Tomasz Gogacz

Uniwersytet Wrocławski

Converging to the Chase

Praca semestralna nr 1

(semestr letni 2011/12)

Opiekun pracy: Jerzy Marcinkowski

Tomasz Gogacz

Uniwersytet Wrocławski

Converging to the Chase

Praca semestralna 1.
semestr letni (2011/2012)

opiekun pracy: Jerzy Marcinkowski

Converging to the Chase – a Tool for Finite Controllability

Tomasz Gogacz, Jerzy Marcinkowski

Institute of Computer Science,
University Of Wrocław

Abstract. We solve a problem, stated in [CGP10], showing that Sticky Datalog$^\exists$, defined in the cited paper as an element of the Datalog$^\pm$ project, has the finite controllability property. In order to do that, we develop a technique, which we believe can have further applications, of approximating $Chase(D, \mathcal{T})$, for a database instance D and some sets of tuple generating dependencies $\mathcal{T}$, by an infinite sequence of finite structures, all of them being models of $\mathcal{T}$.

1 Introduction

Tuple generating dependencies (TGDs), also known as Datalog$^\exists$ rules, are studied in various areas, from database theory to description logics and in various contexts. The context we are here interested in is computing certain answers to queries in the situation when some semantical information about the database is known (in the form of database dependencies, TGDs), but the knowledge of the database facts is limited.

Let us remind the reader that a TGD is a formula of the form $\forall \bar{x}\ (\varPhi(\bar{x}) \Rightarrow \exists y\ Q(y, \bar{y}))$ where $\varPhi$ is a conjunctive query (CQ), Q is a relation symbol, $\bar{x}, \bar{y}$ are tuples of variables and $\bar{y} \subseteq \bar{x}$. The universal quantifier in front of the formula is usually omitted.

For a set $\mathcal{T}$ of TGDs and a database instance D by $Chase(\mathcal{T}, D)$ we denote the least fixpoint of the chase operation: if the body (the left hand side) of some TGD is satisfied in the current database, and the head (the right hand side) is not satisfied, then add to the database a new constant being the free witness for existential formula in the head. By $Chase^i(\mathcal{T}, D)$ we mean the structure being the i-th stage of the fixpoint procedure (so that $Chase^0(\mathcal{T}, D) = D$). Clearly, $Chase(\mathcal{T}, D) \models D, \mathcal{T}$, but there is no reason to think that $Chase^i(\mathcal{T}, D) \models \mathcal{T}$ for any $i \in \mathbb{N}$.

Since $Chase(\mathcal{T}, D)$ is a "free structure", it is easy to see that for any query Φ (being a union of conjunctive queries, or UCQ) $D, \mathcal{T} \models \Phi$ (which reads "Φ is certainly true in D, in presence of $\mathcal{T}$"), if and only if $Chase \models \Phi^1$.

It is easy to see that query answering in presence of TGDs is undecidable. As usually in such situations many sorts of syntactic restrictions on the dependencies are considered, which imply decidability keeping as much expressive power as possible. Recent new interest in such restricted logics comes from the Datalog$^\pm$ project, led by Georg Gottlob, whose aim is translating important concepts and proof techniques from database theory to description logics and *bridging an apparent gap in expressive power between database query languages and description logics (DLs) as ontology languages, extending the well-known Datalog language in order to embed DLs* [CGT09].

From the point of view of Datalog$^\pm$ and of this paper, the interesting logics are:

Linear Datalog$^\exists$ programs consist of TGDs which, as the body, have a single atomic formula, and this formula is joinless – each variable in the body occurs there only once. The **Joinless Logic** we consider in this paper is a generalization of Linear Datalog$^\exists$, in the sense that we no longer restrict the body of the rule to be a single atom, but we still demand that each variable occurs in the body only once. Let us note that allowing variable repetitions in the heads does not increase the expressive power, as we can remember the equalities as relation names, so we w.l.o.g. assume that such repetitions are not allowed in Joinless Logic.

Guarded Datalog$^\exists$ is an extension of Linear Datalog$^\exists$. A TGD is guarded if it has an atom, in the body, containing all the variables that occur anywhere else in the body. Clearly, Linear Datalog$^\exists$ programs are guarded.

Sticky Datalog$^\exists$. A set $\mathcal{T}$ of TGDs is sticky, if some positions in the relations occurring in the rules can be marked as "immortal" in such a way that the following conditions are satisfied:

- If some variable occurs in an immortal position in the body of a rule form $\mathcal{T}$ then the same variable must occur in immortal position in the head of the same rule.

---

[1] We will write just $Chase$ instead of $Chase(\mathcal{T}, D)$ when the context is clear.

– If some variable occurs more than once in the body of a rule form $\mathcal{T}$ then this variable must occur in immortal position in the head of the same rule.

Apart from decidability, the properties of such logics which are considered desirable are:

Bounded Derivation Depth property (BDD). A set $\mathcal{T}$ of TGDs has the bounded derivation depth property if for each query Ψ (the queries we are interested in are UCQs), there is a constant $k_\Psi \in \mathbb{N}$, such that for each database instance D if $Chase(\mathcal{T}, D) \models \Psi$ then $Chase^{k_\Psi}(\mathcal{T}, D) \models \Psi$. The BDD property turns out to be equivalent to first order rewritability [CGT09]: $\mathcal{T}$ has the BDD property if and only if for each UCQ Ψ there exist a UCQ $\bar{\Psi}$ such that for each database instance D it holds that $Chase(D, \mathcal{T}) \models \Psi$ if and only if $D \models \bar{\Psi}$.

Finite Controllability (FC). A set $\mathcal{T}$ of TGDs has the finite controllability property if for each query Ψ such that $Chase(\mathcal{T}, D) \models \neg\Psi$ there exists a finite structure M such that $M \models \mathcal{T}, D, \neg\Psi$.
A logic is said to have property $P \in \{BDD, FC\}$ if each $\mathcal{T}$ in this logic has it.

It is usually very easy to see whether a logic has the BDD property. And it is usually very hard to see whether it has the FC property.

