ with the atom $\hat{U}_i(z,\mathbf{x})$ where $\hat{U}_i$ is a fresh predicate of arity $n+1$;*
- *for every POMSOQ $\mathfrak{Q}'[\mathbf{x}']$ of degree smaller than d that occurs in $\mathfrak{Q}$, the rules $\Sigma(\mathfrak{Q}')$ are added to $\Sigma(\mathfrak{Q})$ and all occurrences of $\mathfrak{Q}'[\mathbf{x}']$ are replaced by $p_{\mathfrak{Q}'}(\mathbf{x}')$.*

Note that the rules obtained by this transformation might be unsafe, i.e., may contain universally quantified variables in the head that do not occur in the body. This is no problem with the logical semantics that we consider.

EXAMPLE 8. *The Datalog translation for the POMSOQ $\mathfrak{Q}_3$ from Example 6 looks as follows:*

$$\begin{array}{lrcl}\Sigma(\mathfrak{Q}_1) & \textit{certifiedBy}(z_1,v) & \rightarrow & \hat{U}_1(v,z_1,z_2)\\ & \hat{U}_1(v,z_1,z_2)\wedge \textit{certifiedBy}(v,v') & \rightarrow & \hat{U}_1(v',z_1,z_2)\\ & \hat{U}_1(z_2,z_1,z_2) & \rightarrow & p_{\mathfrak{Q}_1}(z_1,z_2)\\ \Sigma(\mathfrak{Q}_3)\setminus\Sigma(\mathfrak{Q}_1) & p_{\mathfrak{Q}_1}(x,z_3) & \rightarrow & \hat{U}_3(x,z_3)\\ & \textit{linkedTo}(alice,x) & \rightarrow & \hat{U}_2(x,z_3)\\ & \hat{U}_2(x,z_3)\wedge\hat{U}_3(x,z_3)\wedge \textit{linkedTo}(x,x') & \rightarrow & \hat{U}_2(x',z_3)\\ & \hat{U}_2(bob,z_3) & \rightarrow & p_{\mathfrak{Q}_3}(z_3)\end{array}$$

THEOREM 4. *For every POMSOQ $\mathfrak{Q}$, the set $\Sigma(\mathfrak{Q})$ can be constructed in linear time and both expressions are equivalent in the sense that $\models \forall\mathbf{x}.\mathfrak{Q}[x] \leftrightarrow (\Sigma(\mathfrak{Q})\rightarrow p_{\mathfrak{Q}}(\mathbf{x}))$ is a tautology.*

Using backwards-chaining, the goal $p_{\mathfrak{Q}}(\mathbf{x})$ can be expanded under the rules $\Sigma(\mathfrak{Q})$ to obtain a (possibly infinite) set of CQs that do not contain auxiliary predicates $p_{\mathfrak{Q}'}$. Thus $\mathfrak{Q}$ can be considered as a union of (possibly infinitely many) conjunctive queries, which will be useful in Section 6 below. First, however, we note the following complexity result.

THEOREM 5. *Given a database D, a POMSOQ $\mathfrak{Q}[\mathbf{x}]$, and a list of constants $\mathbf{c}$, checking $D\models\mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ is in PSpace w.r.t. the combined size of D and $\mathfrak{Q}[\mathbf{c}/\mathbf{x}]$. Moreover, it is PTime-complete w.r.t. the size of D.*

Mark that the data complexity thus established contrasts with that for CQs (AC_0) and for C2RPQs (NLogSpace-complete, hardness via graph reachability, membership via a translation into linear Datalog [16]) and gives no improvement over full Datalog queries. However, in terms of combined complexity, Datalog queries are ExpTime-complete [24] and hence harder than POMSOQs. Moreover, in the next sections, we will show that POMSOQs are more well-behaved when it comes to subsumption checking or interaction with rule sets that give rise to infinite structures.

6. CHECKING QUERY SUBSUMPTION

Checking query subsumption is an essential task in database management, facilitating query optimization, information integration and exchange and database integrity checking. The *subsumption* or *containment problem* of two queries $\mathfrak{P}$ and $\mathfrak{Q}$ is the question whether the answers of $\mathfrak{Q}$ are contained in the answers of $\mathfrak{P}$ for any underlying database or rule set. Formally, this is the case if the formula $\forall\mathbf{x}.\mathfrak{Q}[\mathbf{x}]\rightarrow\mathfrak{P}[\mathbf{x}]$ is valid. In this section, we show that this problem is decidable for POMSOQs.

An important tool for obtaining this result is the following notion of *treewidth* of an interpretation.

DEFINITION 5. *Given an interpretation $\mathcal{I}$, a* tree decomposition *of $\mathcal{I}$ is an undirected tree where each node n is associated with a set $\lambda(n)\subseteq\Delta^{\mathcal{I}}$ of domain elements such that:*

- *for every tuple $\langle\delta_1,\ldots,\delta_m\rangle\in p^{\mathcal{I}}$ for some predicate p, there is a node n with $\delta_1,\ldots,\delta_m\in\lambda(n)$;*
- *for every $\delta\in\Delta^{\mathcal{I}}$, the set of nodes $\{n\mid\delta\in\lambda(n)\}$ is connected.*

The width *of a tree decomposition is the maximal cardinality of a set $\lambda(n)$. The* treewidth *of $\mathcal{I}$ is the smallest width of any of its tree decompositions.*

The next theorem is implicit in the works of Courcelle [21, 23].

THEOREM 6 (COURCELLE). *Satisfiability of Monadic-Second Order logic on countable interpretations of bounded treewidth is decidable.*

An early version of this result has been shown in [21] for a slightly different notion of *width*. A modern account of the relevant proof techniques that uses our above notion of treewidth is given in [23] for the case of finite graphs. Formulating the proof of [21] in these terms, one can show Theorem 6 [22]. We omit the details of this extensive argumentation which is well beyond the scope of the present work.

A set of Datalog rules can be viewed as a (possibly infinite) collection of conjunctive queries that are obtained by expanding rules by repeated backward-chaining. The following definition endows expansions with a useful tree structure.

Definition 6. *Let Σ be a set of Datalog rules with at most one atom in the head. For convenience, we assume that rule with empty head have the head $\bot$ and we treat this like a nullary atom.*

An expansion tree *is a tree structure where each node is labeled with an atom (possibly $\bot$). Every rule $\rho \in \Sigma$ is associated with an expansion tree $T(\rho)$: The root of $T(\rho)$ is labeled with the head atom of ρ. For each body atom A of ρ, the root of $T(\rho)$ has a direct child node with label A.*

Let λ be an atom (possibly $\bot$). The set $T(\Sigma, \lambda)$ of expansion trees for Σ and λ *is defined inductively:*

1. *The tree that consists of a single node labeled with λ is in $T(\Sigma, \lambda)$.*
2. *Let $T \in T(\Sigma, \lambda)$ with l a leaf node of T labeled $p(\mathbf{t})$ and let $\rho = \forall \mathbf{x}.\varphi \to p(\mathbf{t}')$ be a variant of a rule in Σ where all variables have been renamed to be distinct from variables in T. If θ is the most general unifier of $p(\mathbf{t})$ and $p(\mathbf{t}')$, then an expansion tree T' is obtained from T by replacing l with $T(\rho)$, and by applying the unifier θ to all node labels.*

A partial expansion *of Σ and λ is the conjunction of all leaf labels of some expansion tree of Σ and λ. An* expansion *is a partial expansion that does not contain head predicates of Σ. An expansion of a POMSOQ $\mathfrak{Q}$ is an expansion of $\Sigma(\mathfrak{Q})$ of Definition 4 with atom $\lambda = \bot$.*

It is well known that a set of Datalog rules is equivalent to the infinite conjunctions of its partial expansions, i.e., that an atom $p(\mathbf{c})$ follows from a database D and rules Σ if and only if there is an expansion $\varphi[\mathbf{x}, \mathbf{y}]$ for Σ and $p(\mathbf{c})$ such that the query $\exists \mathbf{y}.\varphi[\mathbf{c}/\mathbf{x}, \mathbf{y}]$ matches D.

Every expansion $\varphi[\mathbf{x}]$ with variables $\mathbf{x}$ can naturally be associated with an interpretation structure $\mathcal{I}(\varphi)$: its domain $\Delta^{\mathcal{I}(\varphi)}$ is $\mathbf{x} \cup \mathbf{C}$, for each constant $c \in \mathbf{C}$ we set $c^{\mathcal{I}(\varphi)} := c$, and we have $\mathbf{t} \in p^{\mathcal{I}(\varphi)}$ exactly if $p(\mathbf{t})$ occurs in φ. The treewidth of $\mathcal{I}(\varphi)$ is bounded by the sum $|\mathbf{C}| + n_v$ of the number $|\mathbf{C}|$ of constant symbols and the maximal number n_v of variables in individual rules of Σ.[2] Indeed, the expansion tree can be turned into a tree decomposition by associating each node n with the set of all constant symbols and the variables that occur in the labels of n or any of its direct children. It is easy to verify that this is a tree decomposition. We use $\mathsf{tw}(\Sigma, \lambda) := |\mathbf{C}| + n_v$ to denote the uniform treewidth bound that is thus obtained for expansions of Σ and λ.

Theorem 7. *The query subsumption problem for POMSOQs is decidable.*

Proof. Consider two POMSOQs $\mathfrak{Q}[\mathbf{x}]$ and $\mathfrak{P}[\mathbf{x}]$. $\mathfrak{P}$ subsumes $\mathfrak{Q}$ if $\forall \mathbf{x}.\mathfrak{Q}[\mathbf{x}] \to \mathfrak{P}[\mathbf{x}]$ is a tautology, i.e., if $\exists \mathbf{x}.\mathfrak{Q}[\mathbf{x}] \wedge \neg\mathfrak{P}[\mathbf{x}]$ is unsatisfiable.

We show that, if $\exists \mathbf{x}.\mathfrak{Q}[\mathbf{x}] \wedge \neg\mathfrak{P}[\mathbf{x}]$ is satisfiable, then it has a model of treewidth at most $\mathsf{tw}(\Sigma(\mathfrak{Q}))$. Thus assume that there is an interpretation $\mathcal{I}$ with $\mathcal{I} \models \exists \mathbf{x}.\mathfrak{Q}[\mathbf{x}] \wedge \neg\mathfrak{P}[\mathbf{x}]$. By Theorem 4, there must be an expansion $\varphi[\mathbf{x}, \mathbf{y}]$ of $\mathfrak{Q}[\mathbf{x}]$ such that $\mathcal{I} \models \exists \mathbf{x}, \mathbf{y}.\varphi[\mathbf{x}, \mathbf{y}] \wedge \neg\mathfrak{P}[\mathbf{x}]$. Then there is a variable assignment $\mathcal{Z}$ such that $\mathcal{I}, \mathcal{Z} \models \varphi[\mathbf{x}, \mathbf{y}] \wedge \neg\mathfrak{P}[\mathbf{x}]$. In particular, $\mathcal{I}, \mathcal{Z} \models \varphi[\mathbf{x}, \mathbf{y}]$. This holds exactly if there is a homomorphism π from $\mathcal{I}(\varphi)$ to $\mathcal{I}$ (that agrees with $\mathcal{Z}$ on variables).

By construction, $\mathcal{I}(\varphi), \mathcal{Z}_{\text{id}} \models \varphi[\mathbf{x}, \mathbf{y}]$ where $\mathcal{Z}_{\text{id}}(z) = z$ for each variable z in φ. By Theorem 4, $\mathcal{I}(\varphi), \mathcal{Z}_{\text{id}} \models \mathfrak{Q}[\mathbf{x}]$.

We show that $\mathcal{I}(\varphi), \mathcal{Z}_{\text{id}} \models \neg\mathfrak{P}[\mathbf{x}]$. For a contradiction, suppose that $\mathcal{I}(\varphi), \mathcal{Z}_{\text{id}} \models \mathfrak{P}[\mathbf{x}]$. Since there is a homomorphism π from $\mathcal{I}(\varphi)$ to $\mathcal{I}$, Theorem 1 implies $\mathcal{I}, \pi \circ \mathcal{Z}_{\text{id}} \models \mathfrak{P}[\mathbf{x}]$. By construction of π, the assignment $\pi \circ \mathcal{Z}_{\text{id}}$ agrees with $\mathcal{Z}$ on all variables of $\mathbf{x}$, and thus $\mathcal{I}, \mathcal{Z} \models \mathfrak{P}[\mathbf{x}]$; a contradiction.

