Wykład 3: Rekurencja

Czego nam jeszcze brakuje?

- Czy następujące zapytania da się wyrazić w FO/RA/SQLu?
  - `Połączenie(skąd, dokąd, operator, samolot)`

    Czy da się dolecieć z Lublina do Dublina jednym modelem samolotu?
  - `E(x,y)`

    Czy istnieje ścieżka w tym grafie z wierzchołka *s* do wierzchołka *t*?
- Można udowodnić, że się nie da!
  - Używa się do tego gier. Więcej na przedmiocie "Logika dla informatyków".
- Czego nam brakuje? Rekurencji lub iteracji.

Game of Life

- Gramy na grafie nieskierowanym, każdy węzeł jest komórką. Komórka jest żywa lub martwa.
- Przyjmijmy, że ewolucja jest opisana regułą:

    (+) martwa komórka o dwóch żywych sąsiadach ożywa.

  Ewolucja wg. reguły (+) jest *inflacyjna*: żywych komórek nie ubywa.

  System w końcu się ustabilizuje, niezależnie od konfiguracji początkowej.

  Liczba kroków jest liniowa zwn. rozmiar grafu.
- Dodajmy regułę:

    (-) żywa komórka, która ma co najmniej 3 żywych sąsiadów, umiera.

  Ewolucja wg. reguł (+) i (-) jest *nieinflacyjna*: żywych komórek może przybywać i ubywać.

  System może się nigdy nie ustabilizować (wskazać przykład!), ale w pewnym momencie zacznie oscylować (jest skończenie wiele konfiguracji, po pierwszym powtórzeniu cały segment będzie się powtarzał w nieskończoność).

Algebra relacji z iteracją

Game of Life z regułą (+):

```
A := Alive;
loop
    A := A ∪ π₁(σ_{1 = 3, 2 ≠ 4, 2 = 5, 4 = 6}(E×E×A×A));
endloop;
return A;
```

$$A := A \cup \pi_1(\sigma_{1 = 3,\, 2 \neq 4,\, 2 = 5,\, 4 = 6}(E \times E \times A \times A));$$

Game of Life z regułami (+) i (-):

```
A := Alive;
loop
    A := (A ∪ π₁(σ_{1 = 3, 2 ≠ 4, 2 = 5, 4 = 6}(E×E×A×A))
           − π₁(σ_{1 = 3, 1 = 5, 2 ≠ 4, 4 ≠ 6, 6 ≠ 2, 2 = 7, 4 = 8, 6 = 9}(E×E×E×A×A×A));
endloop;
return A;
```

$$A := (A \cup \pi_1(\sigma_{1 = 3,\, 2 \neq 4,\, 2 = 5,\, 4 = 6}(E \times E \times A \times A)) - \pi_1(\sigma_{1 = 3,\, 1 = 5,\, 2 \neq 4,\, 4 \neq 6,\, 6 \neq 2,\, 2 = 7,\, 4 = 8,\, 6 = 9}(E \times E \times E \times A \times A \times A));$$

Algebra relacji z iteracją

Algebra WHILE

Trasa := $\pi_{1,2,4}$**(Połączenie);**

loop

  Trasa := Trasa $\cup$ $\pi_{1,\,5,\,7}$**(**$\sigma_{2\,=\,4,\,3\,=\,7}$**(Trasa**$\times$**Połączenie));**

endloop;

return π_3**(**$\sigma_{1\,=\,`Lublin',\ 2\,=\,`Dublin'}$**(Trasa));**

Algebra WHILE+

Gwarantujemy zachowanie monotoniczne umawiając się, że przypisanie zawsze dodaje:

Trasa += $\pi_{1,2,4}$**(Połączenie);**

loop

  Trasa += $\pi_{1,\,5,\,7}$**(**$\sigma_{2\,=\,4,\,3\,=\,7}$**(Trasa**$\times$**Połączenie));**

endloop;

return π_3**(**$\sigma_{1\,=\,`Lublin',\ 2\,=\,`Dublin'}$**(Trasa));**

Złożoność

- Każde zapytanie algebry WHILE+ da się wyliczyć w czasie wielomianowym.
- Każde zapytanie algebry WHILE da się wyliczyć w pamięci wielomianowej (ale w czasie wykładniczym).
- Można pokazać, że jeśli wszystkie wartości używane w bazie pochodzą ze zbioru liniowo uporządkowanego (np. liczby, napisy, itd), to
  - każda własność sprawdzalna w czasie wielomianowym wyraża się zapytaniem WHILE+,
  - każda własność sprawdzalna w pamięci wielomianowej wyraża się w WHILE.
- Założenie uporządkowania jest naturalne w bazach danych, ale sztuczne w logice. Słynnym problemem otwartym łączącym logikę i teorię złożoności jest pytanie, czy można to założenie usunąć.
  - Hasło: logika dla PTIME.
  - Więcej na przedmiocie “Teoria modeli skończonych”.

