The First Order Truth behind
Undecidability of Regular Path Queries
Determinacy.

Grzegorz Głuch, Jerzy Marcinkowski, Piotr Ostropolski-Nalewaja
Institute of Computer Science, University of Wrocław

Abstract. In our paper [GMO18] we have solved an old problem stated in [CGLV02] showing that 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: the problem remains undecidable even for finite unions of Path Queries.

I. INTRODUCTION

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

The instance of the problem is a set of queries $Q = \{Q_1, \ldots, Q_k\}$, and another query $Q_0$. The question is whether $Q$ determines $Q_0$, which means that for (♣) each two structures (database instances) $D_1$ and $D_2$ such that $Q(D_1) = Q(D_2)$ for each $Q \in Q$, it also holds that $Q_0(D_1) = Q_0(D_2)$.

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 to trace, where QDP is studied, is [LY85], where decidability of QDP is shown for the case where $Q_0$ is a conjunctive query (CQ) and also the set $Q$ 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 thesis [F15], which is a very good read indeed.

A. The context

As we said, this is a technical paper not a survey paper. But still, we need to introduce the reader to the 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 Paths 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 last 10+ years.

The paper [NSV06] was the first to present a negative result. QDP was shown there to be
undecidable if unions of conjunctive queries are allowed in \(Q\) and \(Q_0\). The proof is moderately hard, but the queries themselves are high arity\(^3\) and hardly can be seen as living anywhere close to database practice.

In [NSV10] it was proved that determinacy is also undecidable if the elements of \(Q\) 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 [FGZ12]: determinacy is shown there to be undecidable if \(Q\) is a DATALOG program and \(Q_0\) is a conjunctive query. Finally, closing the classification for the traditional relational model, it was shown in [GM15] and [GM16] that QDP is undecidable for \(Q_0\) and the queries in \(Q\) being conjunctive queries. The queries in [GM15] and [GM16] 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, [NSV10] shows that the problem is decidable for conjunctive queries if each query from \(Q\) has only one free variable.

Then, in [A11] decidability was shown for \(Q\) and \(Q_0\) being “conjunctive path queries” (see Section III-A for the definition). This is an important result from the point of view of the current paper, and the proof in [A11], 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 [A11] begs for generalisations, and indeed it was generalised in [P11] to the the scenario where \(Q\) are conjunctive path queries but \(Q_0\) is any conjunctive query.

**QDP for Regular Path Queries.** A natural extension of QDP to 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 the social networks.

There are many more or less expressive query languages for such databases (see [B13]). 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 [V16] elegantly explains, is the language of regular expressions. This idea is at least 30 years old (see for example [CMW87, CM90]) 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 [AV97] and [BFW98] investigate decidability of the implication problem for path constraints, which are integrity constraints used for RPQ optimisation. Also, containment of conjunctions of regular path queries has been addressed and proved decidable in [CDGL98] and [FLS98], and then, in more general setting, in [JV09] and [RRV15].

Naturally, also query determinacy problem has been stated, and studied, for Regular Path Queries model. This line of research was initiated in [CGLV00], [CGLV00a], [CGLV02] and [CGLV02a], and it was [CGLV02] 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 [A11] can be seen as a special case, where each of the regular languages (defining the queries) only consists of one word (path queries, considered in [A11] constitute in fact the intersection of CQ and RPQ). Another positive result is presented in [F17], where “approximate determinacy” is shown to be decidable if the query \(Q_0\) is (defined by) a single-word regular language (a path query), and the languages defining the queries in \(Q_0\) and \(Q\) are over a single-letter alphabet. See how difficult the analysis is here – despite a lot of effort (the proof of the result in [F17] invokes ideas from [A11] but is incomparably harder) even a subcase (for a single-word regular language) of a sub-case (unary alphabet) was only understood “approximately”.

On the negative side, in [GMO18], we showed (solving the problem from [CGLV02]), that QDP is undecidable for full RPQ.

---

3By arity of a query we mean here the number of free variables.
B. Our contribution

The main result of this paper, and – we think – quite an unexpected one, is the following generalisation of the main result from [GMO18]:

**Theorem I.1.** QDP-FRPQ, the Query Determinacy Problem for Finite Regular Path Queries, is undecidable.

To be more precise, we show that the problem, both in the “finite” and the “unrestricted” version, 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 other name of Finite Regular Path Queries is Unions of Conjunctive Path Queries, so they also fall into the SQL category.

Our result shows that the room for generalising the positive result from [A11] is quite limited. And, since the queries we consider are finite unions of completely practical conjunctive queries (the lengths of the paths in our proof are all bounded by a small constant) they constitute the simplest known undecidable case in each of the two categories (positive SQL queries and RPQ queries). What we however find most surprising is the discovery that it was possible to give a negative answer to the question from [CGLV02], which had been open for 15 years, without talking about RPQs at all – undecidability is already in the intersection of RPQs and (positive) SQL.

As a positive side-result we generalise the result from [A11] showing that:

**Theorem I.2.** QDP is decidable when $Q_0$ is (defined by) an arbitrary regular language and each of $Q$ consists of a single word.

Theorem I.2 is just a slight generalization of the result (and the technique) from [A11]. But, as far as we understand, it is the first known natural decidable case of QDP for queries in RPQ not definable as (first order) conjunctive queries.

Remark. [B13] 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** Sections III–XV of this paper are devoted to the proof of Theorem I.1. In short Section III we introduce (very few) notions and some notations we need to use.

In Section IV we first follow the ideas from [GMO18] defining the red-green signature. Then we define the game of Escape and state a crucial lemma (Lemma 2), asserting that this game really fully characterises determinacy for Regular Path Queries. In Section IV-C we prove this Lemma. This part follows in the footsteps of [GMO18], but with some changes: in [GMO18] Escape is a solitary game, and here we prefer to see it as a two-players one.

At this point we will have the tools ready for proving Theorem I.1 In Section VI we explain what is the undecidable problem we use for our reduction, and present the reduction. In Sections VII–XV we use the characterisation provided by Lemma 2 to prove correctness of this reduction. Proof of Theorem I.2 can be found in Appendix 1.

II. How this paper relates to [GMO18]

This paper builds on the top of the technique developed in [GMO18] 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 [GMO18] 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 $D_0$ 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-players game. But this is a minor difference. There are however two reasons why the possibility of using infinite languages is crucial in [GMO18]. 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 [GMO18] 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 built 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 [GMO18] – 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 but no longer paths could occur later in the game. Since now we only have finite languages, so also $G(Q_0)$ must be finite, we needed here to invent something completely different. This long path is now generated step by step (see Sections XI-XII) using (mainly) the machinery of regular languages and – in his initial move – Fugitive’s cheating on paths no longer than $0$. This at first seemed to us to be an impossible task. And the solution is in a complicated machinery of languages producing edges labelled with $x$ and $y$ (languages $Q_{\text{ugly}}^{12} - Q_{\text{ugly}}^{15}$ producing $x$ and $y$ and all languages in $Q_{\text{ugly}}$ checking constraints).

### III. Preliminaries and notations

**Structures.** When we say “structure” we always mean a directed graph with edges labelled with letters from some signature/alphabet $\Sigma$. In other words every structure we consider is relational structure $D$ over some signature $\Sigma$ consisting of binary predicate names. Letters $D, M, G$ and $H$ are used to denote structures. $\Omega$ is used for a set of structures. Each structure we consider will contain two distinguished constants $a$ and $b$.

For two structures $G$ and $G'$ over $\Sigma$, with sets of vertices $V$ and $V'$, a function $h : V \rightarrow V'$ is (as always) called a homomorphism if for each two vertices $(x, y)$ connected by an edge with label $E \in \Sigma$ in $G$ there is an edge connecting $(h(x), h(y))$, with the same label $E$, in $G'$.