We have thus shown that $\mathcal{I}(\varphi), \mathcal{Z}_{\text{id}} \models \mathfrak{Q}[\mathbf{x}] \wedge \neg\mathfrak{P}[\mathbf{x}]$ and thus $\mathcal{I}(\varphi) \models \exists \mathbf{x}.\mathfrak{Q}[\mathbf{x}] \wedge \neg\mathfrak{P}[\mathbf{x}]$. $\mathcal{I}(\varphi)$ is finite, hence countable. Moreover, the treewidth of $\mathcal{I}(\varphi)$ is bounded by $\mathsf{tw}(\Sigma(\mathfrak{Q}))$. Therefore, if $\exists \mathbf{x}.\mathfrak{Q}[\mathbf{x}] \wedge \neg\mathfrak{P}[\mathbf{x}]$ is satisfiable, then it is satisfied by a countable model of treewidth at most $\mathsf{tw}(\Sigma(\mathfrak{Q}))$. By Theorem 6, the latter can be decided. $\square$

The complexity of POMSOQ subsumption remains to be determined. A natural lower bound is the known ExpSpace-Hardness of subsumption for C2RPQs [18].

7. POMSOQ REWRITABILITY

The expressive power of POMSOQs can be used to capture the semantics of certain TGDs. In this section, we first make this notion of rewritability precise.

Definition 7. *Given a set Σ of TGDs and a conjunctive query $Q[\mathbf{x}]$, a POMSOQ $\mathfrak{Q}_{Q,\Sigma}$ is a* rewriting *of Σ and Q if, for all databases D and potential query answers $\mathbf{c}$, we have $D \cup \Sigma \models Q[\mathbf{c}/\mathbf{x}]$ iff $D \models \mathfrak{Q}_{Q,\Sigma}[\mathbf{c}/\mathbf{x}]$. The rules Σ are* POMSOQ-rewritable *if every conjunctive query Q admits a rewriting for Σ and Q.*

It follows from Theorem 5 that query answering is decidable for POMSOQ-rewritable sets of TGDs and can be done in polynomial time w.r.t. the size of the database.

Rewritability of conjunctive queries entails rewritability of POMSOQ, i.e., the conditions of Definition 7 hold even when considering POMSOQs instead of CQs. Indeed, CQs that occur in rule bodies in a POMSOQ can generally be replaced using a POMSOQ for the respective CQ, provided that the extensionally quantified variables in the CQ are not used anywhere else in the rule body:

Lemma 8 (Replacement Lemma). *Consider a set Σ of TGDs, a conjunctive query $Q = \exists \mathbf{y}.\psi[\mathbf{x}, \mathbf{y}]$, and a POMSOQ $\mathfrak{Q}[\mathbf{x}]$ that is a rewriting for Σ and Q. Then Q and $\mathfrak{Q}$ are equivalent in all models of Σ, i.e., $\Sigma \models \forall \mathbf{x}.Q[\mathbf{x}] \leftrightarrow \mathfrak{Q}[\mathbf{x}]$.*

Let $\psi[\mathbf{t}/\mathbf{x}, \mathbf{y}'/\mathbf{y}]$ be the conjunction of Q with variables $\mathbf{x}$ replaced by terms $\mathbf{t}$ and variables $\mathbf{y}$ replaced by variables $\mathbf{y}'$. We say that $\psi[\mathbf{t}/\mathbf{x}, \mathbf{y}'/\mathbf{y}]$ is a match *in a Datalog rule ρ if ρ is of the form $\psi[\mathbf{t}/\mathbf{x}, \mathbf{y}'/\mathbf{y}] \wedge \varphi \to \chi$ where the $\mathbf{y}'$ occur neither in φ nor in χ.*

Given some POMSOQ $\mathfrak{P}[\mathbf{z}]$ over Σ, let $\mathfrak{P}'[\mathbf{z}]$ denote a POMSOQ obtained by replacing a match $\psi[\mathbf{t}/\mathbf{x}, \mathbf{y}'/\mathbf{y}]$ of Q in some rule of $\mathfrak{P}$ by $\mathfrak{Q}[\mathbf{t}/\mathbf{x}]$, where we assume w.l.o.g. that the bound variables in $\mathfrak{Q}$ do not occur in $\mathfrak{P}$. Then $\mathfrak{P}$ and $\mathfrak{P}'$ are equivalent in all models of Σ, i.e., $\Sigma \models \forall \mathbf{z}.\mathfrak{P}[\mathbf{z}] \leftrightarrow \mathfrak{P}'[\mathbf{z}]$.

Proof. We first show that $\Sigma \models \forall \mathbf{z}.Q[\mathbf{z}] \leftrightarrow \mathfrak{Q}[\mathbf{z}]$. For the one direction, consider a model $\mathcal{I} \models \Sigma$ and a variable assignment $\mathcal{Z}$ such that $\mathcal{I}, \mathcal{Z} \models \mathfrak{Q}[\mathbf{z}]$. According to Theorem 4, there is an expansion $\varphi[\mathbf{y}]$ of $\Sigma(()\mathfrak{Q})$ such that $\mathcal{I}, \mathcal{Z} \models \mathfrak{Q}[\mathbf{y}]$ (where we assume w.l.o.g. that $\mathcal{Z}$ assigns the appropriate domain elements to the fresh variables that φ may contain). Using notation as in Section 6, we find a model $\mathcal{I}(\varphi)$ to which φ matches under the variable assignment $\mathcal{Z}_{\text{id}}$ with $\mathcal{Z}_{\text{id}}(y) = y$ for each y in φ. Then $\mathcal{I}(\varphi), \mathcal{Z}_{\text{id}} \models \mathfrak{Q}[\mathbf{z}]$.

Let $D(\mathcal{I}(\varphi))$ be the model $\mathcal{I}(\varphi)$ considered as a database containing a fact for each of the finitely many relations in $\mathcal{I}(\varphi)$. Introducing finitely many new constants for this purpose is not a problem. Let $\mathbf{c_z}$ denote the constants in $D(\mathcal{I}(\varphi))$ that correspond to $\mathcal{Z}_{\text{id}}(\mathbf{z})$. Then $D(\mathcal{I}(\varphi)) \models \mathfrak{Q}[\mathbf{c_z}/\mathbf{z}]$.

Since $\mathfrak{Q}$ is a rewriting of Q under Σ, we have $D(\mathcal{I}(\varphi)), \Sigma \models Q[\mathbf{c_z}/\mathbf{z}]$. Consider a universal model $\mathcal{J}$ of $D(\mathcal{I}(\varphi)), \Sigma$. Then $\mathcal{J} \models Q[\mathbf{c_z}/\mathbf{z}]$. Moreover, there is a homomorphism from $\mathcal{J}$ to $\mathcal{I}$. Indeed,

[2] A similar observation has been made by Afrati et al. [2].

the mapping $\mathcal{Z}$ induces a homomorphism π from $\mathcal{I}(\varphi)$ to $\mathcal{I}$. This mapping can be extended to a homomorphism π' from $\mathcal{J}$ to $\mathcal{I}$, since $\mathcal{I}$ is a model of Σ. Due to Theorem 1, the query match $\mathcal{J} \models Q[\mathbf{c_z}/\mathbf{z}]$ implies $\mathcal{J} \models Q[\pi'(\mathbf{c_z})/\mathbf{z}]$. Since $\pi'(\mathbf{c_z}) = \pi(\mathbf{c_z}) = \mathcal{Z}(\mathbf{z})$, this shows the claim $\mathcal{I}, \mathcal{Z} \models Q[\mathbf{z}]$.

The other direction can be shown in a similar way, somewhat simplified due to the fact that one does not need to construct an intermediate model $\mathcal{J}$ of Σ to obtain the match for $\mathfrak{Q}$.

Now the rest of the claim follows from Theorem 4. It remains to show the claimed equivalence for $\Sigma(\mathfrak{P})$ and $\Sigma(\mathfrak{P}')$. This is a direct consequence of the Replacement Theorem of first-order logic that allows us to replace the sub-formula $\exists \mathbf{y}'.\psi[\mathbf{t}/\mathbf{x}, \mathbf{y}'/\mathbf{y}]$ by $p_{\mathfrak{Q}}(\mathbf{t})$, both of which have just shown to be equivalent. $\square$

How relevant is this new notion of POMSOQ-rewritability in practice? The fact that every first-order conjunctive query can be expressed as POMSOQ implies that every first-order rewritable rule set is also POMSOQ-rewritable. It is undecidable whether a given TGD set is FO-rewritable (in which case it is also referred to as *finite unification set*), an iterative backward chaining algorithm can be defined that terminates on FO-rewritable rule sets and provides the rewritten FO formula [7]. Moreover, a significant body of research has unveiled a variety of sufficient syntactically checkable criteria for FO-rewritability. Among the known FO-rewritable TGD fragments are *atomic-hypothesis rules* and *domain restricted rules* [7] as well as *linear Datalog+/–* [13] and *sticky sets of TGDs* and *sticky-join sets of TGDs* [14, 15]. The new notion of POMSOQ-rewritability naturally captures all of these but goes significantly beyond. Further, genuinely POMSOQ-rewritable classes of rule sets will be introduced in the subsequent sections.

8. FROM DATALOG TO POMSOQS

In this section, we present a method for finding a POMSOQ rewriting for certain sets of Datalog rules and arbitrary conjunctive queries. In the base case, this leads to POMSOQs of degree 1. We then leverage this idea for constructing POMSOQs of higher degree recursively by transforming a set of Datalog rules "layer by layer". Without loss of generality, we assume that the heads of Datalog rules contain at most one atom (rules of the form $\psi \to \varphi_1 \wedge \varphi_2$ can be simplified to $\psi \to \varphi_1, \psi \to \varphi_2$ which is not possible for arbitrary TGDs).

Definition 8. *A set of Datalog rules Σ is j-*oriented *for the integer j if all head predicates have the same arity n, and $1 \leq j \leq n$, and we have: if a rule's body contains an atom $p(\mathbf{t})$ for some head predicate p and the rule's head contains an atom $q(\mathbf{t}')$, then $\mathbf{t}$ and $\mathbf{t}'$ agree on all positions other than possibly j.*

Intuitively speaking, recursive derivations in j-oriented rule sets can only modify the content of a single position j while keeping all other arguments fixed in all derived facts.

Example 9. *The following rule set Σ_{family} is 3-oriented. We use atoms parentsSon(x, y, z) and parentsDghtr(x, y, z) to denote that z is the son and daughter of x and y, respectively.*

$$\begin{aligned} \mathit{parentsSon}(x, y, z) \wedge \mathit{hasBrother}(z, z') &\to \mathit{parentsSon}(x, y, z') \\ \mathit{parentsSon}(x, y, z) \wedge \mathit{hasSister}(z, z') &\to \mathit{parentsDghtr}(x, y, z') \\ \mathit{parentsDghtr}(x, y, z) \wedge \mathit{hasBrother}(z, z') &\to \mathit{parentsSon}(x, y, z') \\ \mathit{parentsDghtr}(x, y, z) \wedge \mathit{hasSister}(z, z') &\to \mathit{parentsDghtr}(x, y, z') \end{aligned}$$

This can be used to construct POMSOQs for atomic CQs as follows.

