The First Order Truth Behind Undecidability of Regular Path Queries Determinacy

Grzegorz Głuch
Institute of Computer Science, University of Wrocław, Poland

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

Piotr Ostropolski-Nalewaja
Institute of Computer Science, University of Wrocław, Poland

Abstract

In our paper [Głuch, Marcinkowski, Ostropolski-Nalewaja, LICS ACM, 2018] we have solved an old problem stated in [Calvanese, De Giacomo, Lenzerini, Vardi, SPDS ACM, 2000] showing that query determinacy is undecidable for Regular Path Queries. Here a strong generalisation of this result is shown, and – we think – a very unexpected one. We prove that no regularity is needed: determinacy remains undecidable even for finite unions of conjunctive path queries.

2012 ACM Subject Classification Information systems → Graph-based database models

Keywords and phrases database theory, query, view, determinacy, recursive path queries

Digital Object Identifier 10.4230/LIPIcs.ICDT.2019.15

Related Version `https://arxiv.org/abs/1808.07767`

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

1 Introduction

Query determinacy problem (QDP)

Imagine there is a database $\mathbb{D}$ we have no direct access to, and there are views of this $\mathbb{D}$ available to us, defined by some set of queries $\mathcal{Q} = \{Q_1, Q_2, \ldots Q_k\}$ (where the language of queries from $\mathcal{Q}$ is a parameter of the problem). And we are given another query Q_0. Will we be able, regardless of $\mathbb{D}$, to compute $Q_0(\mathbb{D})$ only using the views $Q_1(\mathbb{D}), \ldots Q_k(\mathbb{D})$? The answer depends on whether the queries in $\mathcal{Q}$ *determine*[1] query Q_0. Stating it more precisely, the **Query Determinacy Problem** is[2]:

> The instance of the problem is a set of queries $\mathcal{Q} = \{Q_1, \ldots Q_k\}$, and another query Q_0. The question is whether $\mathcal{Q}$ determines Q_0, which means that for (♣) each two structures (database instances) $\mathbb{D}_1$ and $\mathbb{D}_2$ such that $Q(\mathbb{D}_1) = Q(\mathbb{D}_2)$ for each $Q \in \mathcal{Q}$, it also holds that $Q_0(\mathbb{D}_1) = Q_0(\mathbb{D}_2)$.

QDP is seen as a very natural static analysis problem in the area of database theory. It is important for privacy (when we don't want the adversary to be able to compute the query) and for (query evaluation plans) optimisation (we don't need to access again the database as the given views already provide enough information). And, as a very natural static analysis problem, it has a 30 years long history as a research subject – the oldest paper we were able

[1] Or, using the language of [6], [5] [9] and [8], whether $\mathcal{Q}$ are *lossless* with respect to Q_0.
[2] More precisely, the problem comes in two different flavors, "finite" and "unrestricted", depending on whether the (♣) "each" ranges over finite structures only, or all structures, including infinite.

to trace, where QDP is studied, is [20], where decidability of QDP is shown for the case where Q_0 is a conjunctive query (CQ) and also the set $\mathcal{Q}$ consists of a single CQ.

But this is not a survey paper, so let us just point a reader interested in the history of QDP to Nadime Francis's thesis [14], which is a very good read indeed.

1.1 The context

As we said, this is a technical paper not a survey paper. But still, we need to introduce the reader to the technical context of our results. And, from the point of view of this introduction, there are two lines of research which are interesting: decidability problems of QDP for positive fragments of SQL (conjunctive queries and their unions) and for fragments of the language of Regular Path Queries (RPQs) – the core of most navigational graph query languages.

QDP for fragments of SQL

A lot of progress was done in this area in two past decades. The paper [21] was the first to present a negative result. QDP was shown there to be undecidable if unions of conjunctive queries are allowed in $\mathcal{Q}$ and Q_0. The proof is moderately hard, but the queries are high arity (by arity of a query we mean the number of free variables) and hardly can be seen as living anywhere close to database practice.

In [22] it was proved that determinacy is also undecidable if the elements of $\mathcal{Q}$ are conjunctive queries and Q_0 is a first order sentence (or the other way round). Another somehow related (although no longer contained in the first order/SQL paradigm) negative result is presented in [12]: determinacy is shown there to be undecidable if $\mathcal{Q}$ is a DATALOG program and Q_0 is a conjunctive query. Finally, closing the classification for the traditional relational model, it was shown in [17] and [18] that QDP is undecidable for Q_0 and the queries in $\mathcal{Q}$ being conjunctive queries. The queries in [17] and [18] are quite complicated (the Turing machine there is encoded in the arities of the queries), and again hardly resemble anything practical.

On the positive side, [22] shows that the problem is decidable for conjunctive queries if each query from $\mathcal{Q}$ has only one free variable.

Then, in [2] decidability was shown for $\mathcal{Q}$ and Q_0 being respectively a set of conjunctive path queries and a path query. (see Section 3 for the definition). This is an important result from the point of view of the current paper, and the proof in [2], while not too difficult, is very nice – it gives the impression of deep insight into the real reasons why a set of conjunctive path queries determines another conjunctive path query.

The result from [2] begs for generalisations, and indeed it was generalised in [23] to the scenario where $\mathcal{Q}$ is a set of conjunctive path queries but Q_0 is any conjunctive query.

QDP for Regular Path Queries

A natural extension of QDP to the graph database scenario is considered here. In this scenario, the underlying data is modelled as graphs, in which nodes are objects, and edge labels define relationships between those objects. Querying such graph-structured data has received much attention recently, due to numerous applications, especially for social networks.

There are many more or less expressive query languages for such databases (see [3]). The core of all of them (the SQL of graph databases) is RPQ – the language of Regular Path Queries. RPQ queries ask for all pairs of objects in the database that are connected by a specified path, where the natural choice of the path specification language, as [26]

elegantly explains, is the language of regular expressions. This idea is at least 30 years old (see for example [11], [10]) and considerable effort was put to create tools for reasoning about regular path queries, analogous to the ones we have in the traditional relational databases context. For example [1] and [4] investigate decidability of the implication problem for path constraints, which are integrity constraints used for RPQ optimisation. Containment of conjunctions of regular path queries has been proved decidable in [7] and [13], and then, in more general setting, in [19] and [24].

Naturally, query determinacy problem has also been stated, and studied, for Regular Path Queries. This line of research was initiated in [6], [5], [9] and [8], and it was in [9] where the central problem of this area – decidability of QDP for RPQ – was first stated (called there "losslessness for exact semantics").

On the positive side, the previously mentioned result of Afrati [2] can be seen as a special case, where each of the regular languages defining the queries only consists of one word (conjunctive path queries considered in [2] constitute in fact the intersection of CQ and RPQ). Another positive result is presented in [15], where "approximate determinacy" is shown to be decidable if the query Q_0 is (defined by) a single-word regular language (a conjunctive path query), and the languages defining the queries in Q_0 and $\mathcal{Q}$ are over a single-letter alphabet. See how difficult the analysis is here – despite a lot of effort (the proof of the result in [15] invokes ideas from [2] but is incomparably harder) even a subcase (for a single-word regular language) of a subcase (unary alphabet) was only understood "approximately".

On the negative side, in [16], we showed (solving the problem from [9]), that QDP is undecidable for full RPQ.

1.2 Our contribution

The main result of this paper, and – we think – quite an unexpected one, is the following strong generalisation of the main result from [16]:

▶ **Theorem 1.1.** *Answering Determinacy question for Finite Regular Path Queries is undecidable in unrestricted and finite case (for necessary definitions see Section 3).*

To be more precise, we show that the problem, both in the "finite" and the "unrestricted" versions, is undecidable.

It is, we believe, interesting to see that this negative result falls into both lines of research outlined above. Finite Regular Path Queries are of course a subset of RPQ, where star is not allowed in the regular expressions (only concatenation and plus are), but on the other hand they are also Unions of Conjunctive Path Queries, so unlike general RPQs are first order queries and they also fall into the SQL category.

Our result shows that the room for generalising the positive result from [2] is quite limited. What we however find most surprising is the discovery that it was possible to give a negative answer to the question from [9], which had been open for 15 years, without talking about RPQs at all – undecidability is already in the intersection of RPQs and (positive) SQL.

▶ Remark. [3] makes a distinction between "simple paths semantics" for Recursive Path Queries and "all paths semantics". As all the graphs we produce in this paper are acyclic (DAGs), all our results hold for both semantics.

Organization of the paper.