#### A. Path queries.

Given a set of binary predicate names $\Sigma$ and a word $w = a_1a_2 \ldots a_n$ over $\Sigma^*$ we define a path query $w(x_0, x_n)$ as a conjunctive query:

$$\exists x_1, \ldots, x_{n-1}, a_1(x_0, x_1) \land a_2(x_1, x_2) \land \ldots a_n(x_{n-1}, x_n).$$

We use the notation $w[x_0, x_n]$ to denote the canonical structure (“frozen body”) of query $w(x_0, x_n)$ – the structure consisting of elements $x_0, x_1, \ldots x_n$ and atoms $a_1(x_0, x_1), a_2(x_1, x_2), \ldots a_n(x_{n-1}, x_n)$.

**Regular path queries.** For a regular language $Q$ over $\Sigma$ we define a query, which is also denoted by $Q$, as:

$$Q(x, y) = \exists w \in Q w(x, y)$$

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, then $Q(x, y)$ is a union of conjunctive queries.

We use letters $Q$ and $L$ to denote regular languages and $Q$ and $L$ to denote sets of regular languages. The notation $Q(D)$ has the natural meaning: $Q(D) = \{ (x, y) | D \models Q(x, y) \}$.

### IV. Red-Green Structures and Escape

#### A. Red-green signature and Regular Constraints

For a given alphabet (signature) $\Sigma$ let $\Sigma_G$ and $\Sigma_R$ be two copies of $\Sigma$ one written with “green ink” and another with “red ink”. Let $\Sigma = \Sigma_G \cup \Sigma_R$. 
For any word $w$ from $\Sigma^*$ let $G(w)$ and $R(w)$ be copies of this word written in green and red respectively. For a regular language $L$ over $\Sigma$ let $G(L)$ and $R(L)$ be copies of this same regular language but over $\Sigma_G$ and $\Sigma_R$ respectively. Also for any structure $D$ over $\Sigma$ let $G(D)$ and $R(D)$ be copies of this same structure $D$ but with labels of edges recolored to green and red respectively.

For a pair of regular languages $L$ over $\Sigma$ and $L'$ over $\Sigma'$ we define $\textit{Regular Constraint} L \rightarrow L'$ as a formula

$$\forall x,y L(x,y) \Rightarrow L'(x,y).$$

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

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

$$\mathcal{L}^{\leftrightarrow} = \bigcup_{L \in \mathcal{L}} L^{\leftrightarrow}.$$

Requests of the form $(x,y,t)$ for some RC $t \in L^{\leftrightarrow}$ (resp. by $R(L)$) are generated by $G(L)$ (resp. by $R(L)$). Requests $G(L)$ and $R(L)$ jointly are said to be generated by $L$.

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

**Lemma 1.** A set $\mathcal{Q}$ of regular path queries over $\Sigma$ 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 $M$ (resp. a finite structure) and a pair of vertices $a,b \in M$ such that $M \models \mathcal{Q}^{\leftrightarrow}$ and $M \not\models (G(Q_0))(a,b)$ but $M \not\models (R(Q_0))(a,b)$.

Any structure $M$, as above, will be called counterexample.

**B. The game of Escape**

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

- a finite regular language $Q_0$ of forbidden paths over $\Sigma$.
- a set $\mathcal{Q}$ of finite regular languages over $\Sigma$.

The rules of the game are:

1. First Fugitive picks the initial position of the game as $D_0 = (G(w))(a,b)$ for some $w \in Q_0$.
2. Suppose $D_\beta$ is the current position of some play before move $\beta + 1$ and $S_\beta = \text{rq}(Q^{\leftrightarrow}, D_\beta)$. Then, in move $\beta + 1$, Crocodile picks one request $(x,y,t) \in S_\beta$ and then Fugitive can move to any position of the form:

$$D_{\beta+1} \in \text{Add}(D_\beta, t, (x,y)).$$

- For a limit ordinal $\lambda$ the position $D_\lambda$ is defined as $\bigcup_{\beta < \lambda} D_\beta$.
- If $\text{rq}(Q^{\leftrightarrow}, D_1)$ is empty then for each $j > i$ structures $D_j$ and $D_i$ are equal.
- Fugitive loses when for a final position $D_{\omega^2} = \bigcup_{\beta < \omega^2} D_\beta$ it is true that $D_{\omega^2} \models (R(Q_0))(a,b)$.
- Otherwise he wins.
Notice that we want the game to last $\omega^2$ steps. This is not really crucial (if we were careful enough $\omega$ steps would be enough) but costs nothing and will simplify presentation in Section XIV.

Obviously, different strategies of both players may lead to different final positions. We will denote the set of all final positions reachable (by any sequence of moves of both players) from an initial position $D_0$, for a set of regular languages $\mathcal{L}$, as $\Omega(\mathcal{L}^{\omega}, D_0)$.

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

**Lemma 2.** For an instance of QDP-RPQ consisting of regular language $Q_0$ over $\Sigma$ and a set of regular languages $Q$ over $\Sigma$, the two conditions are equivalent:

(i) $Q$ does not determine $Q_0$.

(ii) Fugitive has a winning strategy in Escape($Q_0$, $Q$).

**C. Universality of Escape. Proof of Lemma 2**

It is clear that (i) $\iff$ (ii) is true. All we need is to use the final position of a play won by Fugitive as the counterexample for determinacy as in Lemma 1. But the other direction is not at all obvious. Notice that it could a priori happen that, while some counterexample exists, is is some terribly complicated structure which Fugitive can not force Crocodile to reach as a final position in a play of the game of Escape.

We should mention here that all the notions of Section XIV are similar to those of GMO18 but are not identical. Most notable difference is in the definition of the game of Escape, as it is no longer a solitary game, as it was in GMO18.

This makes the analysis slightly harder here, but pays off in Sections VII, XV.

**Lemma 3.** Suppose structures $D_0$ and $M$ over $\Sigma$ are such that there exists a homomorphism $h_0 : D_0 \to M$. Let $T$ be a set of RCs and suppose $M \models T$. Then (regardless of the Crocodile’s moves) Fugitive can reach some final position $D_{\omega^2} \in \Omega(T, D_0)$ such that there exists a homomorphism $h$ from $D_{\omega^2}$ to $M$.

**Proof.** Next lemma provides the induction step for the proof of Lemma 5.

Let us define step as arity four relation such that $\langle D, D', T, r \rangle \in \text{step}$ when $D'$ can be the result of one move of Fugitive, in position $D$, in the game of Escape with set of RCs $T$ and a particular request $r \in rq(T, D)$ picked by Crocodile.

**Lemma 4.** Let $D_\beta$, $M$ be structures over $\tilde{\Sigma}$ and $h_\beta : D_\beta \to M$ be a homomorphism. Suppose that for a set $T$ of RCs it is true that $M \models T$. Then for every $r \in T$ there exists some structure $D_{\beta+1}$ such that $\text{step}(D_\beta, D_{\beta+1}, T, r)$ and that there exists homomorphism $h_{\beta+1} : D_{\beta+1} \to M$ such that $h_\beta \subseteq h_{\beta+1}$.

**Proof.** Let $r = \langle x, y, X \to Y \rangle$ for some $x, y \in D_\beta$ and let $x' = h_\beta(x)$ and $y' = h_\beta(y)$. Since $D_\beta \models X(x, y)$ and since $h_\beta$ is a homomorphism we know that $M \models X(x', y')$. But $M \models T$ so there is also $M \models Y(x', y')$ and thus for some $a_0, a_1, \ldots, a_n \in Y$ there is path $p' = a_1(x', x_1')$, $a_2(x_1', x_2') \ldots a_n(x_{n-1}', y')$ in $M$. Let $D_\beta'$ be a structure created by adding to $D_\beta$ new path $p = a_1(x, x_1)$, $a_2(x_1, x_2), \ldots a_n(x_{n-1}, y)$ (with $x_i$ being new vertices). Let $h_\beta' = h_\beta \cup \{ \langle x_i, x_{i+1} \rangle | i \in [n-1] \}$. It is easy to see that $D_\beta'$ and $h_\beta'$ are requested $D_{\beta+1}$ and $h_{\beta+1}$.

