O problemie determinacji zapytań dla języków regularnych

(About the query determinacy problem for regular languages)

Grzegorz Głuch

Praca magisterska

Promotor: prof. Jerzy Marcinkowski

Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Informatyki

21 sierpnia 2018

Streszczenie

Celem pracy jest prezentacja problemu determinacji zapytań, historii jego badania, a także prezentacja współuzyskanych przez autora wyników. Praca składa się z dwóch integralnych części. Pierwsza zawiera wprowadzenie do rozważanych zagadnień oraz jeden wynik o rozstrzygalności pewnego fragmentu problemu determinacji zapytań. Druga część to dwie załączone prace: "Can One Escape Red Chains? Regular Path Queries is Undecidable." oraz "The First Order Truth behind Undecidability of Regular Path Queries Determinacy.", w których udowodniona jest nierozstrzygalność dwóch wariantów problemu determinacji zapytań.

The aim of this work is to present a query determinacy problem, history of its research, as well as the presentation of results co-created by the author. The work consists of two integral parts. The first contains an introduction to the issues under consideration and one result on the decidability of a certain fragment of the query determinacy problem. The second part consists of two attached papers: "Can One Escape Red Chains? Regular Path Queries is Undecidable." and "The First Order Truth behind Undecidability of Regular Path Queries Determinacy.", in which the undecidability of two variants of the query determinacy problem is proved.

Spis treści

Rozdział 1.

Wprowadzenie

1.1. Wstęp

W ostatnich latach popularność zaczęły zyskiwać grafowe bazy danych. Są one używane coraz częściej przy np. modelowaniu sieci społecznościowych w serwisach takich jak Facebook, MySpace czy LinkedIn. W tego typu bazach dane modelowane są jako grafy, gdzie wierzchołki reprezentują obiekty, a etykietowane krawędzie definiują relacje między tymi obiektami.

Popularność grafowych baz danych motywuje rozważanie teoretycznych zagadnień, potencjalnie rozwiązanych już w tradycyjnym modelu relacyjnym, także w modelu grafowym. Właśnie jeden z takich problemów jest przedmiotem tej pracy magisterskiej.

Zajmiemy się tutaj **Problemem determinacji zapytań**. Wyobraźmy sobie bazę danych $\mathbb{D}$, do której nie mamy bezpośredniego dostępu. Mamy natomiast dostęp do zbioru widoków $\mathcal{Q} = \{Q_1, Q_2, \ldots Q_k\}$. Oprócz tego otrzymujemy zapytanie Q_0. Czy jesteśmy w stanie, niezależnie od $\mathbb{D}$, obliczyć Q_0 używając do tego jedynie widoków z $\mathcal{Q}$?

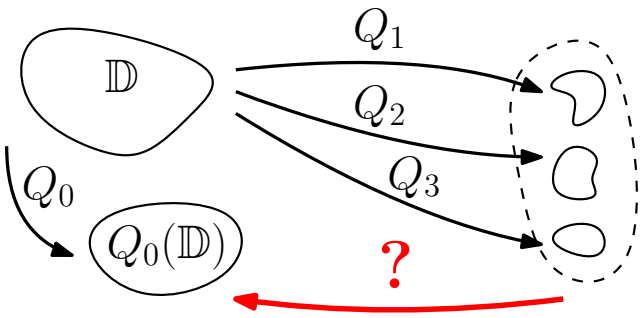
1.2. Cel pracy

Celem pracy jest prezentacja **Problemu determinacji zapytań**, omówienie historii jego badania, a także prezentacja wyników współuzyskanych przez autora dla modelu grafowych baz danych.

Rozdział 2.

Problem determinacji zapytań

2.1. Definicja problemu



Rysunek 2.1

Poniżej wprowadzamy formalną definicję problemu determinacji zapytań, która abstrahuje od konkretnego modelu baz danych.

Instancją problemu jest zbiór zapytań $\mathcal{Q} = \{Q_1, \ldots Q_k\}$, i dodatkowe zapytanie Q_0. Pytamy czy $\mathcal{Q}$ **determinuje** Q_0, co oznacza, że dla każdych dwóch struktur (instancji baz danych) $\mathbb{D}_1$ i $\mathbb{D}_2$, takich że $Q(\mathbb{D}_1) = Q(\mathbb{D}_2)$ dla każdego $Q \in \mathcal{Q}$, zachodzi również $Q_0(\mathbb{D}_1) = Q_0(\mathbb{D}_2)$.

Warto tu zauważyć, że instancją problemu są tu jedynie zapytania. Determinacja jest więc własnością samych zapytań, a nie baz danych. Intuicyjnie $\mathcal{Q}$ determinuje Q_0, gdy informacja zawarta w widokach jest wystarczająca do zdeterminowania wyniku zapytania Q_0.

2.2. Motywacja

Problem, formalnie zdefiniowany w poprzedniej sekcji, jest czysto teoretyczny. Został on jednak zidentyfikowany i wyabstrahowany na podstawie praktycznych za-

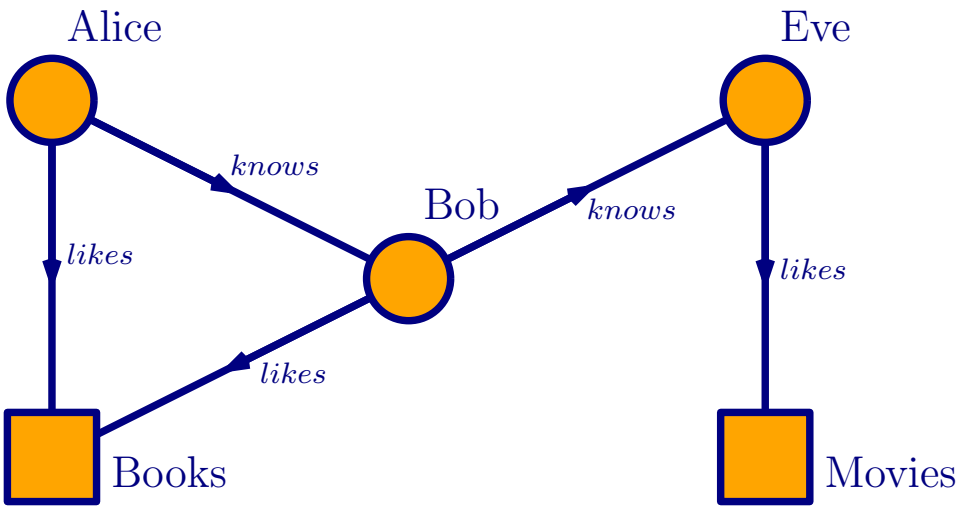
stosowań. Możemy tutaj myśleć o trzech aspektach: optymalizacji, integracji danych i prywatności.

Optymalizacja. W pierwszym przypadku wyobraźmy sobie, że z powodów wydajnościowych nie chcemy uzyskiwać dostępu do głównej bazy danych $\mathbb{D}$. Mamy natomiast efektywny dostęp do pewnego zbioru widoków tej bazy $\mathcal{V} = \{\mathbb{V}_1, \ldots \mathbb{V}_k\}$, które powstały jako rezultat aplikacji zapytań $\mathcal{Q} = \{Q_1, \ldots Q_k\}$ do $\mathbb{D}$, a dokładniej $\mathbb{V}_i := Q_i(\mathbb{D})$ dla każdego i. Możemy myśleć, że np. widoki znajdują się w pamięci podręcznej komputera, a $\mathbb{D}$ znajduje się na dysku i czas dostępu do $\mathcal{V}$ jest znacznie niższy do $\mathbb{D}$. Innym przykładem powyższej sytuacji są ogromne, rozproszone bazy danych, gdzie może się zdarzyć, że $\mathbb{D}$ przechowywane jest w częściach rozdzielonych po wielu centrach obliczeniowych, a widoki z $\mathcal{V}$ znajdują się w tej samej lokalizacji, z której chcemy wykonać zapytanie Q_0. W powyższych przypadkach chcielibyśmy, żeby $\mathcal{Q}$ determinowało Q_0, bo to może oznaczać, że uda nam się obliczyć $Q_0(\mathbb{D})$ jedynie na podstawie widoków $\mathcal{V}$.

Integracja danych. Integracja danych to problem polegający na połączeniu danych pochodzących z różnych źródeł i zapewnieniu użytkownikowi zunifikowanego dostępu do tych danych. Ideałem dla użytkownika byłby tu taki dostęp do danych, jakby pochodziły one z jednej bazy danych. W tym scenariuszu traktujemy bazowe bazy danych jako widoki $\mathcal{V}$ globalnej, wirtualnej bazy $\mathbb{D}$. W momencie gdy użytkownik chce wykonać zapytanie Q_0 do wirtualnej bazy $\mathbb{D}$ musimy zdecydować czy $\mathcal{V}$ determinuje Q_0 i jeśli tak to spróbować obliczyć wynik zapytania Q_0 bezpośrednio z widoków $\mathcal{V}$.

Prywatność. Trzeci aspekt, czyli prywatność, może się pojawić w następującej sytuacji. Potraktujmy informacje, jakie przechowuje o nas portal Facebook, jako bazę danych $\mathbb{D}$. Wśród tych danych znajdują się informacje o adresach, numerach telefonu, preferencjach, historii zakupów itd.; część z nich to informacje poufne. Przez $\mathcal{V} = \{\mathbb{V}_1, \ldots \mathbb{V}_k\}$ (gdzie $\mathbb{V}_i := Q_i(\mathbb{D})$) możemy rozumieć np. informacje, do jakich dostęp mają poszczególni użytkownicy, reklamodawcy czy pracownicy Facebook'a. Portal społecznościowy powinien zapewnić, żeby na podstawie widoków $\mathcal{V}$ nie dało się obliczyć wyników niektórych zapytań Q_0. Nie chcielibyśmy np. żeby użytkownicy niebędący naszymi przyjaciółmi mogli poznać nasze miejsce zamieszkania czy numer telefonu. Jeśli twórcy Facebook'a zadbają o to, żeby $\mathcal{Q}$ nie determinowało Q_0, dla Q_0 będących zapytaniami generującymi poufne dane, to zwiększą oni tym samym prywatność użytkowników.

2.3. Grafowe bazy danych



Rysunek 2.2

W grafowych bazach danych dane modelowane są jako graf. Model ten w naturalny sposób odpowiada danym takim jak sieci społecznościowe, struktura internetu czy dane biologiczne. Takie bazy danych są mniej ekspresywne niż standardowe relacyjne bazy, ale za to są one bardziej elastyczne. Grafowe bazy danych są ważnym tematem badań i zastosowań już od ponad 20 lat.

Definition 1 (**Grafowa baza danych**)**.** Instancją grafowej bazy danych nad alfabetem Σ jest graf $\mathbb{G} = \langle V, E \rangle$, gdzie wierzchołki V reprezentują obiekty, a etykietowane krawędzie $E \subset V \times V \times \Sigma$ definiują relacje między obiektami V. Σ to zbiór możliwych etykiet krawędzi, czyli równocześnie możliwych relacji między obiektami.

I w teorii i w praktyce w grafowych bazach danych używane są różne rodzaje zapytań. Jednym z bardziej popularnych języków zapytań jest język RPQs (ang. Regular-Path-Queries) i to właśnie na tym języku zapytań skupimy się w tej pracy. RPQs to po prostu wyrażenia regularne nad alfabetem etykiet grafu. Wyrażenie takie zwraca pary wierzchołków połączone ścieżką etykietowaną słowem z języka definiowanego tym wyrażeniem. Jak tłumaczy [V16] RPQs to dobry język zapytań dla grafowych baz danych, bo ma rozstrzygalny problem zawierania zapytań, a z drugiej strony pozwala na łatwą nawigację po grafie. Pozwala on między innymi na podążanie sekwencją krawędzi, której długość nie jest podana explicite. Jest to możliwe dzięki rekursji zapewnionej przez wyrażenia regularne.

Definition 2 (**Zapytanie**)**.** Zapytanie do grafowej bazy danych $\mathbb{G} = \langle V, E \rangle$ nad Σ to wyrażenie regularne Reg nad alfabetem Σ, które definiuje język regularny L_{Reg}.

Definition 3 (**Wynik zapytania**)**.** Wynik zapytania Reg na bazie danych $\mathbb{G} = \langle V, E \rangle$ to zbiór par wierzchołków z V połączonych słowem z L_{Reg}. Formalnie $Reg(\mathbb{G}) = \{(u, v) \in V \times V : \exists\ w\ \in L_{Reg}\ \exists$ ścieżka w $\mathbb{G}$ z u do v etykietowana przez $w\}$.

Przykład 1. Rozważmy grafową bazę danych widoczną na Rysunek 2.2. Załóżmy, że chcemy się dowiedzieć jakie rzeczy lubią nasi przyjaciele, przyjaciele naszych przyjaciół itd. W tym celu wykonujemy zapytanie $knows^{+}likes$ i otrzymujemy w tym przypadku odpowiedź: $\{(Alice, Books), (Alice, Movies), (Bob, Movies)\}$.

2.4. Przykłady

W tej sekcji podajemy listę przykładów pozwalających lepiej zrozumieć problem determinacji.

