ssdnm
środowiskowe
studia doktoranckie
z nauk matematycznych

Tomasz Gogacz

Uniwersytet Wrocławski

On the BDD/FC Conjecture

Praca semestralna nr 3

(semestr zimowy 2010/11)

Opiekun pracy: Jerzy Marcinkowski

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

On the BDD/FC Conjecture *

[Extended Abstract]

Tomasz Gogacz
Institute of Computer Science
University of Wroclaw
Poland
gogo@cs.uni.wroc.pl

Jerzy Marcinkowski
Institute of Computer Science
University of Wroclaw
Poland
jma@cs.uni.wroc.pl

ABSTRACT

Bounded Derivation Depth property (BDD) and Finite Controllability (FC) are two properties of sets of datalog rules and tuple generating dependencies (known as Datalog$^\exists$ programs), which recently attracted some attention. We conjecture that the first of these properties implies the second, and support this conjecture by some evidence proving, among other results, that it holds true for all theories over binary signature.

Categories and Subject Descriptors

F.4.1 [**Theory of Computation**]: Mathematical Logic and Formal Languages:Mathematical Logic; H.2.4 [**Database Management**]: Systems - Relational databasesrule-based databases, query processing

Keywords

Bounded Derivation Depth, Tuple Generating Dependencies, Finite Controllability

1. INTRODUCTION

Tuple generating dependencies (TGDs), recently known also 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, and represented by some theory $\mathcal{T}$ (or a Datalog$^\exists$ program), consisting of existential TGDs and plain datalog rules, but it is assumed that our knowledge of the database facts is incomplete (this is known as the open-world assumption).

In this paradigm, for a database instance D (understood here as a set of facts – atomic formulas), the semantics of D, in presence of $\mathcal{T}$ is defined as the (*) set of all the database instances $\bar{D}$ which are supersets of D and satisfy $\mathcal{T}$. In other words, we are interested whether, for a given query[1] Φ, it holds that $\mathcal{T}, D \models \Phi$.

The problem is of course undecidable in general, so some restricted classes of theories are being studied. For example Linear Datalog$^\exists$ programs, which consist of TGDs which, as the body, have a single atomic formula, were studied in [R06], Guarded Datalog$^\exists$, being an extension of Linear (see Section 5.6 for more details) was analyzed in [BGO10] and Sticky Datalog$^\exists$ programs were introduced (in two flavors) in [CGP10] and [CGP10+/-].

As it turns out, decidability of query answering is not that hard to prove for theories from these classes. But there are good reasons why we would like to have more than just decidability. The desired properties of $\mathcal{T}$ are (among others) Bounded Derivation Depth property (BDD) and Finite Controllability (FC).

The theory $\mathcal{T}$ has the Finite Controllability property (for short: "$\mathcal{T}$ is FC"), if the expression "the set of all database instances" in the definition (*) above can be equivalently replaced by, more natural from the database point of view, "the set of all finite database instances". To be more precise:

Definition 1. $\mathcal{T}$ is FC if for each database instance D and each query Φ, if $\mathcal{T}, D \not\models \Phi$ then there exists a database instance $\mathcal{M}$ such that $\mathcal{M} \models D, \mathcal{T}$ but $\mathcal{M} \not\models \Phi$.

The difficult technical results in [R06] (solving an old problem stated in [JK84]) and in [BGO10] concern Finite Controllability of, respectively, Linear and Guarded Datalog$^\exists$. (Actually, the result in [R06] is stated in terms of Inclusion Dependencies rather than TGDs, which, in this context, is another language to talk about the same thing.) The question if the Sticky Datalog$^\exists$ programs are FC was left as an open problem in [CGP10] and was solved, positively, in [GM12].

The theory $\mathcal{T}$ has the Bounded Derivation Depth property (or just: "$\mathcal{T}$ is BDD") if it admits positive first order query rewriting. In other words:

Definition 2. $\mathcal{T}$ is BDD if for each query Φ there exists a union of conjunctive queries Φ' such that for every database instance D the equivalence : $\mathcal{T}, D \models \Phi$ if and only if $D \models \Phi'$ holds. (This is the definition we actually need here, but not the standard one. See Section 1.1 for an equivalent, more standard definition.)

*Supported by Polish Ministry of Science and Higher Education NCN grant N N206 371339.

PODS'13, June 22–27, 2013, New York, New York, USA.

[1]Whenever we say "query" in this paper we mean a conjunctive query without negation. Whenever we say "TGD" we mean a single-head tuple generating dependency.

This means that instead of computing the answer to Φ over the infinite set of databases having D as their subset (or instead of computing the answer to Φ over the infinite database $Chase(D, \mathcal{T})$ – see Section 1.1) it is enough to compute the answer to Φ' over the known finite database D. There is no need to explain how desirable in the database context BDD is, so many of the good classes of Datalog$^\exists$ programs (including Linear Datalog$^\exists$ and Sticky Datalog$^\exists$) are tailored to have this property. It is worth mentioning that, while of course BDD is an undecidable property of $\mathcal{T}$, still in all practical situations we know about, proving the statement "all the programs from class $\mathcal{C}$ are BDD" is an easy exercise (if it is true). This is in sharp contrast to Finite Controllability which is, as we mentioned above, typically quite hard to prove.

BDD is typically easy to prove. FC is hard to prove. But each time we had a class of BDD theories, finally we were able to show that this class is also FC. This leads to a conjecture we would like to state here:

CONJECTURE 1 (THE BDD/FC CONJECTURE).
If some theory $\mathcal{T}$, being a set of existential TGDs and plain datalog rules, is BDD then it is also FC.

The evidence we support our conjecture with is:

THEOREM 1 (THE MAIN RESULT OF THIS PAPER).
Conjecture 1 is true for programs over binary signature.

The proof of Theorem 1 is the main technical contribution of this paper and is, as we believe, quite difficult. It is presented in Section 3 but relies on a system of tools developed in Sections 2 and 4.

Finally, in Section 5 we discuss the possible applications of our tools and their limitations, In subsection 5.1 we show that Theorem 1 can be extended also to quite a wide class of non-binary theories (see Theorem 3). In subsection 5.4 we explain however, why our techniques do not seem to extend to the proof of Conjecture 1 in general. In subsection 5.6 we show how Guarded Datalog$^\exists$ programs can be seen as binary programs, and how our techniques can be easily applied in this context.

1.1 TGDs and Chase – preliminaries

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, Q is a relation symbol, $\bar{x}, \bar{y}$ are tuples of variables and $\bar{y} \subseteq \bar{x}$ (see Section 5.3 for a comment on the multi-head TGDs). The universal quantifier in front of the formula is usually omitted.

Finite sets consisting of existential TGDs and plain datalog rules will be called theories.

For a theory $\mathcal{T}$ and a database instance D we denote by $Chase^1(D, \mathcal{T})$ the result of the following operation. For each tuple $\bar{x}$ in D satisfying a body of an rule $t_i = \forall \bar{x}\ (\Phi(\bar{x}) \Rightarrow \exists y\ Q(y, \bar{y}))$ form $\mathcal{T}$, such that there is no $y \in D$ satisfying $D \models Q(y, \bar{y}))$, we simultaneously add new constant $c_{t_i, \bar{x}}$ into database and an atom $Q(c_{t_i, \bar{x}}, \bar{y})$.

Then define $Chase^{i+1}(D, \mathcal{T})$ as $Chase^1(Chase^i(D, \mathcal{T}), \mathcal{T})$ and by Chase denote $\bigcup_i Chase^i(D, \mathcal{T})$, which is the least fixpoint of the $Chase^1$ operator.

Clearly, we have $Chase(D, \mathcal{T}) \models D, \mathcal{T}$, but there is no reason to think that $Chase^i(D, \mathcal{T}) \models \mathcal{T}$ for any $i \in \mathbb{N}$. Note that the chase we consider in this paper is the non-oblivious one – new elements are only created if needed, as opposed to the blind Chase, which creates a new witness each time it is demanded.

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

A set $\mathcal{T}$ of TGDs is usually said to have Bounded Derivation Depth property if for each query Ψ, there is a constant $k_\Psi \in \mathbb{N}$, such that for each database instance D if $Chase(D, \mathcal{T}) \models \Psi$ then $Chase^{k_\Psi}(D, \mathcal{T}) \models \Psi$. It is easy to see ([CGT09]) that this definition of the BDD property is equivalent to Definition 2.

Notations. When we say that $\mathcal{C}$ is a structure we may mean both, the set of elements and the set of atoms of $\mathcal{C}$. If we feel this may cause confusion we write $\mathrm{Dom}(\mathcal{C})$ for the set of elements of $\mathcal{C}$. By $\mathcal{C} \models R$ (or $\mathcal{C} \models \psi$) we mean that an atom R (or a formula ψ) is true in $\mathcal{C}$. By $\mathcal{C}_1 \models \mathcal{C}_2$ we mean that each atom of $\mathcal{C}_2$ is an atom of $\mathcal{C}_1$. For a structure $\mathcal{C}$ and a set A (or a signature Σ) by $\mathcal{C} \restriction A$ (resp. by $\mathcal{C} \restriction \Sigma$) we mean the structure consisting of such atoms $R(\bar{a})$ that $\mathcal{C} \models R(\bar{a})$ and $\bar{a} \subseteq A$ (resp. $R \in \Sigma$). For a structure $\mathcal{C}$ over some signature Σ by $\mathcal{C}_{con}$ we mean, depending on context, the set of elements of $\mathcal{C}$ which are interpretations of constants from Σ or the structure $\mathcal{C} \restriction \mathcal{C}_{con}$. Similarly, by $\mathcal{C}_{non}$ we mean the set of elements of $\mathcal{C}$ which are not incarnations of constants from Σ or the structure $\mathcal{C} \restriction \mathcal{C}_{non}$.

In paper we consider only Boolean conjunctive queries. Sometimes free variables are omitted to keep the notation light. In such cases one should treat them as existentially quantified. (For example, for a query $\Phi(\bar{x})$ the term $M \models \Phi$ should be read as $M \models \exists \bar{x}\ \Phi(\bar{x})$).

2. TYPES AND PROJECTIONS

2.1 The main ideas and the structure of the proof

In order to prove Theorem 1 we need to construct, for a given BDD theory $\mathcal{T}$ over a binary signature, for a conjunctive query $Q(\bar{x})$ and for a finite structure D, such that $Chase(D, \mathcal{T}) \not\models \exists \bar{x}\ Q(\bar{x})$, a new finite structure M, such that $M \models D, \mathcal{T}$ but $M \not\models \exists \bar{x}\ Q(\bar{x})$.

Such M will always contain a substructure M' being a homomorphic image of $Chase(D, \mathcal{T})$. This M' is easy to construct inside M (if we had M): start from D, which is a substructure of both M and $Chase(D, \mathcal{T})$, and then mimic, inside M, all the applications of rules that led to the construction of $Chase(D, \mathcal{T})$.

