# Deduktive Datenbanken

- ▷ Fakten
- ▷ Anfragen
- ▷ Anfragen mit Variablen
- ▷ Konjunktion von Zielen
- ▷ Relationale Datenbanken
- ▷ Deduktion
- ▷ Regeln
- ▷ Beispiel: Wegplanung

# Fakten

**Relationale Datenbank:** Aufzählung von Faktenwissen über einen zu modellierenden Gegenstandsbereich (extensionale Definition).

```
+------+-----------+-------------------+------+------+
| id   | type      | str               | no   | jahr |
+------+-----------+-------------------+------+------+
| 1    | EFH       | Gaertnerstr       | 15   | 1965 |
| 2    | EFH       | Bahnhofsstr       | 27   | 1943 |
| 3    | EFH       | Bahnhofsstr       | 29   | 1955 |
| 4    | MFH       | Bahnhofsstr       | 28   | 1991 |
| 5    | Bahnhof   | Bahnhofsstr       | 30   | 1901 |
| 6    | Kaufhaus  | Bahnhofsstr       | 26   | 1997 |
| 7    | EFH       | Gaertnerstr       | 17   | 1982 |
+------+-----------+-------------------+------+------+
```

**Simulation** von relationalen Datenbanken in Prolog (Datalog)

**Fakten:** elementare Klauseln

**Klauseln:** Klausel ::= Term '.' | . . .

**Klauseln** werden immer durch einen Punkt abgeschlossen.

```
obj(1,efh,gaertnerstr,15,1965).
obj(2,efh,bahnhofsstr,27,1943).
obj(3,efh,bahnhofsstr,29,1955).
obj(4,mfh,bahnhofsstr,28,1991).
obj(5,bahnhof,bahnhofsstr,30,1901).
obj(6,kaufhaus,bahnhofsstr,26,1997).
obj(7,efh,gaertnerstr,17,1982).
```

# Fakten

**Term:** Term ::= Prädikatsname '(' Argument {',' Argument} ')'

**Prädikatsname:** entspricht dem Tabellennamen in einer relationalen Datenbank

**Argumente:** entsprechen den Spalten einer Datenbanktabelle

**Stelligkeit** des Terms ist Bestandteil der Prädikatsdefinition. Terme unterschiedlicher Stelligkeit stehen für unterschiedliche Prädikatsausdrücke.

**Argumentposition** dient der Identifikation der Tabellenspalten. Spaltennamen stehen nicht mehr zur Verfügung.

**Argumentbelegungen** Argument ::= Konstante | ...

Konstante ::= Zahl | Name | Zeichenkette

**Namen** beginnen immer mit einem Kleinbuchstaben.

**Relation (Prozedur:)** Menge von Fakten mit gleichem Prädikatsnamen und gleicher Stelligkeit

**Datenbasis:** Menge von Relationsdefinitionen (Prozeduren)

# Anfragen

**Anfrage (Ziel, Goal):**

```
?- obj(1,efh,gaertnerstr,15,1965).
```

**Abarbeiten eines Ziels:** Prüfen auf Konsistenz mit den Fakten der Datenbasis

- Ableitbarkeit, Beweisbarkeit bezüglich einer Axiomenmenge (Datenbasis)
- abgeschlossene Welt (closed world assumption), vollständige Beschreibung

**Unifikation (1):** Vergleich des Ziels mit den Fakten der Datenbasis

- identischer Prädikatsname
- gleiche Stelligkeit
- paarweise identische Argumente

**deklaratives Semantikmodell:** Folgt die Aussage P aus der Axiomenmenge?

```
?- obj(1,efh,gaertnerstr,15,1965).
yes
?- obj(1,efh,gaertnerstr,15,1966).
no
```

