Converging to the Chase – a Tool for Finite Controllability *

Tomasz Gogacz,Jerzy Marcinkowski

November 5, 2014

Abstract

We solve a problem, stated in [CGP10a], 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 $Chase(\mathcal{T}, \mathbb{D})$, for a database instance $\mathbb{D}$ and a set of tuple generating dependencies and Datalog rules $\mathcal{T}$, by an infinite sequence of finite structures, all of them being models of $\mathcal{T}$ and $\mathbb{D}$.

1 Introduction

Tuple generating dependencies (TGDs), recently 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 interested in here is computing certain answers to queries in the situation when some semantical information about the database is known (in the form of theory $\mathcal{T}$, consisting of TGDs), but the knowledge of the database facts is limited, so that the known set of facts $\mathbb{D}$ does not necessarily satisfy the dependencies of $\mathcal{T}$.

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 while 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 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* [?].

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

---

*This is the full version of an extended abstract published in the LICS 2013 proceedings

Linear Datalog$^\exists$ programs. They 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. Let us note that allowing variable repetitions in the heads does not change the Finite Controllability status of a program, as we can always remember the equalities as part of the relation name, so we w.l.o.g. assume that such repetitions are not allowed (see Section 5 for much more about this issue). 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[1]

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, as they only have one atom in the body.

Sticky Datalog$^\exists$ is a logic introduced in [CGP10a] and then extended in [CGP10b] as Sticky-Join Datalog$^\exists$. Theory $\mathcal{T}$ is Sticky, if some positions in the predicates from the signature of $\mathcal{T}$ can be marked as "immortal" in such a way that the following conditions are satisfied:

- If some variable occurs in at least one immortal position in the body of a rule from $\mathcal{T}$ then the same variable must occur in an immortal position in the atom being the head of the same rule.

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

The above definition of Sticky Datalog$^\exists$ is a slightly different wording[2] of (equivalent, when restricted to single-head TGDs) Definition 1 from [CGP10a], and resembles what in the paper [CGP10b] is called "the sticky-join property" (see Section 5.1 in [CGP10b]). Actually, both Theorem 2 of our paper and its proof hold for any possible logic having the sticky-join property, which includes Sticky Datalog$^\exists$ and Sticky-Join Datalog$^\exists$ (which is a version defined in [CGP10b]). In fact, the difference between the two logics can only be seen if repeated variables in the heads of the rules are allowed and, as we said before, from the point of view of Finite Controllability we can disallow them w.l.o.g..

Apart from decidability, the properties of such logics which are considered desirable and receive a lot of attention are:

---

[1]The term "Joinless Logic" was used in [CGP10a] (Theorem B.2 there) to denote logic which is not really joinless – a variable may occur more than once there, but only in one atom in the body. It is however very easy to see that any TGD can be simulated by one TGD and one Datalog rule, which are "joinless" in this sense. Unlike [CGP10a], when we say "joinless" we really mean "joinless".

[2] Both the definitions of Sticky Datalog$^\exists$ involve comparing the set J of positions where joins occur with the set V of positions where variables are allowed to vanish. In [CGP10a] authors have chosen to state the condition in the language of pullback of V by by the rules of $\mathcal{T}$ while we prefer to think in terms of pushforward of J by the rules of $\mathcal{T}$.

Bounded Derivation Depth property (BDD). A set $\mathcal{T}$ of TGDs has the bounded derivation depth property if for each UCQ Ψ there is a constant $k_\Psi \in \mathbb{N}$, such that for each database instance $\mathbb{D}$ if $Chase(\mathcal{T}, \mathbb{D}) \models \Psi$ then $Chase^{k_\Psi}(\mathcal{T}, \mathbb{D}) \models \Psi$. The BDD property turns out to be equivalent to positive existential first order rewriteability :

Theorem 1 *$\mathcal{T}$ has the BDD property if and only if for each UCQ Ψ there exist a UCQ Φ such that for each database instance $\mathbb{D}$ (finite or not) it holds that $Chase(\mathcal{T}, \mathbb{D}) \models \Psi$ if and only if $\mathbb{D} \models \Phi$.*

This theorem is stated in [?], not as an equivalence however, but only as the "only if" implication – if a theory is BDD then queries are rewritable as UCQs. We believe that the proof of the "if" implication is folklore, but let us include it here, for sake of completeness:

Fix a theory $\mathcal{T}$ and assume that query Ψ is rewritable. Let $\Phi = \phi_1 \vee \phi_2 \ldots \vee \phi_m$ be the rewriting, where each ϕ_i is a conjunctive query. For each ϕ_i let M_i be the canonical structure of ϕ_i. Clearly, for each i we have $M_i \models \Phi$ so also for each i there is $Chase(\mathcal{T}, M_i) \models \Psi$. Let k_i be a natural number such that $Chase^{k_i}(\mathcal{T}, M_i) \models \Psi$. Now define k_Ψ as $\max\{k_i : 1 \leq i \leq m\}$. It is now easy to see that, for any $\mathbb{D}$, it holds that if $Chase(\mathcal{T}, \mathbb{D}) \models \Psi$ then $Chase^{k_\Psi}(\mathcal{T}, \mathbb{D}) \models \Psi$. $\square$

Finite Controllability (FC). A set $\mathcal{T}$ of TGDs has the finite controllability property if for each UCQ Ψ and each database instance $\mathbb{D}$ if $Chase(\mathcal{T}, \mathbb{D}) \not\models \Psi$ then there exists a finite structure M such that $M \models \mathcal{T}, \mathbb{D}$ but $M \not\models \Psi$.

A logic is said to be FC (or BDD) if each $\mathcal{T}$ in this logic is FC (BDD). A triple $\mathcal{T}$, $\mathbb{D}$, Ψ such that $Chase(\mathcal{T}, \mathbb{D}) \not\models \Psi$ but for each finite structure M if $M \models \mathcal{T}, \mathbb{D}$ then also $M \models \Psi$ will be called a **counterexample for FC**. It is usually quite easy to see whether a given logic is BDD and it is usually very hard to see whether it is FC.

Previous works. 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 $\bar{\mathbb{D}}$, such that $\bar{\mathbb{D}} \models \mathcal{T}, \mathbb{D}, \neg\Psi$" if all counterexamples $\bar{\mathbb{D}}$ for Ψ we can produce are infinite. This problem was solved by Rosati [Ros06], who proved, by a complicated argument, that IDs (Linear Datalog$^\exists$) have the finite controllability property. His result was improved in [BGO10] where FC is shown for Guarded Datalog$^\exists$.

Sticky Datalog$^\exists$ was introduced in [CGP10a], 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 [CGP10a], 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 [CGP10a].

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

Theorem 2 *Sticky Datalog$^\exists$ is FC.*

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

Theorem 3 *Joinless Logic is FC.*

To prove Theorem 3 we propose a technique, which we think is quite elegant[3], and relies on two main ideas. One is that we carefully trace the relations (we call them "family patterns") between pairs of 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 families which the elements (and their ancestors) are members of, and construct an infinite sequence of models as the quotient structures of these equivalence relations. This leads to a sequence of finite models, that, in a sense, "converges" to *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 from a Joinless theory is a huge and very complicated structure, much more complex than the bounded tree-width *Chase* resulting from guarded (or Linear) TGDs, and the ability to control it can give insight into chases generated by logics enjoying better practical motivation – Theorem 2 serves here as a good example. But still Theorem 3 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]. 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].

Open problem: BDD/FC conjecture. Does the BDD property always imply FC? In the proof of Theorem 2 we do not seem to use much much more than just Theorem 3 and the fact that Sticky Datalog$^\exists$ is BDD. In our parallel

---

[3]This is just our opinion. The reader has of course the right to have his own.

paper [GM13] we show that each theory over a binary signature which is BDD is also FC. We also explain there why the full conjecture is not so easy to prove.

Outline of the paper. Next section is devoted to preliminaries. Basic concepts are explained there and notations are introduced.

In Section 3 we prove Theorem 2, assuming Theorem 3.

The proof of Theorem 3, which is the main technical contribution of this paper, is presented in Sections 4–15.

2 Preliminaries

Most of the notions and notations in this paper are standard for mathematical logic and database theory. In particular, if ϕ is a formula, Ψ is a set of formulas and $\mathcal{M}$ is a structure, then by $\mathcal{M} \models \phi$ we mean that ϕ is true in $\mathcal{M}$ and by $\mathcal{M} \models \Psi$ we mean that each formula in Ψ is true in $\mathcal{M}$. By $\Psi \models \phi$ we mean that for each structure $\mathcal{M}$ such that $\mathcal{M} \models \Psi$ there is also $\mathcal{M} \models \phi$.

Let us remind the reader that a *tuple generating dependency* (TGD), or a Datalog$^\exists$ *rule* (or just "rule") is a formula of the form

$$\forall \bar{x} \ (\Phi(\bar{x}) \Rightarrow \exists y \ Q(y, \bar{y}))$$

where Φ is a conjunction of atoms (a conjunctive query without existential quantifiers), 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. Notice that w.l.o.g we only consider single-head TGD, which means that there is always only one atom in the head (i.e. right hand side) of a rule. By a *theory* we mean a finite set consisting of some TGDs and some Datalog rules (which are TGDs without the existential quantifier in the head).

For a theory $\mathcal{T}$ and a database instance $\mathbb{D}$ the structure $Chase(\mathcal{T}, \mathbb{D})$ is defined in the standard way, and by $Chase^i(\mathcal{T}, \mathbb{D})$ we mean the structure being the i-th stage of the fixpoint procedure leading to $Chase(\mathcal{T}, \mathbb{D})$.

More precisely, we define $Chase^0(\mathcal{T}, \mathbb{D}) = \mathbb{D}$. Once $Chase^i(\mathcal{T}, \mathbb{D})$ is defined, we define $Chase^{i+1}(\mathcal{T}, \mathbb{D})$ as the superstructure of $Chase^i(\mathcal{T}, \mathbb{D})$ being the result of the following procedure:

for each rule $\Phi(\bar{x}) \Rightarrow \exists y \ Q(y, \bar{y})$ from $\mathcal{T}$ and **for each** valuation ρ mapping variables in $\bar{x}$ to elements of $Chase^i(\mathcal{T}, \mathbb{D})$ such that $Chase^i(\mathcal{T}, \mathbb{D}) \models \Phi(\rho(\bar{x}))$ but there is no b such that $Chase^i(\mathcal{T}, \mathbb{D}) \models Q(b, \rho(\bar{y}))$, we add to $Chase^{i+1}(\mathcal{T}, \mathbb{D})$ a new element b and the atomic fact $Q(b, \rho(\bar{y}))$;

similarly, **for each** Datalog rule from $\mathcal{T}$ and **for each** relevant valuation an atomic fact is added to $Chase^{i+1}(\mathcal{T}, \mathbb{D})$ if it was not yet there.

Then $Chase(\mathcal{T}, \mathbb{D})$ is defined as the union of all $Chase^i(\mathcal{T}, \mathbb{D})$ for $i \in \mathbb{N}$. We often write $Chase(\mathcal{T})$ (or $Chase$) instead of $Chase(\mathcal{T}, \mathbb{D})$ when $\mathbb{D}$ (and $\mathcal{T}$) can be easily guessed from the context. Notice that when we say "we add b to $Chase^{i+1}(\mathcal{T}, \mathbb{D})$" we think of relational structures as the mathematicians do – as of a set of elements. But when we say "add an atomic fact to $Chase^{i+1}(\mathcal{T}, \mathbb{D})$"

then we see a structure in the way consistent with the database tradition – as a set of facts. We will feel free to use both conventions, depending on which is more convenient at the moment.

Notice that the chase procedure as we define it above is *standard* (lazy) chase. Unlike *oblivious chase*, which is also often considered in database theory, standard chase adds new elements (and atoms which involve them) only when they are needed, that is when the body of some rule is satisfied for some valuation but the head of this rule is not. The choice of standard chase has an implication that will later be useful: the standard chase procedure is idempotent, which means that $Chase(\mathcal{T}, \mathbb{D}) = Chase(\mathcal{T}, Chase(\mathcal{T}, \mathbb{D}))$.

Clearly, we have $Chase(\mathcal{T}, \mathbb{D}) \models \mathbb{D}, \mathcal{T}$, but there is no reason to think that $Chase^i(\mathcal{T}, \mathbb{D}) \models \mathcal{T}$ for any $i \in \mathbb{N}$. Since $Chase(\mathcal{T}, \mathbb{D})$ is a "free structure", it is well known, and very easy to see, that for any query Φ (being a union of positive conjunctive queries, or UCQ; remember that **all queries we consider in this paper are positive**) $\mathbb{D}, \mathcal{T} \models \Phi$, if and only if $Chase(\mathcal{T}, \mathbb{D}) \models \Phi$.

A remark about notations. For any syntactic object X by $Var(X)$ we will mean the set of all the variables in X.

Letters P, Q and R will denote predicates or atoms of variables. Letters A, B, C, D will denote atoms of elements of $Chase$. PP will be used for parenthood predicates (which are a special sort of predicates in our proof) and sometimes also for parenthood atoms.

To denote elements of $Chase$ we will use a, b, c, d, while i, j, k will be positions in atoms or other small numbers.

F, G will be family orderings, and γ and δ will be functions occurring in the family patterns – something we develop in Section 6 and use extensively then.

For an atom $B = Q(b_1, b_2...b_k)$ (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.

The letter M is always used to define a relational structure (usually a finite one). $\mathbb{D}$ is also used in this context, usually as the initial database instance on which chase is run.

Ψ and Φ are formulae, often unions of conjunctive queries. The characters ϕ, ψ and β are used to denote conjunctive queries (or just conjunctions of atoms).

When we say "conjunctive query", or UCQ, we usually mean a boolean conjunctive query or boolean UCQ. This in particular applies (w.l.o.g.) to the definitions of FC and BDD. In order to keep the notation as light as possible, when talking about boolean CQs we often omit the existential quantifiers in front.

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

This Section is devoted to the proof of Theorem 2 (assuming Theorem 3).

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

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

must have some (sticky) join in its derivation and, in consequence each atom of $Chase(\mathcal{T}, \mathbb{D})$ is either an atom of $\mathbb{D}$ or it contains some constant from $\mathbb{D}$ in a marked position. This is because (if $\mathbb{D}, \mathcal{T}$ is weakly saturated) the only way for $\mathcal{T}$ to derive any atoms which are not in $\mathbb{D}$ is to use some rule with the sticky join, which requires immortalizing one of the arguments

Suppose now that Sticky Datalog$^{\exists}$ is not FC and we will consider counterexamples $\mathcal{T}, \mathbb{D}, \Phi$ for FC with sticky $\mathcal{T}$. By "arity" of $\mathcal{T}$ we will mean the maximal arity of atoms in the heads of the rules of $\mathcal{T}$. We will call a counterexample $\mathcal{T}, \mathbb{D}, \Phi$ *minimal* if the arity of $\mathcal{T}$ is smallest possible.

