Frontiers of tractability for recursive queries

Filip Mazowiecki

22 września 2015

Rozprawa poświęcona jest badaniu języków zapytań z rekurencją, dla których klasyczne problemy decyzyjne są na granicy rozstrzygalności. Zajmujemy się problemem spełnialności fragmentów logiki pierwszego rzędu rozszerzonych o dodatkowe operatory, oraz problemami równoważności, zawierania i ograniczoności dla datalogu. Wyniki przedstawione w rozprawie są oparte o prace [7, 13, 38, 39].

1 Wprowadzenie

1.1 Zapytania

W niniejszej rozprawie wszystkie problemy są oparte o zapytania do baz danych. Formalnie zapytanie przypisuje strukturom zbiór krotek. Rozważmy bazę danych D dla filmów z relacjami: *actor(name), movie(title)* (relacje unarne), *played(actor, movie)* (relacje binarne). Rozważmy następujące zapytanie Q: "wymień wszystkich aktorów, którzy grali w jednym filmie z Kevinem Baconem". Zdefiniujemy to zapytanie w trzech językach: SQL, algebrze relacyjnej i logice pierwszego rzędu (FO).

W SQL zapytanie Q można zapisać następująco:

```
SELECT P1.actor
FROM played P1
WHERE EXISTS     (SELECT P2.movie
                  FROM played P2
                  WHERE P1.movie = P2.movie AND
                        P2.actor = 'Kevin Bacon');
```

W algebrze relacyjnej zapytanie Q jest zdefiniowane przez poniższą formułę:

$$\pi_{actor}(played \bowtie_C \rho_{played'(actor',movie')}(played))$$

gdzie C jest warunkiem:

$$movie = movie' \,\text{AND}\, actor' = \text{'Kevin Bacon'}.$$

W FO zapytanie Q jest zdefiniowane następująco:

$$\{x \mid D \models \varphi(x)\},$$

gdzie

$$\varphi(x) = \mathit{actor}(x) \wedge \big(\exists y\ \mathit{movie}(y) \wedge \mathit{played}(\text{Kevin Bacon}, y) \wedge \mathit{played}(x, y)\big).$$

Powszechnie wiadomo, że FO, algebra relacyjna i ograniczony fragment SQL (bez agregacji i bez operacji arytmetycznych) są równoważne [15]. W tej rozprawie pracujemy z językami związanymi z FO, rozszerzonymi o rekursję. Rozstrzygamy złożoność problemów takich jak spełnialność, równoważność lub ograniczoność, które są naturalnymi teoretycznymi pytaniami w tym kontekście. Z drugiej strony te problemy występują też w praktyce. Wiele problemów optymalizacyjnych bazodanowych redukuje się do sprawdzania spełnialności zapytania lub równoważności dwóch zapytań. Inne obszary zastosowań to weryfikacja, gdzie tak zwany problem *model checking* można zredukować do (nie)spełnialności zapytań.

1.2 Fragmenty FO

Spełnialność dla FO jest nierozstrzygalna. Klasyczne wyniki Churcha i Turinga pokazują że problem spełnialności dla logiki pierwszego rzędu jest nierozstrzygalny. Celem tej rozprawy jest znalezienie formalizmów o charakterze logicznym, w który można wyrazić wiele ciekawych i przydatnych zapytań, a z drugiej strony problemy decyzyjne pozostaną rozstrzygalne. Zdefiniujemy pierwszy problem decyzyjny, o którym już wspomnieliśmy.

Problem 1 (Spełnialność)**.** *Dla danego zapytania q rozstrzygnąć czy istnieje struktura $\mathcal{M}$, taka że $\mathcal{M} \models q$.*

Pierwszym krokiem jest znalezienie fragmentów FO, dla których spełnialność jest rozstrzygalna.

Spełnialność dla FO^2 jest rozstrzygalna. Problem spełnialności dla FO pozostaje nierozstrzygalny nawet jeśli liczba zmiennych jest ograniczona do trzech [27]. Sytuacja się zmienia dla FO z tylko dwiema zmiennymi (FO^2). Fragment FO^2 jest powiązany z innymi formalizmami, takimi jak logiki deskryptywne i modalne. Na przykład logika deskryptywna $\mathcal{ALC}$ może być wyrażona jako fragment FO^2 [5].

Na pierwszy rzut oka logika FO^2 wydaje się ograniczona, ale wiele można wyrazić przez wielokrotne używanie zmiennych. Na przykład można wyrazić formułę "istnieje wierzchołek, z którego istnieje ścieżka długości trzy".

$$\exists x\ \exists y E(x, y) \wedge \exists x\ E(y, x) \wedge \exists y\ E(x, y). \tag{1}$$

Przez wielokrotne używanie zmiennych łatwo uogólnić tę formułę dla dowolnej długości ścieżki. Z drugiej strony powszechnie wiadomo, że nie ma formuł, nawet w pełnej logice FO, które wyrażają: "istnieje wierzchołek ze ścieżkami o dowolnej długości", lub "istnieje cykl".

Rozstrzygalność problemu spełnialności dla FO^2 została pokazana przez Mortimera [41]. Dokładna złożoność ustalili później Grädel et al. [23].

Twierdzenie 1 ([23])**.** *Problem spełnialności dla* FO^2 *jest* NExpTime*-zupełny.*

Spełnialność dla $\forall^*$FO jest rozstrzygalna. Nierozstrzygalność spełnialności dla FO z trzema zmiennymi jest pokazana dla formuł postaci $\forall x \exists y \forall z \varphi$, gdzie φ nie używa kwantyfikatorów [27]. Przyjrzymy się kolejnemu fragmentowi FO, tym razem z ograniczoną kwantyfikacją. Rozważmy formuły postaci $\exists^* \forall^* \varphi$, gdzie φ nie używa kwantyfikatorów. Ten fragment jest znany jako klasa Bernaysa-Schönfinkela.

Twierdzenie 2 ([33])**.** *Problem spełnialności dla klasy Bernaysa-Schönfinkela jest* NExpTime*-zupełny.*

Łatwo zauważyć, że klasa Bernaysa-Schönfinkela jest równoważna fragmentowi $\forall^* \varphi$ ze stałymi – wystarczy usunąć kwantyfikatory egzystencjalne i zastąpić je stałymi. Oznaczamy ten fragment przez $\forall^*$FO.

NExpTime-trudność w Twierdzeniu 2 jest zależna od tego, że relacje mogą być dowolnej arności. W tej rozprawie pracujemy z relacjami o arności 1 lub 2 więc możemy założyć, że relacje są ograniczonej arności. Z tym ograniczeniem nie trudno zauważyć, że problem spełnialności dla $\forall^*$FO jest w PSpace.

Równoważność i zawieranie dla UCQ **są rozstrzygalne.** Innym kierunkiem upraszczania FO jest usunięcie negacji z logiki. Ciekawym fragmentem są tak zwane *conjunctive queries* (CQ), które są formułami FO postaci $\exists x_1 \ldots \exists x_k \varphi$, gdzie φ jest koniunkcją atomów. Rozszerzenie na skończone sumy jest nazywane *unions of conjunctive queries* (UCQ). W Sekcji 1.1 wspomnieliśmy, że FO jest równoważne algebrze relacyjnej. Język CQ jest równoważny fragmentowi *select-project-join* algebry relacyjnej. Żeby wyrazić UCQ potrzebna jest jeszcze operacja sumy.

Bez negacji problem spełnialności często staje się trywialny i zazwyczaj w tym kontekście inne problemy decyzyjne są rozważane.