**negatives Resultat** bezieht sich nur auf die gegebene Axiomenmenge (abgeschlossene Welt)

# Anfragen

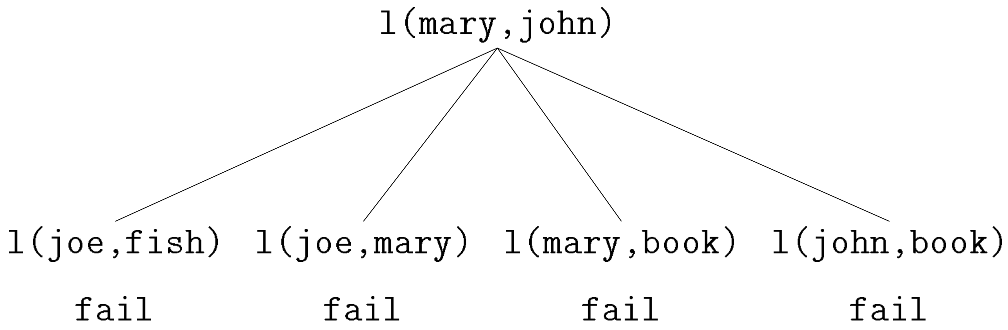
**Prozedurales Semantikmodell:** Suche nach einem unifizierbaren Axiom in der Datenbank

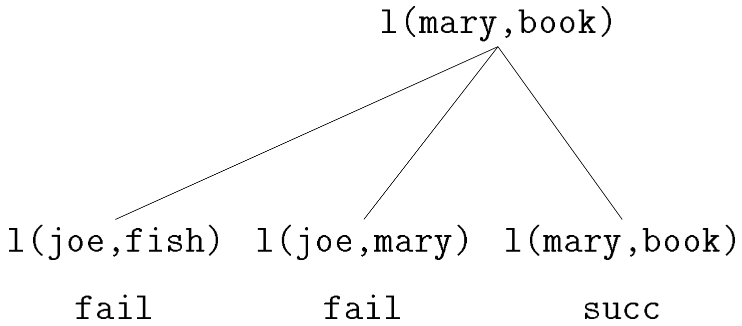
```
likes(joe,fish).
likes(joe,mary).
likes(mary,book).
likes(john,book).

?- likes(joe,money).
   likes(joe,fish).      fail
   likes(joe,mary).      fail
   likes(mary,book).     fail
   likes(john,book).     fail
no

?- likes(mary,book).
   likes(joe,fish).      fail
   likes(joe,mary).      fail
   likes(mary,book).     succ
yes
```

# Anfragen

**Darstellung des Suchraumes** als Baum

- Mutterknoten: Anfrage
- Tochterknoten: Fakten aus der Datenbasis




im Rahmen von Datenbank-Prolog sind die deklarative und die prozedurale Semantik identisch

# Anfragen mit Variablen

**universelle Programmiersprachen:** spezifizieren Berechnungsvorschriften

- ☐ gewünschtes Produkt: Berechnungsergebnisse (Datenobjekte)
- ☐ Datenobjekte werden an Variable gebunden
- ☐ Variablenbindung ist Berechnungsergebnis

**Variablenbindung**

- ☐ nichtinstanziierte Variable: Identität des Objekts (Name) noch unbekannt
- ☐ instanziierte Variable: Bindung an ein Objekt hergestellt

**Syntax:** Variablenbezeichner beginnen immer mit einem Großbuchstaben

**Gültigkeitsbereich einer Variablen:** Klausel
keine Referenz auf einen Speicherplatz!

**Prozeduraufruf**

- ☐ vollständig spezifiziertes Ziel

```
?- likes(mary,john).
```

- ☐ partiell unterspezifiziertes Ziel

```
?- likes(mary,X).
```

- ☐ vollständig unterspezifiziertes Ziel

```
?- likes(X,Y).
```

# Anfragen mit Variablen

**Berechnung (Prozedurabarbeitung):** Instanziierung von Variablen.
Ermitteln einer oder mehrerer Variablenbelegungen, die die Aussage des Ziels wahr machen.

**Unifikation (2):** Ermitteln einer solchen Variablensubstitution, die die Gleichheit von Anfrage und Datenbasiseintrag herstellt.

- ☐ identischer Prädikatsname
- ☐ gleiche Stelligkeit
- ☐ Unifikation der Argumente

| | | |
|---|---|---|
| Atom / Atom | Identitätstest | `mary = john` |
| Variable / Atom | Instanziierung | `X = mary` |
| Atom / Variable | Instanziierung | `john = Y` |
| Variable / Variable | Sharing | `X = Y` |