Isn't M' itself always the finite model we are looking for? No, because it may very well happen that by identifying elements of $Chase(D, \mathcal{T})$ the homomorphism (call it q) created new instances of the bodies of the rules of $\mathcal{T}$ in M', leading to the situation when applications of rules is possible that were not applied in $Chase(D, \mathcal{T})$. For example suppose that $E(x, y), R(y, z) \Rightarrow U(y)$ is a rule of $\mathcal{T}$ and $Chase(D, \mathcal{T}) \models E(a, b), R(b', c)$, but $Chase(D, \mathcal{T}) \not\models U(b)$ and $Chase(D, \mathcal{T}) \not\models U(b')$. Suppose also that $q(b) = q(b')$. Then the fact $U(q(b))$, which may not be homomorphic image of any fact in $Chase(D, \mathcal{T})$, is provable in M. This can lead to a process in which an answer to Q is built in M, something we need to avoid. This can also lead to infinite

chase, while we want M to be a finite structure. Let us illustrate this problem with one more example:

Example 1. Let $\mathcal{T}$ be a theory consisting of three rules:
$E(x,y) \Rightarrow \exists z\ E(y,z)$
$E(x,y), E(y,z), E(z,x) \Rightarrow \exists t\ U(x,t)$
$U(x,y) \Rightarrow \exists z\ U(y,z)$
and a database instance $D = \{E(a,b)\}$.
Then $Chase(D,\mathcal{T})$ is an infinite $E-chain$, beginning with a and b. Consider M' consisting of elements a, b and c and atoms $E(a,b)$, $E(b,c)$ and $E(c,a)$. Then M' is a homomorphic image of $Chase(D,\mathcal{T})$, but is not itself a model of $\mathcal{T}$ – the last rule, which was never triggered when $Chase(D,\mathcal{T})$ was built, can be used in M'. Moreover, it is easy to see that $Chase(M',\mathcal{T})$ is an infinite structure.

The idea of the construction we present in this paper is to make sure that some sort of first order type of each a in $Chase(D,\mathcal{T})$ is always the same as the type of its image $q(a)$ in the homomorphic image of $Chase(D,\mathcal{T})$. The definition of the type should be tailored in such a way that such preservation of types implies that no harmful new applications of rules from $\mathcal{T}$ for $q(a)$ exist in M.

In Section 2 we develop a sort of theory of positive types and their preservation. We built a framework in which the Main Lemma (Lemma 2) can be expressed. In Section 3 this Main Lemma is used to prove Theorem 1. In Section 4 we prove the Main Lemma. Sections 3 and 4 are independent and can be read in any order.

The two most important technical tricks of the paper can be found in proofs of Lemma 5 (in Section 3) and Lemma 11 (in Section 4). In the proof of Lemma 5 we show how the assumption that the theory $\mathcal{T}$ is BDD can be used. The trick in the proof of Lemma 11 relies on the construction, presented already in Section 2.3, where we construct not just one finite structure, but an infinite sequence of finite structures M_n that in some sense converge to $Chase(D,\mathcal{T})$. Then the idea is that if some query Ψ is true in M_{n+1} (which we do not like, as we do not want too many queries to be true in the finite structures we construct) then a query Φ, being a "one-step normalized" version of Ψ may not be true in M_{n+1} but it will be true in M_n. This then implies that if Ψ is true in all M_n then its "normal form" also is. This "converging to the Chase" trick is also used in our another paper [GM13] and we believe it can have further applications.

2.2 Positive types

Definition 3. Let $\mathcal{C}$ be a relational structure over signature Θ. Let $e \in \mathcal{C}$ and let n be a natural number. We define $ptp_n(\mathcal{C}, e, \Theta)$ (which reads "positive n-type of e in $\mathcal{C}$ over Θ") as the set of all such conjunctive queries $\Psi(\bar{x}, y)$ that:

- $|\bar{x}| < n$,
- all relations (and constants) used in Ψ are in Θ
- $\mathcal{C} \models \Psi(\bar{x}, e)$.

We assume that equality belongs to each Θ, which means that atoms of the form $x = c$ (but not of the form $x \neq c$), where x is a variable and c is a constant from Θ, are allowed in the queries.

Example 2. Let $\mathcal{T}$, D and M' be like in Example 1, and let Θ consist of E and U. Then $ptp_2(Chase(D,\mathcal{T})), a, \Theta)$ equals $ptp_2(M', a, \Theta)$, and each of them consists of the same two queries: $E(x,y)$ and $E(y,x)$. But $ptp_3(Chase(D,\mathcal{T})), a, \Theta)$ does not equal $ptp_3(M', a, \Theta)$: the query $E(y,x_1) \wedge E(x_1,x_2) \wedge E(x_2,y)$ belongs to the second of those two types but not to the first one.

Remark 1. Notice that if c is a constant[2] from Θ, and if $a \neq c$ is any other element of $\mathcal{C}$, then $ptp_n(\mathcal{C}, c, \Theta) \neq ptp_n(\mathcal{C}, a, \Theta)$ for each $n \geq 1$. This is because we allowed a query of the form $y = c$, which belongs to $ptp_1(\mathcal{C}, c, \Theta)$, but not to $ptp_1(\mathcal{C}, a, \Theta)$.

All the signatures under consideration are finite, so the number of possible conjunctive queries with at most n variables is finite. In consequence the number of positive n-types is finite, for a given n.

Let us remark here that our positive n-types carry much less information than the standard first order types (in the sense of Geifman or Hanf). Take for example a structure $\mathcal{C}$, over the signature $\Theta = \{R, E\}$, consisting of elements a,b,c,d,e, and atoms $R(a,b)$, $R(a,c)$, $E(a,c)$, $E(d,e)$, $R(d,e)$. Then $ptp_2(\mathcal{C}, a, \Theta) = ptp_2(\mathcal{C}, e, \Theta)$. But the first order 2-types of a and e differ: consider for example the formula: $\psi(x) = \exists z, y\ R(x,y) \wedge E(x,z) \wedge y \neq z$. Then $\mathcal{C} \models \psi(a)$ but $\mathcal{C} \not\models \psi(d)$.

2.3 How the finite structures are born

Definition 4. Let d and e be two elements of $\mathcal{C}$. We define $d \equiv_n e$ if and only if $ptp_n(\mathcal{C}, d, \Theta) = ptp_n(\mathcal{C}, e, \Theta)$.

Notice that both the relation $\equiv_n$ and the structures $M_n(\mathcal{C})$ (as defined below) depend on Θ, and the signature should be added as a parameter in Definitions 4 and 5. We will try to avoid confusion while keeping the notation light, but when really needed we will include the parameter, writing $M_n^{\Theta}(\mathcal{C})$ instead of $M_n(\mathcal{C})$.

Definition 5. For a relational structure $\mathcal{C}$ define $M_n(\mathcal{C})$ as a relational structure whose set of elements is $\mathcal{C}/\equiv_n$, and such that $M_n(\mathcal{C}) \models R(< [a_i]_{\equiv_n} >_i)$ iff $\forall i\ \exists b_i \in [a_i]_{\equiv_n}$ such that $\mathcal{C} \models R(< b_i >_i)$.

In other words, the relations in $M_n(\mathcal{C})$ are defined in the natural way, as minimal (with respect to inclusion) relations such that the quotient mapping $q_n : \mathcal{C} \longrightarrow M_n(\mathcal{C})$ is a homomorphism.

We usually imagine q_n as a projection[3], so that the atoms in $M_n(\mathcal{C})$ are projections of atoms in $\mathcal{C}$.

Clearly each $M_n(\mathcal{C})$ is a finite structure.

LEMMA 1. *If $q_n(d) = q_n(e)$ then $q_{n-1}(d) = q_{n-1}(e)$. The structure $M_{n-1}(\mathcal{C})$ is a homomorphic image of $M_n(\mathcal{C})$.*

For the proof of the first claim notice that it follows from Definition 3 that if the positive n-types of two elements are equal then their positive $(n-1)$-types are also equal. The second is an easy exercise in basic universal algebra.

(♠ 1.) The function q_n, as defined above, has $\mathcal{C}$ as its domain. It will however be convenient to be able to write $q_n(a)$

[2]Strictly speaking, we mean a value of this constant in $\mathcal{C}$, but we are not always going to make this distinction.
[3]"Projection" in the geometric sense not the database sense.

also for $a \in M_{n+1}(\mathcal{C})$. In such a case $q_n(a)$ will be defined as $q_n(b)$, where $b \in \mathcal{C}$ is any element such that $q_{n+1}(b) = a$. It follows from Lemma 1 that the value of $q_n(a)$ does not depend on the choice of b.

We defined a canonical way of building finite structures. But is there any chance that they really resemble the original infinite structure? What we are particularly interested in is what happens to the positive m-types of elements of $\mathcal{C}$. Are they preserved by q_n? It is easy to see that we always have $ptp_m(\mathcal{C}, e, \Theta) \subseteq ptp_m(M_n(\mathcal{C}), q_n(e), \Theta)$. But can the inclusion be replaced with equality? Is the positive m-type of $e \in \mathcal{C}$ always the same as the positive m-type of $q_n(e)$?[4] Unfortunately this is not yet the case:

Example 3. Let $\Sigma = \{E\}$ and let $\mathcal{C}$ be the set $\{a_0, a_1, a_2 \ldots\}$ with $E(a_i, a_{i+1})$ for each i. Notice that the names of the elements a_i are not part of Σ, so they are invisible for the inhabitants of the structure, and the positive n-types of elements a_i and a_j, with $i \neq j$, are equal if and only if $i, j \geq n$. (Actually, not only the positive types of a_i and a_j are equal, but even their n-Gaifman neighborhood are isomorphic.) So $M_n^{\Sigma}(\mathcal{C})$ is a structure with elements $\{b_0, b_1, b_2, \ldots b_n\}$, with $E(b_i, b_{i+1})$ for each $i < n$ and with $E(b_n, b_n)$. Clearly, $q_n(a_n) = b_n$. But the positive 1-type of b_n in $M_n(\mathcal{C})$ contains the query $\exists y R(y, y)$, which is not in the positive 1-type of a_n in $\mathcal{C}$.

2.4 Colored structures

We are not quite happy with the quotient structure we got in Example 3. Too many elements of $\mathcal{C}$ are identified, and even a very small conjunctive query can easily see the difference between $\mathcal{C}$ and $M_n(\mathcal{C})$. But consider another example:

Example 4. Let Σ and $\mathcal{C}$ be like in Example 3. Let $\bar{\Sigma} = \Sigma \cup \{K_0, K_1, \ldots K_m\}$, where $K_0, K_1, \ldots K_m$ are unary predicates (colors) and let $\bar{\mathcal{C}}$ be like $\mathcal{C}$, but with each a_i satisfying also $K_{i \mod (m+1)}$.

Let $n > m$. Then the positive n-types of elements a_i and a_j, with $i \neq j$, are equal if and only if $i, j \geq n$ and $i = j \mod m+1$. So $M_n^{\bar{\Sigma}}(\bar{\mathcal{C}})$ will be a structure with elements $\{b_0, b_1, b_2, \ldots b_{n+m}\}$, with $E(b_i, b_{i+1})$ for each $i < n+m$ and with $E(b_{n+m}, b_n)$. It is not hard to see that now $ptp_m(\bar{\mathcal{C}}, a, \Sigma) = ptp_m(M_n^{\bar{\Sigma}}(\bar{\mathcal{C}}), q_n(a), \Sigma)$ for each element $a \in \bar{\mathcal{C}}$. The positive m-types of the elements of $\mathcal{C}$ are preserved by the quotient operation.

Notice however that the positive $(m+1)$-types are not preserved. This is because, unlike $\mathcal{C}$, the structure $M_n^{\bar{\Sigma}}(\bar{\mathcal{C}})$ contains a cycle of length $m+1$, which is easy to detect with a query with $m+1$ variables. If we want to preserve positive m-types for bigger numbers m we need to use more colors.