Problem 2 (Równoważność)**.** *Dla danych dwóch zapytań* q *i* q' *rozstrzygnąć czy dla każdej struktury* $\mathcal{M}$ *zachodzi* $\mathcal{M} \models q \iff \mathcal{M} \models q'$*.*

Problem 3 (Zawieranie)**.** *Dla danych dwóch zapytań* q *i* q' *rozstrzygnąć czy dla każdej struktury* $\mathcal{M}$ *zachodzi* $\mathcal{M} \models q \implies \mathcal{M} \models q'$*.*

Nie mając ograniczeń na negację te problemy można łatwo zredukować do problemu spełnialności. Na przykład problem zawierania się zapytania q w zapytaniu q' jest równoważny niespełnialności $q \wedge \neg q'$.

Dla CQ i UCQ równoważność i zawieranie są bardziej skomplikowanymi problemami. Chandra i Merlin w [12] rozwiązują ten problem wprowadzając *twierdzenie o homomorfizmie*, podstawowe narzędzie przy rozwiązywaniu problemów dla zapytań z ograniczoną negacją.

Twierdzenie 3 ([12])**.** *Problemy równoważności i zawierania dla* CQ *i* UCQ *są* NP*-zupełne.*

Ograniczenia FO^2**,** $\forall^*\mathrm{FO}$ **i** UCQ**.** Logiki: FO^2, $\forall^*\mathrm{FO}$, i UCQ, są punktami wyjściowymi tej pracy. Można w nich wyrazić wiele zapytań. Na przykład formuła (1) na stronie 2 jest w FO^2. Natomiast w Sekcji 1.1 żeby zdefiniować zapytanie "wymień wszystkich aktorów, którzy grali w jednym filmie z Kevinem Baconem" potrzebowaliśmy tylko egzystencjalnego kwantyfikatora, nie użyliśmy negacji i były nam potrzebne tylko dwie zmienne (ale potrzebowaliśmy stałej 'Kevin Bacon').

Z drugiej strony jest wiele ciekawych zapytań, których nie da się wyrazić w UCQ, FO^2 i $\forall^*\mathrm{FO}$. Na przykład "Czy jest połączenie pociągiem z Londynu do Neapolu?". Jest to spowodowane brakiem *rekursji*, nawet w pełnej logice FO. Logika pierwszego rzędu może być rozszerzona o rekursję na różne sposoby, od operatorów domknięcia przechodniego do silniejszych operatorów punktu stałego. W pozostałej części tej sekcji przedstawimy znane wyniki dotyczące rozszerzeń logik FO^2, $\forall^*\mathrm{FO}$ i UCQ.

1.3 FO^2 i $\forall^*\mathrm{FO}$ z domknięciem przechodnim

Wiele rozszerzeń logik FO^2 i $\forall^*\mathrm{FO}$ było studiowanych, na przykład rozszerzenie o *operator domknięcia na relację równoważności* [31, 30] lub *operator domknięcia przechodniego* [29, 47]. W tej rozprawie badamy rozszerzenie logik z pewną odmianą domknięcia przechodniego. Ograniczamy się do sygnatur tylko z relacjami unarnymi i binarnymi. Operatory domknięcia są wyrażane przez dodanie do sygnatury dodatkowych symboli relacyjnych, które mają odpowiednią interpretację. Przez dodanie domknięcia przechodniego do logiki rozumiemy, że dla każdej binarnej relacji E mamy dostęp do relacji E^+ interpretowanej jako domknięcie przechodnie E.

Dla logik rozszerzonych o operatory domknięcia często się rozważa sygnatury z ograniczoną liczbą relacji binarnych. Bez takich ograniczeń problem spełnialności często jest nierozstrzygalny. Rozważmy logikę FO^2 rozszerzoną o operator domknięcia przechodniego. Już z dwiema relacjami binarnymi problem spełnialności staje się nierozstrzygalny. Wynika to z nierozstrzygalności FO^2 z dwiema relacjami domkniętymi na relację przechodniości [32, 28]. Przypadek kiedy jest tylko jedna relacja binarna jest problemem

otwartym, ale znane są częściowe pozytywne wyniki. Szwast i Tendera rozważają przypadek gdy jest tylko jedna relacja binarna, która jest interpretowana jako relacja domknięta na relację przechodniości.

Twierdzenie 4 ([47]). *Problem spełnialności dla* FO^2 *z jedną relacją binarną domkniętą na relację przechodniości jest w 2-*NExpTime.

Kieroński i Michaliszyn uzyskali inny pozytywny wynik w [29]. Niech $\forall^2_{TC}$ oznacza fragment $\forall^*$FO z tylko dwiema zmiennymi i jedną relacją binarną wraz z jej domknięciem przechodnim.

Twierdzenie 5 ([29]). *Problem spełnialności dla* $\forall^2_{TC}$ *jest w 2-*NExpTime.

Dodanie trzeciej zmiennej lub drugiej relacji binarnej wraz z jej domknięciem przechodnim powoduje, że problem staje się nierozstrzygalny.

1.4 Datalog: UCQz operatorem punktu stałego

Podstawowe definicje. Datalog jest językiem uzyskanym przez dodanie operatora punktu stałego do UCQ (dokładny opis można znaleźć w [2, 10]). *Program datalogu* $\mathcal{P}$ nad sygnaturą S jest zbiorem reguł postaci

$$head \leftarrow body,$$

gdzie *head* jest atomem z S, a *body* jest koniunkcją atomów z S wypisanych po przecinku. Symbole relacyjne używane w części *head* reguł są nazywane *predykatami intensjonalnymi*. Są to nowe predykaty, które program dodaje do bazy danych. Jeden z predykatów intensjonalnych jest wyróżniony, nazywamy go *docelowym* predykatem. Przyjrzyjmy się przykładowemu programowi $\mathcal{P}$ z dwiema regułami r_1, r_2:

$$\begin{aligned} r_1 &: \quad buys(X,Y) \leftarrow likes(X,Y) \\ r_2 &: \quad buys(X,Y) \leftarrow knows(X,Z), buys(Z,Y) \end{aligned} \tag{2}$$

Wejściem dla tego programu jest baza danych z relacjami binarnymi *likes* i *knows*. W pierwszym kroku tylko reguła r_1 może być użyta, dodaje ona $buys(X,Y)$ dla każdej pary (X,Y), dla której zachodzi $likes(X,Y)$. W kolejnych krokach program aplikuje drugą – rekurencyjną – regułę. Fakt $buys(X,Y)$ jest dodany do bazy danych jeśli istnieje wierzchołek Z, taki że zachodzi $knows(X,Z)$ i $buys(Z,Y)$. Program ewaluuje tak długo aż nie da się dodać żadnych nowych faktów do bazy danych. Formuły z fragmentu UCQ odpowiadają programom bez rekurencji, zdefiniowanym przez reguły takie jak r_1, które są aplikowane tylko w pierwszym kroku.

Program $\mathcal{P}$ z przykładu powyżej ma tylko jeden predykat intensjonalny *buys*. Wynikiem tego programu jest zbiór wszystkich faktów $buys(X,Y)$. Dla każdego programu $\mathcal{Q}$ i dla każdej bazy danych D będziemy oznaczać przez $\mathcal{Q}(D)$ zbiór wszystkich faktów na temat docelowego predykatu uzyskanych przez ewaluowanie programu $\mathcal{Q}$ na D.

Równoważność i zawieranie się programów. Problemy równoważności i zawierania dla datalogu są nierozstrzygalne [45]. Jedna z motywacji dla studiowania tych problemów pochodzi z typowych bazodanowych problemów, takich jak minimalizacja i optymalizacja zapytań. Jednym ze sposobów optymalizacji programów datalogu jest usunięcie rekurencji. Ponieważ programy nierekurencyjne odpowiadają fragmentowi UCQ, naturalne jest redukowanie programów datalogu do UCQ.