We are going to prove two Lemmas:

Lemma 4 *Suppose a triple $\mathcal{T}, \mathbb{D}, \Phi$ is a minimal counterexample for FC. Then the the pair $\mathbb{D}, \mathcal{T}$ is not weakly saturated.*

Lemma 5 *Let $\mathcal{T}, \mathbb{D}, \Phi$ be a counterexample for FC. There is a finite database instance $\mathbb{D}'$ such that the pair $\mathbb{D}', \mathcal{T}$ is weakly saturated and the triple $\mathcal{T}, \mathbb{D}', \Phi$ is also a counterexample for FC.*

Notice that proof of Theorem 2 will be finished once the two above lemmas are proved. This is because the assumption that a minimal counterexample exists will lead to a contradiction: by Lemma 5 we will be able to get a minimal weakly saturated counterexample – something that is ruled out by Lemma 4.

Theorem 3 will be used to prove Lemma 5.

Proof of Lemma 4: Let $\mathcal{T}, \mathbb{D}, \Phi$ be a counterexample for FC, with l being the arity of $\mathcal{T}$. Suppose the pair $\mathbb{D}, \mathcal{T}$ was weakly saturated. We will construct a new sticky theory $\mathcal{T}_{\mathbb{D}}$ of   arity at most $l-1$, over a new signature $\Sigma_{\mathbb{D}}$ and a new query $\Phi_{\mathbb{D}}$, such that the triple $\mathcal{T}_{\mathbb{D}}, \emptyset, \Phi_{\mathbb{D}}$ is also a counterexample for FC. This will contradict the assumption that l was minimal possible, and thus end the proof of the Lemma.

Let us start from the definition of $\Sigma_{\mathbb{D}}$. For a predicate $Q \in \Sigma$, of arity j, and for a partial function $\gamma : \{1, 2, \ldots j\} \to \mathbb{D}$ let Q_{γ} be a new predicate, of arity $j - |Dom(\gamma)|$. $\Sigma_{\mathbb{D}}$ will be the set of all possible predicates Q_{γ}, where Q and γ are as above. Since we did not assume that γ is non-empty we have that $\Sigma \subseteq \Sigma_{\mathbb{D}}$ (we identify Q with $Q_{\emptyset}$).

To denote the predicates from $\Sigma_{\mathbb{D}}$ we are going to use the notational convention that will now be described by an example. If $Q(\_, \_, \_)$ is a ternary predicate from Σ, $\gamma = \{\langle 2, c \rangle\}$ and $\gamma' = \{\langle 1, c \rangle, \langle 3, a \rangle\}$ then Q_{γ} will be denoted as $Q(\_, c, \_)$ and $Q_{\gamma'}$ will be denoted as $Q(c, \_, a)$. Notice that the a and c in $Q(\_, c, \_)$ and $Q(c, \_, a)$ are no longer understood to be constants being arguments of the predicate. They are now part of the name of the predicate. Notice that $|Dom(\gamma)| = 1$ and indeed $Q(\_, c, \_)$ is a binary relation, while $|Dom(\gamma')| = 2$ and $Q(c, \_, a)$ is a unary relation.

As we are never going to use the constants from $\mathbb{D}$ as arguments in atoms over relations from $\Sigma_{\mathbb{D}}$, the above notational convention does not lead to confusion as long as we only talk about atoms over $\Sigma_{\mathbb{D}}$. But atoms over $\Sigma_{\mathbb{D}}$ can easily

be confused with atoms over Σ with constants from $\mathbb{D}$ as arguments. And this confusion is exactly what we want!

If ρ is a total function then Q_ρ is an arity zero predicate. In particular each atom of the database instance $\mathbb{D}$ (over Σ) can be read as a zero arity predicate over $\Sigma_\mathbb{D}$.

We are now going to define $\mathcal{T}_\mathbb{D}$.

For a rule T from $\mathcal{T}$ by a constantification[4] of T we will mean a formula $\sigma(T)$, where σ is a mapping that assigns constants from $\mathbb{D}$ to some of the variables from $Var(T)$, in such a way that for at least one variable $x \in Dom(\sigma)$ this x appears in a marked position in T (we mean here the marking of immortal positions, from the definition of Sticky Datalog$^\exists$). For example $Q(c, y, z) \Rightarrow \exists w\ Q(c, z, w)$ (where $c \in \mathbb{D}$) is a constantification of $Q(x, y, z) \Rightarrow \exists w\ Q(x, z, w)$ if position 1 is marked in Q. Clearly, a constantification of a rule from $\mathcal{T}$ is (or "can be seen as") a rule over $\Sigma_\mathbb{D}$.

Let now theory $\mathcal{T}_\mathbb{D}$ over $\Sigma_\mathbb{D}$ consist of all the facts from $\mathbb{D}$ (which now are, as we mentioned before, zero arity facts) and all the possible constantifications of rules from $\mathcal{T}$. It is not hard to see that $\mathcal{T}_\mathbb{D}$ is also sticky (hint: mark as immortal the same positions as in $\mathcal{T}$), and that the arity of $\mathcal{T}_\mathbb{D}$ is at most $l - 1$.

Let now $\mathcal{C}$ be the set of all atoms of $Chase(\mathcal{T}, \mathbb{D})$ (in the standard notation) and let $\mathcal{C}_1$ be the set of all atoms of $Chase(\mathcal{T}_\mathbb{D}, \emptyset)$ (written using the above notational convention).

The assumption that the pair $\mathbb{D}, \mathcal{T}$ is weakly saturated implies now:

Observation 6 $\mathcal{C} = \mathcal{C}_1$.

For the **proof of the Observation** recall that each atom of $Chase(\mathcal{T}, \mathbb{D})$ is either an atom of $\mathbb{D}$ or it contains some constant from $\mathbb{D}$ in a marked position. This is because (as $\mathbb{D}, \mathcal{T}$ is weakly saturated) the only way for $\mathcal{T}$ to derive any atoms which are not in $\mathbb{D}$ is to use some rule with the sticky join, which requires immortalizing one of the arguments. And, when restricted to atoms which contain some constant from $\mathbb{D}$ in a marked position, the theories $\mathcal{T}$ and $\mathcal{T}_\mathbb{D}$ derive exactly the same atoms. $\qquad\square$

Now let us define $\Phi_\mathbb{D}$ as the disjunction of all possible queries $\sigma(\Phi)$, where σ is a mapping that assigns constants from $\mathbb{D}$ to some of the variables from $Var(\Phi)$ By distributivity, if Φ was a UCQ then also $\Phi_\mathbb{D}$ is a UCQ. And $Chase(\mathcal{T}, \mathbb{D}) \models \Phi$ if and only if $Chase(\mathcal{T}, \mathbb{D}) \models \Phi_\mathbb{D}$, which, by the above Observation, is equivalent to $Chase(\mathcal{T}_\mathbb{D}, \emptyset) \models \Phi_\mathbb{D}$. Since we assumed that the triple $\mathcal{T}$, $\mathbb{D}$, Φ is a counterexample for FC, this implies that $Chase(\mathcal{T}_\mathbb{D}, \emptyset) \not\models \Phi_\mathbb{D}$.

In order to prove that $\mathcal{T}_\mathbb{D}$, $\emptyset$, $\Phi_\mathbb{D}$ is a counterexample for FC we still need to show that for each finite structure M over $\Sigma_\mathbb{D}$ there is $M \models \Phi_\mathbb{D}$. So suppose there was a finite M such that $M \models \mathcal{T}_\mathbb{D}$ and $M \not\models \Phi_\mathbb{D}$. Define a new finite model $M^\mathbb{D}$ as a structure over Σ, containing all the elements of M and all the elements

---

[4]Our constantification trick is not claimed to be any sort of novelty – see for example Constantification technique is by no means new. See for example the comment after Theorem 12.5.2 in [?]

of $\mathbb{D}$, and all the atoms true in M. Of course the atoms true in M were over the signature $\Sigma_{\mathbb{D}}$, but to define $M^{\mathbb{D}}$ we read them as atoms over Σ. It is easy to see that $M^{\mathbb{D}} \models \mathcal{T}$ and $M^{\mathbb{D}} \not\models \Phi$, which is however impossible as the triple $\mathcal{T}$, $\mathbb{D}$, Φ was a counterexample for FC. $\qquad\square$

Proof of Lemma 5: Since Sticky Datalog$^{\exists}$ enjoys the BDD property, we know that there exists a positive FO rewriting of Ψ, which is such a UCQ $\bar{\Psi}$ that for each database instance $\mathcal{M}$ (finite or not) it holds that $\mathcal{M} \models \bar{\Psi}$ if and only if $Chase(\mathcal{M}, \mathcal{T}) \models \Psi$.

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

Since $\mathcal{T}_0$ is joinless, we know, from Theorem 3, that there exists a finite structure $\mathbb{D}'$ such that $\mathbb{D}' \models \mathcal{T}_0, \mathbb{D}$ but $\mathbb{D}' \not\models \bar{\Psi}$. Notice that the pair $\mathcal{T}$, $\mathbb{D}'$ is weakly saturated.

Since $\mathbb{D}' \not\models \bar{\Psi}$, using again the fact that $\bar{\Psi}$ is the FO rewriting of Ψ, we get $Chase(\mathbb{D}', \mathcal{T}) \not\models \Psi$. It remains to be shown that for each finite structure M, if $M \models \mathbb{D}', \mathcal{T}$ then $M \models \Psi$. But, since $\mathbb{D}' \models \mathbb{D}$, the structure M is a model of $\mathbb{D}$ and we assumed that $M \models \Psi$ holds for each finite model of $\mathbb{D}$ and $\mathcal{T}$. $\qquad\square$

4 Assumption a contrario and the structure of the proof

Sections 4 – 15 are devoted to the proof of Theorem 3.

It is an a contrario proof so **we assume now that** there exists a counterexample $\mathcal{T}_C$, $\mathbb{D}_C$, Φ_C for FC, with $\mathcal{T}_C$ being a joinless theory.

In Sections 5 and 6 we explain that it can be assumed w.l.o.g. that the counterexample satisfies some additional assumptions.

The additional assumptions from Section 5 concern trivial simplifications of $\mathcal{T}_C$ and $\mathbb{D}_C$. One of them is that $\mathbb{D}_C$ is the $\emptyset$.

The assumptions from Section 6 however can hardly be seen as simplifications and are one of the main ideas of the whole proof. We define *family patterns* there, and show that it can be assumed w.l.o.g. that $\mathcal{T}_C$ respects the family patterns and that this assumption is a useful tool giving some insight into the structure of Chase.

Then, in Sections 7 – 15 we show that if $\mathcal{T}_C$ and Φ_C satisfy the assumptions from Sections 5 and 6, then the triple $\mathcal{T}_C$, $\emptyset$, Φ_C cannot be a counterexample. We lack language to discuss it yet, so the general architecture of this part of the proof will be described in Section 7.

5 Some trivial simplifications

Nothing deep is going to happen here. We are just cleaning our desk before the real work starts. Our feelings will not be hurt if the reader chooses to read only

Lemma 7, first 3 lines of subsection 5.3, first 10 lines of subsection 5.4 and the very short subsection 5.5, and skip the rest of this Section.

5.1 Empty $\mathbb{D}$

Lemma 7 *There exists a counterexample $\mathcal{T}, \emptyset, \Phi$ for FC.*

Proof: Suppose the active domain of $\mathbb{D}_C$ is $\{d_1, d_2, \ldots d_m\}$ Add a new relation symbol D of arity m to the signature of $\mathcal{T}_C$. Let $\mathcal{T}$ consist of all the rules of $\mathcal{T}_C$, of the rule:

$$\Rightarrow \exists x_1, x_2 \ldots x_m \ D(x_1, x_2, \ldots x_m)$$

and of one Datalog rule:

$$D(x_1, x_2, \ldots x_m) \Rightarrow R(x_{i_1}, x_{i_2} \ldots x_{i_k})$$

for each atom $R(d_{i_1}, d_{i_2} \ldots d_{i_k})$ true in $\mathbb{D}_C$.

Then clearly $\mathcal{T}$, $\emptyset$, Φ is a counterexample for FC. $\qquad\square$

From now on we assume, w.l.o.g. that the triple $\mathcal{T}_C$, $\emptyset$, Φ_C is a counterexample for FC.

5.2 Handy lemma

In this and the next Sections we are going to "normalize" theory $\mathcal{T}_C$. This will be done in several steps. The general idea of each of those steps will be that the predicates of $\mathcal{T}_C$ will be "annotated", so that the name of predicate will carry some additional information. This will lead to a new signature and a new theory, and in each case we will prove a "simplifying lemma" saying that the new theory (together with some new query, and with empty database) is still a counterexample for FC.

In this subsection we present a technical lemma which is a workhorse exploited in the proofs of all the simplifying lemmas in Sections 5 and 6.

Definition 8 *For an atomic formula $Q = R(\bar{t})$ over a signature Σ we define $Q_{|\Sigma}$ to be R.*

In other words $Q_{|\Sigma}$ is the predicate symbol of Q.

Definition 9 *Let œ be a function from the set of all atoms over some signature Σ_A to the set of all atoms over signature Σ. We will say that œ is annotation erasing if:*

(i) $\forall C, C' \ \ C_{|\Sigma} = C'_{|\Sigma} \ \Rightarrow \text{œ}(C)_{|\Sigma_A} = \text{œ}(C')_{|\Sigma_A}$

(ii) $\forall P \in \Sigma_A \ \forall i \ \exists j \ \forall C \ \ C_{|\Sigma_A} = P \ \Rightarrow \ C(j) = \text{œ}(C)(i)$

(iii) $\forall P \in \Sigma_A \ \forall i \ \exists j \ \forall C \ \ C_{|\Sigma_A} = P \ \Rightarrow \ C(i) = \text{œ}(C)(j)$

(iv) æ *is onto.*

For an annotation erasing æ *and any formula (or any structure)* X *by* $æ(X)$ *we mean the formula (structure) being the result of replacing each atom* Q *in* X *by* $æ(Q)$.

See that the above definition requires æ to be "data blind": Condition (i) says that the predicate symbol of the atom being the output of æ must only depend on the predicate symbol of the input. Conditions (ii) and (iii) say that all æ is allowed to do is to copy data to the new atom, without really reading them, without inventing new data and without forgetting anything. It can however change the order of arguments, and possibly create, in the output relation, many columns being a copy of a given column in the input. Notice that the domain of æ is the set of all atoms – both ground atoms and atoms containing variables.

Observation 10 *If* æ *is annotation erasing and* h *is a valuation of variables then the equality* $æ \circ h = h \circ æ$ *holds.* □

Definition 11 *Let* æ *be annotation erasing.*

(i) *The preimage of an atom* C *under* æ *is defined as the disjunction* $æ^{-1}(C) = \bigvee_{æ(B)=C} B$.

(ii) *The preimage of a CQ* $\Phi = \exists \bar{x} \bigwedge_i C_i(\bar{x})$ *under* æ *is defined as the UCQ* $æ^{-1}(\Phi) = \exists \bar{x} \bigwedge_i æ^{-1}(C_i(\bar{x}))$.

(iii) *The preimage of a UCQ* $\Phi = \bigvee_i \Phi_i$ *under* æ *the UCQ is defined as* $æ^{-1}(\Phi) = \bigvee_i æ^{-1}(\Phi_i)$.