The query answering problem for Linear Datalog$^\exists$ (or rather for Inclusion Dependencies, which happens to be the same notion as Linear Datalog$^\exists$) was shown to be decidable (and PSPACE-complete) in [JK84]. The problem which was left open in [JK84] was finite controllability – since we mainly consider finite databases, we are not quite happy with the answer that "yes, there exists a database $\bar{D}$, such that $\bar{D} \models \mathcal{T}, D, \neg\Psi$" if all counterexamples $\bar{D}$ for Ψ we can produce are infinite. This problem was solved by Rosati [R06], who proved, by a complicated argument, that IDs (Linear Datalog$^\exists$) have the finite controllability property. His result was improved in [BGO10] where finite controllability is shown for Guarded Datalog$^\exists$.

Sticky Datalog$^\exists$ was introduced in [CGP10], where it was also shown to have the BDD property and where the question of the FC

property of this logic was stated as an open problem. The argument, given in [CGP10], motivating the study of Sticky Datalog$^\exists$ is that it *can express assertions having compositions of roles in the body, which are inherently non-guarded. Sticky sets of TGDs can express constraints and rules involving joins. We are convinced that the overwhelming number of real-life situations involving such constraints can be effectively modeled by sticky sets of TGDs. Of course, since query-answering with TGDs involving joins is undecidable in general, we somehow needed to restrict the interaction of TGDs, when joins are used. But we believe that the restriction imposed by stickiness is a very mild one. Only rather contorted TGDs that seem not to occur too often in real life violate it. For example, each singleton multivalued dependency (MVD) is sticky, as are many realistic sets of MVDs* [CGP10].

1.1 Our contribution

We show two finite controllability results. Probably the more important of them is:

Theorem 1. *Sticky Datalog$^\exists$ has the finite controllability property.*

But this is merely a corollary to a theorem that we consider the main technical achievement of this paper:

Theorem 2. *Joinless Logic has the finite controllability property.*

To prove Theorem 2 we propose a technique, which we think is quite elegant, and relies of two main ideas. One is that we carefully trace the relations (we call them family relations) between elements of Chase which are ever involved in one atom. The second idea is to consider an infinite sequence of equivalence relations, defined by the types of family relations in which the elements (and their ancestors) are involved, and construct an infinite sequence of models as the quotient structures of these equivalence relations. This leads to a sequence of models, that, in a sense, "converges" to the *Chase*.

What concerns the Joinless Logic as such, we prefer not to make exaggerated claims about its importance. We see it just as a mathematical tool – the *Chase* resulting form a Joinless theory is a huge structure, much more complicated than the bounded tree-width

Chase resulting from guarded TGDs, and the ability to control it can give insight into chases generated by logics enjoying better practical motivation – Theorem 1 serves here as a good example. But still Theorem 2 is a very strong generalization of the result of Rosati about Linear Datalog$^\exists$, which itself was viewed as well motivated, while the technique we develop in order to prove it is powerful enough to give, as a by-product, an easier proof of the finite controllability result for sets of guarded TGDs [BGO10] (see Section 9). It also appears that rules with Cartesian products, even joinless, can be seen as interesting from some sort of practical point of view, motivated by Description Logics (where they would be called "concept products"). After all, "All Elephants are Bigger than All Mice" [RKH08].

1.2 Open problem/ future work

Does BDD property imply FC? In the proof of Theorem 1 we do not seem to use much much more than just the fact that Sticky Datalog$^\exists$ has the BDD property.

1.3 Outline of the technical part

In Section 2 Theorem 1 is proved, as a corollary to Theorem 2. The proof of Theorem 2, which is the main technical contribution of this paper, is presented in Sections 3–8. Finally, in a very short Section 9, we comment on the relations between our construction and the FC property for guarded sets of TGDs.

2 From Joinless Logic to Sticky Datalog$^\exists$

For a sticky set of TGDs $\mathcal{T}$ let $\mathcal{T}_0$ be the subset of $\mathcal{T}$ that consists of all the joinless rules in $\mathcal{T}$.

A pair $D, \mathcal{T}$, where D is a database instance, will be called weakly saturated if $D \models \mathcal{T}_0$. So if $D, \mathcal{T}$ is weakly saturated then each new element in $Chase(D, \mathcal{T})$ must have some (sticky) join in its derivation.

Suppose now that Sticky Datalog$^\exists$ does not have the FC property, and that some sticky set of TGDs $\mathcal{T}$, of maximal predicate arity $l > 0$, some finite database instance D and some query Φ are a

minimal counterexample for FC. When we say "minimal" here we mean that l is the smallest possible. By a "counterexample for FC" we mean that $Chase(D, \mathcal{T}) \not\models \Phi$ but $M \models \Phi$ for every finite model M of D and $\mathcal{T}$. The following two lemmas show that the above assumption leads to a contradiction:

Lemma 3 *The pair $D, \mathcal{T}$ is not weakly saturated.*

Lemma 4 *There is a finite database instance M such that the pair $M, \mathcal{T}$ is weakly saturated and M, $\mathcal{T}$ and Φ are also a counterexample for FC.*

Proof of Lemma 3: Suppose the pair $D, \mathcal{T}$ is weakly saturated.
   Let $\mathcal{T}_D$ be the set consisting of:

- all such dependencies $\sigma(T)$ that $T \in \mathcal{T}$, and $\sigma : Var(\mathcal{T}) \longrightarrow D$ is a partial substitution. By "partial substitution" we mean here a mapping that assigns constants from D to some of the variables from $Var(\mathcal{T})$;
- all the atoms true in D.

   We use the notation D here also to denote the active domain of D
   Now the trick is that we do not see the constants any more. Or rather we see them, but as a part of the name of the predicate forming the atom – so that $P(x, y)$, $P(x, a)$, $P(a, y)$, $P(b, y)$, $P(a, b)$ are now understood to be atoms of five different relations, one of them of arity 2, three of arity 1 and one of arity 0^2.
   Clearly, there is a canonical bijection between the elements of $Chase(\mathcal{T}, D) \setminus D$ and $Chase(\mathcal{T}_D, \emptyset)$. Each relation in $Chase(\mathcal{T}, D)$ can be defined as a disjunction of finite number of relations in $Chase(\mathcal{T}_D, \emptyset)$ and thus each UCQ Ψ in the language of $\mathcal{T}$ can be rewritten as some UCQ Ψ' in the language of $\mathcal{T}_D$. And each finite model of $\mathcal{T}_D$ can be seen (after adding the constants from D and forgetting that we used to read the relation names in the strange way) as a finite model of $\mathcal{T}, D$.
   Now define $\mathcal{T}_D^{<l}$ as the result of removing from $\mathcal{T}_D$ all the rules involving any relation of arity l. We have $Chase(\mathcal{T}_D, \emptyset) = Chase(\mathcal{T}_D^{<l}, \emptyset)$.