In short Section 3 we introduce (very few) notions and some notations we need to use. Sections 4–14 of this paper are devoted to the proof of Theorem 1.1.

In Section 4 we first follow the ideas from [16] defining the red-green signature. Then we define the game of Escape and state a crucial lemma (Lemma 4.2), asserting that this game really fully characterises determinacy for Regular Path Queries. This part follows in the footsteps of [16], but with some changes: in [16] Escape is a solitary game, and here we prefer to see it as a two-player one.

At this point we will have the tools ready for proving Theorem 1.1. In Section 5 we explain what is the undecidable problem we use for our reduction, and in Section 6 we present the reduction. In Sections 7 – 14 we use the characterisation provided by Lemma 4.2 to prove correctness of this reduction. Due to the space constraint, we try to explain the main ideas and formulate all the crucial lemmas in the main body of the paper, but the proofs of the lemmas can be found in the full paper.

2 How this paper relates to [16]

This paper builds on top of the technique developed in [16] to prove undecidability of QDP-RPQ for any languages, including infinite.

From the point of view of the high-level architecture the two papers do not differ much. In both cases, in order to prove that if some computational device rejects its input then the respective instance of QDP-RPQ (or QDP-FRPQ) is positive (there is determinacy) we use a game argument. In [16] this game is solitary. The player, called *Fugitive*, constructs a structure/graph database (a DAG, with source a and sink b). He begins the game by choosing a path $\mathbb{D}_0$ from a to b, which represents a word from some regular language $G(Q_0)$. Then, in each step he must "satisfy requests"– if there is a path from some v to w in the current structure, representing a word from some (*) regular language Q then he must add a path representing a word from another language Q' connecting these v and w. He loses when, in this process, a path from a to b from yet another language $R(Q_0)$ is created. In this paper this game is replaced by a two-player game. But this is a minor difference. There are however two reasons why the possibility of using infinite languages is crucial in [16]. Due to these reasons, while, as we said, the general architecture of the proof of the negative result in this paper is the same as in [16], the implementation of this architecture is almost completely different here.

The first reason is as follows. Because of the symmetric nature of the constraints, the language Q (in (*) above) is always almost the same as language Q' (they only have different "colors", but otherwise are equal). For this reason it is not at all clear how to force *Fugitive* to build longer and longer paths. This is a problem for us, as to be able to encode something undecidable we need to produce structures of unbounded size. One can think that paths of unbounded length translate to potentially unbounded length of Turing machine tape.

In order to solve this problem we use – in [16] – a language $G(Q_0)$. It is an infinite language and – in his initial move – *Fugitive* could choose/commit to a path of any length he wished so that the length of the path did not need to increase in the game. But now we only have finite languages, so also $G(Q_0)$ must be finite and we needed to invent something completely different.

The second reason is in $R(Q_0)$. This – one can think – is the language of "forbidden patterns" – paths from a to b that *Fugitive* must not construct. If he does, it means that he "cheats". But now again, $R(Q_0)$ is finite. So how can we use it to detect *Fugitive's* cheating on paths no longer than the longest one in $R(Q_0)$? This at first seemed to us to be an impossible task.

But it wasn't impossible. The solution to both aforementioned problems is in the complicated machinery of languages producing edges labelled with x and y.

3 Preliminaries and notations

Determination

For a set of queries $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_k\}$ and another query Q_0, we say that $\mathcal{Q}$ *determines* Q_0 if and only if: $\forall_{\mathbb{D}_1,\mathbb{D}_2}\ \mathcal{Q}(\mathbb{D}_1) = \mathcal{Q}(\mathbb{D}_2) \to Q_0(\mathbb{D}_1) = Q_0(\mathbb{D}_2)$, where $\mathcal{Q}(\mathbb{D}_1) = \mathcal{Q}(\mathbb{D}_2)$ is defined as $\forall_{Q\in\mathcal{Q}}\ Q(\mathbb{D}_1) = Q(\mathbb{D}_2)$. *Query determinacy* comes in two versions: *unrestricted* and *finite*, depending on whether we allow or disallow infinite structures $\mathbb{D}$ to be considered. When we speak about *determinacy* without specifying explicitly it's version, we assume the *unrestricted* case of the problem.

Structures

When we say "structure" we always mean a directed graph with edges labelled with letters from some signature/alphabet Σ. In other words every structure we consider is a relational structure $\mathbb{D}$ over some signature Σ consisting of binary predicate names. Letters $\mathbb{D}$, $\mathbb{M}$, $\mathbb{G}$ and $\mathbb{H}$ are used to denote structures. Ω is used for a set of structures. Every structure we consider will contain two distinguished constants a and b. For two structures $\mathbb{G}$ and $\mathbb{G}'$ over Σ, with sets of vertices V and V', a function $h : V \to V'$ is called a homomorphism if for each two vertices $\langle u, v\rangle$ connected by an edge with label $e \in \Sigma$ in $\mathbb{G}$ there is an edge connecting $\langle h(u), h(v)\rangle$, with the same label e, in $\mathbb{G}'$.

Conjunctive path queries

Given a set of binary predicate names Σ and a word $w = a_1a_2\ldots a_n$ over Σ^* we define a (conjunctive) path query $w(v_0, v_n)$ as a conjunctive query:

$$\exists_{v_1,\ldots,v_{n-1}} a_1(v_0, v_1) \wedge a_2(v_1, v_2) \wedge \ldots a_n(v_{n-1}, v_n).$$

We use the notation $w[v_0, v_n]$ to denote the canonical structure ("frozen body") of query $w(v_0, v_n)$ – the structure consisting of elements $v_0, v_1, \ldots v_n$ and atoms $a_1(v_0, v_1)$, $a_2(v_1, v_2)$, $\ldots$ $a_n(v_{n-1}, v_n)$.

Regular path queries

For a regular language Q over Σ we define a query, which is also denoted by Q, as $Q(u, v) = \exists_{w\in Q} w(u, v)$

In other words such a query Q looks for a path in the given graph labelled with any word from Q and returns the endpoints of that path. Clearly, if Q is a finite regular language (finite regular path query), then $Q(u, v)$ is a union of conjunctive queries.

We use letters Q and L to denote regular languages and $\mathcal{Q}$ and $\mathcal{L}$ to denote sets of regular languages. The notation $Q(\mathbb{D})$ has the natural meaning: $Q(\mathbb{D}) = \{\langle u, v\rangle \mid \mathbb{D} \models Q(u, v)\}$.

4 Red-Green Structures and Escape

In this section we will provide crucial tool for our proof. First we will introduce red-green structures. Such a structure will consist of two structures each with distinct colour. One can think that we take two databases and then colour one green and another red and then look at them as a whole. This notion is very useful for two coloured *Chase* technique from [17] and [18] that has evolved into Game of Escape in [16] and is also present in this paper.

4.1 Red-green signature and Regular Constraints

For a given alphabet (signature) Σ let Σ_G and Σ_R be two copies of Σ one written with "green ink" and another with "red ink". Let $\bar{\Sigma} = \Sigma_G \cup \Sigma_R$.

For any word w from Σ^* let $G(w)$ and $R(w)$ be copies of this word written in green and red respectively. For a regular language L over Σ let $G(L)$ and $R(L)$ be copies of this same regular language but over Σ_G and Σ_R respectively. Also for any structure $\mathbb{D}$ over Σ let $G(\mathbb{D})$ and $R(\mathbb{D})$ be copies of this same structure $\mathbb{D}$ but with labels of edges recolored to green and red respectively. For a pair of regular languages L over Σ and L' over Σ' we define the *Regular Constraint (RC)* $L \to L'$ as a formula $\forall_{u,v} L(u,v) \Rightarrow L'(u,v)$.

We use the notation $\mathbb{D} \models t$ to say that an RC t is satisfied in $\mathbb{D}$. Also, we write $\mathbb{D} \models T$ for a set T of RCs when for each $t \in T$ it is true that $\mathbb{D} \models t$.

For a graph $\mathbb{D}$ and an RC $t = L \to L'$ let $rq(t, \mathbb{D})$ (as "requests") be the set of all triples $\langle u, v, L \to L' \rangle$ such that $\mathbb{D} \models L(u,v)$ and $\mathbb{D} \not\models L'(u,v)$. For a set T of RCs by $rq(T, \mathbb{D})$ we mean the union of all sets $rq(t, \mathbb{D})$ such that $t \in T$. Requests are there in order to be satisfied:

Algorithm 1 Function Add used to satisfy requests.

function *Add*
arguments:
- Structure $\mathbb{D}$
- RC $L \to L'$
- pair $\langle u, v \rangle$ such that $\langle u, v, L \to L' \rangle \in rq(L \to L', \mathbb{D})$

body:

1: Take a word $w = a_0 a_1 \dots a_n$ from L' and create a new path $w[u,v] = a_0(u, s_1), a_1(s_1, s_2), \dots, a_n(s_{n-1}, v)$ where $s_1, s_2, \dots, s_{n-1}$ are **new** vertices
2: **return** $\mathbb{D} \cup w[u,v]$.