Notice also that if we took $n < m$ then we would get $ptp_m(\bar{\mathcal{C}}, a_n, \Sigma) \neq ptp_m(M_n^{\bar{\Sigma}}(\bar{\mathcal{C}}), q_n(a_n), \Sigma)$. This is because a_n would then be identified with all the elements a_{n+km+1} for $k \in \mathbb{N}$, and therefore the query

$$\exists x_1 \ldots x_{m-1} E(x_1, x_2) \wedge E(x_2, x_3) \wedge \ldots \wedge E(x_{m-1}, q_n(a_n))$$

would be satisfied in $M_n^{\bar{\Sigma}}(\bar{\mathcal{C}})$, while the query

$$\exists x_1 \ldots x_{m-1} E(x_1, x_2) \wedge E(x_2, x_3) \wedge \ldots \wedge E(x_{m-1}, a_n)$$

is not satisfied in $\bar{\mathcal{C}}$.

[4]Notice that we use two natural numbers here: n, which we imagine is big – the bigger it is the more similar $M_n(\mathcal{C})$ and $\mathcal{C}$ are, and m – the smaller it is the easier it is preserve the positive m-types.

The last example motivates the following definitions:

Definition 6. Each of the unary predicates K_h^l for some $h, l \in \mathbb{N}$ will be called a color, with the number h being called the hue of the color and the number l being called its lightness. The set of all colors will be denoted as $\mathcal{K}$.

So far we just defined an infinite set of unary predicates (with strange names, that we will need much later). Now a definition of coloring. A natural one:

Definition 7. For a structure $\mathcal{C}$ over a signature Σ by a coloring of $\mathcal{C}$ we will mean a structure $\bar{\mathcal{C}}$ over some finite signature $\bar{\Sigma}$ such that:

1. $\Sigma \subset \bar{\Sigma} \subset \Sigma \cup \mathcal{K}$
2. $\bar{\mathcal{C}} \restriction \Sigma = \mathcal{C}$
3. for each $a \in \bar{\mathcal{C}}$ there is exactly one color $K \in \mathcal{K}$ such that $\bar{\mathcal{C}} \models K(a)$.

where $\bar{\mathcal{C}} \restriction \Sigma$ is the structure $\bar{\mathcal{C}}$ restricted to the signature Σ.

2.5 Conservative structures

Definition 8. Let $\mathcal{C}$ be a structure and let $m, n \in \mathbb{N}$. We will say that a coloring $\bar{\mathcal{C}}$ of $\mathcal{C}$ is n-conservative up to size m if:

(♠ 2.) $ptp_m(\mathcal{C}, e, \Sigma) = ptp_m(M_n^{\bar{\Sigma}}(\bar{\mathcal{C}}), q_n(e), \Sigma)$ *for each* $e \in \mathcal{C}$, *where* Σ *and* $\bar{\Sigma}$ *are like in Definition 7.*

Being n-conservative up to size m means that the positive m-types (**with respect to the signature** Σ) of elements of $\bar{\mathcal{C}}$ are preserved by the quotient mapping q_n leading to the structure $M_n^{\bar{\Sigma}}(\bar{\mathcal{C}})$. So, for example, the coloring $\bar{\mathcal{C}}$ from Example 4 is n-conservative up to size m, if only $n > m$, but is not n-conservative up to size $m+1$ for any n.

Definition 9. A structure $\mathcal{C}$ is ptp-conservative if for each $m \in \mathbb{N}$ there exist $n \in \mathbb{N}$ and a coloring $\bar{\mathcal{C}}$ of $\mathcal{C}$, such that $\bar{\mathcal{C}}$ is n-conservative up to size m.

The following remark will be useful in Section 4

Remark 2. Consider a coloring $\bar{\mathcal{C}}$ of $\mathcal{C}$ and a number m. Suppose there is no such n that $\bar{\mathcal{C}}$ is n-conservative up to size m. This means that for each $n \in \mathbb{N}$ there is a query $\exists \bar{x} \Psi_n(\bar{x}, y)$, with at most m variables, and an element $e \in \mathcal{C}$ such that $M_n(\bar{\mathcal{C}}) \models \exists \bar{x} \Psi_n(\bar{x}, q_n(e))$ but $\mathcal{C} \not\models \exists \bar{x} \Psi_n(\bar{x}, e)$.

But since there are only finitely many queries of at most m variables, this implies that there is a query Ψ which is Ψ_n for infinitely many numbers n.

Notice also that if $n' < n$ and $M_n(\bar{\mathcal{C}}) \models \exists \bar{x} \Psi(\bar{x}, q_n(e))$ then $M_{n'}(\bar{\mathcal{C}}) \models \exists \bar{x} \Psi(\bar{x}, q_{n'}(e))$.

So, if there is no such n that $\bar{\mathcal{C}}$ is n-conservative up to size m then it must exist a single query $\Psi(\bar{x}, y)$, with $|x| < m$, such that for every n there is an element e of $\mathcal{C}$ such that $M_n(\bar{\mathcal{C}}) \models \exists \bar{x} \Psi(\bar{x}, q_n(e))$ but $\mathcal{C} \not\models \exists \bar{x} \Psi(\bar{x}, e)$.

2.6 Further examples and remarks

Example 5. It is very easy to see that the structure $\mathcal{C}$ from Examples 3 and 4 is ptp-conservative. Given m one just needs to define the coloring $\bar{\mathcal{C}}$ using $m+1$ colors, like in Example 4, and take $n = m+2$. Then $\bar{\mathcal{C}}$ will be n-conservative up to size m.

Example 6. Let $\mathcal{C}$ be any infinite set with a total (irreflexive) order E. Then it is easy to see that $\mathcal{C}$ is not ptp-conservative. Actually, it is impossible to find a coloring $\bar{\mathcal{C}}$ of $\mathcal{C}$ and a number n such that $\bar{\mathcal{C}}$ is n-conservative up to size 1: whatever the coloring, there would be an element e in $\mathcal{C}$ such that $M_n^{\bar{\Sigma}}(\bar{\mathcal{C}}) \models E(q_n(e), q_n(e))$.

Remark 3. It is very important to see the role of the element e in Definition 8. Condition (♠ 2.), which says that each element of $\mathcal{C}$ keeps its positive type after the quotient operation, is **strictly** stronger than:

(♠ 3.) *for each conjunctive query Ψ over Σ, with at most m variables, $\mathcal{C} \models \Psi$ if and only if $M_n^{\bar{\Sigma}}(\bar{\mathcal{C}}) \models \Psi$.*

which says that no new positive m-types appear in $M_n^{\bar{\Sigma}}\bar{\mathcal{C}}$.

To see that, consider a theory $\mathcal{T}$ consisting of the rules:
$E(x,y) \Rightarrow \exists z\ E(y,z)$
$E(x,y), E(y,z) \Rightarrow E(x,z)$
and a database instance $D = \{E(a,a), E(b,c)\}$.

Let $\mathcal{C} = Chase(D, \mathcal{T})$. Then $\mathcal{C}$ satisfies condition (♠ 3.) (since, due to the presence of the atom $E(a,a)$, all possible queries are true in $\mathcal{C}$), but is not ptp-conservative as it contains an infinite irreflexive total order (see Example 6).

Next remark explains what Definition 9 is good for:

Remark 4. Imagine that we have some theory $\mathcal{T}$ over a binary signature Σ, and there is a existential TGD Ψ in $\mathcal{T}$ of the form $\psi(\bar{x}, y) \Rightarrow \exists z\ R(y,z)$, where m is the number of variables in x. Let $\mathcal{C} = Chase(D, \mathcal{T})$. Clearly $\mathcal{C} \models \Psi$, so if for some $e \in \mathcal{C}$ it holds that $\exists \bar{x} \psi(\bar{x}, y) \in ptp_m(\mathcal{C}, e, \Sigma)$, then there exists $d \in \mathcal{C}$ such that $\mathcal{C} \models R(e,d)$.

Suppose we now color $\mathcal{C}$ and project it, using q_n, creating some finite structure $M_n(\bar{\mathcal{C}})$. We would like to be sure that $M_n(\mathcal{C})$ is still a model (or at least some sort of pre-model) of $\mathcal{T}$. So in particular we would like to be sure that $M_n(\mathcal{C}) \models \Psi$.

But if $\bar{\mathcal{C}}$ was n-conservative up to size m, then we can be sure that whenever we have an element $a = q_n(e)$ in $M_n(\bar{\mathcal{C}})$, such that $M_n(\bar{\mathcal{C}}) \models \exists \bar{x} \psi(\bar{x}, a)$ then also $\mathcal{C} \models \exists \bar{x} \psi(\bar{x}, e)$, which implies that $\mathcal{C} \models \exists z\ R(e,z)$, which implies that $M_n(\bar{\mathcal{C}}) \models \exists z\ R(a,z)$ (notice that what we use here is really Condition (♠ 2.) and that Condition (♠ 3.) would not be strong enough)

So, if $\bar{\mathcal{C}}$ was n-conservative up to size m, then $M_n(\bar{\mathcal{C}})$ is a model of Ψ, and if $\mathcal{C}$ is ptp-conservative then we can choose m greater than the maximal size of the body of a existential TGD rule in $\mathcal{T}$ and be sure that there exists a coloring, and number n, leading to $M_n(\bar{\mathcal{C}})$ in which all the existential TGDs of $\mathcal{T}$ are satisfied.

We now know how to turn a ptp-conservative Chase $\mathcal{C}$ of $\mathcal{T}$ into a finite structure $M_n(\bar{\mathcal{C}})$ satisfying all the existential TGDs in $\mathcal{T}$. But does it mean that $M_n(\bar{\mathcal{C}}) \models \mathcal{T}$ then? As the following example shows, not necessarily, even if $\mathcal{T}$ is BDD:

Example 7. Consider the following BDD theory $\mathcal{T}$:
$E(x,y) \Rightarrow \exists z\ E(y,z)$
$E(x,y), E(x',y) \Rightarrow R(x,x')$
and a database instance $D = \{E(a,b)\}$.

Let $\mathcal{C} = Chase(D, \mathcal{T})$. Clearly, $\mathcal{C}$ is an infinite E-chain, with an atom $R(e,e)$ true for each $e \in \mathcal{C}$. Whatever coloring we now use, the only R atoms in $M_n(\bar{\mathcal{C}})$ will be the ones of the form $R(e,e)$. And whatever the coloring, there must be a triple of elements a, b, c in $M_n(\bar{\mathcal{C}})$ such that $a \neq b$ and $M_n(\bar{\mathcal{C}}) \models E(a,c), E(b,c)$, which shows that $M_n(\bar{\mathcal{C}})$ is not a model of the plain datalog rule from $\mathcal{T}$.

Of course all the above definitions – of types, of $\equiv_n$, of M_n and of conservativity, make sense also when we consider any signatures, not just binary. But Remark 4 is not valid any more for such signatures, which means that it is very hard to make sure that M_n will actually resemble a model of $\mathcal{T}$. We will be back to this point in Section 5.4.

2.7 Very Treelike DAGs and the Main Lemma

Most of the notions we defined so far apply to structures over any signature. But what we are really interested in in this paper are binary signatures. They consist of some binary relations, some unary relations and constants. Structures over such signatures can be in a natural way seen as directed graphs with edges, and vertices, labeled with some finite number of labels (i.e. the names of the relations). Thanks to that we can use the language of graphs – for example our infinite structures are usually (directed) trees or DAGs.