Twierdzenie 6 ([14])**.** *Równoważność danego programu datalogu i danego zapytania* UCQ*, oraz zawieranie się programu datalogu w danym zapytaniu* UCQ *są 3-*ExpTime*-zupełne.*

Dla datalogu problemy zawierania i równoważności wzajemnie redukują się do siebie. Niech $\mathcal{P}$ i $\mathcal{Q}$ będą dwoma programami datalogu. Żeby sprawdzić czy $\mathcal{P}$ i $\mathcal{Q}$ są równoważne wystarczy sprawdzić zawieranie w obie strony. Jeśli jesteśmy zainteresowani równoważności programu datalogu i danego UCQ to musimy dodatkowo umieć sprawdzać zawieranie się danego UCQ w programie datalogu. Zawieranie w tę stronę jest zazwyczaj prostsze do sprawdzenia, w szczególności zachodzi poniższe twierdzenie.

Twierdzenie 7 ([11, 17, 44])**.** *Problem zawierania się danego* UCQ *w danym programie datalogu jest* ExpTime*-zupełny.*

Przypuśćmy teraz, że umiemy sprawdzać równoważność programów i chcemy sprawdzić czy $\mathcal{P} \subseteq \mathcal{Q}$. To pytanie można zredukować do pytania czy programy $\mathcal{Q}$ i $\mathcal{Q} \vee \mathcal{P}$ są równoważne. W tej redukcji korzystamy z tego, że datalog jest domknięty na operację sumy. Ze względu na te redukcje skupimy się głównie na problemie zawierania.

Ograniczoność. Rozważmy program datalogu $\mathcal{P}$ z docelowym predykatem P. Przez $\mathcal{P}^i(D)$ oznaczamy zbiór faktów o P, które otrzymamy po i krokach programu $\mathcal{P}$ na bazie danych D. Wtedy oczywiście

$$\mathcal{P}(D) = \bigcup_{i \geqslant 0} \mathcal{P}^i(D).$$

Program $\mathcal{P}$ jest *ograniczony* jeśli istnieje taka liczba n, która zależy tylko od $\mathcal{P}$, taka że dla każdej bazy danych D zachodzi $\mathcal{P}(D) = \mathcal{P}^n(D)$. Intuicyjnie to oznacza, że głębokość rekursji nie zależy od wejściowej bazy danych. Na przykład UCQ są ograniczone i wystarczy wziąć $n = 1$.

Problem 4 (Ograniczoność)**.** *Dla danego programu datalogu $\mathcal{P}$ rozstrzygnąć czy $\mathcal{P}$ jest ograniczony.*

Problem ograniczoności jest nierozstrzygalny [21], więc zaczęto badać różne fragmenty datalogu. Ważne fragmenty to *monadyczne* programy datalogu, dla których predykaty intensjonalne są unarne; oraz *liniowe* programy

datalogu, dla których każda reguła zawiera co najwyżej jeden intensjonalny predykat. Przeanalizujmy program $\mathcal{P}$ zdefiniowany przez (2) na stronie 5. Program $\mathcal{P}$ jest liniowy ponieważ r_1 odwołuje sie tylko raz do predykatu *buys*, a r_2 w ogóle się nie odwołuje do intensjonalnych predykatów. Z drugiej strony $\mathcal{P}$ nie jest monadyczny ponieważ predykat *buys* jest binarny. Dla monadycznych programów datalogu wszystkie rozważane przez nas problemy stają się rozstrzygalne.

Twierdzenie 8 ([16, 8])**.** *Problem zawierania dla monadycznych programów datalogu jest 2-*ExpTime*-zupełny. Problem ograniczoności jest w 3-*ExpTime*. Przy dodatkowym założeniu liniowości problem zawierania jest* PSpace*-zupełny, a problem ograniczoności jest w* ExpSpace*.*

2 Wyniki: FO^2 i $\forall^*\mathrm{FO}$ z operatorem DTC

Zajmujemy się pewną odmianą domknięcia przechodniego zwanego *deterministycznym domknięciem przechodnim* (DTC). Niech E będzie relacją binarną. Oznaczamy przez $E' \subseteq E$ deterministyczną część E, czyli

$$(u, v) \in E' \text{ jeśli } (u, v) \in E \text{ i dla każdego } w \text{ jeśli } (u, w) \in E \text{ to } v = w.$$

Deterministyczne domknięcie przechodnie relacji E definiujemy jako domknięcie przechodnie relacji E'. Będziemy je oznaczać przez $\overline{E}$. Operator DTC pozwala na wyrażenie wielu ciekawych własności. Można wymusić, że relacja binarna jest liniowym porządkiem, drzewem (z odwróconymi krawędziami), lasem, lub funkcją częściową.

Pokażemy jak w logice FO^2 z jedną relacją binarną E i jej deterministycznym domknięciem przechodnim można wymusić żeby model dla formuły był drzewem. Zacznijmy od wymuszenia unikatowego korzenia, z etykietą R.

$$\exists x(R(x) \wedge \forall y(R(y) \rightarrow x = y) \wedge \forall y \neg E(x, y)).$$

Następnie wymuszamy, że pozostałe elementy mają deterministyczną ścieżkę do korzenia:

$$\forall xy(R(x) \wedge \neg R(y) \rightarrow \overline{E}(y, x)).$$

Łatwo sprawdzić, że modele które spełniają te formuły mogą być interpretowane jako drzewa z odwróconymi krawędziami. Konkretnie relacja E jest interpretowana jako $\uparrow$, a $\overline{E}$ jako $\uparrow^+$. Takie drzewa mogą być nieskończone, co więcej wydaje się problematyczne żeby wymusić drzewo skończone bez jawnego założenia skończoności modelu. Jeśli dodamy drugi unikatowy wierzchołek n z etykietą R_2 i wymusimy, że istnieje deterministyczna ścieżka od n do wszystkich pozostałych wierzchołków to uzyskamy modele, które można interpretować jako skończone słowa.

W pracy [25] Immerman pokazał, że na uporządkowanych strukturach logika pierwszego rzędu rozszerzona o DTC jest równoważna LogSpace.

W [24], Grädel, Otto i Rosen udowodnili, że FO^2 poszerzona o operator DTC staje się nierozstrzygalna. W [26], Immerman et al. pokazali fragment FO poszerzony o DTC, dla którego problem spełnialności jest rozstrzygalny.

Twierdzenie 9 ([26])**.** *Problem spełnialności dla* $\forall^*\mathrm{FO}$ *z jedną relacja binarną* E *i pozytywnymi wystąpieniami* $\overline{E}$ *jest* NExpTime*-zupełny.*

Pozostała część tej sekcji jest podzielona na dwie części. W pierwszej przedstawimy wyniki w tej rozprawie dotyczące problemu spełnialności FO^2 i $\forall^*\mathrm{FO}$ rozszerzonych o operator DTC. W drugiej części przedstawimy wyniki dotyczące FO^2 na drzewach. Można o tym myśleć jako dogłębniejszej analizie specjalnego przypadku z pierwszej części. Przykład z początku tej sekcji pokazał, że w FO^2 z DTC można wymusić żeby model był drzewem.

2.1 FO^2 i $\forall^*\mathrm{FO}$ z deterministycznym domknięciem przechodnim

Wyniki z rozprawy przedstawione w tej sekcji pochodzą z [13]. Zacznijmy od dokładniejszej analizy wyników z [24, 26]. Oznaczamy przez $\mathrm{FO}^2 + \mathrm{DTC}(E_1, \ldots, E_k)$ logikę FO^2 z k relacjami binarnymi i ich deterministycznymi domknięciami przechodnimi. Przez $\forall^m + \mathrm{DTC}(E_1, \ldots, E_k)$ oznaczamy uniwersalny fragment FO z m zmiennymi i k relacjami binarnymi oraz ich deterministycznymi domknięciami przechodnimi.