- $Q_0 \in \mathcal{Q} \implies \mathcal{Q}$ determinuje Q_0.

  Dowód. Załóżmy, że $Q(\mathbb{D}_1) = Q(\mathbb{D}_2)$ dla każdego $Q \in \mathcal{Q}$. Mamy wtedy w szczególności, że $Q_0 \in \mathcal{Q}$, więc $Q_0(\mathbb{D}_1) = Q_0(\mathbb{D}_2)$. □

- $\mathcal{Q} = \{ab, bc\}$ nie determinuje $Q_0 = abc$.

  Dowód. Rozważmy bazę danych $\mathbb{D}_1$ złożoną ze ścieżki $x_0 \xrightarrow{a} x_1 \xrightarrow{b} x_2 \xrightarrow{c} x_3$ i bazę $\mathbb{D}_2$ złożoną z dwóch rozłącznych ścieżek $x_0 \xrightarrow{a} u \xrightarrow{b} x_2$, $x_1 \xrightarrow{b} v \xrightarrow{c} x_3$. Wtedy $(ab)(\mathbb{D}_1) = \{(x_0, x_2)\} = (ab)(\mathbb{D}_2)$ i $(bc)(\mathbb{D}_1) = \{(x_1, x_3)\} = (bc)(\mathbb{D}_2)$, ale $(abc)(\mathbb{D}_1) = \{(x_0, x_3)\} \neq \varnothing = (abc)(\mathbb{D}_2)$. □

- $\mathcal{Q} = \{a^*, b^*\}$ determinuje $Q_0 = a^*b^*$.

  Dowód. $(a^*b^*)(\mathbb{D})$ zwraca pary wierzchołków połączone ścieżką etykietowaną słowem z języka a^*b^*. Każdą taką ścieżkę możemy podzielić na część etykietowaną a^* i na część etykietowaną b^*. Mając więc wyniki $(a^*)(\mathbb{D})$ i $(b^*)(\mathbb{D})$ obliczamy $(a^*b^*)(\mathbb{D})$ jako $\{(x, z) : \exists\ y\ (x, y) \in (a^*)(\mathbb{D}),\ (y, z) \in (b^*)(\mathbb{D})\}$. □

2.5. Uzyskane wyniki

Wyniki współuzyskane przez autora umieszczone są w dwóch pracach: "Can One Escape Red Chains? Regular Path Queries is Undecidable." (Załącznik 1, [GMO18]) i "The First Order Truth behind Undecidability of Regular Path Queries Determinacy." (Załącznik 2, [GMO18a]) oraz w Rozdziale 3.

W [GMO18] pokazujemy, że Problem Determinacji Zapytań dla RPQs (Regular Path Queries) jest nierozstrzygalny. Zamyka to tym samym problem postawiony ponad 15 lat temu w [CGLV02]. W [GMO18a], budując na technikach zaprezentowanych w pierwszej pracy, udowadniamy nierozstrzygalność dla skończonych RPQs. Dokładniej pokazujemy, że gdy instancje Problemu Determinacji Zapytań

$\mathcal{Q} = \{Q_1, \ldots Q_k\}$ i Q_0 definiowane są wyrażeniami regularnymi definiującymi skończone języki, to problem dalej jest nierozstrzygalny. Natomiast w Rozdziale 3. pokazujemy, że Problem Determinacji Zapytań jest rozstrzygalny gdy wszystkie wyrażenia z $\mathcal{Q} = \{Q_1, \ldots Q_k\}$ definiują języki jednosłowowe.

2.6. Wkład autorów

W tej sekcji spróbujemy opisać wkład każdego z autorów w uzyskane wyniki. Najłatwiej byłoby przypisać autorom lematy, dowody, czy definicje, których są twórcami. Niestety jest to niemożliwe, gdyż poszczególne fragmenty zawarte w pracach są finalnym produktem miesięcy prób i błędów. Dowody i notacje przechodziły wiele zmian, więc nie sposób przypisać ich jednoznacznie któremuś z autorów. Co więcej, w czasie badań powstały wyniki, które pozwoliły na lepsze zrozumienie problemu, a nie zostały umieszczone w końcowych pracach. Wartość zawarta w tych pośrednich rezultatach nie powinna być pominięta przy opisie wkładu każdego z autorów. Biorąc to wszystko pod uwagę spróbujemy tu opisać historię badań wraz ze wskazaniem ważniejszych momentów i przełomów.

W celu ułatwienia odnoszenia się do poszczególnych autorów zastosujemy konwencję, gdzie piszemy:

- "J.", gdy mówimy o prof. Jerzym Marcinkowskim,
- "P.", gdy mówimy o Panu Piotrze Ostropolskim-Nalewaji,
- "G.", gdy mówimy o Grzegorzu Głuchu.

Badania rozpoczęliśmy od analizy prac o tematyce związanej z problemem determinacji dla zapytań regularnych. Na początku zaznajomiliśmy się z [A11], gdzie rozstrzygalność została udowodniona, gdy wszystkie języki z $\mathcal{Q}$ oraz język Q_0 są jednosłowowe, oraz z [GM15] gdzie nierozstrzygalność została pokazana dla zapytań koniunkcyjnych. Już od tego momentu zaadoptowaliśmy ideę *Red-Green Chase*'a z pracy [GM15], która pozwala sprowadzić problem determinacji do badania ewolucji pewnej czerwono-zielonej struktury. Ewolucja tej struktury może być rozumiana jako pewna gra dwuosobowa (nazwana *Escape*), grana pomiędzy *Fugitive*'em, a *Crocodile*'em, w której trakcie w wyniku ruchów graczy budowana jest stopniowo pewna baza danych. *Fugitive* dąży do zbudowania bazy danych będącej kontrprzykładem na determinację, a *Crocodile* próbuje przeszkodzić *Fugitive*'owi. Rozumowanie w terminach tej gry było obecne w rozważaniach aż do końca.

Pierwszą strategię walki z problemem zaproponował G. Zauważmy, że każde wyrażenie regularne złożone jest z liter z pewnego alfabetu Σ oraz operatorów $^*, +, \bullet$ (gdzie $\bullet$ to konkatenacja). Dopuszczając jedynie pewny podzbiór operatorów można

zmniejszyć ekspresywność wyrażeń. Np. wyrażenia używające jedynie operatora konkatenacji definiują dokładnie języki jednosłowowe. Zaproponowane podejście polegało na rozwiązywaniu następujących problemów: Czy dla podzbiorów operatorów $A, B \subset \{^*, +, \bullet\}$ rozstrzygalny jest problem determinacji, gdy języki z $\mathcal{Q}$ używają jedynie operatorów z A, a język Q_0 używa jedynie operatorów z B? Zauważmy, że gdy A oraz B zawierają wszystkie operatory to otrzymujemy dokładnie problem determinacji. Zauważmy też, że gdy $A = B = \{\bullet\}$ to otrzymujemy dokładnie problem rozważany w [A11], który wiemy, że jest rozstrzygalny. Manipulując więc zbiorami A i B jesteśmy w stanie przechodzić od problemów prostych, przez trudniejsze, aż do docelowego problemu determinacji.

Strategia ta zaowocowała umiarkowanym sukcesem. Dosyć łatwo udowodniona została rozstrzygalność dla szeregu konfiguracji zbiorów A i B, np. gdy $A = \varnothing$ lub $B = \{^*\}$. Najważniejszym wynikiem w tej fazie było udowodnienie przez G. rozstrzygalności, gdy $A = \{\bullet\}$ i $B = \{^*, +, \bullet\}$, co uogólnia wynik z [A11]. Dowód tego rezultatu znajduje się w Rozdziale 3. Trudności z dowodzeniem rozstrzygalności zaczęły się pojawiać już, gdy w A znajdował się więcej niż jeden operator. Co ciekawe owe trudności w pewnym stopniu zostały wyjaśnione przez pracę "The First Order Truth behind Undecidability of Regular Path Queries Determinacy.", gdzie pokazujemy, że problem determinacji jest już nierozstrzygalny, gdy $A = B = \{+, \bullet\}$ (czyli gdy języki są skończone).

Kolejnym etapem badań było rozważanie sytuacji gdzie $A = \{^*, +, \bullet\}$ i $B = \{\bullet\}$, czyli gdy Q_0 jest jednym słowem, a na języki z $\mathcal{Q}$ nie ma nałożonych żadnych restrykcji. Teraz wierzymy, że ten problem jest nierozstrzygalny, ale w tamtym czasie mieliśmy nadzieję na rozstrzygalność. Pierwszym osiągnięciem była tu redukcja problemu do $A = \{+, \bullet\}, B = \{\bullet\}$, czyli eliminacja nieskończonych języków z $\mathcal{Q}$. Dalej udowodniliśmy rozstrzygalność, gdy języki z $\mathcal{Q}$ były skończone i każde słowo z języka miało tę samą długość. Wynik ten otrzymaliśmy przy pomocy idei "czarnych kolumn" stworzonej przez J. już w [GM15]. Ważnym wydarzeniem na tym etapie było udowodnienie przez P. NP-trudności problemu, gdy $A = \{+, \bullet\}, B = \{\bullet\}$. Jednak nie sama NP-trudność była tu ważna, a fakt, że udało się zakodować 3-kolorowanie grafu w ewolucji *Red-Green Chase*'a w nietrywialny sposób. Ten rezultat pokazał nam jak wymusić pewien nietrywialny, kontrolowany mechanizm w problemie determinacji. Analiza tego wyniku była jednak niezwykle skomplikowana i narzędzia, które do tamtej pory stworzyliśmy, nie były wystarczające do poprawy rezultatu.

Nasze doświadczenia do tamtej pory mówiły nam, że obecność słów różnej długości w językach z $\mathcal{Q}$ jest kluczem do rozwiązania problemu. W celu zbadania tego fenomenu zaczęliśmy studiować pracę [F15], gdzie autor pokazuje "przybliżoną" determinację, gdy $A = \{+, \bullet\}$ i $B = \{\bullet\}$ i gdy alfabet jest jednoliterowy. "Przybliżona" determinacja oznacza, że podany w tej pracy algorytm jest w stanie rozstrzygać determinację jedynie, gdy długość słowa Q_0 jest odpowiednio duża w stosunku do słów z $\mathcal{Q}$. Niestety nie udało nam się dokładnie zrozumieć narzędzi użytych w tej pracy. Co więcej, zaczęła rosnąć nasza wiara w to, że problem determinacji jest nierozstrzy-

galny i byliśmy przekonani, że techniki z [F15] nie pomogą nam w jej udowodnieniu. Z powyższych powodów porzuciliśmy ten kierunek poszukiwań.

W tym momencie naszych badań, mniej więcej w tym samym czasie, nastąpiły dwa przełomy. Pierwszy z nich to hipoteza, a tak naprawdę pytanie postawione przez G. na temat tzw. "sprężynek", a drugi to narzędzie wymyślone przez P. do kontroli ewolucji czerwono-zielonej struktury. Pierwsze spostrzeżenie dotyczy pozornie prostej sytuacji, gdy w $\mathcal{Q}$ znajdują się 3 języki takie jak np. $\{ab, b, bc\}$, a słowo abc jest podsłowem pewnego słowa z Q_0. Powyższa konfiguracja prowadzi, niezależnie od ruchów *Fugitive*'a, do pewnej nieskończonej struktury, którą nazwaliśmy "sprężynką". Przez pewien czas byliśmy przekonani, że ta nieskończona struktura nie wnosi nic do problemu, a jest jedynie artefaktem wynikającym ze struktury gry. Jednak po paru nieudanych próbach udowodnienia tego faktu okazało się, że owe "sprężynki" mogą być kluczem do problemu determinacji. Stało się tak, ponieważ P. odkrył pewien mechanizm pozwalający na zabranianie wybranych podstruktur w bazie danych tworzonych przez graczy. Ten mechanizm zaaplikowany do "sprężynki" pozwolił na symulowanie działania deterministycznego skończonego automatu w grze *Escape*. To odkrycie pokazało nam jak kodować pewne obliczenia w ewolucji gry. Na tym etapie były to jedynie obliczenia DFA, ale idee zawarte w stworzonych narzędziach dawały nadzieje na więcej.

W tym momencie byliśmy już mocno przekonani co do nierozstrzygalności problemu determinacji i zaczęliśmy próby tworzenia redukcji z nierozstrzygalnych problemów. Pierwszym celem było wymuszenie w grze *Escape* powstania pewnej dwuwymiarowej struktury, która miałaby służyć jako "plansza", na której odbywa się obliczenie. Można myśleć, że na tej "planszy" mogłaby być zapisana cała historia obliczeń pewnej maszyny Turinga. Pierwszą próbą stworzenia takiej dwuwymiarowej struktury było zastosowanie wielu "sprężynek" obok siebie, które razem tworzyłyby pewną dużą nieskończoną (potencjalnie dwuwymiarową) strukturę. Podejście to, na początku wydawało się bardzo obiecujące, ale zawierało szereg problemów. Po pierwsze powstała struktura była nieskończona, co znacząco utrudniało analizę. Po drugie, po dokładniejszych badaniach okazało się, że owa struktura nie ma wcale natury czysto dwuwymiarowej, a wręcz przeciwnie jest niezwykle skomplikowaną siatką.