---

^{2} Obviously, x, y are variables and $a, b \in D$.

This is because the pair D, $\mathcal{T}$ was weakly saturated, so each new atom derived by a single application of a rule from $\mathcal{T}_D$ to facts from $\mathcal{T}_D$ must have a constant from D on a position which was immortal in $\mathcal{T}$, and – by stickiness – any atom derived later must contain this constant. It is an easy exercise for the reader to verify that if there existed a finite structure N such that $N \models \mathcal{T}_D^{<l}, \neg\Psi'$ then a finite structure M such that $N \models \mathcal{T}_D, \neg\Psi'$ could be constructed. So $D, \mathcal{T}_D^{<l}$ and Ψ' are a counterexample for FC, with maximal arity of relations being equal to $l - 1$. □

Proof of Lemma 4: Since Sticky Datalog$^\exists$ enjoys the BDD property, we know that there exists a query $\bar{\Psi}$ such that for each database instance F it holds that $F \models \bar{\Psi}$ if and only if $Chase(F, \mathcal{T}) \models \Psi$.

Let $\bar{D} = Chase(D, \mathcal{T}_0)$. Clearly, $Chase(\bar{D}, \mathcal{T}) = Chase(D, \mathcal{T})$. So $\bar{D} \not\models \bar{\Psi}$.

Since $\mathcal{T}_0$ is joinless, we know, from Theorem 2, that there exists a finite structure M such that $M \models \mathcal{T}_0, D, \neg\bar{\Psi}$, so in particular the pair $M, \mathcal{T}$ is weakly saturated.

Since $M \not\models \bar{\Psi}$ we get $Chase(M, \mathcal{T}) \not\models \Psi$. It remains to show that for each finite structure N, if $N \models M, \mathcal{T}$ then $N \models \Psi$. But this follows directly from the assumption, as $M \models D$. □

3   Theorem 2 – plan of the proof and our first little trick

We are planning to construct, for given D and $\mathcal{T}$, an infinite sequence of finite structures M_n, which will "converge" to $Chase$, in the sense that the following property will be satisfied:

Property 5 *(i)* $M_n \models D, \mathcal{T}$ *for each* $n \in \mathbb{N}$.
(ii) For each query Ψ, *and each* $n \in \mathbb{N}$ *if* $M_n \not\models \Psi$ *then* $M_{n+1} \not\models \Psi$.
(iii) For each query Ψ, *if* $Chase \not\models \Psi$ *then there exists* $n \in \mathbb{N}$ *such that* $M_n \not\models \Psi$.

Ψ is meant here to be a CQ or a UCQ (union of CQs).

Definition 6 *A formula* Φ *will be called M-true if* $M_n \models \Phi$ *for each* $n \in \mathbb{N}$.

Suppose, that a sequence M_n, satisfying Property 5(i),(ii) is constructed. Then:

Lemma 7 (first little trick) *If Φ is an M-true UCQ then there exists a disjunct of Φ which is M-true.*

Proof: By Property 5(ii) all conjunctive queries true in M_{n+1} are also true in M_n. Since Φ is true in each M_n, some disjunct from Φ must be true infinitely often, and therefore in each M_n. $\square$

The rest of the paper is organized as follows. In Section 4 family patterns are discussed, which constitute the body of our vehicle. In Section 5 the sequence M_n is defined and we present our second little trick, which is the main engine of the proof. In a very short Section 6 a trivial case of cyclic queries (whatever it means) is considered. In Section 7 we define a normal form of a conjunctive query and use our two little tricks to show a sort of normal form theorem:

Lemma 8 *For each M-true CQ ϕ there exist a CQ β in the normal form such that ($*$) β is M-true and ($**$) $Chase \models (\beta \Rightarrow \phi)$.*

Then, in Section 8 we prove:

Lemma 9 *If ϕ is in the normal form and $M_0 \models \phi$ then $Chase \models \phi$.*

As a corollary we get our main technical Lemma which, due to Lemma 7, implies Theorem 2:

Theorem 10. *If a conjunctive query ϕ is M-true then $Chase \models \phi$.*

From now on we will assume that D is empty. This can be done without loss of generality – see the argument from the proof of Lemma 3. We also assume (also w.l.o.g) that each rule from $\mathcal{T}$ is either of the form $Q(\bar{x}) \Rightarrow Q'(\bar{y})$, where $\bar{y} \subseteq \bar{x}$, or $Q_0(\bar{x}_0) \wedge Q_1(\bar{x}_1) \Rightarrow \exists y\, Q(y, \bar{x}_0, \bar{x}_1)$.

4 On the importance of family values

Let l be the maximal arity of the predicates in the signature under consideration.

Imagine a family of at most l members having dinner together. We will be interested in its *family pattern* – the complete information about the family relations between the diners. An important part of it is *family ordering* – the information about the ancestor relation within the family. All the families we are going to consider will be tree-like with this respect:

Definition 11 *By a family ordering we mean any union of ordered trees, whose set of vertices is $\{1, 2, \ldots k\}$ where $k \leq l$. If a family ordering is a tree then 1 is the root of this tree.*

If a family ordering is a tree, the youngest family member is understood to be the root of the tree. If Alice dines with her parents, we have a tree with Alice as the root, and two leaves. If Alice dines only with her boyfriend Bob, they form a family ordering consisting of two elements and no edges – this is why we need unions of trees rather than just trees.

But family ordering alone is not everything we want to know about a family. Alice dining only with her granny form the same ordering as Alice dining with her mother, but they do not form the same family pattern:

Definition 12 *A family pattern is a pair F, δ, where F is a family ordering and δ is a function assigning a number, from the set $\{1, 2, \ldots l\}$, to each pair j, i of elements of F such that $i <_F j$, where $<_F$ is the ancestor relation on F (i is an ancestor of j).*

Clearly, once l is fixed, the set of all possible family patterns is finite.