Wynik o nierozstrzygalności FO^2 z operatorem DTC w [24] wymaga użycia wielu relacji binarnych. W [26] pokazano, że problem spełnialności dla logiki $\forall^4 + \mathrm{DTC}(E_1, E_2)$ jest nierozstrzygalny. Autorzy sugerują, że można poprawić tę redukcję tak, żeby użyć tylko dwóch zmiennych. Niestety nie ma formalnego dowodu, a konstrukcja którą opisują wydaje się wymagać dodatkowych relacji binarnych. W tej rozprawie dowodzimy formalnie, że spełnialność dla $\forall^2 + \mathrm{DTC}(E_1, E_2)$ jest nierozstrzygalna, korzystając z redukcji w pracach [34, 35]. Główny wynik tej części to, że jeśli ograniczymy się do jednej relacji binarnej w sygnaturze to spełnialność staje się rozstrzygalna nawet dla pełnej logiki FO^2.

Twierdzenie 10. *Problem spełnialności* $\mathrm{FO}^2 + \mathrm{DTC}(E)$ *jest* ExpSpace*-zupełny. Spełnialność* $\forall^2 + \mathrm{DTC}(E)$ *jest* NExpTime*-zupełna.*

Porównajmy logiki z Twierdzenia 10 oraz logikę z Twierdzenia 9. Nasze logiki mają ograniczoną liczbę zmiennych do dwóch, ale za to pozwalają na negatywne użycie operatora DTC, oraz w logice $\mathrm{FO}^2 + \mathrm{DTC}(E)$ można używać kwantyfikatorów w nieograniczony sposób.

Na koniec prezentujemy jeszcze kilka wyników dotyczących $\forall^*\mathrm{FO}$ z jedną binarną relacją i jej deterministycznym domknięciem przechodnim, ale z więcej niż dwiema zmiennymi. W [26] autorzy pokazują, że problem spełnialności dla $\forall^*\mathrm{FO}$ z jedną relacją binarną i jej deterministycznym domknięciem przechodnim jest nierozstrzygalny. Formuła użyta w redukcji jest z logiki

$\forall^5 + \mathrm{DTC}(E)$. W tej rozprawie pokazujemy, że można tę redukcję poprawić do formuły z $\forall^4 + \mathrm{DTC}(E)$. Zaskakująco, zredukowanie liczby zmiennych do trzech nie jest już proste i zostawiamy to jako problem otwarty. Wynik pokazujący rozstrzygalność spełnialności dla $\forall^3 + \mathrm{DTC}(E)$ byłby zaskakujący, ponieważ zazwyczaj granica rozstrzygalności jest pomiędzy dwiema a trzema zmiennymi.

2.2 FO^2 na drzewach

Wyniki z rozprawy przedstawione w tej sekcji pochodzą z [7]. W tej sekcji rozważamy modele, które są skończonymi drzewami lub skończonymi słowami. Na takich strukturach często się przyjmuje, że każdy wierzchołek musi spełniać dokładnie jeden predykat unarny, będziemy nazywali to *ograniczeniem unarnego alfabetu* (UAR). Ponieważ rozważamy także struktury, w których w jednym wierzchołku może być spełnionych kilka predykatów unarnych, to będziemy pisać *słowa UAR* oraz *drzewa UAR* żeby zaznaczyć, że UAR jest założone.

Na słowach i drzewach spełnialność jest rozstrzygalna nawet dla monadycznej logiki drugiego rzędu (MSO). Z drugiej strony Stockmeyer [46] pokazał nieelementarne ograniczenie dolne nawet dla problemu spełnialności dla FO. Jeśli chodzi o FO^2 to na słowach wyniki są podobne jak na ogólnych strukturach. Etessami et al. pokazali, że problem spełnialności jest NExpTime-zupełny [18]. Na strukturach takich jak słowa często się rozważa fragmenty z ograniczoną liczbą binarnych relacji. W [18] autorzy pokazują, że NExpTime jest górnym ograniczeniem dla pełnej logiki FO^2, to znaczy z relacjami następnika *succ* i jego domknięcia przechodniego $<$. Dolne ograniczenie NExpTime jest pokazane dla unarnego FO^2, czyli dla fragmentu bez relacji binarnych.

W tej rozprawie skupiamy się na drzewach. Niech $\mathrm{FO}^2[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$ oznacza FO^2 z następującymi relacjami binarnymi $\downarrow, \downarrow_+, \rightarrow, \rightarrow^+$, które interpretujemy jako relacje dziecka, następnego brata i ich domknięć przechodnich. Dla każdego podzbioru $\tau \subseteq \{\downarrow, \downarrow_+, \rightarrow, \rightarrow^+\}$ przez $\mathrm{FO}^2[\tau]$ oznaczamy logikę z sygnaturą, w której relacje binarne są ograniczone do τ. Marx i De Rijke pokazali, że $\mathrm{FO}^2[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$ jest równoważna nawigacyjnemu fragmentowi XPath [37]. Z pracy Marxa [36] wynika, że problem spełnialności dla XPath jest ExpTime-zupełny. Ponieważ tłumaczenie $\mathrm{FO}^2[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$ na XPath w [37] wymaga wykładniczo większej formuły, dostajemy górne ograniczenie 2-ExpTime na złożoność problemu spełnialności dla $\mathrm{FO}^2[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$.

W tej sekcji rozważamy problem spełnialności dla następujących fragmentów i wariantów $\mathrm{FO}^2[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$:

- pełna logika: $\mathrm{FO}^2[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$;
- ograniczenie binarnych relacji: $\mathrm{FO}^2[\tau]$ dla $\tau \subseteq \{\downarrow, \downarrow_+, \rightarrow, \rightarrow^+\}$;

- ograniczone modele: drzewa UAR;
- ograniczenia kwantyfikatorów: *strzeżony fragment.*

W strzeżonym fragmencie pozwalamy na użycie kwantyfikatorów tylko jeśli zmienne są połączone relacjami binarnymi. Strzeżony fragment logiki FO^2 oznaczamy przez GF^2. Wszystkie uniwersalne kwantyfikatory są postaci

$$\forall y\,(\alpha(x,y) \Rightarrow \varphi(x,y))$$

egzystencjalne kwantyfikatory mają postać

$$\exists y\,(\alpha(x,y) \wedge \varphi(x,y)),$$

gdzie α jest atomową relacją (więcej szczegółów można znaleźć w [4]). Dla strzeżonych fragmentów z ograniczoną sygnaturą będziemy używać podobnego oznaczenia jak w ogólnym przypadku, na przykład $\mathrm{GF}^2[\downarrow_+]$ jest strzeżonym fragmentem $\mathrm{FO}^2[\downarrow_+]$.

Pełna logika. Głównym wynikiem w tej sekcji jest ustalenie dokładnej złożoności dla pełnej logiki, poprawiając górne ograniczenie 2-ExpTime.

Twierdzenie 11. *Spełnialność dla* $\mathrm{FO}^2[\downarrow, \downarrow_+, \rightarrow, \rightarrow^+]$ *na drzewach jest* ExpSpace*-zupełna; dolne ograniczenie zachodzi już dla logiki* $\mathrm{GF}^2[\downarrow_+]$.

W pozostałej części tej sekcji będziemy analizowali jak różne ograniczenia wpływają na złożoność problemu spełnialności. Omówimy tylko najciekawsze wyniki z relacją $\downarrow_+$ w sygnaturze. Pełne podsumowanie wyników znajduje się na Rysunku 1 na stronie 11.