Definition 9. *Given a j-oriented set Σ of Datalog rules and a head predicate p of Σ, a POMSOQ $\mathfrak{Q}_p(\Sigma)$ is defined as follows. Let $\mathtt{U}_q$ be a set variable for each head predicate q in Σ, let $\mathtt{V}_i$ be a set variable for each $i \in \{1, \ldots, \mathsf{ar}(p)\}$ with $i \neq j$, and let $\mathbf{z} = z_1, \ldots, z_{\mathsf{ar}(p)}$ be object variables that do not occur in Σ (the free variables of the query). Let $\tilde{z}_j$ be an additional variable not occurring in Σ. The rules of $\mathfrak{Q}_p(\Sigma)[\mathbf{z}]$ are:*

- *a rule $U_p(z_j) \to$ with empty head;*
- *for each set variable $\mathtt{V}_i$, a rule $\to \mathtt{V}_i(z_i)$ with empty body;*
- *for each $\forall \mathbf{x}.\psi \to q(t_1, \ldots, t_n) \in \Sigma$, a rule $\psi' \to \mathtt{U}_q(t_j)$ where ψ' is obtained from ψ by replacing each atom of the form $q'(t_1, \ldots, t'_j, \ldots, t_n)$ for a head predicate q' by $\mathtt{U}_{q'}(t'_j)$, and by adding, for each term t_i with $i \neq j$, a new body atom $\mathtt{V}_i(t_i)$;*
- *for each head predicate q', a rule $q'(z_1, \ldots, \tilde{z}_j, \ldots, z_n) \to \mathtt{U}_{q'}(\tilde{z}_j)$.*

This operation allows us to express the extension of a predicate p by means of a POMSOQ.

Theorem 9. *If Σ is j-oriented and p is a head predicate, then $\mathfrak{Q}_p(\Sigma)[\mathbf{z}]$ is a rewriting for Σ and $p(\mathbf{z})$.*

Proof. We show that, for any database D, list of constants $\mathbf{c} = c_1, \ldots, c_n$, and predicate $\hat{U}_q$, we have $D \cup \Sigma \models q(\mathbf{c})$ iff $D \cup \Sigma(\mathfrak{Q}_p(\Sigma)) \models \hat{U}_q(c_j, c_1, \ldots, c_n)$. From this the claim follows using Theorem 4. The proof is by an easy induction over the derivation of $q(\mathbf{c})$.

Clearly, $\Sigma(\mathfrak{Q}_p(\Sigma)) \models \hat{V}_i(\mathbf{d})$ iff $\mathbf{d}$ is of the form $d_i, d_1, \ldots, d_i, \ldots, d_n$. If there is a rule $\psi \to q(t_1, \ldots, t_n) \in \Sigma$ that has an instance $\psi_c \to q(c_1, \ldots, c_n)$, then $\Sigma(\mathfrak{Q}_p(\Sigma))$ contains a rule $\psi' \to \hat{U}_q(t_j, t_1, \ldots, t_n)$ with an instance $\psi'_c \to \hat{U}_q(c_j, c_1, \ldots, c_n)$. It is easy to verify the claim. $\square$

Example 10. *Considering the 3-oriented rule set from Example 9, we obtain $\mathfrak{Q}_{parentsSon}(\Sigma)[z_1, z_2, z_3]$ with the rules*

$$\begin{aligned} \mathtt{U}_{parentsSon}(z_3) &\to \\ &\to \mathtt{V}_1(z_1) \\ &\to \mathtt{V}_2(z_2) \\ \mathtt{V}_1(x) \wedge \mathtt{V}_2(y) \wedge \mathtt{U}_{parentsSon}(z) \wedge \mathit{hasBrother}(z, z') &\to \mathtt{U}_{parentsSon}(z') \\ \mathtt{V}_1(x) \wedge \mathtt{V}_2(y) \wedge \mathtt{U}_{parentsSon}(z) \wedge \mathit{hasSister}(z, z') &\to \mathtt{U}_{parentsDghtr}(z') \\ \mathtt{V}_1(x) \wedge \mathtt{V}_2(y) \wedge \mathtt{U}_{parentsDghtr}(z) \wedge \mathit{hasBrother}(z, z') &\to \mathtt{U}_{parentsSon}(z') \\ \mathtt{V}_1(x) \wedge \mathtt{V}_2(y) \wedge \mathtt{U}_{parentsDghtr}(z) \wedge \mathit{hasSister}(z, z') &\to \mathtt{U}_{parentsDghtr}(z') \\ \mathit{parentsSon}(z_1, z_2, \tilde{z}_3) &\to \mathtt{U}_{parentsSon}(\tilde{z}_3) \\ \mathit{parentsDghtr}(z_1, z_2, \tilde{z}_3) &\to \mathtt{U}_{parentsDghtr}(\tilde{z}_3) \end{aligned}$$

Note that the $\mathtt{V}$ predicates are not really needed here, since rule bodies do not impose any conditions on the respective variables. In general, one could always replace $\mathtt{V}$ by using the respective free variables if no constants are involved.

Using the Replacement Lemma 8, we can extend this to arbitrary conjunctive queries:

Theorem 10. *Every j-oriented rule set is POMSOQ-rewritable.*

Proof. A rewriting for a j-oriented rule set Σ and a CQ $Q[\mathbf{x}] = \exists \mathbf{y}.p_1(\mathbf{t_1}) \wedge \ldots \wedge p_m(\mathbf{t_m})$ is obtained from the POMSOQ $\neg \forall \mathbf{y}.p_1(\mathbf{t_1}) \wedge \ldots \wedge p_m(\mathbf{t_m}) \to$ by replacing all head atoms $p_1(\mathbf{t_1})$ with the POMSOQ $\mathfrak{Q}_{p_i}(\Sigma)[\mathbf{t_i}/\mathbf{x}]$ as in Lemma 8. We assume a fixed sequence of replacement steps in the construction of the rewriting. Let $\mathfrak{Q}_0$ denote the initial rewriting of the CQ, let $\mathfrak{Q}_i$ with $1 \leq i$ denote the result after each subsequent replacement step, and let $\mathfrak{Q}$ be the final result.

One can show $\Sigma \models \forall \mathbf{x}.Q[\mathbf{x}] \leftrightarrow \mathfrak{Q}_i[\mathbf{x}]$ by induction over i. The base case follows since $\mathfrak{Q}_0$ is clearly equivalent to Q. The induction steps follow from Lemma 8 and Theorem 9.

Hence $\Sigma \models \forall \mathbf{x}.Q[\mathbf{x}] \leftrightarrow \mathfrak{Q}[\mathbf{x}]$, i.e., for every interpretation $\mathcal{I} \models \Sigma$ and every variable assignment $\mathcal{Z}$ for $\mathcal{I}$, we have $\mathcal{I}, \mathcal{Z} \models Q[\mathbf{x}]$ iff $\mathcal{I}, \mathcal{Z} \models \mathfrak{Q}[\mathbf{x}]$ $(*)$.

We show the condition of Definition 7. Thus consider an arbitrary database D and a potential query answer $\mathbf{c}$. Without loss of generality, we assume that D contains no fact of the form $q'(\mathbf{d})$ for some head predicate q' of Σ. Indeed, if it does, we can replace q' by a fresh predicate q'_D and add a rule $\forall \mathbf{x} q'_D(\mathbf{x}) \to q'(\mathbf{x})$ to Σ. This modification clearly preserves j-orientedness.

If $D \models \mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ then clearly $D \cup \Sigma \models \mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ and thus $D \cup \Sigma \models Q[\mathbf{c}/\mathbf{x}]$ by $(*)$. Conversely, assume that $D \cup \Sigma \models Q[\mathbf{c}/\mathbf{x}]$. Then $D \cup \Sigma$ is satisfiable since Σ contains no constraints (rules with empty head). More precisely, for every interpretation $\mathcal{I} \models D$, there is an interpretation $\mathcal{I}' \models \Sigma, D$ that coincides with $\mathcal{I}$ on all constants and all predicates that are not head predicates in Σ, where we use that D contains no head predicates of Σ. By the assumption $\mathcal{I}' \models Q[\mathbf{c}/\mathbf{x}]$. By $(*)$ $\mathcal{I}' \models \mathfrak{Q}[\mathbf{c}/\mathbf{x}]$. Since $\mathfrak{Q}$ contains only predicates for which $\mathcal{I}$ and $\mathcal{I}'$ agree, this implies) $\mathcal{I} \models \mathfrak{Q}[\mathbf{c}/\mathbf{x}]$. Since $\mathcal{I}$ was arbitrary, this establishes the claim. □

One can rarely expect that a given rule set is j-oriented, but a similar result can be obtained in cases where only part of a set of rules can be transformed into a query.

Theorem 11. *Let $\Sigma_1 \cup \Sigma_2$ be a set of TGDs where Σ_2 is a set of j-oriented Datalog rules such that no head predicate of Σ_2 occurs in a body of a rule in Σ_1. Then, for every POMSOQ $\mathfrak{Q}$ there is a POMSOQ $\mathfrak{P}_{\mathfrak{Q},\Sigma_2}$ with the following property: for all databases D for which $D \cup \Sigma_1 \cup \Sigma_2$ is satisfiable, and for all query answers $\mathbf{c}$, we have $D \cup \Sigma_1 \cup \Sigma_2 \models \mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ iff $D \cup \Sigma_1 \models \mathfrak{P}_{\mathfrak{Q},\Sigma_2}[\mathbf{c}/\mathbf{x}]$.*

Proof. The proof proceeds as in the case of Theorem 10, using Lemma 8 and Theorem 9, and arguing that any model of $D \cup \Sigma_1$ can be extended to a model of $D \cup \Sigma_1 \cup \Sigma_2$. □

This result can be applied to rewrite oriented sets of Datalog rules iteratively, since Σ_1 may again contain a j-oriented set of rules that can be rewritten (where j does not have to be as before). The following definition elaborates this idea.

Definition 10. *For a set of TGDs Σ, let $\prec$ be the smallest transitive relation on Σ such that $\rho \prec \rho'$ holds whenever a predicate in the head of ρ occurs in the body of ρ'. Two rules ρ, ρ' are $\prec$-equivalent, written $\rho \approx \rho'$, if $\rho \prec \rho'$ and $\rho' \prec \rho$. The set Σ is* fully oriented *if every equivalence class $[\rho]_\prec = \{\rho' \mid \rho \approx \rho'\}$ is j-oriented (not necessarily for the same j and predicate arity).*

The next theorem establishes that the property of being fully oriented can be easily checked and is a sufficient criterion for POMSOQ rewritability.

Theorem 12. *Given a rule set Σ, it can be detected in polynomial time if Σ is fully oriented. Every fully oriented set Σ is POMSOQ-rewritable.*

Proof. Clearly, the relation $\prec$ can be constructed in polynomial time by checking $\rho \prec \rho'$ for each pair of rules and constructing the transitive closure. The equivalence classes $[\rho]_\prec$ are obtained from this in linear time. It is clear that j-orientedness can be checked for each set of rules in polynomial time.

It remains to show the second part of the claim. We say that $[\rho]_\prec$ is *maximal* if, for all $\rho' \in \Sigma$, we have that $\rho \prec \rho'$ implies $\rho' \in [\rho]_\prec$. If Σ is fully oriented and $[\rho]_\prec$ is maximal, then $\Sigma_1 := \Sigma \setminus [\rho]_\prec$ and $\Sigma_2 := [\rho]_\prec$ satisfy the preconditions of Theorem 11. Moreover, Σ_1 again is fully oriented. Thus, one can apply Theorem 10 (initially) and Theorem 11 (iteratively) to obtain the required rewriting. □

This result can be used to show that positive monadic second-order queries have the same expressivity as fully oriented Datalog. Importantly, this also shows that every POMSOQ-rewritable set of TGDs can equivalently be expressed as a set of rules that can be transformed into a POMSOQ using Theorem 12.

Theorem 13. *For every POMSOQ $\mathfrak{Q}$, the rule set $\Sigma(\mathfrak{Q})$ of Definition 4 is fully oriented. Moreover, for every POMSOQ-rewritable set Σ of TGDs, there is a fully oriented set of Datalog rules Σ' such that:*

- *every predicate p in Σ has a corresponding head predicate q_p in Σ' that does not occur in Σ,*
- *for every database D and conjunctive query $Q[\mathbf{x}]$ that do not contain predicates of the form q_p, and for every list of constants $\mathbf{c}$, we have $D \cup \Sigma \models Q[\mathbf{c}/\mathbf{x}]$ iff $D \cup \Sigma' \models Q'[\mathbf{c}/\mathbf{x}]$ where Q' is obtained from Q by replacing all predicates p with q_p.*

9. STRATIFYING RULE SETS

Though we have just shown that fully directed Datalog has the same expressivity as POMSOQ, rule sets in practice rarely have this specific form. Indeed, even a simple transitivity rule $p(x,y) \wedge p(y,z) \to p(x,z)$ cannot be rewritten to a POMSOQ along the lines of Theorem 12, although it is POMSOQ-rewritable. In this section, we extend our rewriting approach to cover such cases.