Notice that correctness of the above definition follows from condition (iii) of Definition 9 – since æ is not allowed to forget an argument, the preimage-image of an atom is always finite.

Lemma 12 *For a conjunctive query* Φ *and annotation erasing* æ *the query* $æ(æ^{-1}(\Phi))$ *is equivalent to* Φ.

Proof: Because $æ^{-1}$ is applied to each atom separately, it is enough to show that, for an atom C, $æ(æ^{-1}(C))$ is equivalent to C. By definition we have

$$æ(æ^{-1}(C)) = æ(\bigvee_{æ(B)=C} B) = \bigvee_{æ(B)=C} æ(B) = \bigvee_{æ(B)=C} C = C$$

The first equality is a direct application of Definition 11(i). The second equality is a direct application of Definition 11(iii). In the last equality we used the fact, that for each atom C there exists at least one atom B such that $æ(B) = C$. But this follows from Definition 9 (iv).

Definition 13 *For a given theory $\mathcal{T}$ over a signature Σ, and an annotation erasing æ, a theory $\mathcal{T}_A$ over a signature Σ_A is called an æ-annotation of $\mathcal{T}$ if for each rule $\Phi \Rightarrow Q$ in $\mathcal{T}$ (resp. $\Phi \Rightarrow \exists z Q$ in $\mathcal{T}$) and each conjunction of atoms Φ' such that $æ(\Phi') = \Phi$ there exists exactly one rule $\Phi' \Rightarrow Q'$ in $\mathcal{T}_A$ (resp. $\Phi' \Rightarrow \exists z Q'$ in $\mathcal{T}_A$) such that $æ(Q') = Q$.*

The sense of the definition is that the new theory contains annotated versions of the rules of the old theory. There is exactly one new rule for each possible annotation of atoms in the body of an old rule.

Lemma 14 *If $\mathcal{T}_A$ is an æ-annotation of $\mathcal{T}$ then $Chase(\mathcal{T}, \emptyset) = æ(Chase(\mathcal{T}_A, \emptyset))$.*

Proof: By induction one can easily show that $Chase^i(\mathcal{T}, \emptyset) = æ(Chase^i(\mathcal{T}_A, \emptyset))$. The induction step follows directly from Definition 13. Notice that for the $\supseteq$ inclusion the phrase "exactly one" in Definition 13 is crucial. $\square$

Lemma 15 *For an annotation erasing æ and a CQ Φ, if $M \models \Phi$ then $æ(M) \models æ(\Phi)$.*

Proof: Let h be a valuation of $Var(\Phi)$ which shows that $M \models \Phi$. In other words, the image of Φ under h is a substructure of M i.e. $h(\Phi) \subseteq M$. Hence, $æ(h(\Phi)) \subseteq æ(M)$. By Observation 10 we get $æ \circ h = h \circ æ$, so $h(æ(\Phi)) \subseteq æ(M)$. Therefore $æ(M) \models æ(\Phi)$. $\square$

Lemma 16 (Handy Lemma) *If the triple $\mathcal{T}, \emptyset, \Phi$ is a counterexample to FC and $\mathcal{T}_A$ is an æ-annotation of $\mathcal{T}$, then the triple $\mathcal{T}_A, \emptyset, æ^{-1}(\Phi)$ is also a counterexample for FC.*

Proof: Suppose $Chase(\mathcal{T}, \emptyset) \not\models \Phi$. Lemma 14 states that $Chase(\mathcal{T}, \emptyset) = æ(Chase(\mathcal{T}_A, \emptyset))$ and Lemma 12 states that $\Phi = æ(æ^{-1}(\Phi))$, so by contraposition of Lemma 15 we get $Chase(\mathcal{T}_A, \emptyset) \not\models æ^{-1}(\Phi)$.

Let M be an arbitrary finite model of $\mathcal{T}_A$. We need to show that $M \models æ^{-1}(\Phi)$. Because $\mathcal{T}_A$ is an annotation of $\mathcal{T}$, we have that $æ(M)$ is a model of $\mathcal{T}$. Hence, $æ(M) \models \Phi$ – this is because $(\mathcal{T}, \emptyset, \Phi)$ is a counterexample to FC, so Φ must be satisfied in each finite model of $\mathcal{T}$.

Let $\Phi_0 = \exists \bar{x}\ \Psi(\bar{x})$ be a disjunct of Φ which is true in $æ(M)$. There exists h - a valuation of the variables $\bar{x}$ - such that $h(\Psi(\bar{x})) \subseteq æ(M)$. This inclusion implies that there exists a subset M_0 of M (we see M as a set of atoms now) such that $h(\Psi(\bar{x})) = æ(M_0)$.

Now we claim that $M_0 \models æ^{-1}(\Phi)$. Of course when we prove this claim then the proof of Handy Lemma will be finished. But, by definition of preimage we have that $æ^{-1}(\Phi_0)$ logically implies $æ^{-1}(\Phi)$, so it will be enough to notice that $M_0 \models æ^{-1}(\Phi_0)$. Again using definition of preimage (and distributivity) we see that $æ^{-1}(\Phi_0)$ is a disjunction of all possible CQs $\exists \bar{x}\Psi_0$ such that $æ(\Psi_0) = \Psi$. And M_0 is a homomorphic image of one such Ψ_0. $\square$

5.3 Strongly joinless theories

We will call a joinless theory $\mathcal{T}$ *strongly joinless* if heads of all the rules of $\mathcal{T}$ are joinless, which means that if T is a rule from $\mathcal{T}$ then each variable occurs in the head of T at most once.

Lemma 17 *There exists a counterexample $\mathcal{T}_A$, $\emptyset$, Φ_A for FC with strongly joinless $\mathcal{T}_A$.*

Proof :
Annotations. For a natural number k by a k-annotation we will mean any set of equalities of the form $i = j$ for $1 \leq i, j \leq k$ which is closed under logical consequence. For k-annotations α_1 and α_2 by $\alpha_1 \wedge \alpha_2$ we will mean the smallest annotation containing α_1 and α_2. For a k-annotation α by index of α we will mean the number of equivalence classes that α naturally splits $\{1, 2, \ldots k\}$ into.

The signature of $\mathcal{T}_A$. Let Σ be the signature of $\mathcal{T}_C$. The signature Σ_A of $\mathcal{T}_A$ will consist of one predicate R_α for each k-ary predicate $R \in \Sigma$ and for each k-annotation α. The arity of R_α equals to the index of α. The following notational convention will apply: Atoms of R_α will be written as $R_\alpha(X_1, X_2, \ldots X_k)$ with X_i being equal to X_j whenever $i = j$ is in α. For example if $k = 3$ then $R_{2=3}(a, b, b)$ is an atom of the binary relation $R_{2=3}$ while the expression $R_{2=3}(x, x, y)$ is a ($\heartsuit$) syntax error .

Theory $\mathcal{T}_A$. Now we are ready to define theory $\mathcal{T}_A$. For each (joinless) rule of $\mathcal{T}_C$:

$T \hspace{4em} R(x_1, x_2 \ldots x_k), P(y_1, y_2 \ldots y_m) \Rightarrow \exists z\, Q(v_1, v_2, \ldots v_l)$

where each of v_i is either z or one of the x_i or one of the y_i, and for each k-annotation α and each m-annotation β, theory $\mathcal{T}_A$ will contain the rule:

$T_{\alpha,\beta} \hspace{2em} R_\alpha(X_1, X_2 \ldots X_k), P_\beta(Y_1, Y_2 \ldots Y_m) \Rightarrow \exists z\, Q_\gamma(V_1, V_2, \ldots V_l)$

where:

$-X_i = X_j$ if $i = j$ is in α and $Y_i = Y_j$ if $i = j$ is in β;
$- V_i = X_j$ if $v_i = x_j$, $V_i = Y_j$ if $v_i = y_j$, and $V_i = V_j$ if $v_i = v_j$;
$- i = j$ is in γ if and only if $V_i = V_j$.

Notice that the rule $T_{\alpha,\beta}$ is strongly joinless – arity of Q_γ is equal to the index of γ and is equal to the number of different variables in the atom $Q_\gamma(V_1, V_2, \ldots V_l)$.

To keep the notation as simple as possible we defined $T_{\alpha,\beta}$ for a TGD with two atoms in the body. But of course the same must be done for all rules of $\mathcal{T}_C$, including Datalog rules.

Annotation erasing. Now æ is defined as an operation that maps atoms over signature Σ_A to atoms over Σ, in the most natural way one could imagine – by erasing the annotation.

It is easy to notice that æ is indeed an annotation erasing, as defined by Definition 9, and that $\mathcal{T}_A$ satisfies the assumptions of Handy Lemma. So, we can use Handy Lemma to finish the proof of of Lemma 18. $\hspace{2em}\square$

From now on we will assume, w.l.o.g. that the triple $\mathcal{T}_C, \emptyset, \Phi_C$ is a strongly joinless counterexample for FC.

5.4 Almost clean theories

We will call a strongly joinless theory $\mathcal{T}$ *almost clean* if each rule from $\mathcal{T}$ is either a Datalog rule of the form:

(♣1) $Q(\bar{x}) \Rightarrow Q'(\bar{x}^i)$, *where by $\bar{x}^i$ we mean the tuple $\bar{x}$ with i-th element removed*

or a TGD of the form:

(♣2) $Q_0(\bar{x}_0) \wedge Q_1(\bar{x}_1) \Rightarrow \exists y \, Q(y, \bar{x}_0, \bar{x}_1)$

Condition (♣ 2) does not rule out Q_1 to be empty, so in particular a rule of the form $Q_0(\bar{x}_0) \Rightarrow \exists y \, Q(y, \bar{x}_0)$ is also allowed. The important part, of both conditions, is that the variables in the head occur in exactly the same order as the variables in the body.

Lemma 18 *There exists a counterexample $\mathcal{T}_A$, $\emptyset$, Φ_A for FC with $\mathcal{T}_A$ being almost clean.*

Proof: It is trivial to see that $\mathcal{T}_C, \emptyset, \Phi_C$ can w.l.o.g. be assumed to contain only TGDs with at most two atoms in the body (the left hand side) and of Datalog rules that project exactly one element. The slightly more non-trivial part is to show that the ordering condition can also be satisfied.
The signature of $\mathcal{T}_A$. Let Σ be the signature of $\mathcal{T}_C$. The signature Σ_A of $\mathcal{T}_A$ will consist of one predicate R_α for each k-ary predicate $R \in \Sigma$ and for each k-permutation $\pi : \{1, \ldots k\} \to \{1, \ldots k\}$. The arity of R_π is equal to the arity of R.
Annotation erasing æ is defined as $\text{æ}(Q_\pi(x_1, x_2, \ldots x_k) = Q(\pi(\bar{x}))$, where $\pi(\bar{x}) = (x_{\pi(1)}, \ldots x_{\pi(k)})$. Clearly, this operation satisfies the requirements of Definition 9.
Theory $\mathcal{T}_A$. For each Datalog rule T of theory $\mathcal{T}_C$, of the form $Q(\bar{x}) \Rightarrow Q'(\pi_T(\bar{x}^i))$, with Q of some arity k, and for each k-permutation π let there be a rule T_π in $\mathcal{T}_C$, of the form $Q_\pi(\bar{x}) \Rightarrow Q'_{\pi'}(\bar{x}^i)$, where π' is the unique permutation such that T equals, up to the renaming of variables, to $\text{æ}(T_\pi)$. In similar manner we construct one rule for each TGD in $\mathcal{T}_C$ and each possible annotation of the predicates in the body of this rule. Then use Handy Lemma to finish the proof. $\qquad\square$

5.5 Clean theories and clean counterexamples

An almost clean theory $\mathcal{T}$ will be called *clean* if:

- the signature of $\mathcal{T}$ is a union of two disjoint sets: parenthood predicates (or PPs), occurring in the heads of rules of the form ($\clubsuit$ 2), and projection predicates, occurring in the heads of rules of the form ($\clubsuit$ 1);

- 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 $\mathcal{T}$.

We will call a UCQ (or a CQ) Φ *clean* if only the parenthood predicates appear in Φ. A triple $\mathcal{T}, \emptyset, \Phi$ is *clean* if $\mathcal{T}$ and Φ are clean. Using Lemma 18 it is very easy to show that:

Lemma 19 *There exists a clean counterexample $\mathcal{T}$, $\emptyset$, Φ for FC.* $\qquad\square$

6 On the importance of family values

Let $\mathcal{T}$ be a clean theory, as defined in Section 5. From now on we will always have $\mathbb{D} = \emptyset$. Since the context is clear we will simply write $Chase$ instead of $Chase(\mathcal{T}, \emptyset)$.

In this Section we will imagine $Chase$ as the humankind. Generations after generations of elements are being born (by the TGDs) and then projected out (by the Datalog rules). And atoms are like families, as you are going to see. Let l be the maximal predicate arity in the signature of $\mathcal{T}$.

6.1 A fairy tale

In the next subsection we define **family patterns**. This is a crucial tool in our analysis of the structure of $Chase$, but a complicated one. So before we present the technical definitions, the reader is invited to join us for an informal visit to a planet far far away, where very strict rules apply concerning family dinners.

First of all, the participants of a family dinner must always be all the ancestors of some person A (who may be alive or dead at the moment) who are currently alive. The word "ancestor" is understood in the reflexive sense, which means that A must also participate, if she is still alive. A group of people that are allowed to dine together will be called "family".

Due to some curse no family on this planet can ever have more than l members. Notice that the families, as we defined them, are not pairwise disjoint. Adam, Eve and Cain were a family. Adam, Eve and Abel were a family, and after Abel's death (but not before) Adam and Eve were still a family. But there was never a family including both Cain and Abel.

During a dinner all the participants sit behind a long table, always in the same order. When someone is sadly projected out, then the surviving family members shift (so that there is no empty space left), but the order remains the same.

Two families can sometimes have a baby together. One peculiarity is that all the ancestors of A who were alive when A was born are considered parents

of A. When two families have a baby A together then, according to the above rule, they are allowed to dine together. Here is how they are seated during such dinner: A sits first from the left, then all the people from the mother family, in the order they used during their dinners, and then all the people from the father family, also in the order they used during their dinners.

Now imagine being the Police who enforce the rules. You come and just see a row of people behind the table, with no apparent structure at all. This is why a rule was introduced requiring that each family posts information about their *family ordering* on their web page. Family ordering is the binary relation (actually a partial order), on elements $\{1, 2, \ldots k\}$ (where k is the cardinality of the family) containing all the (descendant, ancestor) pairs. Notice that people are identified here with the places they occupy when dining. Since $k \leq l$ there are only finitely many possible family orderings.

Another aspect of the family life that is strictly codified is the way people address their parents (ancestors). When a newborn A dines, for the first time, with all her parents (i.e. living ancestors) she learns to call the person B sitting on the chair i as simply i. Then, as time goes by, some of A's ancestors are projected out, new people are being born, but A and B may still dine together, in different configurations. And A will always address B as i.

The function that maps each (descendant, ancestor) pair (C,D) of a family to a number (not greater than l) used by C to address D is also posted on the family web page. Together with the family ordering they form the **family pattern**. Notice that there are only finitely many possible family patterns.

