What Makes a Variant of Query Determinacy (Un)Decidable?

Jerzy Marcinkowski
Institute of Computer Science, University of Wrocław, Poland

Abstract

This paper was written as the companion paper of the ICDT 2020 invited tutorial. Query determinacy is a broad topic, with literally hundreds of papers published since late 1980s. This paper is not going to be a "survey" but rather a personal perspective of a person somehow involved in the recent developments in the area.

First I explain how, in the last 30+ years, the question of determinacy was formalized. There are many parameters here: obviously one needs to choose the query language of the available views and the query language of the query itself. But – surprisingly – there is also some choice regarding what the word "to compute" actually means in this context.

Then I concentrate on certain variants of the decision problem of determinacy (for each choice of parameters there is one such problem) and explain how I understand the mechanisms rendering such variants of determinacy decidable or undecidable. This is on a rather informal level. No really new theorems are presented, but I show some improvements of existing theorems and also simplified proofs of some of the earlier results.

2012 ACM Subject Classification Theory of computation → Database theory

Keywords and phrases database theory, query, view, determinacy

Digital Object Identifier 10.4230/LIPIcs.ICDT.2020.2

Category Invited Talk

Funding *Jerzy Marcinkowski*: Supported by the Polish National Science Centre (NCN) grant 2016/23/B/ST6/01438.

Acknowledgements The author would not have been in a position to write this invited paper had he not been first blessed with the opportunity to work with extraordinary students: Tomasz Gogacz, Grzegorz Głuch and Piotr Ostropolski-Nalewaja.

1 Introduction (1)

"Assume that a set of derived relations is available in a stored form. Given a query, can it be computed from the derived relations and, if so, how?" is the first sentence of [10], the oldest paper I know addressing the Query Determinacy Problem (QDP). On the very informal level this first sentence still does very good job explaining the idea of QDP. But in order to be really able to work on it, to formulate theorems and to try to prove them, we need to be a bit more precise.

And, as it turns out, there is a huge number of ways in which one can be more precise, each way leading to one variant of QDP. One can choose (1) between various query languages defining the stored views and (2) defining the given query, (3) between information-theoretic notion of determinacy (this paper) and various rewriting languages (not this paper), (4) between considering only finite database instances (QDP^f), or any relational structures, finite or infinite (QDP^∞), and finally (5) between *exact* (QDP_e) and *sound* (QDP_s) semantics. Let us first stick to the exact semantics, which is (I think) simpler to understand (sound semantics will be briefly discussed in Section 7).

Each variant of QDP is a decision problem. The instance is always a set of queries $\mathcal{V} = \{V_1, \ldots V_k\}$ in query language $\mathcal{L}_V$ and another query Q in some query language $\mathcal{L}_Q$. We are asked whether it is true that *for each database instance* D the query $Q(D)$ can be computed by a *device* from some set $\mathcal{C}$, from the set of views $V_1(D)$, $V_2(D) \ldots$, $V_k(D)$.

We are going to use notation $QDP_e^{-}(\mathcal{L}_V, \mathcal{L}_Q, \_)$ to denote the variant of QDP, under exact semantics, with arguments reflecting our choice of parameters. So, for example, $QDP_e^f(UCQ, UCQ, FO)$ will be the version of QDP where views are defined by queries $V_1, \ldots V_k$ being unions of conjunctive queries, query Q is also a union of conjunctive queries, only finite instances are allowed, and we ask whether there is a First Order rewriting of a given query Q, computing, regardless of D, the query $Q(D)$, when applied to the views $V_1(D)$, $V_2(D) \ldots$, $V_k(D)$. As another example $QDP_e^{\infty}(CQ, CQ)$ will denote the problem of deciding, for a set of conjunctive queries $V_1, \ldots V_k$, and another conjunctive query Q, whether $Q(D)$ can **in any way**[1] be computed from $V_1(D)$, $V_2(D) \ldots$, $V_k(D)$ for any (possibly infinite) structure D. Notice that we do not assume that it needs to be computed by an algorithm, or a Turing machine[2]. The only thing that matters is that the complete information about $Q(D)$ is already in $V_1(D)$, $V_2(D) \ldots$, $V_k(D)$. In other words, and more precisely:

▶ **Definition 1.** *We say that a set $\mathcal{V} = \{V_1, V_2 \ldots, V_k\}$ of queries, written in some query language $\mathcal{L}_V$ (finitely)* ***determine*** *query Q, written in some query language $\mathcal{L}_Q$, denoted as $[\mathcal{V}, Q] \in QDP_e^{\infty}(\mathcal{L}_V, \mathcal{L}_Q)$ (resp. $QDP_e^f(\mathcal{L}_V, \mathcal{L}_Q)$) if for each two database instances (resp. finite database instances) D, D' such that for each $1 \leq i \leq k$ there is $V_i(D) = V_i(D')$ there is also $Q(D) = Q(D')$.*

A pair of structures D, D' as above will be sometimes called a counterexample for determinacy.

In this paper we will mainly be interested in complexity (or rather decidability) of problems of the form $QDP_e^{-}(\_,\_)$ which we think of as of proper determinacy problems, rather than "rewriting" problems (which require different methods). Among them, we mainly concentrate on problems of the form $QDP_e^{\infty}(\_,\_)$. But, at least when talking about the negative results, this is only for convenience – analysis for the unrestricted case is simpler and more intuitive than in the finite case, but does not require, as far as we know, significantly different tools than the respective $QDP_e^f(\_,\_)$ variants[3].

2 Preliminaries

We mainly try to use standard notions and notations of relational database theory. In particular, for a query Ψ, with k free variables, and for a database instance D, the notation $\Psi(D)$ denotes the k-ary relation resulting from applying the query Ψ to D. For a conjunctive

[1] The distinction between query rewritability and query determinacy in the information-theoretic sense, as considered in this paper, was not really fully realized before the end of 1990s. The earliest paper I know which makes a clear distinction is [3]. Let me quote it here: *Unfortunately, many of* [the previous papers on view-based query answering] *do not distinguish between view-based query answering and view-based query rewriting, and give raise to a sort of confusion between the two notions. [...] So, in spite of the large amount of work on the subject, the relationship between view-based query rewriting and view-based query answering is not completely clarified yet.*

[2] There are only two arguments now – since there is no rewriting now there is also no need to specify the rewriting language.

[3] In order to translate an undecidability proof for some $QDP_e^{\infty}(\mathcal{L}_V, \mathcal{L}_Q)$ to a proof of the same result for $QDP_e^f(\mathcal{L}_V, \mathcal{L}_Q)$ one usually needs to recall what recursively inseparable sets are.

query ψ by the frozen body of ψ, denoted as $[\psi]$, we mean the relational structure[4] (unique up to isomorphism) whose elements[5] are the variables of ψ and whose relational atoms are the atomic formulas of ψ.

A tuple generating dependency (TGD) is a formula of the form: $\forall \bar{x}, \bar{y}\ (\phi(\bar{x}, \bar{y}) \Rightarrow \exists \bar{z}\ \psi(\bar{x}, \bar{z}))$, where ϕ and ψ are conjunctions of atomic formulas (which means that our TGDs are "multi-head"). Formula ϕ is called the *body* of the TGD and ψ is its *head*. The universal quantifier in front of the formula is always omitted and implicit.

Another notion we often use is the one of disjunctive TGD. They are like TGDs except that ψ is now of the form $\psi = \bigvee_{1\leq i\leq k} \psi_i(\bar{x}, \bar{z}_i)$, for some k, where each ψ_i is a conjunction of atomic formulas and each $\bar{z}_i$ is a subtuple of $\bar{z}$ (in this way we are trying to say that there is the same set $\bar{x}$ of free variables in each of the disjuncts of ψ, but the sets of quantified variables may differ). An analogous condition holds for ϕ.

2.1 The zoo of query languages

Let us now define the classes of languages which occur as $\mathcal{L}_Q$ or as $\mathcal{L}_V$.

We have two kinds of them. First kind are the fundamental languages from the relational databases tradition. CQ is the language of conjunctive queries, being conjunctions of atomic formulas preceded by some existential quantifiers. UCQ are unions of conjunctive queries, which means they are formulas of the form $\psi = \bigvee_{1\leq i\leq k} \exists \bar{z}_i\ \psi_i(\bar{x}, \bar{z}_i)$, for some k, where each ψ_i is a conjunction of atomic formulas (notice that we again assume that free variables are the same in each disjunct).

We will also consider languages of unary (monadic) CQs and unary (monadic) $UCQs$ (notations mCQ and $mUCQ$ will be used). These are queries with only one free variable (so that $\Psi(D)$ is a unary relation, that is a set of elements of D).

DL queries are ones defined by Datalog programs. We only consider boolean DL queries, which means that there is always one special arity zero predicate *goal* in each program, and, for a program P and a database instance D we think that $P(D)$ is true if P proves *goal* in D. Monadic Datalog (mDL) are DL queries defined by a program with all the IDB predicates (this means, the predicates that occur in a head of any rule) being unary.

Then there is the second sort of query languages, coming from the tradition of graph databases. Path Queries (PQ) are conjunctive queries of the form:

$$\exists z_1, z_2, z_{m-1}\ E_1(x, z_1) \wedge E_2(z_1, z_2) \wedge \ldots E_m(z_{m-1}, y)$$

A Path Query always has two free variables. It says, about two elements of the structure, that there is a path between them, labelled with the particular sequence (word) of labels (predicate names). We identify such query with this word, for example the above PQ is usually denoted as simply $E_1 E_2 \ldots E_m$.

A union of Path Queries (UPQ) is a UCQ whose CQs are path queries. Hence it also has two free variables, and it says, about x and y, for some finite set of words W, that there is a path, from x to y, labelled with a $\mathbf{w} \in W$.