Rekurencja w SQLu

Common table expression (CTE), w Oracle'u nazywane *recursive subquery factoring*.

```
WITH [RECURSIVE] Trasa(skąd, dokąd, samolot) AS (
    (SELECT skąd, dokąd, samolot FROM Połączenie)
        UNION ALL
    (SELECT Trasa.skąd, Połączenie.dokąd, Trasa.samolot
     FROM Trasa, Połączenie
     WHERE Trasa.dokąd = Połączenie.skąd
       AND Trasa.samolot = Połączenie.samolot)
)
SELECT samolot
FROM Trasa
WHERE skąd = 'Lublin' AND dokąd = 'Dublin';
```

- tylko inflacyjna rekurencja
- inkrementacyjny algorytm ewaluacji skutkuje nieco egzotyczną semantyką
- w standardzie SQL-99
- DB2 (IBM), Oracle (11g), Microsoft SQL Server, PostgreSQL (8.4)

FO z punktami stałymi

- Chciałoby się tak:

$$Path(x, y) \ = \ E(x, y) \lor \exists z \, Path(x, z) \land E(z, y)$$

- Ale co to znaczy? Które pary powinny należeć do relacji $Path$?
  - Jeśli przyjmiemy, że końce wszystkich ścieżek, to równoważność się zgadza.
  - Ale jeśli przyjmiemy, że wszystkie pary wierzchołków, to też się zgadza.

  Czyli wiele relacji $Path$ spełnia tę definicję. Którą wybrać? Tutaj wiadomo, ale ogólnie?

- Co na pewno musi być w $Path$? Na pewno całe E. Ale skoro tak, to również E^2,E^3, itd. Czyli każde "rozwiązanie" zawiera końce wszystkich ścieżek. Tzn. relacja, którą mamy na myśli, jest najmniejszym (w sensie inkluzji) rozwiązaniem tego równania.
- Stosujemy notację $\mu R.\ \phi(x_1, ..., x_n)$, gdzie ϕ może używać symbolu relacyjnego R, np.

$$\mu \, Path.\ E(x\ , y) \ \lor \ \exists z \, Path(x, z) \land E(z, y).$$

- Całego tego wyrażenia można teraz używać jak relacji:

$$\forall u \, \exists v \, (\mu \, Path.\ E(x\ , y) \ \lor \ \exists z \, Path(x, z) \land E(z, y))(u,\ v).$$

- W instancji I, term $\mu R.\ \phi(x_1, ..., x_n)$ oznacza najmniejsze rozwiązanie równania $R \ = \phi^{(I, R)}$.

Datalog

- Zapytanie/program to zbiór reguł postaci

$$A(\bar{x}) :- \ L_1(\bar{x}, \bar{y}),\ L_2(\bar{x}, \bar{y}),\ \ldots,\ L_m(\bar{x}, \bar{y}),$$

  gdzie $L_1(\bar{x}, \bar{y})$ to atomy lub negacje atomów, ze wskazaniem predykatu zawierającego wynik.

- Predykaty **ekstensjonalne** (wbudowane) tj. relacje ze schematu bazy, występują tylko po prawej stronie (ciało reguły).
- Predykaty **intensjonalne** (użytkownika), to relacje definiowane w programie, występujące po lewej stronie (nagłówek reguły).
- **Warstwowa negacja:** rekurencja nie przechodzi przez negację, tj. jeśli jakaś ścieżka odwołań tworzy cykl w grafie odwołań między predykatami, to na tej ścieżce nie może być negacji.
- Nasze ulubione zapytanie wygląda tak:

```
q():- Trasa('Lublin','Dublin',s).
Trasa(x,y,s) :- Połączenie(x,y,_,s).
Trasa(x,y,s) :- Trasa(x,z,s),Połączenie(z,y,_,s).
```

Uogólnienie twierdzenia Codda

Twierdzenie

- WHILE+ = FO z punktami stałymi dla formuł inflacyjnych = Datalog z warstwową negacją.
- WHILE = FO z punktami stałymi dla dowolnych formuł = Datalog bez ograniczeń.