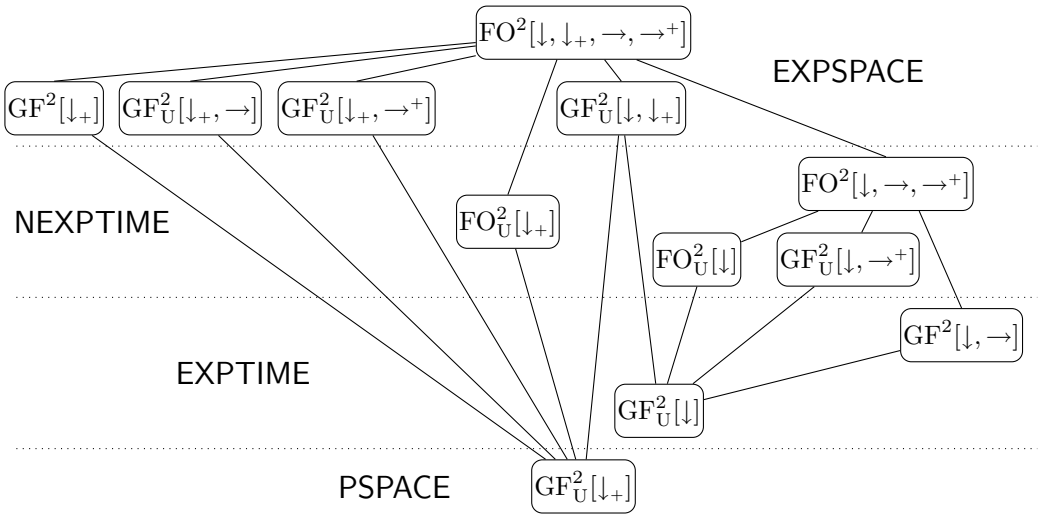
Drzewa UAR. Rozważmy modele z założeniem UAR. Zauważmy, że UAR można łatwo wyrazić w FO^2 w każdej z sygnatur, które rozważamy. W szczególności nie trzeba używać predykatów binarnych. Stąd wszystkie górne ograniczenia dla spełnialności na ogólnych drzewach przenoszą się na drzewa UAR. Analogicznie wszystkie dolne ograniczenia dla spełnialności na drzewach UAR przenoszą się na ogólne drzewa. Weiss pokazał, że problem spełnialności dla $\mathrm{FO}^2[succ]$ pozostaje NExpTime-zupełny dla słów UAR, ale dla $\mathrm{FO}^2[<]$ złożoność spełnialności staje się NP-zupełny [48]. W tej rozprawie pokazujemy, że na drzewach złożoność też spada, ale nie aż tak drastycznie.

Twierdzenie 12. *Problem spełnialności dla* $\mathrm{FO}^2[\downarrow_+]$ *dla drzew UAR jest* NExpTime*-zupełny.*

Ponadto pokazujemy, że jeśli rozszerzymy $\mathrm{FO}^2[\downarrow_+]$, o któryś z operatorów $\downarrow, \rightarrow, \rightarrow^+$ to złożoność spełnialności rośnie do ExpSpace, nawet dla strzeżonych fragmentów.

Strzeżony fragment. W Twierdzeniu 12 spadek złożoności dla $\mathrm{FO}^2[\downarrow_+]$ przez założenie UAR jest mniej drastyczny niż dla $\mathrm{FO}^2[<]$ dla słów. Żeby uzyskać NExpTime-trudność formuła musi mówić o parach x, y takich, że y nie jest ani potomkiem ani przodkiem x. Dlatego zbadaliśmy problem spełnialności dla strzeżonego fragmentu. Z Twierdzenia 11 problem spełnialności dla $\mathrm{GF}^2[\downarrow_+]$ na drzewach (bez UAR) jest ExpSpace-zupełny. Ograniczając się do drzew UAR otrzymujemy podobny spadek złożoności jak w przypadku słów.

Twierdzenie 13. *Problem spełnialności dla* $\mathrm{GF}^2[\downarrow_+]$ *na drzewach jest* PSpace*-zupełny.*



Rysunek 1: Złożoności dla FO^2 na drzewach

Podsumowanie wyników Przypomnijmy, że UAR można łatwo wymusić dla FO^2 w każdej z rozważanych przez nas sygnatur. Oczywiście dla każdego τ_{bin}, logika $\mathrm{GF}^2[\tau_{bin}]$ jest fragmentem $\mathrm{FO}^2[\tau_{bin}]$ i dla każdych $\tau_{bin} \subseteq \tau'_{bin}$, $\mathrm{FO}^2[\tau_{bin}]$ jest fragmentem $\mathrm{FO}^2[\tau'_{bin}]$, oraz $\mathrm{GF}^2[\tau_{bin}]$ jest fragmentem $\mathrm{GF}^2[\tau'_{bin}]$.

Na Rysunku 1 prezentujemy wyniki w formie diagramu, w którym ścieżka od logiki A do logiki B oznacza, że logika A jest fragmentem logiki B, albo logika A redukuje się do logiki B przez ograniczenie UAR. Dolnym indeksem 'U' oznaczamy, że logika jest rozpatrywana na drzewach UAR. Dla każdego wariantu, który jest minimalny w jednym z czterech regionów złożoności (PSPACE, EXPTIME, NEXPTIME, EXPSPACE) mamy odpowiednie dolne ograniczenie, oraz dla każdego maksymalnego fragmentu w swoim regionie mamy odpowiednie górne ograniczenie.

Łatwo sprawdzić, że jest to pełna analiza, to znaczy każdy z wariantów logik $\mathrm{FO}^2/\mathrm{GF}^2$ znajduje się gdzieś pomiędzy dwiema logikami, dla których mamy górne i dolne ograniczenia złożoności.

3 Wyniki: datalog na drzewach

W Sekcji 1.4 przedstawiliśmy znane wyniki dotyczące datalogu na ogólnych strukturach. W tej sekcji skupimy się na wynikach dotyczących datalogu na skończonych drzewach. Rozumiemy przez to, że w sygnaturze poza unarnymi relacjami są tylko relacje binarne dziecko $\downarrow$ i potomek $\downarrow_+$.

Na drzewach nawet problem spełnialności jest nierozstrzygalny [1]. Żeby uzyskać ten wynik potrzebne są binarne predykaty intensjonalne. Dlatego skupimy się tylko na monadycznych programach datalogu.

Monadyczne programy datalogu. Gottlob i Koch pokazali, że na drzewach datalog monadyczny jest równoważny MSO [22]. Z tego możemy od razy wywnioskować rozstrzygalność problemów takich jak zawieranie i równoważność programów datalogu. Dokładna złożoność problemu zawierania została pokaza w [20].

Twierdzenie 14 ([20])**.** *Na drzewach problem zawierania datalogu monadycznego jest* ExpTime*-zupełny.*

W tej rozprawie studiujemy datalog na drzewach z etykietami, które pochodzą z nieskończonego alfabetu. Rozszerzenie do nieskończonego alfabetu jest często badane w kontekście języków bazodanowych. Na przykład XPath, język zapytań dla XMLa powszechnie używany w praktyce i obszernie badany [9, 19, 40, 43]. Aby móc korzystać z tego rozszerzenia dodajemy do sygnatury relację binarną $\sim$, która zachodzi pomiędzy wierzchołkami o tej samej etykiecie. Zawsze kiedy pracujemy z nieskończonym alfabetem zakładamy, że $\sim$ jest w sygnaturze. Z nieskończonym alfabetem problem zawierania staje się nierozstrzygalny, nawet dla ograniczonej klasy monadycznych programów.

