Converging to the Chase – a Tool for Finite Controllability

Tomasz Gogacz, Jerzy Marcinkowski,
Institute of Computer Science, University Of Wroclaw,

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

I. Introduction

Tuple generating dependencies (TGDs), recently also known as Datalog$^\exists$ rules[1], 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 database dependencies), but the knowledge of the database facts is limited.

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

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

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

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* [CGT09].

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

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. The **Joinless Logic** we consider in this paper is a generalization of Linear Datalog$^\exists$, in the sense that we no longer restrict the body of the rule to be a single atom, but we still demand that each variable occurs in the body only once. Let us note that allowing variable repetitions in the heads does not 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 in Joinless Logic (see Section III for slightly more about this issue).

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 [CGP10] and then extended in [CGP10+/-] as Sticky-Join Datalog$^\exists$. A set $\mathcal{T}$ of TGDs is sticky, if some positions in the relations occurring in the rules can be marked as "immortal" in such a way that the following conditions are satisfied:

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

Let us remark here, that the above property, that we use as a definition of Sticky Datalog$^\exists$, is actually called "the sticky-join property" in [CGP12], and is a consequence of slightly more complicated definitions of both Sticky Datalog$^\exists$ in [CGP10] and Sticky-Join Datalog$^\exists$ in [CGP10+/-] (see Theorem 4.3 in [CGP12]). This means that Theorem 1 of our paper holds both for Sticky Datalog$^\exists$ and Sticky-Join Datalog$^\exists$. Actually, the

[1]By a TGD we always mean a single-head TGD.

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:

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 D if $Chase(\mathcal{T}, D) \models \Psi$ then $Chase^{k_\Psi}(\mathcal{T}, D) \models \Psi$. The BDD property turns out to be equivalent to positive first order rewriteability [CGT09]: $\mathcal{T}$ has the BDD property if and only if for each UCQ Ψ there exist a UCQ $\bar{\Psi}$ such that for each database instance D (finite or not) it holds that $Chase(D, \mathcal{T}) \models \Psi$ if and only if $D \models \bar{\Psi}$.

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

It is usually very easy to see whether a logic is BDD and it is usually very hard to see whether it FC.

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

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

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

Theorem 1: 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 2: Joinless Logic is FC.

To prove Theorem 2 we propose a technique, which we think is quite elegant, and relies on two main ideas. One is that we carefully trace the relations (we call them family relations) between elements of $Chase$ which are ever involved in one atom. The second idea is to consider an infinite sequence of equivalence relations, defined by the types of 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 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 form a Joinless theory is a huge and very complicated structure, much more complex than the bounded tree-width $Chase$ resulting from guarded TGDs, and the ability to control it can give insight into chases generated by logics enjoying better practical motivation – Theorem 1 serves here as a good example. But still Theorem 2 is a very strong generalization of the result of Rosati about Linear Datalog$^\exists$, which itself was viewed as well motivated, while the technique we develop in order to prove it is powerful enough to give, as a by-product, an easier proof of the finite controllability result for sets of guarded TGDs [BGO10] (see Section XI). 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 1 we do not seem to use much much more than just the fact that Sticky Datalog$^\exists$ is BDD. In our parallel paper [GM12] 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

In Section II we prove Theorem 1, assuming Theorem 2.

The proof of Theorem 2, which is the main technical contribution of this paper, is presented in Sections III–X.

Finally, in a very short Section XI, we comment on the relations between our construction and the FC property for guarded sets of TGDs.

II. From Joinless Logic to Sticky Datalog$^\exists$

This Section is devoted to the proof of Theorem 1 (assuming Theorem 2).

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

Suppose now that Sticky Datalog$^\exists$ is not FC, and that some sticky theory $\mathcal{T}$ of size l, over the signature Σ, some finite database instance D and some query Φ are a minimal counterexample for FC. By "size" of $\mathcal{T}$ we mean here the maximal arity of atoms in the heads of the rules of $\mathcal{T}$. When we say "minimal" we mean that l is the smallest possible.