Now we consider the limit case. Let $\lambda$ be a limit ordinal such that $\lambda \leq \omega^2$. By definition we know that $D_\lambda = \bigcup_{\beta<\lambda} D_\beta$. Now we need to construct a homomorphism $h_\lambda$. Let $h_\lambda := \bigcup_{\beta<\lambda} h_\beta$. Observe that such $h_\lambda$ is a valid homomorphism from $D_\lambda$ to $M$.

This along with Lemma 4 proves that $D_{\omega^2}$ and $h_{\omega^2}$ are as required by Lemma 5.

Now we will prove the (i)$\iff$(ii) part of Lemma 2.

Assume (i). Let $M$ be a counterexample as in Lemma 1. Let $a$, $b$ and $w \in Q_0$ be such that $M \models (G(w))(a, b)$ and $M \not\models (R(Q_0))(a, b)$. Applying Lemma 3 to $D_0 = G(w)[a, b]$ and to $M$ we know that Fugitive (regardless of the Crocodile’s moves) can reach some winning final position $D_{\omega^2}$ such that there is homomorphism from $D_{\omega^2}$ to $M$. It is clear that $D_{\omega^2} \not\models (R(Q_0))(a, b)$ as we know that $M \not\models (R(Q_0))(a, b)$. This shows that $D_{\omega^2}$ is indeed a winning final position.

This concludes the proof of the Lemma 2.
V. SOURCE OF UNDECIDABILITY

Definition 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.

It is well known that:

Lemma 2. Let $T$ be the set of all Turing Machines. Then sets $T_a = \{ \phi \in T : \phi(0) = 1 \}$ and $T_r = \{ \phi \in T : \phi(0) = 0 \}$ are recursively inseparable. By $\phi(0)$ we mean the returned value of the Turing Machine $\phi$ that was run on an empty tape.

Definition 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 4 (Our Grid Tiling Problem (OGTP)). An instance of this problem is a set of shades $\mathcal{S}$ (gray, black) in $\mathcal{S}$ and a list $\mathcal{F} \subseteq \{V, H\} \times \mathcal{S} \times \{V, H\} \times \mathcal{S}$ of forbidden pairs $(c, d)$ where $c,d \in \{V, H\} \times \mathcal{S}$. Let the set of all these instances be called $\mathcal{I}$.

Definition 5. A proper shading\textsuperscript{4} is an assignment of shades to edges of some square grid $\mathbb{G}$ (see Figure 7) 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 gray\textsuperscript{5}.

(b2) upper-right vertical edge (if exists) is shaded black.

(b3) $\mathbb{G}$ contains no forbidden paths of length 2 labelled by $(c,d) \in \mathcal{F}$.

We define two subsets of instances of OGTP:

$\mathcal{A} = \{ I \in \mathcal{I} | \text{there exists a proper shading of some finite square grid} \}$.

\textsuperscript{4}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.

\textsuperscript{5}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.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{finite_square_grid.png}
\caption{Finite square grid.}
\end{figure}

B = \{ $I \in \mathcal{I}$ | there is no proper shading of any square grid \}.

By standard argument, using Lemma $2$ one can show that:

Lemma 6. Sets $\mathcal{A}$ and $\mathcal{B}$ of instances of OGTP are recursively inseparable.

In Section VI we will construct a function $\mathcal{R}$ (or \textit{Reduction}) from $\mathcal{I}$ (instances of OGTP) to instances of QPD-FRPQ that will satisfy the following:

Lemma 7. For any instance $I = \langle \mathcal{S}, \mathcal{F} \rangle$ of OGTP and for $\langle Q, Q_0 \rangle = \mathcal{R}(I)$:

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

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

That will be enough to prove Theorem $1$. Imagine, for the sake of contradiction, that we have an algorithm $\text{ALG}$ deciding determinacy (in either finite or unrestricted case). Then, in both cases, algorithm $\text{ALG} \circ \mathcal{R}$ would separate $\mathcal{A}$ and $\mathcal{B}$, which contradicts recursive inseparability of $\mathcal{A}$ and $\mathcal{B}$ (Lemma $6$).

VI. THE FUNCTION $\mathcal{R}$

Now we define a function $\mathcal{R}$, as specified in Section V, from OGTP to the QDP-RPQ. Suppose an instance $\langle \mathcal{S}, \mathcal{F} \rangle$ of OGTP is given. We will
will be called notes. This means, for example, that

\[
\begin{align*}
&\text{15 languages:} \\
&\alpha^C + \alpha^W \\
&x^C + x^W \\
&y^C + y^W \\
&\Sigma_0 = \{A, B\} \times \{H, V\} \times \{W, C\} \times \{\mathcal{S}, \mathcal{R}\}
\end{align*}
\]

where \( \Sigma_0 \) is the set of languages. We think of \( H \) and \( V \) as orientations – \( H \) for horizontal and \( V \) for vertical. \( W \) and \( C \) stand for warm and cold. It is worth reminding at this point that relations from \( \Sigma \) will – apart from shade, orientation and temperature – have also a color, red or green.

**Notation VI.1.** We use the following notation for elements of \( \Sigma_0 \):

\[ (p, q, r, s) \in \Sigma_0 \]

Symbol \( \bullet \) and empty space are to be understood as wildcards. This means, for example, that \( \{A_H^W, A_H^C\} \) denotes the set \( \{A_H^W, A_H^C\} \) and \( \{A_H^W, B_H^W\} \) denotes \( \{A_H^W, B_H^W\} \).

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

Now we define \( \mathcal{Q} \) and \( \mathcal{Q}_0 \). Let \( \mathcal{Q}_{\text{good}} \) be a set of 15 languages:

1. \( \omega \)
2. \( \alpha^C + \alpha^W \)
3. \( x^C + x^W \)
4. \( y^C + y^W \)
5. \( s^C + s^W \)
6. \( (B^C_H + (B^W_H) \} \) and \( (A^W_H) \}
7. \( (B^W_H) + (B^C_H) \}
8. \( (A^C_H) \}
9. \( (A^W_H) + (A^C_H) \}
10. \( (B^W_H) + (B^C_H) \}
11. \( (A^W_H) + (A^C_H) \}
12. \( x^C (A^C_H) + (B^C_H) + (A^C_H) + (B^C_H) + x^C + \)
13. \( x^W (A^C_H) + (B^C_H) + (A^C_H) + (B^C_H) + y^C + y^C + \)
14. \( y^W + s^C + (A^W_H) (B^C_H) y^C \)
15. \( y^W + s^C + (A^W_H) (B^C_H) y^C \)

Let \( \mathcal{Q}_{\text{bad}} \) be a set of languages:

1. \( \alpha^W x^W (\bullet^W) y^W \omega \) for each forbidden pair
2. \( \alpha^W x^W (\bullet^W) y^W \omega \) for each shade \( S \setminus \{\text{black}\} \)

Finally, let \( \mathcal{Q}_{\text{ugly}} \) be a set of languages:

1. \( \alpha^C \Sigma_0^4 (\bullet^W) \Sigma_0^4 \omega \)
2. \( \alpha^W \Sigma_0^4 (\bullet^C) \Sigma_0^4 \omega \)

where \( \Sigma_0^4 = \bigcup_{i=1}^4 \Sigma^i \).