Twierdzenie 15 ([1])**.** *Problem zawierania się liniowych monadycznych programów datalogu w* UCQ *jest nierozstrzygalny.*

Pewne pozytywne wyniki udało się uzyskać dla ograniczonych modeli: na drzewach o ograniczonej głębokości problem zawierania monadycznego datalogu jest rozstrzygalny.

Założenie spójności. Niech r będzie regułą dla jakiegoś programu datalogu. Niech G_r będzie grafem, którego wierzchołkami są zmienne użyte w r, a krawędzie są pomiędzy X i Y jeśli w *body* r jest formuła $X{\downarrow}Y$ lub $X{\downarrow_+}Y$. Program $\mathcal{P}$ jest *spójny* jeśli dla każdej reguły $r \in \mathcal{P}$, graf G_r jest spójny.

Wcześniejsze prace o datalogu na ogólnych strukturach często zakładały spójność programów [16, 21]. Praktyczny powód jest taki, że "praktyczne programy" często są spójne i liniowe (cf. [6]). Założenie o spójności wydaje się tym bardziej naturalne jeśli naszymi modelami są drzewa, dlatego zakładamy spójność programów.

Do końca tej sekcji będziemy pisać w skrócie *program datalogu* mając na myśli *monadyczny i spójny program datalogu*. Będziemy oznaczać tę klasę przez Datalog($\downarrow, \downarrow_+$). Ponieważ programy nierekurencyjne odpowiadają UCQ, zakładamy, że UCQs też są spójne. Modelami są drzewa z etykietami, które pochodzą z nieskończonego alfabetu.

3.1 Zawieranie dla programów datalogu

Wyniki z rozprawy przedstawione w tej sekcji pochodzą z [38, 39].

Przyjrzyjmy się dokładniej Twierdzeniu 15, czyli nierozstrzygalności problemu zawierania linowych programów datalogu w UCQ. W tym wyniku są dwa ważne założenia.

1. Redukcja wymaga obu relacji binarnych $\downarrow, \downarrow_+$. Autorzy twierdzą, że można ją poprawić tak żeby działała nawet bez $\downarrow$ w sygnaturze. To pozostawia otwarty problem dla programów bez relacji $\downarrow_+$.

2. Redukcja jest przeprowadzona na słowach i opiera się na tym, że wierzchołki mają jednoznacznego następnika. Na drzewach to nie jest prawdą, ponieważ dzieci nie muszą być jednoznaczne. Żeby ta redukcja działa na drzewach trzeba zmodyfikować programy, żeby szły w górę drzew, ponieważ relacja rodzica już jest jednoznaczna.

W ten sposób otrzymujemy dwie naturalne drogi, żeby ograniczyć programy i liczyć na rozstrzygalność problemu zawierania. Pierwsza droga to rozważyć programy z sygnaturą, w której są tylko dwie relacje binarne $\downarrow$ i $\sim$. Oznaczamy tę klasę programów przez Datalog($\downarrow$). Przez UCQ($\downarrow$) oznaczamy odpowiadającą klasę programów nierekurencyjnych. Druga droga to rozważyć *programy zniżkowe* zdefiniowane przez reguły, którym nie wolno iść w górę drzewa. Formalnie, program $\mathcal{P}$ jest *zniżkowy* jeśli dla każdej reguły $r \in \mathcal{P}$, graf G_r jest skierowanym drzewem, którego korzeń jest zmienną użytą w *head* reguły r. Zauważmy, że programy zniżkowe są zawsze spójne. Aby oznaczyć ograniczenie do programów zniżkowych będziemy używali litery D, na przykład D-Datalog($\downarrow, \downarrow_+$) to programy zniżkowe.

Nasze wyniki dotyczą problemu zawierania dla programów datalogu bez potomka lub zniżkowych. Najpierw zajmujemy się przypadkiem drzew urangowanych, to znaczy takich dla których liczba dzieci jest ograniczona przez pewną stałą R (w szczególności dla $R = 1$ są to słowa). Na drzewach urangowanych problem zawierania pozostaje nierozstrzygalny dla programów zniżkowych, ale jest rozstrzygalny dla programów bez potomka. Następnie zaj-

mujemy się przypadkiem drzew nieurangowanych. Zaskakująco, w tym przypadku wyniki są odwrotne: zawieranie jest nierozstrzygalne dla programów bez potomka, ale rozstrzygalne dla programów zniżkowych. Są to dosyć nietypowe wyniki, zazwyczaj ograniczanie się do drzew urangowanych niewiele zmienia złożoność problemu, albo nawet w ogóle jej nie zmienia. Tutaj w obu przypadkach mamy różnicę w kategorii rozstrzygalności.

Zawieranie dla drzew urangowanych. Pokazujemy, że problem zawierania jest nierozstrzygalny dla programów zniżkowych. Ograniczając się do programów bez potomka odzyskujemy rozstrzygalność.

Twierdzenie 16. *Problem zawierania*

1. *na drzewach urangowanych i słowach jest nierozstrzygalny dla liniowych programów* D-Datalog($\downarrow, \downarrow_+$)*;*
2. *na słowach jest w* ExpSpace *dla* Datalog($\downarrow$) *i* PSpace*-zupełny dla liniowych programów* Datalog($\downarrow$)*;*
3. *na urangowanych drzewach jest w 3-*ExpTime *i 2-*ExpTime*-zupełny dla liniowych programów* Datalog($\downarrow$)*.*

Dolne ograniczenia zachodzą nawet dla zawierania się w UCQ *(z odpowiadającą sygnaturą).*

W appendiksie [38] pokazujemy Twierdzenie 16 z zupełnymi złożonościami. Pokazujemy, że zawieranie dla Datalog($\downarrow$): na słowach jest PSpace-zupełne; na urangowanych drzewach jest 2-ExpTime-zupełne. Inaczej mówiąc, założenie liniowości nie zmienia złożoności.

Zawieranie dla drzew nieurangowanych. Dosyć niespodziewanie na nieurangowanych drzewach otrzymujemy nierozstrzygalność dla programów bez potomka.

Twierdzenie 17. *Na nieurangowanych drzewach zawieranie się programów* Datalog($\downarrow$) *w* UCQ($\downarrow$) *jest nierozstrzygalne.*

Twierdzenie 17 opiera się na symulowaniu relacji $\downarrow_+$ i przepisaniu dowodu Twierdzenia 15. Żeby symulować $\downarrow_+$ potrzebujemy mówić o nieliniowych programach. Ograniczając programy do liniowych odzyskujemy rozstrzygalność.

Twierdzenie 18. *Na nieurangowanych drzewach problem zawierania liniowych programów* Datalog($\downarrow$) *jest w 3-*ExpTime*. Zawieranie się liniowych programów* Datalog($\downarrow$) *w* UCQ($\downarrow$) *jest 2-*ExpTime*-zupełne.*

Pozostał już tylko problem zawierania dla programów zniżkowych.

Twierdzenie 19. *Na nieurangowanych drzewach problem zawierania dla* D-Datalog($\downarrow, \downarrow_+$) *jest 2-*ExpTime*-zupełny. Dla liniowych programów* D-Datalog($\downarrow, \downarrow_+$) *problem zawierania jest* ExpSpace*-zupełne. W obu przypadkach ograniczenie dolne zachodzi nawet dla problemu zawierania się programów w* UCQ*.*

Na koniec podkreślimy znaczenie wyników dla programów zniżkowych. Studiowanie tych programów jest motywowane przez *zapytania regularnych wzorców drzewiastych* (RTPQ), jest to niedawno przedstawiony formalizm z obszaru ActiveXML [3], który pokazujemy że jest równoważny liniowym programom zniżkowym. Stąd nasze wyniki dla programów zniżkowych dowodzą rozstrzygalności problemu zawierania dla RTPQ. Wcześniej znane były tylko częściowe wyniki dla RTPQ z ograniczonym porównywaniem danych.

