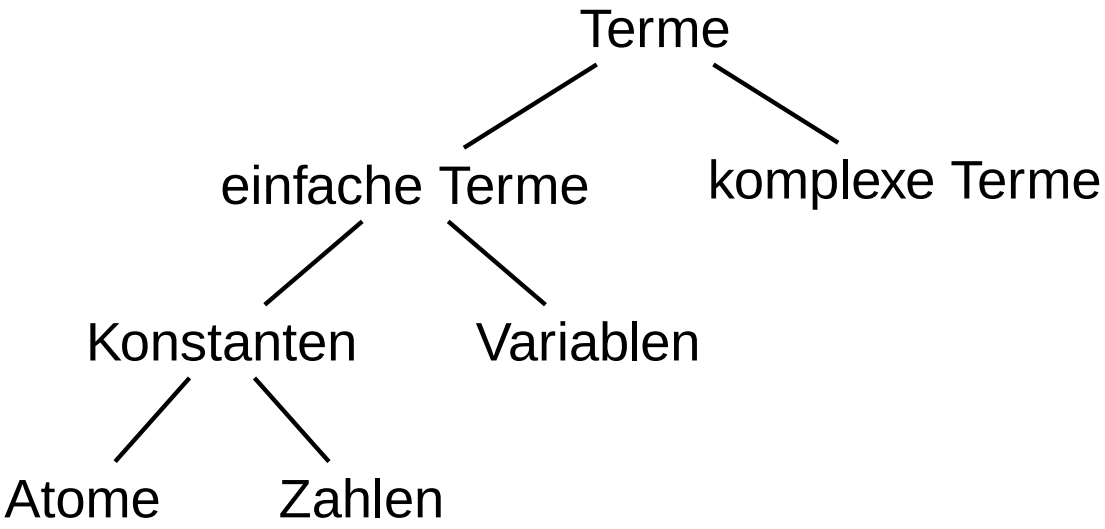
Matching und Beweisstrategie

- Matching: Wie werden in Prolog Variablen instantiiert?
- Beweisstrategie: Wie geht Prolog bei der Suche nach Antworten vor?

Erinnerung: Prologterme



Atome `butch`, `m_monroe2`, `'Vincent'`, `'The Gimp'`, `' '`, `'@&3$'`, `;`, `:-`, `===>`

Zahlen `23`, `1001`, `0`, `-345`

Variablen `X`, `Variable`, `_variable`, `X_234`

komplexe Terme `love(vincent, mia)`, `loves(vincent,wife(X,marcellus))`

Matching

Matching ist eine Operation, die guckt, ob zwei Terme gleich sind bzw. ob sie gleich gemacht werden können, indem Variablen instantiiert werden.

Das eingebaute Prädikat = testet, ob zwei Terme matchen und macht die nötigen Variableninstantiierungen.

Zum Beispiel:

```
?- =(a,a).
yes
?- =(a,b).
no
?- =(a,X).
X=a
```

Beispiele: Konstanten

Zwei Konstanten matchen genau dann, wenn sie das gleich Atom oder die gleiche Zahl sind.

```
?- =(mia,mia).
```

Vordefiniertes Prädikat: `=/2`

Beispiele: Konstanten

Zwei Konstanten matchen genau dann, wenn sie das gleich Atom oder die gleiche Zahl sind.

```
?- =(mia,mia).
yes
```

Vordefiniertes Prädikat: `=/2`

Beispiele: Konstanten

Zwei Konstanten matchen genau dann, wenn sie das gleich Atom oder die gleiche Zahl sind.

```
?- =(mia,mia).
```

Vordefiniertes Prädikat: `=/2`

```
yes
```

```
?- mia = vincent.
```

Andere Schreibweise: `=/2` als Infixoperator

Beispiele: Konstanten

Zwei Konstanten matchen genau dann, wenn sie das gleich Atom oder die gleiche Zahl sind.

```
?- =(mia,mia).
yes
?- mia = vincent.
no
```

Vordefiniertes Prädikat: `=/2`

Andere Schreibweise: `=/2` als Infixoperator

Beispiele: Konstanten

Zwei Konstanten matchen genau dann, wenn sie das gleich Atom oder die gleiche Zahl sind.

```
?- =(mia,mia).
```

Vordefiniertes Prädikat: `=/2`

```
yes
?- mia = vincent.
```

Andere Schreibweise: `=/2` als Infixoperator

```
no
?- 'Mia' = mia.
```

Beispiele: Konstanten

Zwei Konstanten matchen genau dann, wenn sie das gleich Atom oder die gleiche Zahl sind.

```
?- =(mia,mia).
yes
?- mia = vincent.
no
?- 'Mia' = mia.
no
```