**Beispiele**

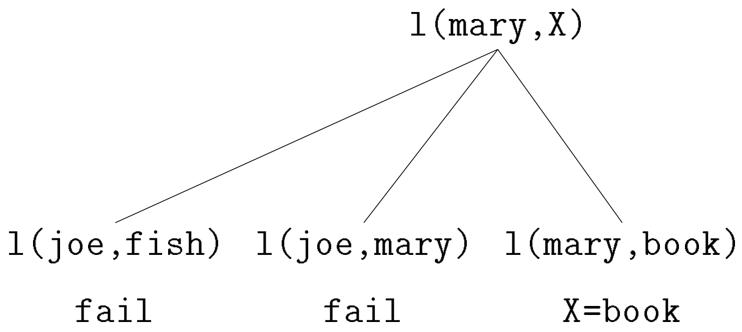
| Anfrage | Fakt | Variablensubstitution |
|---|---|---|
| `likes(mary,X)` | `likes(mary,book)` | `X=book` |
| `likes(X,book)` | `likes(mary,book)` | `X=mary` |
| | `likes(john,book)` | `X=john` |

# Anfragen mit Variablen

**Deklaratives Semantikmodell:** Gibt es eine Belegung für die Variable X, so daß `likes(mary,X)` wahr wird?

**Prozedurales Semantikmodell:** Suche nach einer konsistenten Variablenbindung

```
?- likes(mary,X).
   likes(joe,fish).      fail
   likes(joe,mary).      fail
   likes(mary,book).     succ(X=book)
X=book
```



# Anfragen mit Variablen

**Ermittlung alternativer Variablenbelegungen:** Eingabe von ";" ("oder") im Anschluß an eine Resultatsausgabe

```
?- likes(mary,X).
X=book ;
no

?- likes(X,book).
X=mary ;
X=john ;
no
```

**Auffinden der alternativen Variablenbelegungen** durch Backtracking (Rückverfolgung)

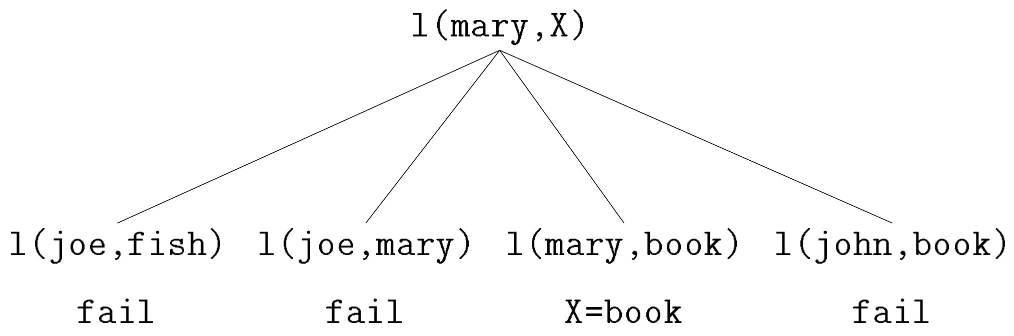
- ☐ bei erfolgreicher Unifikation: Zeiger in die Datenbasis setzen
- ☐ bei Aufforderung:
  - ◦ Zurückgehen bis zum Zeiger
  - ◦ ggf. Variablenbindungen rückgängig machen
  - ◦ Suche nach alternativen Fakten in der Datenbasis

$\longrightarrow$ Prozedurales Semantikmodell

# Anfragen mit Variablen

**Prozeduraufruf mit einem partiell spezifizierten Ziel (1):**

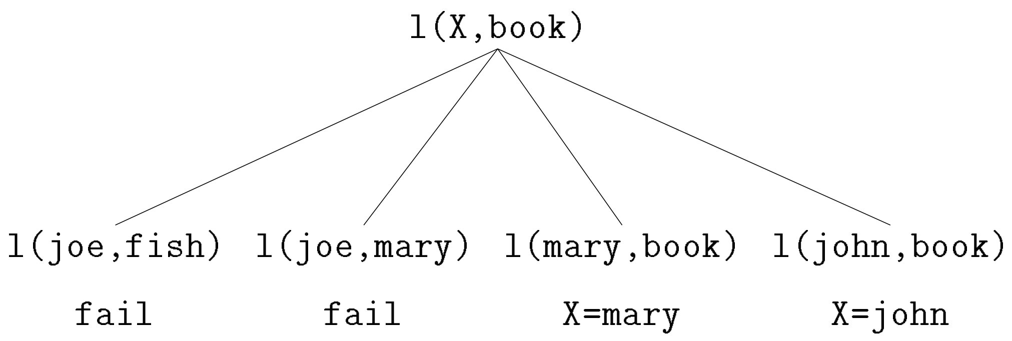
```
?- likes(mary,X).
   likes(joe,fish).      fail
   likes(joe,mary).      fail
   likes(mary,book).     succ(X=book)
  X=book ;
   likes(john,book).     fail
  no
```