We write \( \mathcal{Q}_{\text{good}}, \mathcal{Q}_{\text{bad}}, \mathcal{Q}_{\text{ugly}} \) to denote the i-th language of the corresponding group. Now we can define

\[ \mathcal{Q} := \mathcal{Q}_{\text{good}} \cup \mathcal{Q}_{\text{bad}} \cup \mathcal{Q}_{\text{ugly}} \]

The sense of the construction will (hopefully) become clear later. The regular constraints \( \mathcal{Q}_{\text{good}}^{10} \) and \( \mathcal{Q}_{\text{ugly}}^{10} \) are of the form “for vertices \( x, y, z \) and edges \( e_1(x, y) \) and \( e_2(y, z) \) of some color in the current structure, create a new \( y' \) and add edges \( e_1'(x, y') \) and \( e_2'(y', z) \) of the opposite color” where the pair \( (e_1, e_2) \) comes from some small finite set of possible choices.

On the other hand, each language in \( \mathcal{Q}_{\text{bad}} \cup \mathcal{Q}_{\text{ugly}} \) contains words with some bad or ugly pattern. For \( L \in \mathcal{Q}_{\text{bad}} \cup \mathcal{Q}_{\text{ugly}} \) requests generated by \( L \) are of the form “if you have a short path in the current structure, green or red, between some vertices \( x \) and \( y \), containing such pattern, then add a new path from \( x, y \) to the opposite color, also containing the same pattern”.

A small difference between languages in \( \mathcal{Q}_{\text{bad}} \) and in \( \mathcal{Q}_{\text{ugly}} \) is that languages in \( \mathcal{Q}_{\text{ugly}} \) do not depend on the constraints from the instance of Our Grid Tiling Problem while ones in \( \mathcal{Q}_{\text{bad}} \) encode this instance. One important difference between languages in \( \mathcal{Q}_{\text{good}} \cup \mathcal{Q}_{\text{ugly}} \) and \( \mathcal{Q}_{\text{bad}} \) is that only the last do mention shades.

Finally, define \( \mathcal{Q}_{\text{start}} := \alpha^C x^C (\text{gray} A^C_H) (B^C_H) y^C \omega \), and let:

\[ \mathcal{Q}_0 := \mathcal{Q}_{\text{start}} + \bigoplus_{L \in \mathcal{Q}_{\text{ugly}}} L + \bigoplus_{L \in \mathcal{Q}_{\text{bad}}} L \]

\( \mathcal{Q}_{\text{start}} \) may look like a single word language, but it is not: do not forget that \( \{B^C_H\} \) is a set of symbols (which however all look almost the same, the only difference is the shades).

**VII. The Structure of the Proof of Lemma VII.**

The rest of the paper will be devoted to the proof of Lemma VII (restated here for convenience):
Lemma 7. For any instance $I = \langle S, F \rangle$ of OGTP and for $\langle Q, Q_0 \rangle = R(I)$:

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

(ii) If $I \in B$ then $Q$ determines $Q_0$.

Proof of claim (i) — which will be presented in the end of Section $\S$V$\S$ — 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 2 showing that if the instance $\langle S, F \rangle$ has no proper shading then the Crocodile does have a winning strategy in the Escape($Q$, $Q_0$) (where $\langle Q, Q_0 \rangle = R(\langle S, F \rangle)$). As we remember from Section $\S$V-B$\S$ such a game Fugitive will first choose, as the initial position of the game, a structure $D_0 = w[a,b]$ 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) $D$ and Fugitive will satisfy this request, creating a new (slightly bigger) current $D$. Fugitive will win if he will be able to play forever (by which we mean $\omega^2$ steps), 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 (in order to prove 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 $D_0$ resulting from his initial move must be $(G(w))[a,b]$ for some $w \in Q_{\text{start}}$.

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

(III) He must never allow any path labelled by $G(Q_{\text{bad}}) \cup R(Q_{\text{ugly}})$ 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 such Fugitive. From the point of view of Crocodile’s operational objectives this strategy comprises of three stages.

In each of these stages the 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 $D_0$). In the first stage Fugitive will be forced to build a structure called $\mathbb{F}_1$ (defined in Section $\S$X). In the second stage the specified structures will be called $\mathbb{P}_m$ and $\mathbb{P}_m$ (each defined in Section $\S$X) and in the third stage Fugitive will be forced to construct one of the structures $\mathbb{G}_m$ or $\mathbb{L}_m$ (defined in Section $\S$XIII).

During the three stages of his play Crocodile will only pick requests from the languages in $Q_{\text{good}}$. 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.

They 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, $R(Q_{\text{bad}})(a,b)$ already holds true in the structure Fugitive was forced to construct. This will end the proof of (ii).

VIII. PRINCIPLE 1: $D_0$

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 groups $Q_{\text{bad}}$ and $Q_{\text{ugly}}$, 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 $D_0$ 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, and this problem is solved by having languages from $Q_{ugly} \cup Q_{bad}$ both in $Q$ and in $Q_0$:

**Lemma 1 (Principle I).** Fugitive must choose to start the Escape game from $D_0 = G(q)[a,b]$ for some $q \in Q_{start}$.

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 $D_0$.

**Proof.** If $D_0 = G(q)[a,b]$ for $q \in Q_0 \setminus Q_{start}$ then $D_0 \models G(L)(a,b)$ for some $L \in Q_{ugly} \cup Q_{bad}$. Then in the next step Crocodile can pick request $(a,b,G(L) \rightarrow R(L))$. After Fugitive satisfies this request, a structure $D_1$ is created such that $D_1 \models R(L)(a,b)$ and Crocodile wins. □

From now on we assume that Fugitive obeys Principle I. This implies that $D_0$ as demanded by Principle I will always be a substructure of any current structure $D$.

**IX. Principles II and III**

In this section we will formalise the intuition considering languages from $Q_{ugly}$ as forbidden patterns.

We start with an observation that will simplify our reasoning in the proof of Principle II.

**Observation IX.1.** For vertices $x, y$ in the current structure $D$ if there is a green (red) edge between them then Crocodile can force Fugitive to draw a red (green) edge between $x$ and $y$.

**Proof.** It is possible due to languages $1 - 9$ in $Q_{good}$. □

**Definition 2.** A P2-ready $^6$ structure $D$ is a structure satisfying the following:

- All edges labeled with $\alpha^C$ and $\alpha^W$ are between $a$ and $a'$.
- All edges incident to $b$ are $(b, b')$ with label $G(\omega)$ and $(b', b)$ with label $R(\omega)$.
- All edges labeled with $\omega$ are between $b'$ and $b$.
- For each $v \in D \setminus \{a,b\}$ there is a directed path in $D$, of length at most 4 from $a'$ to $v$ and there is a directed path in $D$, of length at most 4 from $v$ to $b'$.

**Lemma 3 (Principle II).** Suppose that, after Fugitive’s move, the current structure $D$ 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 $D$.

**Proof.** First suppose that there is such a green edge $e = \langle x, y \rangle$ with label ($\bullet^W$) in structure $D$. Let us denote by $P$ a path from $a'$ to $b'$ through $e$. Observe that if some of the edges of $P$ are red then from Observation IX.1 in at most 8 moves Crocodile can force Fugitive to create path $P^*$ which goes through the same vertices as $P$ (and also through $e$) but consists only of green edges. Because of this path there is a request generated by $Q_{ugly}^1$ between $a$ and $b$ so in the next step Crocodile can force Fugitive to create a red path connecting $a$ and $b$ labelled with a word from $Q_{ugly}^1$, which results in Crocodile’s victory.

In the second case assume there is a red edge $e = \langle x, y \rangle$ with label ($\bullet^C$) in structure $D$. Let us denote by $P$ a path from $a'$ to $b'$ through $e$. Observe that if some of the edges of $P$ are green then from Observation IX.1 in at most 8 moves Crocodile can force Fugitive to create path $P'$ which goes through the same vertices as $P$ but consists only of red edges. Because of this path $P'$ there is a red path connecting $a$ and $b$ labelled with a word from $Q_{ugly}^1$. □

**Lemma 4 (Principle III).** Fugitive must not allow any path labelled with a word from $R(Q_{bad}) \cup G(Q_{bad})$ to occur in the current structure $D$ between vertices $a$ and $b$.