Vordefiniertes Prädikat: `=/2`

Andere Schreibweise: `=/2` als Infixoperator

Beispiele: Konstanten

Zwei Konstanten matchen genau dann, wenn sie das gleich Atom oder die gleiche Zahl sind.

```
?- =(mia,mia).
yes
?- mia = vincent.
no
?- 'Mia' = mia.
no
?- 'mia' = mia.
```

Vordefiniertes Prädikat: `=/2`

Andere Schreibweise: `=/2` als Infixoperator

Beispiele: Konstanten

Zwei Konstanten matchen genau dann, wenn sie das gleich Atom oder die gleiche Zahl sind.

```
?- =(mia,mia).
yes
?- mia = vincent.
no
?- 'Mia' = mia.
no
?- 'mia' = mia.
yes
```

Vordefiniertes Prädikat: `=/2`

Andere Schreibweise: `=/2` als Infixoperator

Beispiel: Variablen 1

Falls einer der Terme eine Variable ist, dann matchen die beiden Terme und die Variable wird mit dem Wert des zweiten Terms instantiiert.

```
?- mia = X.
```

Beispiel: Variablen 1

Falls einer der Terme eine Variable ist, dann matchen die beiden Terme und die Variable wird mit dem Wert des zweiten Terms instantiiert.

```
?- mia = X.
X = mia
```

Beispiel: Variablen 1

Falls einer der Terme eine Variable ist, dann matchen die beiden Terme und die Variable wird mit dem Wert des zweiten Terms instantiiert.

```
?- mia = X.
X = mia

?- X = father(butch).
```

Beispiel: Variablen 1

Falls einer der Terme eine Variable ist, dann matchen die beiden Terme und die Variable wird mit dem Wert des zweiten Terms instantiiert.

```
?- mia = X.
X = mia

?- X = father(butch).
X = father(butch)
```

Beispiel: Variablen 1

Falls einer der Terme eine Variable ist, dann matchen die beiden Terme und die Variable wird mit dem Wert des zweiten Terms instantiiert.

```
?- mia = X.
X = mia

?- X = father(butch).
X = father(butch)

?- X = Y.
```

Beispiel: Variablen 1

Falls einer der Terme eine Variable ist, dann matchen die beiden Terme und die Variable wird mit dem Wert des zweiten Terms instantiiert.

```
?- mia = X.
X = mia

?- X = father(butch).
X = father(butch)

?- X = Y.
X = Y
```

Beispiel: Variablen 2

```
?- X = Y, X = mia.
```

Beispiel: Variablen 2

```
?- X = Y, X = mia.
X = mia,
Y = mia
```

Beispiel: Variablen 2

```
?- X = Y, X = mia.
X = mia,
Y = mia

?- X = Y, X = mia, Y= vincent.
```

Beispiel: Variablen 2

```
?- X = Y, X = mia.
X = mia,
Y = mia

?- X = Y, X = mia, Y= vincent.
no
```

Beispiel: Komplexe Terme

Wenn die Terme komplexe Terme sind, dann matchen sie genau dann, wenn 1.) sie den gleichen Funktor haben, 2.) alle korrespondierenden Argumente matchen, und 3.) die Variableninstantiierungen kompatibel sind.

```
?- kill(shoot(gun),Y) = kill(X,stab(knife)).
```

Beispiel: Komplexe Terme

Wenn die Terme komplexe Terme sind, dann matchen sie genau dann, wenn 1.) sie den gleichen Funktor haben, 2.) alle korrespondierenden Argumente matchen, und 3.) die Variableninstantiierungen kompatibel sind.

```
?- kill(shoot(gun),Y) = kill(X,stab(knife)).
X = shoot(gun),
Y = stab(knife)
```

Beispiel: Komplexe Terme

Wenn die Terme komplexe Terme sind, dann matchen sie genau dann, wenn 1.) sie den gleichen Funktor haben, 2.) alle korrespondierenden Argumente matchen, und 3.) die Variableninstantiierungen kompatibel sind.

```
?- kill(shoot(gun),Y) = kill(X,stab(knife)).
X = shoot(gun),
Y = stab(knife)
?- kill(shoot(gun), stab(knife)) = kill(X,stab(Y)).
```