Na tym etapie z pomocą przyszedł J., który wyabstrahował esencję zalet "sprężynek" i zaproponował metodę na stworzenie bardzo prostej siatki dwuwymiarowej. Polegała ona na wybraniu języka Q_0 jako $(ab)^+$ i dodaniu do $\mathcal{Q}$ dwóch języków $\{ab, ba\}$. Taka konfiguracja języków spowodowała, że zależnie od pierwszego ruchu *Fugitive*'a (który można utożsamić z wybraniem liczby $k \in \mathbb{N}$) w grze tworzona była dokładnie kwadratowa siatka wymiaru $k \times k$. Ta siatka stała się "planszą", której poszukiwaliśmy.

Teraz potrzebowaliśmy już tylko zakodować w owej siatce jakiś problem nierozstrzygalny. Na potrzeby tego opracowania możemy myśleć, że jest to problem stopu

dla maszyn Turinga. W tym celu zastosowaliśmy mechanizm zabraniania podstruktur stworzony przez P.. Do końca dowodu potrzeba było jeszcze parę technicznych narzędzi i spostrzeżeń. Trudno byłoby je tu dokładnie opisać bez wprowadzania wielu pojęć, dlatego ograniczymy się jedynie do wymienia osiągnięć i ich autorów.

- J. zaproponował mechanizm, który umożliwił rozróżnianie pomiędzy wymiarami siatki,
- G. zauważył jak znacząco uprościć analizę,
- P. zaprojektował bardzo ważną metodę wymuszania pierwszego ruchu *Fugitive*'a.

W tym momencie twierdzenie o nierozstrzygalności problemu determinacji dla języków regularnych zostało już udowodnione.

Dalej zadaliśmy sobie pytanie, czy problem pozostaje nierozstrzygalny, gdy ograniczymy się do języków skończonych (czyli gdy $A = \{+, \bullet\}$ i $B = \{+, \bullet\}$). Zadziwiające jest, że dzięki nabytym doświadczeniom zdołaliśmy udowodnić nierozstrzygalność tego problemu jedynie w tydzień. Rozwiązując tę wersję problemu, chcieliśmy oczywiście zachować jak najwięcej pomysłów i technik z ogólnego przypadku. Jednak w porównaniu do poprzedniej pracy musieliśmy tu rozwiązać dwa nowe problemy wynikające z tego, że możemy używać tylko języków skończonych. Po pierwsze, gdy języki są skończone. to mechanizm stworzony przez P. jest w stanie zabraniać jedynie skończonych podstruktur. Do udowodnienia nierozstrzygalności potrzebujemy jednak móc tworzyć struktury o dowolnie dużym rozmiarze, pojawia się więc problem, jak kontrolować ich ewolucję. Po drugie, język $Q_0 = (ab)^+$, który w pewnej formie przetrwał w finalnej pracy, odpowiada wyborowi przez *Fugitive*'a rozmiaru siatki. Nieskończoność tego języka jest tu kluczowa, bo każde słowo z języka odpowiada jednemu rozmiarowi siatki, a chcemy przecież pozwolić na tworzenia siatek dowolnego rozmiaru.

Ideę rozwiązania pierwszego problemu zaproponował J.. Pomysł polegał na tym, że struktura, która będzie powstawać w ciągu gry, będzie mogła być dowolnie duża, ale odległości wszystkich wierzchołków do dwóch wyróżnionych wierzchołków a i b będą ograniczone przez stałą. Zostało to zrealizowane za pomocą tzw. "sięgaczy", czyli krawędzi, które łączą a i b ze wszystkimi innymi wierzchołkami. Co ciekawe narzędzie bardzo podobne do technik użytych w "sięgaczach" pozwoliło rozwiązać też drugi problem. To narzędzie to konstrukcja języków w $\mathcal{Q}$ i Q_0, która wymuszała w grze stworzenie nieskończonej dwuwymiarowej siatki. Połączenie tych dwóch technik dało juz poszukiwaną nierozstrzygalność dla skończonych języków. Co ciekawe redukcji dokonywaliśmy w tym momencie z problemu *Mortality* dla maszyn Turinga, problemu bardzo rzadko używanego w podobnego typu redukcjach. Później znaleźlismy jednak sposób jak uprościć rozumowanie i pozwolić *Fugitive*'owi na wybór rozmiaru siatki, a nie zmuszać go do stworzenia nieskończonej struktury. To uproszczenie i pomysł J. na użycie idei rekursywnej nieseparowalności pewnych podzbiorów

maszyn Turinga złożyły się na wersję dowodu, którą umieściliśmy w pracy "The First Order Truth behind Undecidability of Regular Path Queries Determinacy.".

Chapter 3.

Positive result

In this chapter we are going to prove the following theorem:

Theorem 4. *Query Determinacy Problem for Regular Path Queries is decidable when each $Q_i \in \mathcal{Q}$ is a one word language and Q_0 is a regular language.*

This theorem is a generalization of a result from [A11]:

Theorem 5. *Query Determinacy Problem for Regular Path Queries is decidable when each $Q_i \in \mathcal{Q}$ and Q_0 is a one word language.*

3.1. 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 $\mathbb{D}$ and $\mathbb{G}$ are used to denote 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: $Q(\mathbb{D}) = \{\langle x, y\rangle \,|\, \mathbb{D} \vDash Q(x, y)\}$.

Definition 6 (Our2NFA-ϵ)**.** Our two-way nondeterministic finite automaton with ϵ-moves is a 5-tuple $M = (S, \Sigma, \delta, a, r)$ where:

- S is a set of states,
- Σ is the alphabet,
- $\delta : S \times (\Sigma \cup \{\epsilon\}) \to 2^{S \times \{left, stay, right\}}$ It differs from a typical 2NFA in that after reading a letter we can move to the left letter, the right letter or **stay** at the same letter. It doesn't change the expressivity of these automata but simplifies construction in Section 3.3.
- $a \in S$ is the start state and the only accepting state,
- $r \in S$ is the reject state.

Let $w = a_1 a_2 \dots a_n$ be a word. Automaton starts reading the input w with the head over a_1 and then proceeds according to the transition function δ. M accepts w if and only if there exists an execution such that after reading a_n for the first time M is in state a and during this execution the head never moves to the left of the first letter a_1 (that is during this execution the head always stays within the range of the input $a_1 a_2 \dots a_n$).

Observation 1. As Our2NFA-ϵ automata are only slightly modified versions of general 2NFA-ϵ automata all languages recognized by Our2NFA-ϵ automata are regular.

3.2. Characterization of determiniacy

In this section, in the spirit of ideas from [A11], we will characterize determinacy in terms of connectivity of some graphs and inclusion of some languages. This characterization is what will be used for proving determinacy, as it will be shown in Section 3.4.

Definition 7. For a set of words $\mathcal{A}$ and a word $w = a_1a_2\dots a_n$ let $G_w^{\mathcal{A}} = \langle V, E\rangle$ be an undirected graph such that:

- $V = \{0, 1, 2, \dots, n\}$,
- $(i, j) \in E \Leftrightarrow w[i+1, j] \in \mathcal{A}$, where $w[i, j] = a_ia_{i+1}\dots a_j$ for $1 \le i \le j \le n$.

We say that $G_w^{\mathcal{A}}$ is **connected** if vertices 0 and n belong to the same connected component.

The following Lemma is a result from [A11] that charactarizes determinacy in terms of a graph $G_{Q_0}^{\mathcal{Q}}$.

Lemma 8. *When all $Q_i \in \mathcal{Q}$ and Q_0 are one word languages then $\mathcal{Q}$ determines Q_0 if and only if $G_{Q_0}^{\mathcal{Q}}$ is connected.*

Definition 9. For a set of words $\mathcal{A}$ over Σ we define a language $L^{\mathcal{A}} = \{w \in \Sigma^* : G_w^{\mathcal{A}}$ is connected$\}$.

Lemma 10. *If $\mathcal{Q}$ determines w for each $w \in Q_0$, for regular language Q_0 then $\mathcal{Q}$ determines Q_0.*

Proof. Let $\mathbb{D}_1$ and $\mathbb{D}_2$ be such that $\mathcal{Q}(\mathbb{D}_1) = \mathcal{Q}(\mathbb{D}_2)$. Then $Q_0(\mathbb{D}_1) = \{\langle x, y\rangle \mid \mathbb{D}_1 \vDash Q_0(x, y)\} = \bigcup_{w\in Q_0}\{\langle x, y\rangle \mid \mathbb{D}_1 \vDash w(x, y)\} = \bigcup_{w\in Q_0}\{\langle x, y\rangle \mid \mathbb{D}_2 \vDash w(x, y)\} = \{\langle x, y\rangle \mid \mathbb{D}_2 \vDash Q_0(x, y)\} = Q_0(\mathbb{D}_2)$ □

The following Lemma is in the spirit of Lemma 8. It characterizes determinacy in terms of the language $L^{\mathcal{Q}}$.

Lemma 11. *When all $Q_i \in \mathcal{Q}$ are one word languages and Q_0 is a regular language then $\mathcal{Q}$ determines Q_0 if and only if $Q_0 \subset L^{\mathcal{Q}}$.*

Proof. For the "if" direction we will use Lemma 10 and show that $\mathcal{Q}$ determines w for each $w \in L^{\mathcal{Q}}$. Let $w \in L^{\mathcal{Q}}$, in particular it means that $G_w^{\mathcal{Q}}$ is connected which by Lemma 8 proves that $\mathcal{Q}$ determines Q_0.

The "only if" direction is proved by contraposition. This part of the proof is a direct adaptation of a proof of Lemma 8 from [A11]. It is essentially the same proof but the argument needs to be repeated and checked in our setting as Q_0 is a regular language here (not necessarily a single word, as it was in [A11]).

Let $w = a_1a_2\dots a_n \in Q_0 \setminus L^{\mathcal{Q}}$. We will construct two databases $\mathbb{D}'$ and $\mathbb{D}''$ such that $\mathcal{Q}(\mathbb{D}') = \mathcal{Q}(\mathbb{D}'')$ and $Q_0(\mathbb{D}') \neq Q_0(\mathbb{D}'')$. We will first construct four databases $\mathbb{D}_0, \mathbb{D}_1, \mathbb{D}_2$ and $\mathbb{D}_3$, each of which isomorphic to $\mathbb{D}_0$. Then we will set $\mathbb{D}'$ as a union of $\mathbb{D}_0$ and $\mathbb{D}_1$ and $\mathbb{D}''$ as a union of $\mathbb{D}_2$ and $\mathbb{D}_3$.

$\mathbb{D}_0$ is the canonical database of query w with constants $c_0, c_1, \ldots, c_n$. $\mathbb{D}_1$ is a copy of $\mathbb{D}_0$ where each constant is replaced by fresh constant. For ease of reference each constant c is replaced by c'.

We construct $\mathbb{D}_2$ as follows. Let G' be a connected component of $G_w^{\mathcal{Q}}$ that contains vertex 0. Notice that by definition of w G' doesn't contain vertex n. Let h be a natural bijection from $G_w^{\mathcal{Q}}$ to variables of $\mathbb{D}_0$, which maps i to c_i for each vertex i of $G_w^{\mathcal{Q}}$. Now let $\mathbb{D}_2$ be a copy of $\mathbb{D}_0$ where each constant in an image $h[G']$ is exchanged for its primed version. Let $\mathbb{D}_3$ be a copy of $\mathbb{D}_2$ where we changed the primed constants for their non-primed version and the non-primed constants to their primed version.

Now we need to convince ourselves that $\mathcal{Q}(\mathbb{D}') = \mathcal{Q}(\mathbb{D}'')$. First notice that $\mathcal{Q}(\mathbb{D}_0)$, $\mathcal{Q}(\mathbb{D}_1)$, $\mathcal{Q}(\mathbb{D}_2)$ and $\mathcal{Q}(\mathbb{D}_3)$ are pairwise isomorphic. Moreover $\mathcal{Q}(\mathbb{D}_0)$ and $\mathcal{Q}(\mathbb{D}_1)$ are disjoint, so $\mathcal{Q}(\mathbb{D}') = \mathcal{Q}(\mathbb{D}_0) \cup \mathcal{Q}(\mathbb{D}_1)$. Also $\mathcal{Q}(\mathbb{D}'') = \mathcal{Q}(\mathbb{D}_2) \cup \mathcal{Q}(\mathbb{D}_3)$. To finish the claim we need to show that neither $\mathbb{D}_2$ nor $\mathbb{D}_3$ contains any facts that use one primed and one non-primed constant. Assume that $\langle c_i, c'_j \rangle \in \mathcal{Q}(\mathbb{D}_3)$ (other cases work similarly). Then G' is not a connected component of vertex 0 as there is an edge between vertices i and j in $G_w^{\mathcal{Q}}$ and we can add vertex j to G'.

To finish the proof we need to show that $Q_0(\mathbb{D}') \neq Q_0(\mathbb{D}'')$. It is because $\langle c_0, c_n \rangle \in Q_0(\mathbb{D}')$ and $\langle c_0, c_n \rangle \notin Q_0(\mathbb{D}'')$. It is because vertex $n \notin G'$ and because of that $c_0 \in \mathbb{D}_3$ and $c_n \in \mathbb{D}_2$. □

3.3. Automata

In this section we will construct Our2NFA-ϵ $M^{\mathcal{A}}$ that recognizes $L^{\mathcal{A}}$. We will first give some intuitions on how this automaton works and then we will present a formal construction.

