Can One Escape Red Chains? Regular Path Queries Determinacy is Undecidable*

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

ACM Reference Format:
Grzegorz Głuch, Jerzy Marcinkowski, Piotr Ostropolski-Nalewaja Institute of Computer Science, University of Wrocław. 2018. Can One Escape Red Chains? Regular Path Queries Determinacy is Undecidable. In *LICS '18: LICS '18: 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, July 9–12, 2018, Oxford, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209108.3209120

Abstract. For a given set of queries (which are expressions in some query language) $\mathcal{Q} = \{Q_1, Q_2, \ldots Q_k\}$ and for another query Q_0 we say that $\mathcal{Q}$ determines Q_0 if – informally speaking – for every database $\mathbb{D}$, the information contained in the views $\mathcal{Q}(\mathbb{D})$ is sufficient to compute $Q_0(\mathbb{D})$.

Query Determinacy Problem is the problem of deciding, for given $\mathcal{Q}$ and Q_0, whether $\mathcal{Q}$ determines Q_0. Many versions of this problem, for different query languages, were studied in database theory. In this paper we solve a problem stated in [CGLV02] and show that Query Determinacy Problem is undecidable for the Regular Path Queries – the paradigmatic query language of graph databases.

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 $Q_1(\mathbb{D}), Q_2(\mathbb{D}), \ldots Q_k(\mathbb{D})$? 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 problem in the area of database theory, with a 30 years long history as a research subject – see for example [H01], or Nadime Francis thesis [F15] for a survey. In [DPT99] QDP naturally appears in the context of query evaluation plans optimization. More recent examples are [FG12], where the context for QDP is the view update problem or [FKN13], where the context is description logics. In the above examples the goal is optimization/efficiency so we "prefer" Q_0 to be determined by $\mathcal{Q}$. Another context, where it is "preferred" that Q_0 is not determined, is privacy: we would like to release some views of the database, but in a way that does not allow certain query to be computed.

The oldest paper we were able to trace, where QDP is studied, is [LY85]. Over the next 30 years many decidable and undecidable cases have been identified. Let us just cite some more recent results: [NSV10] shows that the problem is decidable for conjunctive queries if each query from $\mathcal{Q}$ has only one free variable; in [A11] decidability is shown for $\mathcal{Q}$ and Q_0 being "conjunctive path queries". This is generalized in [P11] to the the scenario where $\mathcal{Q}$ are conjunctive path queries but Q_0 is any conjunctive query.

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 $\mathcal{Q}$ and Q_0. In [NSV10] 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 negative result is presented in [FGZ12]: 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 [GM15] and [GM16] that QDP is undecidable for Q_0 and the queries in $\mathcal{Q}$

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

LICS '18, July 9–12, 2018, Oxford, United Kingdom

ACM ISBN 978-1-4503-5583-4/18/07…$15.00
https://doi.org/10.1145/3209108.3209120

[1] Or, using the language of [CGLV00], [CGLV00a] [CGLV02] and [CGLV02a], 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.

being conjunctive queries.

QDP for Regular Path Queries. While the determinacy problem is now well understood for the pure relational model[3], it has been, for a long time, open for the graph databases scenario. In this scenario, the underlying data is modeled 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 optimization. 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].

It is natural that 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").

A method for computing a rewriting of a regular path query in terms of other regular expressions (if such rewriting exists) [4] is shown in [CGLV02]. And it is proven that it is 2ExpSpace-complete to decide whether there exists a rewriting of the query that can be expressed as a regular path query. Then a notion of monotone determinacy is defined, meaning that not only $Q_0(\mathbb{D})$ is a function[5] of $\mathcal{Q}(\mathbb{D})$ but this function is also monotone – the greater $\mathcal{Q}(\mathbb{D})$ (in the inclusion ordering) the greater $Q_0(\mathbb{D})$, and it is shown that monotone determinacy is decidable in ExpSpace. This proves that monotone determinacy, which is – like rewritability – also a notion related to determinacy but stronger, does not coincide with the existence of a regular path rewriting, which is 2ExpSpace-complete (while of course the existence of rewriting implies monotonicity). This proof is indirect and it is interesting that a specific example separating monotone determinacy and rewritability has only been shown in [FSS14]. However, [CGLV02a] also provides an example where a regular path view determines a regular path query in a non-monotone way showing that, in this setting, determinacy does not coincide with monotone determinacy.

In [CGLV02], apart from the standard QDP, the authors consider the so called "losslessness under sound semantics". They show that computing "certain answers" (under this semantics) of a regular path query with respect to a regular path view reduces to the satisfiability of (the negation of) uniform CSP (constraint satisfaction problem). Building on this connection and on the known links between CSP and Datalog [FV98], they show how to compute approximations of this CSP in Datalog. This is studied in more detail in [FSS14] and a surprising result is proved, that when a regular path view determines a regular path query in a monotone way, then one of the approximations is exact.

But, despite the considerable body of work in the area around the main problem, little was so far known about the problem of decidability of QDP for RPQ itself. 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, and the languages defining the queries in Q_0 and $\mathcal{Q}$ are over a single-letter alphabet. The failure to solve the problem completely even for this very simple variant shows how complicated things very quickly become. But it is the analysis which is so obviously hard (not QDP itself as a computational problem) and it is not immediately clear how QDP for RPQ could be used to encode anything within. In consequence, no lower bounds have been known so far, except of a simple one from [F15], where undecidability is shown if Q_0 can be context-free rather than just regular.

Our contribution. The main result of this paper is:

Theorem 1.1. *QDP-RPQ, the Query Determinacy Problem for Regular Path Queries, is undecidable.*

To be more precise, we show that the problem, both in the "finite" and the "unrestricted" version, is co-r.e.-hard, which means that if we take, as an input to our encoding, a Turing machine which accepts (the empty input) then, as the result of the encoding we get a negative instance of QDP ("no determinacy"), and if we begin from a non-accepting machine then the resulting instance is positive. Notice that this gives the precise bound on the complexity of the "finite" version of QDP for RPQ – it is easy to see that finite non-determinacy is recursively enumerable. But there is no such upper bound for the "unrestricted" case, and we are not sure

[3] Apparently, when talking about the relational model, there may still be some work to do concerning QDP in the context of bag semantics, see [GB14].

[4] Existence of rewriting is a related property to determinacy, but stronger.

[5] $\mathbb{D}$ is an argument here. Saying that "$Q_0(\mathbb{D})$ is a function of $\mathcal{Q}(\mathbb{D})$" is equivalent to saying that $\mathcal{Q}$ determines Q_0.

what the precise complexity can be. We believe that the problem may be harder than co-r.e.-complete.

