Datalog

Moritz Kaufmann

1. Juni 2015

Technische Universität München

Zusammenfassung

1. Faktenbasis (EDB = extensionale Datenbasis)
2. + logische Herleitungsregeln

$\rightarrow$ Ableitung neuer Fakten (IDB = intensionale Datenbasis)

Datalog Beispiel

EDB

```
% kennt(PersA, PersB, Jahr)
kennt(michael,"Moritz", 2013).
kennt(simon,michael, 2013).
kennt(simon,andreas, 2014).
kennt("Moritz", simon, 2013).
kennt("Moritz", michael, 2013).
```

Herleitungsregeln

```
moritzBekannte(Pers) :- kennt(''Moritz'', Pers, _).
```

IDB

```
moritzBekannte(simon).
moritzBekannte(michael).
```

Datalog Syntax

Atomare Formeln: $q(A_1, ..., A_m).$

Regeln: $p(X_1, ..., X_m) :- q_1(A_{11}, ...A_{1s}), ..., q_n(A_{n1}, ..., A_{nt})$

Variablen: Großbuchstabe + optional weitere Klein-/Großbuchstaben, spezieller Markierung für nicht gebrauchte Werte "_"

Konstanten: Beginnen mit Kleinbuchstaben oder Zahlen

Kommentare: Mit % markiert

Datalog Syntax

Atomare Formeln: $q(A_1, ..., A_m).$

Regeln: $p(X_1, ..., X_m) :- q_1(A_{11}, ...A_{1s}), ..., q_n(A_{n1}, ..., A_{nt})$

Variablen: Großbuchstabe + optional weitere Klein-/Großbuchstaben, spezieller Markierung für nicht gebrauchte Werte "_"

Konstanten: Beginnen mit Kleinbuchstaben oder Zahlen

Kommentare: Mit % markiert

Beispiele

Atomare Formeln: kennt(simon,michael,2013).

Regeln: moritzBekannte(Pers) :- kennt("Moritz",Pers,_).

Variablen: Pers, Jahr, _

Konstanten: "Moritz", simon, 2013

Kommentare: % kennt(A,B,Jahr)

Wichtig: Alle Formeln mit "." beenden!

Datalog Regeln

Konjunktive Prädikate

Wenn mehrere Bedingungen gleichzeitig gelten, können diese mit ”,” verknüpft werden.

verbindungen2014(P1,P2):- kennt(P1,P2,Jahr), Jahr=2014.

Disjunktive Prädikate

Nur eine der Regeldefinitionen muss erfüllt sein, dass ein Wert enthalten ist.

simonMoritzBekannte(P2):- kennt(simon,P2,_).

simonMoritzBekannte(P2):- kennt(”Moritz”,P2,_).

Joins

Datenbasis

% kennt(PersA,PersA,Jahr).

% wohntIn(Pers,Ort).

Regel

```
bekannteImGleichenOrt(A,B,Ort) :-
        kennt(A,B,_), wohntIn(A,Ort), wohntIn(B,Ort).
```

Alle Variablen mit gleichem Namen müssen mit dem selben Wert belegt sein. Somit können die Werte mehrere Prädikate verknüpft werden.

Äquivalenz zu SQL

Datalog

verbindungen2014(P1,P2):- kennt(P1,P2,Jahr), Jahr=2014.

SQL

```
WITH verbindungen2014(P1,P2) as (
  SELECT PersA, PersB FROM kennt WHERE Jahr = 2014
)
select * from verbindungen2014;
```

Äquivalenz zu SQL

Datalog

simonMoritzBekannte(P):- kennt(simon,P,_).
simonMoritzBekannte(P):- kennt(”Moritz”,P,_).

SQL

```
WITH simonMoritzBekannte(P) as (
  SELECT PersB FROM kennt WHERE PersA = 'simon'
  UNION
  SELECT PersB FROM kennt WHERE PersB = 'Moritz'
)
select P from simonMoritzBekannte;
```

Äquivalenz zu SQL

Datalog

bekannteImGleichenOrt(A,B;Ort) :-
kennt(A,B,_), wohntIn(A,Ort), wohntIn(B,Ort).

SQL

```
WITH bekannteImGleichenOrt(A,B,Ort) as (
  SELECT k.PersA, k.PersB, w.Ort
  FROM kennt k, wohntIn w, wohntIn wb
  WHERE k.PersA = w.Pers AND k.PersB = wb.Pers AND w.Ort = wb.Ort
)
select * from bekannteImGleichenOrt;
```

Rekursion

<= 2-hop Freunde

beliebig entfernte Bekannte

Rekursion

<= 2-hop Freunde

zweiHopFreunde(A,B) :- kennt(A,B,_).

zweiHopFreunde(A,C) :- kennt(A,B,_), kennt(B,C,_).

beliebig entfernte Bekannte

Rekursion

<= 2-hop Freunde

zweiHopFreunde(A,B) :- kennt(A,B,_).

zweiHopFreunde(A,C) :- kennt(A,B,_), kennt(B,C,_).

beliebig entfernte Bekannte

bekannte(A,B) :- kennt(A,B,_).

bekannte(A,C) :- bekannte(A,B,_), kennt(B,C,_).

Rekursion

<= 2-hop Freunde

zweiHopFreunde(A,B) :- kennt(A,B,_).

zweiHopFreunde(A,C) :- kennt(A,B,_), kennt(B,C,_).

beliebig entfernte Bekannte

bekannte(A,B) :- kennt(A,B,_).

bekannte(A,C) :- bekannte(A,B,_), kennt(B,C,_).

Datalog erlaubt Rekursion, d.h. Zyklen im Abhängigkeitsgraph. Bei Rekursion werden die Regeln solange ausgewertet, bis keine neuen Fakten mehr abgeleitet werden können.

Äquivalenz zu SQL

Datalog

bekannte(A,B) :- kennt(A,B,_).

bekannte(A,C) :- bekannte(A,B,_), kennt(B,C,_).

SQL

```
WITH RECURSIVE bekannte(A,B) as (
    SELECT k.PersA, k.PersB FROM kennt k
  UNION
    SELECT b.A, k.PersB FROM kennt k, bekannte b
    WHERE k.PersA = b.B
)
select * from bekannte;
```

Nicht alle rekursiven Regeln die in Datalog möglich sind, können auch in SQL übersetzt werden.