3.3.1. Intuitions

First let us recall that $L^{\mathcal{A}} = \{w \in \Sigma^* : G_w^{\mathcal{A}} \text{ is connected}\}$. That means that $w = a_1 a_2 \ldots a_n \in L^{\mathcal{A}}$ if and only if there exists a path in $G_w^{\mathcal{A}}$ that connects vertices 0 and n. Our2NFA-ϵ $M^{\mathcal{A}}$ working on a word w will look for such a path and will accept w if and only if it finds such a path. Now imagine that $M^{\mathcal{A}}$ needs to decide whether $w \in L^{\mathcal{A}}$. That is it should decide whether there exist a path in $G_w^{\mathcal{A}} = \langle V, E \rangle$ that connects vertices 0 and n. Let us further recall that $(i, j) \in E \Leftrightarrow w[i+1, j] \in \mathcal{A}$. Now for the better understanding it is good to think of vertices $V = \{0, 1, \ldots, n\}$ as placed between letters of w like that: $0 - a_1 - 1 - a_2 - 2 - \cdots - a_n - n$. Then $(i, j) \in E$ if and only if a word written between i and j belongs to $\mathcal{A}$.

Now we are ready to give some intuitions on how the automaton works. $M^{\mathcal{A}}$ starts working in the accepting state a with the head over a_1. Then it works in

phases, where each phase corresponds to a traversal of one edge in $G_w^{\mathcal{A}}$. In each phase, which starts with the head over a_i, it nondeterministically chooses a word $u = b_1 b_2 \dots b_k \in \mathcal{A}$ and a direction *dir* (left or right). Then depending on which direction was chosen it checks whether:

1. w[i,i+k-1] = u, if *dir* = right,
2. w[i-k,i-1] = u, if *dir* = left.

This check is done by comparing the input word letter by letter with u. If a mismatch is found then $M^{\mathcal{A}}$ transitions to the reject state and will never accept. However if the condition (depending on *dir* either 1. or 2.) is satisfied then the automaton transitions back to the accepting state a and the next phase starts. After such a phase the head is over a_{i+k} if *dir* = right and over a_{i-k} if *dir* = left.

This description gives us an automaton that recognizes $L^{\mathcal{A}}$. It is because each successful (that is ending in the accepting state) phase corresponds to a traversal of an edge in $G_w^{\mathcal{A}}$ (think about $0 - a_1 - 1 - a_2 - 2 - \dots - a_n - n$). So if $w \in L^{\mathcal{A}}$ and the nondeterministic choices are made correctly, then the automaton will follow a path connecting 0 and n and after reading a_n will accept. On the other hand if $w \notin L^{\mathcal{A}}$ then no matter what nondeterministic choices are made, $M^{\mathcal{A}}$ will not accept as there is no path in $G_w^{\mathcal{A}}$ connecting 0 and n.

3.3.2. Construction

The construction of $M^{\mathcal{A}}$ will take place in stages. First we will create an automaton $M^{\varnothing}$ that recognizes empty language and serves as a template upon which $M^{\mathcal{A}}$ will be built. Then for each word $w \in \mathcal{A}$ we will add new states to the current automaton to finally (after considering each word from $\mathcal{A}$) build $M^{\mathcal{A}}$.

Definition 12. Let $M^{\varnothing} = (S, \Sigma, \delta, a, r)$ be Our2NFA-ϵ where:

- $S = \{a, r\}$ is a set of states,
- δ satisfies:
  - For all $c \in \Sigma$: $\delta(a, c) = \{(r, right)\}$,
  - For all $c \in \Sigma$: $\delta(r, c) = \{(r, right)\}$,

Definition 13. For an Our2NFA-ϵ $M = (S, \Sigma, \delta, a, r)$ and a word $w = a_1 a_2 \dots a_n$ we define $M + w$ as a new Our2NFA-ϵ $M' = (S', \Sigma, \delta', a, r)$ where:

- $S' = S \uplus \{A_1, A_2, \dots, A_n\} \uplus \{A_1^r, A_2^r, \dots, A_n^r\}$,
- $\delta'\big|_{S \setminus \{a\}} = \delta$,

- For all $i \in [1, n]$:

$$\delta'(A_i, a_i) \coloneqq \begin{cases} \{(A_{i+1}, right)\}, & \text{if } i < n \\ \{(a, right)\}, & \text{otherwise.} \end{cases}$$

$$\delta'(A_i^r, a_i) \coloneqq \begin{cases} \{(A_{i-1}^r, left)\}, & \text{if } i > 1 \\ \{(a, stay)\}, & \text{otherwise.} \end{cases}$$

- $\delta'(a, \epsilon) = \delta(a, \epsilon) \cup \{(A_1, stay), (A_n^r, left)\}$,
- All non specified transitions lead to $\{(r, right)\}$.

Definition 14. For a set of words $\mathcal{A}$ we define Our2NFA-ϵ $M^{\mathcal{A}}$ as $M^{\varnothing} + \sum_{w \in \mathcal{A}} w$.

It might be hard to follow the construction of $M^{\mathcal{A}}$ from the definitions only. That is why we present here pictures of two sample automata over $\Sigma = \{x, y, z\}$: $M^{\{xyz\}} = M^{\varnothing} + xyz)$ and $M^{\{xyz,yz\}} = M^{\varnothing} + xyz + yz$, see Figures 3.1 and 3.2. Automaton $M^{\{xyz\}}$ for instance recognizes exactly language $L^{xyz} = (xyz)^*$. You can see that by considering what happens after visiting state a. Let $M^{\{xyz\}}$ be in state a and the "head" be over letter w_i of a word $w = w_1 w_2 \dots w_n$. Now $M^{\{xyz\}}$ nondeterministically chooses either to transition to the state Z^r and move the "head" to the left or transition to the state X and do not move the "head". If the automaton chose to transition to state:

- X then to not to transition to the reject state r the following must be true: $w_i = x, w_{i+1} = y, w_{i+2} = z$ and after reading these 3 letters automaton is in state a once again and the "head" is over w_{i+3}.
- Z^r then to not to transition to the reject state r the following must be true: $w_{i-1} = z, w_{i-2} = y, w_{i-3} = x$ and after reading these 3 letters automaton is in state a once again and the "head" is over w_{i-3}.

On the Figure 3.2 you can see $M^{\{xyz,yz\}}$. When you compare 3.2 to 3.1 you will see how $M^{\{xyz,yz\}}$ is constructed from $M^{\{xyz\}}$ (remember that $M^{\{xyz,yz\}} = M^{\{xyz\}} + yz$). The analysis of this automaton is similar to the analysis of $M^{\{xyz\}}$ and gives that the language recognized by $M^{\{xyz,yz\}}$ is exactly $L^{\{xyz,yz\}}$.

Lemma 15. *For every set of words $\mathcal{A}$, $M^{\mathcal{A}}$ recognizes $L^{\mathcal{A}}$, so $L^{\mathcal{A}}$ is a regular language.*

Proof. Automaton $M^{\mathcal{A}}$ starts in state a. Then it proceeds to work in phases, where a phase starts when $M^{\mathcal{A}}$ is in state a and finishes when $M^{\mathcal{A}}$ is in state a or r.

Whenever $M^{\mathcal{A}}$ reaches state r it stays in this state forever and never accepts, so we focus on situations when r is not reached. Each phase begins in state a and then $M^{\mathcal{A}}$ nondeterministically chooses one of the ϵ-moves and by that we think

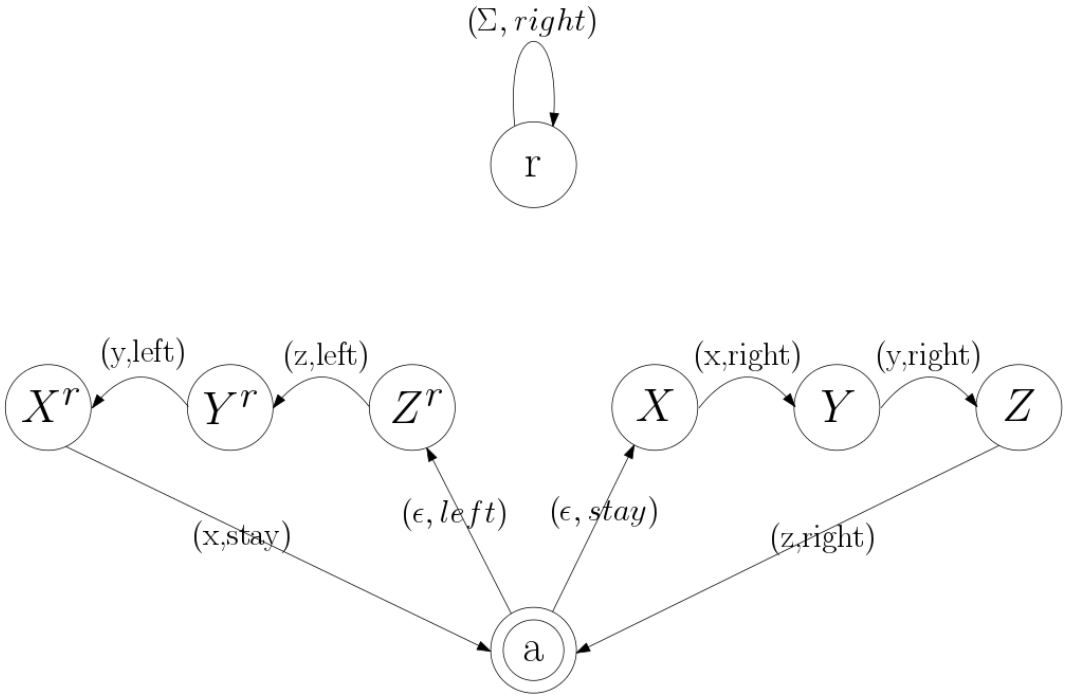


Figure 3.1: Automaton $M^{\{xyz\}} = M^{\emptyset} + xyz$. All nonspecified transitions lead to the state r.

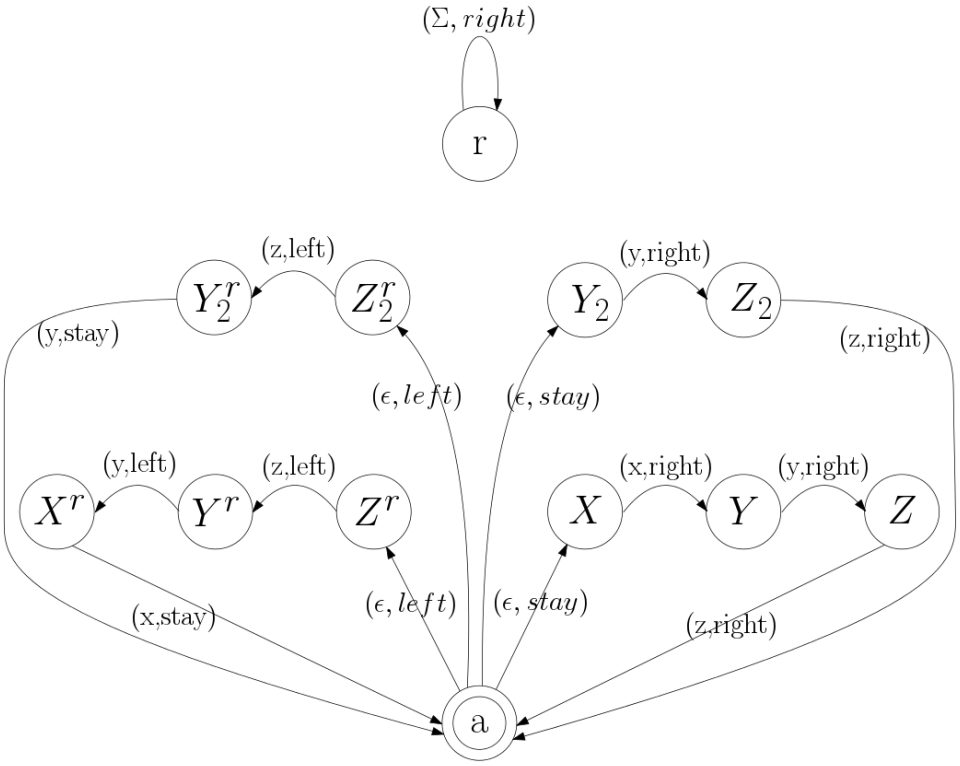


Figure 3.2: Automaton $M^{\{xyz,yz\}} = M^{\emptyset} + xyz + yz$. All nonspecified transitions lead to the state r.

that it chooses a word $w \in \mathcal{A}$ and a direction *dir* (left or right). Then it proceeds with reading the input in chosen direction *dir* and comparing it with w (read either from left to right or from right to left depending on *dir*). If there is a mismatch it transitions to r and stays there forever. Otherwise $M^{\mathcal{A}}$ goes back to state a and another phase begins. $M^{\mathcal{A}}$ accepts a word if after reading its last letter it is in state a.