Finally, a Regular Path Query (RPQ) is like a union of Path Queries, but the set W no longer is assumed to be finite: it can be any regular set over the language of labels. In a sense RPQ is, for the graph databases tradition, what DL is for the relational databases one.

[4] The terms "relational structure" and "database instance" mean the same thing for us.
[5] Relational structures have elements (vertices). We never call them "constants" or "nulls": the term "constant" is reserved (according to the tradition of mathematical logic) for the constants of the language.

2.2 Unfoldings of Datalog programs

We often want to build a minimal structure satisfying some query Ψ. The word "minimal" means minimal from the point of view of positive information. For such minimal structure M and for any other structure D with $D \models \Psi$ we would like to be sure that a homomorphism exists from M to D. This is easy for CQs – the frozen body of Ψ is always such a minimal structure. If Ψ is a UCQ we no longer have **the** minimal M, but we have a finite number $M_1, M_2 \ldots M_k$ of structures (frozen bodies of the disjuncts of Ψ) such that if $D \models \Psi$ then we are sure that there exists a homomorphism from one of the M_i to D.

But how about Datalog queries? Here we no longer have a finite set of such minimal structures (unless the program in question is bounded). The minimal structures are now the unfoldings of the program – all the possible ways of proving *goal*. It will be helpful at some point to see that:

▶ **Exercise 2.** *For each Datalog program P there exists a constant c such that all the unfoldings of P have tree width bounded by c.*

3 Introduction (2)

3.1 Examples

Let us see an example or two.

Suppose Q is a Path Query $BACA$ (see Section 2.1 for explanation) and:

$$\mathcal{V}_1 = \{BAC, ACA, AC\} \qquad \mathcal{V}_2 = \{BAC, ACA, CA\}$$

Then $[\mathcal{V}_1, Q] \in QDP_e^{\infty}(PQ, PQ)$ while $[\mathcal{V}_2, Q]$ is a negative instance of $QDP_e^{\infty}(PQ, PQ)$ – there is no determinacy.

In order to prove the second claim (the one about $[\mathcal{V}_2, Q]$) it is enough to take:

$$D_1 = \{B(a,b), A(b,c), C(c,d), A(d,e)\}$$

$$D_2 = \{B(a,b'), A(b',c'), C(c',d), A(b,c), C(c,d'), A(d',e)\}$$

Then $\mathcal{V}_2(D_1) = \mathcal{V}_2(D_2)$ (exercise!) and $D_1 \models Q$ but $D_2 \not\models Q$. This was easy. But how could we possibly prove the first claim?

3.2 What is this paper about?

Definition 1 is precise and simple. But it does not usually mean that for given $\mathcal{L}_V, \mathcal{L}_Q$ one can easily figure out if the problem $QDP_e^{\infty}(\mathcal{L}_V, \mathcal{L}_Q)$ (or $QDP_e^{f}(\mathcal{L}_V, \mathcal{L}_Q)$) is decidable or not.

The first obstacle is that it is not obvious at all how to work with this definition. How can we make sure that something is true for "each two database instances"?

To deal with this problem we ([8]) invented the notion of Green-Red Chase. The idea to explain determinacy in terms of Chase was not totally absent in some of the previous papers, including [11]. The Green-Red Chase is just a very little step forward. But, as we are going to explain, due to this little step we suddenly can see things we were unable to notice before.

Green-Red Chase will be introduced in Section 4. Then, in Section 5 we are going to show how the insight provided by Green-Red Chase can be very easily used to prove some positive results.

It was shown in [11] that $QDP_e^\infty(mCQ, CQ)$ is decidable. In Section 5.1 the Reader is going to see how this result can be significantly strengthened, and without using the complicated argument from [11]. We will show that, for example, $QDP_e^\infty(mUCQ, CQ)$ is decidable. And everything will be very simple, almost trivial.

Another known positive result is the one from [1] (later slightly improved by [12]). It is shown, in [A11], that $QDP_e^\infty(PQ, PQ)$ is decidable. In Section 5.2 we are going to build on the top of the technique from [1] to show a slightly stronger result, that $QDP_e^\infty(PQ, RPQ)$ is also decidable.

Both the techniques we present in Section 5 rely, apart from the insight given by the Green-Red Chase, on simple automata-theoretic arguments.

Then we are going to move to the negative results. It was proved in the paper [13] that $QDP_e^\infty(UCQ, UCQ)$ is undecidable. Then, in [6] we have shown undecidability of $QDP_e^\infty(RPQ, RPQ)$ (solving a problem left open in the series of papers on "loselessness" of Regular Path Queries, including [4]). In [7] we refined the technique from [6] showing that $QDP_e^\infty(UPQ, UPQ)$ is also undecidable (notice that this result is also strictly stronger than the aforementioned negative result from [13] about $QDP_e^\infty(UCQ, UCQ)$). As it turns out[6], the technique in [6] and [7] relies mainly on, as I call it here, the Cold/Hot Trick. Thanks to this trick one can easily encode a Turing machine computation inside the Green-Red Chase. All that is needed is disjunction, both in $\mathcal{L}_V$ and in $\mathcal{L}_Q$. In Section 6 we explain the trick and show how it can be used to prove undecidability of $QDP_e^\infty(UCQ, UCQ)$.

But how about query languages without disjunction, where the Cold/Hot Trick does not work? It turns out that proving lower bounds for variants of the form $QDP_e^\infty(CQ, \_)$ is quite hard. Indeed, the only negative result for a variant of this form known before 2015 was the one from [5] where undecidability of $QDP_e^\infty(CQ, DL)$ is proved. And, as we explain in Section 7, this result does not really, on the technical level, have much to do with the phenomenon of determinacy. It is just a simple consequence of undecidability of Datalog programs containment.

In Section 4.3 we reveal the reason behind this hardness, which is the Curse of Pâte Feuilletée[7]. Then, in Section 8 we show how to break the Curse of Pâte Feuilletée. This is the most complicated part of this paper and we are only able to explain some ideas of our proof technique from [8] and [9], where undecidability of $QDP_e^\infty(CQ, CQ)$ and of $QDP_e^f(CQ, CQ)$ is proven.

In the meantime, in Section 7, we report some work in progress [2] regarding the sound semantics, which means variants of the form $QDP_s^\infty(\_, \_)$. We explain how Green-Red Chase is also relevant in this case, and also how the simple automata-theoretic techniques from Section 5.1 can be helpful there. We also remark that the Cold/Hot Trick can be useful for the variants of $QDP_s^\infty(\_, \_)$ where disjunction is available, and that the Curse of Pâte Feuilletée seems to be in force for at least one variant of $QDP_s^\infty(\_, \_)$ (for which decidability of determinacy remains open).

4 The Green-Red Chase

The notion of Chase is one of the ubiquitous notions of database theory. Suppose we have a set $\mathcal{T}$ of TGDs, a database instance D and a conjunctive query Q. And, for some reason, we want to know whether $\mathcal{T}, D \models Q$, which means that Q is true in all superstructures D^+ of

[6] From the perspective, it seems to me, that when writing [6] and [7] we did not understood the technique well enough yet, and we did a poor job explaining what constitutes the core of the technique there and what is the implementation.

[7] This is how we called it with Tomek Gogacz, who was my student at that time, when we struggled, in the years 2012-13, to solve the problem of the decidability status of $QDP_e^\infty(CQ, CQ)$.

D satisfying all the TGDs in $\mathcal{T}$. Then we can construct $Chase(\mathcal{T}, D)$ and check whether $Chase(\mathcal{T}, D) \models Q$. If Q is satisfied in $Chase(\mathcal{T}, D)$ then it is satisfied in all structures D^+ as above[8]. Due to this property $Chase(\mathcal{T}, D)$ is called *a universal structure*.

How is this universal structure built? We start from[9] $Chase_0 = D$. Suppose that $Chase_i$ is defined. Then $Chase_{i+1}$ is defined by applying the following step, in parallel, to all possible TGDs $T \in \mathcal{T}$ and all tuples $\bar{c}$ of elements of $Chase_i$: suppose the body $\phi(\bar{c}, \bar{l})$ of T is satisfied in $Chase_i$ for some tuple $\bar{l}$ and that T postulates that there exists some tuple of elements $\bar{k}$ which, together with the tuple $\bar{c}$ of elements of $Chase_i$, satisfy the head $\psi(\bar{c}, \bar{k})$. Then we simply invent a tuple $\bar{k}$ of new elements, and add them to $Chase_i$ together with the atoms which occur in ψ. And – importantly – we do it **in the minimal way, from the point of view of the amount of positive knowledge**: the elements that are being added are all new, and they are never equal unless the TGD T explicitly requires them to be equal.

Finally, $Chase(\mathcal{T}, D) = \bigcup_{i \in \mathbb{N}} Chase_i$ turns out to be the universal structure.

We use the same idea in the context of Definition 1. Imagine we have an instance of the problem $QDP_e^{\infty}(\mathcal{L}_V, \mathcal{L}_Q)$. Such an instance, to recall, will be a pair $\mathcal{I}$ consisting of a set of $\mathcal{V}$ of queries and a query Q. Following the idea of Chase, in order to check whether there is determinacy, we should try to construct a "universal counterexample" – a pair of database instances" D_G (as *green*) and D_R (as *red*) such that ($\clubsuit_1$) $V(D_G) = V(D_R)$ for each $V \in \mathcal{V}$, that[10] ($\clubsuit_2$) $D_G \models Q$, and that the two conditions:

$$(\spadesuit_1)\ D_R \models Q \qquad \text{and} \qquad (\spadesuit_2)\ \mathcal{I} \in QDP_e^{\infty}(\mathcal{L}_V, \mathcal{L}_Q)$$

are equivalent, which means that if there exists any counterexample for determinacy for this instance then D_G and D_R are such an example.