Remark about incest. There is nothing in the rules of the planet that would forbid non-disjoint families to have a baby together. Actually there is nothing in the rules that would require that mother family and father family are different (which some humans may see as strange). And when two non-disjoint families have a baby then there is a person who plays more than one role – he is a member of the mother family and of the father family at the same time. Such person has two (or more) chairs behind the family table, and the way he is addressed by his descendants depends on the chair he currently sits on. Notice that the family ordering is defined as a partial order of chairs rather than people and it is blind to the fact that two chairs are occupied by the same person and thus it is always a tree-like ordering – each two descendants of a given element are always comparable.

Back to the example. As we said, Cain, Eve and Adam were a family. The family pattern was F, δ where the ordering F consisted of two pairs: $2 <_F 1$ and $3 <_F 1$ and δ was defined as $\delta(1, 2) = 2$ and $\delta(1, 3) = 3$.

Also Awan, Eve and Adam were a family. And the family pattern was the same F, δ as before.

Then Awan and Cain had a child together, named Enoch. When Enoch was born Adam and Eve were still alive, so the five people were one family. But there were seven chairs behind the table they needed to dine together. First chair for Enoch, 2nd for Awan, 3rd for Eve, 4th for Adam, 5th for Cain, 6th again for Eve and 7th again for Adam. The new family pattern was G, γ where

G consisted of the pairs $i <_G 1$, for each $2 \leq i \leq 7$ and of the pairs $3 <_G 2$, $4 <_G 2$, $6 <_G 5$ and $7 <_G 5$. What concerns γ, we had $\gamma(1,i) = i$, for each $2 \leq i \leq 7$, and – for example – $\gamma(5,7) = \delta(1,3) = 3$, as the way Cain was calling Adam did not change after Enoch was born.

6.2   Family patterns and how they change over time

Let us now formalize our fairy tale:

Definition 20 *By a (k-ary) family ordering we mean any tree-like partial order, whose set of vertices is $\{1, 2, \ldots k\}$ where $k \leq l$. By a tree-like partial order we mean that each two elements greater than any given one are comparable. If a family ordering is a tree then 1 is the root (the greatest element) of this tree.*

If a family ordering is a tree, the root of the tree is the youngest family member[5].

But – as we explained above – the 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 21 *A (k-ary) family pattern is a pair F, δ, where F is a (k-ary) 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 ordering relation on F (i is an ancestor of j).*

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

Now imagine there had been a family of k people with the family pattern F, δ. But then, at some point of time, the person who sat on chair i was projected out. The surviving family members still dine together, and their new family pattern is of course a function of F, δ and of i. Call the new pattern[6] $\mathrm{project}_i(F, \delta)$ We will never really need to compute $\mathrm{project}_i(F, \delta)$, but maybe it is helpful to see that it is indeed possible:

Observation 22 *Suppose $\mathrm{project}_i(F, \delta) = G, \gamma$. For a natural number $1 \leq j \leq k$ let $g(j) = j$ if $j < i$ and $g(j) = j - 1$ otherwise. Then $j <_F j'$ if and only if $g(j) <_G g(j')$ and, whenever $j <_F j'$ then $\delta(j, j') = \gamma(g(j), g(j'))$*

In a similar manner we can imagine two families, one consisting of k people, with the family pattern F, δ, and another one with k' people, and with the family pattern F', δ', having a baby together. Then, together with the baby, they form a new family, of $1 + k + k'$ people, and the family pattern of the new family is a function of F, δ and F', δ'. Call the new pattern $\mathrm{baby}(F, \delta; F', \delta')$. Again, this is not really needed but we can compute $\mathrm{baby}(F, \delta; F', \delta')$:

---

[5] Mnemonic hint: the one is smaller whose date of birth is a smaller number.

[6] If you are not happy with this definition, then treat Observation 22 as a definition. The same applies to Observation 23.

Observation 23 *Suppose* $baby(F, \delta; F', \delta') = G, \gamma$. *Then:*

(a) $i <_G j \Leftrightarrow (j = 1 \wedge i > 1) \vee$
$\qquad\qquad\quad (i - 1 <_F j - 1 \quad \wedge \quad 1 < i, j \le k + 1) \vee$
$\qquad\qquad\quad (i - k - 1 <_{F'} j - k - 1 \quad \wedge \quad k + 1 < i, j \le k + k' + 1)$

(b) If $j = 1$ *and* $1 < i \le k + k' + 1$ *then* $\gamma(j, i) = i$.
$\qquad$ *If* $1 < j, i \le k + 1$ *then* $\gamma(i, j) = \delta(i - 1, j - 1)$.
$\qquad$ *If* $k + 1 < j, i \le k + k' + 1$ *then* $\gamma(i, j) = \delta'(i - k - 1, j - k - 1)$.

Condition (a) says that the birth of the new child does not change the ancestor relation in the family, except from the fact that each of the members of the two families is now also this child's ancestor. The meaning of condition (b) is that the newborn child learns how to address his ancestors: it addresses them by their positions at the family table, as it sees it at the moment of its birth. The child's birth does not change the way his ancestors are addressing each other.

6.3   Back to the Chase

Definition 24 *A clean theory* $\mathcal{T}$ *respects family patterns* **if:**

1. *Each relation* Q *of arity* k *in the signature of* $\mathcal{T}$ *contains, as a part of its name (as a subscript) a* k*-ary family pattern.*

2. *If* $R_{F, \delta}(\bar{x}) \Rightarrow P_{G, \gamma}(\bar{x}^i)$ *is a Datalog rule of* $\mathcal{T}$ *then* $G, \gamma = project_i(F, \delta)$ *(the meaning of* $\bar{x}^i$ *is as defined in subsection 5.4).*

3. *If* $R_{F, \delta}(\bar{x}) \wedge R'_{F', \delta'}(\bar{x}') \Rightarrow \exists y \, P_{G, \gamma}(y, \bar{x}, \bar{x}')$ *is a TGD of* $\mathcal{T}$ *then we have* $G, \gamma = baby(F, \delta; F', \delta')$

Lemma 25 *There exists a clean counterexample* $\mathcal{T}_A$, $\emptyset$, Φ_A, *with* $\mathcal{T}_A$ *respecting family patterns.*

Proof: Let $\mathcal{T}_C$, $\emptyset$, Φ_C be any clean counterexample, over some signature Σ. Let Σ_A consist of one arity k predicate $Q_{F, \delta}$ for each arity k predicate Q in Σ and each k-ary family pattern F, δ.

Now for each Datalog rule $R(\bar{x}) \Rightarrow P(\bar{x}^i)$ in $\mathcal{T}_C$ and for each family pattern F, δ of arity equal to the arity of R, let $R_{F, \delta}(\bar{x}) \Rightarrow P_{G, \gamma}(\bar{x}^i)$ be a rule in $\mathcal{T}_C$, where $G, \gamma = project_i(F, \delta)$.

Similarly, for each TGD $R(\bar{x}) \wedge R'(\bar{x}') \Rightarrow \exists y \, P(y, \bar{x}, \bar{x}')$ in $\mathcal{T}_C$ and for each pair of family patterns F, δ, F', δ', of arities equal to the arities of R, R' respectively, let $R_{F, \delta}(\bar{x}) \wedge R'_{F', \delta'}(\bar{x}') \Rightarrow \exists y \, P_{G, \gamma}(y, \bar{x}, \bar{x}')$ be a rule in $\mathcal{T}_C$, where $G, \gamma = baby(F, \delta; F', \delta')$. Define the function æ as – literally – removing the annotations. Use Handy Lemma to finish the proof. $\qquad \square$

From now on we assume that
$\mathcal{T}$ is a fixed clean theory which respects family patterns.

Before we end this Section let us study some properties of $Chase(\mathcal{T}, \emptyset)$. The following Lemma is an obvious consequence of the assumption that $\mathcal{T}$ is clean and of freeness of the Chase:

Lemma 26 *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.*

Notice that we use the word "parents" (here and always in the future) to denote all the ancestors of a who were present when a was born. So it it is perfectly normal in our scenario that a and b are parents of c while a is a parent of b.

Definition 27 *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 pair of elements $A(i)$ and $B(i)$ will be called respective parents *of the pair of elements a and b. For tuples $a_1, a_2, \dots a_s$ and $b_1, b_2, \dots b_s$ by $a_1, a_2, \dots a_s \equiv_0 b_1, b_2, \dots b_s$ we mean that $a_i \equiv_0 b_i$ for all $1 \le i \le s$.*

Since the family pattern is part of the name of the predicate, when we say "the same predicate" in Definition 27 we of course mean that the family patterns are also equal.

The next lemma says, using our running metaphor, that the person an element a of $Chase$ calls its granny does not change during its lifetime. Moreover, the way a's father calls a's granny also remains unchanged:

Lemma 28 *Suppose $Chase \models B, C$, for $B = Q_{F, \delta}(\bar{b})$ and $C = PP_{G, \gamma}(a, \bar{a})$. Suppose also that $a = B(i)$ and $j, j' <_F i$. Then:*

1. *$B(j)$ is a parent of a;*

2. *$B(j) = C(\delta(i, j))$;*

3. *$j <_F j'$ if and only if $\delta(i, j) <_G \delta(i, j')$;*

4. *if $j <_F j'$ then $\delta(j', j) = \gamma(\delta(i, j'), \delta(i, j))$.*

The proof of the lemma is easy induction on the structure of $Chase$, and we leave it for the reader as an exercise. Actually, the only possibly non-trivial part of this exercise is to remember what the notations mean. So let us come to your help. The assumption that $a = B(i)$ means that a is somewhere (position i'th) in atom B. The assumption that $j <_F i$ means that in family B the element

in position j, call it c, is (according to the family pattern of this family) an ancestor of a. Now trace the history (or "derivation in $Chase$") of the family (or "of atom") B back to a's birth, and notice that each step of the derivation preserves the properties claimed by the Lemma. The last is because we assume that $\mathcal{T}$ respects the family patterns.

Now we have something slightly more complicated. The following lemma, which will be critically important in Section 15, is where the power of family patterns is seen:

Definition 29 *For a family ordering F and a set $\mathcal{I}$ of positions in F we define the set $PY(\mathcal{I})$ of positions in F as $\bigcap_{i \in \mathcal{I}} \{j \in F : \neg(j \leq_F i)\}$.*

$PY(\mathcal{I})$ (which reads "possibly younger") is exactly the set of family members who potentially can be younger than each of the elements of $\mathcal{I}$. Of course the set PY depends on the ordering F, but we do not make it explicit in the notation as the context is always clear.

Lemma 30 (About the Future) *Let $Chase \models A$ for some $A = PP_{F,\delta}(a, \bar{a})$. Suppose $\mathcal{I} = \{i_1, i_2, \ldots i_s\}$ is a set of pairwise $<_F$-incomparable 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:*

(i) *$Chase \models C$;*

(ii) *$d_1, d_2, \ldots d_s$ equal $C(i_1), C(i_2), \ldots C(i_s)$ respectively;*

(iii) *if $j \in PY(\mathcal{I})$ then $A(j) \equiv_0 C(j)$;*

Lemma 30 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. If $b_1, b_2, \ldots b_s$ and $d_1, d_2, \ldots d_s$ are 0-equivalent tuples of elements and $b_1, b_2, \ldots b_s$ appear in some atom A in $Chase$ (at independent positions) then there exists an atom C, somewhere in $Chase$, which not only has $d_1, d_2, \ldots d_s$ in the same positions, but also is as similar to A as one could dream of: everything that happens in the future of some b_i in A is 0-equivalent to the respective future of the respective d_i in C.

Before we prove Lemma 30, as one more exercise let us show that it follows easily from Lemma 28 that if $j \notin PY(\mathcal{I})$ then the elements $A(j)$ and $C(j)$ are respective parents of some b_k and d_k:

Lemma 31 *If $i_k \in \mathcal{I}$ and $j <_F i_k$ then $A(j)$ and $C(j)$ are respective parents of b_k and d_k (where the notations are like in Lemma 30).* □

Proof of Lemma 31. Let A' be the parenthood atom of b_k and let C' be the parenthood atom of d_k. Of course A' and C' are atoms of the same predicate, as

we assumed that $b_k \equiv_0 d_k$. Then, by Lemma 28.2. we have $A(j) = A'(\delta(i_k, j))$ and $C(j) = C'(\delta(i_k, j))$, where δ is as in Lemma 30. □

Remember that the fact that $b \equiv_0 d$ does not imply that the respective parents of b and d are 0-equivalent.

Proof of Lemma 30. The intuition is that we will trace the genealogy of atom A, as deep to the past as we see families containing one of the b_i. If we go many enough generations back in time we will see, for each i, the family in which b_i was born. Since we assume that $b_1, b_2, \ldots b_s \equiv_0 d_1, d_2, \ldots d_s$ we can find, for each i, another atom, somewhere in *Chase*, of the same predicate (including family pattern), which gave birth to d_i. Now we can tell the families where d_i were born: "mimic the behavior of the parenthood atoms of b_i". And they can do it, because the rules are joinless, which implies that all atoms of the same predicate are equally able to participate in derivations.

To be more precise, we consider (a fragment of) the derivation tree of the atom A in *Chase*, which we will call $\mathcal{D}$. Verticies of $\mathcal{D}$ will be atoms of *Chase*, with A being the root. $\mathcal{D}$ is defined by induction, together with an equivalence relation sv (as "same variable") on the set of all positions in the atoms of $\mathcal{D}$, and with the set of painted positions:

- Atom A is the root of $\mathcal{D}$ (and thus an inner node of $\mathcal{D}$). Positions $i_1, i_2, \ldots i_s$ in A are *painted*.

- Suppose an atom $B = Q_{G,\gamma}(e, \bar{e})$ is a node of $\mathcal{D}$ with some non-root position painted[7]. Suppose $B' = Q'_{G',\gamma'}(\bar{e}_1)$ and $B'' = Q''_{G'',\gamma''}(\bar{e}_2)$ 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 $X' = Q'_{G',\gamma'}(\bar{x}_1)$, $X'' = Q''_{G'',\gamma''}(\bar{x}_2)$ and $X = Q_{G,\gamma}(x, \bar{x})$. Then B' and B'' are nodes of $\mathcal{D}$, and children of B.

  If $X(i) = X'(j)$ (or $X''(j)$), which means that the variables on position i in X and on position j in X' (or X'') are equal, then the pair of positions i in B and j in B' (or B'') is added to the relation sv (and sv is always extended to be an equivalence). A position in B' or B'' is painted if it is sv with some previously painted position.

  The case when B was derived by a projection rule $X' \Rightarrow X$ is handled analogously[8].

- A node of $\mathcal{D}$ with no painted positions is a leaf, called an unpainted leaf. A node which is a PP atom, and whose only painted position is its root is a leaf of $\mathcal{D}$, called a painted leaf. All other nodes of $\mathcal{D}$ are inner nodes.

---

[7] Recall that the root of a parenthood atom is its position 1 – the root of the family ordering, which is a tree. An atom which is not a PP-atom may or may not not be a tree and thus it is possible for it to contain only non-root positions.

[8] Notice that B while for B being PP-atoms we can always identify unique pair B', B'' in *Chase* that led to B in one derivation step, this is not always the case if B is a result of a projection. In such case we take B' to be any atom of *Chase* which led to creation of B.