Notice that the result $Add(D, L \to L', \langle u, v \rangle)$ depends on the choice of $w \in L'$. So the procedure is non-deterministic.

For a regular language L we define $L^{\rightarrow} = G(L) \to R(L)$ and $L^{\leftarrow} = R(L) \to G(L)$. All regular constraints we are going to consider are either $L^{\rightarrow}$ or $L^{\leftarrow}$. For a regular language L we define $L^{\leftrightarrow} = \{L^{\rightarrow}, L^{\leftarrow}\}$ and for a set $\mathcal{L}$ of regular languages we define: $\mathcal{L}^{\leftrightarrow} = \bigcup_{L \in \mathcal{L}} L^{\leftrightarrow}$.

Requests of the form $\langle u, v, t \rangle$ for some RC t of the form $L^{\rightarrow}$ ($L^{\leftarrow}$) are *generated by* $G(L)$ (resp. *by* $R(L)$). Requests that are generated by $G(L)$ or $R(L)$ are said to be *generated by* L.

The following lemma is straightforward to prove and characterises determinacy in terms of regular constraints:

▶ **Lemma 4.1.** *A set $\mathcal{Q}$ of regular path queries over Σ does not determine (does not finitely determine) a regular path query Q_0, over the same alphabet, if and only if there exists a structure $\mathbb{M}$ (resp. a finite structure) and a pair of vertices $u, v \in \mathbb{M}$ such that $\mathbb{M} \models \mathcal{Q}^{\leftrightarrow}$ and $\mathbb{M} \models (G(Q_0))(u,v)$ but $\mathbb{M} \not\models (R(Q_0))(u, v-)$.*

Any structure $\mathbb{M}$, as above, will be called a *counterexample*. One can think of $\mathbb{M}$ as a pair of structures, being green and red parts of $\mathbb{M}$. Note that those two structures both agree on $\mathcal{Q}$ but don't on Q_0, thus proving that $\mathcal{Q}$ does not determine Q_0.

4.2 The game of Escape

Here we present the essential tool for our proof. One can note that the game of Escape is very simmilar to the well known *Chase* technique. This is indeed the case, as one can think about RCs as of *Tuple Generating Dependencies* (TGDs) from the *Chase*. Divergence from

standard *Chase* comes from "nondeterminism" that is inherent part of RCs (request can be satisfied by any word from a language) phenomenon not present in TGDs.

An instance Escape(Q_0, $\mathcal{Q}$) of a game called *Escape*, played by two players called *Fugitive* and *Crocodile*, is:

- A finite regular language Q_0 of *forbidden paths* over Σ.
- A set $\mathcal{Q}$ of finite regular languages over Σ,

The rules of the game are:

- First *Fugitive* picks the *initial position* of the game as $\mathbb{D}_0 = (G(w))[a, b]$ for some $w \in Q_0$.
- Suppose $\mathbb{D}_\beta$ is the current position of some play before move $\beta+1$ and let $S_\beta = rq(\mathcal{Q}^{\leftrightarrow}, \mathbb{D}_\beta)$. Then, in move $\beta + 1$, *Crocodile* picks one request $\langle u, v, t\rangle \in S_\beta$ and then *Fugitive* can move to any position of the form:

$$\mathbb{D}_{\beta+1} := Add(\mathbb{D}_\beta, t, \langle u, v\rangle)$$

- For a limit ordinal λ the position $\mathbb{D}_\lambda$ is defined as $\bigcup_{\beta<\lambda} \mathbb{D}_\beta$.
- If $rq(\mathcal{Q}^{\leftrightarrow}, \mathbb{D}_i)$ is empty then for each $j > i$ the structures $\mathbb{D}_j$ and $\mathbb{D}_i$ are equal.
- *Fugitive* loses when for a *final position* $\mathbb{D}_{\omega^2} = \bigcup_{\beta<\omega^2} \mathbb{D}_\beta$ it is true that $\mathbb{D}_{\omega^2} \models (R(Q_0))(a, b)$, otherwise he wins. Obviously if there is some $\beta < \omega^2$ such that $\mathbb{D}_\beta \models (R(Q_0))(a, b)$ then the result of the game is already known (*Fugitive* loses), but technically the game still proceeds.

Notice that we want the game to last ω^2 steps. This is not really crucial (if we were careful ω steps would be enough) but costs nothing and will simplify presentation in Section 10.

Obviously, different strategies of both players may lead to different final positions.

Now we can state the crucial Lemma, that connects the game of Escape and QDP-RPQ:

▶ **Lemma 4.2.** *For an instance of QDP-RPQ consisting of regular language Q_0 over Σ and a set of regular languages $\mathcal{Q}$ over Σ the two conditions are equivalent:*

(i) *$\mathcal{Q}$ does not determine Q_0,*

(ii) Fugitive *has a winning strategy in Escape(Q_0, $\mathcal{Q}$).*

The proof of this Lemma can be found in the full paper.

We should mention here that all the notions of Section 4 are similar to those of [16] but are not identical. The most notable difference is in the definition of the game of Escape, as it is no longer a solitary game, as it was in [16].

This makes the analysis slightly harder here, but pays off in Sections 7 – 14.

5 Source of undecidability

In this section we will define tiling problem that we will reduce to QDP-FRPQ. In order to prove undecidability for both finite and unrestricted case we will build our tiling problem upon notion of *recursively inseparable sets*.

▶ **Definition 5.1 (Recursively inseparable sets).** *Sets A and B are called recursively inseparable when each set C, called a separator, such that $A \subseteq C$ and $B \cap C = \emptyset$, is undecidable [25].*

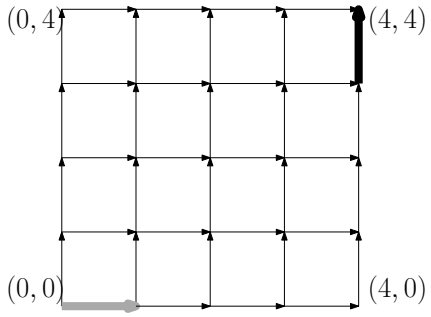

Figure 1 Finite square grid.

It is well known that:

▶ **Lemma 5.2.** *Let T be the set of all Turing Machines. Then sets $T_{acc} = \{\phi \in T | \phi(\varepsilon) = 1\}$ and $T_{rej} = \{\phi \in T | \phi(\varepsilon) = 0\}$ are recursively inseparable. By $\phi(\varepsilon)$ we mean the returned value of the Turing Machine ϕ that was run on an empty tape.*

▶ **Definition 5.3 (Square Grids).** *For a $k \in \mathbb{N}$ let $[k]$ be the set $\{i \in \mathbb{N} | 0 \leq i \leq k\}$. A square grid is a directed graph $\langle V, E\rangle$ where $V = [k] \times [k]$ for some natural $k > 0$ or $V = \mathbb{N} \times \mathbb{N}$. E is defined as $E(\langle i, j\rangle, \langle i + 1, j\rangle)$ and $E(\langle i, j\rangle, \langle i, j + 1\rangle)$ for each relevant $i, j \in \mathbb{N}$.*

▶ **Definition 5.4 (Our Grid Tiling Problem (OGTP)).** *An instance of this problem is a set of **shades** $\mathcal{S}$ having at least two elements ($\boldsymbol{gray}$,$\boldsymbol{black} \in \mathcal{S}$) and a set $\mathcal{F} \subseteq \{V, H\} \times \mathcal{S} \times \{V, H\} \times \mathcal{S}$ of forbidden pairs $\langle c, d\rangle$ where $c, d \in \{V, H\} \times \mathcal{S}$. Let the set of all these instances be called $\mathcal{I}$.*

▶ **Definition 5.5.** *A proper shading*[3] *is an assignment of shades to edges of some square grid $\mathbb{G}$ (see Figure 1) such that:*

(a1) *each horizontal edge of $\mathbb{G}$ has a label from $\{H\} \times \mathcal{S}$.*
(a2) *each vertical edge of $\mathbb{G}$ has a label from $\{V\} \times \mathcal{S}$.*
(b1) *bottom-left horizontal edge is shaded $\boldsymbol{gray}$*[4]*.*
(b2) *upper-right vertical edge (if it exists) is shaded $\boldsymbol{black}$.*
(b3) *$\boldsymbol{G}$ contains no forbidden paths of length 2 labelled by $\langle c, d\rangle \in \mathcal{F}$.*

We define two subsets of instances of OGTP:
$\boldsymbol{\mathcal{A}} = \{I \in \mathcal{I} | \text{there exists a proper shading of some finite square grid}\}$.
$\boldsymbol{\mathcal{B}} = \{I \in \mathcal{I} | \text{there is no proper shading of any square grid}\}$.