We will concentrate on Very Treelike DAGs:

Definition 10. For an element $e \in \mathcal{C}$ we define $\mathcal{P}(e) = \{e\}$ if $e \in \mathcal{C}_{con}$ and $\mathcal{P}(e) = \{e\} \cup \{x \in \mathcal{C}_{non} : \mathcal{C} \models R(x,e)$ for some $R \in \Sigma\}$ if $e \in \mathcal{C}_{non}$

Definition 11. A structure $\mathcal{C}$ is called a Very Treelike DAG (VTDAG) if $\mathcal{C}_{non}$ is a DAG and:

- for each binary relation R and each $e \in \mathcal{C}_{non}$ there is at most one $d \in \mathcal{C}_{non}$ such that $R(d,e)$;
- for each $e \in \mathcal{C}_{non}$ if $d, d' \in \mathcal{P}(e)$ then $d \in \mathcal{P}(d')$ or $d' \in \mathcal{P}(d)$.

The first condition says that each non-constant e has at most one non-constant "direct predecessor" in each binary relation. The second says that the set of "direct predecessors" of e is a (directed) clique.

Each tree is trivially a VTDAG. In order to prove Theorem 1 it is enough to restrict the attention to trees only. VTDAGs which are not trees will not be considered before Section 5.

The main tool in the proof of Theorem 1 is:

LEMMA 2. *[The Main Lemma]*
Each VTDAG is ptp-conservative.

In Section 3 we use Lemma 2 to prove Theorem 1. Then, in Section 4, we present a proof of Lemma 2. Sections 3 and 4 are independent and can be read in any order.

3. FROM THE MAIN LEMMA TO THEOREM 2

3.1 Hiding the query inside the theory

Nothing complicated happens in this subsection. We are just making some simplifying (although without loss of generality) assumptions about the BDD theory under consideration. This will help us to keep the notations simpler in the rest of Section 3

For a binary BDD theory $\mathcal{T}_0$ and a conjunctive query $Q(\bar{x}, y)$ define a new theory $\mathcal{T}$ as $\mathcal{T}_0$ enriched with a new TGD:

($\spadesuit$ 4.) $Q(\bar{x}, y) \Rightarrow \exists z\ F(y, z)$

where F is a new predicate symbol. It is now easy to see that, for any database instance D such that F does not occur in D, a finite structure $\mathcal{M}$ such that $\mathcal{M} \models \mathcal{T}_0, D, \neg Q$ exists if and only if a finite structure $\mathcal{M}$ such that $\mathcal{M} \models \mathcal{T}, D, \neg F$ exists. This means that, in order to prove Theorem 1 it is enough to show:

THEOREM 2. *For a binary BDD theory $\mathcal{T}$, containing a rule of the form ($\spadesuit$ 4.), with predicate F not occurring anywhere else in $\mathcal{T}$, and for each database instance D, if F does not occur in $Chase(D, \mathcal{T})$ then there exists a finite structure $\mathcal{M}$ such that $\mathcal{M} \models D, \mathcal{T}$, without any atom of predicate F occurring in $\mathcal{M}$.*

From now on we assume that $\mathcal{T}$ is like in the assumptions of the above Theorem. We also assume, in order to keep the notations simple, that:

($\spadesuit$ 5.)

- *the head of each existential TGD in $\mathcal{T}$ is of the form $\exists z\ R(y, z)$, which means that the witness, whose existence is demanded by the TGD, is the second argument of the predicate in the head;*
- *if the predicate R occurs as the head of some existential TGD in $\mathcal{T}$ then it does not occur as the head of any datalog rule in $\mathcal{T}$. We call such predicates TGPs – tuple generating predicates.*

We leave it for the readers as an exercise to see that every $\mathcal{T}$ can be easily modified to satisfy ($\spadesuit$ 5.), for the cost of some additional predicates and datalog rules, and this modification neither changes the BDD status of the theory nor its FC status.

Hint: For each predicate R in the signature introduce two new predicates R' and R''. Add to theory datalog rules $R'(x, y) \leftarrow R(x, y)$ and $R''(x, y) \leftarrow R(y, x)$. Replace each head of an existential TGD which is of the form $\exists z\ R(y, z)$ by $\exists z\ R'(y, z)$ and each head of the form $\exists z\ R(z, y)$ by $\exists z\ R''(y, z)$.

3.2 The structure $\mathcal{S}(D, \mathcal{T})$

Let now D be a database instance without atoms of F and let Θ be the signature of D and $\mathcal{T}$. Define $\Sigma \supseteq \Theta$ as a new signature which contains, apart from the relations and constants from Θ, a name for each element in D. Why do we prefer the elements of D to be named? Because we want to be sure that their positive types in $Chase(D, \mathcal{T})$ differ, and, in consequence, that they remain distinct after a quotient operation (see Remark 1).

Now we are going to define the structure to which the techniques of Section 2 will be applied. Since we want to make use of Lemma 2, this structure must be a tree (or at least a VTDAG). And of course we cannot expect $Chase(D, \mathcal{T})$ to be a VTDAG.

Definition 12. By $\mathcal{S}(D, \mathcal{T})$ (or just $\mathcal{S}$, as the context is always clear) we mean the substructure of $Chase(D, \mathcal{T})$ consisting of all the elements of $Chase(D, \mathcal{T})$, all the atoms in D and all the atoms of the TGPs. We understand that $\mathcal{S}$ is a structure over the signature Σ.

We will call the atoms in $\mathcal{S}$ *skeleton atoms*, as we imagine $\mathcal{S}$ as a sort of a skeleton of $Chase(D, \mathcal{T})$. The atoms of $Chase(D, \mathcal{T})$ which are not in $\mathcal{S}$ will be called *flesh atoms*. So the flesh atoms are the ones created in the process of chase by the datalog rules.

It follows easily from ($\spadesuit$ 5.) that:

LEMMA 3. *(i) The graph $\mathcal{S}_{non}$ is acyclic;*

(ii) the in-degree of any element of $\mathcal{S}_{non}$ is 1;

(iii) $\mathcal{S}_{non}$ is a forest;

(iv) the degree of the elements of $\mathcal{S}_{non}$ is bounded by $|\Sigma|+1$;

Remember that all the elements of D are constants from Σ, so they are not in $\mathcal{C}_{non}$.

PROOF. For the proof of (i) and (ii) notice that the only way a TGP atom $R(a, b)$ can be created is to be created together with a new element b. Acyclicity follows from the fact that b is always a "younger" element of $Chase(D, \mathcal{T})$ than a. The claim (iii) follows from (i) and (ii). Finally, (iv) follows from the fact, that the chase we consider is a non-oblivious one, so for any fixed $a \in \mathcal{S}$ and for any TGP R from Σ at most one $b \in \mathcal{S}$ can exist such that $\mathcal{S} \models R(a, b)$. □

Let us now think of $\mathcal{S}$ as of a new database instance:

LEMMA 4. $Chase(D, \mathcal{T}) \models Chase(\mathcal{S}, \mathcal{T})$. *In particular,* $\mathrm{Dom}(Chase(\mathcal{S}, \mathcal{T})) = \mathrm{Dom}(Chase(D, \mathcal{T})) = \mathrm{Dom}(\mathcal{S})$

For the (easy) proof of Lemma 5 see the full version. It is very easy to see that also $Chase(\mathcal{S}, \mathcal{T}) \models Chase(D, \mathcal{T})$, so we get that $Chase(D, \mathcal{T}) = Chase(\mathcal{S}, \mathcal{T})$.

The idea behind the Lemma is that while $\mathcal{S}$ is a simple structure – simple enough to be ptp-conservative – still not only it contains all elements of $Chase(D, \mathcal{T})$ but also the complete information about the relations between elements of $Chase(D, \mathcal{T})$ that need a witness and the needed witnesses. Thanks to that $Chase(D, \mathcal{T})$ can be rebuilt, starting from the skeleton $\mathcal{S}$, in a process of a (non-oblivious) chase that only triggers datalog rules, but never the existential TGDs. Notice that this would no longer be true if a single atom $R(a, a')$ was removed from $\mathcal{S}$ (even if the elements a,a' were kept, as arguments of some other atoms). This is because at some point a TGD with the head $\exists x\ R(a, x)$ would be triggered, and a **new** element a'' would be created.

3.3 Proof of Theorem 2

Recall that for a BDD theory $\mathcal{T}$ and query Ψ by Ψ' we mean the positive first order rewriting of Ψ, which means that Ψ' is such a query (a union of conjunctive queries), that for each database instance D it holds that $Chase(D, \mathcal{T}) \models \Psi \Leftrightarrow D \models \Psi'$.

Let $\kappa = \max\{|Var(\Psi')| : \Psi \Rightarrow \psi \text{ is a rule in } \mathcal{T}\}$. In other words, κ is the maximal number of variables in a query being a positive first order rewriting of a body of some rule of the theory $\mathcal{T}$. By Lemma 2 there exists a coloring $\bar{\mathcal{S}}$ of $\mathcal{S}$ and $\eta \in \mathbb{N}$ such that $\bar{\mathcal{S}}$ is η-ptp conservative up to the size κ,

which means that the elements of $M_\eta^{\bar{\Sigma}}(\bar{\mathcal{S}})$ have the same positive κ-types over Σ as their counter-images in $\mathcal{S}$.

Now there are five structures one should imagine:

(i) $\mathcal{S}$

(ii) $\text{Chase}(D, \mathcal{T}) = Chase(\mathcal{S}, \mathcal{T})$

(iii) $M_\eta^{\bar{\Sigma}}(\bar{\mathcal{S}})$

(iv) $\text{Chase}(M_\eta^{\bar{\Sigma}}(\bar{\mathcal{S}}), \mathcal{T})$

(v) $q_\eta(\text{Chase}(D, \mathcal{T}))$

The first two of them were already introduced in this Section. The third is the result of the quotient operation applied to $\bar{\mathcal{S}}$ – something we discussed in Section 2. Since $\bar{\mathcal{S}}$ is η-ptp conservative up to the size κ, we know that $M_\eta^{\bar{\Sigma}}(\bar{\mathcal{S}})$ is a model for all existential TGDs in $\mathcal{T}$ (see Remark 4). But, as we saw in Example 7, we cannot be sure that $M_\eta^{\bar{\Sigma}}(\bar{\mathcal{S}}) \models \mathcal{T}$, as some datalog rules from $\mathcal{T}$ may be false in $M_\eta^{\bar{\Sigma}}(\bar{\mathcal{S}})$. So to get a model of $\mathcal{T}$ we apply chase to $M_\eta^{\bar{\Sigma}}(\bar{\mathcal{S}})$, which leads to our fourth structure, $\text{Chase}(M_\eta^{\bar{\Sigma}}(\bar{\mathcal{S}}), \mathcal{T})$. So far we know nothing about this structure, in particular we do not even know whether $\text{Chase}(M_\eta^{\bar{\Sigma}}(\bar{\mathcal{S}}), \mathcal{T})$ is finite.