The idea here is that we trace the derivation of A back to the parenthood 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 in meaningful positions – the positions that lead to i's in A, but also in non-meaningful ones, not connected, by the rules of $\mathcal{T}$, to any of the i's in A.

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

- If B is an unpainted leaf of $\mathcal{D}$ then $h(B) = B$ is the respective leaf of $\mathcal{D}'$.

- If B is a painted leaf of $\mathcal{D}$, which means that B is the parenthood atom of some b_i, and if E is the parenthood atom of d_i then $h(B) = E$ is the respective leaf of $\mathcal{D}'$ (see Observations 32–34 if you feel an argument is needed here).

- If B is an inner node of $\mathcal{D}$, being a result of applying some rule T from $\mathcal{T}$ to atoms B' and B'' (or just to B', if T was a projection) and if we already know $h(B')$ and $h(B'')$ then let $h(B)$ be the result of applying the rule T to the atoms $h(B')$ and $h(B'')$.

Clearly, $\mathcal{D}'$ is also a part of $Chase$ and $h(B)$ is always an atom of the same predicate as B. Notice however that if $\mathcal{T}$ was not joinless, the last step of the construction would not always be possible in $Chase$.

Now, the atom $h(A)$ in the root of $\mathcal{D}'$ is going to be the C from the Lemma. What remains to be proved is that it indeed satisfies conditions (ii) and (iii) from the Lemma.

It easily follows from the construction that:

Observation 32 *If B, B' are atoms of $\mathcal{D}$ and the pair of positions i in B and j in B' is in sv then $B(i) = B'(j)$.*

Notice also that, since $\mathcal{T}$ is joinless, which means that a variable in the head of a rule occurs in at most one atom in the body of this rule, we have:

Observation 33 *For an atom B in $\mathcal{D}$ and position i in B, the set of nodes of $\mathcal{D}$ which contain some position being sv to position i in B is a directed path in $\mathcal{D}$.*

Since, for a B in $\mathcal{D}$ we add children of B to $\mathcal{D}$ as long as B has some non-root position painted, it follows from the construction that:

Observation 34 *For each position $i_j \in \mathcal{I}$ there is exactly one leaf B of $\mathcal{D}$ such that root of B and position i_j in atom A are sv .*

Condition (ii). First of all notice that, as $\mathcal{D}'$ is isomorphic to $\mathcal{D}$, the relation sv can be in a natural way seen as a relation on positions in $\mathcal{D}'$ (positions i in $h(B)$ and j in $h(B')$ are sv iff positions i in B and j in B' are), and that Observations 32– 34 still hold true (with A replaced by C in Observation 34).

For $i_j \in \mathcal{I}$ consider the leaf B of $\mathcal{D}$ such that root of B and position i_j in atom A are sv . By Observation 32 we have $B(1) = A(i_j)$. Then, by construction of $\mathcal{D}'$ we have $(h(B))(1) = d_j$, and, since Observation 32 remains true in $\mathcal{D}'$, we have $C(i_j) = d_j$, as needed.

Condition (iii). Let j be a position in $PY(\mathcal{I})$ in A. Observation 33 says that atoms of $\mathcal{D}$ which contain some position being sv to position j in A form a directed path in $\mathcal{D}$. A is one end of this path. Let B be the other end. There are two possibilities: either B is unpainted leaf of $\mathcal{D}$ or it is an inner node.

If B is an inner node, then B is the parenthood atom of $A(j)$. Then $h(B)$ is also the parenthood atom of some element and $B(1) \equiv_0 (h(B))(1)$. But we know that position 1 in $h(B)$ and position j in C are sv , so we have $(h(B))(1) = C(j)$, what needed to be proved.

If B is unpainted leaf of $\mathcal{D}$ then let i be the position in B which is sv to position j in A. Using the definition of $h(B)$ for the case of unpainted leaves we have $A(j) = B(i) = (h(B))(i) = C(j)$. $\qquad\square$

7 General scheme of the proof. The first little trick

In the following Sections 7– 15 we show that a clean triple $\mathcal{T}$, $\emptyset$, Ψ, where $\mathcal{T}$ respects the family patterns, is never a counterexample for FC.

We will construct, for our theory $\mathcal{T}$, an infinite sequence of finite structures $\{M_n\}_{n \in \mathbb{N}}$, which will "converge" to $Chase$. The following property will be satisfied:

Property 35 *(i) $M_n \models \mathcal{T}$ for each $n \in \mathbb{N}$.*

(ii) For each UCQ Ψ and each $n \in \mathbb{N}$ if $M_n \not\models \Psi$ then $M_{n+1} \not\models \Psi$.

Assume – till the end of this section – that a sequence $\{M_n\}_{n \in \mathbb{N}}$, satisfying Property 35 (i), (ii) is constructed. Then:

Definition 36 *A formula Φ will be called M-true if $M_n \models \Phi$ for each $n \in \mathbb{N}$.*

Lemma 37 (First Little Trick) *If Φ is an M-true UCQ then there exists a disjunct of Φ which is M-true.*

Proof: By Property 35 (ii) all 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. $\qquad\square$

The rest of the paper is organized as follows: In Section 9 the sequence M_n is defined. In Section 10 we present our Second Little Trick, which not only is the main engine of the proof of The Normal Form Lemma but also the main technical idea of the whole paper.

In the very short Section 11 a trivial case of cycled queries (whatever it means) is considered. In Section 12 we finally define a normal form of a conjunctive query and explain the main idea of the proof of:

Lemma 38 (The Normal Form Lemma) *For each clean M-true CQ ϕ there exist a clean CQ β in the normal form such that:*
(∗) *β is M-true and*
(∗∗) *$Chase \models (\beta \Rightarrow \phi)$.*

In Sections 13 and 14 we continue the proof of the Normal Form Lemma. Finally, in Section 15 we prove:

Lemma 39 (The Lifting Lemma) *If a clean CQ β is in the normal form and $M_0 \models \beta$ then $Chase \models \beta$.*

Assuming existence of a sequence $\{M_n\}_{n \in \mathbb{N}}$, satisfying Property 35, and assuming Lemmas 38 and 39 we can now present:

The main body of the proof of Theorem 3:

Suppose a clean triple $\mathcal{T}$, $\emptyset$, Ψ, where $\mathcal{T}$ respects the family patterns, is a counterexample for FC. This means there is no finite model satisfying $\mathcal{T}$ and not satisfying Ψ, so in particular Ψ is M-true. Let ϕ be an M-true disjunct of Ψ (which must exist due to the First Little Trick). Since $\mathcal{T}$, $\emptyset$, Ψ is a counterexample for FC we know that $Chase(\mathcal{T}, \emptyset) = Chase \not\models \Psi$, so in particular $Chase \not\models \phi$. Let β be the normal form of ϕ as described in the Normal Form Lemma. We know from (*) that β is M-true, so in particular $M_0 \models \beta$. The Lifting Lemma tells us that $Chase \models \beta$. But, by (**), we have that $Chase \models (\beta \Rightarrow \phi)$, so also $Chase \models \phi$. Contradiction. $\qquad\square$

The proof above was a high-level one. We neither bothered to know what the structures M_n are, nor what the normal form could actually be. It was enough for us to know that they are tailored for Lemmas 38 and 39 to be true. **The real work begins now.**

8 Aside: a philosophical remark

We feel we need to address an issue raised by one of the reviewers of our LICS submission, and explain the connections between our structures M_n and the finite structures from [Ros06] and [BGO10].

The general idea both here and in [Ros06] and [BGO10] is that a finite structure constructed by identification of terms which have the same top n levels, for some natural n, can be used to approximate the Herbrand universe with respect to the properties of elements which only depend on the recent history of those elements.

This general idea is natural, by no means new, and it was reinvented many times by many authors. We know it from [Mar95] (and its journal version [MP03]), but it is already present, in some sense, in [JK84].

The devil is not in the general idea here, but in the details. The procedure in [Ros06] and [BGO10] is the following:

- start from some database instance $\mathbb{D}$;

- give a name to each Skolem function resulting from Skolemization of the TGDs in $\mathcal{T}$;

- fix n and a constant c_n to denote the branch stubs;

- consider the (finite) universe U_n of the terms of depth up to n over the defined signature (the constants are c_n and the constants from $\mathbb{D}$);

- define $\mathcal{T}'$ by replacing each TGD in $\mathcal{T}$ by a PROLOG rule in the natural way; run the program $\mathcal{T}'$ on U_n to get a model $\bar{U}_n$ of $\mathcal{T}$.

In this way, the resulting structure is always a model of $\mathcal{T}$. But the cost is that the $\bar{U}_n$ are very complicated and hard to analyze. In particular it is not even clear which atoms are true in $\bar{U}_n$, so it is very hard to lift a valuation satisfying a query in $\bar{U}_n$ to $Chase(\mathbb{D}, \mathcal{T})$. This is – if we understand it correctly – the main source of complications in [Ros06] and [BGO10].

Our way is very much different. We:

- **first** run $\mathcal{T}$ on $\mathbb{D}$ to get $Chase(\mathbb{D}, \mathcal{T})$;

- **then** construct, for given n, the structure M_n identifying elements of $Chase(\mathbb{D}, \mathcal{T})$ which have the same history, up to level n.

In consequence, not for each $\mathcal{T}$ we can be sure that our structure M_n is a model of $\mathcal{T}$. But there are two important properties that we get for free, which are not shared by the structures $\bar{U}_n$ above:

- M_n is a homomorphic image of M_{n+1}. This is a crucial property for the normalization step that we call *second little trick*.

- The only way of being an atom in M_n is to be an atom in $Chase(\mathbb{D}, \mathcal{T})$ before. This makes lifting easy.

The two above properties make us think that the structures M_n do not just *approximate* the Chase. They *converge* to it.

9 The canonical models M_n

Proving that a theory is FC is about building finite models. And finally, in this section we build them. Actually we define an infinite sequence of finite models M_n, which will "converge" to $Chase$.

Definition 40 *By 1-history of an element $a \in Chase$ (denoted as $H^1(a)$) we mean the set consisting all the parents of a. By the $n+1$-history of a we mean the set $H^{n+1}(a) = \bigcup_{b \in H^1(a)} H^n(b) \cup \{b\}$.*

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

Definition 41 *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.*

Definition 42 *For two elements a, b of $Chase$ and for $n \in \mathbb{N}$ by $a \simeq_0^n b$ we mean that $a \equiv_0 b$ and a and b have the same n-color. By $a \simeq_{k+1}^n b$ we mean that $a \simeq_0^n b$ and that $a' \simeq_k^n b'$ for each pair a', b' of respective parents of a, b.*

To see what $a \simeq_n^n b$ means imagine that each element of $Chase$ keeps the record of its family history. It knows its n-color and the name of the predicate it was born with, the n-colors of its parents and the names of the predicates its parents were born with[9]. And so on, n generations back. Equivalence $\simeq_n^n$ identifies elements of $Chase$ if and only if the records they keep are equal.

Definition 43 *For a natural $n \geq 1$ we define two elements $a, b \in Chase$ to be n-equivalent (denoted as $a \equiv_n b$) if $a \simeq_k^k b$ for each $k \leq n$ and if $a' \equiv_k b'$ for each pair a', b' of respective parents of a, b and each $k < n$.*

The reader should not feel too much confused by the colors here. They will only be needed to deal with one trivial case, in Section 11. Everywhere else all that needs to be remembered is:

Observation 44 *The relation $\equiv_n$ is an equivalence relation of finite index. If $a \equiv_{n+1} b$ then:*

- *parenthood atoms of a and b are atoms of the same predicate;*

- *$a \equiv_n b$;*

- *whenever a' and b' are respective parents of a and b then $a \equiv_n b$;*

Proof of the Observation is by a straightforward application of Definition 42 and Definition 43.

Now the next definition hardly comes as a surprise:

Definition 45 *Let M_n be the relational structure whose set of elements is $Chase/\!\!\equiv_n$, and such that $M_n \models R([a_1], \ldots, [a_n])$ if and only if there are $b_1, \ldots, b_n \in Chase$ such that $b_1 \in [a_1], \ldots, b_n \in [a_n]$ and $Chase \models R(b_1, \ldots, b_n)$.*

---

[9]Remember that $a \equiv_0 b$ means that parenthood atoms of a and b are atoms of the same predicate.

In other words, the relations in M_n are defined in the natural way, as minimal with respect to inclusion relations such that the quotient mapping $q_n : Chase \longrightarrow M_n$ is a homomorphism. If you find Definition 45 complicated, please skip Lemma 46 and go first to Definition 47 and Lemma 48 – we hope they will shed some light.

Since being $(n+1)$-equivalent implies being n-equivalent (Observation 44) the structure M_n is, for each natural n, a homomorphic image of M_{n+1}, and this implies that the sequence of structures $\{M_n\}_{n\in\mathbb{N}}$ satisfies Property 35 (ii). It is also easy to see that it satisfies Property 35 (i):

Lemma 46 $M_n \models \mathcal{T}$ for each $n \in \mathbb{N}$.

Proof: To keep notations as light as possible imagine a rule T from $\mathcal{T}$ of the form $P(x_1, x_2) \wedge Q(y_1, y_2) \Rightarrow \exists z\, R(z, x_1, x_2, y_1, y_2)$ (the argument is exactly the same for any TGD, and even simpler for a plain Datalog rule). Suppose that atoms $P([c_1]_{\equiv_n}, [c_2]_{\equiv_n})$ and $Q([c_3]_{\equiv_n}, [c_4]_{\equiv_n})$ are true in M_n. We need to show that there is an element $[e]_{\equiv_n} \in M_n$ such that $R([e]_{\equiv_n}, [c_1]_{\equiv_n}, [c_2]_{\equiv_n}, [c_3]_{\equiv_n}, [c_4]_{\equiv_n})$ is also true in M_n.

By definition of M_n there exist elements $a_1, \ldots a_4$ of $Chase$ such that $a_i \equiv_n c_i$ for each $i \in \{1, \ldots 4\}$ and that the atoms $P(a_1, a_2)$ and $Q(a_3, a_4)$ are true in $Chase$. But – since $Chase$ is a model of $\mathcal{T}$ – this means that there is an element e of $Chase$ such that the atom $R(e, a_1, a_2, a_3, a_4)$ is also true in $Chase$. This however implies that $R([e]_{\equiv_n}, [a_1]_{\equiv_n}, [a_2]_{\equiv_n}, [a_3]_{\equiv_n}, [a_4]_{\equiv_n})$ is true in M_n, which is exactly what we needed to prove. $\square$

Notice that joinlessness of $\mathcal{T}$ was a crucial assumption here. Suppose the body of T had the form $P(x, v)\ \wedge\ Q(y, v)$ and atoms $P([c_1]_{\equiv_n}, [c]_{\equiv_n})$ and $Q([c_2]_{\equiv_n}, [c]_{\equiv_n})$ were true in M_n. This still would imply existence of atoms $A_1 = P(a_1, a)$ and $A_2 = Q(a_2, a')$, both true in $Chase$, and such that $a_1 \equiv_n c_1$, $a_2 \equiv_n c_2$ and $a \equiv_n a' \equiv_n c$. But a and a' would not need to be equal, and so rule T could not be applied to A_1 and A_2 in $Chase$. This remark explains why – in order to prove Finite Controllability of Sticky Datalog$^\exists$ – we first reduced the problem to Finite Controllability for Joinless Logic.