If j, i are members of some family, with pattern F, δ, and i is an ancestor of j then the value of $\delta(j, i)$ should be understood as "how j addresses i". One can imagine that *father* or *maternal grandmother* are possible values for $\delta(j, i)$.

We are soon going to see what the notions are good for. But first we need:

A remark about notations. For any syntactic object X by $Var(X)$ we will mean the set of all the variables in X. Symbol Q will be used to denote relation symbols. Letters P, R and T will denote atoms of variables. Letters A, B, C, D will denote atoms of elements

of *Chase*. *PP* will be used for parenthood predicates and sometimes also for parenthood atoms. To denote elements of *Chase* we will use a, b, c, d, while i, j, k will always be positions in atoms. F, G will be family orderings. For an atom $B = P_{F,\delta}(b_1, b_2...b_k)$ (where $b_1, b_2...b_k$ are constants in *Chase*) we define a notation $B(i) = b_i$. The same applies for atoms of variables.

Definition 13 *A set of joinless TGDs $\mathcal{T}$ respects family patterns if:*

1. *Each relation R of arity k in the signature of $\mathcal{T}$ contains, as a part of its name (as a subscript) a family pattern, with the family ordering F having exactly k vertices.*
2. *If $R \Rightarrow P$ is a rule of $\mathcal{T}$, where $R = Q_{F,\delta}(\bar{x})$, $P = Q'_{G,\gamma}(\bar{y})$ then $\bar{y} \subseteq \bar{x}$, and if $R(i) = P(j)$ and $R(i') = P(j')$ then:*
   - *$i <_F i'$ if and only if $j <_G j'$*
   - *if $i <_F i'$ then $\delta(i', i) = \gamma(j', j)$*
   By $\bar{y} \subseteq \bar{x}$ we mean that each element of the tuple $\bar{y}$ occurs in $\bar{x}$.
3. *If $R \wedge R' \Rightarrow \exists z\, P$ is a rule of $\mathcal{T}$, where arity of Q is k, arity of Q' is k', $R = Q_{F,\delta}(\bar{x})$ and $R' = Q'_{F',\delta'}(\bar{y})$ then $P = S_{G,\gamma}(z, \bar{x}, \bar{y})$ for some $S_{G,\gamma}$ and:*
   - *$i <_G j \Leftrightarrow (j = 1 \wedge i > 1) \vee (i - 1 <_F j - 1 \wedge 1 < i, j \leq k + 1) \vee (i - k - 1 <_{F'} j - k - 1 \wedge k + 1 < i, j \leq k + k' + 1)$*
   - *If $j = 1$ and $1 < i \leq k + k' + 1$ then $\gamma(i, j) = i$. If $1 < i, j \leq k + 1$ then $\gamma(i, j) = \delta(i - 1, j - 1)$. If $k + 1 < i, j \leq k + k' + 1$ then $\gamma(i, j) = \delta'(i - k - 1, j - k - 1)$.*
4. *The signature of $\mathcal{T}$ is a union of two disjoint sets: parenthood predicates (sometimes called PPs), which occur as the right hand sides of rules as in Condition (3), and projection predicates, which occur as the right hand sides of rules as in Condition (2).*
5. *For each projection predicate Q there is a parenthood predicate Q' such that $Q(\bar{t}) \Rightarrow \exists t\, Q'(t, \bar{t})$ and $Q'(t, \bar{t}) \Rightarrow Q(\bar{t})$ are rules of $\mathcal{T}$:*

Let us use our running metaphor to explain what is going on: Condition (1) says that relation atoms should be understood as families. To see the meaning of Condition (2) imagine that Alice used to live with her two ancestors: her father and her parental grandmother, whom she called "granny". Then something very sad happened, and

now she lives only with her grandmother. Condition (2) says that the grandmother is still her ancestor, and Alice still calls her "granny".

Now Condition (3). There were two families. Now they somehow have a child together, so they form one family. Each of the members of the two families is this child's ancestor. The child learns to address his ancestors by their positions in the family ordering, as he sees the ordering at the moment of his birth. The child's birth does not change the way his ancestors are addressing each other (notice that we do not care how the x's address the y's and vice versa – maybe they do not talk to each other at all?).

Conditions (4) and (5) cost nothing, and they help us to keep the proofs short. They are not exactly about family patterns but it was convenient to hide them all in one definition.

Now notice that, without loss of generality, we can assume that the set of TGDs under consideration $\mathcal{T}$ respects family patterns. This can be enforced by introducing new predicate names – one new predicate for each old predicate and for each way of arranging its arguments into family pattern. As each old predicate can be now seen as a disjunction of new predicates, by distributivity each UCQ can be rewritten as an equivalent UCQ in the new context. From now on we will **assume that $\mathcal{T}$ respects family patterns**.

Notice also, that due to conditions (4) and (5) each CQ can be always seen as a conjunction of parenthood predicates (possibly with additional fresh variables). From now on we will **assume that only parenthood predicates occur in queries**.

The following Lemma is an obvious consequence of the above assumption:

Lemma 14 *For each element a of Chase there exists exactly one parenthood predicate atom $A = PP(a, \bar{a})$ such that $Chase \models A$. It will be called* the Parenthood Atom *of a, and the elements of $\bar{a}$ will be called parents of a.*

Definition 15 *For two elements a, b of Chase we will say that a and b are 0-equivalent (denoted $a \equiv_0 b$) if the Parenthood Atoms of a and b are atoms of the same predicate. Suppose $a \equiv_0 b$, and A and B are Parenthood Atoms of a and b (resp.). Then, for each i, the elements $A(i)$ and $B(i)$ will be called* respective parents *of a and b.*

12

The next lemma can be easily proved by induction on the structure on Chase. Its second part says, using our running metaphor, that the way an element addresses its ancestors does not change during its lifetime:

Lemma 16 *If $Chase \models P$, for $P = Q_{F,\delta}(\bar{b})$ and $i <_F j$ then $P(i)$ is a parent of $P(j)$. If $R = PP_{G,\gamma}(\bar{a})$ is the parenthood atom of $P(j)$ then $P(i) = R(\delta(i,j))$.*

Now we have something slightly more complicated. The following lemma, which is is not going to be needed before Section 8, is where the whole power of family patterns is used:

Definition 17 *For a family ordering F and positions $i_1, i_2, \ldots i_s$ in F we define the set $PY_F(i_1, i_2, \ldots i_s)$ of positions in F as $\bigcap_{d=1\ldots s}\{j : \neg j \leq_F i_d\}$. When the context is clear we will write PY instead of PY_F.*

PY stands here for "possibly younger" – this is exactly the set of family members who potentially can be younger than each of $i_1, i_2, \ldots i_s$.