The fifth structure, $q_\eta(\text{Chase}(D, \mathcal{T}))$ is only needed in example 8 which we hope explains some issues concerning $\text{Chase}(M_\eta^{\bar{\Sigma}}(\bar{\mathcal{S}}), \mathcal{T})$. **If** you feel you not need more explanations **go directly to Lemma 5**. $q_\eta(\text{Chase}(D, \mathcal{T}))$ is defined as:

- $\text{Dom}(q_\eta(\text{Chase}(D, \mathcal{T}))) = \text{Dom}(M_\eta^{\bar{\Sigma}}(\bar{\mathcal{S}}))$;
- relations are defined in $q_\eta(\text{Chase}(D, \mathcal{T}))$ as the minimal relations such that q_η, understood as a mapping from $\text{Chase}(D, \mathcal{T})$ to $q_\eta(\text{Chase}(D, \mathcal{T}))$, is a homomorphism.

So while the relations $M_\eta^{\bar{\Sigma}}(\bar{\mathcal{S}})$ are defined as projections of the skeleton relations, the relations $q_\eta(\text{Chase}(D, \mathcal{T}))$ are projections of both, the skeleton and the flesh atoms.

One can see that $\text{Chase}(M_\eta^{\bar{\Sigma}}(\bar{\mathcal{S}}), \mathcal{T}) \models q_\eta(\text{Chase}(D, \mathcal{T}))$. Indeed, any atom in $q_\eta(\text{Chase}(D, \mathcal{T}))$ which is not in $M_\eta^{\bar{\Sigma}}(\bar{\mathcal{S}})$ is a projection of some flesh atom in $\text{Chase}(D, \mathcal{T})$. This last atom must have been proved by some derivation in $\text{Chase}(D, \mathcal{T})$. But a projection of a valid derivation from $\text{Chase}(D, \mathcal{T})$ is a valid derivation in $\text{Chase}(M_\eta^{\bar{\Sigma}}(\bar{\mathcal{S}}), \mathcal{T})$.

At this point it would be reasonable to conjecture that maybe $q_\eta(\text{Chase}(D, \mathcal{T})) = \text{Chase}(M_\eta^{\bar{\Sigma}}(\bar{\mathcal{S}}), \mathcal{T})$. But this is not always the case, as the following example shows:

Example 8. Let $\mathcal{T}$ and D be like in Example 7. Let $\mathcal{C} = Chase(\mathcal{T}, D)$. Now $\mathcal{S}$ is the structure $\mathcal{C}$ from Example 4. Let m be the number of colors, $n > m$ and let $\bar{\mathcal{S}}$ be a coloring of $\mathcal{S}$, like in Example 4. Now, the only R atoms in $q_n(M_n^{\bar{\Sigma}}(\bar{\mathcal{S}}))$ are atoms of the form $R(a, a)$ for some $a \in M_n^{\bar{\Sigma}}$. But it is easy to see that $\text{Chase}(M_n^{\bar{\Sigma}}, T) \models R(b_{n-1}, b_{n+m})$ (where b_{n-1}, b_{n+m} are again like in Example 4).

In the last example an atom was derived in $\text{Chase}(M_\eta^{\bar{\Sigma}}(\bar{\mathcal{S}}), \mathcal{T})$, which was not a projection of any flesh atom. The meaning of the next lemma is that while, in the process on chase on $M_\eta^{\bar{\Sigma}}(\bar{\mathcal{S}})$, some datalog derivations can arise not being projections of datalog derivations in chase on $\mathcal{S}$, still (like in Lemma 4) no existential TGDs will be used, and no new elements will be created. To be more precise:

LEMMA 5. $\text{Dom}(Chase(M_\eta(\bar{\mathcal{S}}), \mathcal{T})) = \text{Dom}(M_\eta(\bar{\mathcal{S}}))$

For the proof of Lemma 5 see the full version. This proof is not very long but we believe it is quite tricky. It is here where things really happen: Lemma 2 meets the assumption that $\mathcal{T}$ is BDD.

As we show in Section 5.4, there is no hope to have anything similar to Lemma 5 in the general (non-binary) case.

We are ready to present the **proof of Theorem 2:**

In order to prove Theorem 2 (and, in consequence, Theorem 1) we need to show a finite model of D and $\mathcal{T}$ without any atom of the predicate symbol F. The structure $Chase(M_\eta(\bar{\mathcal{S}}), \mathcal{T})$ is clearly a model of $D, \mathcal{T}$. It follows from the Lemma 5 that its domain is exactly the domain of $M_\eta(\bar{\mathcal{S}})$, so it is finite.

Since no atom of the relation F occurs in $\mathcal{S}$ there is also no such atom in $M_\eta(\bar{\mathcal{S}})$. So the only way any such atom could appear in $Chase(M_\eta(\bar{\mathcal{S}}), \mathcal{T})$ would be to derive it in the process of chase. But the only rule that derives F is a existential TGD which demands a new element, and no such rule could have been used, due to Lemma 5. □

4. PROOF OF THE MAIN LEMMA

Fix a VTDAG $\mathcal{C}$ and a natural number m. Let Σ be the signature of $\mathcal{C}$. To prove Lemma 2 we need to find $n \in \mathbb{N}$ and a coloring $\bar{\mathcal{C}}$ of $\mathcal{C}$ such that $\bar{\mathcal{C}}$ is n-conservative up to the size m.

First let us define the coloring:

Definition 13. For $e \in \mathcal{C}$ let $\mathcal{P}(e)$ be like in Definition 10.

- For $e \in \mathcal{C}_{non}$ let $\mathcal{P}_0(e) = \mathcal{P}(e)$.
- For $e \in \mathcal{C}_{non}$ let $\mathcal{P}_k(e) = \bigcup_{a \in \mathcal{P}_{k-1}(e)} P(a)$

Definition 14. A coloring $\bar{\mathcal{C}}$ of $\mathcal{C}$ will be called natural if it satisfies the following conditions:

- if $e, e' \in \mathcal{C}$ are such that $e' \in \mathcal{P}_m(e)$ and if $\bar{\mathcal{C}} \models K_h^l(e), K_{h'}^{l'}(e')$ then $h \neq h'$;
- if $e, e' \in \mathcal{C}$ are such that $\bar{\mathcal{C}} \models K_h^l(e), K_{h'}^l(e')$ then $\mathcal{C} \restriction (\mathcal{P}(e) \cup \mathcal{C}_{con})$ and $\mathcal{C} \restriction (\mathcal{P}(e') \cup \mathcal{C}_{con})$ are isomorphic.

It is easy to see that for each VTDAG $\mathcal{C}$ there exists a natural coloring $\bar{\mathcal{C}}$. From now on by $\bar{\mathcal{C}}$ we will mean a fixed natural coloring of $\mathcal{C}$.

By Remark 2 the proof of Lemma 2 will be finished when we show:

LEMMA 6. *For each query $\Phi(\bar{x}, y)$ over Σ, with $|\bar{x}| < m$, there exists $n \in \mathbb{N}$ such that for each element $e \in \mathcal{C}$:*
$M_n(\bar{\mathcal{C}}) \models \exists \bar{x} \Phi(\bar{x}, q_n(e))$ if and only if $\mathcal{C} \models \exists \bar{x} \Phi(\bar{x}, e)$

Proof of Lemma 6 begins here.
It will take till the end of Section 4 to finish.

First of all notice that if the Lemma 6 was false, then there would exists a **counterexample** – a conjunctive query $\Phi(\bar{x}, y)$ such that:

(♣) for each $n \in \mathbb{N}$ there exists an element e_n^Φ of $\mathcal{C}$ and a valuation $\sigma_n^\Phi : Var(\Phi) \to M_n(\bar{\mathcal{C}})$, with $\sigma_n^\Phi(y) = q_n(e_n^\Phi)$, such that $M_n(\bar{\mathcal{C}}) \models \sigma_n^\Phi(\Phi)$ and $\mathcal{C} \not\models \exists \bar{x} \Phi(\bar{x}, e_n^\Phi)$.

Each time we will say that query Φ is a counterexample we will think that is satisfies condition (♣).

By a colors statement we will mean a query of the form:

$$\bigwedge_{z \in Var(\Phi)} K_{h_z}^{l_z}$$

where $K_{h_z}^{l_z}$ is any of the possible colors from $\mathcal{K}$. By a color closure of Φ we will mean any query of the form $\Phi \wedge \Upsilon$, where Υ is a colors statement. Of course there are finitely many colors statements, and so there are finitely many possible color closures of Φ. A query which is a color closure of some other query will be called color closed.

LEMMA 7. *(i) Let Φ be a counterexample. Then for each n there is a query Φ_c, being a color closure of Φ, such that $M_n(\bar{\mathcal{C}}) \models \sigma_n^\Phi(\Phi_c)$ and $\bar{\mathcal{C}} \not\models \exists \bar{x} \Phi_c(\bar{x}, e_n^\Phi)$.*

(ii) For each counterexample Φ there exists a color closure $\bar{\Phi}$ of Φ, which also is a counterexample.

(iii) If there exists a query Φ being a color closed counterexample, then there also exists another color closed counterexample Ψ such that for each constant c from Σ, for each variable $z \in Var(\Psi)$ and for each $n \in \mathbb{N}$ there is $\sigma_n^\Psi(z) \neq c$, where σ_n^Ψ is as (♣). We will say that counterexample Ψ avoids constants.

PROOF. (i) The elements $\sigma_n^\Phi(z)$, where $z \in Var(\Phi)$ have some colors. Adding to Φ a statement asserting that they have the colors they really have will not make the new query $\sigma_n(\Phi_C)$ less true in $M_n(\bar{\mathcal{C}})$ than $\sigma_n^\Phi(\Phi)$ was.

On the other hand, $\exists \bar{x} \Phi(\bar{x}, e_n)$ was false in $\mathcal{C}$ already before the color statement was added and adding more constraints never makes a query more true.

(ii) Use (i) and an argument like in Remark 2.

(iii) Suppose Φ is a color closed counterexample and $\sigma_n^\Phi(z) = c$ for some constant $c \in \Sigma$, some variable $z \in Var(\Psi)$ and some $n \in \mathbb{N}$. By the definition of natural coloring, the color of c is unique in $\mathcal{C}_{con}$ and thus the equality $\sigma_n^\Phi(z) = c$ must hold for each n, and thus Ψ being the result of replacing each occurrence of z in Φ by c is also a counterexample. □

We are now going to view queries as graphs. What we mean here is a sort of Gaifman graphs, where vertices are the variables in the query and the edges are the atoms of the query. As we only have binary and unary atoms, we can in a natural way see each query as a directed (labeled) graph. Concerning the constants in the query, they are not understood to be vertices in the graph, and it is good to think that an atom of the form $R(a, x)$ in a query, where a is a constant and x is a variable, is just a unary predicate, telling us something about x alone. Notice that atoms of the form $R(a, b)$ in a query, where both a and b are constants, are irrelevant from the point of view of Lemma 6, as the part of $\mathcal{C}$ consisting of the constants remains unchanged after our projections.

Now our plan of the proof of Lemma 6 is as follows. We want to show that no query is an avoiding constants color closed counterexample. So first we will notice (Lemma 8 and Lemma 9) that neither a query being an undirected tree, nor a query containing a directed cycle can ever be a counterexample. At this point we will know that if there is any avoiding constants color closed counterexample Φ then Φ must contain an undirected cycle (but not a directed one). But then, in Lemma 10 we show that if such a Φ existed, then also another counterexample would exist, being a tree or containing a directed cycle. That would however contradict Lemma 8 and Lemma 9.