Podsumowanie naszych wyników znajduje sie w Tabeli 1.

	Drzewa nieurangowane		Drzewa urangowane
	liniowe	nieliniowe	liniowe/nieliniowe
D-Datalog($\downarrow, \downarrow_+$)	ExpSpace	2-ExpTime	Undec.
Datalog($\downarrow$)	w 3-ExpTime	Undec.	2-ExpTime

Tabela 1: Złożoność problemu zawierania dla różnych fragmentów datalogu.

Zapytania boolowskie i unarne. Na programy datalogu można patrzeć jako na zapytania boolowskie lub unarne. Jeśli program datalogu $\mathcal{P}$ jest rozważany jako zapytanie unarne to rozumiemy przez to, że zwraca $\mathcal{P}(D)$. Zapytania boolowskie zwracają odpowiedź 'tak' jeśli $\mathcal{P}(D)$ jest niepusty i 'nie' w przeciwnym razie. Stąd problem zawierania dla unarnych zapytań oznacza $\mathcal{P}(D) \subseteq \mathcal{Q}(D)$ dla każdej bazy danych D. Natomiast dla boolowskich zapytań oznacza $\mathcal{P}(D) \neq \emptyset \implies \mathcal{Q}(D) \neq \emptyset$ dla wszystkich baz danych D.

Wszystkie zaprezentowane wyniki do tej pory były dla boolowskich zapytań. Używając znanych metod można zredukować problem unarnego zawierania do boolowskiego zawierania (niestety nie przez zwykłe przepisanie zapytań). Natomiast redukcja w drugą stronę wymaga relacji $\downarrow_+$ lub rekursji (intuicyjnie ponieważ używając $\downarrow_+$ lub rekursji można się przemieścić z każdego wierzchołka X do dowolnego wierzchołka Y). To oznacza, że jedyny przypadek, dla którego możemy liczyć na poprawienie wyników to zawieranie się programów w UCQ($\downarrow$).

W Twierdzeniu 17 pokazaliśmy, że zawieranie się programów Datalog($\downarrow$) w UCQ($\downarrow$) jest nierozstrzygalne, ale redukcja korzysta z założenia, że zapytania są boolowskie. Intuicyjnie program UCQ($\downarrow$) jest użyty żeby znaleźć błędy w kodowaniu, ograniczając się do unarnych zapytań możemy sprawdzać błędy tylko w bliskim otoczeniu wyróżnionego wierzchołka X. Dzięki zmianie zapytań na unarne udało nam się poprawić wynik w Twierdzeniu 18.

Twierdzenie 20. *Na nieurangowanych drzewach problem zawierania dla liniowych programów* Datalog(↓) *w* UCQ(↓) *jest* ExpSpace*-zupełny jeśli zapytania są unarne.*

Niestety dowód Twierdzenia 20 nie uogólnia się to nieliniowych programów. Problem rozstrzygalności zawierania się programów Datalog(↓) w UCQ(↓) dla unarnych zapytań pozostaje otwarty.

3.2 Ograniczoność programów datalogu

Wyniki z rozprawy przedstawione w tej sekcji pochodzą z [39].

Ograniczoność i UCQ-definiowalność. Przypomnijmy, że program datalogu $\mathcal{P}$ jest ograniczony jeśli liczba kroków potrzebnych do wydedukowania wszystkich faktów jest ograniczona przez uniwersalną stałą, która nie zależy od wejściowej bazy danych. Ważny przykład takich programów to UCQ. Piszemy, że program $\mathcal{P}$ jest UCQ-definiowalny jeśli istnieje UCQ $\mathcal{Q}$ równoważne $\mathcal{P}$. Jeśli program $\mathcal{P}$ jest bez potomka, to wymagamy dodatkowo żeby $\mathcal{Q} \in \text{UCQ}(\downarrow)$.

Łatwo pokazać, że jeśli program jest ograniczony to jest UCQ-definiowalny. Implikacja w drugą stronę nie jest zawsze prawdą. Dla programów datalogu (nawet bez założenia monadyczności lub spójności) na ogólnych strukturach wiadomo, że program jest ograniczony wtedy i tylko wtedy gdy jest UCQ-definiowalny [42]. Pokazujemy, że jest to też prawdą dla programów Datalog(↓) na drzewach.

Twierdzenie 21. *Niech* $\mathcal{P} \in \text{Datalog}(\downarrow)$*. Wtedy* $\mathcal{P}$ *jest ograniczony wtedy i tylko wtedy gdy jest* UCQ*-definiowalny.*

Dzięki Twierdzeniu 21 dla programów bez potomka problem ograniczoności można przetłumaczyć na problem UCQ-definiowalności.

Problem 5 (UCQ-definiowalność)**.** *Dla danego programu* $\mathcal{P}$ *rozstrzygnąć czy jest* UCQ*-definiowalny.*

Twierdzenie 21 pokazuje jak problem ograniczoności jest powiązany z problemem równoważności. Dla problemu równoważności programu datalogu i UCQ mamy dane zarówno program datalogu jak i UCQ. Dla problemu UCQ-definiowalności tylko program datalogu jest dany. W ogólności te problemy różnią się, nawet jeśli chodzi o rozstrzygalność. Przypomnijmy z Sekcji 1.4, że na ogólnych strukturach problem ograniczoności jest nierozstrzygalny [21], natomiast równoważność danemu UCQ jest rozstrzygalna [14].

Rozważmy poniższy program.

$$\mathcal{P} \qquad \begin{array}{l} P(X) \leftarrow X{\downarrow}Y, Q(Y) \\ Q(X) \leftarrow X{\downarrow}Y, Q(Y) \\ Q(X) \leftarrow b(X) \end{array}$$

Program $\mathcal{P} \in \text{Datalog}(\downarrow, \downarrow_+)$ nie jest ograniczony – znalezienie b w drzewie może zająć dowolnie wiele kroków. Program $\mathcal{P}'$, zdefiniowany przez

$$\mathcal{P}' \qquad P(X) \leftarrow X{\downarrow_+}Y, b(Y)\,,$$

jest UCQ równoważnym $\mathcal{P}$. Te programy pokazują, że dla programów Datalog$(\downarrow, \downarrow_+)$ UCQ-definiowalność nie implikuje ograniczoności.

Złożoność problemu ograniczoności. Pokazujemy, że problem ograniczoności dla linowego Datalog$(\downarrow, \downarrow_+)$ jest nierozstrzygalna na słowach, drzewach urangowanych i nieurangowanych. Otrzymujemy pozytywne wyniki dla programów bez potomka na słowach i drzewach urangowanych.

Twierdzenie 22. *Problem ograniczoności dla* Datalog$(\downarrow)$*:*

1. *jest w* PSpace *dla słów.*
2. *jest w 2-*ExpTime *dla urangowanych drzew.*

Pytanie o rozstrzygalność problemu ograniczoności dla programów bez potomka na drzewach nieurangowanych pozostaje otwarte, nawet dla programów liniowych.

Literatura

[1] Serge Abiteboul, Pierre Bourhis, Anca Muscholl, and Zhilin Wu. Recursive queries on trees and data trees. In *ICDT*, pages 93–104, 2013.

[2] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison Wesley, 1995.

[3] Serge Abiteboul, Balder ten Cate, and Yannis Katsis. On the equivalence of distributed systems with queries and communication. In *ICDT*, pages 126–137, 2011.