**Proof.** First consider a case where $D \models R(Q_{bad})(a,b)$. Then Fugitive has already lost as $Q_{bad} \subset Q_0$. □
The second case is when $D \models G(Q_{\text{bad}})(a, b)$ and $D \not\models R(Q_{\text{bad}})(a, b)$. Then Crocodile can pick request $(a, b, Q_{\text{bad}})\langle i \rangle$ (for some $i$) for Fugitive to satisfy. In both cases after at most one move Fugitive loses.  

\[ \]

\section{X. The paths $P_m$ and $SP_m$}

\textbf{Definition 1.} (See Figure 2 and Figure 3) $P_m$, 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 $(p')_q$, where -- like before -- $p \in \{A, B\}$, $q \in \{H, V\}$ and $r \in \{W, C\}$. Each label has to also be either red or green. Notice that there is no $s \in S$ here: the labels we now use are sets of symbols from $\Sigma$ like in Notation VI.7.

The edges of $P_m$ are as follows:

- Each Cold edge (labelled with a symbol in \((\cdot)^C\)) is green.
- Each Warm edge (labelled with a symbol in \((\cdot)^W\)) is red.
- Each edge $\langle v_{2i}, v_{2i+1} \rangle$ is from $(A_H)_{1}$.
- Each edge $\langle v_{2i+1}, v_{2i+2} \rangle$ is from $(B_V)_{1}$.
- 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 $G(a^C)$ and $\langle a, a' \rangle$ with label $R(a^W)$ are in $E$.
- Edges $\langle b', b \rangle$ with label $G(\omega)$ and $\langle b', b \rangle$ $R(\omega)$ are in $E$.

\textbf{Definition 2.} $SP_m$ for $m \in \mathbb{N}_+$ is $P_m$ with two additional edges:

- $\langle v_{2m}, b' \rangle \in E$ with label $G(S^C)$.
- $\langle v_{2m}, b' \rangle \in E$ with label $R(S^W)$.

One may notice that $D_0$ is a substructure of both $P_m$ and $SP_m$, and that:

\[ \]

\[ \]

\section{XI. Stage I}

Recall that till the end of Section XIV we watch, and analyse, Fugitive’s and Crocodile’s play in shade filtering glasses. And we (of course) assume that Fugitive obeys Principle I, II and III.

\textbf{Definition 1 (Crocodile’s strategy).} 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 $S = (l) \vdash S'$ 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 Crocodile proceeds with strategy \( S' \).

Now we define a set of strategies for Crocodile. All languages that will appear in these strategies are from \( Q_{\text{good}} \) so instead of writing \( Q'_{\text{good}} \) we will just write \( i \). Let:

\[
\begin{align*}
S_{\text{color}} & := (3, 4, 5, 6, 7, 8, 9), \\
S_{\text{cycle}} & := (15, 14) ++ S_{\text{color}} ++ (12, 13) + S_{\text{color}}, \\
S_{\text{start}} & := (1, 2) ++ S_{\text{cycle}}.
\end{align*}
\]

Recall that \( D_0 \) is the Fugitive's initial structure (consisting of green edges only), as demanded by Principle I.

**Lemma 2.** Crocodile's strategy \((1, 2)\) applied to the current structure \( D_0 \) forces Fugitive to add \( R(\alpha^W)\langle a, a' \rangle \) and \( R(\omega)\langle b', b \rangle \).

**Proof.** Consider these languages one by one:

1 = \( \omega \): This language generates only one request \( \langle b', b, Q^2_{\text{good}} \rangle \) (one because edge \( \langle b', b \rangle \) with label \( G(\omega) \) is the only one in \( D_0 \) labelled with \( \omega \), which has to be satisfied with \( \omega \langle b', b, Q^2_{\text{good}} \rangle \) consists of only one word.

2 = \( \alpha^C + \alpha^W \): There is a green edge labelled with \( \alpha^C \) in \( D_0 \) and thus this language generates a request \( \langle a, a', Q^2_{\text{good}} \rangle \) (and no other requests). This request can be satisfied by Fugitive either by adding the edge \( R(\alpha^C)\langle a, a' \rangle \) or by adding the edge \( R(\alpha^W)\langle a, a' \rangle \). Suppose that Fugitive satisfies this request with \( R(\alpha^C)\langle a, a' \rangle \). Notice that Crocodile can now require Fugitive to satisfy requests \( \text{Reqs} = \{ \langle a', v_0, Q^2_{\text{good}} \rangle, \langle v_2, b, Q^2_{\text{good}} \rangle, \langle v_0, v_1, Q^2_{\text{good}} \rangle, \langle v_1, v_2, Q^2_{\text{good}} \rangle \} \) which will force the Fugitive to build a red path from \( a' \) to \( b' \). Each of these requests has to be satisfied with a red edge with some label \( \text{em warm} \) (with the upper index \( W \) or \( \text{cold} \) with \( C \)).

Consider what happens if one of these requests is satisfied with a \( \text{warm} \) letter. Then we have that \( D \models R(Q_{\text{ugly}})(a, b) \) and Fugitive loses. It means that each request from \( \text{Reqs} \) must be satisfied with a red edge labelled with a \( \text{cold} \) letter. But then notice that \( D \models R(Q_{\text{start}})(a, b) \) and Fugitive also loses.

---

A careful Reader could ask here: “Why did we need to work so hard to prove that the newly added red edge must be \text{warm}. Don’t we have Principle II which says that red edges must always be \text{warm} and green must be \text{cold}?” But we cannot use Principle II here – the structure is not \( P2\text{-ready} \) yet. Read the proof of Principle II again to notice that this red \( \alpha^W \) between \( a \) and \( a' \) is crucial there. And this is actually, what Stage I is all about: it is here where Crocodile forces Fugitive to construct a structure which is \( P2\text{-ready} \). From now on all the current structures will be \( P2\text{-ready} \) and Fugitive will indeed be a slave of Principle II.

uuu!

The following Lemma explains the role of \( S_{\text{color}} \) and is a first cousin of Observation \([X.1] \).

**Lemma 3 (\( S_{\text{color}} \)).** Strategy \( S_{\text{color}} \) applied to a \( P2\text{-ready} \) \( D \) forces Fugitive to create a \( P2\text{-ready} \) \( D' \) such that:

- Sets of vertices of \( D \) and \( D' \) are equal.
- There are no requests generated by \( Q_{\text{good}} \) in \( D' \), which means that each edge has its counterpart (incident to the same vertices) of the opposite color and temperature.

**Proof.** The proof is an easy consequence of Principle II and the fact that all words from \( Q_{\text{good}} \) have length one (which means that when satisfying the requests Fugitive only creates new edges, but no new vertices are added) and that these languages contain all symbols from \( \Sigma \).

**Lemma 4.** Strategy \( S_{\text{start}} \) applied to \( D_0 \) forces Fugitive to build \( P_1 \).

**Proof.** Consider languages from \( S_{\text{start}} \) one by one:

1 = \( \omega \): By Lemma 2 this language forces Fugitive to add \( R(\omega)\langle b', b \rangle \).

2 = \( \alpha^C + \alpha^W \): By Lemma 2 this language forces Fugitive to add \( R(\alpha^W)\langle a, a' \rangle \).

15 = \( y^W + x^C \): This language generates only one request \( \langle v_0, b', Q^2_{\text{good}} \rangle \) since neither \( y^W \) nor \( x^C \) occurs in the current structure. This request has to be satisfied with \( R(y^W)\langle v_0, b' \rangle \) by Principle II. We can use Principle II since after strategy \((1, 2)\) was applied the structure is \( P2\text{-ready} \).