By a standard argument, using Lemma 5.2, one can show that:

▶ **Lemma 5.6.** *Sets $\mathcal{A}$ and $\mathcal{B}$ of instances of OGTP are recursively inseparable.*

In Section 6 we will construct a function $\mathfrak{R}$ ($\mathfrak{R}$ like $\mathfrak{R}$eduction) from $\mathcal{I}$ (instances of OGTP) to instances of QPD-FRPQ that will satisfy the following:

[3] We would prefer to use the term "coloring" instead, but we already have colors, red and green, and they shouldn't be confused with shades.
[4] We think of $(0, 0)$ as the bottom-left corner of a square grid. By 'right' we mean a direction of the increase of the first coordinate and by 'up' we mean a direction of increase of the second coordinate.

▶ **Lemma 5.7.** *For any instance $I = \langle \mathcal{S}, \mathcal{F} \rangle$ of OGTP and for $\langle \mathcal{Q}, Q_0 \rangle = \mathfrak{R}(I)$:*

(i) *If $I \in \mathcal{A}$ then $\mathcal{Q}$ does not finitely determine Q_0.*

(ii) *If $I \in \mathcal{B}$ then $\mathcal{Q}$ determines Q_0.*

Now the need for notion of recursive inseparability should be clear. Imagine, for the sake of contradiction, that we have an algorithm ALG deciding determinacy in either finite or unrestricted case. Then, in both cases, algorithm $ALG \circ \mathfrak{R}$ would separate $\mathcal{A}$ and $\mathcal{B}$, which contradicts the recursive inseparability of $\mathcal{A}$ and $\mathcal{B}$ (Lemma 5.6). That will be enough to prove Theorem 1.1.

6 The function $\mathfrak{R}$

Now we define a function $\mathfrak{R}$, as specified in Section 5, from the set of instances of OGTP to the set of instances of QDP-FRPQ. Suppose an instance $\langle \mathcal{S}, \mathcal{F} \rangle$ of OGTP is given. We will construct an instance $\langle \mathcal{Q}, Q_0 \rangle = \mathfrak{R}(\langle \mathcal{S}, \mathcal{F} \rangle)$ of QDP-FRPQ. The edge alphabet (signature) will be: $\Sigma = \{\alpha^C, \alpha^W, x^C, x^W, y^C, y^W, \$^C, \$^W, \omega\} \cup \Sigma_0$ where $\Sigma_0 = \{\mathbf{A}, \mathbf{B}\} \times \{H, V\} \times \{W, C\} \times \mathcal{S}$. We think of H and V as **orientations** – *Horizontal* and *Vertical*. W and C stand for *warm* and *cold*. It is worth reminding at this point that relations from $\bar{\Sigma}$ will – apart from shade, orientation and temperature – have also a **color**, red or green.

▶ **Notation 6.1.** *We will denote $(\mathbf{l}, o, t, s) \in \Sigma_0$ as $({}_s\mathbf{l}^t_o)$.*

Symbol $\bullet$ and empty space are to be understood as wildcards. This means, for example, that $({}_s\mathbf{A}_H)$ denotes the set $\{({}_s\mathbf{A}^W_H), ({}_s\mathbf{A}^C_H)\}$ and $({}_s\bullet^W_H)$ denotes $\{({}_s\mathbf{A}^W_H), ({}_s\mathbf{B}^W_H)\}$.

*Symbols from $(\bullet^W)$ and $\{\alpha^W, x^W, y^W, \$^W\}$ will be called **warm** and symbols from $(\bullet^C)$ and $\{\alpha^C, x^C, y^C, \$^C\}$ will be called **cold**.*

Now we define $\mathcal{Q}$ and Q_0. Let $\mathcal{Q}_{good}$ be a set of 15 languages:

1. ω
2. $\alpha^C + \alpha^W$
3. $x^C + x^W$
4. $y^C + y^W$
5. $\$^C + \$^W$
6. $(\mathbf{B}^C_V) + (\mathbf{B}^W_V)$
7. $(\mathbf{B}^W_H) + (\mathbf{B}^C_H)$
8. $(\mathbf{A}^W_V) + (\mathbf{A}^C_V)$
9. $(\mathbf{A}^C_H) + (\mathbf{A}^W_H)$
10. $(\mathbf{B}^W_H)(\mathbf{A}^W_V) + (\mathbf{B}^C_V)(\mathbf{A}^C_H)$
11. $(\mathbf{A}^C_H)(\mathbf{B}^C_V) + (\mathbf{A}^W_V)(\mathbf{B}^W_H)$
12. $x^C\big((\mathbf{A}^C_H)+(\mathbf{B}^C_H)+(\mathbf{A}^C_V)+(\mathbf{B}^C_V)\big)+x^C + x^W$
13. $\big((\mathbf{A}^C_H)+(\mathbf{B}^C_H)+(\mathbf{A}^C_V)+(\mathbf{B}^C_V)\big)\, y^C+y^C+y^W$
14. $x^W + x^C + x^C(\mathbf{A}^C_H)(\mathbf{B}^C_V)$
15. $y^W + \$^C + (\mathbf{A}^C_H)(\mathbf{B}^C_V)y^C + (\mathbf{B}^C_V)y^C$

Let $\mathcal{Q}_{bad}$ be a set of languages:

1. $\alpha^W x^W ({}_s\bullet^W_d)({}_{s'}\bullet^W_{d'})y^W\omega$ for each forbidden pair $\langle (d, s), (d', s') \rangle \in \mathcal{F}$.
2. $\alpha^W x^W ({}_{shade}\mathbf{B}^W_V)\$^W\omega$ for each $shade \in \mathcal{S} \setminus \{black\}$.

Finally, let $\mathcal{Q}_{ugly}$ be a set of languages: 1. $\alpha^C\Sigma^{\leq 4}(\bullet^W)\Sigma^{\leq 4}\omega$, 2. $\alpha^W\Sigma^{\leq 4}(\bullet^C)\Sigma^{\leq 4}\omega$ and 3. $\alpha^C x^C(\mathbf{B}^C_V)(\mathbf{B}^C_V)y^C\omega$. Where $\Sigma^{\leq 4}$ is language over Σ of words shorter than 5.

We write $Q^i_{good}, Q^i_{bad}, Q^i_{ugly}$ to denote the i-th language of the corresponding group. Now we can define: $\mathcal{Q} := \mathcal{Q}_{good} \cup \mathcal{Q}_{bad} \cup \mathcal{Q}_{ugly}$

The sense of the construction will (hopefully) become clear later, but we can already say something about the structure of the defined languages.

Languages from Q_{good} serve as a building blocks during the game of Escape, they are used solely to build a shaded square grid structure, that will correspond to some tiling of square grid. On the other hand the languages from Q_{bad} and Q_{ugly} serve as a "stick" during play. It will become clear later that whenever *Fugitive* has to answer a request generated by

one of languages in Q_{bad} or Q_{ugly} it is already too late for him to win. More precisely every request generated by Q_{bad} is due to *Fugitive* assigning shades to grid in a forbidden manner and leads to *Fugitive's* swift demise. Similarly requests generated by Q_{ugly} make *Fugitive* to use "right" building blocks from Q_{good}, by forcing that edge is red if and only if it is *warm* (this will be thoroughly explained in Section 8), or else he will loose. Third language from Q_{ugly} ensures that there won't be two consecutive $(\mathbf{B}_V^C)$ symbols in the structure, it is a feature used exclusively in proof of Lemma 11.1.