Regarding the technique we use: clearly we were tempted to save as much as possible from the techniques of [GM15] and [GM16]. But hardly anything survived in the new situation (one exception is that the idea of the green-red Chase from [G15] evolved into the notion of Escape here). The two important constructions in [GM15] and [GM16] used queries with high number of free variables (this is where states of the Turing machine are encoded, in the form of spiders with fancy colorings) and queries which can be homomorphically, non-trivially, mapped into themselves – this is how the original small structure ("green spider" in [GM15] and [GM16] or (green) $\mathbb{D}_0$ in this paper) could grow. None of the mechanisms is available in the current context, so in principle the whole proof was built from scratch.

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 The rest of this paper is devoted to the proof of Theorem 1.1. In short Section 2 we introduce the (very few) notions and some notations we need to use.

In Section 3 we first follow the ideas from [GM15] defining red-green signature. Then we define the game of Escape and state a crucial lemma (Lemma 3.3), asserting that this game really fully characterizes determinacy for Recursive Path Queries. In Section 3.3 we prove this Lemma.

At this point we will have all the tools ready for proving Theorem 1.1. In Section 4 we explain what is the undecidable problem we use for our reduction, and present the reduction. In Sections 5 – 10 we use the characterization provided by Lemma 3.3 to prove correctness of this reduction.

2 Preliminaries

Structures. When we say "structure" we always mean a directed graph with edges labeled with letters from some signature/alphabet Σ. In other words every structure we consider is 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.

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

Chains and chain queries. Given a set of binary predicate names Σ and a word $w = a_1 a_2 \dots a_n$ over Σ^* we define a chain query $w(x_0, x_n)$ as a conjunctive query:

$$\exists_{x_1,\dots,x_{n-1}} a_1(x_0, x_1) \wedge a_2(x_1, x_2) \wedge \dots a_n(x_{n-1}, x_n).$$

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 $x_0, x_1, \dots x_n$ and atoms $a_1(x_0, x_1)$, $a_2(x_1, x_2), \dots a_n(x_{n-1}, x_n)$.

Regular path queries. For a regular language Q over Σ 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 labeled with any word from Q and returns the endpoints of that path.

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 of: $Q(\mathbb{D}) = \{\langle x, y\rangle \mid \mathbb{D} \models Q(x, y)\}$.

3 Red-Green Structures and Escape

3.1 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 *Regular Constraint* $L \to L'$ as a formula

$$\forall_{x,y} L(x, y) \Rightarrow L'(x, y).$$

We use the notation $\mathbb{D} \models r$ to say that an RC r 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 $\langle x, y, L \to L'\rangle$ such that $\mathbb{D} \models L(x, y)$ and $\mathbb{D} \not\models L'(x, y)$. 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:

---

function ADD
arguments:
- Structure $\mathbb{D}$
- RC $L \to L'$
- pair $\langle x, y\rangle$ such that $\langle x, y, L \to L'\rangle \in rq(L \to L', \mathbb{D})$

body:
1: Take a word $w = a_0 a_1 \dots a_n$ from L' and create a new path $w[x, y] = a_0(x, x_1), a_1(x_1, x_2), \dots, a_n(x_{n-1}, y)$ where $x_1, x_2, \dots, x_{n-1}$ are **new** vertices
2: **return** $\mathbb{D} \cup w[x, y]$.

Notice that the result $Add(\mathbb{D}, L \to L', \langle x, y\rangle)$ 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 some regular language L.

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 $\langle x, y, t\rangle$ for some RC $t \in L^{\rightarrow}$ ($t \in L^{\leftarrow}$) are *generated by* $G(L)$ (resp. *by* $R(L)$). Requests generated by $G(L)$ or by $R(L)$) are said to be *generated by* L.

The following lemma is straightforward to prove and characterizes both determinacy and finite dterminacy in terms of regular constraints:

Lemma 3.1. *A set $\mathcal{Q}$ of regular path queries over Σ does not determine (does not finitely determine) regular path query Q_0, over the same alphabet, if and only if there exists a structure (resp. a finite structure) $\mathbb{M}$ and a pair of vertices $a, b \in \mathbb{M}$ such that $\mathbb{M} \models \mathcal{Q}^{\leftrightarrow}$ and $\mathbb{M} \models (G(Q_0))(a, b)$ but $\mathbb{M} \not\models (R(Q_0))(a, b)$.*

Any structure $\mathbb{M}$, as above, will be called *counterexample*.

3.2 The game of Escape

An instance Escape(Q_0, $\mathcal{Q}$) of a solitary game called *Escape*, played by a player called *Fugitive*, is:

- a regular language Q_0 of *forbidden chains* over Σ.
- a set of regular languages $\mathcal{Q}$ 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 $\mathbb{D}_i$ is the position of the game after Fugitive move i and $S_i = rq(\mathcal{Q}^{\leftrightarrow}, \mathbb{D}_i)$. Then, in move $i + 1$, Fugitive can move to any position of the form:

$$\mathbb{D}_{i+1} = \bigcup_{\langle x,y,t\rangle \in S_i} Add(\mathbb{D}_i, t, \langle x, y\rangle)$$

- Fugitive loses when for a *final position* $\mathbb{H} = \bigcup_{i=0}^{\infty} \mathbb{D}_i$ it is true that $\mathbb{H} \models (R(Q_0))(a, b)$.

In other words, in order to get $\mathbb{D}_{i+1}$, Fugitive needs to create, simultanously for each request in $\mathbb{D}_i$, a new path that satisfies this request, and add all these paths, in a free way, to $\mathbb{D}_i$. This is of course very much non-deterministic, so position $\mathbb{D}_{i+1}$ depends on the Fugitive's choice[6].

Let us note that $\mathbb{D}_{i+1} = \mathbb{D}_i$ when $rq(\mathcal{Q}^{\leftrightarrow}, \mathbb{D}_i)$ is empty.

It also would not hurt if, before proceeding with the reading, the Reader wanted to solve:

Exercise 3.2. *Notice that if i is even (odd) then all the requests from $\mathcal{S}_i$ are generated by $G(L)$ (resp. $R(L)$), for some $L \in \mathcal{Q}$ which means that all the edges added by* Fugitive *in his move $i + 1$ are red (resp. green).*

[6]Like in any reasonable game, the position after each move depends here on the position before this move, on the rules of the game, and on the decisions of the player who makes this move.