Let us also remark that it easy to see that if $\mathcal{T}$ is a theory in Guarded Datalog$^\exists$ then Lemma 46 remains true. This is why our technique can be directly applied to show the FC property for Guarded Datalog$^\exists$. Actually, proof in this case is much easier than in the in the case of the Joinless Logic, as the technical details of the proof of Lemma 38 significantly simplify.

Definition 47 *For a conjunctive query ϕ let $Occ(\phi)$ be the set of all variable occurrences in ϕ. More precisely, $Occ(\phi) = \bigcup_{R \in \phi}(\{1, 2 \ldots arity(R)\} \times \{R\})$.*

An n-evaluation of ϕ is a function $f : Occ(\phi) \to Chase$ assigning, to each atom R from ϕ and each position i in R, an element $f(i, R) \in Chase$, in such a way that:

(*) for each pair of atoms R, R' in ϕ if $R(i) = R'(i')$ then $f(i, R) \equiv_n f(i', R')$.

(**) for each atom R in ϕ it holds that $Chase \models f(R)$.

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

It is easy to notice that:

Lemma 48 $M_n \models \phi$ if and only if there exists an n-evaluation of ϕ. $\qquad\square$

See how simple it is: in order to analyze the behavior of queries in the structures M_n we do not need to imagine these complicated finite structures at all! The only structure we need to think about is *Chase*, together with the equivalence relation $\equiv_n$. Imagine a CQ ϕ written in the following way. First there is a conjunction of atoms, and each variable occurs in this conjunction at most once. Then there is a conjunction of equalities between variables. Of course every CQ can be written like this. Now, let ϕ' be ϕ with each equality symbol replaced by $\equiv_n$. What Lemma 48 really says is that there is no need to ever imagine M_n, because $M_n \models \phi$ if and only if $Chase \models \phi'$.

To see Lemma 48 in action let us now prove the following two lemmas. We will need them at some point in the future:

Lemma 49 Consider an M-true conjunctive query $\phi = PP \wedge R \wedge \psi$, where PP is a parenthood atom of some variable x (which means that $PP(1) = x$), and where $R = Q_{F,\delta}(\bar{w})$ and $R(i) = x$. Let $j <_F i$ be a position in R. Then the position $\delta(i, j)$ exists in the atom PP.

Notice that if we assumed that ϕ is true in *Chase* then the claim of the Lemma would follow from Lemma 28 (well, actually it would be Lemma 28 then, modulo obvious rewritings): we are already used to the fact that if an atom R is true in *Chase*, and there is an argument a in R which calls another argument b "granny", then b must occur on the granny position in the parenthood atom PP of a.

It is not however immediately clear why the weaker assumption, that ϕ is just M-true would be sufficient.

Proof of Lemma 49: We know that ϕ is M-true, so also $M_0 \models \phi$. Lemma 48 tells us that there exists a 0-valuation f of ϕ, which means that the atoms $f(PP)$, $f(R)$ and $f(P)$, for each atom P in ψ, are all true in *Chase* and f is such that each two different occurrences of the same variable in ϕ are mapped on 0-equivalent elements of *Chase*. So consider the atoms $f(PP)$ and $f(R)$ in *Chase*. Define $(f(PP))(1) = a$ and $(f(R))(i) = a'$. Since $PP(1) = x$ and $R(i) = x$ we have that $a \equiv_0 a'$. Consider the parenthood atom A of a' in *Chase*. By Lemma 28 we have that position $\delta(i, j)$ exists in A. And since $a \equiv_0 a'$ we have that A and PP are atoms of the same predicate, so position $\delta(i, j)$ must also exist in PP. $\qquad\square$

Lemma 50 Let ψ be an M-true query and let $P_{F,\delta}$ and $R_{G,\gamma}$ be atoms in ψ. Suppose $x = P(1) = R(j)$, for some variable x and some position j in R. Suppose also that positions j' and j'' in R are such that $j' <_G j'' <_G j$. Let i'

28

and i'' be such positions in P that $i' = \gamma(j, j')$ and $i'' = \gamma(j, j'')$. Then $i' <_F i''$ and $\delta(i'', i') = \gamma(j'', j')$

Notice that positions i' and i'' exist in P due to Lemma 49.

Proof: The query ψ is M-true. So consider a 0-evaluation f of ψ. Let $a_P = f(1, P)$ and $a_R = f(j, R)$. Of course $a_P \equiv_0 a_R$. Let also $C_P = f(P)$, $C_R = f(R)$ and let C be the parenthood atom of a_R in *Chase*. Of course C_P and C are atoms of the same predicate (because $a_P \equiv_0 a_R$).

Now use Lemma 28 for $a = a_R$, to show that $i' <_F i''$ and $\delta(i'', i') = \gamma(j'', j')$ hold in C. This of course implies that they also hold in C_P. $\qquad\square$

10   The second little trick

As we said in Section 7, for each M-true CQ ϕ we will construct its "normal form" β. The following lemma describes a single step of the normalization process. Its proof relies on what we find to be the nicest technical idea of this paper[10], so please try to have fun:

Lemma 51 (Second Little Trick) *Consider an M-true conjunctive query $\phi = P \wedge R \wedge \psi$, where P is a parenthood atom of some variable x (which means that $P(1) = x$), and where $R = Q_{F,\delta}(\bar{w})$ 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 $P(\delta(i, j))$ (which exists, due to Lemma 49). Then $\sigma(\phi)$ is also M-true.

Clearly, $\sigma(\phi)$ is more constrained than ϕ, so whatever structure $\mathcal{M}$ we consider it holds that $\mathcal{M} \models (\sigma(\phi) \Rightarrow \phi)$ (this observation has something to do with condition (**) of the Normal Form Lemma).

Notice however that, despite the fact that $\sigma(\phi)$ appears to be more constrained, we also have: $Chase \models (\phi \Rightarrow \sigma(\phi))$. This follows from Lemma 28, which says that each element – call it b – of *Chase* has a unique tuple of parents, and whenever $b = R(i)$ for some atom R the element $R(j)$ (with $j <_F i$, where F, δ are the family pattern of R) must be the same as the element in position $\delta(i, j)$ in the parenthood atom of b.

This implies that every satisfying valuation of ϕ in *Chase* must substitute the same element for the variables $R(j)$ and $P(\delta(i, j))$ anyway, and so the unification from the Lemma does not really lead to more constraints.

But the situation in the structures M_n is different. Lemma 28 is not valid there, as elements of M_n can have more than one tuple of parents. This is because when we identify two n-equivalent elements of *Chase* each of them comes with its own parents, and we cannot be sure that the respective parents will also be n-equivalent, and thus identified. What we know however is that the respective parents will be at least $(n-1)$-equivalent. And this turns out to be sufficient for:

---

[10]And explains why the structures M_n are defined as they are.

Proof of Lemma 51: We want to show that for each natural n the query $\sigma(\phi)$ is true in M_n. Fix $n \in \mathbb{N}$. We know that ϕ is M-true, so $M_{n+1} \models \phi$.

Suppose f is an $(n+1)$-evaluation of ϕ. The lemma will be proved if we can show that the same function f is also an n-evaluation of $\sigma(\phi)$. Of course condition (**) of Definition 47 is still satisfied, as it neither depends on n nor on the equalities between the variables. Also condition (*) is satisfied for the pairs of variables that were already equal in ϕ. What remains to be proved is that condition (*) holds true also for pairs of variables unified by σ. In other words, we need to show 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 $f(P(1), P) \equiv_{n+1} f(R(i), R)$. This is because the variables $P(1)$ and $R(i)$ are equal (to x), so f, being an $(n+1)$-evaluation, must map them to elements of $Chase$ which are $(n+1)$-equivalent. Since f satisfies condition (**) of Definition 47, we know (by Lemma 28) 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, to end the proof, use the fact that respective parents of $(n+1)$-equivalent elements of $Chase$ are n-equivalent. $\qquad\square$

11 The ordering $\rightarrow_\phi$ and cycled queries

We are already used to the fact that each atom comes with an ordering ("family ordering") of its arguments. Now we will extend the (family) ordering on positions of individual atoms to the ordering on variables of conjunctive query the atoms form. Then we will study the new ordering very carefully.

Let us recall that a CQ is clean if it only contains atoms of parenthood predicates. It is also good to remember that if $Q_{F,\delta}$ is a parenthood predicate then (the ordering defined by) F is a tree and that position 1 is always the root of this tree.

Definition 52 *Let ϕ be a clean CQ.*

By $\rightarrow_\phi$ we mean the smallest transitive (but not necessarily 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 $x \rightarrow_\phi y$.

A CQ ϕ is non-cycled *if $\rightarrow_\phi$ is a partial order[11] on $Var(\phi)$ (which in particular means that it is antisymmetric). Otherwise it is* cycled.

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

Lemma 53 *If ϕ is a cycled query consisting of k atoms, then $M_{k+1} \not\models \phi$. So a cycled query is never M-true.*

Proof: Let sequence of atoms $R_1, \ldots R_{j-1}$ for $j \leq k$ be a witness of the fact that Φ ic cycled. It means that there exist a sequence of variables $x_1, \ldots x_j$ such that x_i is a parent of x_{i+1} in atom R_i and $x_1 = x_j$.

---

[11] When $x \rightarrow_\phi y$ then we think that y is smaller than x. Mnemonic hint: the arrowhead of $\rightarrow$ looks like $>$.

Suppose that ϕ is true in M_k (and therefore in M_j). Let f be a j-evaluation of ϕ. From f we can extract two sequences $a_1, \ldots, a_{j-1}$ and $b_2, \ldots, b_j$ such that

- a_i is a parent of b_{i+1} in $Chase$

- $\forall_{1 < i < j} \; a_i \equiv_j b_i$

- $a_1 \equiv_j b_j$

By abusing the notation a bit we could just say that $a_i = f(x_i, R_i)$ and $b_{i+1} = f(x_{i+1}, R_i)$.

Observation 54 *There exists a sequence $c_1, \ldots, c_j$ of elements of $Chase$ such that*

- c_i *is a parent of c_{i+1} in $Chase$*

- $\forall_{i < j} \; c_i \simeq_i^j a_i$

- $c_j = b_j$

Notice that once the Observation is proved, the proof of Lemma 53 will be finished: this is because it follows from the Observation that $c_1 \simeq_1^j a_1 \equiv_j b_j = c_j$, which means that c_1 has the same j-color as c_j. But this leads to a contradiction since c_1 is in a j-history of c_j, and this is exactly what is prohibited by Definition 41.

Proof of Observation: This sequence will be constructed by induction. Let $c_j = b_j$ and $c_{j-1} = a_{j-1}$.

Suppose that c_{i+1} has been defined. Since $c_{i+1} \simeq_{i+1}^j a_{i+1} \equiv_j b_{i+1}$ we have $c_{i+1} \simeq_{i+1}^j b_{i+1}$. Because a_i is a parent of b_{i+1}, there must exist a respective parent c_i of c_{i+1} such that $c_i \simeq_i^j a_i$. $\qquad\square$

This was fortunately the last time we needed to think about colors.

It follows from Lemma 53 that in the proof of the Normal Form Lemma (Lemma 38) we only need to consider non-cycled queries.

12 Non-cycled queries and the normal form

Now please be ready for the most technical part of the paper. Let ϕ be an non-cycled and M-true CQ and let $\rightarrow_\phi$ be the partial order on $Var(\phi)$, as defined in the previous section.

Definition 55 *Call a variable $x \in Var(\phi)$ important if $x = P(1)$ for some atom[12] P in ϕ. Otherwise x is called ordinary.*

---

[12] Do not forget that only parenthood atoms appear in queries

So the important variables are the ones we know a lot about – we know all their parents by name.

Let us remind the reader that the notation PY was introduced in Definition 29.

Definition 56   • For an atom $P = Q_{F,\delta}(\bar{t})$ of ϕ let $\mathcal{I}(P)$ denote the set of such non-root positions i in P that the variable $P(i)$ is important and that for each $j \neq 1$ if $i <_F j$ then $P(j)$ is ordinary.

• For an atom P of ϕ define $top.pos(P) = PY(\mathcal{I}(P))$. Let $top.pos(\phi) \subseteq Occ(\phi)$ be the set of such variable positions (i, P) that $i \in top.pos(P)$.

• For an atom P of ϕ let $top.var(P) = \{P(j) : j \in top.pos(P)\}$. A variable $y \in Var(\phi)$ is a top variable if $y \in top.var(P)$ for some atom P of ϕ.

In other words $top.pos(\phi)$ is the set of positions in the atoms of ϕ, which are, in a certain sense "close to the roots" of the respective atoms – there are no important variables between this position and the root of the atom. The set $top.var(P)$ is a set of variables – these variables that occur in one of the "top positions" of P.

Now we can define the normal form of a conjunctive query:

Definition 57 A CQ ϕ is in the normal form if:

Ideological condition: If P is an atom in ϕ which is a parenthood atom of an important variable x, if $R = Q_{F,\delta}(\bar{t})$ is another atom in ϕ, such that $R(i) = x$, and if j is a position in R such that $j <_F i$, then $R(j) = P(\delta(i, j))$.

Technical condition: Each variable from $Var(\phi)$ occurs in at most one position in $top.pos(\phi)$.

Notice that it follows from the Ideological Condition, that an important variable x of a query ϕ in the normal form can be in the root position in only one atom of ϕ (a query is a set of atoms, so equal atoms count as one). In order to see that suppose that there are two such atoms, P and R. Since ϕ is assumed to be M-true, P and R must be atoms of the same predicate. Now apply the Ideological Condition to P and R and see that it follows that variables in the same positions in P and R must be equal, so P and R are in fact one atom. Call this unique atom having x in the root position PP_x.

Since the root positions are the only positions of important variables which are in $top.pos(\phi)$ this means that the Technical Condition for the important variables is implied by the Ideological Condition.

Notice also that the Ideological Condition is the condition from Lemma 51. So one can imagine now, how we are going to prove Lemma 38 – we will start from the query ϕ (or from something similar – actually it is not going to be exactly ϕ) and perform the unifications from Lemma 51 on it, as long as possible.

The main difficulty in the proof of Lemma 38 is to make sure that the final result of such a unification procedure indeed satisfies the Technical Condition for the ordinary variables, which will be very much needed (in Section 15) for the proof of the Lifting Lemma.

For the (mostly boring and syntactical) details of the proof of Lemma 38 see the next two sections.