Finally, define $Q_{start} := \alpha^C x^C ({}_{gray}\mathbf{A}_H^C)(\mathbf{B}_V^C) y^C \omega$, and let:

$$Q_0 := Q_{start} + \bigoplus_{L \in \mathcal{Q}_{ugly}} L + \bigoplus_{L \in \mathcal{Q}_{bad}} L$$

Intuitively, we need to put the languages Q_{bad} and Q_{ugly} into Q_0 so that they can serve as aforementioned "sticks" such that whenever *Fugitive* makes a mistake he loses. Recall that *Fugitive* loses when he writes a red word from Q_0 connecting the beginning and the end of the starting structure. On the other side, this creates a problem regarding the starting structure (now *Fugitive* can pick a word from Q_{bad} or Q_{ugly} to start from instead of a word from Q_{start}), but fortunately it can be dealt with and will be resolved in Section 8.

7 Structure of the proof of Lemma 5.7

The rest of the paper will be devoted to the proof of Lemma 5.7 (restated):

▶ **Lemma 7.1.** *For any instance $I = \langle \mathcal{S}, \mathcal{F} \rangle$ of OGTP and for $\langle \mathcal{Q}, Q_0 \rangle = \mathfrak{R}(I)$:*

(i) *If $I \in \mathcal{A}$ then $\mathcal{Q}$ does not finitely determine Q_0.*

(ii) *If $I \in \mathcal{B}$ then $\mathcal{Q}$ determines Q_0.*

Proof of claim (i) – which will be presented in the end of Section 14 – will be straightforward once the reader grasps the (slightly complicated) constructions that will emerge in the proof of claim (ii).

For the proof of claim (ii) we will employ Lemma 4.2, showing that if the instance $\langle \mathcal{S}, \mathcal{F} \rangle$ has no proper shading then *Crocodile* **does have** a winning strategy in Escape($\mathcal{Q}$, Q_0) (where $\langle \mathcal{Q}, Q_0 \rangle = \mathfrak{R}(\langle \mathcal{S}, \mathcal{F} \rangle)$). As we remember from Section 4.2, in such a game *Fugitive* will first choose, as the initial position of the game, a structure $\mathbb{D}_0 = w[a, b]$ for some $w \in G(Q_0)$. Then, in each step, *Crocodile* will pick a request in the current structure (current position of the game) $\mathbb{D}$ and *Fugitive* will satisfy this request, creating a new (slightly bigger) current $\mathbb{D}$. *Fugitive* will win if he will be able to play forever (by which, formally speaking, we mean ω^2 steps, for more details see Section 4.2), or until all requests are satisfied, without satisfying (in the constructed structure) the query $(R(Q_0))(a, b)$. While talking about the strategy of *Fugitive* we will use the words "must not" and "must" as shorthands for "or otherwise he will quickly lose the game". The expression "*Fugitive* is forced to" will also have this meaning.

Analysing a two-player game (proving that certain player has a winning strategy) sounds like a complicated task: there is this (infinite) alternating tree of positions, whose structure somehow needs to be translated into a system of lemmas. In order to prune this game tree our plan is first to notice that in his strategy *Fugitive* must obey the following principles:

(I) The structure $\mathbb{D}_0$ resulting from his initial move must be $(G(w))[a, b]$ for some $w \in Q_{start}$.

(II) He must not allow any green edge with warm label and any red edge with cold label to appear in $\mathbb{D}$.

(III) He must never allow any path labelled by a word from the language $G(\mathcal{Q}_{bad}) \cup R(\mathcal{Q}_{bad})$ to occur between vertices a and b.

Then we will assume that *Fugitive's* play indeed follows the three principles and we will present a strategy for *Crocodile* which will be winning against *Fugitive*. From the point of view of *Crocodile's* operational objectives this strategy comprises three stages.

In each of these stages *Crocodile's* operational goal will be to force *Fugitive* to build some specified structure (where, of course all the specified structures will be superstructures of $\mathbb{D}_0$). In the first stage *Fugitive* will be forced to build a structure called $\mathbb{P}_1$ (defined in Section 9). In the second stage the specified structures will be called $\mathbb{P}_m$ and ${}^{\$}\mathbb{P}_m$ (each defined in Section 9) and in the third stage *Fugitive* will be forced to construct one of the structures $\mathbb{G}_m$ or $\mathbb{L}^k_m$ (defined in Section 12)

During the three stages of his play *Crocodile* will only pick requests from the languages in $\mathcal{Q}_{good}$. These languages, as we said before, are shade-insensitive, so we can imagine *Crocodile* playing in a sort of shade filtering glasses. Of course *Fugitive*, when responding to *Crocodile's* requests, will need to commit on the shades of the symbols he will use, but *Crocodile's* actions will not depend on these shades.

The shades will however play their part after the end of the third stage. Assuming that the original instance of OGTP has no proper shading, we will get that, at this moment, $R(\mathcal{Q}_{bad})(a, b)$ already holds true in the structure *Fugitive* was forced to construct. This will end the proof of (ii).

8 Principles of the Game

The rules of the game of Escape are such that *Fugitive* loses when he builds a path (from a to b) labelled with $w \in R(Q_0)$. So – when trying to encode something – one can think of words in Q_0 as of some sort of forbidden patterns. And thus one can think of Q_0 as of a tool detecting that *Fugitive* is cheating and not really building a valid computation of the computing device we encode. Having this in mind the reader can imagine why the words from languages from the sets $\mathcal{Q}_{bad}$ and $\mathcal{Q}_{ugly}$, which clearly are all about suspiciously looking patterns, are all in Q_0.

But another rule of the game is that at the beginning *Fugitive* picks his initial position $\mathbb{D}_0$ as a path (from a to b) labelled with some $w \in G(Q_0)$, so it would be nice to think of Q_0 as of initial configurations of this computing device. The fact that the same object is playing the set of forbidden patterns and, at the same time, the set of initial configurations is a problem. We solved it by having languages from $\mathcal{Q}_{ugly} \cup \mathcal{Q}_{bad}$ both in $\mathcal{Q}$ and in Q_0:

▶ **Lemma 8.1 (Principle I).** Fugitive *must choose to start the Escape game from* $\mathbb{D}_0 = G(w)[a, b]$ *for some* $w \in Q_{start}$.

Notice that, from the point of view of the shades-blind *Crocodile* the words in Q_{start} are indistinguishable and thus *Fugitive* only has one possible choice of $\mathbb{D}_0$. Proof of this Lemma can be found in the full paper.

From now on we assume that *Fugitive* obeys Principle I. This implies that for some $w \in Q_{start}$ structure $G(w)[a, b]$ is contained in all subsequent structures $\mathbb{D}$ at each step of the game.

Now we will formalise the intuition about languages from $\mathcal{Q}_{ugly}$ as forbidden patterns. We start with an observation that simplifies reasoning in the proof of Principle II.

▶ **Observation 8.2.** *For some vertices* u, v *in the current structure* $\mathbb{D}$ *if there is a green (red) edge between them then* Crocodile *can force* Fugitive *to draw a red (green) edge between* u *and* v.

Proof. It is possible due to languages $1 - 9$ in $\mathcal{Q}_{good}$. ◀

▶ **Definition 8.3.** *A P2-ready*[5] *structure* $\mathbb{D}$ *is a structure satisfying the following:*

- $\mathbb{D}_0$ *is a substructure of* $\mathbb{D}$*;*
- *there are only two edges incident to* a*:* $\langle a, a'\rangle$ *with label* $G(\alpha^C)$ *and* $\langle a, a'\rangle$ *with label* $R(\alpha^W)$*;*
- *all edges labeled with* α^C *and* α^W *are between* a *and* a'*;*
- *there are only two edges incident to* b*:* $\langle b', b\rangle$ *with label* $G(\omega)$ *and* $\langle b', b\rangle$ *with label* $R(\omega)$*;*
- *all edges labeled with* ω *are between* b' *and* b*;*
- *for each* $v \in \mathbb{D} \setminus \{a, b\}$ *there is a directed path in* $\mathbb{D}$*, of length at most* 4*, from* a' *to* v *and there is a directed path in* $\mathbb{D}$*, of length at most* 4*, from* v *to* b'*.*

▶ **Lemma 8.4 (Principle II).** *Suppose that, after* Fugitive's *move, the current structure* $\mathbb{D}$ *is a P2-ready structure. Then neither a green edge with label from* $(\bullet^W)$ *nor a red edge with label from* $(\bullet^C)$ *may appear in* $\mathbb{D}$*.*

The proof of this Lemma can be found in the full paper.

▶ **Lemma 8.5 (Principle III).** Fugitive *must not allow any path labelled with a word from* $R(\mathcal{Q}_{bad}) \cup G(\mathcal{Q}_{bad})$ *to occur in the current structure* $\mathbb{D}$ *between vertices* a *and* b*.*

The proof of this Lemma can be found in the full paper.