Beispiel: Komplexe Terme

Wenn die Terme komplexe Terme sind, dann matchen sie genau dann, wenn 1.) sie den gleichen Funktor haben, 2.) alle korrespondierenden Argumente matchen, und 3.) die Variableninstantiierungen kompatibel sind.

```
?- kill(shoot(gun),Y) = kill(X,stab(knife)).
X = shoot(gun),
Y = stab(knife)
?- kill(shoot(gun), stab(knife)) = kill(X,stab(Y)).
X = shoot(gun),
Y = knife
```

Beispiel: Komplexe Terme

Wenn die Terme komplexe Terme sind, dann matchen sie genau dann, wenn 1.) sie den gleichen Funktor haben, 2.) alle korrespondierenden Argumente matchen, und 3.) die Variableninstantiierungen kompatibel sind.

```
?- kill(shoot(gun),Y) = kill(X,stab(knife)).
X = shoot(gun),
Y = stab(knife)
?- kill(shoot(gun), stab(knife)) = kill(X,stab(Y)).
X = shoot(gun),
Y = knife
?- loves(X,X) = loves(marcellus,mia).
```

Beispiel: Komplexe Terme

Wenn die Terme komplexe Terme sind, dann matchen sie genau dann, wenn 1.) sie den gleichen Funktor haben, 2.) alle korrespondierenden Argumente matchen, und 3.) die Variableninstantiierungen kompatibel sind.

```
?- kill(shoot(gun),Y) = kill(X,stab(knife)).
X = shoot(gun),
Y = stab(knife)
?- kill(shoot(gun), stab(knife)) = kill(X,stab(Y)).
X = shoot(gun),
Y = knife
?- loves(X,X) = loves(marcellus,mia).
no
```

Zyklische Terme

```
?- father(X) = X.
```

Zyklische Terme

```
?- father(X) = X.
```

in älteren Prolog-Implementationen:

```
?- father(X) = X.
X = father( father( father( father( father( father( father
father( father( father( father( father( father( father
(father( father( father( father( father (father(...
Not enough memory to complete query!
```

Zyklische Terme

```
?- father(X) = X.
```

in älteren Prolog-Implementationen:

```
?- father(X) = X.
X = father( father( father( father( father( father( father
father( father( father( father( father( father
(father( father( father( father( father( father (father(...
Not enough memory to complete query!
```

in SICStus, SWI ...

```
X = father( father( father( father(...)))))  ?
yes
```

Matching in Prolog

(1) *term1*, *term2* sind Konstanten: Sie matchen genau dann, wenn sie das gleich Atom oder die gleiche Zahl sind.

(2) *term1* ist eine Variable: *term1* und *term2* matchen und *term1* wird mit *term2* instantiiert.

term2 ist eine Variable: analog

(3) *term1*, *term2* sind komplexe Terme: Sie matchen genau dann, wenn

(a) *term1* und *term2* den gleichen Funktor und die gleiche Arität haben,

(b) alle korrespondirenden Argumente matchen,

(c) die Variableninstantiierungen kompatibel sind.

(4) Ansonsten matchen *term1* und *term2* nicht.

Programmieren mit Matching

```
vertical(line(point(X,Y), point(X,Z))).
horizontal(line(point(X,Y), point(Z,Y))).
```

Programmieren mit Matching

```
vertical(line(point(X,Y), point(X,Z))).
horizontal(line(point(X,Y), point(Z,Y))).

?- vertical(line(point(1,1), point(1,5))).
```

Programmieren mit Matching

```
vertical(line(point(X,Y), point(X,Z))).
horizontal(line(point(X,Y), point(Z,Y))).

?- vertical(line(point(1,1), point(1,5))).
yes
```

Programmieren mit Matching

```
vertical(line(point(X,Y), point(X,Z))).
horizontal(line(point(X,Y), point(Z,Y))).


?- vertical(line(point(1,1), point(1,5))).
yes
?- horizontal(line(point(1,2),point(3,X))).
```

Programmieren mit Matching

```
vertical(line(point(X,Y), point(X,Z))).
horizontal(line(point(X,Y), point(Z,Y))).


?- vertical(line(point(1,1), point(1,5))).
yes
?- horizontal(line(point(1,2),point(3,X))).
X = 2 ?;
no
```

Programmieren mit Matching

```
vertical(line(point(X,Y), point(X,Z))).
horizontal(line(point(X,Y), point(Z,Y))).


?- vertical(line(point(1,1), point(1,5))).
yes
?- horizontal(line(point(1,2),point(3,X))).
X = 2 ?;
no
?- horizontal(line(point(1,2),P)).
```

Programmieren mit Matching

```
vertical(line(point(X,Y), point(X,Z))).
horizontal(line(point(X,Y), point(Z,Y))).


?- vertical(line(point(1,1), point(1,5))).
yes
?- horizontal(line(point(1,2),point(3,X))).
X = 2 ?;
no
?- horizontal(line(point(1,2),P)).
P = point(_A,2) ?;
no
```

Prologs Beweisstrategie

Modus Ponens:

B :- A (Wenn A, dann B.)