4.1 Green and red structures and queries

We prefer however (and this is exactly the idea of the Green-Red Chase from [GM15]) to, instead of constructing two structures D_G and D_R, over some signature Σ, construct a single structure over a new signature $\Sigma_G \cup \Sigma_R$ consisting of colored (green and red) versions of the predicates from Σ.

To speak about objects over this new signature it will be convenient to have two operators: G and R, painting any object over Σ green or red. So, for example, for a query Φ over Σ we will have its red version $R(\Phi)$, over Σ_R. Another operator we will sometimes need is daltonization (denoted $dalt()$). It takes green or red objects (over the signature $\Sigma_G \cup \Sigma_R$) and returns the same objects with colors removed (over Σ).

Now, instead of producing two structures over the signature Σ of $\mathcal{I}$ we will construct one structure D_{GR} over $\Sigma_G \cup \Sigma_R$ such that for each $V \in \mathcal{V}$ there will be ($\clubsuit_1$) $R(V)(D_{\mathrm{GR}}) = G(V)(D_{\mathrm{GR}})$, such that ($\clubsuit_2$) $D_{\mathrm{GR}} \models G(Q)$ and that the conditions:

$$(\spadesuit_1)\ D_{\mathrm{GR}} \models R(Q) \qquad \text{and} \qquad (\spadesuit_2)\ \mathcal{I} \in QDP_e^{\infty}(\mathcal{L}_V, \mathcal{L}_Q)$$

are equivalent.

[8] The "only if" direction is of course trivially true.
[9] From now on we will skip the arguments of Chase: we will write simply $Chase_n$ instead of $Chase_n(\mathcal{T}, D)$. Unless we think this can lead to any confusion.
[10] As it turns out, one can restrict the attention to Q being a boolean query, by which we mean a query without free variables.

4.2 The Green-Red chase. The CQ case.

Now, to begin with, imagine the simplest case, that $\mathcal{L}_V = \mathcal{L}_Q = CQ$.

We want to construct D_{GR} satisfying ($\clubsuit_1$) $R(V)(D_{GR}) = G(V)(D_{\text{GR}})$ for each $V \in \mathcal{V}$. Let V be $\exists \bar{x}\ \phi(\bar{x}, \bar{y})$, where ϕ is a conjunction of atoms. Then ($\clubsuit_1$) is equivalent to the conjunction of two TGDs:

$$G(\phi)(\bar{x}, \bar{y}) \Rightarrow \exists \bar{x}'\ R(\phi)(\bar{x}', \bar{y}) \qquad\qquad R(\phi)(\bar{x}, \bar{y}) \Rightarrow \exists \bar{x}'\ G(\phi)(\bar{x}', \bar{y})$$

Let $\mathcal{T}_{\mathcal{V}}$ be the set of all TGDs generated in this way from the queries in $\mathcal{V}$.

Since we also want to have $D_{\text{GR}} \models G(Q)$, first take a *minimal, from the point of view of the amount of positive knowledge* structure which satisfies $G(Q)$, and see it as $Chase_0$. In the CQ case, which we now consider, there is just one such minimal structure: the frozen body of $G(Q)$. So $Chase_0 = G([Q])$.

And now it follows from all we know about the Chase, that $D_{\text{GR}} = Chase(\mathcal{T}_{\mathcal{V}}, G([Q]))$ **is indeed the universal structure** we were looking for.

4.3 Discussion and the Curse of Pâte Feuilletée

Let us go back to the instance $[\mathcal{V}_1, Q]$ from Section 3.1. One of the views from $\mathcal{V}_1$, is the conjunctive query BAC, or $\exists z_1, z_2\ B(x, z_1), A(z_1, z_2), C(z_2, y)$. Then the two Green-Red TGDs generated by BAC are:

$(\heartsuit_{gr})$ $G(B)(x, z_1), G(A)(z_1, z_2), G(C)(z_2, y) \Rightarrow \exists z_1', z_2'\ R(B)(x, z_1'), R(A)(z_1', z_2'), R(C)(z_2', y)$

and:

$(\heartsuit_{rg})$ $R(B)(x, z_1), R(A)(z_1, z_2), R(C)(z_2, y) \Rightarrow \exists z_1', z_2'\ G(B)(x, z_1'), G(A)(z_1', z_2'), G(C)(z_2', y)$

Imagine how $(\heartsuit_{gr})$ is applied. Suppose we have some $Chase_{2k}$ already constructed[11] and there are some a_1, a_2, a_3 and a_4 there, which form a green BAC-path in this $Chase_{2k}$ (or, in other words, we have a green copy of the frozen body of BAC). The TGD $(\heartsuit_{gr})$ tells us that there also should also be such red BAC-path, from a_1 to a_4. If there is no such path in $Chase_{2k}$ then a new copy $R(B)(a_1', a_2'), R(A)(a_2', a_3'), R(C)(a_3', a_4')$ of the frozen body of $R(BAC)$ is created, it is added to $Chase_{2k}$, with a_1' identified with a_1 in the new structure and a_4' identified with a_4. The vertices a_2' and a_3' are not identified with anything in the old structure – they are new in $Chase_{2k+1}$.

An important take away from the example is that when constructing $Chase_{n+1}$ from $Chase_n$ we produce many copies of the (colored versions) of the frozen bodies of queries in $\mathcal{V}$ and join them with $Chase_n$ by identifying the elements that relate to the free variables of the respective V (like the x and y in the example) with elements of $Chase_n$. The elements that relate to the quantified variables of the respective V are the "new" elements of $Chase_{n+1}$.

Back to the $[\mathcal{V}_1, Q]$ from Section 3.1. Now we can prove that indeed $[\mathcal{V}_1, Q] \in QDP_e^{\infty}(PQ, PQ)$. Let us run the Green-Red chase. There will be[12] $Chase_0 = \{G(B)(a, b), G(A)(b, c), G(C)(c, d), G(A)(d, e)\}$ for some elements a, b, c, d, e. Then, $Chase_1 = Chase_0 \cup \{R(B)(a, b_1), R(A)(b_1, c_1), R(C)(c_1, d), R(A)(b, c_2), R(C)(c_2, d), R(A)(b, c_3), R(C)(c_3, d_3)\}$

There will be also, among some other atoms: $G(A)(b_1, c_4), G(C)(c_4, d)$ in $Chase_2$ and $R(A)(b_1, c_5), R(C)(c_5, d_5)$ and $R(A)(d_5, e)$ in $Chase_3$. But the last three atoms, together with $R(B)(a, b_1)$ form $R([Q])$ and, by universality of the Green-Red chase we get that $[\mathcal{V}_1, Q] \in QDP_e^{\infty}(PQ, PQ)$.

[11] Why $2k$? Wait, Observation 3 is coming.
[12] The Reader is invited to run the chase herself, to make sure that what I write here makes sense.

▶ **Observation 3.** *When constructing $Chase_{2k}$ only green atoms are added. When constructing $Chase_{2k+1}$ only red atoms are added.*

Proof. Induction. Clearly, since there is nothing red in $Chase_0$ only red atoms will be added when constructing $Chase_1$. For the induction step, since nothing red (resp. green) was added while constructing $Chase_{2k}$ (resp. $Chase_{2k+1}$), all the TGDs of the form ($\heartsuit_{rg}$) (resp. ($\heartsuit_{gr}$)) TGDs are satisfied in $Chase_{2k}$ (resp. $Chase_{2k+1}$). ◀

For the following Observation recall that we still assume that $\mathcal{L}_V = CQ$.

▶ **Observation 4.** *For each $n \in \mathbb{N}$ there exists a homomorphism h_n from $dalt(Chase_n)$ to $dalt(Chase_0)$, with $h_n \subseteq h_{n+1}$.*

Similar observation, in the context of the chase with two separate structures rather than one green-red structure can be found in [13].

Proof. Induction. Clearly, h_0 is the identity. For the induction step, in order to keep the notations light, we will use the above example. Suppose some elements of $Chase_{n+1}$ were added by an application of the ($\spadesuit_{gr}$), as in the example. Then define $h_{n+1}(a_2') = h_n(a_2)$ and $h_{n+1}(a_3') = h_n(a_3)$. For any element a of $Chase_n$ define $h_{n+1}(a) = h_n(a)$. ◀

It immediately follows from the observation that $\bigcup_{n\in\mathbb{N}} h_n$ is a homomorphism from $\bigcup_{n\in\mathbb{N}} dalt(Chase_n)$ to $dalt(Chase_0)$.

The Curse of Pâte Feuilletée. When trying to prove a lower bound for a problem of the form $QDP_e^\infty(CQ, \_)$ one soon realizes that Observation 4 is the main obstacle. How can we possibly encode anything in the structure of $Chase$ if all we get there is basically $Chase_0$ repeated infinitely many times? Each time something new is added to the structure it is merely a (re-colored) version of something that already was there. What we get is a pâte feuilletée, with infinite number of almost identical layers, each of them being a copy of $G([Q])$ and nothing but air between them.

4.4 The Green-Red Chase. The non-CQ case.

Now let $\mathcal{L}_V = \mathcal{L}_Q = UCQ$. Imagine we have an instance $[\mathcal{V}, Q]$ of $QDP_e^\infty(UCQ, UCQ)$ and recall that we assume that for each $V \in \mathcal{V}$ all the disjuncts of V have the same free variables.

Again, we want to have $D_{\text{GR}} \models G(Q)$. But no longer **the** *minimal, from the point of view of the amount of positive knowledge* structure which satisfies $G(Q)$ exists. There are several such minimal structures, namely the (green versions) of the frozen bodies of the CQs being the disjuncts of Q.

We need to choose one of these disjuncts, call it Q', and put $Chase_0 = G([Q'])$.

But who is this "we" here? There was no need to ask this question in Section 4.2 as the Chase there was a normal, deterministic chase, leading to the same result regardless of who performs it. But now there is a choice, so who makes it and with what goal on mind?