This means that $M^{\mathcal{A}}$ accepts exactly $L^{\mathcal{Q}}$ as this language consists of exactly these words w for which $G_w^{\mathcal{Q}}$ is connected. □

3.4. Putting it all together

Now we are ready to prove the main theorem (restated here for convenience):

Theorem 4. *Query Determinacy Problem for Regular Path Queries is decidable when each $Q_i \in \mathcal{Q}$ is a one word language and Q_0 is a regular language.*

Proof. By Lemma 11 it is enough to be able to decide whether, for given Q_0 and $\mathcal{Q}$, it holds that $Q_0 \subset L^{\mathcal{Q}}$. By Lemma 15 we know that $L^{\mathcal{Q}}$ is a regular language. As we know that inclusion of regular languages is decidable it ends the proof. □

Bibliografia

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

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

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

[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;

[GM018a] G. Głuch, J. Marcinkowski, P. Ostropolski-Nalewaja *The First Order Truth behind Undecidability of Regular Path Queries Determinacy.*;

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

Dodatek A

Załącznik 1

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*.

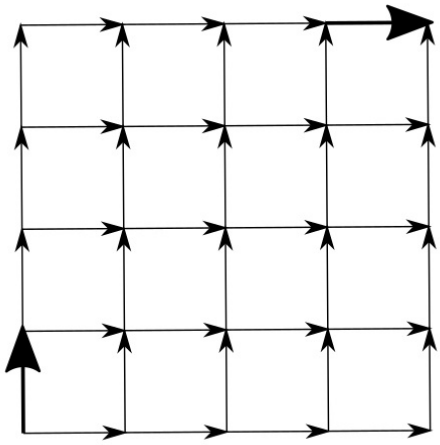


Figure 1. Our Grid.

$a_2(x'_1, x'_2) \dots a_n(x'_{n-1}, y')$ in $\mathbb{M}$. Let $\mathbb{D}^r_i$ 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 $h^r_i = h_i \cup \{\langle x_i, x'_i\rangle | i \in [n-1]\}$. Now let $\mathbb{D}' = \bigcup_{r\in R_i} \mathbb{D}^r_i$ and $h'_i = \bigcup_{r\in R_i} h^r_i$. 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 $\bullet$ *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. $\beta\Big(\bigoplus_{s\in\mathcal{S}\setminus\{black\}}({}_s\mathbf{A}^W_V)\Big)\Sigma^\star_0\omega$
2. $\beta\Sigma^\star_0\Big(\bigoplus_{s\in\mathcal{S}\setminus\{black\}}({}_s\mathbf{B}^W_H)\Big)\omega$
3. $\beta\Sigma^\star_0({}_a\bullet^W_d)({}_b\bullet^W_{d'})\Sigma^\star_0\omega$ for each forbidden $\langle(d, a), (d', b)\rangle \in \mathcal{F}$.

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

1. $\alpha\Sigma^\star_0(\bullet^W)\Sigma^\star_0\omega$
2. $\beta\Sigma^\star_0(\bullet^C)\Sigma^\star_0\omega$

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 $(\mathbf{p}^r_q)$, 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 $\mathbb{D}_0 = G(\alpha[(\mathbf{A}^C_H)(\mathbf{B}^C_V)]^m\omega)[a, b]$ 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 $\mathbb{D}_0 = G(\alpha[(\mathbf{A}^C_H)(\mathbf{B}^C_V)]^m\omega)[a, b]$. 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 $\mathbb{L}^G_i$ and $\mathbb{L}^R_i$ be parts of $\mathbb{L}_i$ consisting of (resp.) green and red edges. Then:*

(i) $\mathbb{D}_0 = \mathbb{L}^G_0$,
(ii) $\mathbb{D}_{2i} = \mathbb{L}^G_{2i} \cup \mathbb{L}_{2i-1}$,
(iii) $\mathbb{D}_{2i+1} = \mathbb{L}^R_{2i+1} \cup \mathbb{L}_{2i}$.

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

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

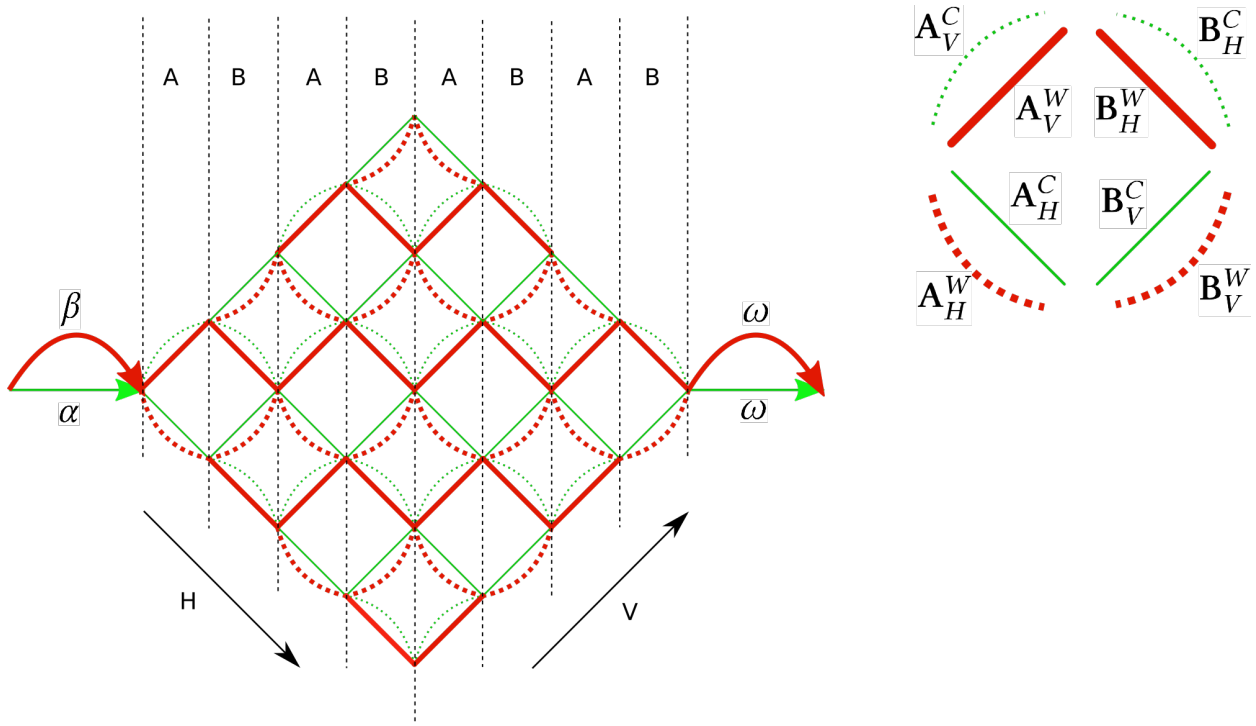


Figure 2. $\mathbb{G}_m$ with $m = 4$ (left). Smaller picture in the top-right corner explains how the different line styles on the main picture map to Σ_0.[9]

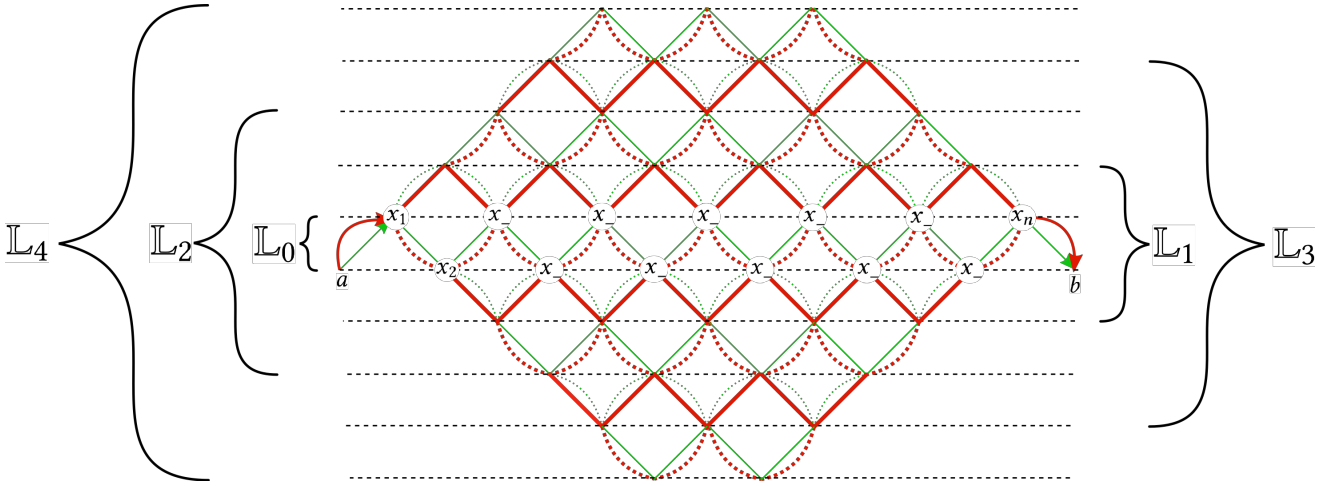


Figure 3. Five first Layers of $\mathbb{G}_m$ with $m = 6$.

$\mathbb{H}$. 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 $\mathbb{H} \models \mathcal{Q}^{\leftrightarrow}_{good}$).

Reasoning for red edges is analogous. □

In his first move *Fugitive* must satisfy all the requests in $S_0 = rq(\mathcal{Q}^{\leftrightarrow}, \mathbb{D}_0)$. 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 $\mathcal{Q}^{\rightarrow}_{good}$. 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 $2i + 1$: 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) $\Rightarrow$ (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)$\rightarrow$ (ii).

For the proof Lemma 5.1 ($\neg$i)$\rightarrow$ ($\neg$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 $Q^{\leftrightarrow}_{good}$ (Observation 9.10) and from $Q^{\leftrightarrow}_{ugly}$ (for trivial reasons, as no paths from any $G(L) \cup R(L)$ with $L \in Q_{ugly}$ 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;

Dodatek B

Załącznik 2

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 [] 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 $\mathbb{D}$ 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 $Q_0(\mathbb{D}_1) = Q_0(\mathbb{D}_2)$.

[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.

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 $\mathcal{Q}$ consists of a single CQ.

But this is not a survey paper, so let us just point a reader interested in the history of QDP to Nadime Francis 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 $\mathcal{Q}$ 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 $\mathcal{Q}$ are conjunctive queries and Q_0 is a first order sentence (or the other way round). Another somehow related (although no longer contained in the first order/SQL paradigm) negative result is presented in [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 $\mathcal{Q}$ being conjunctive queries. The queries in [GM15] and [GM16] are quite complicated (the Turing machine there is encoded in the arities of te 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 $\mathcal{Q}$ has only one free variable.

Then, in [A11] decidability was shown for $\mathcal{Q}$ 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 $\mathcal{Q}$ 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 $\mathcal{Q}$ are over a single-letter alphabet. See how difficult the analysis is here – despite a lot of effort (the proof of the result in [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.

[3]By 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 $\mathcal{Q}$ 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 $\mathbb{D}_0$ from a to b, which represents a word from some regular language $G(Q_0)$. Then, in each step he must "satisfy requests"– if there is a path from some v to w in the current structure, representing a word from some (*) regular language Q then he must add a path representing a word from another language Q' connecting these v and w. He loses when, in this process, a path from a to b from yet another language $R(Q_0)$ is created.

In this paper this game is replaced by a two-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 $\mathcal{Q}^{12}_{good} - \mathcal{Q}^{15}_{good}$.

The second reason is in $R(Q_0)$. This – one can think – is the language of "forbidden patterns" – paths from a to b that *Fugitive* must not construct. If he does, it means that he "cheats". But now again, $R(Q_0)$ is finite. So how can we use it to detect Fugitive's cheating on paths no longer than the longest one in $R(Q_0)$? This at first seemed to us to be an impossible task. And the solution is in a complicated machinery of languages producing edges labelled with x and y (languages $\mathcal{Q}^{12}_{good} - \mathcal{Q}^{15}_{good}$ producing x and y and all languages in $\mathcal{Q}_{ugly}$ 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 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. Each structure we consider will contain two distinguished constants a and b.

For two structures $\mathbb{G}$ and $\mathbb{G}'$ over Σ, with sets of vertices V and V', a function $h : V \to V'$ is (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 $\mathbb{G}'$.

A. Path 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 path 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 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 $\mathcal{Q}$ and $\mathcal{L}$ to denote sets of regular languages. The notation $Q(\mathbb{D})$ has the natural meaning: $Q(\mathbb{D}) = \{\langle x, y\rangle \mid \mathbb{D} \models Q(x, y)\}$.

IV. Red-Green Structures and Escape

A. 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 $Add(D, L \to L', \langle x, y \rangle)$ depends on the choice of $w \in L'$. So the procedure is non-deterministic.

For a regular language L we define $L^{\to} = G(L) \to R(L)$ and $L^{\leftarrow} = R(L) \to G(L)$. All regular constraints we are going to consider are either $L^{\to}$ or $L^{\leftarrow}$ for some regular L.