9 The paths $\mathbb{P}_m$ and ${}^{\$}\mathbb{P}_m$

▶ **Definition 9.1.** *(See Figure 2, please use a color printer if you can)* $\mathbb{P}_m$*, for* $m \in \mathbb{N}_+$*, is a directed graph* (V, E) *where* $V = \{a, a', b', b\} \cup \{v_i : i \in [0, 2m]\}$ *and the edges* E *are labelled with symbols from* $\Sigma \setminus \Sigma_0$ *or with symbols of the form* $(\mathbf{l}^t_o)$*, where – like before –* $\boldsymbol{l} \in \{A, B\}$*,* $o \in \{H, V\}$ *and* $t \in \{W, C\}$*. Each label has to also be either red or green. Notice that there is no* $s \in \mathcal{S}$ *here: the labels we now use are sets of symbols from* $\bar{\Sigma}$ *like in Notation 6.1: we watch the play in* Crocodile's *shade filtering glasses.*

The edges of $\mathbb{P}_m$ *are as follows:*

- *Vertex* a' *is a successor of* a *and vertex* b *is a successor of* b'*. For each* $i \in [2m]$ *the successors of* v_i *are* v_{i+1} *(if it exists) and* b' *and the predecessors of* v_i *are* v_{i-1} *(if it exists) and* a'*. From each node there are two edges to each of its successors, one red and one green, and there are no other edges.*
- *Each Cold edge (labelled with a symbol in* $(\bullet^C)$*) is green.*
- *Each Warm edge (labelled with a symbol in* $(\bullet^W)$*) is red.*
- *Each edge* $\langle v_{2i}, v_{2i+1}\rangle$ *is from* $(\mathbf{A}_H)$*.*
- *Each edge* $\langle v_{2i+1}, v_{2i+2}\rangle$ *is from* $(\mathbf{B}_V)$*.*
- *Each edge* $\langle a', v_i\rangle$ *is labelled by either* x^C *or* x^W*.*
- *Each edge* $\langle v_i, b'\rangle$ *is labelled by either* y^C *or* y^W*.*
- *Edges* $\langle a, a'\rangle$ *with label* $G(\alpha^C)$ *and* $\langle a, a'\rangle$ *with label* $R(\alpha^W)$ *are in* E*.*
- *Edges* $\langle b', b\rangle$ *with label* $G(\omega)$ *and* $\langle b', b\rangle$ *with label* $R(\omega)$ *are in* E*.*

▶ **Definition 9.2.** ${}^{\$}\mathbb{P}_m$ *for* $m \in \mathbb{N}_+$ *is* $\mathbb{P}_m$ *with two additional edges:* $\langle v_{2m}, b'\rangle \in E$*, with label* $G(\$^C)$*, and* $\langle v_{2m}, b'\rangle \in E$*, with label* $R(\$^W)$*.*

[5] Meaning "ready for Principle II".

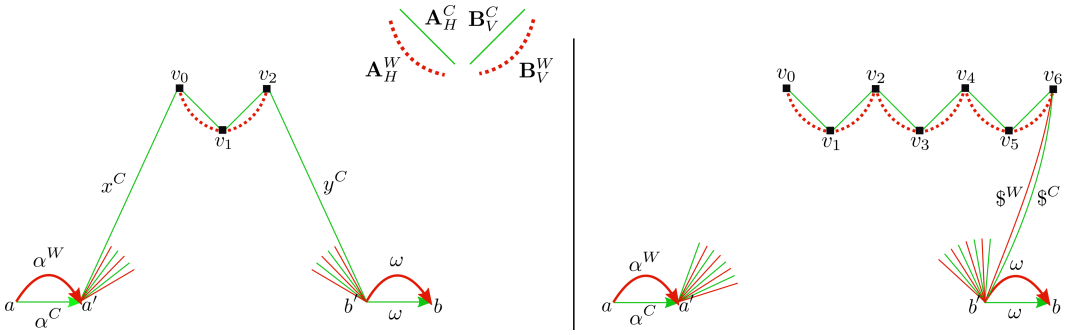

Figure 2 $\mathbb{P}_1$ (left) and ${}^{\$}\mathbb{P}_3$ (right).

One may notice[6] that $\mathbb{D}_0$ is a substructure of both $\mathbb{P}_m$ and ${}^{\$}\mathbb{P}_m$, and that:

▶ **Exercise 9.3.** *For any m, the only requests generated by $\mathcal{Q}_{good}$ in ${}^{\$}\mathbb{P}_m$ are those generated by Q^{10}_{good} and Q^{11}_{good}.*

▶ **Exercise 9.4.** *Each $\mathbb{P}_m$ and each ${}^{\$}\mathbb{P}_m$ is a P2-ready structure.*

10 Stage I

Recall that untill the end of Section 13 we watch, analyse *Fugitive's* and *Crocodile's* play in shade filtering glasses. And we assume that *Fugitive* obeys Principle I and III.

▶ **Definition 10.1 (Crocodile's strategy).** *The sequence of languages $S = (l_1, l_2, \ldots, l_n)$, for some $n \in \mathbb{N}$, defines a strategy for Crocodile as follows: If $S = (l) +\!\!+ S'$ (where $+\!\!+$ denotes sequence concatenation) then Crocodile demands Fugitive to satisfy requests generated by l one by one (in any order) until (it can take infinitely many steps) there are no more requests generated by l in the current structure. Then*[7] *Crocodile proceeds with strategy S'.*

Now we define a set of strategies for *Crocodile*. All languages that will appear in these strategies are from $\mathcal{Q}_{good}$ so instead of writing Q^i_{good} we will just write i. Let:

- $S_{color} := (3, 4, 5, 6, 7, 8, 9)$,
- $S_{cycle} := (15, 14) +\!\!+ S_{color} +\!\!+ (12, 13) +\!\!+ S_{color}$,
- $S_{start} := (1, 2) +\!\!+ S_{cycle}$.

Recall that $\mathbb{D}_0$ is *Fugitive's* initial structure (consisting of green edges only), as demanded by Principle I.

▶ **Lemma 10.2.** Crocodile's *strategy* $(1, 2)$ *applied to the current structure* $\mathbb{D}_0$ *forces* Fugitive *to add* $R(\alpha^W)[a, a']$ *and* $R(\omega)[b', b]$.

The proof of this Lemma can be found in full paper.

A careful reader could ask here: "Why did we need to work so hard to prove that the newly added red edge must be *warm*. Don't we have Principle II which says that red edges

[6] Not all anonymous reviewers equally appreciated our decision to use the "exercise" environment in this paper. In our opinion proving some simple facts themselves, rather than skipping the proofs, can help the readers to develop intuitions needed to understand what is to come in the paper. We discussed this issue with several colleagues and none of them felt that using this environment is arrogant.

[7] For this "then" to make sense we need the total number of moves of the game to be ω^2 rather than ω.

must always be *warm* and green must be *cold*?". But we cannot use Principle II here: the structure is not *P2-ready* yet. Read the proof of Principle II again to notice that this red α^W between a and a' is crucial there. And this is what Stage I is all about: it is here where *Crocodile* forces *Fugitive* to construct a structure which is *P2-ready*. From now on all the current structures will be *P2-ready* and *Fugitive* will indeed be a slave of Principle II.

The following Lemma explains the role of S_{color} and is a first cousin of Observation 8.2:

▶ **Lemma 10.3 ($\mathbf{S_{color}}$).** *Strategy S_{color} applied to a P2-ready $\mathbb{D}$ forces* Fugitive *to create a P2-ready $\mathbb{D}'$ such that:*

- *Sets of vertices of $\mathbb{D}$ and $\mathbb{D}'$ are equal.*
- *There are no requests generated by Q^{1-9}_{good} in $\mathbb{D}'$, which means that each edge has its counterpart (incident to the same vertices) of the opposite color and temperature.*

The proof of this Lemma can be found in the full paper.

▶ **Lemma 10.4.** *Strategy S_{start} applied to $\mathbb{D}_0$ forces* Fugitive *to build $\mathbb{P}_1$.*

The proof of this Lemma can be found in the full paper.

11 Stage II

Note that from now on *Fugitive* must obey all Principles.

Now we imagine that $\mathbb{P}_1$ has already been created and we proceed with the analysis to the later stage of the Escape game where either $\mathbb{P}_{m+1}$ or ${}^{\$}\mathbb{P}_k$ for some $k \leq m$ will be created.