Our strategy is to transform sets Datalog of rules into a *stratified* form that enforces a certain order of rule applications. Every "stratum" of rule applications is j-oriented for a particular argument position j.

Stratification still requires certain syntactic regularities. We say that a rule is *complex* if it has more than one body atom. A set of Datalog rules Σ is *basic* if all head predicates have the same arity n, and every non-complex rule $p(\mathbf{t}) \to q(\mathbf{s})$ is such that $\mathbf{t} = \mathbf{s}$. In this section, we restrict to such basic rule sets Σ, and use n for the respective arity throughout.

Definition 11. *The* 0-stratum *of Σ, denoted $\Sigma\langle 0\rangle$, consists of the rules:*

- *$p_{\mathsf{db}}(\mathbf{x}) \to p_0(\mathbf{x})$ for each head predicate p of Σ;*
- *for every rule $\varphi \to p(\mathbf{t}) \in \Sigma$, the rule $\varphi' \to p_0(\mathbf{t})$ where φ' is obtained from φ by replacing every head atom $q(\mathbf{s})$ by$q_0(\mathbf{s})$ if $\mathbf{s} = \mathbf{t}$, and $q_{\mathsf{db}}(\mathbf{s})$ otherwise.*

*For $k \in \{1, \ldots, n\}$, the k-*stratum *of Σ, denoted $\Sigma\langle k\rangle$, consists of the rules:*

- *$p_0(\mathbf{x}) \to p_k(\mathbf{x})$ for each head predicate p of Σ;*
- *for every rule $\varphi \to p(\mathbf{t}) \in \Sigma$, the rule $\varphi' \to p_k(\mathbf{t})$ where φ' is obtained from φ by replacing every head atom $q(\mathbf{s})$ by:*
  - *(i) $q_k(\mathbf{s})$ if $\mathbf{s}$ agrees with $\mathbf{t}$ on all positions other than possibly i, and*
  - *(ii) $q_0(\mathbf{s})$ otherwise.*

Given two distinct numbers $i, j \in \{0, \ldots, n\}$, the set $\Sigma\langle i \to j\rangle$ consists of the rules $\{p_i(\mathbf{x}) \to p_j(\mathbf{x}) \mid p$ head predicate in $\Sigma\}$.

*Let $\pi = \langle \pi(1), \ldots, \pi(n)\rangle$ be a permutation of the numbers $1, \ldots, n$, and let $\pi(0)$ denote 0. The π-*stratification *of Σ, denoted $\Sigma \uparrow \pi$, is the set*

$$\bigcup_{k=0}^{n} \Sigma\langle \pi(k)\rangle \cup \bigcup_{k=1}^{n-1} \Sigma\langle \pi(k) \to \pi(k+1)\rangle.$$

EXAMPLE 11. *The following is the $\langle 1,2\rangle$-stratification of the rule $p(x,y) \land p(y,z) \to p(x,z)$:*

$$\begin{array}{ll} p_{\mathsf{db}}(x,y) \to p_0(x,y) & p_{\mathsf{db}}(x,y) \land p_{\mathsf{db}}(y,z) \to p_0(x,z) \\ p_0(x,y) \to p_1(x,y) & p_0(x,y) \land p_1(y,z) \to p_1(x,z) \\ p_0(x,y) \to p_2(x,y) & p_2(x,y) \land p_0(y,z) \to p_2(x,z) \\ p_1(x,y) \to p_2(x,y). & \end{array}$$

where each line corresponds to a stratum. This set of rules is fully directed and it is easy to see that this is always the case for π-stratifications. In particular, each k-stratum is k-oriented and the dependency order $\prec$ is such that $\rho \prec \rho'$ holds only if ρ is in a lower stratum than ρ'.

Example 11 is also correct in the following sense:

DEFINITION 12. *A π-stratification $\Sigma \uparrow \pi$ is* correct *if, for every database D, conjunctive query $Q[\mathbf{x}]$, and potential query answer $\mathbf{c}$, we have $D, \Sigma \models Q[\mathbf{c}/\mathbf{x}]$ iff $D', \Sigma \uparrow \pi \models Q'[\mathbf{c}/\mathbf{x}]$ where D' is obtained by replacing all occurrences of head predicates p with p_{db}, and Q' is obtained by replacing all occurrences of head predicates p with $p_{\pi(n)}$.*

A set of rules is stratifiable *if it has a correct stratification. A (not necessarily basic) set of rules Σ is* fully stratifiable *if every equivalence class $[\rho]_\prec = \{\rho' \mid \rho \approx \rho'\}$ is stratifiable.*

Using an iterative rewriting as in Theorem 12, we obtain the following.

THEOREM 14. *Every fully stratifiable set of TGDs is POMSOQ-rewritable.*

Unfortunately, not all stratifications are correct. For example, the rule $l(x,y) \land p(y,z) \land r(z,u) \to p(x,u)$ clearly has no correct stratification since the body atom $p(y,z)$ will always belong to a lower stratum than the head, so that no arbitrary recursion is possible.

Our goal is to find a way to detect if a set of rules can be correctly stratified. In other words, we want to find out if every (partial) expansion of Σ has a corresponding (partial) expansion of $\Sigma \uparrow \pi$. Since there are infinitely many expansions in general, this is not a criterion that we can check effectively. However, we can obtain a sufficient condition by restricting attention to a particular finite set of *essential* expansions, defined next.

DEFINITION 13. *Consider an expansion tree T for Σ and λ as in Definition 6. Every edge $n_1 \to n_2$ in T is constructed using a rule $\mathsf{rule}(n_1) \in \Sigma$. The edge $n_1 \to n_2$ is* internal *if n_2 is not a leaf. The edge $n_1 \to n_2$ is* invariant *if the head atom of $\mathsf{rule}(n_1)$ has the form $p(\mathbf{t})$ and the body atom of $\mathsf{rule}(n_1)$ for which n_2 was created has the form $q(\mathbf{t})$, with the same $\mathbf{t}$.*

An expansion tree T is essential *if*

(1) each node has at most one child that is not a leaf, so that internal edges form a chain $n_0 \to \ldots \to n_\ell$;

(2) $\mathsf{rule}(n_0)$ and $\mathsf{rule}(n_\ell)$ are complex;

(3) for each $k \in \{1,\ldots,\ell-1\}$, $n_k \to n_{k+1}$ is invariant and all nodes $n_1,\ldots,n_\ell$ have different labels.

The child nodes of n_ℓ are called bottom leafs. *All other leafs are* side leafs.

Figure shows three essential expansion trees. The rules that were applied resemble that of Example 11, but only tree (A) is actually an expansion tree for that particular stratification.

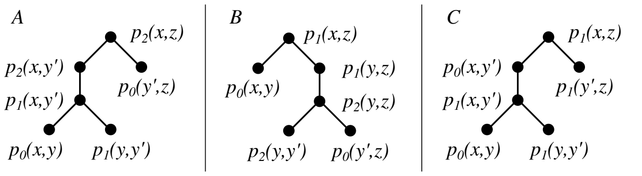

Figure 2: Examples of essential expansion trees

It is easy to see that there are at most exponentially many essential expansion trees for Σ and an atom $p(\mathbf{x})$. Indeed, each such essential expansion tree is uniquely determined by two (initial and final) complex rules and a chain of invariant rule applications. Since the chain of invariant rule application is required to have no repeated node labels, every rule in Σ can occur at most once in this chain.

We use essential expansion trees in two distinct conditions that allow us to find correct stratifications. The first property ensures that every expansion tree of Σ has a corresponding expansion tree w.r.t. the union of all possible stratifications. The latter is called a *pre-stratification* – an expansion tree where stratified rules may be applied in any order. The second property ensures that applications of stratified rules that are not in the right order can be swapped, which allows us to eventually obtain an expansion tree for a particular stratification $\Sigma \uparrow \pi$.

DEFINITION 14. *Let $\Sigma_{ps} := \bigcup_{i=1}^{n} \Sigma\langle i\rangle \cup \bigcup_{i,j\in\{1,\ldots,n\}, i\neq j} \Sigma\langle i \to j\rangle$. A* 0-*tree of Σ is an essential expansion tree T for $\Sigma_{ps} \cup \bigcup_{i=1}^{n} \Sigma\langle i \to 0\rangle$ and $p_k(\mathbf{t})$ for some $k \in \{1,\ldots,n\}$.*

Let L be the set of all leaf labels of T and all labels of the form $q_0(\mathbf{s})$ where $q_i(\mathbf{s})$ is the label of a bottom leaf in T for some $i \in \{1,\ldots,n\}$. A pre-stratification *of a* 0-*tree T is an expansion tree T_{ps} of Σ_{ps} and $p_k(\mathbf{t})$, where each leaf in T_{ps} is labeled with an atom from L. Σ is* pre-stratifiable *if all of its* 0-*trees have a pre-stratification.*

DEFINITION 15. *Let Σ be a set of Datalog rules with head predicates of the same arity n, let $i, j \in \{1,\ldots,n\}$ be indices with $i \neq j$. An expansion tree T is an $\langle i,j\rangle$*-tree *for Σ if there is a head predicate p of Σ such that T is an expansion tree for $\Sigma\langle i\rangle \cup \Sigma\langle j\rangle \cup \Sigma\langle i \to j\rangle$ and $p_j(x_1,\ldots,x_n)$, and T contains complex nodes n_i and n_j with $\mathsf{rule}(n_i) \in \Sigma\langle i\rangle$ and $\mathsf{rule}(n_j) \in \Sigma\langle j\rangle$.*

Let $\mathsf{leafs}(T)$ denote the multiset of leaf labels of T, i.e. the set of labels together with the number of leafs where they occur. Let $\mathsf{leafs}_{i\mapsto j}(T)$ be obtained from $\mathsf{leafs}(T)$ by replacing all atoms of the form $p_i(\mathbf{t})$ with $p_j(\mathbf{t})$, i.e., by incrementing the multiplicity of $p_j(\mathbf{t})$ by that of $p_i(\mathbf{t})$.

An $\langle i,j\rangle$-tree T for $p_j(\mathbf{x})$ is invertible *if there is a $\langle j,i\rangle$-tree T' for $p_i(\mathbf{x})$ such that $\mathsf{leafs}(T')$ is a sub-multiset of $\mathsf{leafs}_{i\mapsto j}(T)$. Σ is* $\langle i,j\rangle$-invertible *if all essential $\langle i,j\rangle$-trees are invertible.*

EXAMPLE 12. *Tree (B) in Fig. is a $\langle 2,1\rangle$-tree based on Example 11. Its inverted expansion tree is tree (A), which is a correct expansion tree in the stratification of Example 11. Tree (C) is a* 0-*tree for that example. A pre-stratification for that tree needs to derive $p_1(x,z)$ from leafs $p_0(x,y)$, $p_0(y,y')$, and $p_1(y',z)$. Note that $p_0(y,y')$ is obtained frmo the bottom leaf label $p_1(y,y')$ here. The pre-stratification is not shown in the figure but is easy to find.*

Combining the above two notions, we obtain a sufficient criterion for checking whether a stratification is correct. The proof proceeds by iteratively transforming arbitrary expansion trees of Σ into expansion trees of $\Sigma \uparrow \pi$ as outlined above. The full formalization of this transformation is provided in the Appendix.

THEOREM 15. *The stratification* $\Sigma \uparrow \pi$ *is correct whenever the following hold:*

- Σ *is pre-stratifiable,*
- *for all* $i > j$, Σ *is* $\langle \pi(i), \pi(j) \rangle$*-invertible.*

THEOREM 16. *It is decidable whether the conditions of Theorem 15 are satisfied for a set of Datalog rules* Σ.