For a regular language L we define $L^{\leftrightarrow} = \{L^{\to}, L^{\leftarrow}\}$ 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 $t \in L^{\to}$ ($t \in L^{\leftarrow}$) 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 Σ does not determine (does not finitely determine) a regular path query Q_0, over the same alphabet, if and only if there exists a structure $\mathbb{M}$ (resp. a finite structure) and a pair of vertices $a, b \in \mathbb{M}$ 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*.

B. The game of Escape

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

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

The rules of the game are:

- First Fugitive picks the *initial position* of the game as $\mathbb{D}_0 = (G(w))[a,b]$ for some $w \in Q_0$.
- Suppose $\mathbb{D}_\beta$ is the current position of some play before move $\beta + 1$ and $S_\beta = rq(\mathcal{Q}^{\leftrightarrow}, \mathbb{D}_\beta)$. Then, in move $\beta + 1$, *Crocodile* picks one request $\langle x, y, t \rangle \in S_\beta$ and then *Fugitive* can move to any position of the form:
$$\mathbb{D}_{\beta+1} \in Add(\mathbb{D}_\beta, t, \langle x, y \rangle)$$
- For a limit ordinal λ the position $\mathbb{D}_\lambda$ is defined as $\bigcup_{\beta < \lambda} \mathbb{D}_\beta$.
- If $rq(\mathcal{Q}^{\leftrightarrow}, \mathbb{D}_i)$ is empty then for each $j > i$ structures $\mathbb{D}_j$ and $\mathbb{D}_i$ are equal.
- *Fugitive* loses when for a *final position* $\mathbb{D}_{\omega^2} = \bigcup_{\beta < \omega^2} \mathbb{D}_\beta$ it is true that $\mathbb{D}_{\omega^2} \models (R(Q_0))(a,b)$. Otherwise he wins.

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

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 $\mathbb{D}_0$, 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 QDP-RPQ:

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

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

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

C. Universality of Escape. Proof of Lemma 2

It is clear that $(i) \Leftarrow (ii)$ 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 IV 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 $\mathbb{D}_0$ 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 (regardless of the Crocodile's moves) Fugitive can reach some final position $\mathbb{D}_{\omega^2} \in \Omega(T, \mathbb{D}_0)$ such that there exists a homomorphism h from $\mathbb{D}_{\omega^2}$ to $\mathbb{M}$.*

Proof. Next lemma provides the induction step for the proof of Lemma 3.

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

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

Proof. Let $r = \langle x, y, X \to Y\rangle$ for some $x, y \in \mathbb{D}_\beta$ and let $x' = h_\beta(x)$ and $y' = h_\beta(y)$. Since $\mathbb{D} \models X(x, y)$ and since h_β is a homomorphism we know that $\mathbb{M} \models X(x', y')$. But $\mathbb{M} \models T$ so there is also $\mathbb{M} \models Y(x', y')$ and thus for some $a_1 a_2 \dots a_n \in Y$ there is path $p' = a_1(x', x'_1)$, $a_2(x'_1, x'_2) \dots a_n(x'_{n-1}, y')$ in $\mathbb{M}$. Let $\mathbb{D}'_\beta$ be a structure created by adding to $\mathbb{D}_\beta$ 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 $h'_\beta = h_\beta \cup \{\langle x_i, x_{i+1}\rangle | i \in [n-1]\}$. It is easy to see that $\mathbb{D}'_\beta$ and h'_β are requested $\mathbb{D}_{\beta+1}$ and $h_{\beta+1}$. □

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

This along with Lemma 4 proves that $\mathbb{D}_{\omega^2}$ and h_{ω^2} are as required by Lemma 3. □

Now we will prove the (i)⇒(ii) part of Lemma 2.

Assume (i). Let $\mathbb{M}$ be a counterexample as in Lemma 1. Let a, b and $w \in Q_0$ be such that $\mathbb{M} \models (G(w))(a, b)$ and $\mathbb{M} \not\models (R(Q_0))(a, b)$. Applying Lemma 3 to $\mathbb{D}_0 = G(w)[a, b]$ and to $\mathbb{M}$ we know that *Fugitive* (regardless of the *Crocodile*'s moves) can reach some winning final position $\mathbb{D}_{\omega^2}$ such that there is homomorphism from $\mathbb{D}_{\omega^2}$ to $\mathbb{M}$. It is clear that $\mathbb{D}_{\omega^2} \not\models (R(Q_0))(a, b)$ as we know that $\mathbb{M} \not\models (R(Q_0))(a, b)$. This shows that $\mathbb{D}_{\omega^2}$ 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 ϕ 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 Tilling 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 $\langle c, d\rangle$ where $c, d \in \{V, H\} \times \mathcal{S}$. Let the set of all these instances be called $\mathcal{I}$.*

Definition 5. *A proper shading*[4] *is an assignment of shades to edges of some square grid $\mathbb{G}$ (see Figure 1) such that:*

(a1) *each horizontal edge of $\mathbb{G}$ has a label from $\{H\} \times \mathcal{S}$.*
(a2) *each vertical edge of $\mathbb{G}$ has a label from $\{V\} \times \mathcal{S}$.*
(b1) *bottom-left horizontal edge is shaded **gray**[5].*
(b2) *upper-right vertical edge (if exists) is shaded **black**.*
(b3) ***G** contains no forbidden paths of length 2 labelled by $(c, d) \in \mathcal{F}$.*

We define two subsets of instances of OGTP:

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

[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.

[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.

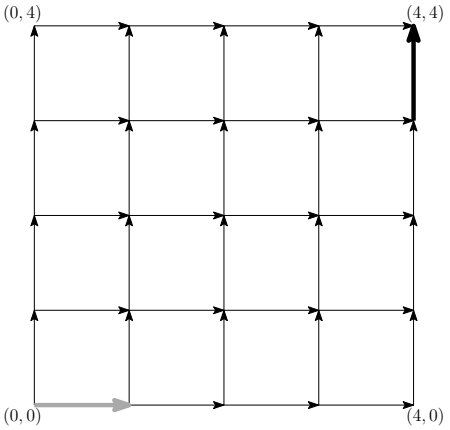


Figure 1. Finite square grid.

$\mathcal{B} =$ *$\{I \in \mathcal{I}$|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 $\mathfrak{R}$ ($\mathfrak{R}$ like $\mathfrak{R}$eduction) 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 $\langle \mathcal{Q}, Q_0\rangle = \mathfrak{R}(I)$:*

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

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

VI. The function $\mathfrak{R}$

Now we define a function $\mathfrak{R}$, 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

construct an instance $\langle \mathcal{Q}, Q_0 \rangle = R(\langle \mathcal{S}, \mathcal{F} \rangle)$ of QDP-RPQ.

The edge alphabet (signature) will be

$$\Sigma = \{\alpha^C, \alpha^W, x^C, x^W, y^C, y^W, \$^C, \$^W, \omega\} \cup \Sigma_0$$

where $\Sigma_0 = \{A, B\} \times \{H, V\} \times \{W, C\} \times \mathcal{S}$. We think of H and V as **orientations** – *Horizontal* and *Vertical*. W and C stand for *warm* and *cold*. It is worth reminding at this point that relations from $\bar{\Sigma}$ will – apart from shade, orientation and temperature – have also a **color**, red or green.

Notation VI.1. *We use the following notation for elements of* Σ_0*:*

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

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

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

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

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

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

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

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

1) $\alpha^C \Sigma^{\leq 4}(\bullet^W)\Sigma^{\leq 4}\omega$
2) $\alpha^W \Sigma^{\leq 4}(\bullet^C)\Sigma^{\leq 4}\omega$

where $\Sigma^{\leq 4} = \bigcup_{i=1}^{4} \Sigma^i$.

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. The regular constraints $(Q^{10}_{good})^{\leftrightarrow}$ and $(Q^{11}_{good})^{\leftrightarrow}$ 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.

On the other hand, each language in $\mathcal{Q}_{bad} \cup \mathcal{Q}_{ugly}$ contains 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 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 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 $Q_{start} := \alpha^C x^C ({}_{gray}\mathbf{A}^C_H)(\mathbf{B}^C_V) y^C \omega$, and let:

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

Q_{start} may look like a single word language, but it is not: do not forget that $(\mathbf{B}^C_V)$ 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 7

The rest of the paper will be devoted to the proof of Lemma 7 (restated here for convenience):

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

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

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

Proof of claim (i) – which will be presented in the end of Section XV – 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 $\langle \mathcal{S}, \mathcal{F} \rangle$ has no proper shading then the *Crocodile* **does have** a winning strategy in the Escape($\mathcal{Q}$, Q_0) (where $\langle \mathcal{Q}, Q_0 \rangle = \mathfrak{R}(\langle \mathcal{S}, \mathcal{F} \rangle)$). As we remember from Section IV-B, in such a game *Fugitive* will first choose, as the initial position of the game, a structure $\mathbb{D}_0 = w[a, b]$ for some $w \in G(Q_0)$. Then, in each step, Crocodile will pick a request in the current structure (current position of the game) $\mathbb{D}$ and Fugitive will satisfy this request, creating a new (slightly bigger) current $\mathbb{D}$. Fugitive will win if he will be able to play forever (by which me mean ω^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 $\mathbb{D}_0$ resulting from his initial move must be $(G(w))[a, b]$ for some $w \in Q_{start}$.
(II) He must not allow any green edge with warm label and any red edge with cold label to appear in $\mathbb{D}$.
(III) He must never allow any path labelled by $G(\mathcal{Q}_{bad}) \cup R(\mathcal{Q}_{ugly})$ 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 $\mathbb{D}_0$). In the first stage *Fugitive* will be forced to build a structure called $\mathbb{P}_1$ (defined in Section X). In the second stage the specified structures will be called $\mathbb{P}_m$ and ${}^{\$}\mathbb{P}_m$ (each defined in Section X) and in the third stage *Fugitive* will be forced to construct one of the structures $\mathbb{G}_m$ or $\mathbb{L}^k_m$ (defined in Section XIII)

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

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

VIII. 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) 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 $\mathcal{Q}_{bad}$ and $\mathcal{Q}_{ugly}$, which clearly are all about suspiciously looking patterns, are all in Q_0

But another rule of the game is that at the beginning *Fugitive* picks his initial position $\mathbb{D}_0$ as a path (from a to b) labelled with some $w \in G(Q_0)$,

so it would be nice to think of Q_0 as of initial configurations of this computing device. The fact that the same object is playing the set of forbidden patterns and, at the same time, the set of initial configurations is a problem, and this problem is solved by having languages from $\mathcal{Q}_{ugly} \cup \mathcal{Q}_{bad}$ both in $\mathcal{Q}$ and in Q_0:

Lemma 1 (**Principle I**). *Fugitive must choose to start the Escape game from $\mathbb{D}_0 = G(q)[a, b]$ 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 $\mathbb{D}_0$.

Proof. If $\mathbb{D}_0 = G(q)[a, b]$ for $q \in Q_0 \setminus Q_{start}$ then $\mathbb{D}_0 \models G(L)(a, b)$ for some $L \in \mathcal{Q}_{ugly} \cup \mathcal{Q}_{bad}$. Then in the next step *Crocodile* can pick request $\langle a, b, G(L) \to R(L)\rangle$. After *Fugitive* satisfies this request, a structure $\mathbb{D}_1$ is created such that $\mathbb{D}_1 \models R(L)(a, b)$ and *Crocodile* wins. □

From now on we assume that *Fugitive* obeys Principle I. This implies that $\mathbb{D}_0$ as demanded by Principle I will always be a substructure of any current structure $\mathbb{D}$.

IX. Principles II and III

In this section we will formalise the intuition considering languages from $\mathcal{Q}_{ugly}$ 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 $\mathbb{D}$ 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 $\mathcal{Q}_{good}$. □

Definition 2. *A P2-ready*[6] *structure $\mathbb{D}$ is a structure satisfying the following:*

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

[6]Meaning "ready for Principle II".

Lemma 3 (**Principle II**). *Suppose that, after* Fugitive*'s move, the current structure $\mathbb{D}$ is a P2-ready structure. Then neither a green edge with label from $(\bullet^W)$ nor a red edge with label from $(\bullet^C)$ may appear in $\mathbb{D}$.*

Proof. First suppose that there is such a green edge $e = \langle x, y\rangle$ with label $(\bullet^W)$ in structure $\mathbb{D}$. 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^1_{ugly} 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^1_{ugly}, 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 $\mathbb{D}$. 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^2_{ugly}. □

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

Proof. First consider a case where $\mathbb{D} \models R(\mathcal{Q}_{bad})(a, b)$. Then Fugitive has already lost as $\mathcal{Q}_{bad} \subset Q_0$.