We are going to prove two Lemmas:

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

Lemma 4: There is a finite database instance D' such that the pair $D', \mathcal{T}$ is weakly saturated and the triple $\mathcal{T}$, D', and Φ is also a counterexample for FC.

Notice that once the two above lemmas are proved we get a contradiction, as the size of the counterexample $\mathcal{T}$, D', Φ from Lemma 4 is l, so it is a minimal counterexample, and thus, by Lemma 3 the pair $D', \mathcal{T}$ cannot be weakly saturated. So once the lemmas are proved the proof of Theorem 1 will be finished. Theorem 2 will be used to prove Lemma 4.

Proof of Lemma 3: Suppose the pair $D, \mathcal{T}$ was weakly saturated. We will construct a new sticky theory $\mathcal{T}_D$ of size at most $l-1$, over a new signature Σ_D and a new query Φ_D, such that the triple $\mathcal{T}_D$, $\emptyset$, Φ_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 form the definition of Σ_D. For a predicate $Q \in \Sigma$, of arity j, and for a partial function $\gamma : \{1, 2, \dots j\} \to D$ let Q_γ be a new predicate, of arity $j - |Dom(\gamma)|$. Σ_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 $\Sigma \subseteq \Sigma_D$ (we identify Q with $Q_\emptyset$).

To denote the predicates from Σ_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'_γ 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 D as arguments in atoms over relations from Σ_D, the above notational convention does not lead to confusion as long as we only talk about atoms over Σ_D. But atoms over Σ_D can easily be confused with atoms over Σ with constants from 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 D (over Σ) can be read as a zero arity predicate over Σ_D.

We are now going to define $\mathcal{T}_D$.

For a rule T from $\mathcal{T}$ by a constantification of T we will mean a formula $\sigma(T)$, where σ is a mapping that assigns constants from 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 $c \in D$) 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 Σ_D.

Let now theory $\mathcal{T}_D$ over Σ_D consist of all the facts from 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 $\mathcal{T}_D$ is also sticky (hint: mark as immortal the same positions as in $\mathcal{T}$), and that the size of $\mathcal{T}_D$ is at most $l - 1$.

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

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

Observation 5: $\mathcal{C} = \mathcal{C}_1$.

For the **proof of the Observation** notice that each atom of $Chase(\mathcal{T}, D)$ is either an atom of D or it contains some constant from D in a marked position. This is because (as $D, \mathcal{T}$ is weakly saturated) the only way for $\mathcal{T}$ to derive any atoms which are not in 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 D in a marked position, the theories $\mathcal{T}$ and $\mathcal{T}_D$ derive exactly the same atoms. $\square$

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

In order to prove that $\mathcal{T}_D$, $\emptyset$, Φ_D is a counterexample for FC we still need to show that for each finite structure M over Σ_D there is $M \models \Phi_D$. So suppose there was a finite M such that $M \models \mathcal{T}_D$ and $M \not\models \Phi_D$. Define a new finite model M^D as a structure over Σ, containing all the elements of M and all the elements of D, and all the atoms true in M. Of course the atoms true in M were over the signature Σ_D, but to define M^D we read them as atoms over Σ. It is easy to see that $M^D \models \mathcal{T}$ and $M^D \not\models \Phi$, which is however impossible as the triple $\mathcal{T}$, D, Φ was a counterexample for FC. $\square$

Proof of Lemma 4: 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 F (finite or not) it holds that $F \models \bar{\Psi}$ if and only if $Chase(F, \mathcal{T}) \models \Psi$.

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

Since $\mathcal{T}_0$ is joinless, we know, from Theorem 2, that there exists a finite structure D' such that $D' \models \mathcal{T}_0, D$ but $D' \not\models \bar{\Psi}$. Notice that the pair $\mathcal{T}$, D' is weakly saturated.

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

OUTLINE

Sections III – X are devoted to the proof of Theorem 2.

In Sections III and IV we explain that if there existed any counterexample $\mathcal{T}$, D, Φ for FC, with $\mathcal{T}$ being a joinless theory, then there would also exist another counterexample $\mathcal{T}'$, D', Φ' satisfying additional assumptions. One of the assumptions is that $D' = \emptyset$.