# Anfragen mit Variablen

**Prozeduraufruf mit einem partiell spezifizierten Ziel (2):**

```
?- likes(X,book).
   likes(joe,fish).        fail
   likes(joe,mary).        fail
   likes(mary,book).       succ(X=mary)
X=mary ;
   likes(john,book).       succ(X=john)
X=john ;
no
```



# Anfragen mit Variablen

**Prozeduraufruf mit einem vollständig unterspezifizierten Ziel:**

```
?- likes(X,Y).
X=joe,Y=fish ;
X=joe,Y=mary ;
X=mary,Y=book ;
X=john,Y=book ;
no
```

```
                          l(X,Y)

l(joe,fish)  l(joe,mary)  l(mary,book)  l(john,book)
   X=joe        X=joe        X=mary        X=john
   Y=fish       Y=mary       Y=book        Y=book
```

Verschieden stark spezifizierte Ziele können unterschiedliches Prozedurverhalten ausgelösen:

| | |
|---|---|
| vollständig spezifiziert | Enthaltensein in einer Relation |
| partiell (unter-)spezifiziert | Teilmenge einer Relation |
| vollständig unterspezifiziert | gesamte Relation |

# Konjunktion von Zielen

**Komplexe Anfragen:** Konjunktion mehrerer Prädikate

```
?- likes(X,mary),likes(X,fish).
X=joe
```

**Konjunktion von Zielen** entspricht einem Join zwischen Tabellen einer relationalen Datenbank

**Suche bei komplexen Anfragen**

```
?- likes(X,mary),likes(X,fish).
X=joe ;
no
```

# Konjunktion von Zielen

## Suche bei komplexen Anfragen

```
?- likes(X,mary),likes(X,fish).
   ?- likes(X,mary).            #1
      likes(joe,fish).          fail
      likes(joe,mary).          succ(X=joe)
      ?- likes(joe,fish).       #2
         likes(joe,fish).       succ(X=joe)
X=joe ;
         likes(joe,mary).       fail
         likes(mary,book).      fail
         likes(john,book).      fail
      BT #1
      likes(mary,book).         fail
      likes(john,book).         fail
no
```

```
                       l(X,mary),l(X,fish)

       l(joe,fish)  l(joe,mary)  l(mary,book)  l(john,book)
          fail        X=joe         fail          X=fail
                    l(joe,fish)

l(joe,fish)  l(joe,mary)  l(mary,book)  l(john,book)
   succ         fail          fail          fail
```

# Konjunktion von Zielen

## Suche bei komplexen Anfragen

```
?- likes(joe,X),likes(X,book).
   ?- likes(joe,X).                #1
      likes(joe,fish).             succ(X=fish)
      ?- likes(fish,book).
         likes(joe,fish).          fail
         likes(joe,mary).          fail
         likes(mary,book).         fail
         likes(john,book).         fail
      BT #1
      likes(joe,mary).             succ(X=mary)
      ?- likes(mary,book).         #2
         likes(joe,fish).          fail
         likes(joe,mary).          fail
         likes(mary,book).         succ(X=mary)
 X=mary ;
         likes(john,book).         fail
      BT #1
      likes(mary,book).            fail
      likes(john,book).            fail
no
```

# Relationale Datenbanken

**Unbenannte Variable** wird verwendet, wenn Variablenbindung irrelevant ist (Projektion)

```
select id,type,jahr from obj where str='bahnhofsstr'

?- obj(Id,Type,gaertnerstr,_,Jahr).
   Id=1, Type=efh, Jahr=1965 ;
   Id=7, Type=efh, Jahr=1982 ;
   no
```

**Mehrere anonyme Variable in einer Klausel:** unabhängige Instanziierung

```
select str,no from obj where type='mfh'

?- obj(_,mfh,Str,No,_).
   Str=bahnhofstr, No=28 ;
   no
```