Proofs of Lemma 8 and Lemma 9 are easy. Proof of Lemma 10, where we deal with queries containing an undirected cycle, is much more complicated. A technique of normalization of queries is used there, which we find to be the deepest idea of this paper (we also employ this technique, in different context, in [GM12], where it is called *second little trick*). Why are the undirected cycles in the query so much harder to deal with than directed ones? The answer is in:

Example 9. Let a theory $\mathcal{T}$ consist of the rules:

$F(x, y) \Rightarrow \exists z\ F(y, z)$ $\quad$ $F(x, y) \Rightarrow \exists z\ G(y, z)$

$G(x, y) \Rightarrow \exists z\ F(y, z)$ $\quad$ $G(x, y) \Rightarrow \exists z\ G(y, z)$

Let $D = \{F(a, b)\}$ and let $\mathcal{C}$ be $Chase(D, \mathcal{T})$, which means that $\mathcal{C}$ is an infinite tree, where each element has exactly two successors. Or, in other words, $\mathcal{C}$ consists, except from a and b, of all the elements $w(b)$, where $w \in \{f, g\}^*$. Let $\bar{\mathcal{C}}$ be a natural coloring of $\mathcal{C}$.

Let e_1, e_2 be two elements of $\mathcal{C}$ of the form $e_1 = vfw(b)$, $e_2 = vgw(b)$, where $v, w \in \{f, g\}^*$ and where $|v| = n - 1$.

Then $a_1 = q_n(e_1) \neq q_n(e_2) = a_2$ are two distinct elements of $M_n(\bar{\mathcal{C}})$ – the length of v is not big enough to hide the slight difference in the positive types of e_1 and e_2. Of course also $a_3 = q_n(f(e_1)) \neq q_n(g(e_1)) = a_4$ are two distinct elements (each of them distinct than a_1 and a_2). But $a_3 = q_n(f(e_2))$ and $a_4 = q_n(g(e_2))$. This means that the atoms $F(a_1, a_3), F(a_2, a_3), G(a_2, a_4), G(a_1, a_4)$ are all true in $M_n(\bar{\mathcal{C}})$, and so there is an undirected cycle in $M_n(\bar{\mathcal{C}})$ consisting of 4 distinct elements.

As we saw in Example 4, by using coloring we can easily make sure that there are no small directed cycles in $M_n(\bar{\mathcal{C}})$. But we cannot rule out small undirected new (not present in $\bar{\mathcal{C}}$) cycles in $M_n(\bar{\mathcal{C}})$. So we need to prove that, while the new cycles exist, no small query can actually notice them.

LEMMA 8. *Let $n \geq m$. Then for each element $e \in \mathcal{C}$ and each query $\Psi(\bar{x}, y)$, which is an undirected tree:*
$\Psi \in ptp_m(\bar{\mathcal{C}}, e, \bar{\Sigma})$ if and only if $\Psi \in ptp_m(M_n(\bar{\mathcal{C}}), q_n(e), \bar{\Sigma})$.

It of course follows from the lemma that no query being an undirected tree can be a counterexample.

LEMMA 9. *Let $n \geq m$. Suppose Φ is a query containing a directed cycle, by which we mean a sub-query of the form:*
$R_1(x_1, x_2), R_2(x_2, x_3), \ldots R_{k-1}(x_{k-1}, x_k), R_k(x_k, x_1)$
where $k < m$ and R_i are relation symbols from Σ. Then $M_n(\bar{\mathcal{C}}) \not\models \Phi$.

Clearly, as being a counterexample means, among other conditions, being true in $M_n(\bar{\mathcal{C}})$, the lemma implies that Φ, containing a directed cycle, never is a counterexample.

For the proofs of Lemma 8 and 9 see the full version. Now, Lemma 6 follows from Lemma 7, Lemma 8, Lemma 9 and from the following:

LEMMA 10. *If there exists a color closed counterexample Φ which avoids constants and which contains an undirected cycle then there exists also a counterexample being a tree or a counterexample containing a directed cycle.*

4.1 Proof of Lemma 10

Consider a query $\Psi(\bar{x}, y)$ which contains an undirected cycle, which is not a directed cycle. Then Ψ must be of the form:

($\heartsuit$) $R_1(z', z) \wedge R_2(z'', z) \wedge \psi(\bar{x}, y)$

for some relations $R_1, R_2 \in \Sigma$ and some $z, z', z'' \in Var(\Psi)$.

LEMMA 11 (NORMALIZATION OF QUERIES). *If any color closed, avoiding constants, query $\Psi(\bar{x}, y)$ of the form ($\heartsuit$) is a counterexample, then there is a binary relation $P \in \Sigma$ such that one of the following queries is also a color closed, avoiding constants, counterexample:*

- $\psi(\bar{x}, y) \wedge R_1(z', z) \wedge z' = z''$
- $\psi(\bar{x}, y) \wedge R_1(z', z) \wedge P(z'', z')$
- $\psi(\bar{x}, y) \wedge R_2(z'', z) \wedge P(z', z'')$

To see how Lemma 11 implies Lemma 10, **while** Ψ is a counterexample of the form ($\heartsuit$) **do** replace it with another counterexample, the one whose existence is assured by Lemma 11. The only way to leave the while-loop is to produce a counterexample which is a tree or contains a directed cycle. So it is enough to prove that the while-loop indeed terminates.

If the first possibility from the Lemma is used as the replacement, then the new query has less variables than the old one (since adding an equivalence of variables is the same as unifying the variables). But the last two possibilities do not decrease the number of variables. So aren't they going to be applied forever? Consider the following measure of the size of a query:

$$\text{Measure}(\Phi) = \Sigma_{x \in Var(\Psi)}\ occ(x) smaller(x)$$

where $occ(x)$ is the number of the occurrences of variable x in Ψ and $smaller(x)$ is the number of variables from which x is reachable by a directed path in the graph of the query. It is easy to see that Measure(Ψ) is a natural number which decreases each time Lemma 11 is applied.

Before we prove Lemma 11 notice that the first condition in Definition 11 implies:

LEMMA 12. *Suppose a, b, c, d are non-constant elements of $\bar{\mathcal{C}}$, such that $\bar{\mathcal{C}} \models R(a, b), R(c, d)$ for some relation $R \in \Sigma$. Then $b \equiv_n d$ implies $a \equiv_{n-1} c$.*

PROOF. Suppose there was a query $\psi(\bar{x}, y)$, with $|\bar{x}| < n - 1$, such that $\bar{\mathcal{C}} \models \exists \bar{x} \psi(\bar{x}, a)$ but $\bar{\mathcal{C}} \not\models \exists \bar{x} \psi(\bar{x}, c)$. Then $\bar{\mathcal{C}} \models \exists \bar{x} x' \psi(\bar{x}, x') \wedge R(x', b)$ but $\bar{\mathcal{C}} \not\models \psi(\bar{x}, x') \wedge R(x', d)$. But this would mean that $b \not\equiv_n d$. Notice that the assumption that $\mathcal{C}$ is a VTDAG was used here. □

Proof of Lemma 11 Suppose Ψ is a color closed counterexample of the form ($\heartsuit$). Consider the color of z (call it color(z)). More precisely, color(z) is the color that is enforced by Ψ on any valuation of z that satisfies Ψ . What we are interested in is not really the full information about color(z), but its lightness – the information about the isomorphic type of $\mathcal{P}(e)$ for any $e \in \bar{\mathcal{C}}$ having the color that Ψ enforces on z.

Now please be ready for the most complicated argument of this paper. Let e be like in the previous paragraph. The set $\mathcal{P}(e)$ contains some elements e' and e'' such that $R_1(e', e) \wedge R_2(e'', e)$ are true in $\bar{\mathcal{C}}$. It follows from Definition 11 that in such case there must be an atom $Q(e', e'')$ true in $\bar{\mathcal{C}}$, where $Q(e', e'')$ is either $P(e', e'')$ or $P(e'', e')$ for some relation $P \in \Sigma$, or $e' = e''$ (this happens when $R_1 = R_2$). Notice that the atom Q only depends on the color of z not on the choice of e. Suppose Q is $P(e', e'')$, the other two possibilities are analogous. Now, we claim that $\Phi = \psi(\bar{x}, y) \wedge R_2(z'', z) \wedge P(z', z'')$ is also a counterexample, with $\sigma_n^\Phi = q_n \circ \sigma_{n+1}^\Psi$, $e_n^\Phi = e_{n+1}^\Psi$ and σ_n.

Notice that we use the notation q_n here in the sense defined in (♠ 1.): σ_{n+1}^Ψ, for an argument being a variable of Ψ (or Φ – they have the same set of variables) returns an element of $M_{n+1}(\bar{\mathcal{C}})$ and q_n for an argument from $M_{n+1}(\bar{\mathcal{C}})$ returns an element of $M_n(\bar{\mathcal{C}})$.

We need to show that the conditions from (♣) are now satisfied. It is easy to see that $\sigma_n^\Phi(y) = q_n \circ \sigma_{n+1}^\Psi(y) = q_n(e_{n+1}^\Psi) = q_n(e_n^\Phi)$.

What remains to be shown is that for each $n \in \mathbb{N}$:

(*) $M_n(\bar{\mathcal{C}}) \models \sigma_n^\Phi(\Phi)$ and (**) $\bar{\mathcal{C}} \not\models \exists \bar{x} \Phi(\bar{x}, e_n^\Phi)$.

Let us begin with (**), which is easier. Suppose $\bar{\mathcal{C}} \models \Phi(\bar{x}, e_n^\Phi)$. So there exists a valuation $\gamma : Var(\Phi) \to \bar{\mathcal{C}}$, with $\gamma(y) = e_n^\Phi$, such that $\bar{\mathcal{C}} \models \gamma(\Phi)$. Notice that $\gamma(y) = e_{n+1}^\Psi$. We claim that $\bar{\mathcal{C}} \models \gamma(\Psi)$ and this will be in contradiction with what we assumed about Ψ and e_{n+1}^Ψ.

For the proof of the last claim it is enough to show that $\bar{\mathcal{C}} \models R_1(\gamma(z'), \gamma(z))$, as this is the only atom of Ψ missing in Φ. But this follows from what we know about the isomorphic type of $\gamma(z)$, from the fact that $\bar{\mathcal{C}} \models P(\gamma(z'), \gamma(z'')) \wedge R_2(\gamma(z''), \gamma(z))$ and from the assumption that the in-degree of each of the relations in $\bar{\mathcal{C}}$ is at most 1 (first condition in Definition 11).

Now we are going to prove (*). We know that $M_{n+1}(\bar{\mathcal{C}}) \models \sigma_{n+1}^\Psi(\Psi)$, so also $M_n(\bar{\mathcal{C}}) \models q_n \circ \sigma_{n+1}^\Psi(\Psi)$. What remains to be proved is that

(*) $M_n(\bar{\mathcal{C}}) \models P(q_n \sigma_{n+1}^\Psi(z'), q_n \sigma_{n+1}^\Psi(z''))$.