Then, in Sections V – X we show that if $\mathcal{T}$ and Φ satisfy the assumptions from Sections III and IV, then the triple $\mathcal{T}$, $\emptyset$, Ψ cannot be a counterexample. The general architecture of this part of the proof is described in Section V.

III. Some trivial simplifications

Nothing deep happens in this Section. We are just cleaning our desk before the real work starts.

We will call a joinless theory $\mathcal{T}$ *clean* if:

- heads of the rules 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;
- each rule from $\mathcal{T}$ is either a Datalog rule of the form
  (♣1:) $Q(\bar{x}) \Rightarrow Q'(\bar{y})$, with $\bar{y} \subseteq \bar{x}$, where by $\bar{y} \subseteq \bar{x}$ we mean that each element of the tuple $\bar{y}$ occurs in $\bar{x}$,
  or a TGD of the form
  (♣2:) $Q_0(\bar{x}_0) \wedge Q_1(\bar{x}_1) \Rightarrow \exists y\, Q(y, \bar{x}_0, \bar{x}_1)$;
- 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 (♣ 2), and projection predicates, occurring in the heads of rules of the form (♣ 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 $\mathcal{T}$, D, Φ is *clean* if $\mathcal{T}$ and Φ are clean, and D is empty.

Lemma 6: If there exists any counterexample $\mathcal{T}$, D, Φ for FC, with $\mathcal{T}$ being a joinless theory, then there exists also a clean counterexample.

This is a simple Lemma and we leave its proof as an exercise for the reader. Hint: for the condition that D is empty use the constantification technique from the proof of Lemma 3. Once D is empty, all the atoms in $Chase(\mathcal{T}, D)$ are produced by the rules of $\mathcal{T}$, so we can rewrite the program to make sure that repeated variables in the heads of the rules are unnecessary because they are remembered as a part of the name of a predicate. The argument, from Section II of representing queries over the new signature as finite disjunctions of queries over the old signature would need to be used here again. The remaining conditions are trivial.

IV. On the importance of family values

Let $\mathcal{T}$ be a clean theory, as described in Section III. From now on we will always have $D = \emptyset$. 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 $\mathcal{T}$.

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

Definition 7: By a 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 smaller than any given one are comparable. If a family ordering is a tree then 1 is the root of this tree.

If a family ordering is a tree, the youngest family member is the root of the tree. If Alice dines with her parents, we have a tree with Alice as the root, and two leaves. If Alice dines only with her boyfriend Bob, they form a family ordering consisting of two elements and no edges – this is why we need unions of trees rather than just trees. As our family orderings are tree-like there are never any siblings in the families.

But 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 8: A family pattern is a pair F, δ, where F is a family ordering and δ is a function assigning a number, from the set $\{1, 2, \ldots l\}$, to each pair j, i of elements of F such that $i <_F j$, where $<_F$ is the ordering relation on F (i is an ancestor[2] of j).

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

If j, i are members of some family, with pattern F, δ, and i is an ancestor of j then the value of $\delta(j, i)$ should be understood

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

as "how j addresses i". *Father* or *maternal grandmother* are, strictly speaking, not among the possible values for $\delta(j,i)$, but one could imagine that they are.

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

A remark about notations. For any syntactic object X by $Var(X)$ we will mean the set of all the variables in X. Symbol Q will be used to denote relation symbols. Letters P and R will denote atoms of variables. Letters A, B, C, D will denote atoms of elements of $Chase$. PP will be used for parenthood predicates and sometimes also for parenthood atoms. To denote elements of $Chase$ we will use a, b, c, d, while i, j, k will always be positions in atoms. F, G will be family orderings. For an atom $B = Q_{F,\delta}(b_1, b_2 ... b_k)$ (where $b_1, b_2 ... b_k$ are constants in $Chase$) we define a notation $B(i) = b_i$. The same applies for atoms of variables.

