Introduction to
Data and Knowledge Engineering
SS2010 – 7. Tutorium

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Aufgabe 7.5: Produktbewertungen

Prädikate

TECHNISCHE UNIVERSITÄT DARMSTADT

Vorhandene Prädikate:

```
review(Review_Id,Reviewer_Id,Rating,Product_Id,Review_Body,Date)
product(Product_ID,Manufacturer,AVG_Rating,SalesRanking)
reviewer(Reviewer_ID,Total_Reviews)
```

Aufgabe 7.5: Produktbewertungen

Argumente der Prädikate

TECHNISCHE UNIVERSITÄT DARMSTADT

Bedeutung der Argumente:

`Reviewer_ID:`	Eindeutige Nummer für jeden Kunden bzw. Reviewer
`Review_ID:`	Eindeutige Nummer für jedes Review
`Rating:`	Eine ganze Zahl zwischen 1 und 5
`Product_Id:`	Eindeutige Nummer für jedes Produkt
`Review_Body:`	Inhalt von dem Review als String
`Date:`	Abgabedatum von dem Review
`Manufacturer:`	Herstellername
`AVG_Rating:`	Durchscnittliches Rating für das Produkt
`SalesRanking`	Rangzahlen zwischen 1 und 10.000.
`Total_Reviews:`	Anzahl von Reviews von einem Reviewer

Aufgabe 7.5: Produktbewertungen

a)

TECHNISCHE UNIVERSITÄT DARMSTADT

a) Finden Sie die Reviews, die von den Reviewers geschrieben wurden, die mindestens 1000 Reviews geschrieben haben.

- Reviewer, der mindestens 1000 Review geschrieben hat:
  ```
  reviewer_high_reviews(A) :- reviewer(A,B), B >= 1000 .
  ```
- Reviews, die von den Reviewers geschrieben wurden, die mindestens 1000 Reviews geschrieben haben:
  ```
  review_biased_reviewerA(A) :-  review(A, B, _, _, _, _),
                                 reviewer_high_reviews(B).
  ```

Alternative Lösung :

```
review_biased_reviewerA(A) :-  review(A, B, _, _, _, _),
                               reviewer(B,C), C >= 1000.
```

Aufgabe 7.5: Produktbewertungen

b) , c)

TECHNISCHE UNIVERSITÄT DARMSTADT

b) Finden Sie die Reviews, die von den Reviewers geschrieben wurden, die ein Produkt mehr als einmal und zwar am selben Tag bewertet haben.

- Reviewer, der am selben Tag zweimal oder mehr ein Produkt bewertet hat:

```
reviewer_multi_reviews(B) :-  review(A, B, _, C, _, D),
                              review(E, B, _, C , _, D), A\=E.
```

- Reviews, die von den Reviewers geschrieben wurden, die ein Produkt mehr als einmal und zwar am selben Tag bewertet haben:

```
review_biased_reviewerB(A) :-  review(A, B ,_, _, _),
                               reviewer_multi_reviews(B).
```

Aufgabe 7.5: Produktbewertungen

c) - e)

TECHNISCHE UNIVERSITÄT DARMSTADT

c) Finden Sie die Duplikate d.h. Reviews mit denselben Inhalten.

```
duplicates(A) :-  review(A, _, _, _, C, _),
                  review(B, _, _, _, C, _), A\= B.
```

d) Finden Sie die Produkte mit niedriger Anzahl der verkauften Exemplare. Wir definieren dies als ein SalesRanking größer als 5000.

```
lowSalesRanking(A) :- product(A, _, _, B), B > 5000.
```

e) Finden Sie die Reviews mit einem hohen Rating(5) für Produkte mit lowSalesRranking.

```
highRating_biased(A) :-  review(A, _, B, C, _, _),
                         lowSalesRanking(C), B = 5.
```

Aufgabe 7.5: Produktbewertungen

f), g)