PROOF. Both conditions in Theorem 15 refer to specific essential expansion trees of Σ, and it suffices to consider trees constructed from a root label $p(\mathbf{x})$ where $\mathbf{x}$ consists of variables only. As argued earlier, there are only a finite number of such essential expansion trees.

Given an essential expansion tree T, the conditions of Definition 14 and 15 can be decided. Both criteria can easily be related to checking the entailment of a certain fact from a set of Datalog rules. Indeed, we merely need to replace variables in the labels of leaf nodes by new constants and then check entailment. This approach suffices for checking Definition 14. For Definition 15, one needs to additionally restrict attention to derivations that use every leaf node (input fact) at most once. Yet, this is also clearly decidable by recording, for every fact that is derived, the set of leaf facts that have been contributing to it. Derivations are only allowed if no two premises use overlapping sets of facts, and the newly derived atom is annotated with the union of the leaf sets of its premises. □

How useful can the above criteria be in practice? The proof of Theorem 16 indicates that checking the conditions of Theorem 15 may be exponential. In practice, however, we expect the essential proof trees to be very small, so that the related checks are not a performance issue. Indeed, it seems unlikely that any practical rule set admits many different chains of invariant rule applications for a given atom. Moreover, the checking procedure does not have to be repeated for all possible permutations π. Rather, one can check which pairs $\langle i, j \rangle$ lead to essential $\langle i, j \rangle$-trees that cannot be be inverted in order to derive all relevant restrictions on permutations, from which a correct permutation can easily be retrieved if there is one.

The other practically relevant question is whether our criteria have a chance of covering a relevant amount of rule sets. The restriction to rules where all head predicates have the same arity may seem rather severe. Note, however, that this requirement only applies to individual $\prec$ equivalence classes as in Theorem 14. Moreover, description logics (DLs) provide a relevant application area where predicate arities can only be 1 or 2. In particular, DL *role inclusion axioms* (RIAs) are a very specific kind of Datalog rules that use binary predicates only. A common restriction on sets of RIAs in DL is called *regularity* that can be viewed a special case of our stratification. Kazakov has studied a generalization of regularity that is based on a notion of stratifiability that is similar to ours [29]. This work can apply simpler definitions by exploiting the special structure of RIAs, while we need to consider a much more general case. For example, we explicitly require the preservation of leaf multiplicities in Definition 15, while this follows implicitly for RIAs. DLs also give an example of a situation where our conditions can be checked in polynomial time.

10. BOUNDED TREEWIDTH SETS

This section investigates how well our novel querying notion can be combined with another very general condition that ensures decidability of CQ entailment, namely the bounded treewidth model property.

DEFINITION 16. *A rule set* Σ *is called* bounded treewidth set (bts) *if for every database D,* $\mathcal{I}(D \cup \Sigma)$ *has bounded treewidth.*

Recognizing whether a rule set has this property is undecidable [7], yet a plethora of criteria ensuring this property have been identified. To start with, the condition is trivially fulfilled if $\mathcal{I}(D \cup \Sigma)$ is even guaranteed to be finite (in other words: if the chase is known to terminate). Pure Datalog (also referred to as *full implicational dependencies* [20] or *total TGDs* [11]) is an immediate case, as no new domain elements are created at all. Weakly acyclic TGDs [26, 27] 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 [8].

Cases where $\mathcal{I}(D \cup \Sigma)$ may be infinite but treewidth-bounded are also manifold: 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* [12] 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* [10] and *glut-guarded TGDs* [30].

As mentioned before, the question $D, \Sigma \models Q$ is decidable if Σ is bts and Q is a conjunctive query. We extend this result to POMSOQs.

THEOREM 17. *Let D be a database and let* Σ *be a set of rules for which the treewidth of* $\mathcal{I}(D \cup \Sigma)$ *is bounded. Let* $\mathfrak{Q}[\mathbf{x}]$ *be a POMSOQ. Then*

1. $D \cup \Sigma \models \mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ *if and only if* $D \cup \Sigma \cup \{\neg\mathfrak{Q}[\mathbf{c}/\mathbf{x}]\}$ *has no countable model with bounded treewidth, and*
2. *the problem* $D \cup \Sigma \models \mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ *is decidable.*

PROOF. We start by proving the first claim. For the "only if" direction, it is obvious that $D \cup \Sigma \models \mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ implies that $D \cup \Sigma \cup \{\neg\mathfrak{Q}[\mathbf{c}/\mathbf{x}]\}$ can have no model (and therefore no model with bounded treewidth).

For proving the "if" direction, we assume that $D \cup \Sigma \cup \{\neg\mathfrak{Q}[\mathbf{c}/\mathbf{x}]\}$ has no countable model with bounded treewidth. Then it must be unsatisfiable for the following reason: toward a contradiction assume it has a model $\mathcal{I}$. Since $\mathcal{I} \models D \cup \Sigma$, there must be a homomorphism from $\mathcal{I}(D \cup \Sigma)$ to $\mathcal{I}$. Thus, by contraposition of the closedness of models of $\mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ under homomorphisms (Theorem 1), $\mathcal{I}(D \cup \Sigma)$ itself satisfies $\neg\mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ and is therefore a model of $D \cup \Sigma \cup \{\neg\mathfrak{Q}[\mathbf{c}/\mathbf{x}]\}$. Moreover, since Σ is bts, we obtain that $\mathcal{I}(D \cup \Sigma)$ has bounded treewidth. Obviously it is also countable, which yields the desired contradiction and ensures unsatisfiability of $D \cup \Sigma \cup \{\neg\mathfrak{Q}[\mathbf{c}/\mathbf{x}]\}$.

For the second claim we start from the previous claim and note that, since D and Σ are first-order theories and $\neg\mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ is an MSO formula, $D \cup \Sigma \cup \{\neg\mathfrak{Q}[\mathbf{c}/\mathbf{x}]\}$ is an MSO theory. Thus, we we can invoke Theorem 6 to obtain the desired result. □

This result shows that POMSOQ can be used for querying in the presence of bts TGDs. We now exploit this compatibility further. In the preceding sections, we have identified a variety of criteria that guarantee POMSOQ rewritability and hence decidability of the query entailment problem. However, there are many cases where only part of the given TGDs is rewritable. Query answering can still be decided, if the TGDs that are not rewritable are layered "below" the rewritable part. This idea is formalized by the following notion of rule dependencies that was first described in [6] and introduced independently in [25]. It can be viewed as a refinement of the relation $\prec$ of Definition 10. Our presentation is similar to [9].

Definition 17 (Baget et al.). *Let* $R_1 = B_1 \to H_1$ *and* $R_2 = B_2 \to H_2$ *be two TGDs. We say that* R_2 depends on R_1 *if there is*

- *a database* D*,*
- *a substitution* θ *of all variables in* B_1 *with terms in* D *such that* $\theta(B_1) \subseteq D$*, and*
- *a substitution* θ' *of all variables in* B_2 *with terms in* $D \cup \theta(H_1)$ *such that* $\theta'(B_2) \subseteq D \cup \theta(H_1)$ *but* $\theta'(B_2) \nsubseteq D$*.*

A (directed) cut *of a set of rules* Σ *is a partition* (Σ_1, Σ_2) *of* Σ *such that no rule in* Σ_1 *depends on a rule in* Σ_2*. It is denoted* $\Sigma_1 \rhd \Sigma_2$*.*

It is not hard to see that the notion of rule dependencies encodes which rule can possibly trigger which other rule. Baget et al. establish a criterion to compute the set of all dependencies of a given rule set and show that this is an NP-complete task. The next theorem intuitive establishes that for a directed cut, rule sets can be exhaustively applied after each other.

Theorem 18 (Baget et al.). *Let* Σ *be a set of rules admitting a cut* $\Sigma_1 \rhd \Sigma_2$*. Then, for a database* D *and a conjunctive query* $Q[\mathbf{x}]$ *we have that* $D \cup \Sigma \models Q[\mathbf{x}/\mathbf{c}]$ *exactly if there is a Boolean conjunctive query* Q' *such that* $D \cup \Sigma_1 \models Q'$ *and* $Q', \Sigma_2 \models Q[\mathbf{x}/\mathbf{c}]$ *hold.*

We now are ready to establish the main result of this section: decidability of conjunctive query entailment for $\Sigma_1 \rhd \Sigma_2$ rule sets where Σ_2 is POMSOQ rewritable and Σ_1 is bts.

Theorem 19. *Let* D *be a database and let* Σ *be a set of rules for which the treewidth of* $\mathcal{I}(D \cup \Sigma)$ *is bounded. Let* Σ' *be a rule set with* $\Sigma \rhd \Sigma'$ *and let* $Q[\mathbf{x}]$ *be a Boolean conjunctive query for which a rewriting* $\mathfrak{Q}_{Q,\Sigma'}[\mathbf{x}]$ *exists. Then*

1. $D \cup \Sigma \cup \Sigma' \models Q[\mathbf{c}/\mathbf{x}]$ *if and only if* $D \cup \Sigma \models \mathfrak{Q}_{Q,\Sigma'}[\mathbf{c}/\mathbf{x}]$*, and*
2. *the problem* $D \cup \Sigma \cup \Sigma' \models Q[\mathbf{c}/\mathbf{x}]$ *is decidable.*

Proof. We start by proving the first claim. For the "only if" direction, assume $D \cup \Sigma \cup \Sigma' \models Q[\mathbf{c}/\mathbf{x}]$. We apply Theorem 18 to obtain a Q' with $D \cup \Sigma \models Q'$ and $Q' \cup \Sigma' \models Q[\mathbf{c}/\mathbf{x}]$. By the definition of rewritability, we obtain $Q' \models \mathfrak{Q}_{Q,\Sigma'}[\mathbf{c}/\mathbf{x}]$. Due to $D \cup \Sigma \models Q'$, we obtain $D \cup \Sigma \models \mathfrak{Q}_{Q,\Sigma'}[\mathbf{c}/\mathbf{x}]$.

For proving the "if" direction we assume $D \cup \Sigma \models \mathfrak{Q}_{Q,\Sigma'}[\mathbf{c}/\mathbf{x}]$ and conclude $\mathcal{I}(D \cup \Sigma) \models \mathfrak{Q}_{Q,\Sigma'}[\mathbf{c}/\mathbf{x}]$, which in turn means that there must be an expansion $\varphi[\mathbf{y}]$ of $\mathfrak{Q}_{Q,\Sigma'}[\mathbf{c}/\mathbf{x}]$ such that $\mathcal{I}(D \cup \Sigma) \models \exists \mathbf{y}.\varphi[\mathbf{y}]$, and therefore – due to universality of $\mathcal{I}(D \cup \Sigma)$ and homomorphism-closedness of models of $\exists \mathbf{y}.\varphi[\mathbf{y}]$ – also $D \cup \Sigma \models \exists \mathbf{y}.\varphi[\mathbf{y}]$. Since trivially $\exists \mathbf{y}.\varphi[\mathbf{y}] \models \mathfrak{Q}_{Q,\Sigma'}[\mathbf{c}/\mathbf{x}]$, we can conclude $\exists \mathbf{y}.\varphi[\mathbf{y}] \cup \Sigma' \models Q[\mathbf{c}/\mathbf{x}]$ by Skolemization and the definition of rewriting. This leads to the final conclusion $D \cup \Sigma \cup \Sigma' \models Q[\mathbf{c}/\mathbf{x}]$.

The second claim is a direct consequence from the claim just proven and the second claim of Theorem 17. □

Corollary 20. *For any rule set* $\Sigma \cup \Sigma'$ *where* Σ *is a bounded treewidth set,* Σ' *is POMSOQ-rewritable and* $\Sigma \rhd \Sigma'$*, CQ answering is decidable.*

Roughly speaking, the introduced framework allows us to deal with a specific sort of "mildly recursive" (also non-guarded) Datalog that is layered on top of TGD sets featuring value invention. Note that in general, query entailment for $\Sigma_1 \rhd \Sigma_2$ with bts Σ_1 and Datalog Σ_2 is undecidable, which can be shown along the lines of the proof of undecidability of conjunctive query entailment in $\mathcal{EL}^{++}$ [31]. In order to guarantee decidability, the Datalog part has to exhibit a sort of regularity property which can also be observed in the RBox definition of highly expressive DLs.