In [6] and [7] we do not use the word "Chase" at all in this context. Instead, we consider a game, with a single player, called the Fugitive, and we call this version of Chase "the game of Escape". The goal of this player is to show that the QDP instance in question is a negative one – there is no determinacy. In the process of his Escape the Fugitive tries to construct a green-red structure D_{GR} for which it holds that ($\clubsuit_1$) $G(\mathcal{V})(D_{\text{GR}}) = R(\mathcal{V})(D_{\text{GR}})$ and ($\clubsuit_2$) $D_{\text{GR}} \models G(Q)$ but $D_{\text{GR}} \not\models R(Q)$.

So it is the Fugitive who begins the game choosing some disjunct Q' of Q, and defining $Chase_0$ as $G([Q'])$. At this point he is already sure that $(\clubsuit_2)$ is satisfied. But how about $(\clubsuit_1)$?

Like in Section 4.2 we need to ask what it means, for a structure D_{GR}, that $(\clubsuit_1^V)$ $R(V)(D_{\text{GR}}) = G(V)(D_{\text{GR}})$ for a $V \in \mathcal{V}$ (which is a UCQ now). It is easy to see that now $(\clubsuit_1^V)$ is equivalent to the conjunction of two disjunctive TGDs:

$(\heartsuit_{gr})$ $\bigvee_{0 \le i < k} G(\phi_i)(\bar{x}'_i, \bar{y}) \Rightarrow \bigvee_{0 \le i < k} \exists \bar{x}'_i\, R(\phi_i)(\bar{x}_i, \bar{y})$

' $(\heartsuit_{rg})$ $\bigvee_{0 \le i < k} R(\phi_i)(\bar{x}'_i, \bar{y}) \Rightarrow \bigvee_{0 \le i < k} \exists \bar{x}'_i\, G(\phi_i)(\bar{x}_i, \bar{y})$

where k is the number of disjuncts in V, the formula ϕ_i is the quantifier-free part of the i-th disjunct, and $\bar{y}$ are the free variables of (each conjunct of) V.

Now suppose the body of $(\heartsuit_{gr})$ is satisfied in some[13] $Chase_{2n}$. Then $Chase_{2n} \models G(\phi_i)(\bar{a}, \bar{b})$ for some ϕ_i and some tuple $\bar{a}, \bar{b}$ of elements of $Chase_{2n}$. An application of our disjunctive TGD will produce a tuple $\bar{a}'$ in $Chase_{2n+1}$ such that $Chase_{2n+1} \models R(\phi_{i'})(\bar{a}', \bar{b})$. But there is no reason for i' to equal i, and it is the job of the Fugitive to choose the i' which suits him best (remember, his goal is to reach the fixpoint without satisfying $R(Q)$).

Now, a lemma in [6] and [7] is that such a game indeed characterizes determinacy: the Fugitive has a winning strategy if and only if $[\mathcal{V}, Q] \notin QDP_e^\infty(UCQ, UCQ)$. The proof of the lemma goes in the footsteps of the standard proof of universality of Chase, which means that "induction" and "homomorphism" are the keywords.

5 Some applications on the positive side

5.1 Unary queries

It is proven in [11] (it is not a long proof but the argument is not so easy to understand) that $QDP_e^\infty(mCQ, CQ, CQ)$ equals to $QDP_e^\infty(mCQ, CQ)$. In other words, if a set of unary conjunctive queries determines a CQ, then there exists a rewriting, which itself is a conjunctive query. Then a corollary follows in [11] that $QDP_e^\infty(mCQ, CQ)$ is decidable.

But it seems to me that a stronger decidability result, not provable in any obvious way by the aforementioned rewriting argument, can be easily proved using the insight given by the Green-Red Chase[14]. Let $\mathcal{L}_V$ be any query language, consisting of unary queries, and such that (*) all the minimal bodies of queries are of bounded tree width (so, for example $\mathcal{L}_V$ may contain unions of unary conjunctive queries and/or Monadic DL queries). And let also $\mathcal{L}_Q$ be any[15] query language satisfying (*). Then $QDP_e^\infty(\mathcal{L}_V, \mathcal{L}_Q)$ is decidable.

Let us now explain the proof idea using the example of $QDP_e^\infty(mUCQ, CQ)$. Suppose that we have given a set $\mathcal{V}$ of mUCQs and a CQ Q. Then (and this is the sentence which summarizes the whole proof) any structure which the Fugitive can construct as his $Chase(\mathcal{T}_\mathcal{V}, G([Q]))$ is a structure having the tree width bounded by some k: indeed, the bags of the tree decomposition in question are the (red or green versions of) the frozen bodies of the conjunctive queries being the disjuncts of the queries from $\mathcal{V}$. This is because such

[13] There are many structures now which can be built by the Fugitive as his $Chase_m$, and as his final $Chase$. But – hoping this will not lead to additional confusion – we call each of them $Chase_m$ or $Chase$.

[14] While reading this subsection the Reader may notice that the idea of having, instead of two structures over Σ, a single structure over a green-red signature is of critical importance here.

[15] If it was a regular paper we should be slightly more formal here: condition (*) should also require that the set of minimal bodies of queries is *regular*. This is of course satisfied for mUCQs and for mDL.

a frozen body, when being added to $Chase$ at some point, only connects to the current structure via the elements being substituted for its free variables, which in this case is a single element[16].

Now one can construct an automaton A_1 which, for a given structure $\mathcal{A}$, of tree width bounded by k, decides whether $\mathcal{A} \models \mathcal{T}_\mathcal{V}$ and $\mathcal{A} \models G([Q])$. It will accept any structure that can be built as $Chase(\mathcal{T}_\mathcal{V}, G([Q]))$ and probably also many other structures which, while satisfying $\mathcal{T}_\mathcal{V}$ and containing $G([Q])$, cannot be produced by a Green-Red Chase procedure.

One can also construct another automaton A_2 which, on any structure with the tree width bounded by k, decides whether $R([Q])$ is contained in this structure. Then we apply the standard automata-theoretic procedure to check whether there exists a structure accepted by A_1 but not by A_2.

5.2 Slightly beyond Path Queries

It is proved in [1] that $QDP_e^\infty(PQ, PQ)$ is decidable. And the decision algorithm, while quite simple, gives a really very nice insight into PQ determinacy. In our language the algorithm can be expressed as follows.

Given a path query Q and a set $\mathcal{V}$ of path queries first construct $Chase_0$, which, as we know, will be $G([Q])$. This structure is a green path. Let s be the starting point of this path and let t be its endpoint.

Then construct $Chase_1$. This means that a set of new red paths connecting some pairs of elements of $Chase_0$ will be added. Consider now the graph being the red part of $Chase_1$ (that is all the edges that are in $Chase_1$ but not in $Chase_0$). It is not terribly hard to prove (and is left for the Reader as a rather non-trivial exercise) that $[\mathcal{V}, Q] \in QDP_e^\infty(PQ, PQ)$ if and only if s and t are in the same connected component[17] of this graph.

Notice that this really works for the $[\mathcal{V}_1, Q]$ from Section 3.1: the a and e there indeed are connected via the red edges of $Chase_1$ (one needs to make 3 steps going "forward", then two steps "backward" and then again 3 steps "forward").

Using the above "connectivity" criterion one[18] can show that:

▶ **Theorem 5.** *$QDP_e^\infty(PQ, RPQ)$ is decidable.*

For the proof of the theorem suppose Q and $\mathcal{V}$ are given. Q is a query defined as a union of some regular set $\mathcal{R}$ of path queries, and each of the queries in $\mathcal{V}$ is a path query, so there is no disjunction there. This means that the only choice the Fugitive has here is when he constructs $Chase_0$. He can take, as $Chase_0$, any green path $G([\mathbf{w}])$, from some s to t, for some $\mathbf{w} \in \mathcal{R}$

He is deemed to lose if for each such choice of $\mathbf{w}$ the vertices s and t will be in the same connected component of the red part of $Chase_1(\mathcal{T}_\mathcal{V}, G([\mathbf{w}]))$. He wins (and so $[\mathcal{V}, Q] \notin QDP_e^\infty(PQ, RPQ)$) if he can find a $\mathbf{w}$ for which s and t will be in two different connected components.

[16] As I learned from a discussion with Sebastian Rudolph, this argument holds true even for non-unary queries if all the free variables always occur in a single atom of each disjunct of V. The general picture is that what we actually do here is deciding query entailment for the theory $\mathcal{T}_\mathcal{V}$, and this is decidable for sets of TGDs which are frontier guarded. One can also notice here that for positive results for the respective $QDP_e^f(\mathcal{L}_V, \mathcal{L}_Q)$ variants known theorems regarding Finite Controllability for certain sets of TGDs can be applied.

[17] We think of an undirected connected component of a directed graph here.

[18] Theorem 5 comes from the unpublished master's thesis by my student Grzegorz Głuch.

How can we decide whether such $\mathbf{w}$ exists? First of all recall how $Chase_1(\mathcal{T}_\mathcal{V}, G([\mathbf{w}]))$ is constructed: for two vertices a, b of $G([w])$ connected with some word $G(\mathbf{v})$ (which is a sub-word of $\mathbf{w}$) a new red path from a to b, labelled with $R(\mathbf{v})$ is created in $Chase_1$ if and only if $\mathbf{v} \in \mathcal{V}$. We imagine we have a green straight path from s to t, with $\mathbf{w}$ being the sequence of the labels of its edges (this is $Chase_0$), and red arcs over this green straight path joining each two vertices which are connected with a path being some green $\mathbf{v} \in \mathcal{V}$.