# Relationale Datenbanken

## Mehrtabellenanfragen

| id | type | str | no | jahr |
|---|---|---|---|---|
| 1 | EFH | Gaertnerstr | 15 | 1965 |
| 2 | EFH | Bahnhofsstr | 27 | 1943 |
| 3 | EFH | Bahnhofsstr | 29 | 1955 |
| 4 | MFH | Bahnhofsstr | 28 | 1991 |
| 5 | Bahnhof | Bahnhofsstr | 30 | 1901 |
| 6 | Kaufhaus | Bahnhofsstr | 26 | 1997 |
| 7 | EFH | Gaertnerstr | 17 | 1982 |

| id | obj_id | verk | kauf | preis | datum |
|---|---|---|---|---|---|
| 1 | 1 | Mueller | Meier | 450000 | 97.01.01 |
| 2 | 3 | Schulze | Schneider | 560000 | 88.12.13 |
| 3 | 3 | Schneider | Mueller | 615000 | 96.12.01 |
| 4 | 5 | Bund | Piepenbrink | 3500000 | 91.06.01 |

```
select bew.kauf,obj.str,obj.no from bew,obj
       where bew.obj_id=obj.id
```

| kauf | str | no |
|---|---|---|
| Meier | Gaertnerstr | 15 |
| Schneider | Bahnhofsstr | 29 |
| Mueller | Bahnhofsstr | 29 |
| Piepenbrink | Bahnhofsstr | 30 |

# Relationale Datenbanken

## Mehrtabellenanfragen

```
obj(1,efh,gaertnerstr,15,1965).
obj(2,efh,bahnhofsstr,27,1943).
obj(3,efh,bahnhofsstr,29,1955).
obj(4,mfh,bahnhofsstr,28,1991).
obj(5,bahnhof,bahnhofsstr,30,1901).
obj(6,kaufhaus,bahnhofsstr,26,1997).
obj(7,efh,gaertnerstr,17,1982).

bew(1,1,mueller,meier,450000,"97.01.01").
bew(2,3,schulze,schneider,560000,"88.12.13").
bew(3,3,schneider,mueller,615000,"96.12.01").
bew(4,5,bund,piepenbrink,3500000,"91.06.01").

?- obj(X,_,Str,No,_),bew(_,X,_,Kaeufer,_,_).
   X=1, Str=gaertnerstr, No=15, Kaeufer=meier ;
   X=3, Str=bahnhofsstr, No=29, Kaeufer=schneider ;
   X=3, Str=bahnhofsstr, No=29, Kaeufer=mueller ;
   X=5, Str=bahnhofsstr, No=30, Kaeufer=piepenbrink ;
   no
```

# Deduktion

**Ableitung** von Folgerungen aus einer Axiomenmenge

**einfachster Fall:**

**Axiome:** Atomare Formeln und Implikationen

**Schlußregel:** modus ponens

$$a \land (a \rightarrow b) \models b$$

**Prädikatenlogik**

$$\frac{\begin{array}{c}\text{Hans ist ein Mann.}\\ \text{Alle Männer sind Verbrecher.}\end{array}}{\text{Hans ist ein Verbrecher.}}$$

$$\frac{\begin{array}{c}\text{man(hans)}\\ \forall x.\text{man}(x) \rightarrow \text{criminal}(x)\end{array}}{\text{criminal(hans)}}$$

**Ziel:** Redundanzbeseitigung

- ☐ Fakten müssen nicht alle explizit spezifiziert werden
- ☐ lassen sich bei Bedarf mit Hilfe der Regeln erzeugen

# Regeln

**Horn-Klauseln:** spezieller prädikatenlogischer Formeltyp

$$\neg P_1 \vee \ldots \vee \neg P_n \vee P_0$$

$$P_1 \wedge \ldots \wedge P_n \rightarrow P_0$$

**Prolog-Notation:** reverse Implikation

```
P0 :- P1,...,Pn.
```

**Kopf:** Conclusio

**Körper:** Konjunktion von Prämissen

**implikative Lesart:** $P_0$ if $P_1$ and ... and $P_n$

```
likes(tom,X):-likes(X,book).
```

| |
|---|
| Fakten sind spezielle Horn-Klauseln (mit leerem Körper).<br>$\mathbf{T} \rightarrow P_0$ |