A

———————

B

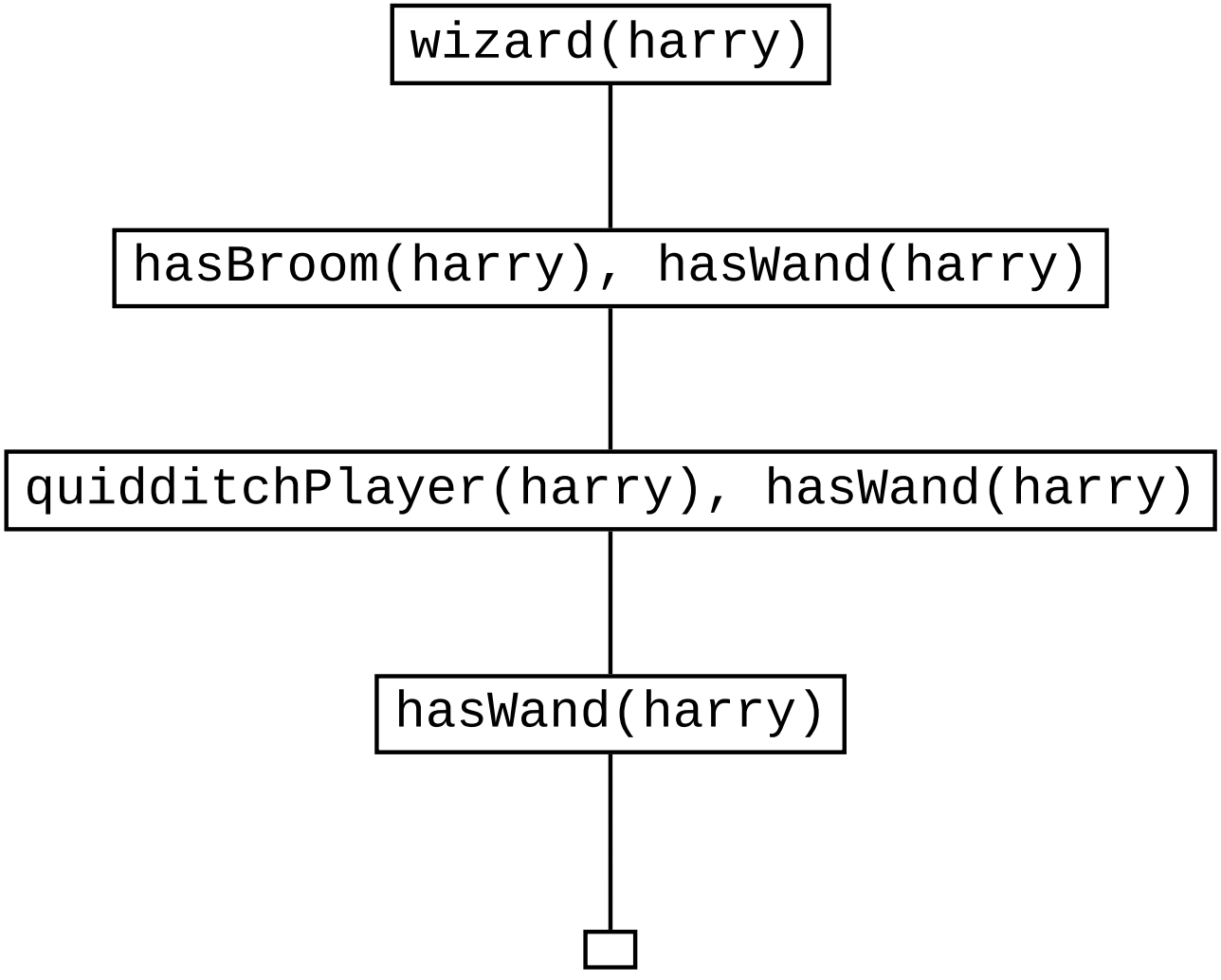
Prolog:

- ?- B. Ich will B beweisen.
- Kann ich ein Fakt oder den Kopf einer Regel aus der Wissensbasis mit B matchen?
- Ja, und diese Regel sagt mir, dass B wahr ist, falls auch A wahr ist. D.h. ich muss jetzt A beweisen.
- A matcht mit einem Fakt in meiner Wissensbasis.
- Damit ist A bewiesen und somit B auch.