It is now easy to construct a two-way non-deterministic finite automaton which will accept $\mathbf{w}$ if and only if s and t are in the same connected component of the red part of $Chase_1$: the automaton starts its run in the head in s, and then, at each stage of the run[19] it first guesses whether it should now walk down some red arc (towards the t) or up (back, towards s), and which $\mathbf{v} \in \mathcal{V}$ is the label of the arc it now takes. And then it just walks down or up $G([\mathbf{v}])$ imagining it moves down or up a red arc (and checking whether it indeed is labelled with $\mathbf{v}$). It accepts when t is reached after completing some number of stages.

Notice that in the case of of the instance $[\mathcal{V}_1, Q]$ from Section 3.1 our two-way automaton will first go towards t (the $\mathbf{v}$ will be BAC), then back towards s (the $\mathbf{v}$ will be AC), and finally it will reach t after the third stage where $\mathbf{v}$ will be ACA.

Now, this two-way nondeterministic finite automaton can be translated into a normal DFA $A_\mathcal{V}$. We also have a DFA $A_\mathcal{R}$, deciding the language $\mathcal{R}$. All we need to do to see whether the Fugitive is deemed to lose is to decide, using handbook methods, the containment of languages for $A_\mathcal{R}$ and $A_\mathcal{V}$.

▶ **Exercise 6.** *Why doesn't it prove that also $QDP_e^\infty(RPQ, RPQ)$ is decidable? Exactly the same automata trick would work in this case.*

Notice that, instead of building this two-way automaton we could just use the fact that (since the red arcs are short and local) the structure $Chase_1$ is of bounded tree width (and the bound does not depend on $\mathbf{w}$).

6 How disjunction leads to undecidability. $QDP_e^\infty(UCQ, UCQ)$.

If disjunction is available in $\mathcal{L}_Q$ and $\mathcal{L}_V$ then the Curse of Pâte Feuilletée is not in force: the Fugitive **can**, while executing the Green-Red Chase, add to the structure something that is not just a copy of $Chase_0$. For example, suppose that some query $V \in \mathcal{V}$ equals $\exists \bar{x}\ \phi(y, z, \bar{x}) \wedge \phi'(y, z, \bar{x})$, for some CQs ϕ and ϕ', and that $R(V)$ is satisfied in some $Chase_i$, because there is $Chase_i \models R(\phi)(a, b)$ for some elements a and b. Suppose however that $G(V)(a, b)$ is not satisfied in $Chase_i$. Then the Fugitive must satisfy $G(V)$ in $Chase_{i+1}$ and he can do it by adding to $Chase_{i+1}$ a new copy of $G([\phi])$, connected to $Chase_i$ via a and b ("a re-colored copy of something we already saw") or a new copy of $G([\phi'])$ (connected in the same way). So he **can** produce something new. But can we force him to? This is what the Cold/Hot trick is about, which we are going to present in this Section.

The example we are going use is $QDP_e^\infty(UCQ, UCQ)$. We will show that the problem is undecidable. The result comes from [11], but the proof we present here is based on the ideas from [6], where we employed the cold/hot trick to show undecidability of $QDP_e^\infty(RPQ, RPQ)$ (and of $QDP_e^f(RPQ, RPQ)$) and from [7] where analogous results were shown for UPQ as both $\mathcal{L}_V$ and $\mathcal{L}_Q$. Clearly, both RPQ and UPQ support disjunction, and disjunction is all we need for the Cold/Hot trick to work.

[19] The run will consist of an unbounded number of stages, each of them comprising a bounded number of steps.

6.1 The Cold/Hot Trick

Imagine that our signature $\Sigma = \Sigma_C \dot{\cup} \Sigma_H$, so it is a disjoint union of *cold* and *hot* relation symbols[20]. This means that there can possibly be four kinds of atoms in the Green-Red Chase: red-warm, red-cold, green-warm and green-cold. The idea is to construct[21] Q and $\mathcal{V}$ in such a way that if the Fugitive chooses anything green-hot or red-cold then he loses the game immediately. In other words he will need to always make sure that the green part of the Chase is entirely cold and the red part is entirely hot.

Let *Lukewarm* be the set of all conjunctive queries of the form[22] $C(x, x') \wedge H(y, y')$ where – unsurprisingly – C is cold and H is hot. Our UCQ Q is:

$$\exists x, x', y, y' \quad \alpha_C(x, y) \vee \omega_H(x, y) \quad \vee \bigvee_{\phi \in Lukewarm} \phi(x, x', y, y')$$

where α_C is a certain cold relation symbol and ω_H is a certain hot one. Our set of UCQs $\mathcal{V}$ is the disjoint union of $\mathcal{V}_{good}$ and $\mathcal{V}_{bad}$.

Each UCQ $V \in \mathcal{V}_{good}$ is of the form $V^H \vee V^C$ where V^H is a CQ being a conjunction of hot atoms and the V^C is a CQ being a conjunction of cold ones. We assume that $\alpha_C(x, y) \wedge \beta_H(x, y)$ is one of the queries in $\mathcal{V}_{good}$, for some hot β_H, and that this is the only place where α_C occurs in $\mathcal{V}_{good}$ The set of queries $\mathcal{V}_{good}$ is where the instance of some undecidable problem is going to be encoded. But we do not need to think of the details now.

At this point $\mathcal{V}_{bad}$ is more interesting, which is defined as the set containing all the queries from *Lukewarm* and the query $\omega_H(x, y)$. All the queries in $\mathcal{V}_{bad}$ are CQs.

Let, as always, $\mathcal{T}_\mathcal{V}$ be the set of all green-red disjunctive TGDs generated by $\mathcal{V}$.

Now let us analyze what the Fugitive choices for $Chase_0$ are:

▶ **Observation 7.** *The Fugitive must pick $G([\alpha_C])$ as $Chase_0$ or he will lose immediately.*

Proof. See what his other choices of minimal structures satisfying Q are. One is to pick $G([\omega_H])$. But notice that $G(\omega_H)(x, y) \Rightarrow R(\omega_H)(x, y)$ is one of the TGDs in $\mathcal{T}_\mathcal{V}$. Its body would be satisfied in $Chase_0$ so its head would need to be satisfied in $Chase_1$. So, it would be that $Chase_1 \models R(\omega_H)$ and hence $Chase_1 \models R(Q)$ and the game is over for the Fugitive.

The other choice would be to pick, as $Chase_0$, the structure $G([\phi])$ for some lukewarm query ϕ. But then he loses again, since in this case $G(\phi) \Rightarrow R(\phi)$ is one of the TGDs in $\mathcal{T}_\mathcal{V}$. ◀

Once we know that $Chase_0 = \{G(\alpha_C)(s, t)\}$ for some s, t, let the Fugitive build $Chase_1$:

▶ **Observation 8.** $Chase_1 = Chase_0 \cup \{R(\beta_H)(s, t)\}$ *or the Fugitive loses immediately.*

Proof. Recall that $\alpha_C(x, y) \wedge \beta_H(x, y)$ is one of the queries in $\mathcal{V}_{good}$. Hence the TGD:

$$G(\alpha_C)(x, y) \vee G(\beta_H)(x, y) \Rightarrow R(\alpha_C)(x, y) \vee R(\beta_H)(x, y)$$

is in $\mathcal{T}_\mathcal{V}$. So there either must be $R(\alpha_C)(s, t)$ in $Chase_1$ or $R(\beta_H)(s, t)$. But having $R(\alpha_C)(s, t)$ means that $Chase_1 \models R(Q)$ and loses the game for the Fugitive. ◀

[20] Let us also assume that all the relations in Σ are binary.

[21] This is an undecidability proof, so we construct Q and $\mathcal{V}$, depending on the instance of our favourite undecidable problem.

[22] Queries from the set *Lukewarm*, as defined here, are UCQs, but they are not UPQs (or RPQs). When implementing the idea of the Cold/Hot Trick in order to prove undecidability of $QDP_e^\infty(UPQ, UPQ)$ (or $QDP_e^\infty(RPQ, RPQ)$) one needs to invent something that would play the same role as *Lukewarm* but would also be expressible as path queries. This is easy in the *RPQ* case [6] but a bit complicated in the case of *UPQ* [7].

We are now sure that (unless the Fugitive is suicidal) there must be one green cold atom (namely, $G(\alpha_C)(s,t)$) and one red hot atom (namely, $G(\beta_H)(s,t)$) in $Chase_1$. Finally:

▶ **Observation 9.** *The Fugitive loses immediately if he ever produces a red cold atom or a green hot atom.*

Proof. Clearly, if a red cold atom was ever produced then it would satisfy, together with $R(\beta_H)(s,t)$, some $R(\phi)$ for a lukewarm query ϕ. And red lukewarm queries are forbidden (by Q) if the Fugitive wants to win[23]. Now notice that if a green hot atom was ever produced then it would satisfy, together with $G(\alpha_C)(s,t)$, some $G(\phi)$ for a lukewarm query ϕ. In this case, in the next step of Chase, $R(\phi)$ would also be satisfied. ◀

6.2 Now undecidability follows easily

To explain the remaining part of the proof let us use an example. Imagine $\mathcal{V}_{good}$ consists, apart from $\alpha_C(x,y) \wedge \beta_H(x,y)$, of the queries:

(i) $\beta_H(x,y) \vee C^{EF}_{\beta}(x,y)$
(ii) $C^{EF}_{\beta}(x,z) \vee \exists y\ E(x,y) \vee F(y,z)$
(iii) $F(x,y) \vee C^{EF}_{F}(x,y)$
(iv) $C^{EF}_{F}(x,z) \vee \exists y\ E(x,y) \vee F(y,z)$

Where E and F are hot and C^{EF}_{β} and $C^{EF}_{F}(x,y)$ are cold.