**Prolog-Notation**

```
likes(joe,mary):-true.
likes(joe,mary).
```

# Regeln

**Regeln** dienen der *intensionalen* Spezifikation einer Relation

- ☐ keine Aufzählung
- ☐ Angabe der Berechnungsvorschrift

$$\mathcal{R}_i = \{(x_1, \ldots, x_n) | \textit{Bedingungen}\}$$

**Anwendung einer Regel**

- ☐ Unifikation des Regelkopfes mit der Anfrage
- ☐ falls erfolgreich:
  - Umbenennen der Variablen zur Vermeidung von Namenskonflikten
  - Ersetzen der Anfrage durch Regelkörper
  - Verwenden der durch die Unifikation hergestellten Variablenbindungen
  - Abarbeiten des Regelkörpers als neue, komplexe Anfrage

# Regeln

## erweiterte Datenbank

```
likes(joe,fish).
likes(joe,mary).
likes(mary,book).
likes(john,book).
likes(tom,X):-likes(X,book).
```

## Prozeduraufruf mit vollständig instanziiertem Ziel (I)

```
?- likes(tom,john).                      #1
   likes(joe,fish).                      fail
   likes(joe,mary).                      fail
   likes(mary,book).                     fail
   likes(john,book).                     fail
   likes(tom,X1):-likes(X1,book).        succ(X1=john)
   ?- likes(john,book).                  #2
      likes(joe,fish).                   fail
      likes(joe,mary).                   fail
      likes(mary,book).                  fail
      likes(john,book).                  succ
yes
```

# Regeln

## Prozeduraufruf mit vollständig instanziiertem Ziel (II)

```
?- likes(tom,john).

                  l(t,j)

l(j,f) l(j,m) l(m,b) l(j,b) l(t,X1):-l(X1,b)
 fail   fail   fail   fail        X1=j
                                 l(j,b)

                    l(j,f) l(j,m)  l(m,b)  l(j,b)
                     fail   fail    fail    succ
```

→ Test auf Enthaltensein eines Elementes in der Relation

# Regeln

## Prozeduraufruf mit partiell instanziiertem Ziel (I)

```
?- likes(tom,X).                               #1
   likes(joe,fish).                            fail
   likes(joe,mary).                            fail
   likes(mary,book).                           fail
   likes(john,book).                           fail
   likes(tom,X1):-likes(X1,book).              succ(X=X1)
   ?- likes(X1,book).                          #2
      likes(joe,fish).                         fail
      likes(joe,mary).                         fail
      likes(mary,book).                        succ(X=X1=mary)
X=mary ;
      likes(john,book).                        succ(X=X1=john)
X=john ;
      likes(tom,X2):-likes(X2,book).           succ(X1=tom,
                                               X2=book)
      ?- likes(book,book).                     #3
         likes(joe,fish).                      fail
         likes(joe,mary).                      fail
         likes(mary,book).                     fail
         likes(john,book).                     fail
         likes(tom,X3):-likes(X3,book).        fail
      BT #2
   BT #1
no
```

# Regeln

## Prozeduraufruf mit partiell instanziiertem Ziel (II)

```
?- likes(tom,X).

                l(t,X)

l(j,f)  l(j,m)  l(m,b)  l(j,b)  l(t,X1):-l(X1,b)
 fail    fail    fail    fail         X=X1
                                    l(X1,b)

        l(j,f)  l(j,m)  l(m,b)  l(j,b)  l(t,X2):-l(X2,b)
         fail    fail    X1=m    X1=j      X1=t,X2=b
                                             l(b,b)

                l(j,f)  l(j,m)  l(m,b)   l(j,b)   l(t,X2):-l(X2,b)
                 fail    fail    fail     fail          fail
```

# Regeln

## Prozeduraufruf mit partiell instanziiertem Ziel (III)

```
?- likes(X,book).                        #1
   likes(joe,fish).                      fail
   likes(joe,mary).                      fail
   likes(mary,book).                     succ(X=mary)
X=mary ;
   likes(john,book).                     succ(X=john)
X=john ;
   likes(tom,X1):-likes(X1,book).        succ(X=tom,X1=book)
   ?- likes(book,book).                  #1
      likes(joe,fish).                   fail
      likes(joe,mary).                   fail
      likes(mary,book).                  fail
      likes(john,book).                  fail
      likes(tom,X2):-likes(X2,book).     fail
   BT #1
no
```