Lemma 18 (About the future.) *Let $A = PP_{F,\delta}(a, \bar{a})$ and suppose $Chase \models A$. Suppose $i_1, i_2, \ldots i_s$ are, pairwise incomparable by $<_F$, positions in F and let $b_1, b_2, \ldots b_s$ be equal to $A(i_1), A(i_2), \ldots A(i_s)$ respectively. Suppose $d_1, d_2, \ldots d_s$ is another tuple of elements of Chase such that $b_1, b_2, \ldots b_s \equiv_0 d_1, d_2, \ldots d_s$. Then there exists an atom $C = PP_{F,\delta}(c, \bar{c})$, such that:*

1. *$Chase \models C$;*
2. *$d_1, d_2, \ldots d_s$ equal $C(i_1), C(i_2), \ldots C(i_s)$ respectively;*
3. *if $j \in PY_F(i_1, i_2, \ldots i_s)$ then $A(j) \equiv_0 C(j)$;*
4. *if i is a position in F and $i <_F i_m$ then $A(i)$ and $C(i)$ are respective parents of $A(i_m)$ and $C(i_m)$.*

The lemma says that the potential of forming atoms in *Chase* only depends on the $\equiv_0$ equivalence class of elements (and tuples of independent elements), not on the elements themselves.

Proof. First of all notice that Claim (4) follows directly from Claim (2) and from Lemma 16.

For the proof of Claims (1)-(3) we will consider (a fragment of) the derivation tree of A in *Chase*, which we will call $\mathcal{D}$:

– Atom A is a root of $\mathcal{D}$ (and thus an inner node of $\mathcal{D}$). Positions $i_1, i_2, \ldots i_s$ are *marked* in A.
– Suppose an atom $B = P_{G,\gamma}(e, \bar{e})$ is an inner node of $\mathcal{D}$ and $Chase \models B$. Suppose $B' = P'_{G',\gamma'}(\bar{e}_1)$ and $B'' = P'_{G'',\gamma''}(\bar{e}_2)$ are such two atoms, true in *Chase* that B was derived in *Chase*, from B' and B'', by a single use of the rule: $X' \wedge X'' \Rightarrow \exists x \ X$, where $X' = P'_{G',\gamma'}(\bar{x}_1)$, $X'' = P'_{G'',\gamma''}(\bar{x}_2)$ and $X = P_{G,\gamma}(\bar{x}, x)$ Then B' and B'' are nodes of $\mathcal{D}$. If position i was marked in B and $X(i) = X'(j)$ then position j is marked in B'. Similarly, if position i was marked in B and $X(i) = X''(j)$ then position j is marked in B''. The case when B was derived by a projection rule $X' \Rightarrow X$ is handled analogously.
– A node with no marked positions is a leaf, called an unmarked leaf. A node which is a PP atom, and whose only marked position is its root is a leaf, called a marked leaf. All other nodes are inner nodes.

The idea here is that we trace the derivation of A back to the PP atoms of the elements b_i. The way we formulated it was a bit complicated, but we could not simply write "an atom is a leaf of $\mathcal{D}$ if it does not contain any of $b_1, b_2, \ldots b_s$". This was due to the fact, that b's can occur in the derivation not only on important positions – the positions that lead to i's in A, but also on unimportant positions, not connected, by the rules of $\mathcal{T}$, to any of the i's in A.

Now, once we have $\mathcal{D}$, consider another derivation $\mathcal{D}'$, with the underlying tree isomorphic to $\mathcal{D}$, defined as follows:

– If B is an unmarked leaf of $\mathcal{D}$ then B is also the respective leaf of $\mathcal{D}'$.
– If B is a marked leaf of $\mathcal{D}$, which means that B is the Parenthood Atom of some b_i, then the Parenthood Atom of d_i is the respective leaf of $\mathcal{D}'$.
– If B is an an inner node of $\mathcal{D}$, being a result of applying some rule from $\mathcal{T}$ to atoms B' and B'' (or just to B') then the same rule is used to create the respective atom in $\mathcal{D}'$

Notice that if $\mathcal{T}$ was not joinless, the last step would not always be possible. Now, the atom in the root of $\mathcal{D}'$ is the C from the Lemma. $\qquad\square$

5 The canonical models M_n and our second little trick

In this section we first define an infinite sequence of finite models M_n, which will "converge" to $Chase$. Then Lemma 24, which is the main engine of our machinery, will be proved.

Definition 19 $-$ *By 1-history of an element $a \in Chase$ (denoted as $H^1(a)$) we mean the set consisting all the parents of a.*
$-$ *$n + 1$-history of a is defined as $H^{n+1}(x) = \{x\} \cup \bigcup_{y \in H^1(x)} H^n(y)$.*

Consider an infinite well-ordered set of colors. For each natural number k we need to define the k-coloring of $Chase$:

Definition 20 $-$ *Then the k-coloring is the coloring of elements of Chase, such that each element of Chase has the smallest color not used in its k-history.*
$-$ *Define $type_0^k$ of an element a as the k-color of this element, and $type_{n+1}^k$ of a as a tuple consisting of the k-color of a and the tuple of $types_n^k$ of all the parents of a.*

Definition 21 *Two elements $a, b \in Chase$ are $n + 1$-equivalent (denoted as $a \equiv_{n+1} b$) if they are of the same $type_{n+1}^{n+1}$, their Parenthood Atoms are atoms of the same predicate, they are n-equivalent, and all their respective parents are n-equivalent.*

The reader should not feel too much confused by the colors and types. They will only be needed to deal with one trivial case. The real message is that "If $a \equiv_{n+1} b$ then their Parenthood Atoms are atoms of the same predicate and all their respective parents are n-equivalent".