TECHNISCHE UNIVERSITÄT DARMSTADT

f) Finden Sie die Reviews, die mindestens eine der oben gennanten Eigenschaften (a,b,c,e) besitzen.

```
spam(A) :-  review_biased_reviewerA(A).
spam(A) :-  review_biased_reviewerB(A).
spam(A) :-  duplicates(A).
spam(A) :-  highRating_biased(A).
```

g) Gesucht sind Produkte von Apple und Sony, für die es Reviews gibt, die mit anderen Reviews im Imhalt genau übereinstimmen.

```
g(p) :  -  product(P, 'Apple', _, _), duplicates(R),
           review(R, _, _, P,_, _).
g(P) : -   product(P, 'Apple', _, _), duplicates(R),
           review(R, _, _, P,_, _).
```

Aufgabe 7.6: Negation

Prädikate

TECHNISCHE UNIVERSITÄT DARMSTADT

Vorhandene Prädikate und ihre Bedeutung:

`player(P,C)`	Spielsteine des Spielers P haben die Farbe C.
`color(S,C)`	Spielstein S hat die Farbe C.
`point(S,P)`	Spielstein S liegt auf dem P-Point.
`home(S)`	Spielstein S befindet sich im Heimfeld.

Aufgabe 7.6: Negation

a),b)

TECHNISCHE UNIVERSITÄT DARMSTADT

a) `player_stone(P,S)` ist wahr, genau dann wenn Spielstein S dem Spieler P gehört.

```
player_stone(P,S) :-  player(P,C), color(S,C).
```

b) `can_beaten(S)` ist wahr, genau dann wenn Spielstein S geschlagen werden kann. Das bedeutet, dass Spielstein S allein auf einem Point liegt.

`cannot_beaten(S)` ist wahr, genau dann wenn Spielstein S nicht geschlagen werden kann.

```
cannot_beaten(S) :-  point(S,F), point(T,F), player_stone(P,S)
                     player_stone(P,T), S\= T.
can_beaten(S) :-  not(cannot_beaten(S)), point(S,P).
```

Aufgabe 7.6: Negation

c)

TECHNISCHE UNIVERSITÄT DARMSTADT

c) `can_move(A,P)` ist wahr, genau dann wenn Spieler A seine eigenen Spielsteine auf das Feld(Point) P ziehen kann. Das heißt, dass Feld P nicht von mehr als einem gegnerischen Stein besetz ist.

Point P ist nur von einem gegnerischen Stein besetzt:

```
one_opponent(A,P) :-  point(S,P), can_beaten(S), color(S,C),
                      player(A,D), C\= D.
```

Point P ist von keinem gegnerischen Stein besetzt:

$\forall S : \neg point(S, P) \vee (player\_stone(A, S))$

Aufgabe 7.6: Negation

Fortsetzung c)

TECHNISCHE UNIVERSITÄT DARMSTADT

Allquantifizierung

- Variablen, die nur im Body vorkommen, sind $\exists$-quantifiziert
- hier wird Aussage über *alle* Steine, Spieler und Points gemacht,
- Umformulierung mit Negation

$\neg\,(\exists S : \mathit{point}(S, P) \wedge (\neg \mathit{player\_stone}(A, S)))$

```
aux_pred(A,P) :- point(S,P), not(player_stone(A,S)).
no_opponent(A,P) :- not(aux_pred).
```

Also:

```
can_move(A,P) :- one_opponent(A,P)
can_move(A,P) :- no_opponent(A,P)
```

Aufgabe 7.6: Negation

d), e)

TECHNISCHE UNIVERSITÄT DARMSTADT

d) `start_move_out(P)` ist wahr, wenn Spieler P seine eigenen Spielsteine hinauswürfeln kann, das heißt wenn sich alle seine eigenen Spielsteine im Heimfeld befinden.

```
cannot_start_move_out(P) :- not(home(S)), player_stone(P,S).
start_move_out(P) :- not(cannot_start_move_out(P)).
```

e) `player_win(P)` ist wahr, genau dann wenn Spieler P gewonnen hat, das heißt es gibt keine Spielsteine mehr am Brett, die dem Spieler P gehören.

```
player_win(P) :-  not( player_on(P) ).
player_on(P) :-   player_stone(P, S), point(S,Z).
```