Let *step* be ternary relation such that $\langle \mathbb{D}, \mathbb{D}', \mathcal{L}\rangle \in step$ when $\mathbb{D}'$ can be the result of one move of Fugitive, in position $\mathbb{D}$, in the game of Escape with set of regular languages $\mathcal{L}$.

Obviously, different strategies of Fugitive may lead to different final positions. We will denote set of all final positions reachable from a starting structure $\mathbb{D}_0$, for a set of regular languages $\mathcal{L}$, as $\Omega(\mathcal{L}^{\leftrightarrow}, \mathbb{D}_0)$.

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

Lemma 3.3. *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, $\mathcal{Q}$).

3.3 Universality of Escape. Proof of Lemma 3.3

First let us leave it as an easy exercise for the Reader to prove:

Lemma 3.4. *For each set of RCs T, for each initial position $\mathbb{D}_0$ and for each $\mathbb{H} \in \Omega(T, \mathbb{D}_0)$ it holds that $\mathbb{H} \models T$.*

With the above Lemma, the proof of Lemma 3.3 (ii)⇒(i) is straightforward: the winning final position of Fugitive can serve as the counterexample $\mathbb{M}$ from Lemma 3.1.

The opposite direction, (i)⇒(ii) is not completely obvious. Notice that it could *a priori* happen that, while some counterexample exists, it is some terribly complicated structure which cannot be constructed as a final position in a play of the game of Escape. We should mention here that all the notions of Section 3 have their counterparts in [G15]. Instead of Regular Constrains however, in [G15] one finds conventional Tuple Generating Dependencies[7], and instead of the game of Escape one finds the conventional notion of Chase. But, while in [G15] the counterpart of Lemma 3.3 follows from the well-known fact that Chase is a universal structure, here we do not have such convenient tool available off-the-shelf, and we need to built our own.

Lemma 3.5. *Suppose structures $\mathbb{D}_0$ and $\mathbb{M}$ over $\bar{\Sigma}$ are such that there exists a homomorphism $h_0 : \mathbb{D}_0 \to \mathbb{M}$. Let T be a set of RCs and suppose $\mathbb{M} \models T$. Then from some final position $\mathbb{H} \in \Omega(T, \mathbb{D}_0)$ there exists a homomorphism $h : \mathbb{H} \to \mathbb{M}$ such that $h_0 \subset h$.*

Proof. First we need to prove:

Lemma 3.6. *For structures $\mathbb{D}_i$, $\mathbb{M}$ over $\bar{\Sigma}$, a homomorphism $h_i : \mathbb{D}_i \to \mathbb{M}$ and set of RCs T if $\mathbb{M} \models T$ then there exists some structure $\mathbb{D}_{i+1}$ such that $step(\mathbb{D}_i, \mathbb{D}_{i+1}, T)$ and there exists homomorphism $h_{i+1} : \mathbb{D}_{i+1} \to \mathbb{M}$ such that $h_i \subseteq h_{i+1}$.*

Proof. For $r = \langle x, y, X \to Y\rangle$ in $R_i = rq(T, \mathbb{D}_i)$ let $x' = h_i(x)$ and $y' = h_i(y)$. We know that $\mathbb{M} \models T$ so $\mathbb{M} \models Y(x', y')$ and thus for some $a_1 a_2 \dots a_n \in Y$ there is path $p' = a_1(x', x'_1)$,

[7]Notice that if all each of the languages in $\mathcal{Q}$ consists of a single word, then RCs degenerate into $TGDs$ and *Escape* degenerates into *Chase*.

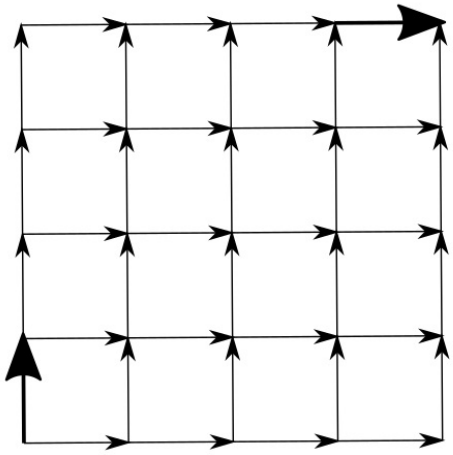

Figure 1. Our Grid.

$a_2(x'_1, x'_2) \dots a_n(x'_{n-1}, y')$ in $\mathbb{M}$. Let $\mathbb{D}'_i$ be a structure created by adding to $\mathbb{D}_i$ new path $p = a_1(x, x_1)$, $a_2(x_1, x_2), \dots a_n(x_{n-1}, y)$ (with x_i being new vertices). Let $h'_i = h_i \cup \{\langle x_i, x'_i\rangle | i \in [n-1]\}$. Now let $\mathbb{D}' = \bigcup_{r\in R_i} \mathbb{D}'_i$ and $h'_i = \bigcup_{r\in R_i} h'_i$. It is easy to see that $\mathbb{D}'_i$ and h'_i are requested $\mathbb{D}_{i+1}$ and h_{i+1}. □

To end the proof of Lemma 3.5 notice that if $\mathbb{D}_0, \mathbb{D}_1, \dots$ are as constructed by Lemma 3.6 then $\bigcup_{i=0}^{\infty} \mathbb{D}_i$ is equal to some final position from $\Omega(T, \mathbb{D}_0)$ and that $\bigcup_{i=0}^{\infty} h_i$ is required homomorphism h. □

Now we will prove the (i)⇒(ii) part of Lemma 3.3.

Let $\mathbb{M}$ be a counterexample from Lemma 3.1, a, b and $w \in Q_0$ such that $\mathbb{M} \models (G(w))(a, b)$ and $\mathbb{M} \not\models (R(Q_0))(a, b)$. Applying Lemma 3.5 to $\mathbb{D}_0 = G(w[a, b])$ and to $\mathbb{M}$ we know that there exists a final position $\mathbb{H}$ such that there is homomorphism from $\mathbb{H}$ to $\mathbb{M}$. It is clear that $\mathbb{H} \not\models (R(Q_0))(a, b)$ as we know that $\mathbb{M} \not\models (R(Q_0))(a, b)$. This shows that $\mathbb{H}$ is indeed a winning final position.

This concludes the proof of the Lemma 3.3.