Now, let us recall that if the Fugitive wants to win, there must be $R(\beta)(s,t)$ in $Chase_1$. Then the red-green TGD generated by (i) forces the Fugitive to have, in $Chase_2$, either $G(\beta_H)(s,t)$ or $G(C^{EF}_{\beta})(s,t)$. But, by Observation 9, $G(\beta_H)(s,t)$ is forbidden, so there will be $G(C^{EF}_{\beta})(s,t)$ in $Chase_2$. Then, by analogous reasoning, there will be a new element s_1 in $Chase_3$, such that $Chase_3 \models R(E)(s,s_1), R(F)(s_1,t)$.

▶ **Exercise 10.** *There will be* $Chase_5 \models R(E)(s,s_1), R(E)(s_1,s_2), R(F)(s_2,t)$ *or the Fugitive will lose.*

In this way, using queries like (i)-(iv) we can easily encode the word problem for finitely represented semigroups: for each word $\mathbf{w}$ which is, in the given semigroup, equivalent to the word β_H we will finally get a red path in Chase, from s to t, labelled with the symbols of $\mathbf{w}$. Like in our example, where the semigroup is represented by $\beta_H = EF$ and $F = EF$ we soon forced the Fugitive to produce the path EEF. The Fugitive of course loses if at some point he is forced to produce atom ω_H. But it is a very well known undecidable problem whether, in a given finitely represented semigroup, there is any word $\mathbf{w}$ which contains ω_H and is equivalent to the word β_H.

7 Aside: determinacy under sound semantics

A variant of Query Determinacy Problem, studied in a number of papers in 1990s and early 2000s, and enjoying some new interest recently [2] is determinacy under sound semantics. It combines determinacy as formalized by Definition 1 with the observation that one never can be sure whether a queried database represents all the facts about some phenomenon,

[23] Rev. 3:16 (ESV) "So, because you are lukewarm, and neither hot nor cold, I will spit you out of my mouth."

and thus all the views should be seen as correct but potentially incomplete. The precise definition is complicated[24], and we decided not to copy it here. And we actually do not need to copy it here, because we have another one, which happens to be equivalent[25]:

▶ **Definition 11.** *$QDP_s^\infty(\mathcal{L}_V, \mathcal{L}_Q)$ is the set of such instances $[\mathcal{V}, Q]$, with $Q \in \mathcal{L}_Q$, $\mathcal{V} \subseteq \mathcal{L}_V$ that regardless of the strategy of the Fugitive it holds that $Chase_1 \models R(Q)$.*

Notice that it is almost like our characterization of QDP_e^∞ with the only difference, that for determinacy under "exact semantics" to hold, the Fugitive must be deemed to satisfy $R(Q)$ anywhere at any point of the Green-Red Chase, while for sound semantics this must happen already in $Chase_1$. This in particular means that QDP_s^∞ is a (much) stronger notion than QDP_e^∞ for the same parameters. It also follows directly from the definition that if languages $\mathcal{L}_Q$ and $\mathcal{L}_V$ have the property that each query has only finitely many "minimal structures" satisfying this query, then there are only finitely many possible structures $Chase_1$ and $QDP_s^\infty(\mathcal{L}_Q, \mathcal{L}_V)$ is trivially decidable (so, for example $QDP_s^\infty(UCQ, UCQ)$ is decidable).

▶ **Exercise 12** (CGLV02)**.** *Show that $QDP_s^\infty(RPQ, RPQ)$ is decidable.*

Hint: Like in Section 5.2 (compare to Exercise 6). But easier: a one-way finite automaton is sufficient here.

On the other hand, it is very easy to see that:

▶ **Observation 13.** *$QDP_s^\infty(CQ, DL)$ is undecidable, even if the CQs defining the views are projection-free.*

Proof. For the proof of the observation first recall that the containment of Datalog programs is undecidable. In other words it is undecidable whether, for two given programs ϕ and ψ, it holds that for each database instance D if $goal \in \phi(D)$ then $goal \in \psi(D)$. One can of course assume here that the sets of IDB predicates of ϕ and of ψ are disjoint (except of course for the arity zero predicate *goal* which is an IDB both in ϕ and in ψ). We also assume that both the programs are over some set Σ_0 of EDB predicates.

Now, let tr (like "trigger") be a new arity zero EDB predicate, and let ϕ' be a new Datalog program, which is exactly like ϕ but with the additional atom tr in the body of each rule. Which means that ϕ' behaves exactly like ϕ on the instances where tr is true, and does nothing at all on the instances where tr is false.

Let now our Q be the union of ψ and ϕ' and let our $\mathcal{V}$ contain a query $E(\bar{x})$ for each[26] predicate E in Σ_0. We claim that $[\mathcal{V}, Q] \in QDP_s^\infty(CQ, DL)$ if and only if ψ contains ϕ.

To see why the claim is true first suppose that $Chase_0$ is somehow constructed and notice that, due to the way $\mathcal{V}$ is defined, each atom of $Chase_1$ is either some $G(A)$ which was already in $Chase_0$ or $R(A)$. Or, in other words, $Chase_1$ is a union of $Chase_0$ and a red version of $Chase_0$. Except for tr: it may happen that $Chase_0 \models G(tr)$, but there is no way to have $R(tr)$ in $Chase_1$. This in particular means that no rule of $R(\phi')$ can ever be applied in $Chase_1$ and the only way to have $Chase_1 \models R(goal)$ is to prove the $R(goal)$ using $R(\psi)$.

Now let us go back one step and think of the ways in which the Fugitive can pick $Chase_0$. It is any minimal structure in which the program $G(Q)$ proves $G(goal)$, in other words it is an "unfolding" of the Datalog program $G(Q)$. There are infinitely many possible choices

[24] The notion of certain answers plays a role there.
[25] This equivalence, I understand, is proved in [2] (I didn't have the opportunity to see the paper yet). The main difficulty here is to understand the original definition of QDP_s (see for example [4]).
[26] Recall that $tr \notin \Sigma_0$.

for the Fugitive (if there is any recursion in Q). But the main choice he has, from the point of view of this argument, is whether he wants $G(Q)$ to be proven by means of the program $G(\psi)$ or $G(\phi')$.

In the first case, when $Chase_0$ is an unfolding of $G(\psi)$, it follows from the previous paragraph, that the red part of $Chase_1$ is an unfolding of $R(\psi)$, so $R(goal)$ can be proved there and the Fugitive loses. So the only way for him is to take, as $Chase_0$, an unfolding of $G(\phi')$.

Notice that in such case the red part of $Chase_1$ is any (chosen by the Fugitive) unfolding of $R(\phi)$ (not $R(\phi')$ but $R(\phi)$, as there is no $R(tr)$ in $Chase_1$!). Now, ψ contains ϕ if and only if in every such unfolding $R(goal)$ will be provable by $R(\psi)$. ◀

The variants of $QDP_s^\infty(\mathcal{L}_V, \mathcal{L}_Q)$ studied in [2] are ones with $\mathcal{L}_Q$ being Datalog programs syntactically restricted in such a way that containment is decidable, and the above very simple argument cannot be applied. One of the theorems they prove is that:

▶ **Theorem 14.** *$QDP_s^\infty(UCQ, mDL)$ is undecidable.*

The proof is by a clever application of the cold/hot trick.

On the positive side [2] shows that:

▶ **Theorem 15.** *$QDP_s^\infty(CQ, mDL)$ is decidable.*

I did not have the opportunity to see the proof from [2] so far, but I understand that it uses a tree automata argument, basically following the ideas presented in Section 5.1. Any unfolding of a monadic datalog program (and thus also every possible $Chase_0$) is a tree of bounded tree width. Then, for a Pâte Feuilletée reason nothing really new is added, only elements which were close to each other in $Chase_0$ can be close to each other in $Chase_1$, and the red part of $Chase_1$ is of bounded tree width too. And then, since the Datalog program in question is monadic, it can be decided by an automaton whether $R(goal)$ can be proven on such a bounded tree width $Chase_1$.

I am summarizing the proof of Theorem 15 in order to remark that things are a bit subtle here. For example, one could ask, why isn't $Chase_1$ a structure of bounded tree width even in the $QDP_s^\infty(UCQ, mDL)$ case? The answer is in a query:

$$(A(x) \wedge B(y)) \vee \exists z \ (E(x,z) \wedge E(y,z))$$

With such query in $\mathcal{V}$ two remote elements a, b of $Chase_0$, such that $Chase_0 \models G(A(a))$ and $Chase_0 \models G(B(b))$ can be connected[27], in $Chase_1$, via a new element e such that $Chase_1 \models R(E(a,e)), R(E(b,e))$.

Another subtlety regards the situation where we allow constants in the language. Normally, one would think, adding constants to the signature of a Datalog program should not change much: we can always replace $S(x,y,c)$ (where S is a relation symbol and c is a constant of the language) with an atom of a new predicate $S_{3=c}(x,y)$. Since there are finitely many constants, such an operation could possibly cost us in the terms of complexity of problems, but should not impact their decidability.

But imagine that for every variable x occurring somewhere in the body of any rule in Q, there is an atom $E(x,c)$ in this body. Which means that every element of $Chase_0$ will be connected, by $G(E)$, to c. It looks innocent. But imagine also that the query

[27] This works because one of the CQs in our UCQ is connected and the other is not. It seems to me that the version of $QDP_s^\infty(UCQ, mDL)$ where only such UCQs are allowed which have each of their CQs connected, is decidable, by the same proof which works for $QDP_s^\infty(CQ, mDL)$.

$\exists z\ E(x,z), E(y,z)$ is in $\mathcal{V}$. Then, by the rules of the Green-Red Chase, for each pair of elements a, b of $Chase_0$ there will be a new element s in $Chase_1$ such that $Chase_1 \models R(E)(a,s), R(E)(b,s)$.

In consequence $Chase_1$ will not have bounded tree width. Decidability of the version of $QDP^{\infty}_{s}(mDL, CQ)$ where constants are allowed in Datalog is, to the best of my knowledge, left open in [2].