We know that $M_n(\bar{\mathcal{C}}) \models R_1(\sigma_{n+1}^\Psi(z'), \sigma_{n+1}^\Psi(z))$ and that $M_n(\bar{\mathcal{C}}) \models R_2(\sigma_{n+1}^\Psi(z''), \sigma_{n+1}^\Psi(z))$. This means that there are elements a', a, b'', b of $\bar{\mathcal{C}}$ such that: $q_{n+1}(a) = q_{n+1}(b) = \sigma_{n+1}^\Psi(z)$, $q_{n+1}(a') = \sigma_{n+1}^\Psi(z')$, $q_{n+1}(a'') = \sigma_{n+1}^\Psi(z'')$ and $\bar{\mathcal{C}} \models R_1(a', a) \wedge R_2(b'', b)$. The color of a and of b is the color of z, so the isomorphic type of $\mathcal{P}(a)$ is the same as the isomorphic type of $\mathcal{P}(b)$, and the same as the isomorphic type of $\mathcal{P}(e)$, where e is as in the beginning of Lemma 11. This means that there is an element $a'' \in \bar{\mathcal{C}}$ such that $\bar{\mathcal{C}} \models R_2(a'', a)$ and $\bar{\mathcal{C}} \models P(a'', a')$. There is no reason to think that $a'' \equiv_{n+1} b''$. But from Lemma 12 we get that $a'' \equiv_n b''$. So a'' and a are two elements of $\bar{\mathcal{C}}$ such that $\bar{\mathcal{C}} \models R_2(a'', a)$, that $q_n(a) = q_n \sigma_{n+1}^\Psi(z)$ and that $q_n(a'') = q_n \sigma_{n+1}^\Psi(z'')$. □

This ends the proofs of Lemmas 10,11 , 6 and 2.

5. DISCUSSION

5.1 Beyond the binary case (slightly)

As a careful reader might already have noticed, our proof of Theorem 2 can also be read as a proof of:

THEOREM 3. *Let $\mathcal{T}$ be a set of existential TGDs and plain datalog rules, with each of its existential TGDs of the form: $\Psi(\bar{x}, y) \Rightarrow \exists \bar{z}\ \Phi(y, \bar{z})$. Then, if $\mathcal{T}$ is BDD, then it is also FC.*

It is because in the proof of Theorem 2 we only used the binarity assumption for heads of existential TGDs.

Notice that we can rewrite existential TGDs from Theorem 3 into conjunction of existential TGDs with binary

heads and some arbitrary datalog rules. Hence the whole proof of Theorem 2 survives.

Hint: For each TGD $\Psi(\bar{x}, y) \Rightarrow \exists \bar{z}\ \Phi(y, \bar{z})$ we add new relational symbols $R^1_\Phi(y, z_1) \ldots R^n_\Phi(y, z_n)$ where $n = |\bar{z}|$. We add to the theory rules $\Psi(\bar{x}, y) \Rightarrow \exists \bar{z}\ R^i_\Phi(y, z_i)$ and datalog rules $R^1_\Phi(y, z_1) \wedge \ldots \wedge R^n_\Phi(y, z_n) \rightarrow \Phi(y, \bar{z})$.

5.2 The ternary case

Usually, once we know that some property holds for binary signatures, it is easy to prove, by some sort of reduction, that it remains true in the general case. This rule does not seem to be valid for the BDD/FC conjecture. What we can however easily show is:

THEOREM 4. *If the BDD/FC conjecture for ternary signatures is true then it is true in the general case.*

Instead of presenting a detailed proof of the theorem, which would be boring, let us show an example of how the reduction works. Suppose we have a theory $\mathcal{T}$ with a rule like:

(*) $P(x, y, z, x) \Rightarrow \exists t\ R(x, y, z, t)$

then rewrite it into the following three rules:

$P(x, y, z, x) \Rightarrow \exists w_1\ R_1(x, y, w_1)$

$P(x, y, z, x) \wedge R_1(x, y, w_1) \Rightarrow \exists w_2\ R_2(w_1, z, w_2)$

$P(x, y, z, x) \wedge R_1(x, y, r) \wedge R_2(r, z, s) \Rightarrow \exists t\ R'(s, t)$

The idea is here that using ternary predicates we can give names to lists of variables, in the good old Prolog way. We appear to still have non-ternary predicates in the bodies of the rules. But just don't think of them as of predicates any more! The P in the body of (*) is just a view over the real predicates P_1, P_2 and P' now, which relate to P in the same was as R_1, R_2 and R' relate to R.

In this way we constructed a new, ternary theory, call it $\mathcal{T}'$. What we would now need to show (if it was a real detailed proof) would be that (i) if $\mathcal{T}$ is BDD then $\mathcal{T}'$ also is, and (ii) if $\mathcal{T}'$ is FC then $\mathcal{T}$ also is. To see how (ii) works take a database instance D and query Q. Rewrite D and Q into D' and Q' in the new ternary language (possibly adding some new elements to denote lists of elements of D). Of course if $Chase(\mathcal{T}, D) \not\models Q$ then also $Chase(D', \mathcal{T}') \not\models Q'$. So, if $\mathcal{T}'$ is FC, there exists a finite $\mathcal{M}'$ being a model of $\mathcal{T}'$ and D' such that $\mathcal{M}' \not\models Q'$. Now, to finish the proof of (i), define the relations of $\mathcal{M}$ as views over respective relations in $\mathcal{M}'$.

Showing (i) is not really hard either.

5.3 Multi-head TGDs

The TGDs we consider in this paper are assumed to be single-head. Of course if the arity is not restricted, then the validity of the BDD/FC conjecture does not depend on this assumption, as every multi-head TGD Ψ can be replaced by a single-head TGD having, as its head, the join of all the atoms in the head of Ψ, and by some datalog rules splitting this join back into smaller atoms. But such a simple transformation is not possible for binary signatures. It is actually easy to see that the BDD/FC conjecture for multi-head TGDs over binary signatures is already equivalent to the full conjecture, as any ternary Datalog$^\exists$ program can be encoded in this format. For example the rule:

$P_1(x, y, z) \wedge P_2(x, y, z') \Rightarrow \exists w\ P(x, z, w)$

can be encoded as (read $A^i(t, x)$ as "x is the i'th argument in the atom t"):

$A^1_{P_1}(t_1, x) \wedge A^2_{P_1}(t_1, y) \wedge A^3_{P_1}(t_1, z) \wedge A^1_{P_2}(t_2, x) \wedge A^2_{P_2}(t_2, y) \wedge A^3_{P_2}(t_2, z') \Rightarrow \exists t\ A^1_P(t, x) \wedge A^2_P(t, y)$

and $A^1_P(t, x) \wedge A^2_P(t, y) \Rightarrow \exists w\ A^3_P(t, w)$.

5.4 Why M_n are too poor to be models (in the non-binary case)

The main idea of our proof of Theorem 1 was first to find, for a BDD theory $\mathcal{T}$ and a database instance D the skeleton $\mathcal{S}$ which is a substructure of $Chase(D, \mathcal{T})$ on one hand being simple enough to be ptp-conservative, but on the other hand not only containing all the elements of $Chase(D, \mathcal{T})$, but also sufficient information about the relations between elements which require a witness and the witnesses. Then the idea was to prove (Lemma 5) that the finite model M_n constructed from this simple structure by a quotient operation can be saturated, using the datalog rules from $\mathcal{T}$, to a model of $\mathcal{T}$, without adding any new elements being necessary.

The first reason this line of reasoning cannot be used in the general (non-binary) case is that the distinction between existential TGDs and plain datalog rules makes then no sense any more: each datalog rule can be turned into a TGD by adding a new (existentially quantified) dummy variable to the atom on the right hand side of the query. But what we view as an even more serious obstacle is that, as the following example shows, it is hard to imagine how anything analogous to Lemma 5 could be true in the general case:

Let the rules of $\mathcal{T}$ be: $R(x, x', y, z) \Rightarrow E(y, z)$

and $E(x, y), E(t, y) \Rightarrow \exists z\ R(x, t, y, z)$

and let a database instance D be $\{E(a, b)\}$.

Clearly, $\mathcal{T}$ is BDD. And $\mathcal{C} = Chase(D, \mathcal{T})$ is a very simple structure: an infinite E-chain, with additional atom $R(x, x, y, z)$ for each three consecutive elements x, y, z of this chain. But whenever any two elements of $\mathcal{C}$ are identified by a quotient operation, a new tuple satisfying the body of the (only) TGD form $\mathcal{T}$ emerges (something we have already seen in Example 7), and a new witness z is required for this tuple. Since the new witness is a function of the whole tuple, not just of (the element substituted for) y, the (already existing) element t of $M_n(\bar{\mathcal{C}})$ such that $E(y, t)$ cannot be used now, and a new one must be created. If there was just one element this would be something we could live with – our main goal is just to keep the structure finite. But notice that once the new witness z, with $E(y, z)$ is created, it enforces a new infinite E-chain to be built.

5.5 Beyond BDD. The dead end of the ordering conjecture.

Anyone asked to give some examples of theories which are not FC will begin from the most natural one – the infinite total ordering from Remark 3.

And it is not immediately clear how to come out with something really different. For quite some time, we believed that the following conjecture could be true:

CONJECTURE 2 (FALSE). *$\mathcal{T}$ is not FC if and only if $\mathcal{T}$ defines an ordering, by which we mean that there exists a database instance D, an infinite set $A \subseteq Chase(D, \mathcal{T})$ and a query $\Phi(x, y)$, which is a conjunctive query with projections, with two free variables, such that $Chase(D, \mathcal{T}) \not\models \exists \bar{x} \Phi(x, x)$ and Φ defines a strict total ordering on A.*

Notice how beautiful it would be. Even if our BDD/FC conjecture is true (which we believe it is) it does not give a

full explanation of the phenomenon of Finite Controllability, as they are many theories (for example guarded) which are FC but not BDD. Had Conjecture 2 be true, it would have given a sort of such explanation, and a very elegant one, since the above property of "defining an ordering" is very close to (the negation of) the standard, and very important, model-theoretic notion of stability [S69]. Besides, it could give the BDD/FC conjecture as a corollary, if we only could prove that a BDD theory never defines an ordering, which we believe should not be very hard.

Clearly, the "if" implication of the conjecture holds true: if D, A and Φ like in the conjecture existed, then $\exists x \Phi(x,x)$ would be a query false in $Chase(D,\mathcal{T})$ but true in each finite model of $\mathcal{T}, D$ (as each such finite model must contain a homomorphic image of $Chase(D,\mathcal{T})$, and some two elements of A must be mapped, by this homomorphism, to the same element of the finite structure)

However, as the following notorious example shows, the opposite implication is not true. Let $\mathcal{T}$ be:

$E(x,y) \Rightarrow \exists z\ E(y,z)$

$R(x,y), E(x,x'), E(y,z), E(z,y') \Rightarrow R(x',y')$

It is not hard to see that $\mathcal{T}$ does not define an ordering. We are going to show that $\mathcal{T}$ is not FC. Let D consist of the atoms $E(a_0,a_1)$ and $R(a_0,a_0)$.

Then $\mathcal{C} = Chase(D,\mathcal{T})$ is an infinite E-chain like in Example 3, but with additional atoms $R(a_i,a_{2i})$ for each i. Let $\Phi(x,y) = E(x,y) \wedge R(y,y)$. Clearly, $\mathcal{C} \not\models \Phi$ – the only element a_0 of $\mathcal{C}$ which satisfies $R(y,y)$ has no E-predecessor. But, as we are going to prove, if $\mathcal{M}$ is any finite model of $\mathcal{T}, D$ then $\mathcal{M} \models \Phi$.

Indeed, whatever the structure $\mathcal{M}$ is, it must contain a sequence $a_0, a_1, \ldots a_m \ldots a_{m+n}$ of elements such that $a_m = a_{m+n}$ and $\mathcal{M} \models E(a_i, a_{i+1})$ for each $i < m+n$. The datalog rule form $\mathcal{T}$ can then prove that $\mathcal{M} \models R(a_m, a_{m+(m \mod n)})$, and then that $\mathcal{M} \models R(a_{m+l \mod n}, a_{m+(m+2l \mod n)})$. Let $l = -m \mod n$. Now take $y = a_{m+l \mod n}$ and $x = a_{m+l-1 \mod n}$ to get $R(y,y)$.