Definition 9: A clean joinless 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 family pattern, with the family ordering F having exactly k vertices.*
2) *If $R \Rightarrow P$ is a Datalog rule of $\mathcal{T}$, where $R = Q_{F,\delta}(\bar{x})$, $P = Q'_{G,\gamma}(\bar{y})$ and if $R(i) = P(j)$ and $R(i') = P(j')$ then:*
   (a) $i <_F i'$ *if and only if* $j <_G j'$
   (b) *if* $i <_F i'$ *then* $\delta(i', i) = \gamma(j', j)$
3) *If $R \wedge R' \Rightarrow \exists z\ P$ is a TGD from $\mathcal{T}$, where $R = Q_{F,\delta}(\bar{x})$, $R' = Q'_{F',\delta'}(\bar{y})$, arity of $Q_{F,\delta}$ is k and arity of $Q'_{F',\delta'}$ is k', then $P = Q''_{G,\gamma}(z, \bar{x}, \bar{y})$ for some $Q''_{G,\gamma}$ of arity $1 + k + k'$ and:*
   (a) $i <_G j \Leftrightarrow (j = 1 \wedge i > 1) \vee (i - 1 <_F j - 1 \wedge 1 < i, j \leq k + 1) \vee (i - k - 1 <_{F'} j - k - 1 \wedge k + 1 < i, j \leq k + k' + 1)$
   (b) *If* $j = 1$ *and* $1 < i \leq k + k' + 1$ *then* $\gamma(j, i) = i$. *If* $1 < j, i \leq k + 1$ *then* $\gamma(j, i) = \delta(i - 1, j - 1)$. *If* $k + 1 < j, i \leq k + k' + 1$ *then* $\gamma(j, i) = \delta'(i - k - 1, j - k - 1)$.

Definition 9 is written in an ugly low-level language, so let us use our running metaphor to explain what is going on.

Condition (1) says that relation atoms should be understood as families, and that the complete information about the family pattern should be a part of the predicate name.

To see the meaning of Condition (2) imagine that Alice used to live with her two ancestors: her father and her paternal grandmother, whom she called "granny". Then something very sad happened, and now she lives only with her grandmother. Condition (2a) says that the grandmother is still her ancestor and condition (2b) says that Alice still calls her "granny". Notice that since we only know how to name people by their positions at the table, and the table used by the smaller family is not the same as the one that was used by the bigger family, we need to formulate the condition in a complicated form. For example we say "If the one at position i (or i') before the projection and the one at position j (or, resp., j') after the projection are both the same person (i.e. if $R(i) = P(j)$, or resp. $R(i') = P(j')$) then the name i' used to address i (i.e. $\delta(i', i)$) is the same as the name j' uses to address j (i.e. $\gamma(j', j)$)".

Now Condition (3). There were two families R and R'. They somehow have a new child together, so they dine together, as one family. The way they are seated (i.e. the order of arguments in the rule head) is determined by condition (♣ 2). Condition (a) says that the birth of the new child's 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.

Lemma 10: If there exists any clean counterexample $\mathcal{T}$, $\emptyset$, Φ to FC then there also exists a clean counterexample $\mathcal{T}'$, $\emptyset$, Φ', with $\mathcal{T}'$ respecting family patterns.

Proof sketch: For the proof of this easy Lemma follow the ideas of the proof of Lemma 3. Introduce new predicate names – one new predicate for each old predicate and for each way of arranging its arguments into a family pattern. Then construct the theory $\mathcal{T}'$ by replacing each rule $T \in \mathcal{T}$ by a set of rules: one rule for each way of assigning the family patterns to the atoms in the body of T. As each old predicate can be now seen as a disjunction of new predicates, by distributivity UCQ Φ can be rewritten as an equivalent UCQ Φ' in the new context. □

OUTLINE

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:

Lemma 11: 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" 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 12: For two elements a, b of $Chase$ we will say that a and b are 0-equivalent (denoted $a \equiv_0 b$) if the parenthood atoms of a and b are atoms of the same predicate.

Suppose $a \equiv_0 b$, and A and B are parenthood atoms of a and b (resp.). Then, for each i, the elements $A(i)$ and $B(i)$ will be called respective parents *of a and b. For tuples $a_1, a_2, \ldots$*

a_s and b_1,b_2,… b_s by a_1,a_2,… $a_s \equiv_0 b_1$,b_2,… b_s we mean that $a_i \equiv_0 b_i$ for all $1 \leq i \leq s$.