As it turns out, the assumption that an M-true query ϕ is in the normal form, or even that it satisfies the Ideological Condition alone, implies a lot about the ordering $\rightarrow_\phi$:

Definition 58 *Let $y, y' \in Var(\phi)$. We will call y' a successor of y if $y' \rightarrow_\phi y$ and there is no such $z \in Var(\phi)$ that $y' \rightarrow_\phi z$ and $z \rightarrow_\phi y$.*

Lemma 59 *Let ϕ be an non-cycled M-true query satisfying the Ideological Condition. Then:*

    A. *Every variable in $Var(\phi)$ is a top variable.*

    B. *If an ordinary variable y' is a successor of an ordinary variable y then there is an atom PP_x such that $y, y' \in top.var(PP_x)$. If an important variable x is a successor of an ordinary variable y then $y \in top.var(PP_x)$.*

Proof of A: Suppose there is a variable $y \in Var(\phi)$ which is not a top variable. Let z be a minimal, with respect to the ordering $\rightarrow_\phi$, important variable such that $y \in Var(PP_z)$. Let $<_F, \delta$ be the family pattern of PP_z.

We know that $y \notin top.var(PP_z)$, so there must be an important variable $x \in Var(PP_z)$ such that $x \neq z$ and $i <_F j$, where $PP_z(i) = y$ and $PP_z(j) = x$. But this means, since ϕ satisfies the Ideological Condition, that y occurs in the atom PP_x (in position $\delta(j, i)$), which contradicts the minimality of z.

Notice that we silently used Lemma 49 here, and this is where the assumption that ϕ is M-true was needed.

Proof of B: If y' (ordinary or important) is a successor of y then, by the definition of $\rightarrow_\phi$, there must be an atom PP_x, with the family ordering $<_F$, and positions i, i' in PP_x, such that $i <_F i'$, $PP_x(i) = y$, $PP_x(i') = y'$. Notice also that, if i and i' are as above, there is no position j satisfying $i <_F j <_F i'$ – this is because the variable $PP_x(j)$ would be between y and y' in the ordering $\rightarrow_\phi$. Let x be a minimal, with respect to the ordering $\rightarrow_\phi$ variable such that PP_x satisfies the above requirements. Now, use the argument from the proof of claim A. to show that i is a top position in PP_x. $\qquad\square$

Lemma 60 *Let ϕ be an non-cycled M-true query in the normal form. Then:*

    A. *Each ordinary variable has exactly one successor.*

    B. *Suppose $y \in top.var(PP_x)$, the variable z is important and $z \rightarrow_\phi y$. Then $z \rightarrow_\phi x$.*

Proof of Lemma 60. Claim A. follows directly from Lemma 59B and from the Technical Condition. Claim B. follows directly from A. ☐

Now all the notions appearing in the Normal Form Lemma and in the Lifting Lemma are defined and what remains to be done is proving the two Lemmas. The next two sections are devoted to the proof of Lemma 38. But once you know the main idea, which is performing the unifications from the Second Little Trick as long as needed/possible, the proof is hardly exciting. Then, in the last section of the paper, the Lifting Lemma (Lemma 39) is proved, and this is where the rabbit is pulled out of the hat. So maybe it is not a bad idea to skip Sections 13 and 14 and jump directly to Section 15.

13 Proof of Lemma 38. Part one: the normal form of ϕ.

In this Section we consider some fixed M-true CQ ϕ and construct a CQ β being the normal form of ϕ, as specified by Lemma 38 and Definition 57.

The definition of β itself (Definition 65) is quite natural and not very complicated. The really technical part begins right after Definition 65, where we prove that the defined query is indeed the normal form of ϕ. There are no deep ideas there, we just need to carefully analyze the consequences of the unifications resulting from applications of the Second Little Trick, and such analysis is, by its nature, a very syntactic thing.

NOTATIONAL CONVENTIONS. The typical situation in this part of the paper will be that we will consider some **fixed** CQ θ, and restrict attention only to queries being equality variants of θ. By this we mean queries that can be obtained from θ by renaming some of the occurrences of variables.

We need a convenient language for this scenario, so let us start from defining such a language.

Equality variants of θ only differ by the names of the variables, and they all have the same set of positions. We will imagine that θ is a conjunction of some atoms $P^l_{F_l,\delta_l}$, with $l \in V$ for some set V, and we will denote by $\mathcal{P}$ the set of all positions in θ (which is a disjoint union of the sets of positions in the atoms). By saying "let $i \in \mathcal{P}$" we can now address a position directly, without specifying in which of the atoms of θ it is located. The cost to pay is that no longer we can use 1 for the name of the position in the root of the atom, so by $root(i)$ we will mean that $i \in \mathcal{P}$ is a position in the root of some P^l. By $\mathcal{P}_\xi(i)$ (or just $\mathcal{P}(i)$ when the context is clear) we will mean the variable in position $i \in \mathcal{P}$ in the equality variant ξ of θ.

Let $\prec$ be the disjoint union of all relations $<_{F_l}$, so that by $i \prec j$, for $i, j \in \mathcal{P}$, we mean that positions i and j are in the same atom P^l, for some l, and $i <_{F_l} j$. Similarly, let δ be the disjoint union of all the functions δ_l.

It will be also convenient to have a notation $\Delta(i, i, j', j')$ for the formula $root(i) \wedge \delta(i, i') = \delta(j, j')$.

In other words (for those who do not like our new language) $\Delta(i, i, j', j')$

means that there are l and l' such that i is the position in the root of P^l, i' is a position in P^l, j and j' are positions in $P^{l'}$, and $\delta_l(i, i') = \delta_{l'}(j, j')$.

Since the objects defined in this subsection ($\mathcal{P}$, Δ, $\prec$, δ) depend on our current choice of θ, they only have meaning in the contexts where θ is defined.

See how conveniently the Ideological Condition from Definition 57 can now be expressed:

($\heartsuit$) for each $i, i', j, j' \in \mathcal{P}$, if $\Delta(i, i', j, j')$ and $\mathcal{P}(i) = \mathcal{P}(j)$ then $\mathcal{P}(i') = \mathcal{P}(j')$.

THE UNIFICATION PROCEDURE. For a query ψ let $u(\psi)$ be a result of:
The unification procedure:
fix θ as ψ;
    /* So that the above notations apply */
$\xi := \psi$
while there exist: $i, j, i', j' \in \mathcal{P}$ such that $\Delta(i, i', j, j')$, $\mathcal{P}_\xi(i) = \mathcal{P}_\xi(j)$ and $\mathcal{P}_\xi(i') \neq \mathcal{P}_\xi(j')$
do
{
replace all occurrences of $\mathcal{P}_\xi(j')$ in ξ by $\mathcal{P}_\xi(i')$
(in other words $\xi := \xi[\mathcal{P}_\xi(j')/\mathcal{P}_\xi(i')]$);
}
forget that θ was ψ;
    /* So that we can use θ somewhere else */
remove the repeating atoms from ξ;
return ξ as $u(\psi)$;
end of the unification procedure.

What this procedure does is exactly checking if the Ideological Condition is satisfied in ξ, and if it isn't, unifying the variables that violate the Ideological Condition, using the Second Little Trick. Clearly, the procedure always terminates and $u(\psi)$ always satisfies the Ideological Condition. We also know, from Lemma 51, that if ψ is M-true then $u(\psi)$ also is. It is also obvious that $Chase \models (u(\psi) \Rightarrow \psi)$.

We are however not claiming that $u(\psi)$ is always the normal form of ψ. This is because there is no reason for the Technical Condition to be satisfied in $u(\psi)$. One could for example easily take ψ to be a query which already satisfies the Ideological Condition (so that $u(\psi) = \psi$) but not the Technical Condition.

The unification procedure is nondeterministic – at each step it nondeterministically selects, for the unification, a pair of variables. But:

Lemma 61 *The result of the unification procedure – the $u(\psi)$ – is unique for ψ, in the sense that it does not depend on the nondeterministic choices made by the procedure.*

Proof: Since the set of positions $\mathcal{P}$ is fixed, a query ξ can be identified with its equality relation $=_\xi$ on the set of positions (this relation says that the variables

in two positions are equal in ξ). What the unification procedure does is computing the fixpoint of some Datalog program. The relations Δ and $=_\psi$ are the input predicates of this program while $=_{u(\psi)}$ is its output predicate. The rules of the program are the condition ($\heartsuit$) above, and the reflexivity, symmetricity and transitivity axioms for $=_{u(\psi)}$. And of course the fixpoint of a Datalog program does not depend on the order of execution. $\qquad\square$

ELEVATING THE IMPORTANCE OF THE VARIABLES. As we said, $u(\psi)$ is not always in the normal form, as it may not satisfy the Technical Condition. The Technical Condition concerns the ordinary variables, and the reason why ψ may not satisfy it is that there may be some unwelcome equalities between ordinary variables in ψ. Our way towards the solution of the problem is to elevate the (potentially) misbehaving ordinary variables to the position of importance, so that they are allowed more.

Definition 62 *For a query ψ by a closure of ψ we will mean any query of the form $\psi \wedge \bigwedge_{x \in Var_{ord}(\psi)} R(x, \bar{x})$ where $Var_{ord}(\psi)$ is the set of all the ordinary variables of ψ, R is any parenthood predicate and $\bar{x}$ is a tuple of fresh variables.*

It is now straightforward to see that:

Lemma 63 *if ψ' is any closure of ψ then:*

- *if $x \in Var(\psi)$ then x is important in ψ';*

- *each ordinary variable in ψ' occurs in ψ' only once;*

- *$Chase \models (\psi' \Rightarrow \psi)$;*

- *ψ' satisfies the Technical Conditions (although not necessarily the Ideological Condition).*

It is also not hard to show that:

Lemma 64 *If ψ is M-true then there exists an M-true ψ' being a closure of ψ.*

Proof: For each $n \in \mathbb{N}$ if $M_n \models \psi$ then also $M_n \models \psi$ for some closure ψ' of ψ. This is because each element of M_n is a child in some parenthood atom valid in M_n.

Since there are only finitely many possible closures of ψ, if ψ is M-true, then there is a closure ψ' which is true in M_n for infinitely many numbers n. Now use the argument from the First Little Trick. $\qquad\square$

From now on, for an M-true conjunctive query ψ by $c(\psi)$ we will denote an M-true closure of ψ.

THE QUERY β – THE NORMAL FORM OF ϕ. We are finally ready to name the query β which is the normal form of ϕ:

Definition 65 $\beta = u(c(\phi))$.

Lemma 66   1. β is M-true;

2. $Chase \models (\beta \Rightarrow \phi)$;

3. β satisfies the Ideological Condition;

4. β satisfies the Technical Condition.

Claims 1)–3) follow immediately from the construction. But Claim 4) is not obvious at all, it needs a proof, and this proof, while not really complicated, is unfortunately not going to be short. Notice however that once Lemma 66 is proved then of course also the proof of Lemma 38 will be finished.

14   Proof of Lemma 38. Part two: proof of Lemma 66.4.

What remains to be done to show that β is indeed the normal form of ϕ is proving that it satisfies the Technical Condition. The main proof technique is a patient syntactical analysis of the unifications that led to β.

Let now θ – the query with respect to which the notations are defined in the beginning of the previous Section – be equal to β. And this is not going to change any more.

Before we show Lemma 66 let us try to imagine how β looks like. There are two kinds of atoms in β. One are those that originated in ϕ. Now they contain only important variables. Second kind are the atoms that were originally added to ϕ when $c(\phi)$ was created. They may contain ordinary variables, but also, after all the unifications on $c(\phi)$ they contain some important variables in non-root positions.

Proof of Lemma 66.4. Please allocate memory for two more equality variants of β. They will be called β_0 and β_{wu} (as "weakly unified"), which will at the end turn out to actually be equal to β.

We need to do something strange now. Due to a reason that will be explained later, we need to destroy the structure of β, to some extend, and then to rebuild it again:

Definition 67 *Let β_0 be the result of substituting a fresh variable for each occurrence of an ordinary variable in β.*

Of course β_0 is not simply $c(\phi)$. The unifying procedure run on $c(\phi)$ **a)** unified some of the fresh variables in the new atoms of $c(\phi)$ with the variables from $Var(\phi)$), and **b)** unified some of these fresh variables with other fresh variables. The query β_0 is the result of undoing the unifications from **b)**, but not from **a)** .

Lemma 68 $u(\beta_0) = \beta$;

This is because $\beta = u(c(\phi))$ is more unified than β_0 and β_0 is more unified than $c(\phi)$. Use the datalog fixpoint argument from the proof of Lemma 61. $\square$

Clearly, β_0 satisfies the Technical Condition.

Now we are going to run a version of the unification procedure on β_0, which will lead us to a new query β_{wu}. The query β_{wu} is in fact β, but this is a secret yet. In this new unification procedure the pairs of variables to be unified, will be carefully hand-picked in the *correct order* and nothing will be left to nondeterminism. Thanks to that we will be able to make sure that the Technical Condition keeps being satisfied. One of course could ask here why did we bother to define β first, if then we run another unification procedure on β_0 anyway? And the answer is, that we only can know the correct order once we know β! So we need to know β, constructed in any way, to be able to construct β again in the careful way.

Notice that whatever our order of the execution of the unification procedure is going to be, we will never unify any important variable with any other variable (important or ordinary). If x is an important variable in β then it is also important in β_0 and for each $i \in \mathcal{P}$ we have that $\mathcal{P}_\beta(i) = x$ if and only if $\mathcal{P}_{\beta_0}(i) = x$. This observation leads to a series of definitions:

Definition 69 *Call a position $j \in \mathcal{P}$ ordinary, if the variable $\mathcal{P}_{\beta_0}(j)$ is ordinary (or – equivalently – if the variable $\mathcal{P}_\beta(j)$ is ordinary). Otherwise j is important. Let $\mathcal{P}_{ord}$ and $\mathcal{P}_{imp}$ denote, respectively, the sets of ordinary and important positions.*

Definition 70 *For an ordinary position $j \in \mathcal{P}$ denote by nearest.pos(j) the smallest, with respect to the ordering $\prec$, important position i in $\mathcal{P}$ such that $j \prec i$. By nearest.var(j) denote the variable $\mathcal{P}(nearest.pos(j))$.*

In other words *nearest.pos*(j) is the first important position on the path from j to the root of the atom where j is located, and *nearest.var*(j) is the name of the important variable that lives there.

Definition 71
For an important variable x let layer$(x) = \{j \in \mathcal{P}_{ord} : nearest.var(j) = x\}$.

Of course:

Lemma 72 *The sets layer(x), for $x \in Var_{imp}(\beta)$, form a partition of $\mathcal{P}_{ord}$ (by which we mean that they are pairwise disjoint and that their union equals $\mathcal{P}_{ord}$).*

Let us also remind that an ordinary position j is a top position if $root(nearest.pos(j))$ (this is Definition 56 in our new language).

Now we are ready for:

The weak unification procedure:
$\xi := \beta_0$;
$to\text{-}be\text{-}considered := Var_{imp}(\beta_0)$

while $to\text{-}be\text{-}considered \neq \emptyset$
do:
{   /* $\diamond$ */
Let x be a minimal, with respect to the ordering $\rightarrow_\beta$, variable in the set $to\text{-}be\text{-}$
$considered$;
   /* See! Here is where we need to know β.   */
Let $i \in \mathcal{P}$ be such that $\mathcal{P}(i) = x$ and $root(i)$;
   /* We took the position in the root of the atom PP^x.   */
For each non-top position j' such that $j' \in layer(x)$,
 and for each i' such that $\Delta(i, i', nearest.pos(j'), j')$
   substitute the variable $\mathcal{P}(j')$ in ξ by the variable $\mathcal{P}(i')$;
   /* Call the above the "unification step"   */
Remove the variable x from $to\text{-}be\text{-}considered$;
}
Return ξ as β_{wu}.
end of the procedure.