Let us define $\{S_k\}$ inductively for $k \in \mathbb{N}_+$ in the following fashion:

- $S_1 \coloneqq S_{start}$,
- $S_k \coloneqq S_{k-1} +\!\!+ S_{cycle}$ for $k > 1$.

▶ **Lemma 11.1.** *For all $m \in \mathbb{N}_+$ strategy S_m applied to $\mathbb{D}_0$ forces* Fugitive *to build a structure isomorphic, depending on his choice, either $\mathbb{P}_{m+1}$ or ${}^{\$}\mathbb{P}_k$ for some $k \leq m$.*

The proof of this Lemma can be found in the full paper.

12 The grids $\mathbb{G}_m$ and partial grids $\mathbb{L}^k_m$

▶ **Definition 12.1.** *$\mathbb{G}_m$, for $m \in \mathbb{N}_+$, is a directed graph (V, E) where:*

$V = \{a, a', b', b\} \cup \{v_{i,j} : i, j \in [0, m]\}$ and the edges E are labelled (as in $\mathbb{P}_m$) with $\Sigma \setminus \Sigma_0$ or one of the symbols of the form $(\mathbf{l}^t_o)$, which means that the shade filtering glasses are still on.

The edges of $\mathbb{G}_m$ are as follows:

- *Vertex a' is a successor of a, b is a successor of b'. All $v_{i,j}$ are successors of a' and the successors of each $v_{i,j}$ are $v_{i+1,j}, v_{i,j+1}$ (when they exist) and b'. From each node there are two edges to each of its successors, one red and one green. There are no other edges.*
- *Each* cold *edge, labelled with a symbol in $(\bullet^C)$, is green.*
- *Each* warm *edge, labelled with a symbol in $(\bullet^W)$, is red.*
- *Each edge $\langle v_{i,j}, v_{i+1,j}\rangle$ is horizontal – its label is from $(\bullet_H)$.*
- *Each edge $\langle v_{i,j}, v_{i,j+1}\rangle$ is vertical – its label is from $(\bullet_V)$.*
- *The label of each edge leaving $v_{i,j}$, with $i + j$ even, is from $(\mathbf{A})$, the label of each edge leaving $v_{i,j}$, with $i + j$ odd, is from $(\mathbf{B})$.*
- *Each edge $\langle a', v_i\rangle$ is labeled by either x^C or x^W.*
- *Each edge $\langle v_i, b'\rangle$ is labeled by either y^C or y^W.*

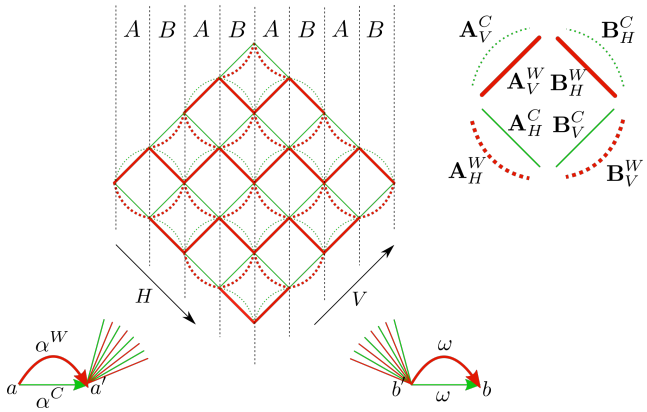

Figure 3 $\mathbb{G}_4$ (left). Smaller picture in the top-right corner explains how different line styles on the main picture map to Σ_0 (please use a color printer if you can).

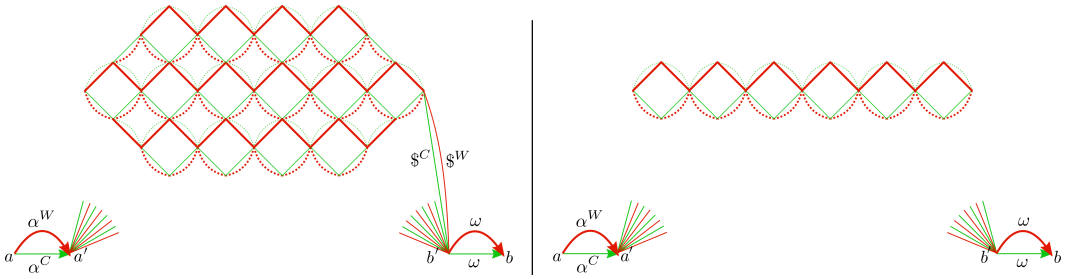

Figure 4 ${}^{\$}\mathbb{L}_6^3$ (left) and $\mathbb{L}_6^1$ (right).

- *Edges* $\langle a, a' \rangle$ *with label* $G(\alpha^C)$ *and* $\langle a, a' \rangle$ *with label* $R(\alpha^W)$ *are in* E.
- *Edges* $\langle b', b \rangle$ *with label* $G(\omega)$ *and* $\langle b', b \rangle$ *with label* $R(\omega)$ *are in* E.

▶ **Definition 12.2.** *Let* $\mathbb{L}_m^k = (V', E')$*, for* $m, k \in \mathbb{N}_+$ *where* $k \leq m$*, is a subgraph of* $\mathbb{G}_m = (V, E)$ *induced by the set* $V' \subseteq V$ *of vertices defined as* $V' = \{a, a', b', b\} \cup \{v_{i,j} : i, j \in [0, m]; \mid i - j \mid \leq k \}$.

▶ **Definition 12.3.** *Let* ${}^{\$}\mathbb{G}_m$ *for* $m \in \mathbb{N}_+$ *is* $\mathbb{G}_m$ *with two edges added:* $\langle v_{m,m}, b' \rangle$ *with label* $G(\$^C)$ *and* $\langle v_{m,m}, b' \rangle$ *with label* $R(\$^W)$.

▶ **Definition 12.4.** *Let* ${}^{\$}\mathbb{L}_m^k$ *for* $m \in \mathbb{N}_+, k \in \mathbb{N}_+ \cup \{0\}, k \leq m$ *is* $\mathbb{L}_m^k$ *with two edges added:* $\langle v_{m,m}, b' \rangle$ *with label* $G(\$^C)$ *and* $\langle v_{m,m}, b' \rangle$ *with label* $R(\$^W)$.

▷ Fact 12.5. For all m: $\mathbb{L}_m^m$ is equal to $\mathbb{G}_m$ and ${}^{\$}\mathbb{L}_m^m$ is equal to ${}^{\$}\mathbb{G}_m$.

▶ **Exercise 12.6.** *Languages from* $\mathcal{Q}_{good}$ *or* $\mathcal{Q}_{ugly}$ *do not generate requests in any* ${}^{\$}\mathbb{G}_m$.

13 Stage III

Now we imagine that either $\mathbb{P}_{m+1}$ or ${}^{\$}\mathbb{P}_k$ for some $k \leq m$ was created as the current position in a play of the game of Escape and we proceed with the analysis to the later stage of the play, where either $\mathbb{G}_{m+1}$ or ${}^{\$}\mathbb{G}_k$ will be created.

▶ **Lemma 13.1.** *For any* $m \in \mathbb{N}_+$ *Crocodile can force Fugitive to build a structure isomorphic, depending on Fugitive's choice, to either* $\mathbb{G}_{m+1}$ *or to* ${}^{\$}\mathbb{G}_k$ *for some* $k \leq m$.

Notice that by Exercise 12.5, in order to prove Lemma 13.1 it is enough to prove that for any $m \in \mathbb{N}_+$ *Crocodile* can force *Fugitive* to build a structure isomorphic to either $\mathbb{L}^{m+1}_{m+1}$ or to ${}^{\$}\mathbb{L}^k_k$ for some $k \leq m$.

As we said, we assume that *Crocodile* already forced *Fugitive* to build a structure isomorphic to either $\mathbb{P}_{m+1}$ or to ${}^{\$}\mathbb{P}_k$ for some $k \leq m$. Rename each v_i in this $\mathbb{P}_{m+1}$ (or ${}^{\$}\mathbb{P}_k$) as $v_{i,i}$. If the structure which was built is $\mathbb{P}_{m+1}$ we will show a strategy leading to $\mathbb{L}^{m+1}_{m+1}$ and when ${}^{\$}\mathbb{P}_k$ was built, we will show a strategy leading to ${}^{\$}\mathbb{L}^k_k$.

Now we define a sequence of strategies S^k_{layer}, which, similarly to strategies for building ${}^{\bullet}\mathbb{P}_{\bullet}$ consist only of languages from $\mathcal{Q}_{good}$, so instead of writing Q^i_{good} we will just write i.