The next lemma can be easily proved by induction on the structure on $Chase$. It 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 13: 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 $\delta(j', j) = \gamma(\delta(i, j'), \delta(i, j)$.*

Now we have something slightly more complicated. The following lemma, which is is not going to be needed before Section X, is where the importance of family patterns is seen:

Definition 14: 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 $\bigcap_{i\in\mathcal{I}}\{j : \neg(j \leq_F i)\}$.

$PY(\mathcal{I})$ (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 15 (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 incomparable by $<_F$, positions in F and let $b_1, b_2, \ldots b_s$ be equal to $A(i_1), A(i_2), \ldots A(i_s)$ respectively. Suppose $d_1, d_2, \ldots d_s$ is another tuple of elements of $Chase$ such that $b_1, b_2, \ldots b_s \equiv_0 d_1, d_2, \ldots d_s$. Then there exists an atom $C = PP_{F,\delta}(c, \bar{c})$, such that:

1) *$Chase \models C$;*
2) *$d_1, d_2, \ldots d_s$ equal $C(i_1), C(i_2), \ldots C(i_s)$ respectively;*
3) *if $j \in PY(\mathcal{I})$ then $A(j) \equiv_0 C(j)$;*

Lemma 15 (which is proved in Appendix A) 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. Remember that the fact that $b \equiv_0 d$ does not imply that the respective parents of b and d are 0-equivalent, and it follows easily from Lemma 13 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 16: If $i_k \in \mathcal{I}$ and $j <_F i_k$ then $A(j)$ and $C(j)$ are respective parents of b_k and and d_k (where the notations are like in Lemma 15). □

OUTLINE

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

V. General scheme of the proof. The first little trick

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

Property 17: (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, that a sequence M_n, satisfying Property 17 (i), (ii) is constructed. Then:

Definition 18: A formula Φ will be called M-true if $M_n \models \Phi$ for each $n \in \mathbb{N}$.

Lemma 19 (First Little Trick): If Φ is an M-true UCQ then there exists a disjunct of Φ which is M-true.

Proof: By Property 17 (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. □

The rest of the paper is organized as follows:

In Section VI the sequence M_n is defined. In Section VII we present our second little trick, which is the main engine of the proof. In the very short Section VIII a trivial case of cyclic queries (whatever it means) is considered. In Section IX we define a normal form of a conjunctive query and use our two little tricks to prove:

Lemma 20 (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*
$(**)$ *$Chase \models (\beta \Rightarrow \phi)$.*

Then, in Section X we prove:

Lemma 21 (The Lifting Lemma): If a clean CQ ϕ is in the normal form and $M_0 \models \phi$ then $Chase \models \phi$.

Assuming existence of a sequence M_n, satisfying Property 17, and assuming Lemmas 20 and 21 we can now present the main body of **Proof of Theorem 2:**

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. $\square$

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 20 and 21 to be true. **The real work begins now.**

VI. The canonical models M_n

In this section we define an infinite sequence of finite models M_n, which will "converge" to $Chase$.

Definition 22: By the 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* $H^{n+1}(a) = \{a\} \cup \bigcup_{b \in H^1(a)} H^n(b)$.

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

Definition 23: 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 24: For two elements a, b of $Chase$ *and for* $n \in \mathbb{N}$ *by* $a \simeq^n_0 b$ *we mean that* $a \equiv_0 b$ *and a and b have the same n-color. By* $a \simeq^n_{k+1} b$ *we mean that* $a \simeq^n_0 b$ *and that* $a' \simeq^n_k b'$ *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. 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 25: 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 VIII. The real message to be remembered is that if $a \equiv_{n+1} b$ then the parenthood atoms of a and b are atoms of the same predicate and all the respective parents of a and b are pairwise n-equivalent. As an exercise the reader may want to prove that n-equivalent tuples of parents always produce n-equivalent children. It is also very easy to notice that:

Lemma 26: $\equiv_n$ *is an equivalence relation of finite index.*

Now the next definition hardly comes as a surprise:

Definition 27: $M_n = Chase/\equiv_n$*. Relations on* M_n *are defined, in the natural way, as minimal relations such that the quotient mapping is a homomorphism.*

Since being $(n+1)$-equivalent implies being n-equivalent (which is another easy exercise) the sequence of structures M_n satisfies Property 17 (ii). It is also easy to see that it satisfies Property 17 (i) (the assumption that $\mathcal{T}$ is joinless needs to be used in the proof):

Lemma 28: $M_n \models \mathcal{T}$ *for each* $n \in \mathbb{N}$.