14 = \( x^W + x^C + x^C(\mathbf{A}^C_U)\langle \mathbf{B}^C_V \rangle \): This language generates only two requests.

---

\(^8\)In order for this “then” to make sense we need the total number of moves of the game to be \( \omega^2 \) rather than \( \omega \).
then we are done, and notice that the last $S_{\text{good}}$ proven for Proof.

For all $\text{Lemma 1.}$

Now $\text{Crocodile}$ uses strategy $S_{\text{color}}$ to add missing edges of opposite colors (and, by Principle II, of opposite temperatures).

- $12 = x^C ((A^{C}_{H}) + (B^{C}_{H}) + (A^{C}_{V}) + (B^{C}_{V})) + x^C + x^W$: This language generates only one request: $\langle a', v_1, Q^{12}_{\text{good}} \rangle$. It is because there are no requests generated by either $x^C$ nor $x^W$ in $Q^{12}_{\text{good}}$ by Lemma 3. There are also no other requests generated by $x^C ((A^{C}_{H}) + (B^{C}_{H}) + (A^{C}_{V}) + (B^{C}_{V}))$ in $Q^{12}_{\text{good}}$ as the only path labeled with a word from this language is $a' \rightarrow v_0 \rightarrow v_1$. $\langle a', v_1, Q^{12}_{\text{good}} \rangle$ has to be satisfied with $R(x^W)[a', v_1]$ by Principle II.

- $13 = ((A^{C}_{V}) + (B^{C}_{V}) + (A^{C}_{V}) + (B^{C}_{V})) y^C + y^W$: This language generates one request $\langle v_1, b', Q^{14}_{\text{good}} \rangle$. It has to be satisfied with $R(y^W)[v_1, b']$ by Principle II.

Finally $\text{Crocodile}$ uses strategy $S_{\text{color}}$ to add two missing edges $\langle a', v_1 \rangle$ with label $G(x^C)$ and $\langle v_1, b' \rangle$ with label $G(y^C)$ to build $\mathbb{P}_1$.

$\square$

XII. Stagel II

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:

- $S_1 := S_{\text{start}},$
- $S_k := S_{k-1} \Rightarrow S_{\text{cycle}}$ for $k > 1$.

Lemma 1. For all $m \in \mathbb{N}_+$ strategy $S_m$ applied to $\mathbb{P}_m$ forces Fugitive to build, depending on his choice, either $\mathbb{P}_{m+1}$ or $\$\mathbb{P}_k$ for some $k \leq m$.

Proof. Notice that by, Lemma 4 this is already proven for $m = 1$. Now assume that $\text{Crocodile}$, using strategy $S_{m-1}$, forced $\text{Fugitive}$ to build $\mathbb{P}_m$ or $\$\mathbb{P}_k$, for some $k \leq m - 1$. If Fugitive built $\$\mathbb{P}_k$ already as the result of $\text{Crocodile}$’s strategy $S_{m-1}$ then we are done, and notice that the last $S_{\text{cycle}}$ will not change the current structure any more — this is because, due to Exercise ?? there are no requests from languages $Q^{1-9}_{\text{good}}$ and $Q^{12-15}_{\text{good}}$ in the current structure at this point.

So we only need to consider the case where $\mathbb{P}_m$ was built. Now $\text{Crocodile}$ uses strategy $S_{\text{cycle}}$ to force $\text{Fugitive}$ to build $\mathbb{P}_{m+1}$ or $\$\mathbb{P}_m$. Consider languages from $S_{\text{cycle}}$ one by one:

- $15 = y^W + \$C + (A^{C}_{H})(B^{C}_{V})y^C$. The only request generated by this language is $\langle v_{2m+1}, b', Q^{15}_{\text{good}} \rangle$, resulting from the red edge labelled with $y^W$ connecting $v_{2m+1}$ and $b'$. This is since there is no $\$C$ anywhere in the current structure, and since for each $k < m$ there are already both a red edge labelled with $y^W$ from $v_{2k}$ to $b'$ and a green paths labelled with $(A^{C}_{H})(B^{C}_{V})y^C$ between these vertices.

This only request can be possibly satisfied in two different ways (it follows from Principle II): either by $G((A^{C}_{H})(B^{C}_{V})y^C)[v_{2m+1}, b']$ or by $G(\$C)[v_{2m+1}, b']$. The case when this request is satisfied with $G(\$C)[v_{2m+1}, b']$ will be considered in the last paragraph of the proof. So now we assume that this request is satisfied with $G((A^{C}_{H})(B^{C}_{V})y^C)[v_{2m+1}, b']$. Let us name the two new vertices as $v_{2m+1}$ and $v_{2m+2}$.

- $14 = x^W + x^C + x^C(A^{C}_{H})(B^{C}_{V})$: the only request generated by this language is $\langle a', v_{2m+2}, Q^{14}_{\text{good}} \rangle$ resulting from the (partially) newly created green path from $a'$ to $v_{2m+2}$, via $v_{2m}$ and $v_{2m+1}$, labelled with $x^C(A^{C}_{H})(B^{C}_{V})y^C$.

This request has to be satisfied with $R(x^W)[a', v_{2m+2}]$ due to Principle II.

Now $\text{Crocodile}$ uses strategy $S_{\text{color}}$ to add missing edges of opposite colors.

- $12 = x^C ((A^{C}_{H}) + (B^{C}_{H}) + (A^{C}_{V}) + (B^{C}_{V})) + x^C + x^W$: This language generates one request: $\langle a', v_{2m+1}, Q^{12}_{\text{good}} \rangle$. It has to be satisfied with $R(x^W)[a', 2m+1]$ by Principle II.

- $13 = ((A^{C}_{V}) + (B^{C}_{V}) + (A^{C}_{V}) + (B^{C}_{V})) y^C + y^W$: This language generates one request $\langle v_{2m+1}, b', Q^{13}_{\text{good}} \rangle$. It has to be satisfied with $R(y^W)[2m+1, b']$ by Principle II.

Now $\text{Crocodile}$ uses strategy $S_{\text{color}}$ (as $S_{\text{cycle}} = (15, 14) \Rightarrow S_{\text{color}} \Rightarrow (12, 13) \Rightarrow S_{\text{color}}$). We apply Lemma 3 to conclude that $\text{Fugitive}$ is forced to build $\mathbb{P}_{m+1}$, as what is left to create $\mathbb{P}_{m+1}$ is to only add some edges of opposite colors and
temperatures.

Notice that during play, after application of each language in Crocodile’s strategy, each of the constructed structures is P2-ready, as distances from \( a' \) and to \( b' \) are smaller than 4.

As we said, we assume that \( S \) generated by the remaining languages from \( Q \) is isomorphic to \( L \). Notice that the only request generated by the remaining languages from \( S_{cycle} \) is: \( (v_{2m}, b', Q_{good}) \), which will be satisfied by \( R(W) \), and the resulting structure will be isomorphic to \( P_m \). This ends the proof of Lemma 1.

XIII. The Grids \( G_m \) and Partial Grids \( L^k_m \)

Definition 1. \( G_m \) for \( m \in \mathbb{N}_+ \) is a directed graph \( (V, E) \) where:

\[
V = \{a, a', b, b\} \cup \{v_{i,j} : i, j \in [0, m]\} \text{ and the edges } E \text{ are labelled (as in } P_m \text{) with } \sigma_1 \setminus \sigma_0 \text{ or one of the symbols of the form } (p'_q), \text{ which means that the shade-filtering glasses are still on.}
\]

The edges of \( G_m \) 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, and there are no other edges.

- Each cold edge, labelled with a symbol in \( \sigma C \), is green.

- Each warm edge, labelled with a symbol in \( \sigma W \), is red.

- Each edge \( (v_{i,j}, v_{i+1,j}) \) is horizontal – its label is from \( \bullet_h \).