4 The Reduction

Definition 4.1 (Our Grid Tiling Problem (OGTP)). Given a set of **shades** $\mathcal{S}$ (**black** $\in \mathcal{S}$) and a list $\mathcal{F} \subseteq \{V, H\} \times \mathcal{S} \times \{V, H\} \times \mathcal{S}$ of forbidden pairs $\langle a, b\rangle$ where $a, b \in \{V, H\} \times \mathcal{S}$ determine whether there exists a square grid $\mathbb{G}$ (a directed graph, as in Figure 1. but of any size) 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 vertical edge has the label (V, **black**);
(b2) upper-right horizontal edge has the label (H, **black**);
(b3) $\mathbb{G}$ contains no forbidden paths of length 2 labeled by $(a, b) \in \mathcal{F}$.

By standard argument one can show that:

Lemma 4.2. *Our Grid Tiling Problem is undecidable.*

Now we present a reduction from OGTP to the QDP-RPQ. Suppose an instance $\langle \mathcal{S}, \mathcal{F}\rangle$ of OGTP is given, we will construct an instance $\langle \mathcal{Q}, Q_0\rangle$ of QDP for RPQ.
The edge alphabet (signature) will be $\Sigma = \{\alpha, \beta, \omega\} \cup \Sigma_0$, where $\Sigma_0 = \{A, B\} \times \{H, V\} \times \{W, C\} \times \mathcal{S}$. We think of H and V as **directions** – *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 a value from $\{A, B\}$, shade, direction and **temperature** – have also **color**, red or green.

Notation 4.3. *We use the following notation for elements of* Σ_0*:*

$$({}_s\mathbf{p}^r_q) := (\mathbf{p}, q, r, s) \in \Sigma_0$$

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

Now we define $\mathcal{Q}$ and Q_0. Let $\mathcal{Q}_{good}$ be a set of 8 languages:

1. ω
2. $\alpha + \beta$
3. $(\mathbf{B}^W_H)(\mathbf{A}^W_V) + (\mathbf{B}^C_V)(\mathbf{A}^C_H)$
4. $(\mathbf{A}^C_H)(\mathbf{B}^C_V) + (\mathbf{A}^W_V)(\mathbf{B}^W_H)$
5. $(\mathbf{B}^C_V) + (\mathbf{B}^W_V)$
6. $(\mathbf{B}^W_H) + (\mathbf{B}^C_H)$
7. $(\mathbf{A}^W_V) + (\mathbf{A}^C_V)$
8. $(\mathbf{A}^C_H) + (\mathbf{A}^W_H)$

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

1. $\beta\Big(\bigoplus_{s\in\mathcal{S}\setminus\{black\}} ({}_s\mathbf{A}^W_V)\Big)\Sigma_0^\star\omega$
2. $\beta\Sigma_0^\star\Big(\bigoplus_{s\in\mathcal{S}\setminus\{black\}} ({}_s\mathbf{B}^W_H)\Big)\omega$
3. $\beta\Sigma_0^\star({}_a\bullet^W_d)({}_b\bullet^W_{d'})\Sigma_0^\star\omega$ for each forbidden $\langle(d, a), (d', b)\rangle \in \mathcal{F}$.

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

1. $\alpha\Sigma_0^\star(\bullet^W)\Sigma_0^\star\omega$
2. $\beta\Sigma_0^\star(\bullet^C)\Sigma_0^\star\omega$

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 already at this point the reader can notice that there is a fundamental difference between languages from $\mathcal{Q}_{good}$ and languages from $\mathcal{Q}_{bad} \cup \mathcal{Q}_{ugly}$. Languages from $\mathcal{Q}_{good}$ are all finite. The regular constraints $(Q^3_{good})^{\leftrightarrow}$ and $(Q^4_{good})^{\leftrightarrow}$ 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 $\langle e_1, e_2\rangle$ comes from some small finite set of possible choices. Satisfying requests generated by the remaining languages in $\mathcal{Q}_{good}$ do not even allow/require adding a new vertex y' – just one new edge is added.

On the other hand, each language in $\mathcal{Q}_{bad} \cup \mathcal{Q}_{ugly}$ contains infinitely many words – all words with some bad or ugly pattern. For $L \in \mathcal{Q}_{bad} \cup \mathcal{Q}_{ugly}$ requests generated by L are of the form "if you have any path in the current structure, green or red, between some verticies x and y, containing such pattern, then add any new path from x to y, of the opposite color, also containing the same pattern".

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

Finally, define $Q_{start} := \alpha[(\mathbf{A}_H^C)(\mathbf{B}_V^C)]^+\omega$, and let:

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

5 The structure of the proof of correctness

To end the proof of Theorem 1.1 we need to prove:

Lemma 5.1. *The following three conditions are equivalent:*

(i) An instance $\langle \mathcal{S}, \mathcal{F} \rangle$ of OGTP has no solution.
(ii) $\mathcal{Q}$ determines Q_0.
(iii) $\mathcal{Q}$ finitely determines Q_0.

The (ii) $\Rightarrow$ (iii) implication is obvious[8].

Next 4 pages will be devoted to the proof of the (i) $\Rightarrow$ (ii) implication. We will employ Lemma 3.3, showing that if the instance $\langle \mathcal{S}, \mathcal{F} \rangle$ has no solution then *Fugitive* does not have a winning strategy in the Escape($\mathcal{Q}$, Q_0). As we remember from Section 3.2, in such a game *Fugitive* will first choose, as the initial position of the game, a structure $w[a, b]$ for some $w \in G(Q_0)$. Then, in each step, he will identify all the requests present in the current structure and satisfy them. He will win if he will be able to play forever without satisfying the query $(R(Q_0))(a, b)$.

While analyzing the strategy of *Fugitive* we will use the words "must not" and "must" as shorthands for "or otherwise he will quickly lose the game".

Now our plan is first to notice that in his strategy *Fugitive* must obey the following principles:

(I) The structure resulting from his initial move must be $(G(w))[a, b]$ for some $w \in Q_{start}$.
(II) He must never allow any request generated by $\mathcal{Q}_{bad} \cup \mathcal{Q}_{ugly}$ to form in the current structure. Notice that if no such words ever occur in the structure then all the requests are generated by languages from $\mathcal{Q}_{good}$.