→ Aufzählung aller Elemente der Relation, die der Aufrufbedingung genügen

# Regeln

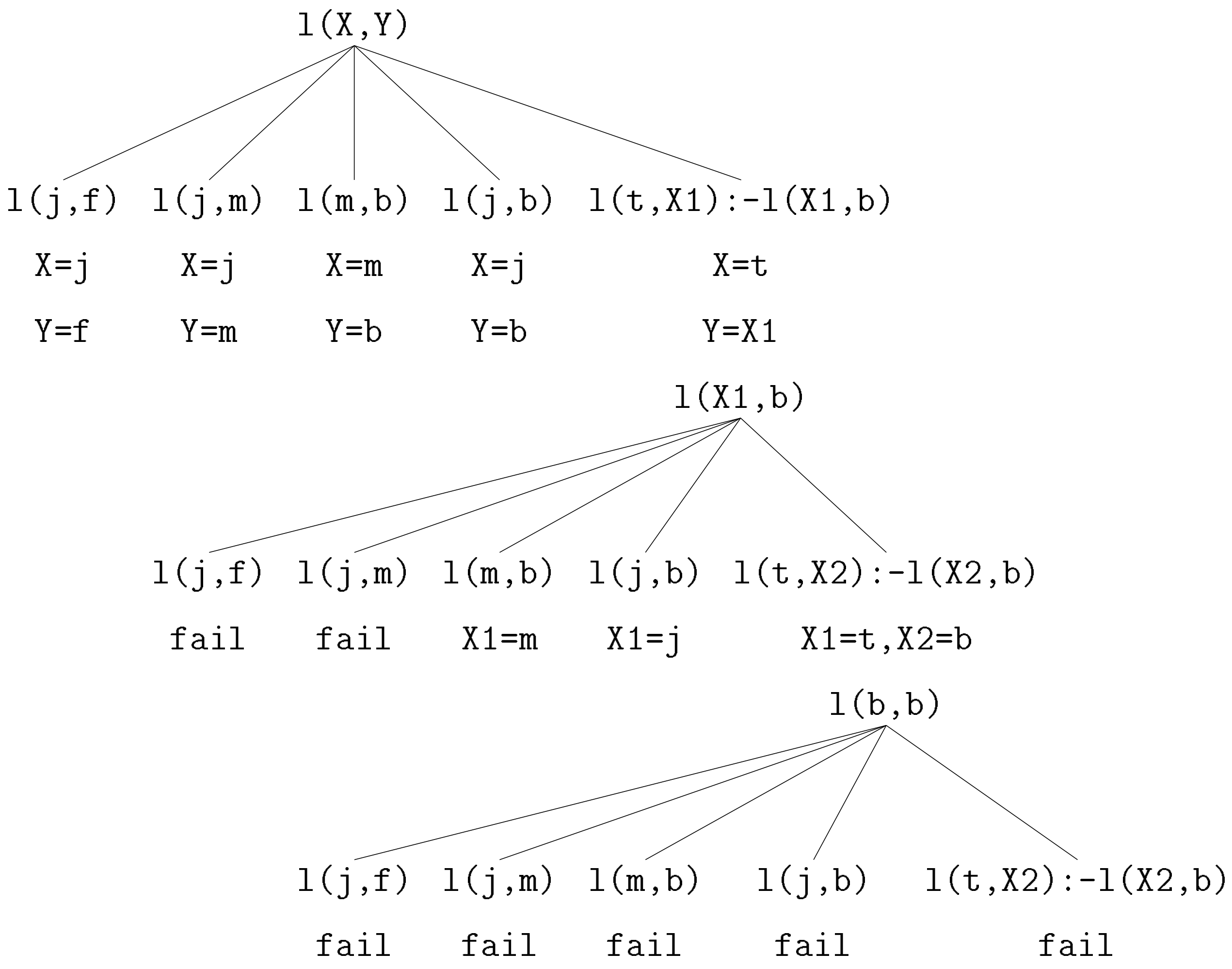
## Prozeduraufruf mit vollständig uninstanziiertem Ziel (I)

```
?- likes(X,Y).                           #1
   likes(joe,fish).                      succ(X=joe,Y=fish)
X=joe,Y=fish ;
   likes(joe,mary).                      succ(X=joe,Y=mary)
X=joe,Y=mary ;
   likes(mary,book).                     succ(X=mary,Y=book)
X=mary,Y=book ;
   likes(john,book).                     succ(X=john,Y=book)
X=john,Y=book ;
   likes(tom,X1):-likes(X1,book).        succ(X=tom,Y=X1)
   ?- likes(X1,book).                    #2
      likes(joe,fish).                   fail
      likes(joe,mary).                   fail
      likes(mary,book).                  succ(X=tom,
                                         Y=X1=mary)
X=tom,Y=mary ;
      likes(john,book).                  succ(X=tom,
                                         Y=X1=john)
X=tom,Y=john ;
      likes(tom,X2):-likes(X2,book).     succ(X=tom,Y=X1=tom,
                                         X2=book)
      ?- likes(book,book).               #3
      . . .
      BT #2
   BT #1
no
```