- Each edge \( (v_{i,j}, v_{i,j+1}) \) is vertical – its label is from \( \bullet_v \).

- The label of each edge leaving \( v_{i,j} \), with \( i + j \) even, is from \( (A) \), the label of each edge leaving \( v_{i,j} \), with \( i + j \) odd, is from \( (B) \).

- Each edge \( (a', v_i) \) is labeled by either \( xC \) or \( xW \).

- Each edge \( (v_i, b') \) is labeled by either \( yC \) or \( yW \).

- Edges \( (a, a') \) with label \( G(\sigma C) \) and \( (a, a') \) with label \( R(\sigma W) \) are in \( E \).

- Edges \( (b', b) \) with label \( G(\omega) \) and \( (b', b) \) with label \( R(\omega) \) are in \( E \).

Definition 2. \( L^k_m = (V', E') \) for \( m \in \mathbb{N}_+, k \in \mathbb{N}_+, k \leq m \) is a subgraph of \( G_m = (V, E) \) induced by the set of vertices \( V' \subseteq V, V' = \{a, a', b, b\} \cup \{v_{i,j} : i, j \in [0, m]; i - j \leq k; j - i \leq k\} \).

Definition 3. \( L^k_m \) for \( m \in \mathbb{N}_+ \) is \( G_m \) with two edges added:

- \( \langle v_{m,m}, b' \rangle \) with label \( G(\sigma C) \)

- \( \langle v_{m,m}, b' \rangle \) with label \( R(\sigma W) \)

Definition 4. \( L^{k+1}_m \) for \( m \in \mathbb{N}_+, k \in \mathbb{N}_+ \cup \{0\}, k \leq m \) is \( L^k_m \) with two edges added:

- \( \langle v_{m,m}, b' \rangle \) with label \( G(\sigma C) \)

- \( \langle v_{m,m}, b' \rangle \) with label \( R(\sigma W) \)

Exercise XIII.5. For all \( m \):

- \( L^m \) is equal to \( G_m \).

- \( L^m \) is equal to \( G_m \).

Exercise XIII.6. For all \( m \) there are no requests generated by languages from \( Q_{good} \) or \( Q_{ugly} \) in \( L^m \).

XIV. Stage III

Now we imagine that either \( P_{m+1} \) or \( L^k \) 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 \( G_{m+1} \) or \( G_k \) will be created.

Lemma 1. For any \( m \in \mathbb{N}_+ \), Crocodile can force Fugitive to build a structure isomorphic, depending on Fugitive’s choice, to either \( G_{m+1} \) or \( G_k \) for some \( k \leq m \).

Notice that by Exercise XIII.5 in order to prove Lemma 1 it is enough to prove that for any \( m \in \mathbb{N}_+ \), Crocodile can force the Fugitive to build a structure isomorphic to either \( L_{m+1} \) or \( L^k \) for some \( k \leq m \).

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

Now we define a sequence of strategies \( S^k_{layer} \), which, similarly to strategies for building \( P_m \),
Figure 4. $G_q$ (left). Smaller picture in the top-right corner explains how different line styles on the main picture map to $\Sigma_0^\P$.

Figure 5. $L_6^1$.

Figure 6. $L_6^1$.

consist only of languages from $Q_{\text{good}}$, so instead of writing $Q_{\text{good}}'$ we will just write $i$. Let:

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

Lemma 2. For all $k \in \mathbb{N}$ strategy $S^1_{\text{layer}}$ applied to the current structure $\$P_k$ forces the Fugitive to build $\$L_1^1$.

Proof. Assume the current structure is $\$P_k$. Consider languages from $S^1_{\text{layer}}$:

- $11 = (A^C_H)(B^C_V) + (A^W_V)(B^W_H)$: This language generates one request of the form $\langle v_i, v_{i+2}, Q_{\text{good}} \rangle$ for every $i \in [0, 2k - 2]$. 

Page 15 of 19
Each of these requests results from a green path labeled with \( G((A'_q^c)(B'_w^c)) \) connecting \( v_i \) and \( v_{i+2} \).

Notice that there are no requests generated by \( Q^{11, -} \). It is because neither \( (A'_q^w) \) nor \( (B'_w^c) \) occurs in \( S^k \).

All generated requests have to be satisfied with \( R((A'_q^w)(B'_w^c)) \) by Principle II. Notice that when satisfying each request a new vertex is created.

- \( S_{\text{color}} = (3, 4, 5, 6, 7, 8, 9) \): This sequence of languages adds missing green edges \( G(A'_q^c) \) and \( G(B'_w^c) \) to the edges \( R(A'_q^w) \) and \( R(B'_w^c) \) created by language 11.

- \( 12 = x^C ((A'_q^c) + (B'_w^c) + (A'_q^w) + (B'_w^c)) + x^C + x^W \): This language generates requests of the form \( \{i, t, Q^{12, -} \} \) for all new vertices \( t \) created by language 11. Each of these requests results from a green path labeled with \( x^C ((A'_q^c) + (B'_w^c) + (A'_q^w) + (B'_w^c)) \) connecting \( a' \) and \( t \) for some vertex \( t \) created by language 11.

Notice that there are no other requests generated since by Lemma 3 after applying strategy \( x^C \) has its counterpart labeled with \( R(x^W) \).

All generated requests have to be satisfied with \( R(x^W) \) by Principle II.

- \( 13 = ((A'_q^c) + (B'_w^c) + (A'_q^w) + (B'_w^c)) y^C + y^C + y^W \): This language generates requests of the form \( \{(t', b', Q^{13, -} )\} \) for all new vertices \( t' \) created by language 11. Each of these requests results from a green path labeled with \( ((A'_q^c) + (B'_w^c) + (A'_q^w) + (B'_w^c)) y^C \) connecting \( t \) and \( b' \) for some vertex \( t \) created by language 11.

Notice that there are no other requests generated since by Lemma 3 after applying strategy \( x^C \) has its counterpart labeled with \( R(y^W) \).

All these requests have to be satisfied with \( R(y^W) \) by Principle II.

- \( S_{\text{color}} = (3, 4, 5, 6, 7, 8, 9) \): This sequence of languages adds missing green edges \( G(x^C) \) and \( G(y^C) \) to edges added by languages 12 and 13.

\[ \square \]

Lemma 3. For all \( k, m \in \mathbb{N}, k < m \) strategy \( S^\text{odd} \) (for \( k + 1 \) odd) and \( S^\text{even} \) (for \( k + 1 \) even) applied to \( S^k \) forces the Fugitive to build \( S^{k+1} \).

Proof. Assume the Escape game starts from \( S^k \) for odd \( k < m \). The proof for the case where \( k \) is even is analogous. Consider languages from \( S^\text{even} \):

- \( 10 = (B'_w^c)(A'_q^w) + (A'_q^c)(B'_w^c) \): generates exactly \( \{v_{i,j}, v_{i+1,j+1}, Q^{10, -} \} \) \( i - j = k, i,j \in [0, m - 1] \). All requests in the first group result from paths labeled with \( G((B'_w^c)(A'_q^w)) \) and all requests in the second group result from paths labeled with \( R((B'_w^c)(A'_q^w)) \).

All requests in the first group have to be satisfied with \( R((B'_w^c)(A'_q^w)) \) (name the new vertices \( v_{i+1,j} \) and all requests in the second group have to be satisfied with \( G((B'_w^c)(A'_q^w)) \) (name the new vertices \( v_{i,j+1} \)). All happen by Principle II.

- \( S_{\text{color}} \): adds missing edges of opposite colors incident to newly created vertices by language 11.