The second case is when $\mathbb{D} \models G(\mathcal{Q}_{bad})(a,b)$ and $\mathbb{D} \not\models R(\mathcal{Q}_{bad})(a,b)$. Then Crocodile can pick request $\langle a, b, Q^{i\rightarrow}_{bad}\rangle$ (for some i) for Fugitive to satisfy. In both cases after at most one move Fugitive loses. □

X. The paths $\mathbb{P}_m$ and ${}^{\$}\mathbb{P}_m$

Definition 1. *(See[7] Figure 2 and Figure 3) $\mathbb{P}_m$, for $m \in \mathbb{N}_+$, is a directed graph (V, E) where $V = \{a, a', b', b\} \cup \{v_i : i \in [0, 2m]\}$ and the edges E are labelled with symbols from $\Sigma \setminus \Sigma_0$ or with symbols of the form $(\mathbf{p}^r_q)$, 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 $s \in \mathcal{S}$ here: the labels we now use are sets of symbols from $\bar{\Sigma}$ like in Notation VI.1: we watch the play in* Crocodile's *shade filtering glasses.*

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

- *Vertex a' is a successor of a and vertex b is a successor of b'. For each $i \in [0, 2m]$ the successors of v_i are v_{i+1} (if it exists) and b' and the predecessors of v_i are v_{i-1} (if it exists) and a'. From each node there are two edges to each of its successors, one red and one green, and there are no other edges.*
- *Each Cold edge (labelled with a symbol in $(\bullet^C)$) is green.*
- *Each Warm edge (labelled with a symbol in $(\bullet^W)$) is red.*
- *Each edge $\langle v_{2i}, v_{2i+1}\rangle$ is from $(\mathbf{A}_H)$.*
- *Each edge $\langle v_{2i+1}, v_{2i+2}\rangle$ is from $(\mathbf{B}_V)$.*
- *Each edge $\langle a', v_i\rangle$ is labelled by either x^C or x^W.*
- *Each edge $\langle v_i, b'\rangle$ is labelled by either y^C or y^W.*
- *Edges $\langle a, a'\rangle$ with label $G(\alpha^C)$ and $\langle a, a'\rangle$ with label $R(\alpha^W)$ are in E.*
- *Edges $\langle b', b\rangle$ with albel $G(\omega)$ and $\langle b', b\rangle$ $R(\omega)$ are in E.*

Definition 2. *${}^{\$}\mathbb{P}_m$ for $m \in \mathbb{N}_+$ is $\mathbb{P}_m$ with two additional edges:*

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

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

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

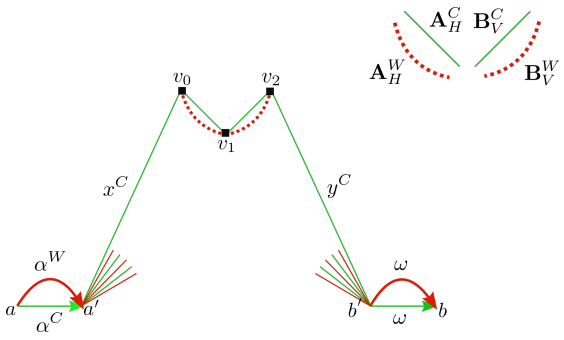


Figure 2. $\mathbb{P}_1$.

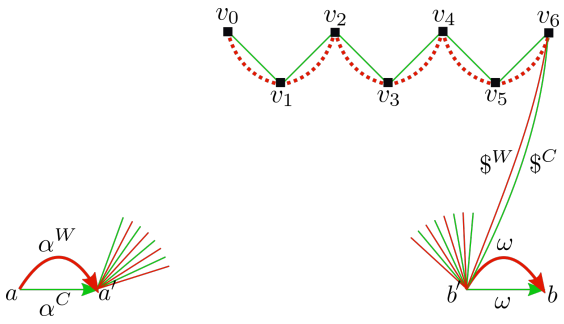


Figure 3. ${}^{\$}\mathbb{P}_3$.

Exercise X.3. *The only requests generated by $\mathcal{Q}_{good}$ in ${}^{\$}\mathbb{P}_m$ are those generated by Q^{10}_{good} and Q^{11}_{good}.*

Exercise X.4. *Each $\mathbb{P}_k$ and each ${}^{\$}\mathbb{P}_k$ is a P2-ready structure.*

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.

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 $S = (l) +\!\!+ S'$ 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[8] *Crocodile proceeds with strategy S'.*

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

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

Recall that $\mathbb{D}_0$ 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 $\mathbb{D}_0$ forces Fugitive to add $R(\alpha^W)[a,a']$ and $R(\omega)[b',b]$.*

Proof. Consider these languages one by one:

$1=\omega$: This language generates only one request $\langle b',b,Q^{1\rightarrow}_{good}\rangle$ (one because edge $\langle b',b\rangle$ with label $G(\omega)$ is the only one in $\mathbb{D}_0$ labelled with ω), which has to be satisfied with $R(\omega)[b',b]$ as language Q^1_{good} consists of only one word.

$2=\alpha^C+\alpha^W$: There is a green edge labelled with α^C in $\mathbb{D}_0$ and thus this language generates a request $\langle a,a',Q^{2\rightarrow}_{good}\rangle$ (and no other requests). This request can be satisfied by *Fugitive* either by adding the edge $R(\alpha^C)[a,a']$ or by adding the edge $R(\alpha^W)[a,a']$. Suppose that *Fugitive* satisfies this request with $R(\alpha^C)[a,a']$. Notice that *Crocodile* can now require *Fugitive* to satisfy requests $Reqs = \{\ \langle a',v_0,Q^{3\rightarrow}_{good}\rangle,\ \langle v_2,b',Q^{4\rightarrow}_{good}\rangle,\ \langle v_0,v_1,Q^{9\rightarrow}_{good}\rangle,\ \langle v_1,v_2,Q^{6\rightarrow}_{good}\rangle\}$ which will force the Fugitive to build a red path from a' to b'. Each of these request has to be satisfied with a red edge with some label em warm (with the upper index W) or *cold* (with C).

Consider what happens if one of these requests is satisfied with a *warm* letter. Then we have that $\mathbb{D}\models R(Q^1_{ugly})(a,b)$ and *Fugitive* loses. It means that each request from $Reqs$ must be satisfied with a red edge labelled with a *cold* letter. But then notice that $\mathbb{D}\models R(Q_{start})(a,b)$ and *Fugitive* also loses. □

[8] In order for this "then" to make sense we need the total number of moves of the game to be ω^2 rather than ω.

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

uuu!

The following Lemma explains the role of S_{color} and is a first cousin of Observation IX.1:

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

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

Proof. The proof is an easy consequence of Principle II and the fact that all words from Q^{1-9}_{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 Σ. □

Lemma 4. *Strategy S_{start} applied to $\mathbb{D}_0$ forces* Fugitive *to build $\mathbb{P}_1$.*

Proof. Consider languages from S_{start} one by one:

- $1=\omega$: By Lemma 2 this language forces *Fugitive* to add $R(\omega)[b',b]$.
- $2=\alpha^C+\alpha^W$: By Lemma 2 this language forces the Fugitive to add $R(\alpha^W)[a,a']$.
- $15 = y^W + \$^C + (\mathbf{A}^C_H)(\mathbf{B}^C_V)y^C$: This language generates only one request $\langle v_0,b',Q^{15\rightarrow}_{good}\rangle$ since neither y^W nor $\$^C$ occurs in the current structure. This request has to be satisfied with $R(y^W)[v_0,b']$ by Principle II. We can use Principle II since after strategy $(1,2)$ was applied the structure is *P2-ready*.
- $14 = x^W + x^C + x^C(\mathbf{A}^C_H)(\mathbf{B}^C_V)$: This language generates only two requests

$\langle a', v_2, Q^{14\rightarrow}_{good}\rangle$ and $\langle a', v_0, Q^{14\rightarrow}_{good}\rangle$. The first request has to be satisfied with $R(x^W)[a', v_0]$ and the second with $R(x^W)[a', v_2]$, both due to Principle II.

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

- $12 = x^C\left((\mathbf{A}^C_H) + (\mathbf{B}^C_H) + (\mathbf{A}^C_V) + (\mathbf{B}^C_V)\right) + x^C + x^W$: This language generates only one request: $\langle a', v_1, Q^{12\rightarrow}_{good}\rangle$. It is because there are no requests generated by neither x^C nor x^W in Q^{12}_{good} by Lemma 3. There are also no other requests generated by $x^C\left((\mathbf{A}^C_H) + (\mathbf{B}^C_H) + (\mathbf{A}^C_V) + (\mathbf{B}^C_V)\right)$ in Q^{12}_{good} as the only path labeled with a word from this language is $a' \rightarrow v_0 \rightarrow v_1$. $\langle a', v_1, Q^{12\rightarrow}_{good}\rangle$ has to be satisfied with $R(x^W)[a', v_1]$ by Principle II.
- $13 = \left((\mathbf{A}^C_H) + (\mathbf{B}^C_H) + (\mathbf{A}^C_V) + (\mathbf{B}^C_V)\right) y^C + y^C + y^W$: This language generates one request $\langle v_1, b', Q^{13\rightarrow}_{good}\rangle$. It has to be satisfied with $R(y^W)[v_1, b']$ by Principle II.

Finally *Crocodile* uses strategy S_{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 $\mathbb{P}_1$.

□

XII. STAGE 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:

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

Lemma 1. *For all $m \in \mathbb{N}_+$ strategy S_m applied to $\mathbb{D}_0$ 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 *Crocodile*, using strategy S_{m-1}, forced *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 *Crocodile*'s strategy S_{m-1} then we are done, and notice that the last S_{cycle} will not change the current structure any more – this is because, due to Exercise ??? there are no requests from languages Q^{1-9}_{good} and Q^{12-15}_{good} in the current structure at this point.

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

- $15 = y^W + \$^C + (\mathbf{A}^C_H)(\mathbf{B}^C_V)y^C$. The only request generated by this language is $\langle v_{2m}, b', Q^{15\leftarrow}_{good}\rangle$, resulting from the red edge labelled with y^W connecting v_{2m} 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 $(\mathbf{A}^C_H)(\mathbf{B}^C_V)y^C$ between these vertices.
  This only request can be possibly satisfied in two different ways (it follows from Principle II): either by $G((\mathbf{A}^C_H)(\mathbf{B}^C_V)y^C)[v_{2m}, b']$ or by $G(\$^C)[v_{2m}, b']$. The case when this request is satisfied with $G(\$^C)[v_{2m}, b']$ will be considered in the last paragraph of the proof. So now we assume that this request is satisfied with $G((\mathbf{A}^C_H)(\mathbf{B}^C_V)y^C)[v_{2m}, b']$. Let us name the two new vertices as v_{2m+1} and v_{2m+2}.
- $14 = x^W + x^C + x^C(\mathbf{A}^C_H)(\mathbf{B}^C_V)$: the only request generated by this language is $\langle a', v_{2m+2}, Q^{14\rightarrow}_{good}\rangle$ resulting from the (partially) newly created green path from a' to v_{2m+2}, via v_{2m} and v_{2m+1}, labelled with $x^C(\mathbf{A}^C_H)(\mathbf{B}^C_V)y^C$.
  This request has to be satisfied with $R(x^W)[a', v_{2m+2}]$ due to Principle II.

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

- $12 = x^C\left((\mathbf{A}^C_H) + (\mathbf{B}^C_H) + (\mathbf{A}^C_V) + (\mathbf{B}^C_V)\right) + x^C + x^W$: This language generates one request: $\langle a', v_{2m+1}, Q^{12\rightarrow}_{good}\rangle$. It has to be satisfied with $R(x^W)[a', 2m+1]$ by Principle II.
- $13 = \left((\mathbf{A}^C_H) + (\mathbf{B}^C_H) + (\mathbf{A}^C_V) + (\mathbf{B}^C_V)\right) y^C + y^C + y^W$: This language generates one request $\langle v_{2m+1}, b', Q^{13\rightarrow}_{good}\rangle$. It has to be satisfied with $R(y^W)[2m+1, b']$ by Principle II.

Now *Crocodile* uses strategy S_{color} (as $S_{cycle} = (15, 14) +\!\!+ S_{color} +\!\!+ (12, 13) +\!\!+ S_{color}$). We apply Lemma 3 to conclude that *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.

Now we finally consider the case where *Fugitive* satisfied the request generated by language 15 with $G(\$^C)[v_m, b']$. Notice that the only request generated by the remaining languages from S_{cycle} is: $\langle v_{2m}, b', Q^{5\rightarrow}_{good}\rangle$, which will be satisfied by $R(\$^W)[v_{2m}, b']$ and the resulting structure will be isomorphic to ${}^{\$}\mathbb{P}_m$. This ends the proof of Lemma 1. □

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

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

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

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

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

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

Definition 3. *${}^{\$}\mathbb{G}_m$ for $m \in \mathbb{N}_+$ is $\mathbb{G}_m$ with two edges added:*

- *$\langle v_{m,m}, b'\rangle$ with label $G(\$^C)$*
- *$\langle v_{m,m}, b'\rangle$ with label $R(\$^W)$*

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

- *$\langle v_{m,m}, b'\rangle$ with label $G(\$^C)$*
- *$\langle v_{m,m}, b'\rangle$ with label $R(\$^W)$*

Exercise XIII.5. *For all m:*

- *$\mathbb{L}^m_m$ is equal to $\mathbb{G}_m$,*
- *${}^{\$}\mathbb{L}^m_m$ is equal to ${}^{\$}\mathbb{G}_m$.*

Exercise XIII.6. *For all m there are no requests generated by languages from $\mathcal{Q}_{good}$ or $\mathcal{Q}_{ugly}$ in ${}^{\$}\mathbb{G}_m$.*

XIV. Stage III

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

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

Notice that by Exercise 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 $\mathbb{L}^{m+1}_{m+1}$ or to ${}^{\$}\mathbb{L}^k_k$ for some $k \leq m$.