Then we will assume that *Fugitive's* play indeed follows the two principles and we will imagine us watching him playing, but watching in special glasses that make us insensitive to the shades from $\mathcal{S}$. Notice that, since the only requests *Fugitive* will satisfy, are from $\mathcal{Q}_{good}$, we will not miss anything – as the definitions of languages in $\mathcal{Q}_{good}$ are themselves shade-insensitive. In Section 9 we will prove that *Fugitive* must construct some particular structure, defined earlier in Section 7 and called $\mathbb{G}_m$, for some $m \in \mathbb{N}$. Then, in a short Section 10 we will take off our glasses and recall that the edges of $\mathbb{G}_m$ actually have shades. Assuming that the original instance of OGTP has no solution, we will get that $R(\mathcal{Q}_{bad})(a, b)$ holds in the constructed structure. This will end the proof of the (i)$\Rightarrow$(ii) direction. For the implication (¬i)$\Rightarrow$(¬ii) we will notice, again in Section 10 that if $\langle \mathcal{S}, \mathcal{F} \rangle$ has a solution, then one of the structures $\mathbb{G}_m$, with shades duly assigned to edges, forms a counterexample $\mathbb{M}$ as required by Lemma 3.1. Since this $\mathbb{M}$ will be finite, we will show that if the instance $\langle \mathcal{S}, \mathcal{F} \rangle$ of OGTP has a solution, then $\mathcal{Q}$ does not finitely determine Q_0 (which is a stronger statement than just saying that $\mathcal{Q}$ does not determine Q_0).

6 Principle I : $\mathbb{D}_0$

The rules of the game of Escape are such that *Fugitive* loses when he builds a path (from a to b) labeled 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 the player 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 $\mathcal{Q}_{bad}$ 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) labeled 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. But this problem is solvable, as we are going to show in this Section. And having the languages $\mathcal{Q}_{bad} \cup \mathcal{Q}_{ugly}$ also in Q_0 is part of the solution.

Assume that $\mathbb{H}$ is a final position of a play of the *Escape* game that started with $\mathbb{D}_0 = G(w)[a, b]$ for some $w \in Q_0$. This means, by Lemma 3.4, that $\mathbb{H} \models \mathcal{Q}^{\leftrightarrow}$. Recall that $\mathbb{H}$ is a structure over $\bar{\Sigma}$, which means that each edge of $\mathbb{H}$ is either red or green.

Observation 6.1. *For all $x, y \in \mathbb{H}$ if $\mathbb{H} \models G(L)(x, y)$ for some $L \in \mathcal{Q}_{ugly} \cup \mathcal{Q}_{bad}$ then $\mathbb{H} \models R(Q_0)(x, y)$.*

Proof. Notice that $G(L) \rightarrow R(L) \in \mathcal{Q}^{\rightarrow}$ so $\mathbb{H} \models R(L)(x, y)$ and as $L \subseteq Q_0$ it follows that $\mathbb{H} \models R(Q_0)(x, y)$. □

Lemma 6.2 (Principle I). *Fugitive must choose to start the Escape game from $\mathbb{D}_0 = G(q)[a, b]$ for $q \in Q_{start}$.*

Proof. If $q \in Q_0 \setminus Q_{start}$ then $\mathbb{D}_0 \models G(L)(a, b)$ for some $L \in \mathcal{Q}_{ugly} \cup \mathcal{Q}_{bad}$ and it follows from Observation 6.1. that *Fugitive* loses. □

[8] Notice that we are of course not going to prove that determinacy coincides with finite determinacy. It does not! But for the instances resulting from our reduction they indeed coincide.

7 The grid $\mathbb{G}_m$

Definition 7.1. $\mathbb{G}_m$, for $m \in \mathbb{N}$, is (see Fig. 2) a directed graph (V, E) where

$V = \{a, b\} \cup \{v_{i,j} : i, j \in [0, m]\}$ and where the edges from E are labeled with symbols α or β or ω or one of the symbols of the form $(\mathbf{p}_q^r)$, where – like before – $\mathbf{p} \in \{A, B\}$, $q \in \{H, V\}$ and $r \in \{W, C\}$. Each label has to also be either red or green (this gives us $(3 + 2^3)2$ possible labels, but only 12 of them will be used). 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 4.3. One should imagine that we watch *Fugitive's* play in shade filtering glasses.

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

- Vertex $v_{0,0}$ is a successor of a. Vertex b is a successor of $v_{m,m}$. The successors of $v_{i,j}$ are $v_{i+1,j}$ and $v_{i,j+1}$ (if they exist). Each node is connected to each of its successors with two edges, one green and one red.
- Each "Cold" edge, labeled with a symbol in $(\bullet^C)$, is green.
- Each "Warm" edge, labeled 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 $v_{i,j} \neq v_{m,m}$, with $i + j$ even, is from $(\mathbf{A})$, the label of each edge leaving $v_{i,j} \neq v_{m,m}$, with $i + j$ odd, is from $(\mathbf{B})$.
- Edges $(a, v_{0,0})$ with label $G(\alpha)$ and $(a, v_{0,0})$ with label $R(\beta)$ are in E.
- Edges $(v_{m,m}, b)$ with label $G(\omega)$ and $(v_{m,m}, b)$ with label $R(\omega)$ are in E.

8 Principle II

In this section we assume that the *Fugitive* obeys Principle I and he selects the initial structure $\mathbb{D}_0 = G(\alpha[(\mathbf{A}_H^C)(\mathbf{B}_V^C)]^m\omega)[a, b]$ for some m.

Lemma 8.1. *Suppose $\mathbb{H}$ is the final position of a play of the Escape game which started from $\mathbb{D}_0$.*

1. *Every edge $e \in \mathbb{H}$ labeled with $G(\alpha), R(\alpha), G(\beta)$ or $R(\beta)$ begins in a.*
2. *Every edge $e \in \mathbb{H}$ labeled with $G(\omega)$ or $R(\omega)$ ends in b.*

Proof. (1) By induction we show that the claim is true in every $\mathbb{D}_i$. It is clearly true in $\mathbb{D}_0$. For the induction step use the fact that for every language $L \in \mathcal{Q}$ and for each word $w \in L$ if w contains α or β then:
– this α or β is the first letter of w and
– all words in L begin from α or β.
(2) Analogous. □

[9]Please use a color printer if you can.

Lemma 8.2 (Principle II). *Fugitive must never allow any request generated by $\mathcal{Q}_{bad}$ and $\mathcal{Q}_{ugly}$ to form in the current structure.*

Proof. Let $\mathbb{D}$ be the current structure and $L \in \mathcal{Q}_{bad} \cup \mathcal{Q}_{ugly}$.

First assume that $\mathbb{D} \models R(L)(x, y)$ for some x, y. Notice that from Lemma 8.1 $x = a$ and $y = b$. Because of that $\mathbb{D} \models R(L)(a, b)$ which means that $\mathbb{D} \models R(Q_0)(a, b)$ and *Fugitive* loses.