[4] Hajnal Andréka, Johan van Benthem, and Istvan Németi. Modal languages and bounded fragments of predicate logic. *J. Phil. Logic*, 27:217–274, 1998.

[5] Franz Baader, Ian Horrocks, and Ulrike Sattler. Description Logics. In Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter, editors, *Handbook of Knowledge Representation*, chapter 3, pages 135–180. Elsevier, 2008.

[6] François Bancilhon and Raghu Ramakrishnan. An amateur's introduction to recursive query processing strategies. In *ACM SIGMOD*, pages 16–52, 1986.

[7] Saguy Benaim, Michael Benedikt, Witold Charatonik, Emanuel Kieronski, Rastislav Lenhardt, Filip Mazowiecki, and James Worrell. Complexity of two-variable logic on finite trees. In *ICALP*, pages 74–88, 2013.

[8] Michael Benedikt, Pierre Bourhis, and Pierre Senellart. Monadic datalog containment. In *ICALP*, pages 79–91, 2012.

[9] Michael Benedikt, Wenfei Fan, and Floris Geerts. Xpath satisfiability in the presence of dtds. *J. ACM*, 55(2), 2008.

[10] Stefano Ceri, Georg Gottlob, and Letizia Tanca. *Logic programming and databases*. Springer-Verlag New York, Inc., 1990.

[11] Ashok K. Chandra, Harry R. Lewis, and Johann A. Makowsky. Embedded implicational dependencies and their inference problem. In *STOC*, pages 342–354, 1981.

[12] Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the 9th Annual ACM Symposium on Theory of Computing, May 4-6, 1977, Boulder, Colorado, USA*, pages 77–90, 1977.

[13] Witold Charatonik, Emanuel Kieronski, and Filip Mazowiecki. Decidability of weak logics with deterministic transitive closure. In *CSL-LICS*, pages 29:1–29:10, 2014.

[14] Surajit Chaudhuri and Moshe Y. Vardi. On the equivalence of recursive and nonrecursive datalog programs. In *PODS*, pages 55–66, 1992.

[15] E. F. Codd. Relational completeness of data base sublanguages. *In: R. Rustin (ed.): Database Systems: 65-98, Prentice Hall and IBM Research Report RJ 987, San Jose, California*, 1972.

[16] Stavros S. Cosmadakis, Haim Gaifman, Paris C. Kanellakis, and Moshe Y. Vardi. Decidable optimization problems for database logic programs (preliminary report). In *STOC*, pages 477–490, 1988.

[17] Stavros S. Cosmadakis and Paris C. Kanellakis. Parallel evaluation of recursive rule queries. In *PODS*, pages 280–293, 1986.

[18] Kousha Etessami, Moshe Y. Vardi, and Thomas Wilke. First-order logic with two variables and unary temporal logic. *Inf. Comput.*, 179(2):279–295, 2002.

[19] Diego Figueira. Satisfiability of downward xpath with data equality tests. In *PODS*, pages 197–206, 2009.

[20] André Frochaux, Martin Grohe, and Nicole Schweikardt. Monadic datalog containment on trees. In *Proceedings of the 8th Alberto Mendelzon Workshop on Foundations of Data Management*, 2014.

[21] Haim Gaifman, Harry G. Mairson, Yehoshua Sagiv, and Moshe Y. Vardi. Undecidable optimization problems for database logic programs. *J. ACM*, 40(3):683–713, 1993.

[22] Georg Gottlob and Christoph Koch. Monadic datalog and the expressive power of languages for web information extraction. *J. ACM*, 51(1):74–113, 2004.

[23] E. Grädel, P. Kolaitis, and M. Vardi. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic*, 3(1):53–69, 1997.

[24] E. Grädel, M. Otto, and E. Rosen. Undecidability results on two-variable logics. *Archiv für Mathematiche Logik und Grundlagenforschung*, 38(4-5):313–354, 1999.

[25] N. Immerman. Languages that capture complexity classes. *SIAM Journal of Computing*, 16:760–778, 1987.

[26] N. Immerman, A. Rabinovich, T. Reps, S. Sagiv, and G. Yorsh. The boundary between decidability and undecidability for transitive-closure logics. In *Computer Science Logic*, volume 3210 of *LNCS*, pages 160–174. Springer, 2004.

[27] A.S. Kahr, E.F. Moore, and H. Wang. Entscheidungsproblem reduced to the $\forall\exists\forall$ case. *Proc. Nat. Acad. Sci. U.S.A.*, 48:365–377, 1962.

[28] Y. Kazakov. *Saturation-based decision procedures for extensions of the guarded fragment*. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 2006.

[29] E. Kieronski and J. Michaliszyn. Two-variable universal logic with transitive closure. In *Computer Science Logic*, volume 16 of *LIPIcs*, pages 396–410. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

[30] E. Kieroński, J. Michaliszyn, I. Pratt-Hartmann, and L. Tendera. Two-variable first-order logic with equivalence closure. In *Logic in Computer Science*, pages 431–440. IEEE, 2012.

[31] E. Kieroński and M. Otto. Small substructures and decidability issues for first-order logic with two variables. *Journal of Symbolic Logic*, 77:729–765, 2012.

[32] Emanuel Kieroński. Results on the guarded fragment with equivalence or transitive relations. In *CSL*, pages 309–324, 2005.

[33] Harry R. Lewis. Complexity results for classes of quantificational formulas. *J. Comput. Syst. Sci.*, 21(3):317–353, 1980.

[34] A. Manuel. Two variables and two successors. In *MFCS*, pages 513–524, 2010.

[35] A. Manuel. *Counter automata and classical logics for data words.* PhD thesis, Homi Bhabha National Institute, India, 2012.

[36] Maarten Marx. XPath with conditional axis relations. In *EDBT*, pages 477–494, 2004.

[37] Maarten Marx and Maarten de Rijke. Semantic characterization of navigational XPath. In *TDM*, pages 73–79, 2004.

[38] Filip Mazowiecki, Filip Murlak, and Adam Witkowski. Monadic datalog and regular tree pattern queries. In *MFCS*, pages 426–437, 2014.

[39] Filip Mazowiecki, Joanna Ochremiak, and Adam Witkowski. Eliminating recursion from monadic datalog programs on trees. Accepted for MFCS, 2015.

[40] Gerome Miklau and Dan Suciu. Containment and equivalence for a fragment of xpath. *J. ACM*, 51(1):2–45, 2004.

[41] M. Mortimer. On languages with two variables. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 21:135–140, 1975.

[42] Jeffrey F. Naughton and Yehoshua Sagiv. A simple characterization of uniform boundedness for a class of recursions. *J. Log. Program.*, 10(3–4):233 – 253, 1991.

[43] Frank Neven and Thomas Schwentick. On the complexity of xpath containment in the presence of disjunction, dtds, and variables. *Logical Methods in Computer Science*, 2(3), 2006.

[44] Yehoshua Sagiv. Optimizing datalog programs. In *Foundations of Deductive Databases and Logic Programming.*, pages 659–698. Morgan Kaufmann, 1988.

[45] Oded Shmueli. Equivalence of datalog queries is undecidable. *J. Log. Program.*, 15(3):231–241, 1993.

[46] Larry J. Stockmeyer. *The Complexity of Decision Problems in Automata Theory and Logic*. PhD thesis, Massachusetts Institute of Technology, 1974.

[47] Wieslaw Szwast and Lidia Tendera. FO^2 with one transitive relation is decidable. In *STACS*, volume 20 of *LIPIcs*, pages 317–328. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.

[48] Philipp Weis. *Expressiveness and succinctness of first-order logic on finite words*. PhD thesis, University of Massachusetts Amherst, USA, 2011.