11. CONCLUSION

Positive monadic second-order queries represent a well-balanced middle ground between expressivity and computability. They significantly extend the querying capabilities of (unions of) conjunctive queries and conjunctive 2-way regular path queries (at the cost of higher complexities), yet – as opposed to full-fledged Datalog queries – maintain beneficial computational properties such as decidability of query subsumption and of query entailment in the presence of TGDs that have the bounded treewidth model property.

Thereby, the notion of POMSOQ-rewritability significantly extends first-order rewritability – which has been proven useful for theoretical considerations and practical realization of query answering alike – to much larger classes of TGD sets covering features like transitivity which are considered difficult to handle within the known decision frameworks.

Furthermore, POMSOQ-rewritable TGD sets can be smoothly integrated with bounded treewidth TGD sets as long as certain dependency constraints are obeyed. This provides a valuable perspective on rule-based data access as a task that can be solved by combining bottom-up techniques like the *chase* with top-down techniques such as query rewriting.

While these results are already very promising, this work marks only the very first steps in the research on positive monadic second-order queries. Our work immediately raises a number of interesting open questions: How general are POMSOQs? Is there a larger, more expressive fragment which jointly satisfies all the established properties? Is every rewriting for a TGD set and a given CQ that is expressible in MSO logic equivalent to a POMSOQ? What are the precise combined complexities of query answering on databases and for deciding query subsumption? Which more general syntactic criteria ensure POMSOQ-rewritability? Can all fragments of TGDs (including those considered in description logics) for which conjunctive query answering is known to be decidable be captured as a combination of bts and POMSOQ-rewriting? Answering these questions will not only contribute insights to the specific form of POMSOQs considered herein, but also provide a more unified view on query answering under constraints in general.

Acknowledgments. We thank Bruno Courcelle for very helpful comments concerning his work on monadic second-order logic and structures of bounded tree-width. We thank Diego Calvanese for helping to clarify complexity issues of C2RPQs.

12. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley, 1994.

[2] F. Afrati, S. Cosmadakis, and E. Foustoucos. Datalog programs and their persistency numbers. *ACM Transactions on Computational Logic*, 6(3):481–518, 2005.

[3] H. Andréka, I. Németi, and J. van Benthem. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998.

[4] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, second edition, 2007.

[5] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[6] J.-F. Baget. Improving the forward chaining algorithm for conceptual graphs rules. In D. Dubois, C. A. Welty, and M.-A. Williams, editors, *KR*, pages 407–414. AAAI Press, 2004.

[7] J.-F. Baget, M. Leclère, and M.-L. Mugnier. Walking the decidability line for rules with existential variables. In F. Lin, U. Sattler, and M. Truszczynski, editors, *Proceedings of the 12th International Conference on Principles of Knowledge Representation and Reasoning (KR'10)*. AAAI Press, 2010.

[8] J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat. Extending decidable cases for rules with existential variables. In C. Boutilier, editor, *Proceedings of the 21st International Conference on Artificial Intelligence (IJCAI'09)*, pages 677–682. IJCAI, 2009.

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

[10] J.-F. Baget, M.-L. Mugnier, S. Rudolph, and M. Thomazo. Walking the complexity lines for generalized guarded existential rules. In Walsh [36], pages 712–717.

[11] C. Beeri and M. Y. Vardi. The implication problem for data dependencies. In *Proceedings of the 8th Colloquium on Automata, Languages and Programming*, pages 73–85, London, UK, 1981. Springer-Verlag.

[12] A. Calì, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. In G. Brewka and J. Lang, editors, *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR'08)*, pages 70–80. AAAI Press, 2008.

[13] A. Calì, G. Gottlob, and T. Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. In J. Paredaens and J. Su, editors, *Proceedings of the 28th Symposium on Principles of Database Systems (PODS 2009)*, pages 77–86. ACM, 2009.

[14] A. Calì, G. Gottlob, and A. Pieris. Advanced processing for ontological queries. *Proceedings of VLDB*, 3(1):554–565, 2010.

[15] A. Calì, G. Gottlob, and A. Pieris. Query answering under non-guarded rules in Datalog+/-. In P. Hitzler and T. Lukasiewicz, editors, *Proceedings of Web Reasoning and Rule Systems - Fourth International Conference, RR 2010*, volume 6333 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2010.

[16] D. Calvanese. Personal communication, September 2011.

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

[18] D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Y. Vardi. Containment of conjunctive regular path queries with inverse. In *KR*, pages 176–185, 2000.

[19] D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Y. Vardi. Reasoning on regular path queries. *SIGMOD Record*, 32(4):83–92, 2003.

[20] A. K. Chandra, H. R. Lewis, and J. A. Makowsky. Embedded implicational dependencies and their inference problem. In *Conference Proceedings of the 13th Annual ACM Symposium on Theory of Computation (STOC'81)*, pages 342–354. ACM, 1981.

[21] B. Courcelle. The monadic second-order logic of graphs, ii: Infinite graphs of bounded width. *Mathematical Systems Theory*, 21(4):187–221, 1989.

[22] B. Courcelle. Personal communication, August 2011.

[23] B. Courcelle and J. Engelfriet. Graph structure and monadic second-order logic, a language theoretic approach. manuscript, to be published at Cambridge University Press; available at http://www.labri.fr/perso/courcell/Book/TheBook.pdf, April 2011.

[24] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.

[25] A. Deutsch, A. Nash, and J. Remmel. The chase revisited. In *Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '08, pages 149–158, New York, NY, USA, 2008. ACM.

[26] A. Deutsch and V. Tannen. Reformulation of XML queries and constraints. In D. Calvanese, M. Lenzerini, and R. Motwani, editors, *Proceedings of the 9th International Conference on Database Theory (ICDT 2003)*, volume 2572 of *Lecture Notes in Computer Science*, pages 225–241. Springer, 2003.

[27] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, 2005.

[28] D. Florescu, A. Levy, and D. Suciu. Query containment for conjunctive queries with regular expressions. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, PODS '98, pages 139–148, New York, NY, USA, 1998. ACM.

[29] Y. Kazakov. An extension of complex role inclusion axioms in the description logic $\mathcal{SROIQ}$. In *Proceedings of the 5th International Joint Conference on Automated Reasoning (IJCAR 2010)*, LNCS. Springer, 2010.

[30] M. Krötzsch and S. Rudolph. Extending decidable existential rules by joining acyclicity and guardedness. In Walsh [36], pages 963–968.

[31] M. Krötzsch, S. Rudolph, and P. Hitzler. Conjunctive queries for a tractable fragment of OWL 1.1. In K. Aberer, K.-S. Choi, N. Noy, D. Allemang, K.-I. Lee, L. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber, and P. Cudré-Mauroux, editors, *Proceedings of the 6th International Semantic Web Conference (ISWC'07)*, volume 4825 of *LNCS*, pages 310–323. Springer, 2007.

[32] M. Ortiz, S. Rudolph, and M. Simkus. Query answering in the horn fragments of the description logics shoiq and sroiq. In Walsh [36], pages 1039–1044.

[33] O. Shmueli. Equivalence of datalog queries is undecidable. *J. Log. Program.*, 15(3):231–241, 1993.

[34] L. J. Stockmeyer. *The Complexity of Decision Problems in Automata Theory and Logic*. PhD thesis, Massachusetts Institute of Technology, 1974.

[35] M. Y. Vardi. The complexity of relational query languages. In H. R. Lewis, B. B. Simons, W. A. Burkhard, and L. H. Landweber, editors, *STOC*, pages 137–146. ACM, 1982.

[36] T. Walsh, editor. *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*. IJCAI/AAAI, 2011.

APPENDIX

This appendix provides additional proofs that have been omitted in the main paper for reasons of space.

Theorem 1. *For a POMSOQ $\mathfrak{Q}[\mathbf{x}]$, the set of models of $\exists\mathbf{x}.\mathfrak{Q}$ is closed under homomorphisms.*

Proof. The result is an immediate consequence of the following claim: Consider a homomorphism $\mu : \mathcal{I} \to \mathcal{J}$ between interpretations $\mathcal{I}, \mathcal{J}$. Then for all variable assignments $\mathcal{Z} : \mathbf{x} \to \Delta^{\mathcal{I}}$ we find that $\mathcal{I}, \mathcal{Z} \models \mathfrak{Q}$ implies $\mathcal{J}, \mu \circ \mathcal{Z} \models \mathfrak{Q}$.

We prove the claim by induction on the degree d of $\mathfrak{Q}$. For $d = 0$ the correspondence follows by the definition of homomorphisms. For $d > 0$, we first note that $\mathfrak{Q}$ can be transformed into an equivalent formula of shape $\forall \mathtt{U}_1, \ldots, \mathtt{U}_m.\exists v_1, \ldots, v_k.\varphi$ where φ is a positive boolean formula having as arguments POMSOQs of lower degree and monadic literals of the form $\mathtt{U}_i(x)$ and $\neg\mathtt{U}_i(x)$. Now suppose for the indirect proof $\mathcal{I}, \mathcal{Z} \models \mathfrak{Q}$ but $\mathcal{J}, \mu \circ \mathcal{Z} \not\models \mathfrak{Q}$ then there must be an assignment $\mathfrak{Z}$ of $\mathtt{U}_1, \ldots, \mathtt{U}_m$ to subsets of $\Delta^{\mathcal{J}}$ such that for all assignments of $v_1, \ldots, v_k$ to elements from $\Delta^{\mathcal{J}}$ the formula φ is not satisfied. We now define an assignment $\mathfrak{Z}' : \{\mathtt{U}_1, \ldots, \mathtt{U}_m\} \to 2^{\Delta^{\mathcal{I}}}$ by $\mathfrak{Z}'(\mathtt{U}_i) \coloneqq \{\delta \mid \mu(\delta) \in \mathfrak{Z}(\mathtt{U}_i)\}$. Since $\mathcal{I}, \mathcal{Z}$ satisfies $\forall \mathtt{U}_1, \ldots, \mathtt{U}_m \exists v_1, \ldots, v_k \varphi$ there must be a variable assignment $\mathcal{Z}_v : \{v_1, \ldots, v_k\} \to \Delta^{\mathcal{I}}$ such that $\mathcal{I}, \mathcal{Z}, \mathfrak{Z}', \mathcal{Z}_v \models \varphi$. Yet, for any literal or POMSOQ λ from φ we obtain that $\mathcal{I}, \mathcal{Z}, \mathfrak{Z}', \mathcal{Z}_v \models \lambda$ implies $\mathcal{J}, \mu \circ \mathcal{Z}, \mathfrak{Z}, \mu \circ \mathcal{Z}_v \models \lambda$. For POMSOQs, this follows from the induction hypothesis, whereas for monadic literals of the form $\mathtt{U}_i(x)$ and $\neg\mathtt{U}_i(x)$ it follows by construction of $\mathfrak{Z}'$. Then, since φ is a positive combination of these literals, we obtain $\mathcal{J}, \mu\circ\mathcal{Z}, \mathfrak{Z}', \mu\circ\mathcal{Z}_v \models \varphi$ which contradicts the above assumption that there is no such $\mathcal{Z}_v$. Hence the original claim is proven. □

Theorem 4. *For every POMSOQ $\mathfrak{Q}$, the set $\Sigma(\mathfrak{Q})$ can be constructed in linear time and both expressions are equivalent in the sense that $\models \forall\mathbf{x}.\mathfrak{Q}[x] \leftrightarrow (\Sigma(\mathfrak{Q}) \to p_{\mathfrak{Q}}(\mathbf{x}))$ is a tautology.*

Proof. The claimed linear time bound is immediate from the definition. The claimed equivalence is obvious for queries of degree 0. For degrees ≥ 1, it is shown by induction on the degree of $\mathfrak{Q}$. We first show the base case of degree 1. Let $\mathfrak{Q}[\mathbf{x}] = \forall \mathtt{U}_1, \ldots, \mathtt{U}_m \neg\forall\mathbf{y}. \bigwedge_{R\in\mathcal{R}} R$.