The second sentence of Definition 27 sounds complicated. We hope the next definition and lemma will shed some light:

Definition 29: For a conjunctive query ϕ *let* $Occ(\phi)$ *be the set of all variable occurrences in* ϕ*. More precisely,* $Occ(\phi) = \bigcup_{R \in \phi}(\{1, 2 \dots arity(R)\} \times \{R\})$.

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 30: $M_n \models \phi$ *if and only if there exists an n-evaluation of* ϕ. $\square$

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 30 really says is that $M_n \models \phi$ if and only if $Chase \models \phi'$.

Next two lemmas can now be seen as exercises. But at some point we will need them:

Lemma 31: 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* $j <_F i$ *be a position in R. Then the position* $\delta(i, j)$ *exists in the atom P.*

Hint: This is not *a priori* clear that it must always be so. For example one could imagine R being a high arity atom, with the position i close to its root, so that the number of such positions j in R that $j <_F i$ is higher than the arity of P.

This would however contradict the assumption that ϕ is M-true. To see it use the fact that $M_0 \models \phi$, Lemma 30 and Lemma 13. $\square$

Lemma 32: 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' *and* i'' *be such positions in P that* $i' = \gamma(j, j')$ *and* $i'' = \gamma(j, j'')$*. Then* $i' <_F i''$ *and* $\delta(i'', i') = \gamma(j'', j')$

Hint: 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 13 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. $\square$

VII. The second little trick

As we said in Section V, 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, so please try to have fun:

Lemma 33 (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 31). 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 13, 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 13 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 the proof of Lemma 33:

Proof of Lemma 33: 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 29 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 29, we know (by Lemma 13) 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. $\square$

VIII. Cyclic 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 ordering from individual atoms to the conjunctive query they form and study carefully the new ordering.

Definition 34: Let ϕ be a CQ.

1) *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 $x \rightarrow_\phi y$.*
2) *ϕ is called* acyclic *if $\rightarrow_\phi$ is a partial order[3] on $Var(\phi)$ (which means that it is antisymmetric). Otherwise it is cyclic.*

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

Lemma 35: If ϕ is a cyclic query consisting of k atoms, then $M_{k+1} \not\models \phi$. So a cyclic query is never M-true.

Proof of the lemma is left as an exercise for the reader. Hint: notice that, by Definition 23 and the first claim of Lemma 13, $M_{k+1} \models \phi$ would imply the existence of an element of $Chase$ having an $k+1$-equivalent element in its $k+1$-history, which is impossible, for coloring reasons. This was the only place where we needed to think about colors.

It follows from Lemma 35 that in the proof of the Normal Form Lemma we only need to consider acyclic queries.

IX. Acyclic queries and the normal form

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

Definition 36: Call a variable $x \in Var(\phi)$ important *if $x = P(1)$ for some atom[4] P in ϕ. Otherwise x is called ordinary.*

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 14.

Definition 37: • *For an atom $P = Q_{F,\delta}(\bar{t})$ let $\mathcal{I}(P)$ denote the set of such non-root positions i in P that the variable*

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

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

$P(i)$ 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 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 38: 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 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). Call this atom 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 33. So one can imagine now, how we are going to prove Lemma 20 – 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 33 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 X) for the proof of the Lifting Lemma.

For the (mostly boring and syntactical) details of the proof of Lemma 20 see Appendix B.

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 39: 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 40: Let ϕ be an acyclic 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 $y \in top.var(PP_x)$.*

Lemma 41: Let ϕ be an acyclic 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 $z \rightarrow_\phi x$.*

For the (nice, short, ideological) proofs of Lemmas 40 and 41 see Appendix C.