It is worth mentioning that the structure $\mathcal{C}$ from the above example is ptp-conservative. As the degree of the elements of $\mathcal{C}$ is bounded by 4, this follows from:

LEMMA 13. *Each binary structure of bounded degree is ptp-conservative.*

Proof (hint): For a given number m, color the structure in such a way, that each neighborhood of radius m consists of elements whose colors are pairwise different. Then mimic the reasoning from Section 4. □

This shows ptp-conservativity of $Chase$, which – as explained in Remark 4 – guarantees that, if only n is big enough, $M_n(\overline{Chase})$ will be a model for all the existential TGDs from the theory, does not buy us much more: the devil can very well be in the plain datalog rules. Notice however, that not all datalog rules are troublemakers:

Remark 5. Suppose $\bar{\mathcal{C}}$ is n-conservative up to the size m. Let $\Psi \Rightarrow Q(x)$ be a datalog rule with at most m variables and with a unary predicate in the head. If this true in $\bar{\mathcal{C}}$ then it is also true in $M_n(\bar{\mathcal{C}})$. Proof: positive m-types of x and of $q_n(x)$ are the same.

5.6 Guarded TGDs

Guarded Datalog$^\exists$ programs, proved to be FC in [BGO10], consist of guarded rules (datalog rules and TGDs) which have an atom in the body, called the guard, containing all the variables that occur in the body of this rule. There is no restriction on the arity, in particular on the arity of the predicates in the heads of TGDs.

The witness generated by a guarded TGD appears to depend on all the variables in the head of the rule, and in consequence such a rule seems to be inherently non-binary, not even in the broad sense of Section 5.1. But, as it turns out, Guarded Datalog$^\exists$ programs are binary in disguise. And, while they are not BDD, still nothing beyond the techniques developed in Sections 2 and 4 is needed to prove they are FC.

To be more precise, suppose there exists a Guarded Program $\mathcal{T}$, a database instance D and a query Φ which are a counterexample for FC. Of course D can be also hardwired into $\mathcal{T}$ so we can assume it is empty.

Now we will show how to rewrite $\mathcal{T}$ and Φ into a binary signature, without changing their status of a counterexample. Then we will use the toolkit from Sections 2 and 4 to show very easily that the resulting binary program is FC.

(i) First step is similar to the one in the end of Section 3.1 – we want the predicates which are in the heads of TGDs (the TGPs) to be distinct that the ones in the heads of datalog rules. We also want the rules to respect the order of variables in atoms – if x is left of y in some atom in the rule then x never can be right of y in any atom of the same rule. This can be done by remembering the order of arguments as a part of the name of each predicate. Of course Φ must be rewritten – each new predicate is now a disjunction of the old predicates. Notice that guardedness implies that if $\mathcal{T}$ respects the order of variables, if $Chase(\mathcal{T}) \models R(\bar{a}, c)$ for some TGP R and if $Chase(\mathcal{T}) \models P(\bar{b}, c)$ then $\bar{b} \subseteq \bar{a}$. The elements in $\bar{a}$ are "parents of c", who were present in the atom R when c was born, and no rule can add anything else left of c in any atom.

Rename the variables in each rule in such a way, that the rightmost variable of the guard of each rule is y. Call this y the leading variable of the rule.

(ii) We want the elements to know their parents by name. If $R(x_1, \ldots x_k, y)$ is a TGP in $\mathcal{T}$ we add to T new rules:

$$R(x_1, \ldots x_k, y) \Rightarrow F_i(x_i, y)$$

for each $i \leq k$, where F_i are new binary predicates.

(iii) Replace each TGD of the form $\Psi \Rightarrow \phi$, with the leading variable y, with all possible rules of the form:

$$(\spadesuit\ 6.) \qquad \Psi \wedge F_{i_1}(x_1, y) \wedge \ldots \wedge F_{i_k}(x_k, y) \Rightarrow \phi$$

where $x_1, \ldots x_k$ are all the non-leading variables in Ψ and $i_1, \ldots i_k \leq K$, where K is the maximal arity of the predicates in $\mathcal{T}$. This changes nothing, as the elements to be substituted for x_i must have been some parents of y anyway.

(iv) Now, again in the manner of Section 3.1 rewrite the current $\mathcal{T}$ and Φ in such a way, that each TGP only occurs in one rule head (this can be easily done for the cost of some renaming datalog rules).

(v) Now perform step **(iii)** for the datalog rules of the current program.

At some point in Section 5.1 we wrote: *All we need in the proof in Section 3 is that (...) the witness generated by the rule only depends on one element in the body (the y), while the additional elements in the body are just needed to make sure that y has a positive type which allows it to demand a witness.* Notice that this is exactly the case with

our program now: all rules are in the form (♠ 6.) and the elements of $Chase(\mathcal{T})$ that can possibly be substituted for elements of $\bar{x}$ in the body of such rule are themselves functions of y. So t is just a function of y, not of all the elements of $\bar{x}$! The only reason why non-binary predicates could be necessary does not exist any more. Let us get rid of them.

Since the original $\mathcal{T}$ was guarded, each atom $P(\bar{a})$ in $Chase(\mathcal{T})$ was contained in some TGP atom $R(\bar{b}, c)$. Our idea is that full information about $P(\bar{a})$ will be remembered, **without materializing** $P(\bar{a})$, in a monadic way, by the element c. It will need to remember which of its parents are involved in each predicate. Of course it also needs to remember the links to its parents – this is why the relations F_i were introduced.

(vi) Replace each TGD of the form: $\Psi \Rightarrow \exists z\ R(x_1, \ldots x_k, z)$ (x_k may, or may not, by equal to y) by the following rules: $\Psi \Rightarrow \exists z\ E^R(y, z)$ and $\Psi \wedge E^R(y, z) \Rightarrow R^m(z)$ where E^R is a new binary predicate. $E^R(y, z)$ means something like "the (unique) rule which derives R was applied to a tuple led by y and a witness z was created". The newly created element z must also learn who its parents are. For each $i \in \{1, \ldots k\}$, if $F_j(x_i, y)$ was an atom in Ψ, add to the current $\mathcal{T}$ the rule:

$$(\diamondsuit) \qquad F_j(x_i, y) \wedge E^R(y, z) \Rightarrow F_i(x_i, z)$$

and replace each TGP atom $R(x_1, \ldots x_k, z)$ in the body of any rule by:

$$F_1(x_1, z) \wedge \ldots F_k(x_k, z) \wedge R^m(z)$$

Now the program does not have TGPs of arity higher than 2 any more. Notice that for each variable $x \neq y$ in any rule, there exists i such that the atom $F_i(x, y)$ is in the body of this rule. We are ready to get rid also of the non-TGPs:

(vii) In each rule, with the atoms $F_{i_1}(w_1, y), \ldots F_{i_1}(w_l, y)$ in its body, replace each occurrence of a non-TGP atom $Q(w_1, \ldots w_l)$ with $Q_{i_1 i_2 \ldots i_l}(y)$, where $Q_{i_1 i_2 \ldots i_l}(y)$ is a new monadic predicate (in which y remembers what his parents with numbers $i_1, i_2, \ldots i_l$ are involved in). For each two monadic predicates of the form $Q_{i_1 i_2 \ldots i_l}(y)$ and $Q_{j_1 j_2 \ldots j_l}$ add to the program all possible rules of the form:

$$F_{i_1}(x_1, y) \ldots \wedge F_{i_l}(x_l, y) \wedge F_{j_1}(x_1, z) \ldots \wedge F_{j_l}(x_l, z) \wedge$$
$$\wedge Q_{i_1 i_2 \ldots i_l}(y) \Rightarrow Q_{j_1 j_2 \ldots j_l}(z)$$

The role of the last rule is to make sure that, once an atom of the predicate Q, involving $x_1 \ldots x_l$ is derived, all the elements that have $x_1 \ldots x_l$ among their parents are aware of that and ready to use this fact in further derivations.

We now have a new program over a binary signature, call it $\mathcal{T}'$. It follows from the construction that $\mathcal{C} = Chase(\mathcal{T}')$ is almost the same structure as $Chase(\mathcal{T})$ (where $\mathcal{T}$ is the original guarded program). They both have the same elements, and the predicates of each of them can be seen as views over the predicates of the other one. But notice that $Chase(\mathcal{T}')$ is a binary structure satisfying the assumptions of of Lemma 2. So it is ptp-conservative. This means that if n is big enough then $M_n(\bar{\mathcal{C}})$ is a model for all the existential TGDs in $\mathcal{T}'$ and that $M_n(\bar{\mathcal{C}}) \not\models \Phi'$ (where Φ' is the original query Φ after all the rewritings). To finish the proof of FC for Guarded Datalog$^\exists$ programs we only need to show that $M_n(\bar{\mathcal{C}})$ is also a model of all the datalog rules in $\mathcal{T}'$. All the datalog rules except from the rules of the form ($\diamondsuit$) have a unary atom in the head, so (by Remark 5) we do not need to bother about them at all. What remains to be seen is that the rules of the form ($\diamondsuit$) also remain true in $M_n(\bar{\mathcal{C}})$. So suppose $M_n(\bar{\mathcal{C}}) \models F_j(a, b) \wedge E^R(b, c)$ for some a, b, c. This means that there exist a', b', b'', c'' in $\bar{\mathcal{C}}$ such that $\bar{\mathcal{C}} \models F_j(a', b')$, $\bar{\mathcal{C}} \models E^R(b'', c'')$, $q_n(a') = a$, $q_n(b') = q_n(b'') = b$ and $q_n(c') = c$. But $\bar{\mathcal{C}}$ can be seen as a Chase of the guarded theory $\mathcal{T}$ with the natural coloring, so the types of successors of an element only depend on the type of this element, and it is easy to see that if $b' \equiv_n b''$ and $\bar{\mathcal{C}} \models E^R(b'', c'')$ then there must exist $c' \equiv_n c''$ such that $\bar{\mathcal{C}} \models E^R(b', c')$. Since the rule ($\diamondsuit$) was true in $\bar{\mathcal{C}}$ we get that $\bar{\mathcal{C}} \models F_i(a', c')$ which implies that $M_n(\bar{\mathcal{C}}) \models F_i(a, c)$. We proved that $\diamondsuit$ remains true in $M_n(\bar{\mathcal{C}})$.

6. REFERENCES

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

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

[CGT12] A. Cali, G. Gottlob, and T. Lukasiewicz; *A general datalog-based framework for tractable query answering over ontologies*; J. Web Sem. 14, 2012, 57-83

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

[CGP10+/-] A. Cali, G. Gottlob, and A. Pieris; *Query Answering under Non-guarded Rules in Datalog+/-* ; Web Reasoning and Rule Systems Lecture Notes in Computer Science, 2010, Volume 6333, pp 1-17;

[GM12] T. Gogacz, J. Marcinkowski; *Converging to the Chase and Some Finite. Controllability Results*; Proc. of the 28th IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, USA, to appear;

[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;

[R11] R. Rosati; *On the decidability and finite controllability of query processing in databases with incomplete information*; J. Comput. Syst. Sci. 77(3),2011, pp. 572-594

[S69] S. Shelah; (1969), *Stable theories*; Israel J. Math. 7 (3): 187-202.