Let us try to explain the substitution step of the procedure.

Once x is fixed (which is one of the $\rightarrow_\beta$ minimal variables not yet considered) we look for all possible positions $j' \in \mathcal{P}_{ord}$, such that the if we started, in j', a path (in the ordering $\prec$) towards the root of the atom where j' is located, the first important position on this path would be some non-root position $j = nearest.pos(j')$, and the variable there would be x.

Then we ask j: "how do you call j' ?". And we get some answer "$\delta(j, j')$". So we ask i: "whom do you call $\delta(j, j')$?". And we get some answer "i'". Then we say: "So, since the variables in i and j are equal, the Ideological Condition wants the variables in i' and j' to unify. From now on the one in j' will adopt the name of the one in i'".

Of course unification means more than just renaming the variable in j'. We need to rename all the occurrences of $\mathcal{P}(j')$ in the current ξ. But the trick is that:

Lemma 73 *Each time the control passes the point marked with $\diamond$, if $x \in$ to-be-considered and $j \in layer(x)$ then $\mathcal{P}(j)$ is a fresh variable (which means that it only occurs once in ξ).*

Proof: There are two ways for a variable to lose its freshness. One is to be copied somewhere, which means being the i' from the unification step, the other is to be substituted with another variable, which means being the j' from the unification step.

But notice that each non-top position in $\mathcal{P}$ is exactly once the j' from the unification step, and right after that the variable $nearest.var(j')$ is removed from the set *to-be-considered*. Notice also, that each position that, at some point of time, had already been the i' of the unification step, must be a position in some atom PP^z, with z not being in the set *to-be-considered* any more (because in the unification step we take the names for the variables from the atom having the currently considered variable x in the root). And if $k \in layer(x)$ and $x \in$ *to-be-considered* then k is a position in the atom PP^z for some z such that $z \rightarrow_\beta x$, which implies that $z \in$ *to-be-considered*. $\qquad \square$

The meaning of the last lemma is that the substitution in the unification step is just a renaming of one variable occurrence – the one in j'. It does not propagate, in the sense that it does not force any other renamings. This means that there is just one chance for a position, during the execution of the procedure, to have its variable changed – when this position is the j' from the unification step. Since only non-top positions are ever the j', the next lemma follows:

Lemma 74 *If j is a top position in $\mathcal{P}$ then $\mathcal{P}_{\beta_0}(j) = \mathcal{P}_{\beta_{wu}}(j)$*

Lemma 74 implies that the query β_{wu} satisfies the Technical Condition. But we still cannot be sure that it also satisfies the Ideological Condition. While the while loop from the original unification procedure (from Section 13) really checks for the premise of the Ideological Condition and, if this premise holds, it performs the unifications, and does it as long as needed, the loop in the weak unification procedure only performs some hand-picked unifications. We need one more lemma to improve our understanding of how the query β_{wu} looks like:

Lemma 75 *If, at some point of the execution of the weak unification procedure, the variables in positions i' and j' were unified (i.e. the variable from i' was copied to j') then they remain equal in β_{wu}*

Proof: As we said before, the variable in each position can only be changed once by the weak unification procedure. So the variable in j' will not be changed any more. We need to make sure that the variable in i' will not be changed after it was copied to j'. Suppose the variable x was being considered when the variables in positions i' and j' were unified. This means that either i' is a top position in PP_x (which means, as we observed before, that the variable there can never be changed) or $i' \in layer(z)$ for some z such that $x \rightarrow_\beta z$. But this means that at the moment of the unification z is no longer in the set *to-be-considered*, and so the variable in i' was already substituted, and it never will again. $\qquad \square$

Now the last lemma we need to show in order to finish the proof of Lemma 66:

Lemma 76 *The query β_{wu} satisfies the Ideological Condition. In consequence, $\beta_{wu} = \beta$.*

Proof: We know from Lemma 75 that β_{wu} is weakly unified, which means that if i, i', j, j' are positions in $\mathcal{P}$ such that $\Delta(i, i'j, j')$, if $\mathcal{P}_{\beta_{wu}}(i) = \mathcal{P}_{\beta_{wu}}(j)$, and if $j = nearest.pos(j')$ then $\mathcal{P}_{\beta_{wu}}(i') = \mathcal{P}_{\beta_{wu}}(j')$.

What we need to show is that the Ideological Condition holds, that is if i, i', j, j' are positions in $\mathcal{P}$ such that $\Delta(i, i'j, j')$, if $\mathcal{P}_{\beta_{wu}}(i) = \mathcal{P}_{\beta_{wu}}(j)$, then $\mathcal{P}_{\beta_{wu}}(i') = \mathcal{P}_{\beta_{wu}}(j')$.

Suppose that the above is not true and let x be a minimal, with respect to the ordering $\rightarrow_\beta$, important variable such that there exist positions i, i', j, j' in $\mathcal{P}$ such that $\Delta(i, i'j, j')$ and $\mathcal{P}_{\beta_{wu}}(i) = \mathcal{P}_{\beta_{wu}}(j)$ but $\mathcal{P}_{\beta_{wu}}(i') \neq \mathcal{P}_{\beta_{wu}}(j')$.

Let y be an important variable such that $j' \in layer(y)$, and let $k_j = nearest.pos(j')$ (so that $\mathcal{P}_{\beta_{wu}}(k_j) = y$). Of course it cannot be that $k_j = j$, as this would contradict the assumption that β_{wu} was weakly unified. So we have $j' \prec k_j \prec j$.

Let $k_i \prec i$ be such position that $\delta(i, k_i) = \delta(j, k_j)$. From Lemma 50 we know that $i' \prec k_i$ and $\delta(k_i, i') = \delta(k_j, j')$.

Notice that $\delta(i, k_i) = \delta(j, k_j)$ implies that $\mathcal{P}_\beta(k_i) = \mathcal{P}_\beta(k_j)$. This is because the variables in i and j are equal in β and β satisfies the Ideological Condition. But $\mathcal{P}_\beta(k_i) = \mathcal{P}_\beta(k_j) = y$ is an important variable, so we have that $\mathcal{P}_{\beta_{wu}}(k_i) = \mathcal{P}_{\beta_{wu}}(k_j) = y$.

Let now $k \in \mathcal{P}$ be such that $root(k)$ and $\mathcal{P}_{\beta_{wu}}(k) = y$. Such k must exist because each important variable is a root somewhere. Let k' be such that $\delta(k, k') = \delta(k_j, j')$ (and thus also $\delta(k, k') = \delta(k_i, i')$).

Since $x \rightarrow_\beta y$, by the minimality of x we now get that $\mathcal{P}_{\beta_{wu}}(k') = \mathcal{P}_{\beta_{wu}}(j')$ and $\mathcal{P}_{\beta_{wu}}(k') = \mathcal{P}_{\beta_{wu}}(i')$. Contradiction. $\qquad\square$

This ends the proof of Lemma 66 and of Lemma 38.

15 Proof of the Lifting Lemma

In this section we show what remains to be shown: that if $M_0 \models \psi$ and ψ is in the normal form then also $Chase \models \psi$.

As we remember from Section 9, $M_0 \models \psi$ means that there exists a 0-evaluation of ψ. Such a 0-evaluation is a function assigning to each variable occurrence in ψ an element of $Chase$ in such a way that the atoms in ψ map into atoms true in $Chase$ and (different occurrences of) equal variables map to 0-equivalent elements of $Chase$. $Chase \models \psi$ means almost the same, the only difference is that equal variables map to equal elements of $Chase$, not just to 0-equivalent.

Definition 77 *A 0-evaluation f is faithful with respect to a set $S \subseteq Var(\psi)$ 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 $f(i, R)$.

Being faithful with respect to S means to look, inside S like a real valuation of a ψ in *Chase*. Clearly $Chase \models \psi$ if and only if there exists a 0-evaluation faithful with respect to $Var(\psi)$. On the other hand, since $M_0 \models \psi$, we know that there exists a 0-evaluation faithful with respect to $\emptyset$. We are going to gradually modify this 0-evaluation to make it more and more faithful, until we get one faithful with respect to $Var(\psi)$.

The sets S we will be interested in are ideals in $Var(\psi)$:

Definition 78 *Subset $S \subseteq Var(\psi)$ is an important ideal if:*

1. *If $x \in S$ and $x \rightarrow_\psi y$ then also $y \in S$.*

2. *All maximal elements of S are important variables.*

From now on let S be an important ideal and let $x \in Var(\psi)$ be a minimal important variable not in S. Let $PP_x = Q_{F,\delta}(x, \bar{x})$ be, as usually, the parenthood atom of x in ψ. Let S' be the important ideal generated by x and S.

Lemma 79   1. *If R is an atom in ψ such that $Var(R) \subseteq S'$ but $Var(R) \not\subseteq S$ then $R = PP_x$.*

2. *$S' \setminus S = top.var(PP_x)$*

Proof: 1) Each atom in ψ is the PP atom of some important variable. If R is the PP atom of some $y \in S$ then $Var(R) \subseteq S$. If R is the PP atom of some $y \notin S'$ then of course $Var(R) \not\subseteq S'$. And x is the only important variable in $S' \setminus S$.

2) This follows easily from Lemmas 59 and 60. Let us show, for example, that $top.var(PP_x) \subseteq S' \setminus S$. Of course $top.var(PP_x) \subseteq S'$ so what we need to show is that $top.var(PP_x) \cap S = \emptyset$. Let $y \in top.var(PP_x)$. Suppose $y \in S$. This would mean that there exists an important $z \in S$ such that $z \rightarrow_\psi y$. But, by Lemma 60, this would imply that $z \rightarrow_\psi x$, which is a contradiction. The proof of the other inclusion is left as an easy exercise. $\square$

We will need the following easy observation about local (restricted to one atom only) modifications of 0-evaluations:

Definition 80 *Suppose f is a 0-evaluation, $f' : Occ(\psi) \to Chase$ 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;*

- *$Chase \models f'(P)$*

- *$f'(i, P) \equiv_0 f(i, P)$ for each position i in P.*

Lemma 81 *If f is a 0-evaluation and f' is P-similar to f then f' is also a 0-evaluation.* $\square$

Let S, S' and x be as above. In view of Lemma 79 1) and Lemma 81, due to an induction argument, in order to prove Lemma 39, it now only remains to show:

Lemma 82 *Let f a 0-evaluation faithful with respect to S. Then there exists a 0-evaluation f', PP_x-similar to f and faithful with respect to S'.*

Proof: First we of course define $f'(i, R) = f(i, R)$ for each atom $R \neq PP_x$, and each position i in R, so the first condition of Definition 80 is satisfied.

We will now define $f'(PP_x)$. Then we will notice that the second and third conditions from Definition 80 hold, so f' is indeed a 0-evaluation. Finally we will see that f' is faithful with respect to S'.

Let $\mathcal{I}(PP_x) = \{i_1, i_2 \ldots i_s\}$, where $\mathcal{I}(PP_x)$ is the set of maximal important non-root positions, as in Definition 56. Let $y_1, y_2, \ldots y_s$ be the important variables in positions $i_1, i_2 \ldots i_s$ in PP_x (the variables may repeat, this does not bother us). For each $1 \leq j \leq s$ let $d_j = f(y_j)$ (notice that this definition makes sense, because $y_j \in S$ for each j) and let $b_j = f(i_j, PP_x)$.

Clearly, since f is an evaluation, we have $b_j \equiv_0 d_j$ for all j. But it means that we are now in the situation of Lemma About the Future (Lemma 30), where $A = f(PP_x)$.

So let C be as in Lemma 30. For any position $j \in top.pos(PP_x)$ define $f'(j, PP_x)$ as $C(j)$. Notice, that we can be sure (thanks to Lemma 30) that $f'(j, PP_x) \equiv_0 f(j, PP_x)$.

Let now j be a position in PP_x which is not in $top.pos(PP_x)$. That means that the variable $z = PP_x(j)$ is in S. Define $f'(j, PP_x)$ as $f(z)$. The condition $f'(j, PP_x) \equiv_0 f(j, PP_x)$ now holds trivially, since f was a 0-evaluation.

We defined a function f', which satisfies the first and the third condition from Definition 80. Now we need to check that $Chase \models f'(PP_x)$. We know that $Chase \models C$, so this part of proof would be finished if we could show that $f'(PP_x) = C$. Of course by the definition of f' the atoms $f'(PP_x)$ and C have equal elements of $Chase$ in the root and in all the positions in the set $top.pos(PP_x)$. But this is not that clear what happens in the remaining positions. 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 8 lines:

Consider two positions in PP_x: $\quad i \in \{i_1, i_2 \ldots i_s\}$ and $j <_F i$. Let $z = P(j)$ and let y be the variable in position i. Since y is important, its parenthood atom, PP_y, is in ψ.

Since ψ is in the normal form, we know, by the Ideological Condition, that $PP_y(\delta(i, j)) = z$. Since we defined $f'(j, PP_x)$ to be $f(z)$, we get $f'(j, PP_x) = f(\delta(i, j), PP_y)$. What we want to show is that $f'(j, PP_x) = C(j)$. But this now follows directly from Lemma 28.

In order to finish the proof of the Lemma we still need to notice that f' is S'-faithful. The atoms described by Definition 77 are now all the atoms that were already contained in S, and one new atom PP_x. If $PP_x(j)$ was in S we defined $f(j, PP_x)$ as $f(PP_x(j))$, so we did not spoil anything. The only problem could

be with the values assigned to positions in PP_x with variables from $S' \setminus S$. But, by the Technical Condition each of these variables occurs in PP_x only once, so the condition from Definition 77 is trivially satisfied. □

References

[BGO10]   V. Barany, G. Gottlob, and M. Otto. Querying the guarded fragment. *Proceedings of Logic in Computer Science*, page 1–10, 2010.

[CGP10a] A. Cali, G. Gottlob, and A. Pieris. Advanced processing for ontological queries. *Proceedings of VLDB*, 3(1):554–564, 2010.

[CGP10b] A. Cali, G. Gottlob, and A. Pieris. Query answering under nonguarded rules in datalog+/-. *Web Reasoning and Rule Systems Lecture Notes in Computer Science*, 6333:1–17, 2010.

[GM13]   T. Gogacz and J. Marcinkowski. On the BDD/FC conjecture. *Proceedings of ACM Symposium on Principles of Database Systems*, pages 127–138, 2013.

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

[Mar95]   J. Marcinkowski. Undecidability of the horn clause finite implication problem. *Annual Conference on Computer Science Logic*, page preprint, 1995.

[MP03]   J. Marcinkowski and L. Pacholski. Thue trees. *Ann. Pure Appl. Logic*, 119(1-3):19–59, 2003.

[RKH08]   S. Rudolph, M. Krötzsch, and P. Hitzler. All elephants are bigger than all mice. *21st Description Logic Workshop, Dresden, Germany*, 2008.

[Ros06]   R. Rosati. On the decidability and finite controllability of query processing in databases with incomplete information. *Proceedings of ACM Symposium on Principles of Database Systems*, pages 356–365, 2006.