8 Encoding. Spiders live here.

In Section 6 we explained how the Green-Red Chase can be used to simulate some computing device (which, in this case, was the word problem for semigroups). The mechanism we constructed there crucially required disjunction in the views from $\mathcal{V}$ (and in Q). Can anything similar be done without disjunction? In [8] and [9] Spiders are the answer.

8.1 Spiders and spider queries

Let $K \in \mathbb{N}$ be a fixed natural number. Full Spider is any structure $\mathcal{S}$ isomorphic to $\{H(a)\} \cup \{T_i(a,b_i), C_i(b_i,c_i) : 1 \leq i \leq K\} \cup \{2 \text{ more atoms to come}\}$. There can be of course Full Red Spider, which is $R(\mathcal{S})$, and Full Green Spider, $G(\mathcal{S})$.

But the workhorses of our construction are Lame Spiders. An i-Lame Red Spider $\mathcal{S}^i_R$, for $1 \leq i \leq K$, is a Full Red Spider with the atom $R(C_i)(b_i,c_i)$ replaced with $G(C_i)(b_i,c_i)$. An i-Lame Green Spider $\mathcal{S}^i_G$ is defined in an analogous way.

So a (Green or Red) Full Spider is a creature, either entirely red or entirely green, with a head (where the predicate H is) and K legs of length two. Legs are distinguishable and each of them comprises a thigh (predicate T) and a calf (predicate C). A Lame Spider, from the daltonized point of view, looks like a Full Spider, but has one calf of the opposite color.

To operate on Lame Spiders we define Spider Queries. For $1 \leq i,j \leq K$ we define the spider query $\Psi_{i,j}$ as a CQ (with variables of the form z_n as free variables):

$$\exists x, \bar{y}\quad H(x) \wedge T_i(x,z_i) \wedge T_j(x,z_j) \wedge \bigwedge_{k \neq i,j} (T_k(x,y_k) \wedge C_k(y_k,z_k) \wedge\ 2 \text{ more atoms to come}$$

So each Spider Query looks like a colorless Full Spider, but with two calves missing.

A pivotal example now. Let $K = 4, i = 2, j = 3$ and imagine there is an i-Lame Red Spider $\mathbb{S}$ somewhere[28] in $Chase_l$ (with the nodes $a, b_1, \ldots b_4, c_1 \ldots c_4$). Suppose also that $\Psi_{i,j}$ is in $\mathcal{V}$, meaning that the Red-Green TGD θ generated by $\Psi_{i,j}$ must be satisfied in Chase.

Let us convince ourselves that the body of θ matches with $\mathbb{S}$: there is an atom $R(H)(x)$ in the body of θ and there indeed is $R(H)(a)$ in $\mathbb{S}$. There is $R(T_1)(x,y_1) \wedge R(C_1)(y_1,z_1)$ in the body of θ and there indeed are $R(C_1)(a,b_1)$ and $R(C_1)(b_1,c_1)$ in $\mathbb{S}$ (and same if we took 4 instead of 1). Finally, there are $R(T_2)(x,z_2)$ and $R(T_3)(x,z_3)$ in the body of θ and $R(T_2)(a,b_2)$ and $R(T_3)(a,b_3)$ in $\mathbb{S}$.

▶ **Exercise 16** (Important in order to understand the idea). *Notice that there would be no such match if we considered the same $\mathbb{S}$ but $\Psi_{1,3}$ instead of $\Psi_{2,3}$.*

Back to our example. We have already noticed that the body of the TGD θ matches with $\mathbb{S}$. Now suppose the head of θ is not satisfied in $Chase_l$ (for this match). Let us see what will be produced when the red-green TGD generated by θ is applied. We know that new

[28] $\mathbb{S}$ is a copy, in some $Chase_l$, of $\mathcal{S}^i_R$

elements will be added in $Chase_{l+1}$ for all existentially quantified variables in θ: new a' will be produced, with $G(H)(a')$ and with $G(T_2)(a', b_2)$ and $G(T_3)(a', b_3)$. And also new b'_1 and b'_4 will be created, with $G(T_1)(a', b'_1)$, $G(C_1)(b'_1, c_1)$ and $G(T_4)(a', b'_4)$, $G(C_4)(b'_4, c_4)$.

Now see, the newly produced atoms, together with atoms $R(C_2)(b_2, c_2)$ and $G(C_1)(b_1, c_1)$ of $\mathbb{S}$ form a j-Lame Green Spider! We assumed that an i-Lame Red Spider is in $Chase_l$ and that $\Psi_{i,j}$ is in $\mathcal{V}$ and we proved that in this case there must be a j-Lame Green Spider in $Chase_{l+1}$! And of course the same holds true for the colors swapped. We proved (by example):

▶ **Observation 17** (The law of Spider Algebra). *If there is S^i_G somewhere in $Chase$, and $\Psi_{i,j}$ is in $\mathcal{V}$ then S^j_R is also in the same $Chase$ (and the same for the colors swapped).*

And we almost[29] proved:

▶ **Observation 18.** *Let $\mathcal{G} = \langle \{1, 2, \ldots K\}, E\rangle$ be an undirected graph. Let $\mathcal{V}$ consist of all the queries $\Psi_{i,j}$ such that $[i, j] \in E$. Let $1 \le k, k' \le K$. Then the following two conditions are equivalent:*

- *k and k' are in the same connected component of $\mathcal{G}$;*
- *some (red or green) k-Lame Spider is in $Chase$ if and only if some k'-Lame Spider is.*

Observation 18 shows how the mechanism of spiders (together with spider queries) can do some computing for us. But of course proving that determinacy is at least as difficult as graph reachability is not a big deal. And even if it was, the proof is not yet complete. The input/output procedures still remain to be a small problem: we need to have $G([Q])$ as $Chase_0$, which is entirely green. So where can we get our k-Lame Spider from? And we know that determinacy holds once $R([Q])$ occurs somewhere in $Chase$, but $R([Q])$ is entirely red, so it is not our k'-Lame Spider either. This small problem is solved in [8] by a minor modification of the definition of spider query. But let us skip it here.

8.2 High level view of spiders

In the proof of undecidability of $QDP^\infty_e(UCQ, UCQ)$, as presented in Section 6, we constructed our example $\mathcal{V}_{good} \subseteq \mathcal{V}$ in such a way, that (for example) whenever there were two vertices a, b in $Chase$ such that $R(EF)(a, b)$ was true in $Chase$ (since it was true in some $Chase_i$) then also $R(EEF)(a, b)$ was true in $Chase$ (since it was true in some $Chase_{i+2}$). This example was meant to convince the Reader that any instance of the word problem for finitely represented semigroups can be encoded.

If we wanted to be more precise we would probably have said that for a proof of undecidability **two tricks** are needed: **first**, we need to be able to ensure that whenever there are two vertices a, b in $Chase$ such that if $Chase \models R(AZ)(a, b)$ then also $Chase \models R(Z'A')(a, b)$ (this reflects a single operation of a Turing machine: think of Z as the machine head, in certain state). **Second**, we need to always be able to give this Turing machine more space, so we need to be able to ensure that whenever there are two vertices a, b in $Chase$ such that $Chase \models R(B)(a, b)$ then also $Chase \models R(BB)(a, b)$.

In [8] we show how to employ spiders for the two tricks. Two disjoint ideas are needed for that, and here we only have room to try to explain one of them – the first one.

It is now time to reveal what the *two more atoms* in the definition of Spider are. They are $An(a, an)$ and $Ta(ta, a)$. The elements an and ta are called the Spider's antenna and tail (a is , as it was earlier, its head). We also have respective two atoms $An(x, x_a) \wedge Ta(x_t, x)$ as the *two more atoms* in the definition of Spider Query.

[29] Clearly, the $\Leftarrow$ implication is missing.

So far nothing has changed, apart from things being slightly more complicated. Observation 17 still holds true.

But imagine now a complicated structure, full of Spiders (Lame and Full, Red and Green), possibly some of them sharing some body parts (like they do in the *Chase* in Observation 18). And imagine there are two kinds of vertices of this structure: major and minor ones. The major elements serve only as antennas and tails of some Spiders (and each tail or antenna is a major element). Minor elements are spiders' heads and the elements of the legs. Now imagine you are taking your reading glasses off and now you only see the major vertices – the tails and antennas. And you also see the Spiders between them, but only as abstract objects, without being able to notice the details. What you now get is a graph, whose vertices are the major vertices of the old structure, and whose edges are labelled with labels from the set $\{\mathcal{S}^i_R, \mathcal{S}^i_G : 1 \leq i \leq K\} \cup \{\mathcal{S}_R, \mathcal{S}_G\}$ – there is an edge labelled with $\mathcal{S}^i_R$ from a vertex a to b in the "abstract" graph if there are major vertices a and b and a Spider $\mathcal{S}^i_R$ in the original structure, with a being the tail of this Spider and b being its antenna. Nobody is going to be surprised that we will think of such Spider as of an atom $\mathcal{S}^i_R(a, b)$.

8.3 Two-spider queries

Let now $\Psi_{i,j}$ be a Spider Query, as defined in Section 8.1, with the two additional atoms that came in Section 8.2. Let us write $\Psi_{i,j}$ as $\psi_{i,j} \wedge An(x, x_a) \wedge Ta(x_t, x)$. Let also $\Psi'_{i',j'}$ be a new spider query, like $\Psi_{i',j'}$, but with a fresh set of variables – every variable that occurs in $\Psi_{i',j'}$ is primed[30] in $\Psi_{i',j'}$. Let $\Psi'_{i',j'} = \psi'_{i',j'} \wedge An(x', x'_a) \wedge Ta(x'_t, x')$.