First, consider an interpretation $\mathcal{I}$ and variable assignment $\mathcal{Z}$ such that $\mathcal{I}, \mathcal{Z} \models \mathfrak{Q}[\mathbf{x}]$. $\mathcal{R}$ is of the form $\mathcal{R}_+ \cup \mathcal{R}_-$, where the rules in $\mathcal{R}_+$ have heads of the form $U_i(t)$ and rules in $\mathcal{R}_+$ have empty heads. Thus, $\mathcal{I}, \mathcal{Z} \models \forall \mathtt{U}_1, \ldots, \mathtt{U}_m(\forall\mathbf{y}. \bigwedge_{R\in\mathcal{R}_+} R \to \neg\forall\mathbf{y}. \bigwedge_{R\in\mathcal{R}_-} R)$.

Consider a domain element $\delta \in \Delta^{\mathcal{I}}$ and index $i \in \{1, \ldots, m\}$ such that $\mathcal{I}, \mathcal{Z}\{z \mapsto \delta\} \models \forall \mathtt{U}_1, \ldots, \mathtt{U}_m.\forall\mathbf{y}. \bigwedge_{R\in\mathcal{R}_+} R \to U_i(z)$. We show that $\mathcal{I}, \mathcal{Z}\{z \mapsto \delta\} \models \Sigma(\mathfrak{Q}) \to \hat{U}_i(z, \mathbf{x})$ $(*)$. Due to the rule structure of $\mathcal{R}_+$, this can be shown by induction over the "derivation" of "$U_i(\delta)$".

The assumption implies that there is a rule $R = \varphi \to U_i(y) \in \mathcal{R}_+$ (with y in $\mathbf{y}$) and a variable assignment $\mathcal{Z}'$ that agrees with $\mathcal{Z}\{z \mapsto \delta\}$ on all variables other than possibly $\mathbf{y}$ such that $\mathcal{Z}'(y) = \delta$ and $\mathcal{I}, \mathcal{Z}' \models \varphi$. There is a rule $\forall\mathbf{x},\mathbf{y}.\varphi' \to \hat{U}_i(y, \mathbf{x})' \in \Sigma(\mathfrak{Q})$ that is obtained from R. It is easy to see that $\mathcal{I}, \mathcal{Z}' \models \varphi'[\mathbf{x}, \mathbf{y}]$. This is immediate for atoms that occur in both φ and φ'. For atoms of the form $\hat{U}_j(y', \mathbf{x})$, it follows by induction. No other atoms can occur in queries of depth 1. Thus, since $\mathcal{I} \models \forall\mathbf{x},\mathbf{y}.\varphi' \to \hat{U}_i(y, \mathbf{x})$, we conclude $\mathcal{I}, \mathcal{Z}' \models \hat{U}_i(y, \mathbf{x})$ and (since $\mathcal{Z}'(y) = \delta = \mathcal{Z}(z)$), we obtain $\mathcal{I}, \mathcal{Z} \models \hat{U}_i(z, \mathbf{x})$. This shows $(*)$.

Consider a rule $R = \varphi \to\ \in \mathcal{R}_-$ and $\mathcal{Z}'$ that agrees with $\mathcal{Z}$ on all variables other than possibly $\mathbf{y}$ such that $\mathcal{I}, \mathcal{Z}' \models \varphi$. Let $\varphi' \to p_{\mathfrak{Q}}(\mathbf{x}) \in \Sigma(\mathfrak{Q})$ be the rule obtained from R. Using $(*)$ on all atoms of form $U_j(y)$ in φ, we find that $\mathcal{I}, \mathcal{Z}' \models \varphi'$ and thus $\mathcal{I}, \mathcal{Z}' \models p_{\mathfrak{Q}}(\mathbf{x})$ as required.

This shows one direction of the claim for queries of degree 1. The other direction is similar and we omit the details.

The result for degrees $d > 1$ follows by an easy induction on d where $d = 1$ is the base case. Indeed, the treatment of subqueries can be viewed as the replacement of a subquery with an equivalent formula, which leads to an equivalent formula by the Replacement Theorem of first-order logic. □

Theorem 5. *Given a database D, a POMSOQ $\mathfrak{Q}[\mathbf{x}]$, and a list of constants $\mathbf{c}$, checking $D \models \mathfrak{Q}[\mathbf{c}/\mathbf{x}]$ is in PSpace w.r.t. the combined size of D and $\mathfrak{Q}[\mathbf{c}/\mathbf{x}]$. Moreover, it is PTime-complete w.r.t. the size of D.*

Proof. PSpace membership for combined complexity is a direct consequence of PSpace completeness of model checking in monadic second-order logic [34, 35], keeping in mind that due to Corollary 2, entailment coincides with model checking if the premise is a set of ground facts. For PTime membership, according to Theorem 4, the question can be decided by using $\Sigma(\mathfrak{Q})$. The result therefore follows from the known respective complexity result for Datalog [24]. For PTime hardness, we reduce entailment in propositional Horn logic to POMSOQ answering. Given a set $\mathcal{H}$ of propositional Horn clauses, we introduce for every propositional atom a occurring therein a constant c_a. We also introduce one additional constant *nil*. Moreover, for every Horn clause $C \in \mathcal{H}$ with $C = a_1 \wedge \ldots \wedge a_n \to a$, we introduce constants $b_{C,1}, \ldots, b_{C,n}$ and ground atoms $\mathit{entails}(b_{C,1}, c_a)$, $\mathit{first}(b_{C,i}, c_{a_i})$ for all $i \in \{1, \ldots, n\}$, $\mathit{rest}(b_{C,n}, \mathit{nil})$, and $\mathit{rest}(b_{C,i}, b_{C,i+1})$ for all $i \in \{1, \ldots, n-1\}$. Then the a propositional atom is entailed by $\mathcal{H}$ exactly if it is an answer for the POMSOQ $\mathfrak{Q}_{true}[x] = \forall\mathtt{U}\neg\forall y, z, z' \bigwedge_{R\in\mathcal{R}} R$ with $\mathcal{R}$ consisting of the rules

$$\begin{array}{rcl} & \to & \mathtt{U}(\mathit{nil}) \\ \mathit{first}(y,z) \wedge \mathtt{U}(z) \wedge \mathit{rest}(y,z') \wedge \mathtt{U}(z') & \to & \mathtt{U}(y) \\ \mathtt{U}(y) \wedge \mathit{entails}(y,z) & \to & \mathtt{U}(z) \\ \mathtt{U}(x) & \to & . \end{array}$$

□

Theorem 13. *For every POMSOQ $\mathfrak{Q}$, the rule set $\Sigma(\mathfrak{Q})$ of Definition 4 is fully oriented. Moreover, for every POMSOQ-rewritable set Σ of TGDs, there is a fully oriented set of Datalog rules Σ' such that:*

- *every predicate p in Σ has a corresponding head predicate q_p in Σ' that does not occur in Σ,*
- *for every database D and conjunctive query $Q[\mathbf{x}]$ that do not contain predicates of the form q_p, and for every list of constants $\mathbf{c}$, we have $D \cup \Sigma \models Q[\mathbf{c}/\mathbf{x}]$ iff $D \cup \Sigma' \models Q'[\mathbf{c}/\mathbf{x}]$ where Q' is obtained from Q by replacing all predicates p with q_p.*

Proof. The first part of the claim follows by induction over the degree d of $\mathfrak{Q}$. The case of $d = 0$ is clear. The case of $d = 1$ follows by observing that the rules with head predicate $p_{\mathfrak{Q}}$ in $\Sigma(\mathfrak{Q})$ cannot be $\prec$-smaller than any other rule, and thus form a maximal j-oriented (for any position j in $p_{\mathfrak{Q}}$) subset of $\Sigma(\mathfrak{Q})$. Likewise, the set of rules with head of the form $\hat{U}_i(y, \mathbf{z})$ is clearly 1-oriented. To show the claim for $d > 1$, we observe that no rule in $\Sigma(\mathfrak{Q})$ is $\prec$-smaller than any rule in $\Sigma(\mathfrak{Q}')$ for some subquery $\mathfrak{Q}'$ of $\mathfrak{Q}$. The claim follows by induction.

For the second part of the claim, let $Q_p[\mathbf{y}]$ be the CQ $p(\mathbf{y})$ for each predicate p that is not of the form $q_{p'}$. There is a rewriting $\mathfrak{Q}_p[\mathbf{y}]$ for $Q_p[\mathbf{y}]$ and Σ. By Lemma 8, $\Sigma \models \forall\mathbf{y}.\mathfrak{Q}_p[\mathbf{y}] \leftrightarrow Q_p[\mathbf{y}]$. By Theorem 4, $\models \forall\mathbf{y}.\mathfrak{Q}_p[\mathbf{y}] \leftrightarrow (\Sigma(\mathfrak{Q}_p) \to p_{\Sigma(\mathfrak{Q}_p)}(\mathbf{y}))$. Thus, $\Sigma \models \forall\mathbf{y}.Q_p[\mathbf{y}] \leftrightarrow (\Sigma(\mathfrak{Q}_p) \to p_{\Sigma(\mathfrak{Q}_p)}(\mathbf{y}))$.

Using arguments as in the proof of Theorem 10, we find that $D \models \forall\mathbf{y}.(\Sigma \to Q_p[\mathbf{y}]) \leftrightarrow (\Sigma(\mathfrak{Q}_p) \to p_{\Sigma(\mathfrak{Q}_p)}(\mathbf{y}))$ whenever $D \cup \Sigma$ is consistent. To cover the case that $D \cup \Sigma$ is inconsistent, let $\mathfrak{Q}_\perp$ be a POMSOQ without free variables such that, for all databases D', $D' \cup \Sigma$ is inconsistent iff $D' \models \mathfrak{Q}_\perp$. To find such a $\mathfrak{Q}_\perp$, consider $\mathfrak{Q}_p$ for a predicate p that does not occur in Σ and delete from $\Sigma(\mathfrak{Q}_p)$ all rules that use the predicate p (these rules check for occurrences of p in the input database). $\mathfrak{Q}_\perp$ can easily be obtained from this.

By the first part of the claim, $\Sigma(\mathfrak{Q}_p)$ and $\Sigma(\mathfrak{Q}_\perp)$ are fully directed. We set $q_p := p_{\Sigma(\mathfrak{Q}_p)}$. Let $\Sigma_\perp$ be the fully directed rule set obtained from $\Sigma(\mathfrak{Q}_\perp)$ by replacing each rule $\forall \mathbf{y}.\varphi \to$ that has an empty head by new rules $\forall \mathbf{x}, \mathbf{y}.\varphi \to q_p(\mathbf{x})$ for each of the predicates q_p where $\mathbf{x}$ is a list of fresh variables of the appropriate length. Thus, $\Sigma_\perp$ entails all possible facts over predicates q_p from D whenever $D \cup \Sigma$ is inconsistent.

Now we can set $\Sigma' := \Sigma_\perp \cup \bigcup_p \Sigma(\mathfrak{Q}_p)$ where we assume w.l.o.g. that any two $\Sigma(\mathfrak{Q}_p)$ and $\Sigma(\mathfrak{Q}_{p'})$ use mutually disjoint sets of head predicates. It is easy to verify that the claim is satisfied for this choice. □

Theorem 15. *The stratification $\Sigma \uparrow \pi$ is correct whenever the following hold:*

- *Σ is pre-stratifiable,*
- *for all $i > j$, Σ is $\langle \pi(i), \pi(j) \rangle$-invertible.*

Proof. The result is shown by transforming expansion trees of Σ into according expansion trees for $\Sigma \uparrow \pi$ with root and leaf labels replaced as in Definition 12. Clearly, this suffices to establish the result for arbitrary conjunctive queries. Thus consider an arbitrary expansion tree T for Σ and $p(\mathbf{x})$.