- \( 12 = x^C ((A'_q^c) + (B'_w^c) + (A'_q^w) + (B'_w^c)) + x^C + x^W \): generates exactly \( \{i', t, Q^{12, -} \} \) \( i,j \in [0, m - 1] \) \( \cup \{i', v_{i+1,j+1}, Q^{12, -} \} \) \( i,j \in [0, m - 1] \). All requests in the first group result from paths labeled with \( x^C ((A'_q^c) + (B'_w^c) + (A'_q^w) + (B'_w^c)) \) connecting \( a' \) and \( t \) for some vertex \( t \) created by language 10.

Notice that there are no other requests generated since by Lemma 3 after applying strategy \( x^C \) has its counterpart labeled with \( R(x^W) \).

All the requests have to be satisfied with \( R(x^W) \) by Principle II.

- \( 13 = ((A'_q^c) + (B'_w^c) + (A'_q^w) + (B'_w^c)) y^C + y^C + y^W \): generates exactly \( \{v_{i,j}, b', Q^{13, -} \} \) \( i,j \in [0, m - 1] \) \( \cup \{v_{i,j}, b', Q^{13, -} \} \) \( i,j \in [0, m - 1] \). Each of these requests results from a green path labeled with \( ((A'_q^c) + (B'_w^c) + (A'_q^w) + (B'_w^c)) y^C \) connecting \( t \) and \( b' \) for some vertex \( t \) created by language 10.

Notice that there are no other requests generated since by Lemma 3 after applying strategy \( x^C \) has its counterpart labeled with \( R(y^W) \).

All the requests have to be satisfied with \( R(y^W) \) by Principle II.

- \( S^\text{odd} = (3, 4, 5, 6, 7, 8, 9) \): This sequence of languages adds missing green edges \( G(x^C) \) and \( G(y^C) \) to edges added by languages 12 and 13.

\[ \square \]
Let \( m \) be value from Lemma 1. By Lemma 1 the Crocodile can force the Fugitive to build a structure isomorphic to either \( G_{m+1} \) or \( G_k \) 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 \( \alpha, \omega, x \) and \( y \)) having one of the shades from \( S \). 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 \( S \).

Now we consider two cases:

- If \( G_{m+1} \) was built then clearly condition (b3) of Definition 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 \( Q_{bad}^{1} \).
- If \( G_k \) for \( k \leq m \) was built then clearly condition (b2) or (b3) of Definition 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 \( Q_{bad}^{1} \).

This ends the proof of Lemma 7 (ii).

For the proof of Lemma 7 (i) assume the original instance \( \langle S, F \rangle \) of Our Grid Tiling Problem has a proper shading— a labeled grid of side length \( m \). Call this grid \( G \).

Recall that \( G_m \) satisfies all regular constraints from \( Q_{good}^{++} \) and from \( Q_{ugly}^{++} \) (Exercise XIII.6). Now copy the shades of the edges of \( G \) to the respective edges of \( G_m \). Call this new structure \( \tilde{G}_m \) with shades added \( \tilde{M} \). It is easy to see that \( \tilde{M} \) constitutes a counterexample, as in Lemma 1.

**References**

[AV97] S. Abiteboul and V. Vianu, Regular path
queries with constraints; Proc. of the 16th PODS, pp. 122–133, 1997;

[A11] F. N. Afrati, Determinacy and query rewriting for conjunctive queries and views; Th.Comp.Sci. 412(11):1005–1021, March 2011;

[AG08] R. Angles, C. Gutierrez, Survey of Graph Database Models; ACM Comp. Surveys Vol. 40 Issue 1, February 2008;

[B13] P. Barceló, Querying graph databases. Simple Paths Semantics vs. Arbitrary Path Semantics; PODS 2013, pp. 175–188;

[CMW87] I. F. Cruz, A. O. Mendelzon, and P. T. Wood, A graphical query language supporting recursion; Proc. of ACM SIGMOD Conf. on Management of Data, 1987;

[CGL98] D. Calvanese, G. De Giacomo, and M. Lenzerini, On the decidability of query containment under constraints; in Proc. of the 17th PODS," pp. 149–158, 1998;

[CGLV00] D. Calvanese, G. De Giacomo, M. Lenzerini, M.Y. Vardi. Answering regular path queries using views; Proc. 16th Int. Conf. on Data Engineering, pages 389–398, IEEE, 2000;

[CGLV00a] D. Calvanese, G. De Giacomo, M. Lenzerini, M. Y. Vardi. View-based query processing and constraint satisfaction; Proc. of 15th IEEE LICS, 2000;

[CGLV02] D. Calvanese, G. De Giacomo, M. Lenzerini, M.Y. Vardi. Lossless regular views; Proc. of the 21st PODS, pages 247–258, 2002;

[CGLV02a] D. Calvanese, G. De Giacomo, M. Lenzerini, and M.Y. Vardi. Rewriting of regular expressions and regular path queries; Journal of Comp. and System Sc., 64:443–465, 2002;

[DPT99] A. Deutsch, L. Popa, and Val Tannen, Physical data independence, constraints, and optimization with universal plans; Proc. of 25th VLDB, pages 459–470, 1999;

[F15] Nadime Francis, PhD thesis, ENS de Cachan, 2015;

[F17] N. Francis; Asymptotic Determinacy of Path Queries Using Union-of-Paths Views; Th.Comp.Syst. 61(1):156-190 (2017);

[FG12] E. Franconi and P. Guagliardi The view update problem revisited CoRR, abs/1211.3016, 2012;

[FGZ12] Wenfei Fan, F. Geerts, and Lixiao Zheng, View determinacy for preserving selected information in data transformations; Inf. Syst., 37(1):1–12, March 2012;

[FLS98] D. Florescu, A. Levy, and D. Suciu, Query containment for conjunctive queries with regular expressions; Proc. of the 17th PODS," pp. 139–148, 1998;

[FV98] T. Feder and M. Y. Vardi, The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory; SIAM Journal on Computing, 28(1):57–104, 1998;

[FSS14] N. Francis, L. Segoufin, C. Sirangelo Datalog rewritings of regular path queries using views; Proc. of ICDT, pp 107–118, 2014;

[GB14] M. Guarnieri, D. Basin, Optimal Security-Aware Query Processing; Proc. of the VLDB Endowment, 2014;

[GM15] T. Gogacz, J. Marcinkowski, The Hunt for a Red Spider: Conjunctive Query Determinacy Is Undecidable; LICS 2015: 281-292;

[GM16] T. Gogacz, J. Marcinkowski, Red Spider Meets a Rainworm: Conjunctive Query Finite Determinacy is Undecidable; PODS 2016: 121-134;

[GM018] G. Głuch, J. Marcinkowski, P. Ostropolski-Nalewaja Can One Escape Red Chains? Regular Path Queries is Undecidable.; LICS 2018: 492-501;

[JV09] V. Juge and M. Vardi, On the containment of Datalog in Regular Datalog; Technical report, Rice University, 2009;

[LY85] Per-Ake Larson and H. Z. Yang, Computing queries from derived relations; Proc. of the 11th International Conference on Very Large Data Bases - Volume 11, VLDB’85, pages 259–269. VLDB Endowment, 1985;

[NSV06] A. Nash, L. Segoufin, and V. Vianu, Determinacy and rewriting of conjunctive queries using views: A progress report; Proc. of ICDT 2007, LNCS vol. 4353; pp 59–73;

[NSV10] A. Nash, L. Segoufin, and V. Vianu. Views and queries: Determinacy and rewriting; ACM Trans. Database Syst., 35:21:1–21:41, July 2010;

[P11] D. Pasaila, Conjunctive queries determinacy and rewriting; Proc. of the 14th ICDT, pp. 220–
231, 2011;

[RRV15] J. Reutter, M. Romero, M. Vardi, Regular queries on graph databases; Proc. of the 18th ICDT; pp 177–194; 2015;

[V16] M.Y. Vardi, A Theory of Regular Queries; PODS/SIGMOD keynote talk; Proc. of the 35th ACM PODS 2016, pp 1-9;