Now assume that $\mathbb{D} \models G(L)(x, y)$ for some x, y. Similarly, from Lemma 8.1, $x = a$ and $y = b$. We have that $\langle a, b, L^{\rightarrow}\rangle \in rq(\mathcal{Q}^{\leftrightarrow}, \mathbb{D})$ so *Fugitive* must satisfy this request with $R(w)[a, b]$ for some $w \in L$ which loses, as $L \subseteq Q_0$. □

9 Now we do not see the shades

As we already said, now we are going to watch, and analyze, *Fugitive's* play in shade filtering glasses. We assume he obeys Principle I, otherwise he would lose. We also assume he obeys Principle II, but wearing our glasses we are not able to tell whether any word from $G(\mathcal{Q}_{bad}) \cup R(\mathcal{Q}_{bad})$ occurs in the current structure. For this reason we cannot use, in our analysis, arguments referring to languages in $\mathcal{Q}_{bad}$. We are however free to use arguments from Principle II, referring to languages in $\mathcal{Q}_{ugly}$.

Lemma 9.1. *Suppose in his initial move* Fugitive *selects $\mathbb{D}_0 = G(\alpha[(\mathbf{A}_H^C)(\mathbf{B}_V^C)]^m\omega)[a, b]$. Then the final position $\mathbb{H}$ must be equal (from the point of view of a shades-insensitive spectator) to $\mathbb{G}_m$.*

To prove Lemma 9.1 it is enough to show that:

Lemma 9.2. *Let $\mathbb{L}_i$ be like on Figure 3 and $\mathbb{L}_i^G$ and $\mathbb{L}_i^R$ be parts of $\mathbb{L}_i$ consisting of (resp.) green and red edges. Then:*

(i) $\mathbb{D}_0 = \mathbb{L}_0^G$,
(ii) $\mathbb{D}_{2i} = \mathbb{L}_{2i}^G \cup \mathbb{L}_{2i-1}$,
(iii) $\mathbb{D}_{2i+1} = \mathbb{L}_{2i+1}^R \cup \mathbb{L}_{2i}$.

Lemma 9.2 (i) is Principle I restated. Next subsections of this Section are devoted to the proof of Lemma 9.2 (ii) and (iii). This will be done by induction on i.

9.1 General rules for the Fugitive

Now assume $\mathbb{D}_0$ as demanded by Lemma 9.1 was really selected and denote vertices of this $\mathbb{D}_0$ by $a, x_1, \ldots, x_n, b$, with $n = 2m + 1$ (see Figure 3).

Lemma 9.3. *For every final position $\mathbb{H}$ that was built obeying Principles I and II:*

1. *Every edge $e \in \mathbb{H}$ labeled with $G(\alpha), R(\alpha), G(\beta)$ or $R(\beta)$ connects a and x_1.*
2. *Every edge $e \in \mathbb{H}$ labeled with $G(\omega)$ or $R(\omega)$ connects x_n and b.*

Proof. Notice that by Principle II there were no requests formed by either $\mathcal{Q}_{bad}$ or $\mathcal{Q}_{ugly}$ during the game that led to

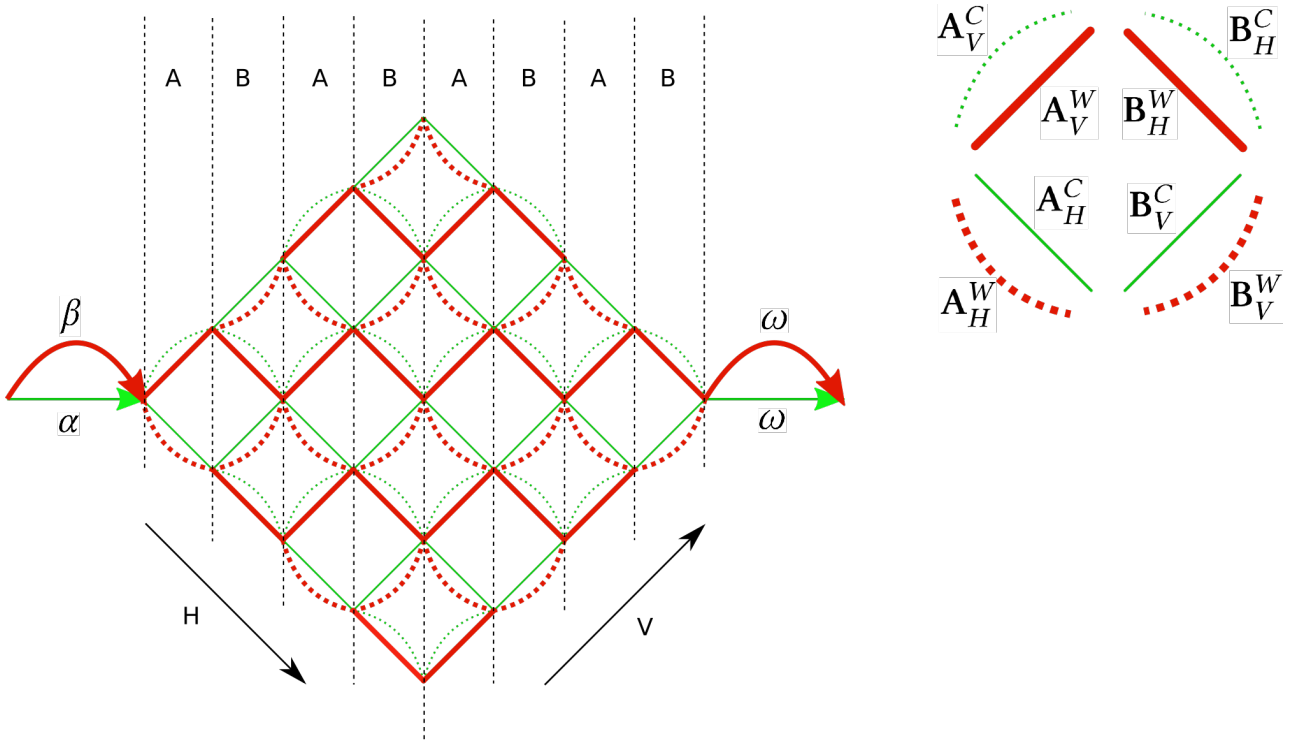

Figure 2. $\mathbb{G}_m$ with $m = 4$ (left). Smaller picture in the top-right corner explains how the different line styles on the main picture map to Σ_0.[9]