# Regeln

## Prozeduraufruf mit vollständig uninstanziiertem Ziel (I)

```
?- likes(X,Y).
```




→ vollständige Aufzählung der Relation

# Beispiel: Wegplanung

**Basis: Erreichbarkeit**

```
con(hamburg,berlin).
con(hamburg,hannover).
con(hamburg,bremen).
con(bremen,duesseldorf).
con(berlin,hannover)
con(hannover,duesseldorf).
```

**Symmetrie:** Eine extensional definierte, zweistellige Relation ist standardmäßig unsymmetrisch.

```
?- con(hamburg,bremen).
   yes
?- con(bremen,hamburg).
   no
```

# Beispiel: Wegplanung

**Herstellung der Symmetrieeigenschaft** erfordert zusätzlichen Aufwand.

**extensionale Spezifikation:** Angabe der Inversen

```
con(hamburg,berlin).
con(hamburg,hannover).
con(hamburg,bremen).
con(bremen,duesseldorf).
con(berlin,hannover)
con(hannover,duesseldorf).
con(berlin,hamburg).
con(hannover,hamburg).
con(bremen,hamburg).
con(duesseldorf,bremen).
con(hannover,berlin)
con(duesseldorf,hannover).

?- con(hamburg,hannover).
   yes
?- con(hannover,hamburg).
   yes
```

# Beispiel: Wegplanung

**intensionale Spezifikation (I):** Symmetrie über Regeln

```
con1(hamburg,berlin).
con1(hamburg,hannover).
...
con(X,Y):-con1(X,Y).
con(X,Y):-con1(Y,X).

?- con(hamburg,hannover).
   yes
?- con(hannover,hamburg).
   yes
```

**intensionale Spezifikation (II):** Symmetrie über rekursive Regeln

```
con(hamburg,berlin).
con(hamburg,hannover).
...
con(X,Y):-con(Y,X).

?- con(hamburg,hannover).
   yes
?- con(hannover,hamburg).
   yes
```

# Beispiel: Wegplanung

**unterspezifizierter Aufruf** einer rekursiven Regel

```
con(hamburg,berlin).
con(X,Y):-con(Y,X).

?- con(X,Y).
   X=hamburg, Y=berlin ;
   X=berlin, Y=hamburg ;
   X=hamburg, Y=berlin ;
   ...
```

→ **Differenz** zwischen deklarativer und prozeduraler Semantik:

**deklarativ:** Relation enthält zwei Elemente
**prozedural:** Suche erzeugt beliebig viele Variablenbindungen

**Scheitern** einer rekursiven Regel

```
con(hamburg,berlin).
con(X,Y):-con(Y,X).

?- con(hamburg,muenchen).
   ...
```

→ **Differenz** zwischen deklarativer und prozeduraler Semantik:

**deklarativ:** Ziel ist in der Relation nicht enthalten (Resultat: no)
**prozedural:** Suche terminiert nicht

# Beispiel: Wegplanung

**Transitivität:** Wenn eine Verbindung von A nach B existiert und eine Verbindung von B nach C, so existiert auch eine Verbindung von A nach C.

```
con1(hamburg,berlin).
con1(hamburg,hannover).
con1(hamburg,bremen).
con1(bremen,duesseldorf).
con1(berlin,hannover)
con1(hannover,duesseldorf).
con(X,Y):-con1(X,Z),con(Z,Y).

?- con(hamburg,duesseldorf).
   yes
?- con(hannover,hamburg).
   yes
?- con(hanburg,X).
   X=berlin ;
   X=hannover ;
   X=bremen ;
   X=duesseldorf ;
   X=duesseldorf ;
   X=hannover ;
   X=duesseldorf ;
   no
```