It is easy to see that $\equiv_n$ is an equivalence relation of finite index.

Definition 22 $M_n = Chase/\equiv_n$. *Relations on M_n are defined, in the natural way, as minimal relations such that the quotient mapping is a homomorphism.*

Clearly, the sequence M_n satisfies Property 5(ii). It is also very easy to see that it satisfies Property 5(i) (the assumption that $\mathcal{T}$ is joinless needs to be used in the proof):

Lemma 23 $M_n \models \mathcal{T}$ *for each* $n \in \mathbb{N}$ *(and also* $M_n \models D$*, since we assume* D *to be empty)* .

The following lemma, says that if Ψ is M-true, then also some simpler query, which logically implies Ψ, is M-true.

Lemma 24 (second little trick) *Consider an M-true conjunctive query* $\Psi = P \wedge R \wedge \psi$*, where* P *is a Parenthood Atom of some variable* x*,* $R = Q_{F,\delta}(\bar{w})$ *and* $R(i) = x$*.*
Let σ *be a unification, which for every position* $j <_F i$ *in* R *identifies the variable* $R(j)$ *with the variable* $P(\delta(i,j))$*. Then* $\sigma(\Psi)$ *is also M-true.*

Notice that the lemma above would be clearly true if we wrote it its statement (twice) "true in *Chase*" instead of "M-true". This is since each element in *Chase*, so in particular the value of x, has a unique tuple of parents. The variables in R, occupying positions $j <_F i$ must be interpreted as those parents, and in order to satisfy Ψ in *Chase* we have no other choice but to interpret them as the respective parents of (the interpretation of) x. It is not that simple in M_n, since an element of this structure no more has a unique tuple of "parents".

To prove the Lemma we first need to understand what does it mean for a query to be true in M_n.

Definition 25 *For a CQ* ϕ *let* $Occ(\phi) = \bigcup_{T \in \phi}(\{1, 2 \ldots arity(T)\} \times \{T\})$*. An n-evaluation of* ϕ *is a function* $f : Occ(\phi) \rightarrow Chase$ *assigning, to each atom* T *from* ϕ *and each position* i *in* T*, an element* $f(i, T) \in Chase$*, in such a way that:*

(*) *for each pair of atoms* T, T' *in* ϕ *if* $T(i) = T(i')$ *then* $f(i, T) \equiv_n f(i', T')$*.*
(**) *for each atom* T *in* ϕ *it holds that* $Chase \models f(T)$*.*

Where by $f(T)$ *we mean the atomic formula resulting from replacing, in* T*, each* $T(i)$ *(which is variable) by* $f(i, T)$ *(which is an element of Chase).*

It is easy to see that:

Lemma 26 $M_n \models \phi$ *if and only if there exists an n-evaluation of* ϕ.

Proof (of Lemma 24): We want to show that for each natural n the query $\sigma(\Psi)$ is true in M_n. We know that Ψ is M-true, so $M_{n+1} \models \Psi$.

Suppose f is an $n + 1$-evaluation of Ψ. Lemma will be proved if we can show that f is an n-evaluation of $\sigma(\Psi)$, and in order to show that it is enough to prove that $f(P(\delta(i,j)), P) \equiv_n f(R(j), R)$ for each position $j <_F i$ in R.

But we know that $f(P(1), P) \equiv_{n+1} f(R(i, R))$, and since f satisfies condition (**) of Definition 25, we know that $f(P(\delta(i,j)), P)$ and $f(R(j), R)$ are respective parents of $f(P(1), P)$ and $f(R(i, R))$. Now use the definition of the relation $\equiv_n$ to end the proof. $\qquad\square$

6 Cyclic queries

Definition 27 *Let* ϕ *be a CQ.*

1. *By* $\xrightarrow{*}_\phi$ *we mean the transitive (but not reflexive) relation such that for each* $x, y \in Var(\phi)$ *if there is an atom* $P = Q_{F,\delta}(\bar{t})$ *in* ϕ *and positions* i, j *in* F, *such that* $P(i) = y$, $P(j) = x$ *and* $i <_F j$, *then* $x \xrightarrow{*}_\phi y$.
2. ϕ *is called* acyclic *if* $\xrightarrow{*}_\phi$ *is a partial order[3] on* $Var(\phi)$ *(which means that it is antisymmetric). Otherwise it is cyclic.*

Clearly, if ϕ is cyclic then $Chase \not\models \phi$. But it is also very easy to see that:

Lemma 28 *If* ϕ *is a cyclic query consisting of* k *atoms, then* $M_{k+1} \not\models \phi$. *So a cyclic query is never M-true.*

Proof of the lemma is left as an exercise for the reader. Hint: notice that, by Definition 20 and the first claim of Lemma 16, $M_{k+1} \not\models \phi$ would imply the existence of an element of $Chase$ having an $k + 1$-equivalent element in its $k + 1$-history, which is impossible, for coloring reasons. This was the only place where we needed to think about colors.

From now on we only consider acyclic queries.

---

[3] When $x \xrightarrow{*}_\phi y$ then we think that y is smaller than x. Mnemonic hint: the arrowhead of $\to$ looks like $>$.