First, we construct an expansion tree T_1 for $\Sigma\langle \pi(n) \rangle \cup \Sigma\langle \pi(n) \to 0 \rangle$ and $p_{\pi(n)}(\mathbf{x})$. This can be done inductively following the construction of T. We begin with a root node labeled $p_{\pi(n)}(\mathbf{x})$. For each application of a rule $\rho \in \Sigma$ in the construction of T, we apply the corresponding rule in $\Sigma\langle \pi(n) \rangle$, followed by applications of rules of the form $q_{\pi(n)}(\mathbf{y}) \to q_0(\mathbf{y})$ to each new leaf node with label $q_0(\mathbf{s})$, yielding new leaf nodes $q_{\pi(n)}(\mathbf{s})$. The latter can be further expanded following the construction of T.

Second, we construct from T_1 an expansion tree T_2 for the set Σ_{ps} as in Definition 14. The set of leaf node labels of is the same as that of T_1, but possibly with atoms of the form $q_i(\mathbf{s})$ replaced by some other atom $q_j(\mathbf{s})$ with $i, j \in \{0, \ldots, n\}$, $i \neq j$. We say that a *critical node* is a node c with $\mathsf{rule}(c)$ of the form $q_k(\mathbf{y}) \to q_0(\mathbf{y})$. We iteratively eliminate critical nodes from T_1.

In a first stage, we remove all critical nodes that have no complex rule applications above them in the tree. Let c be such a node of minimal distance to the root. Let $q_0(\mathbf{t})$ be the label of c. Then the child $\check{c}$ of c is labeled $q_{\pi(n)}(\mathbf{t})$. Since the root is of form $p_{\pi(n)}(\mathbf{t})$ with $\pi(n) \neq 0$, there must be a node d above c. We transform T_1 recursively.

Base case: If the label of d is of form $r_{\pi(n)}(\mathbf{t})$, then $r = q$ and $\mathsf{rule}(d)$ has the form $q_0(\mathbf{y}) \to q_{\pi(n)}(\mathbf{y})$. Then T_1 is transformed by removing node c and identifying node d and $\check{c}$.

Recursion: If the label of d is of form $r_0(\mathbf{t})$, then $\mathsf{rule}(d) \in \Sigma\langle 0 \rangle$ has a single body atom $q_0(\mathbf{s})$ and head $r_0(\mathbf{s})$. This is so since a non-complex rule in a basic Datalog rule set can only produce invariant expansion edges. Thus, there is a rule $\rho_{\pi(n)} = q_{\pi(n)}(\mathbf{s}) \to r_{\pi(n)}(\mathbf{s}) \in \Sigma\langle \pi(n) \rangle$. Replace c with a new node c' with label $r_{\pi(n)}(\mathbf{t})$, and replace d with a new node d' with label $r_0(\mathbf{t})$. A correct expansion tree is obtained with $\mathsf{rule}(c') = \rho_{\pi(n)}$ and $\mathsf{rule}(d') = r_{\pi(n)}(\mathbf{y}) \to r_0(\mathbf{y})$. After the recursion, d' is critical and has a smaller distance to the root than c. Hence, the transformation eventually terminates.

We thus obtain a expansion tree where every critical node occurs below a rule application that is complex. In a second stage, we remove the remaining critical nodes iteratively. The construction temporarily introduces rule applications of the form $q_i(\mathbf{y}) \to q_0(\mathbf{y})$ for arbitrary $i \in \{1, \ldots, n\}$ which are also taken into account in the elimination steps.

In each step, select a critical node c of minimal distance to the root. Let $\check{c}$ be the child of c, and let $q_0(\mathbf{t})$ be the label of c. Then the child $\check{c}$ of c is labeled $q_k(\mathbf{t})$. At first c is "pushed downwards" by exhaustively applying the following transformation rules. Do the following as long as $\check{c}$ has a child node and $\mathsf{rule}(\check{c})$ is not complex:

(A) If $\mathsf{rule}(\check{c}) \in \Sigma\langle 0 \to k \rangle$, delete $\check{c}$ and identify c with the (unique) child node of $\check{c}$ – it already has the same label.

(B) If $\mathsf{rule}(\check{c}) \in \Sigma\langle i \to k \rangle$ for $i \geq 1$, delete $\check{c}$. The (unique) child of $\check{c}$ can be obtained from c using rule $q_i(\mathbf{y}) \to q_0(\mathbf{y})$.

(C) Otherwise, if $\mathsf{rule}(\check{c})$ is not complex, replace c with a new node c' with label $q_0(\mathbf{t})$ and $\mathsf{rule}(c') \in \Sigma\langle 0 \rangle$ being the stratum 0 rule that corresponds to $\mathsf{rule}(\check{c})$. The former child $r_k(\mathbf{y})$ of $\check{c}$ becomes a child of c' where rule $r_k(\mathbf{y}) \to r_0(\mathbf{y})$ is applied.

It is clear that this procedure terminates since the number of critical nodes is either reduced or stays the same while critical nodes move down in the tree.

After having pushed c down, we distinguish two cases:

Case 1: $\check{c}$ is a leaf node. Then $\check{c}$ is deleted from the tree. This corresponds to a replacement of a leaf label $q_k(\mathbf{t})$ with $q_0(\mathbf{t})$, which is allowed in the construction.

Case 2: $\check{c}$ has a child d such that $\check{c} \to d$ is complex. By assumption, there is a complex rule is applied above c. Let e be the lowest node above c that was expanded with a complex rule. Then the path from e to d induces a subtree S that satisfies Definition 13 (1), (2). If S fails to satisfy (3), then two nodes in the derivation chain have the same label. An essential expansion tree S' can be obtained from S by identifying pairs of nodes that have the same label.

By Definition 14, S' has a pre-stratification S_p. We replace S' by S_p where we attach sub expansion trees to leafs of S_p as follows. Consider a leaf l of S_p that is labeled with λ. If λ is a leaf of S', then connect the sub expansion tree rooted at λ to l. Otherwise, $\lambda = r_0(\mathbf{s})$ and S' has a bottom leaf labeled $r_i(\mathbf{s})$, $i \leq 1$. Apply the rule $r_i(\mathbf{y}) \to r_0(\mathbf{y})$ to l, and attach the new child with label $r_i(\mathbf{s})$ to the sub expansion tree rooted in the original bottom leaf with label $r_i(\mathbf{s})$.

The repeated application of these reductions terminates. Indeed, Case 2 may introduce new sub expansion trees with critical root nodes. Since they are constructed from bottom leafs, each such tree is a subtree of the tree S that has been replaced, and is thus of strictly smaller size. Termination is shown be considering the multiset of the sizes of all sub expansion trees with critical root node, ordered under the standard multiset order [5]. Thus, we obtain the claimed expansion tree T_2.

Third, we transform T_2 into an expansion tree T_3 for $\Sigma \uparrow \pi$. The transformation proceeds in various iterative steps.

Suppose that T_2 contains two nodes c and d that have complex rules, c is above d and $s(c) < s(d)$. We can choose c and d such that all nodes e between c and d are expanded by non-complex rules. Indeed, if some such edge e would be expanded with a complex rule, then either $s(e) > s(c)$ or $s(e) < s(d)$, so that it could be used instead of one of the chosen nodes. The pair of nodes c and d is called a *critical pair*.

We proceed by finding and replacing certain $\langle i, j \rangle$-trees in T_2 with their inverted form. Select a critical pair of nodes c and d. The path between c and d induces an expansion tree S that satisfies Definition 13 (1), (2). Let S' be the expansion tree constructed from S by pushing down (and merging) all applications of rules $q_i(\mathbf{y}) \to r_i(\mathbf{y})$, as done for critical nodes in the second step of this transformation above. Then S' only uses rules of stratum $s(c)$, followed by an eventual use of a rule of form $r_{s(d)}(\mathbf{y}) \to r_{s(c)}(\mathbf{y})$ and the final application of $\mathsf{rule}(c)$. An essential expansion tree S'' is obtained from S' by identifying nodes with the same label until condition (3) of Definition 15 is met. Then S'' is an essential $\langle s(d), s(c)\rangle$-tree.

We thus can invert S'' to find an $\langle s(c), s(d)\rangle$-tree S_i as in Definition 15. We now replace the original expansion tree S using S_i as follows. The root of S has a label $q_{s(c)}(\mathbf{s})$ while the root of S_i has a label $q_{s(d)}(\mathbf{s})$. A new application of rule $q_{s(d)}(\mathbf{y}) \to q_{s(c)}(\mathbf{y})$ is used to attach S_i to T_2. Likewise, for every leaf l of S_i labeled by an atom of the form $q_i(\mathbf{s})$, there is a leaf of S of the form $q_j(\mathbf{s})$. Hence, T_2 contains an expansion tree for $q_j(\mathbf{s})$ that can be attached to l through an application of $q_j(\mathbf{y}) \to q_i(\mathbf{y})$. By the condition on multiplicities of leafs in Definition 15, this is possible in such a way that no sub expansion tree of T_2 is attached to more than one leaf node of S_i.

To show termination, we use a multiset order based on paths, defined as follows. A node n of an expansion tree is *complex* if $\mathsf{rule}(n)$ is. If n is complex, then $\mathsf{rule}(n) \in \Sigma\langle\pi(k)\rangle$ for a unique k; we write $s(n)$ to denote this k. Now every path $\eta = n_0 \to \ldots \to n_\ell$ is associated with the sequence $s(\eta) = s(m_0), \ldots, s(m_h)$ of strata of the complex nodes in η (in order). Such sequences of natural numbers are ordered lexicographically based on the inverse order of numbers (i.e. smaller sequences have higher strata first). Now with every expansion tree S considered herein, we associate a multiset $s(S)$ of the sequences $s(\eta)$ for all paths η of S. Such multisets are ordered by the standard multiset extension of the order on sequences $s(\eta)$ [5].

Every replacement of a sub expansion tree S as above leads to a decrease of $s(T_2)$. Indeed, using notation as above, only paths that included node c are affected. According to Definition 13, the root node m of S_i is complex and by Definition 15, it must be of stratum $s(m) = s(d)$. Let ℓ be the number of complex nodes above c (before the replacement) viz. above m (after the replacement). Then the set of paths through node c is replaced by a set of new paths through node m. The associated sequences of strata for these paths agree up to position ℓ. Thereafter, the paths before the replacement have $s(c)$ while the paths after the replacement have $s(d)$. Since $s(c) < s(d)$, the paths in the replacement are all strictly smaller than the original ones, i.e., we found a sub-multiset of elements in the multiset $s(T_2)$ that was replaced by a (possibly larger) sub-multiset of strictly smaller elements. Hence the value of $s(T_2)$ has decreased.

To show termination, we need to show that there is no infinite decreasing sequence of multisets $s(T_2)$. This is not true in general: there are infinite decreasing sequences of sequences of strata in their lexicographic order. However, in our case, the length of these sequences is bounded and hence the set of possible sequences is finite, which suffices for termination. To see that the length of sequences is bounded, observe that the multiset of leaf nodes of T_2 cannot grow in a replacement step. This is an easy consequence of the respective condition in Definition 15. Therefore, the total number of complex nodes that can be obtained in the transformation is bounded since every new complex node has at least two children where the total number of leafs is bounded. This shows the claimed finite bound for sequences of strata.

To conclude, we have shown that the iterative replacement of critical edge pairs in T_2 terminates. Thereafter, T_2 can still contain applications of rules $q_i(\mathbf{y}) \to r_j(\mathbf{y}) \notin \Sigma \uparrow \pi$, but not between complex nodes. Accordingly, they can be eliminated by pushing them upwards or downwards as illustrated for critical nodes above.

Thus we obtain an expansion tree T_3 using rules as in $\Sigma \uparrow \pi$. T_3 can easily be turned into an expansion tree as required by Definition 12 by extension every leaf node of the form $q_i(\mathbf{s})$ by applying rules $q_{\mathsf{db}}(\mathbf{x}) \to q_0(\mathbf{x})$ and $q_0(\mathbf{x}) \to q_i(\mathbf{x})$. □