Let:

- $S^{odd} \coloneqq (11) +\!\!+ S_{color} +\!\!+ (12, 13) +\!\!+ S_{color}$,
- $S^{even} \coloneqq (10) +\!\!+ S_{color} +\!\!+ (12, 13) +\!\!+ S_{color}$,
- $S^k_{layer} \coloneqq \begin{cases} [\,], & \text{if } k = 0 \\ S^{k-1}_{layer} +\!\!+ S^{odd} & \text{if } k \text{ odd} \\ S^{k-1}_{layer} +\!\!+ S^{even} & \text{otherwise} \end{cases}$

Proofs of the following four Lemmas can be found in the full paper.

▶ **Lemma 13.2.** *For all $k \in \mathbb{N}$ strategy S^1_{layer} applied to the current structure ${}^{\$}\mathbb{P}_k$ forces* Fugitive *to build ${}^{\$}\mathbb{L}^1_k$.*

▶ **Lemma 13.3.** *For all $k, m \in \mathbb{N}, k < m$ strategy S^{odd} (for $k+1$ odd) and S^{even} (for $k+1$ even) applied to ${}^{\$}\mathbb{L}^k_m$ forces* Fugitive *to build ${}^{\$}\mathbb{L}^{k+1}_m$.*

▶ **Lemma 13.4.** *For all $k, m \in \mathbb{N}, k < m$ strategy S^1_{layer} applied to $\mathbb{P}_k$ forces* Fugitive *to build $\mathbb{L}^1_k$, strategy S^{odd} (for $k+1$ odd) and S^{even} (for $k+1$ even) applied to $\mathbb{L}^k_m$ forces* Fugitive *to build $\mathbb{L}^{k+1}_m$.*

▶ **Lemma 13.5.** *For all $m \in \mathbb{N}$ strategy S^m_{layer} forces* Fugitive *to build ${}^{\$}\mathbb{L}^m_m$ from ${}^{\$}\mathbb{P}_m$ and $\mathbb{L}^m_m$ from $\mathbb{P}_m$.*

▶ **Observation 13.6.** *By Exercise 12.5 Lemma 13.5 proves Lemma 13.1.*

14 And now we finally see the shades again

Now we are ready to finish the proof of Lemma 5.7. First assume the original instance of Our Grid Tiling Problem has no *proper shading*. The following is straightforward from König's Lemma by noticing that if there were arbitrary grids with proper shading, then there would be an infinite one:

▶ **Lemma 14.1.** *If an instance I of OGTP has no proper shading then there exist natural m such that for any $k \geq m$ a square grid of size k has no shading that satisfies conditions (a1), (a2), (b1) and (b3) of proper shading.*

Let m be the value from Lemma 14.1. By Lemma 13.1 *Crocodile* can force *Fugitive* to build a structure isomorphic to either $\mathbb{G}_{m+1}$ or ${}^{\$}\mathbb{G}_k$ for some $k \leq m$. Now suppose the play ended, in some final position $\mathbb{H}$ isomorphic to one of these structures. We take off our glasses, and not only we still see this $\mathbb{H}$, but now we also see the shades, with each edge (apart from edges labeled with α, ω, x, y and \$) having one of the shades from $\mathcal{S}$. Now concentrate on the red edges labeled with $(\bullet^W)$ of $\mathbb{H}$. They form a grid, with each vertical edge labeled with V, each horizontal edge labeled with H, and with each edge labeled with a shade from $\mathcal{S}$. Now we consider two cases:

- If $\mathbb{G}_{m+1}$ was built then clearly condition (b3) of Definition 5.5 is unsatisfied. But this implies that a path labeled with a word from one of the languages Q_{bad} occurs in $\mathbb{H}$ between a and b, which is in breach with Principle III because of language Q^1_{bad}.
- If ${}^{\$}\mathbb{G}_k$ for $k \leq m$ was built then clearly condition (b2) or (b3) of Definition 5.5 is unsatisfied. This is because we assumed that there is no *proper shading*. But this implies that a path labeled with a word from one of the languages Q_{bad} occurs in $\mathbb{H}$ between a and b, which is in breach with Principle III because of language Q^1_{bad}.

This ends the proof of Lemma 5.7 (ii).

For the proof of Lemma 5.7 (i) assume the original instance $\langle \mathcal{S}, \mathcal{F} \rangle$ of Our Grid Tiling Problem has a *proper shading* – a labeled grid of side length m. Call this grid $\mathbb{G}$.

Recall that ${}^{\$}\mathbb{G}_m$ satisfies all regular constraints from $\mathcal{Q}^{\leftrightarrow}_{good}$ and from $\mathcal{Q}^{\leftrightarrow}_{ugly}$ (Exercise 12.6). Now copy the shades of the edges of $\mathbb{G}$ to the respective edges of ${}^{\$}\mathbb{G}_m$. Call this new structure (${}^{\$}\mathbb{G}_m$ with shades added) $\mathbb{M}$. It is easy to see that $\mathbb{M}$ constitutes a counterexample, as in Lemma 4.1.

References

1. Serge Abiteboul and Victor Vianu. Regular Path Queries with Constraints. In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, PODS '97, pages 122–133, New York, NY, USA, 1997. ACM. doi:10.1145/263661.263676.
2. 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.
3. Pablo Barceló. Querying graph databases. In *PODS*, 2013.
4. Peter Buneman, Wenfei Fan, and Scott Weinstein. Path Constraints in Semistructured Databases. *Journal of Computer and System Sciences*, 61(2):146–193, 2000. doi:10.1006/jcss.2000.1710.
5. D. Calvanese, G. de Giacomo, M. Lenzerini, and M. Y. Vardi. View-based query processing and constraint satisfaction. In *Proceedings Fifteenth Annual IEEE Symposium on Logic in Computer Science (Cat. No.99CB36332)*, pages 361–371, June 2000. doi:10.1109/LICS.2000.855784.
6. D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Answering regular path queries using views. In *Proceedings of 16th International Conference on Data Engineering (Cat. No.00CB37073)*, pages 389–398, February 2000. doi:10.1109/ICDE.2000.839439.
7. Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. On the Decidability of Query Containment Under Constraints. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, PODS '98, pages 149–158, New York, NY, USA, 1998. ACM. doi:10.1145/275487.275504.
8. Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. Rewriting of Regular Expressions and Regular Path Queries. In *Proceedings of the Eighteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '99, pages 194–204, New York, NY, USA, 1999. ACM. doi:10.1145/303976.303996.
9. 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.
10. Mariano P. Consens and Alberto O. Mendelzon. GraphLog: a Visual Formalism for Real Life Recursion. In *PODS*, 1990.
11. Isabel F. Cruz, Alberto O. Mendelzon, and Peter T. Wood. A Graphical Query Language Supporting Recursion. In *Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data*, SIGMOD '87, pages 323–330, New York, NY, USA, 1987. ACM. doi:10.1145/38713.38749.

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

13 Daniela Florescu, Alon Levy, and Dan Suciu. Query Containment for Conjunctive Queries with Regular Expressions. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, PODS '98, pages 139–148, New York, NY, USA, 1998. ACM. doi:10.1145/275487.275503.

14 Nadime Francis. *Vues et requetes sur les graphes de donnees: determinabilite et reecritures*. PhD thesis, Ecole Normale Superieure de Cacha, 2015.

15 Nadime Francis. Asymptotic Determinacy of Path Queries Using Union-of-Paths Views. *Theory of Computing Systems*, 61:156–190, 2016.

16 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.

17 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.

18 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.

19 V. Juge and Moshe Vardi. On the containment of Datalog in Regular Datalog. Technical report, Rice University, 2009.

20 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.

21 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.

22 Alan Nash, Luc Segoufin, and Victor Vianu. Views and Queries: Determinacy and Rewriting. *ACM Trans. Database Syst.*, 35(3):21:1–21:41, July 2010. doi:10.1145/1806907.1806913.

23 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.

24 Juan L. Reutter, Miguel Romero, and Moshe Y. Vardi. Regular Queries on Graph Databases. *Theor. Comp. Sys.*, 61(1):31–83, July 2017. doi:10.1007/s00224-016-9676-2.

25 Hartley Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. MIT Press, Cambridge, MA, USA, 1987.

26 Moshe Y. Vardi. A Theory of Regular Queries. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '16, pages 1–9, New York, NY, USA, 2016. ACM. doi:10.1145/2902251.2902305.