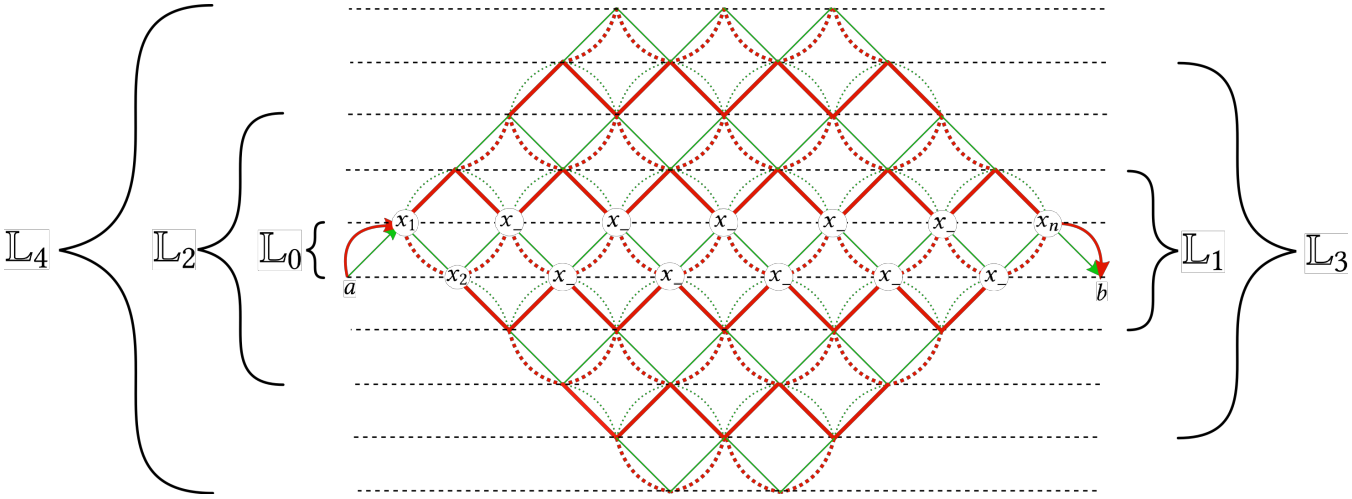

Figure 3. Five first Layers of $\mathbb{G}_m$ with $m = 6$.

$\mathbb{H}$. It means that all requests were generated by Q_{good}. But for every language $L \in Q_{good}$ for each $w \in L$ if w contains α, β or ω then w is a one letter word, and also all other words of this language contain one letter. So satisfying a request involving α, β or ω never requires creating new vertices. □

Lemma 9.4. *For each $y \in \mathbb{H}, y \neq a$ there exist, in $\mathbb{H}$:*

- *a red path from x_1 to y,*
- *a green path from x_1 to y,*

For each $y \in \mathbb{H}, y \neq b$ there exist, in $\mathbb{H}$:

- *a red path from y to x_n,*
- *a green path from y to x_n.*

Proof. Notice that for each $c \in \Sigma_0$ there exists a language $L \in Q_{good}$ such that $c \in L$. This means that for all $u, w \in \mathbb{H}$ such that these vertices are endpoints of a green edge $e =$

$(u, w, G(c)), c \in \Sigma_0$ there is also a red path connecting u and $w \in \mathbb{H}$ (this is since $\mathbb{H} \models Q^{\leftrightarrow}_{good}$).

Reasoning for red edges is analogous. □

In his first move *Fugitive* must satisfy all the requests in $S_0 = rq(Q^{\leftrightarrow}, \mathbb{D}_0)$. Notice that (since all the edges of $\mathbb{D}_0$ are green and there are no bad or ugly patterns in $\mathbb{D}_0$) all requests in S_0 are actually generated by RCs in $Q^{\rightarrow}_{good}$. And one of them is generated by $(Q^2_{good})^{\rightarrow}$. Next lemma does not look spectacular, but this is how we get our foot in the door:

Lemma 9.5. *Request $req = \langle a, x_1, (\alpha + \beta)^{\rightarrow}\rangle$ in S_0 must be satisfied with $R(\beta)[a, x_1]$.*

Proof. First notice that there are numerous requests in S_0 generated by Q^4_{good}, all of them of the form $\langle x_i, x_{i+2}, Q^{4\,\rightarrow}_{good}\rangle$. Each of them can potentially be satisfied in one of two ways: either by adding a new path labeled with a word $R((\mathbf{A}^W_V)(\mathbf{B}^W_H))$ from x_i, x_{i+2} or by adding a new path labeled with $R((\mathbf{A}^C_H)(\mathbf{B}^C_V))$.

Consider what would happen if *Fugitive* tried to satisfy req with $R(\alpha)$ instead of $R(\beta)$. First assume that there exists $req \in S_0$ generated by Q^4_{good} that is satisfied with $R((\mathbf{A}^W_V)(\mathbf{B}^W_H))$. Then $\mathbb{D}_1 \models R(Q^1_{ugly})(a, b)$ and this is forbidden by Principle II. So all requests in S_0 generated by Q^4_{good} must be satisfied with $R((\mathbf{A}^C_H)(\mathbf{B}^C_V))$. But then $\mathbb{D}_1 \models R(Q_{start})(a, b)$ and *Fugitive* loses. □

Now we know that, alongside the green α, there must exist the red β leading to x_1 (see Figure 2). From this we get that:

Lemma 9.6. *If $\mathbb{H}$ is a final position that was built obeying Principles I and II (which started with $\mathbb{D}_0$) then: for each edge $e \in \mathbb{H}$,*

1. *e is labeled with $c \in R(\Sigma_0) \Leftrightarrow c \in R(\bullet^W)$*
2. *e is labeled with $c \in G(\Sigma_0) \Leftrightarrow c \in G(\bullet^C)$*

Proof. (1) Assume by contradiction that there exists a red edge $e \in \mathbb{H}$, from some x to some x', labeled with $c \in R(\bullet^C)$. By Lemma 9.4 there is a path, consisting of edges from $R(\Sigma_0)$, from x_1 to x and another such path from x' to x_n. This implies that $\mathbb{H} \models Q^2_{ugly}(a, b)$ which is forbidden by Principle II. (2) Like (1) but then $\mathbb{H} \models Q^1_{ugly}(a, b)$. □

Notice that each Q^i_{good} for $i = 3 \ldots 8$ consists of two words (from the point of view of a shades-insensitive spectator). This sounds like good news for *Fugitive*: when satisfying requests generated by these languages he has some choice. But actually he does not, as the next lemma tells us:

Lemma 9.7. *Let $i \in \{3 \ldots 8\}$ and let $Q^i_{good} = \{w_i, w'_i\}$.*

1. *If $\mathbb{D}_j \models G(w_i)(x, y)$, for some j, and $\mathbb{D}_j \not\models R(Q^i_{good})(x, y)$ then $\langle x, y, Q^{i\,\rightarrow}_{good}\rangle \in rq(Q^{i\,\rightarrow}_{good}, \mathbb{D}_j)$ and the Fugitive must satisfy this request with $R(w'_i)[x, y]$.*
2. *If $\mathbb{D}_j \models R(w_i)(x, y)$, for some j, and $\mathbb{D}_j \not\models G(Q^i_{good})(x, y)$ then $\langle x, y, Q^{i\,\leftarrow}_{good}\rangle \in rq(Q^{i\,\leftarrow}_{good}, \mathbb{D}_j)$ and the Fugitive must satisfy this request with $G(w'_i)[x, y]$.*

Proof. (1) Let $i \in \{3, \ldots, 8\}$ and let j be such that $\mathbb{D}_j \models G(w_i)(x, y)$ and $\mathbb{D}_j \not\models R(Q^i_{good})(x, y)$. Assume by contradiction that *Fugitive* satisfies $\langle x, y, Q^{i\,\rightarrow}_{good}\rangle$ with $R(w_i)[x, y]$. Then $\mathbb{D}_{j+1} \models G(w_i)(x, y)$ and $\mathbb{D}_{j+1} \models R(w_i)(x, y)$. Let c be any letter of w_i (notice that $c \in \Sigma_0$). We have that there exist vertices $u, w, p, q \in \mathbb{D}_{j+1}$ such that $\mathbb{D}_{j+1} \models G(c)(u, w)$ and $\mathbb{D}_{j+1} \models R(c)(p, q)$ and this contradicts Lemma 9.6. (2) Analogous to the proof of (1). □

Now, in Section 9.2 we assume that $\mathbb{D}_{2i} = \mathbb{L}^G_{2i} \cup \mathbb{L}_{2i-1}$ and show that $\mathbb{D}_{2i+1}$ is as claimed in Lemma 9.2 (ii) and in Section 9.3 we assume that $\mathbb{D}_{2i+1} = \mathbb{L}^R_{2i+1} \cup \mathbb{L}_{2i}$ and show that $\mathbb{D}_{2i+2}$ is as claimed in Lemma 9.2 (iii).

9.2 Fugitive's move 2*i*: from $\mathbb{D}_{2i}$ to $\mathbb{D}_{2i+1}$

Observation 9.8. *For $\mathbb{D}_{2i}$ it is true that:*

(1) All requests in $\mathbb{D}_{2i}$ generated by Q^4_{good} must be satisfied with $R((\mathbf{A}^W_V)(\mathbf{B}^W_H))$.
(2) All requests in $\mathbb{D}_{2i}$ generated by Q^3_{good} must be satisfied with $R((\mathbf{B}^W_H)(\mathbf{A}^W_V))$.
(3) All requests in $\mathbb{D}_{2i}$ generated by Q^5_{good} must be satisfied with $R(\mathbf{B}^W_V)$.
(4) All requests in $\mathbb{D}_{2i}$ generated by Q^8_{good} must be satisfied with $R(\mathbf{A}^W_H)$.

Proof. For (1). By hypothesis all requests that are generated by Q^4_{good} in $\mathbb{D}_{2i}$ are of the form $\langle x, y, G((\mathbf{A}^C_H)(\mathbf{B}^C_V)) \rightarrow R(Q^4_{good})\rangle$ (Note that $(\mathbf{A}^C_H)(\mathbf{B}^C_V) \in Q^4_{good}$). By Lemma 9.7 *Fugitive* must satisfy all such requests with $R((\mathbf{A}^W_V)(\mathbf{B}^W_H))$. Rest of the proofs for (2)-(4) are analogous. □

9.3 Fugitive's move 2*i* + 1: from $\mathbb{D}_{2i+1}$ to $\mathbb{D}_{2i+2}$

Proof of the following Observation is analogous to the one of Observation 9.8.

Observation 9.9. *For $\mathbb{D}_{2i+1}$ it is true that:*

1. *All requests in $\mathbb{D}_{2i+1}$ generated by Q^4_{good} must be satisfied with $G((\mathbf{A}^C_H)(\mathbf{B}^C_V))$.*
2. *All requests in $\mathbb{D}_{2i+1}$ generated by Q^3_{good} must be satisfied with $G((\mathbf{B}^C_V)(\mathbf{A}^C_H))$.*
3. *All requests in $\mathbb{D}_{2i+1}$ generated by Q^7_{good} must be satisfied with $G(\mathbf{A}^C_V)$.*
4. *All requests in $\mathbb{D}_{2i+1}$ generated by Q^6_{good} must be satisfied with $G(\mathbf{B}^C_H)$.*

9.4 The end. No more requests!

Now it is straightforward to verify that:

Observation 9.10. *All requests generated by Q_{good} are already satisfied in $\mathbb{D}_{m+1} = \mathbb{G}_m$.*

10 And now we see the shades again

Now we can finish the proof of Lemma 5.1 (i) ⇒ (ii).

Suppose the Fugitive's play ended, in some final position $\mathbb{H} = \mathbb{G}_m$. We take off our glasses, and not only we still see this $\mathbb{H}$, but now we see it in full colors, with each edge (apart from edges labeled with α, β and ω) having one of the shades from $\mathcal{S}$. Assume that the original instance $\mathcal{S}, F$ of Our Grid Tiling Problem has no solution, and concentrate on the red edges of $\mathbb{H}$. They form a square grid, with each vertical edge labeled with V, each horizontal edge labeled with H, and with each edge labeled with a shade from $\mathcal{S}$. So clearly, one of the conditions (b1)-(b3) of Definition 4.1 is unsatisfied. But this implies that a path labeled with a word from one of the languages Q^1_{bad}– Q^3_{bad} occurs in $\mathbb{H}$, which is in breach of Principle II. This ends the proof of Lemma 5.1 (i)→ (ii).

For the proof Lemma 5.1 (¬i)→ (¬iii) assume the original instance $\langle \mathcal{S}, \mathcal{F} \rangle$ of Our Grid Tiling Problem has a solution – a labeled grid $m \times m$ for some m. Call this grid $\mathbb{G}$.

Recall that $\mathbb{G}_m$ is finite and it satisfies all regular constraints from $\mathcal{Q}^{\leftrightarrow}_{good}$ (Observation 9.10) and from $\mathcal{Q}^{\leftrightarrow}_{ugly}$ (for trivial reasons, as no paths from any $G(L) \cup R(L)$ with $L \in \mathcal{Q}_{ugly}$ occur in $\mathbb{G}_m$). 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 finite counterexample, as in Lemma 3.1.

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. Guagliardo *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;
[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;