X. 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 VI, $M_0 \models \psi$ 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 42: 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 43: 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 Let $PP_x = P_{\langle F,\delta \rangle}(x, \bar{x})$. be, as usually, the parenthood atom of x in ψ. Let S' be the important ideal generated by x and S.

Lemma 44: 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 40 and 41. 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 41, this would imply that $z \rightarrow_\psi x$, which is a contradiction. The proof of the other inclusion is left as an easy exercise. □

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

Definition 45: Suppose f is a 0-evaluation, $f' : Occ(\psi) \rightarrow Chase$ is any function, and P is an atom in ψ. We say that f' is P-similar to f if:

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

Lemma 46: If f is a 0-evaluation and f' is P-similar to f then f' is also a 0-evaluation. □

Let S, S' and x be as above. In view of Lemma 44 1) and Lemma 46, due to an induction argument, in order to prove Lemma 21, is now only remains to show:

Lemma 47: 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 45 is satisfied.

We will now define $f'(PP_x)$. Then we will make sure that the second and third conditions from Definition 45 hold, so f' is indeed a 0-evaluation. Finally we will notice that f' is faithful with respect to S'.

Let $\mathcal{I}(PP_x) = \{i_1, i_2 \dots i_s\}$, where $\mathcal{I}(PP_x)$ is the set of maximal important non-root positions, as in Definition 37. Let $y_1, y_2, \dots y_s$ be the important variables in positions $i_1, i_2 \dots i_s$ 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 this means that we are now in the situation of Lemma About the Future (Lemma 15), where $A = f(PP_x)$.

So let C be as in Lemma 15. 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 15) 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 45. Now we need to make sure 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: $i \in \{i_1, i_2 \dots i_s\}$ 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 13.

In order to finish the proof of the Lemma we still need to notice that f' is S'-faithful. The atoms described by Definition 42 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 42 is trivially satisfied. □

XI. Remark about guarded TGDs

The proof in Sections V-X can be also read as a new proof of the FC property for guarded TGDs [BGO10]. The only difference is that, in order to construct M_n, it is not enough, in the guarded case, to remember by which rules last n generations of parents of an element were born, but also what other atoms are true about the elements. That is why a condition "and the n-histories of a and b are isomorphic" needs to be added to Definition 25. On the other hand, all the family orderings we would need to consider in the guarded case are total orderings, which significantly simplifies everything – for example the Technical Condition of the normal form comes for free, being implied by acyclicity.

XII. References

[BGO10] V. Barany, G. Gottlob, and M. Otto; *Querying the guarded fragment*; Proceedings LICS 2010 pp. 1–10, 2010.

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

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

[GM12] T. Gogacz, J. Marcinkowski; *On the BDD/FC conjecture*; submitted.

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

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

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

XIII. Appendix A. Proof of the Lemma About the Future.

We will consider (a fragment of) the derivation tree of the atom A in $Chase$, which we will call $\mathcal{D}$. This is how $\mathcal{D}$ is defined:

- Atom A is a root of $\mathcal{D}$ (and thus an inner node of $\mathcal{D}$). Positions $i_1, i_2, \dots i_s$ are *painted* in A.
- Suppose an atom $B = Q_{G,\gamma}(e, \bar{e})$ is an inner node of $\mathcal{D}$ and $Chase \models B$. 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 $X = Q_{G,\gamma}(\bar{x}, x)$ Then B' and B'' are nodes of $\mathcal{D}$. If position i was painted in B and $X(i) = X'(j)$ then position j is painted in B' (reminder: $X(i)$ and $X'(j)$ are variables, at position i in the atom X and position j in X'). Similarly, if position i was painted in B and $X(i) = X''(j)$ then position j is painted in B''. The case when B was derived by a projection rule $X' \Rightarrow X$ is handled analogously.
- A node 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, called a painted leaf. All other nodes of $\mathcal{D}$ are inner nodes.