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

Now we define a sequence of strategies S^k_{layer}, which, similarly to strategies for building ${}^{\bullet}\mathbb{P}_{\bullet}$,

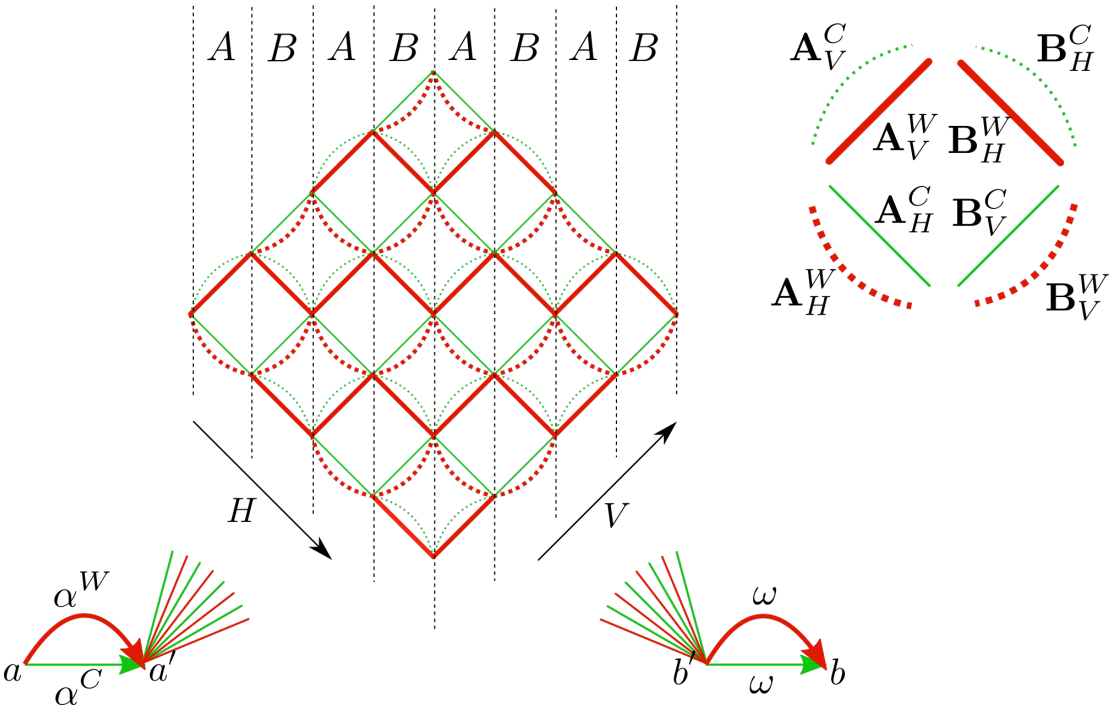


Figure 4. $\mathbb{G}_4$ (left). Smaller picture in the top-right corner explains how different line styles on the main picture map to Σ_0.[9]

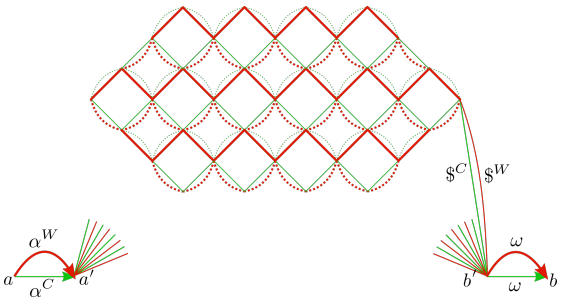


Figure 5. ${}^{\$}\mathbb{L}_6^3$.

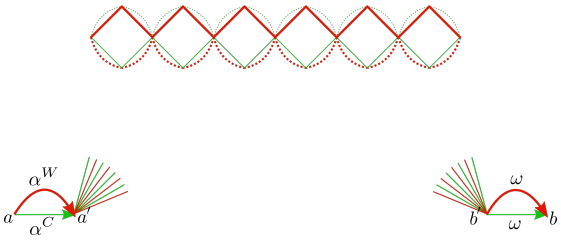


Figure 6. $\mathbb{L}_6^1$.

consist only of languages from $\mathcal{Q}_{good}$, so instead of writing Q^i_{good} we will just write i. Let:

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

.

Lemma 2. *For all $k \in \mathbb{N}$ strategy S^1_{layer} applied to the current structure ${}^{\$}\mathbb{P}_k$ forces the Fugitive to build ${}^{\$}\mathbb{L}^1_k$.*

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

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

Each of these requests results from a green path labeled with $G((\mathbf{A}_H^C)(\mathbf{B}_V^C))$ connecting v_i and v_{i+2}.
Notice that there are no requests generated by $Q_{good}^{11\leftarrow}$. It is because neither $(\mathbf{A}_V^W)$ nor $(\mathbf{B}_H^W)$ occurs in ${}^{\$}\mathbb{P}_k$.
All generated requests have to be satisfied with $R((\mathbf{A}_V^W)(\mathbf{B}_H^W))$ by Principle II. Notice that when satisfying each request a new vertex is created.

- $S_{color} = (3,4,5,6,7,8,9)$: This sequence of languages adds missing green edges $G(\mathbf{A}_H^C)$ and $G(\mathbf{B}_V^C)$ to the edges $R(\mathbf{A}_H^W)$ and $R(\mathbf{B}_V^W)$ created by language 11.
- $12 = x^C\left((\mathbf{A}_H^C) + (\mathbf{B}_H^C) + (\mathbf{A}_V^C) + (\mathbf{B}_V^C)\right) + x^C + x^W$: This language generates requests of the form $\{\langle a', t, Q_{good}^{12\rightarrow}\rangle$ for all new vertices t created by language 11. Each of these requests results from a green path labeled with $x^C\left((\mathbf{A}_H^C) + (\mathbf{B}_H^C) + (\mathbf{A}_V^C) + (\mathbf{B}_V^C)\right)$ 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 S_{color} each edge labeled with $G(x^C)$ has its counterpart labeled with $R(x^W)$.
  All generated requests have to be satisfied with $R(x^W)$ by Principle II.
- $13 = \left((\mathbf{A}_H^C) + (\mathbf{B}_H^C) + (\mathbf{A}_V^C) + (\mathbf{B}_V^C)\right) y^C + y^C + y^W$: This language generates requests of the form $\{\langle t, b', Q_{good}^{13\rightarrow}\rangle$ for all new vertices t created by language 11. Each of these requests results from a green path labeled with $\left((\mathbf{A}_H^C) + (\mathbf{B}_H^C) + (\mathbf{A}_V^C) + (\mathbf{B}_V^C)\right) y^C$ 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 S_{color} each edge labeled with $G(y^C)$ has its counterpart labeled with $R(y^W)$.
  All these requests have to be satisfied with $R(y^W)$ by Principle II.
- $S_{color} = (3,4,5,6,7,8,9)$: This sequence of languages adds missing green edges $G(x^C)$ and $G(y^C)$ to edges added by languages 12 and 13.

□

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

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

- $\mathbf{10} = (\mathbf{B}_H^W)(\mathbf{A}_V^W) + (\mathbf{B}_V^C)(\mathbf{A}_H^C)$: generates exactly $\{\langle v_{i,j}, v_{i+1,j+1}, Q_{good}^{10\rightarrow}\rangle | i - j = k, i, j \in [0, m-1]\} \cup \{\langle v_{i,j}, v_{i+1,j+1}, Q_{good}^{10\leftarrow}\rangle | i - j = k, i, j \in [0, m-1]\}$. All requests in the first group result from paths labeled with $G((\mathbf{B}_V^C)(\mathbf{A}_H^C))$ and all requests in the second group result from paths labeled with $R((\mathbf{B}_H^W)(\mathbf{A}_V^W))$.
  All requests in the first group have to be satisfied with $R((\mathbf{B}_H^W)(\mathbf{A}_V^W))$ (name the new vertices $v_{i+1,j}$) and all requests in the second group have to be satisfied with $G((\mathbf{B}_V^C)(\mathbf{A}_H^C))$ (name the new vertices $v_{i,j+1}$). All happens by Principle II.
- S_{color}: adds missing edges of opposite colors incident to newly created vertices by languahe 11.
- $\mathbf{12} = x^C\left((\mathbf{A}_H^C) + (\mathbf{B}_H^C) + (\mathbf{A}_V^C) + (\mathbf{B}_V^C)\right) + x^C + x^W$: generates exactly $\{\langle a', v_{i,j}, Q_{good}^{12\rightarrow}\rangle | i - j = k + 1, i, j \in [0, m]\} \cup \{\langle a', v_{i,j}, Q_{good}^{12\rightarrow}\rangle | j - i = k + 1, i, j \in [0, m]\}$. Each of these requests results from a green path labeled with $x^C\left((\mathbf{A}_H^C) + (\mathbf{B}_H^C) + (\mathbf{A}_V^C) + (\mathbf{B}_V^C)\right)$ 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 S_{color} each edge labeled with $G(x^C)$ has its counterpart labeled with $R(x^W)$
  All generated requests have to be satisfied with $R(x^W)$ by Principle II.
- $\mathbf{13} = \left((\mathbf{A}_H^C) + (\mathbf{B}_H^C) + (\mathbf{A}_V^C) + (\mathbf{B}_V^C)\right) y^C + y^C + y^W$: generates exactly $\{\langle v_{i,j}, b', Q_{good}^{13\rightarrow}\rangle | i - j = k + 1, i, j \in [0, m]\} \cup \{\langle v_{i,j}, b', Q_{good}^{13\rightarrow}\rangle | j - i = k + 1, i, j \in [0, m]\}$. Each of these requests results from a green path labeled with $\left((\mathbf{A}_H^C) + (\mathbf{B}_H^C) + (\mathbf{A}_V^C) + (\mathbf{B}_V^C)\right) y^C$ connecting t and b', for some vertex t created by language 10.
  Notice that there are no other requests gener-

ated since by Lemma 3 after applying strategy S_{color} each edge labeled with $G(y^C)$ has its counterpart labeled with $R(y^W)$
All generated requests have to be satisfied with $R(y^W)$ by Principle II.
- S_{color}: adds edges with labels $G(x^C)$ and $G(y^C)$ to edges added by languages 12 and 13.

□

Lemma 4. *For all $k, m \in \mathbb{N}, k < m$ strategy S^1_{layer} applied to $\mathbb{P}_k$ forces the Fugitive to build $\mathbb{L}^1_k$, strategy S^{odd} (for $k+1$ odd) and S^{even} (for $k+1$ even) applied to $\mathbb{L}^k_m$ forces the Fugitive to build $\mathbb{L}^{k+1}_m$.*

Proof. Similar analysis to that in Lemma 2 and Lemma 3 can be applied here. Structures $\mathbb{P}_m$ and ${}^{\$}\mathbb{P}_m$ differ by only two edges labeled with $R(\$^W)$ and $G(\$^C)$. Letters $\$^C$ and $\$^W$ occur only in languages $Q^{5,15}_{good}$ and this languages didn't generate any request in the process of building ${}^{\$}\mathbb{L}^{k+1}_m$ from ${}^{\$}\mathbb{L}^k_m$ in the proof of Lemma 3 and building ${}^{\$}\mathbb{L}^1_m$ from ${}^{\$}\mathbb{P}_m$ in the proof of Lemma 2. □

Lemma 5. *For all $m \in \mathbb{N}$ strategy S^m_{layer} forces the Fugitive to build ${}^{\$}\mathbb{L}^m_m$ from ${}^{\$}\mathbb{P}_m$ and $\mathbb{L}^m_m$ from $\mathbb{P}_m$..*

Proof. That is an easy consequence of Lemmas 2,3,4 and the definition of S^m_{layer}. □

Observation XIV.6. *By Exercise XIII.5 Lemma 5 proves Lemma 1.*

XV. And now we finally see the shades again

Now we are ready to finish the proof of Lemma 7.

First assume the original instance of Our Grid Tilling Problem has no *proper shading*.

The following is straightforward from König's Lemma:

Lemma 1. *If an instance I of OGTP has no proper shading then there exist natural m such that for any $k \geq m$ a square grid of size k has no shading that satisfies conditions (a1), (a2), (b1) and (b3) of proper shading.*

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

Now we consider two cases:

- If $\mathbb{G}_{m+1}$ was built then clearly condition (b3) of Definition 5 is unsatisfied. But this implies that a path labeled with a word from one of the languages Q_{bad} occurs in $\mathbb{H}$ between a and b, which is in breach with Principle III because of language Q^1_{bad}.
- If ${}^{\$}\mathbb{G}_k$ for $k \leq m$ was built then clearly condition (b2) or (b3) of Definition 5 is unsatisfied. This is because we assumed that there is no *proper shading*. But this implies that a path labeled with a word from one of the languages Q_{bad} occurs in $\mathbb{H}$ between a and b, which is in breach with Principle III because of language Q^1_{bad}.

This ends the proof of Lemma 7 (ii).

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

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

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

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;

[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;