Now define $\Phi^{i',j'}_{i,j}$ as the query[31]:

$$\exists x_t \;\; \psi_{i,j} \wedge \psi'_{i',j'} \wedge An(x, x_a) \wedge Ta(x_t, x) \wedge An(x', x'_a) \wedge Ta(x'_t, x') \wedge \; x_t = x'_a$$

The query $\Psi^{i',j'}_{i,j}$, like any other query, generates two Green-Red TGDs, call them $\heartsuit_{gr}$ and $\heartsuit_{rg}$. Let us try to understand when the body of $\heartsuit_{gr}$ is satisfied in some $Chase_l$. The no-glasses view is that there need to be three major vertices in the structure, a, $b = a'$ and b' and it must hold that $Chase_l \models \mathcal{S}^i_G(a, b)$ (suppose this is the case) or $Chase_l \models \mathcal{S}^j_G(a, b)$ and that $Chase_l \models \mathcal{S}^{i'}_G(a', b')$ (suppose this is the case) or $Chase_l \models \mathcal{S}^{j'}_G(a', b')$. Then the TGD $\heartsuit_{gr}$ will be applied, creating a new node c (matching with the existentially quantified variable $x_t = x'_a$) , and – according to the Law of Spider Algebra – new spiders/edges between the major vertices: $\mathcal{S}^j_R(a, c)$ and $\mathcal{S}^{j'}_R(c, b')$.

Notice that our $\Psi^{i',j'}_{i,j}$ (or, rather, the TGDs it generates) does exactly the "first trick" from Section 8.2.

8.4 An easy hill to climb: undecidability of $QDP^\infty_e(CQ, MDL)$

Without the second trick (as we called it in Section 8.2) we of course will not be able to present the proof here, of the result from [8], that $QDP^\infty_e(CQ, CQ)$ is undecidable. But at least we can briefly explain how one can use the "first trick" from Section 8.3 to prove:

▶ **Theorem 19.** *$QDP^\infty_e(CQ, mDL)$ is undecidable.*

[30] Of course, i' and j' are not variables. They are some fixed natural numbers, which may, or may not, be different from i and j.

[31] We use equality in this CQ, which is of course only needed to keep the notation reasonably simple.

It is shown in [5] that $QDP_e^\infty(CQ, DL)$ is undecidable. The proof is essentially what the Reader could see in Section 7, and relies on undecidability of Datalog programs containment. Notice that Theorem 19 is already beyond the reach of the technique from [5], as containment of monadic Datalog programs is decidable.

To cook a proof of Theorem 19 we will need three ingredients. First is a trivial lemma:

▶ **Lemma 20.** *The following problem is undecidable:*

Given a set of word equations[32] *of the form* $\mathcal{S}_G^i\mathcal{S}_G^{i'} = \mathcal{S}_R^j\mathcal{S}_R^{j'}$, *and three numbers*[33] $1 \le k, k', k'' \le K$. *Is it true that, for each* $m \in \mathbb{N}$, *the above equations imply that:*

$$\mathcal{S}_G^{k'}(\mathcal{S}_G^k)^m\mathcal{S}_G^{k''} = \mathcal{S}_R^{k'}(\mathcal{S}_R^k)^m\mathcal{S}_R^{k''} \qquad ?$$

Our second ingredient, which we are not going to discuss in details here, is the the input/output procedure. We can write a query, quite similar to a Spider Query, which will add[34], to the Chase, the edge $\mathcal{S}_R^k(a,b)$ everywhere where it was $\mathcal{S}_G(a,b)$ and will add the edge $\mathcal{S}_R(a,b)$ everywhere where it was $\mathcal{S}_G^k(a,b)$. And also, we can write two more similar queries, one of them will add $\mathcal{S}_R^{k'}(a,b)$ everywhere where it was $\mathcal{S}_G(a,b) \wedge \alpha(a)$, where α is a new unary predicate, and the other one will add $\mathcal{S}_R^{k''}(a,b)$ everywhere where it was $\mathcal{S}_G(a,b) \wedge \omega(b)$, with ω being another new unary predicate. This may sound cryptic, but only until you read about the third ingredient, which is our monadic Datalog program Q. Let Φ be the conjunctive query whose frozen body is the Full Spider $\mathcal{S}$, as defined at the beginning of Section 8.1, with variables x_a and x_t matching with the spider's antenna and tail. The program Q will consist of 3 rules:

$$\begin{aligned}
\Phi(x_t, x_a), \alpha(x_t) &\Rightarrow M(x_a)\\
M(x_t), \Phi(x_t, x_a) &\Rightarrow M(x_a)\\
M(x_t), \Phi(x_t, x_a), \omega(x_a) &\Rightarrow goal
\end{aligned}$$

Now imagine a Green-Red chase for Q and $T_\mathcal{V}$, which are the Green-Red TGDs representing our given set of word equations. The elements of $\mathcal{V}$ are two-spider queries, which are CQs, so there will be no room for the Fugitive for any maneuver there. But he can choose $Chase_0$, as it is going to be the green version of some unfolding of Q. Unfoldings of Q are chains of Full Spiders, with the antenna of a predecessor always being the tail of its successor, and with the first tail marked with α and last antenna marked with ω. The input/output rules will produce another chain, of Lame Red Spiders, with the same antennas and tails, and with $\mathcal{S}_R^{k'}$ as the first Spider, $\mathcal{S}_R^{k''}$ as the last one, and with $\mathcal{S}_R^k$ everywhere in between. Then, in the process of Chase, all the words equal to $\mathcal{S}_R^{k'}(\mathcal{S}_R^k)^m\mathcal{S}_R^{k''}$ (modulo the equations enforced by $T_\mathcal{V}$) will be created. Which means that if $\mathcal{S}_G^{k'}(\mathcal{S}_G^k)^m\mathcal{S}_G^{k''} = \mathcal{S}_R^{k'}(\mathcal{S}_R^k)^m\mathcal{S}_R^{k''}$ then the word $\mathcal{S}_G^{k'}(\mathcal{S}_G^k)^m\mathcal{S}_G^{k''}$ will be created in *Chase* at some stage. Then, in the next step of Chase, due to the input/output queries, a red version of an unfolding of Q will be produced, causing the Fugitive to lose the game. Of course many details remain unexplained here, including the "only if" direction, but all the important ideas have been presented here.

[32] We consider a semigroup here, whose generators are (names of) Green and Red Lame Spiders.
[33] We already met these, or at least similar, k and k', in Observation 18.
[34] More precisely, a red-green TGD generated by this query will add.

References

1 Foto N. Afrati. Determinacy and query rewriting for conjunctive queries and views. *Theoretical Computer Science*, 412(11):1005–1021, 2011. doi:10.1016/j.tcs.2010.12.031.

2 Michael Benedikt, Stanislav Kikot, Piotr Ostropolski-Nalewaja, and Miguel Romero Orth. Unpublished manuscript, 2019.

3 Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. What is View-Based Query Rewriting? In *Proceedings of the 7th International Workshop on Knowledge Representation meets Databases (KRDB 2000), Berlin, Germany, August 21, 2000*, pages 17–27, 2000. URL: http://ceur-ws.org/Vol-29/02-cdlv.ps.

4 Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. Lossless Regular Views. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '02, pages 247–258, New York, NY, USA, 2002. ACM. doi:10.1145/543613.543646.

5 Wenfei Fan, Floris Geerts, and Lixiao Zheng. View determinacy for preserving selected information in data transformations. *Inf. Syst.*, 37:1–12, 2012.

6 Grzegorz Gluch, Jerzy Marcinkowski, and Piotr Ostropolski-Nalewaja. Can One Escape Red Chains?: Regular Path Queries Determinacy is Undecidable. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, pages 492–501, New York, NY, USA, 2018. ACM. doi:10.1145/3209108.3209120.

7 Grzegorz Gluch, Jerzy Marcinkowski, and Piotr Ostropolski-Nalewaja. The First Order Truth Behind Undecidability of Regular Path Queries Determinacy. In *22nd International Conference on Database Theory, ICDT 2019, March 26-28, 2019, Lisbon, Portugal*, pages 15:1–15:18, 2019. doi:10.4230/LIPIcs.ICDT.2019.15.

8 Tomasz Gogacz and Jerzy Marcinkowski. The Hunt for a Red Spider: Conjunctive Query Determinacy Is Undecidable. In *Proceedings of the 2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, LICS '15, pages 281–292, Washington, DC, USA, 2015. IEEE Computer Society. doi:10.1109/LICS.2015.35.

9 Tomasz Gogacz and Jerzy Marcinkowski. Red Spider Meets a Rainworm: Conjunctive Query Finite Determinacy Is Undecidable. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '16, pages 121–134, New York, NY, USA, 2016. ACM. doi:10.1145/2902251.2902288.

10 Per-Åke Larson and H. Z. Yang. Computing Queries from Derived Relations. In *Proceedings of the 11th International Conference on Very Large Data Bases - Volume 11*, VLDB '85, pages 259–269. VLDB Endowment, 1985. URL: http://dl.acm.org/citation.cfm?id=1286760.1286784.

11 Alan Nash, Luc Segoufin, and Victor Vianu. Determinacy and Rewriting of Conjunctive Queries Using Views: A Progress Report. In Thomas Schwentick and Dan Suciu, editors, *Database Theory – ICDT 2007*, pages 59–73, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

12 Daniel Pasailă. Conjunctive Queries Determinacy and Rewriting. In *Proceedings of the 14th International Conference on Database Theory*, ICDT '11, pages 220–231, New York, NY, USA, 2011. ACM. doi:10.1145/1938551.1938580.

13 Luc Segoufin and Victor Vianu. Views and queries: determinacy and rewriting. In *Proceedings of the Twenty-fourth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 13-15, 2005, Baltimore, Maryland, USA*, pages 49–60, 2005. doi:10.1145/1065167.1065174.