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 $b_1, b_2, \dots b_s$". 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 $\mathcal{D}$, consider 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 B is also 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, then the parenthood atom of d_i is the respective leaf of $\mathcal{D}'$.
- If B is an inner node of $\mathcal{D}$, being a result of applying some rule from $\mathcal{T}$ to atoms B' and B'' (or just to B') then the same rule is used to create the respective atom in $\mathcal{D}'$.

Clearly, $\mathcal{D}'$ is also a part of $Chase$. Notice however that if $\mathcal{T}$ was not joinless, the last step would not always be possible in $Chase$.

Now, the atom in the root of $\mathcal{D}'$ is the C from the Lemma. $\square$

XIV. Appendix B, part one. The query which is the normal form of ϕ.

Outline

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 20 and Definition 38.

The definition of β itself (Definition 52) is quite natural and not very complicated. The really technical part begins right after Definition 52, 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 38 can now be expressed:

(♡) 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 $\theta$ as $\psi$;
    /* 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 $\xi$ by $\mathcal{P}_\xi(i')$
(in other words $\xi := \xi[\mathcal{P}_\xi(j')/\mathcal{P}_\xi(i')]$);
}
forget that $\theta$ was $\psi$;
    /* So that we can use $\theta$ somewhere else */
remove the repeating atoms from $\xi$;
return $\xi$ 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 33 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 48: 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. $\square$

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 49: 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 50: 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 51: 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. $\square$

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 52: $\beta = u(c(\phi))$.

Lemma 53: 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 53 is proved then of course also the proof of Lemma 20 will be finished.

OUTLINE

We defined the query β. What remains to be done is showing that β is indeed the normal form of ϕ. The main proof technique is a patient syntactical analysis of the unifications that led to β.

XV. Appendix B, part two. Proof of Lemma 53

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

Before we show Lemma 53 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 53. 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 54: 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 55: $u(\beta_0) = \beta$;

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 48) . $\square$

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} (as "weakly unified"). 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 56: 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 57: 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 58: For an important variable x let $layer(x) = \{j \in \mathcal{P}_{ord} : nearest.var(j) = x\}$.

Of course:

Lemma 59: 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 37 in our new language).

Notice that while the set of top positions could again be equivalently defined with respect to β instead of β_0, this no longer seems to be true for the set of top variables. Top variables in β_0 are the fresh ordinary variables in top positions of $beta_0$. But then the unifications come, and many things can happen, as we are going to see.

Now we are ready for:

The weak unification procedure:

```
ξ := $\beta_0$;
to-be-considered := $Var_{imp}(\beta_0)$

while to-be-considered $\neq \emptyset$
do:
{                                                          /* ◇ */
Let $x$ be a minimal, with respect to the ordering $\rightarrow_\beta$, variable
in the set to-be-considered;
   /* See! Here is where we need to know $\beta$.               */
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 $\xi$ by the variable $\mathcal{P}(i')$;
   /* Call the above the "unification step"                     */
Remove the variable $x$ from to-be-considered;
}
Return $\xi$ as $\beta_{wu}$.
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 60: Each time the control passes the point marked with $\diamondsuit$, 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.* $\square$

The meaning of the last lemma is that the substitution in the unification step is just a renaming of one variable occurance – 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 61: If j is a top position in $\mathcal{P}$ then $\mathcal{P}_{\beta_0}(j) = \mathcal{P}_{\beta_{wu}}(j)$

Lemma 61 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 XIV) 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 62: 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. $\square$

Now the last lemma we need to show in order to finish the proof of Lemma 53:

Lemma 63: The query β_{wu} satisfies the Ideological Condition. In consequence, $\beta_{wu} = \beta$.

Proof: We know from Lemma 62 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 32 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. $\square$

This ends the proof of Lemma 53 and of Lemma 20.

XVI. Appendix C. Proofs of Lemmas 40 and 41

Proof of Lemma 40A. 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 $PP_z(j) = z$. 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 Remark 31 here, and this is where the assumption that ϕ is M-true was needed. $\square$

Proof of Lemma 40B. 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. $\square$

Proof of Lemma 41. Claim A. follows directly from Lemma 40B and from the Technical Condition. Claim B. follows directly from A. $\square$