7 Acyclic queries and the normal form

Definition 29 *Let ϕ be an acyclic CQ.*

1. *We call a variable in $Var(\phi)$ a master if it is a child in some Parenthood Atom in ϕ. Otherwise it is called a serf.*
2. *By $\rightarrow_\phi$ we mean the direct successor subrelation of $\xrightarrow{*}_\phi$ (i.e. the smallest subrelation of $\xrightarrow{*}_\phi$ giving $\xrightarrow{*}_\phi$ as its transitive closure).*
3. *By $x \xrightarrow{m}_\phi y$ we mean that x is a minimal, with respect to $\xrightarrow{*}_\phi$, among such masters that $x \xrightarrow{*}_\phi y$. If y is a serf and $x \xrightarrow{m}_\phi y$ we say that y is a serf of x.*
4. *For an atom $P = Q_{F,\delta}(\bar{t})$ let $\mathcal{I}(P)$ denote the "set of maximal master positions", that is the set of such non-root positions i in P that the variable $P(i)$ is master and for each $j \neq 1$ such that $i <_F j$, $P(j)$ is a serf.*

We are ready to define the normal form of a conjunctive query:

Definition 30 *A conjunctive query β is in the normal form if:*

1. *If P is the Parenthood Atom of a master variable x, if $R = Q_{F,\delta}(\bar{t})$ is another atom, such that $R(i) = x$, and if j is a position in R such that $j <_F i$, then $R(j) = P(\delta(i,j))$.*
2. *No one is a serf of two masters.*
3. *If $i \in PY(\mathcal{I}(P))$ and $j \neq i$ are positions in some atom P, then $P(i) \neq P(j)$.*

The first of the conditions above is the one from Lemma 24, and it reflects the idea, that we can restrict our attention to queries where variables have unique tuples of parents. The two remaining conditions are technical and will be needed in Section 8. The rest of this Section is devoted to the **proof of Lemma 8**:

For a variable x and parenthood predicate Q let P_x^Q be an atom having Q as the relation symbol, x as the child, and fresh variables in all remaining positions. Let Ψ_x be a disjunction of all possible P_x^Q (one for each Q).

Let $\bar{\phi}$ be $\phi \wedge \Phi$, where $\Phi = \bigwedge_{x \in Var(\phi)} \Psi_x$. Clearly, if ϕ was M-true then $\bar{\phi}$ is also M-true – the only new constraint is that (the element

being the realization of) each variable from ψ was born, in one way or another.

By distributivity $\bar{\phi}$ is a UCQ. By First Little Trick we know that there is a disjunct in $\bar{\phi}$ which is M-true. Call it β_0. It is easy to see that β_0 is a conjunction of ϕ and of one P_x^Q (call it PP_x) for each $x \in Var(\phi)$, so $Chase \models (\beta_0 \Rightarrow \phi)$.

What we did so far is that we elevated the variables being serfs in ϕ to masterhood. We did it because they might have been involved in some equalities in ϕ, which are beyond our control and could violate their serf status, as described by conditions (ii) and (iii) of Definition 30.

Let now β be a result of the following **Procedure**:

$\beta := \beta_0$;
while there are P and R in β and a position i in R, which violate condition (i), unify them by Second Little Trick to get a new β.

Clearly, this procedure terminates (as the number of variables decreases), and the final β satisfies (i). Notice that as Φ contained PP atoms also for variables which had already been masters in ϕ, there is some redundancy in β – the procedure gave as (possibly unified) copy of ϕ as a part of Φ.

Lemma 31 β *satisfies (ii) and (iii).*

Proof of the lemma: Consider the unification step from Second Little Trick. Suppose the variables in R, in positions $j <_F i$ are fresh. Then σ can be defined in such a way, that it is the identity function on $Var(P)$ (so in particular, σ does not identify variables in P).

Let PP_x^β be the atom PP_x, as it appears in β, that is PP_x after all the unifications executed during the Procedure. Let y be a master, such that $x \xrightarrow{m}_\beta y$. Then it is easy to see that y occurs in PP_x^β, and $\{i : PP_x^\beta(i) = y\} \subseteq \mathcal{I}(PP_x^\beta)$.

Now let us go back to Φ. First we are going to rename its (fresh) variables. The variable in position i in the PP atom of x will now be called $\langle x, i, y \rangle$ if and only if it is substituted with y in β. So for example the variable in the root of PP_x will now be $\langle x, 1, x \rangle^4$.

---

[4] This is not quite true. If you notice that, you know how to correct that.

Notice that we did not change the formula – we still have the same Φ, with fresh variables, but the complete information about β is already encoded in the names of the variables. Let F_x be the family ordering of atom PP_x.

Consider now the following procedure:

$\gamma := \Phi$;

do { take a $\overset{*}{\to}_\beta$ minimal master x that was not yet considered;
     for each position $i \in \mathcal{I}(PP_x^\beta)$
          { let $y = PP_x^\beta(i)$;
           modify γ by substituting $\langle y, 1, y \rangle$ for $PP_x(i)$;
           execute Second Little Trick for the query γ,
              atoms PP_x as R, PP_y as P and position i;}
     mark x as considered;}

The new procedure performs all the unifications needed to turn Φ into β, but does it in an order defined by $\overset{*}{\to}_\beta$, which is only known once we know β. We leave it as an exercise for the reader to see that the γ resulting from the above procedure indeed equals β (modulo repeated atoms).

Another exercise is that for a serf $\langle y, i, t \rangle$ and master $\langle x, 1, x \rangle$ it holds that $\langle x, 1, x \rangle \overset{m}{\to}_\beta \langle y, i, t \rangle$ if and only if $y = x$. Condition (ii) follows directly from that. Hint: Notice that now each atom T in Φ first plays the role of the R from the Second Little Trick some number of times. The names of the variables in $PY(\mathcal{I}(T))$ remain unchanged during this phase. Then it always plays P, while the respective variables in R are fresh, so, according to the observation we made at the beginning of this section, the variables in T are not being changed any more. This hint also proves that condition (iii) holds for γ.

This ends the proof of Lemma 31 and of Lemma 8.

8   Proof of Lemma 9

In this section we show what remains to be shown: that if $M_0 \models \psi$ and ψ is in normal form then also $Chase \models \psi$. It will be done by induction on $\overset{*}{\to}_\psi$.

Definition 32 *Subset $S \subseteq Var(\psi)$ is a master ideal of $\langle Var(\psi), \xrightarrow{*}_{\psi} \rangle$ if:*

1. *If $x \in S$ and $x \xrightarrow{*}_{\psi} y$ then also $y \in S$.*
2. *All maximal elements of S are master variables.*

The following definition shows that the difference between the fact that $M_0 \models \psi$ and that $Chase \models \psi$ is quantitative rather than qualitative:

Definition 33 *A 0-evaluation f is faithful with respect to a master ideal S if for each pair of atoms R, P in ψ such that $Var(R), Var(P) \subseteq S$ if $R(i) = P(i')$ then $f(i, R) = f(i', P)$*

If f is faithful with respect to S then for an atom R in ψ, such that $Var(R) \subseteq S$, and for $z = R(i)$, we write $f(z)$ instead of $f(i, R)$.

Clearly $Chase \models \psi$ if and only if there exists a 0-evaluation faithful with respect to $Var(\psi)$. On the other hand, since $M_0 \models \psi$, there exists a 0-evaluation faithful with respect to $\emptyset$. We are going to gradually reconstruct this 0-evaluation to make it more and more faithful, until we get one faithful with respect to $Var(\psi)$. We will need the following easy remark about 0-evaluations:

Definition 34 *Suppose f is a 0-evaluation, $f' : Occ(\psi) \rightarrow Chase$ is any function, and P is an atom in ψ. We say that f' is P-similar to f if:*

- *$f'(i, R) = f(i, R)$ for each atom $R \neq P$, and each position i in R;*
- *$Chase \models f'(P)$*
- *$f'(i, P) \equiv_0 f(i, P)$*

Lemma 35 *If f is a 0-evaluation and f' is P-similar to f then f' is also a 0-evaluation.*

Due to an induction argument, to prove Lemma 9, is remains to show:

Lemma 36 *Let S be a master ideal, and f a 0-evaluation faithful with respect to S. Let $x \in Var(\psi)$ be a minimal master variable not*

in S and P the Parenthood Atom of x in ψ. Let S' be the master ideal generated by x and S.

Then there exists a 0-evaluation f', P-similar to f and faithful with respect to S'.

Proof: Suppose $P = PP_{\langle F, \delta \rangle}(x, \bar{x})$. We will define $f'(P)$. Then we will check that the conditions from Definition 34 hold, so f' is a 0-evaluation. Finally we will notice that f' is S'-faithful.

Let $y_1, y_2, \ldots y_{s_0}$ be all the master variables which are $\overset{*}{\rightarrow}_\psi$ maximal in S and smaller than x. They all occur, in F-incomparable positions $i_1, i_2 \ldots i_s$, in P. Let $d_i = f(y_i)$ and let $b_i = f(i, P)$. Clearly, since f is an evaluation, we have $b_i \equiv_0 d_i$. Notice that possibly $s > s_0$, since multiple occurrences of y's are allowed.

Notice also that $S' \setminus S = \{P(j) : j \in PY(i_1, i_2 \ldots i_s)\}$. This is thanks to Definition 30 (2). It is also going to be important for us that each of the new variables occurs in P only once: due to Definition 30 (3), for $j, j' \in PY(i_1, i_2 \ldots i_s)$ if $j \neq j'$ then $P(j) \neq P(j')$.

We are now in the situation of Lemma 18, where $A = f(P)$. Let C be as in the Lemma. For any position j in $PY(i_1, i_2 \ldots i_s)$ define $f'(j, P)$ as $C(j)$. Notice, that we can be sure (thanks to Lemma 18) that $f'(j, P) \equiv_0 f(j, P)$.

Let now j be a position in P which is not in $PY(i_1, i_2 \ldots i_s)$. That means that the variable $P(j)$ is in S. Define $f'(j, P)$ as $f(P(j))$. The condition $f'(j, P) \equiv_0 f(j, P)$ now holds trivially, since f was a 0-evaluation.

We defined a function f', which satisfies the first and the third condition from Definition 34. Now we need to make sure that $Chase \models f'(P)$. We know that $Chase \models C$, so this part of proof would be finished if we could show that $f'(P) = C$. Surprisingly, this is the crucial moment, the one we spent long pages preparing for. The full power of the normal form and family patterns is going to be used in the next 6 lines:

Consider two positions in P: $i \in \{i_1, i_2 \ldots i_s\}$ and $j <_F i$. Let $y \in \{y_1, y_2, \ldots y_{s_0}\}$ be such that $y = P(i)$ and let $z = P(j)$. The variable y is a master, so its Parenthood Atom, call it P^y, is in ψ.

Since ψ is in the normal form, we know that $P^y(\delta(i, j)) = z$. Since we defined $f'(j, P)$ to be $f(z)$, we get $f'(j, P) = f(\delta(i, j), P^y)$.

What we want to show is that $f'(j, P) = C(j)$. But this now follows directly from Lemma 16.

In order to finish the proof of the Lemma we still need to notice that f' is S'-faithful. The atoms described by Definition 33 are now all the atoms that were already contained in S, and one new atom P. If $P(j)$ was in S we defined $f(j, P)$ as $f(P(j))$, so we did not spoil anything. The only problem could be with the values assigned to positions in P with variables from $S'\backslash S$. But, as we mentioned above, each of the new variables occurs in P only once, so the condition from Definition 33 is trivially satisfied . $\qquad\square$

9   Remark about guarded TGDs

The proof in Sections 3-8 can be also read as a new proof of the FC property for guarded TGDs [BGO10]. The only difference is that, in order to construct M_n, it is not enough, in the guarded case, to remember by which rules last n generations of parents of an element were born, but also what other atoms are true about the elements. That is why a condition "and the n-histories of a and b are isomorphic" needs to be added to Definition 21. On the other hand, all the family orderings we would need to consider in the guarded case are total orderings, which significantly simplifies everything – for example the ordering $\overset{*}{\to}_\psi$ of variables in a query ψ in a normal form would now be a tree.

10   References

[BGO10] V. Barany, G. Gottlob, and M. Otto. *Querying the guarded fragment*; Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010, Edinburgh, UK, pp. 1–10, 2010.

[CGT09] A. Cali G. Gottlob, and T. Lukasiewicz; *A general datalog-based framework for tractable query answering over ontologies*; In Proc. of PODS, 2009.

[CGP10] A. Cali, G. Gottlob, and A. Pieris. *Advanced processing for ontological queries*; Proc. VLDB-10, 3(1):554–565, 2010.

[JK84] D. S. Johnson and A. C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. JCSS 28(1):167-189, 1984.

[R06] R. Rosati; *On the decidability and finite controllability of query processing in databases with incomplete information*; in Proc. PODS 2006, pp. 356–365.

[RKH08] S. Rudolph and M. Krötzsch and P. Hitzler; *All Elephants are Bigger than All Mice*; 21st Description Logic Workshop Dresden, Germany, 2008.