Beweissuche: Beispiel 1

```
hasWand(harry).
quidditchPlayer(harry).
wizard(X) :- hasBroom(X),
             hasWand(X).
hasBroom(X) :- quidditchPlayer(X).


?- wizard(harry).
```

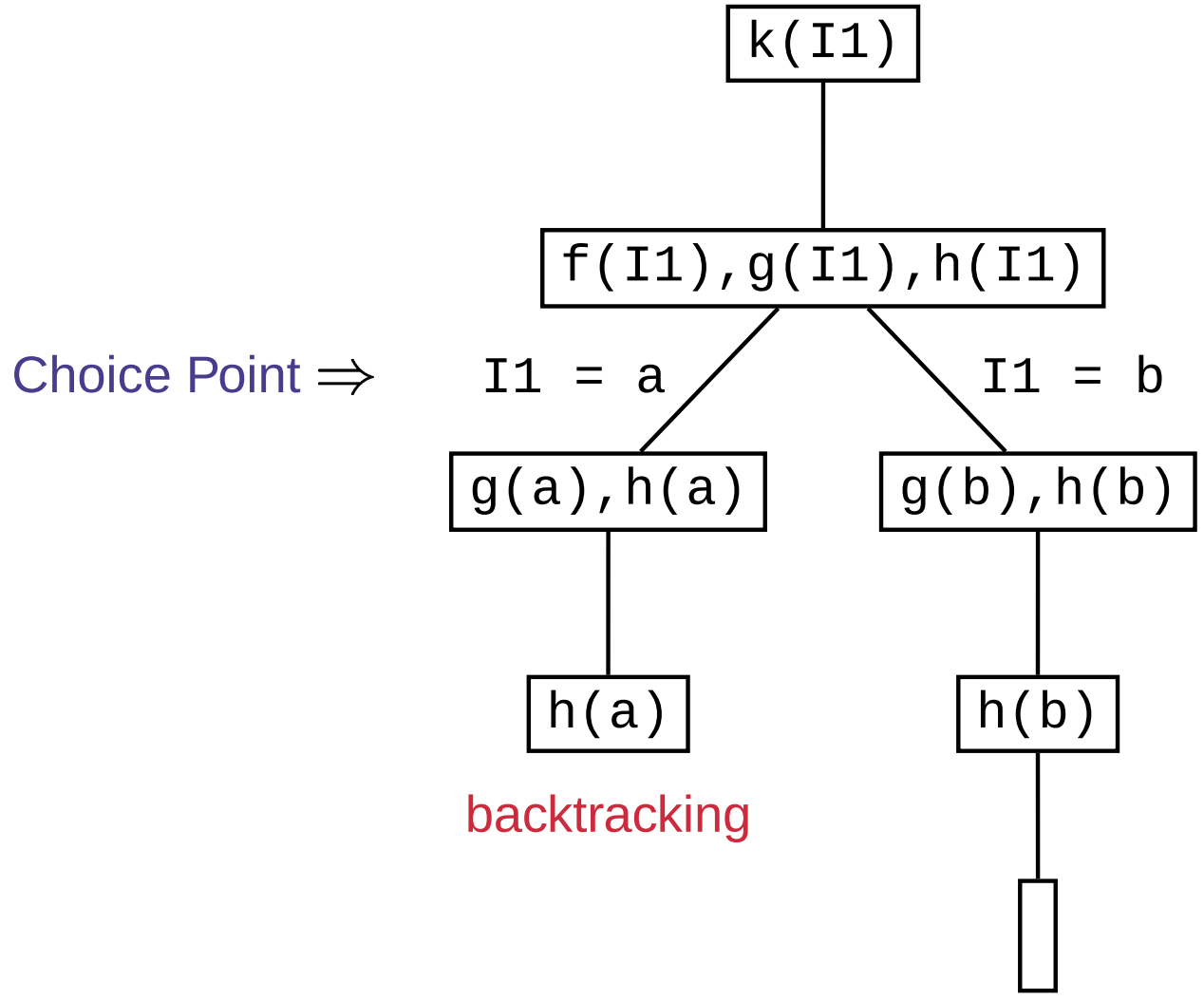
Suchbaum für Beispiel 1



Beweissuche: Beispiel 2

```
f(a).
f(b).

g(a).
g(b).

h(b).

k(X) :- f(X), g(X), h(X).


?- k(X).
```

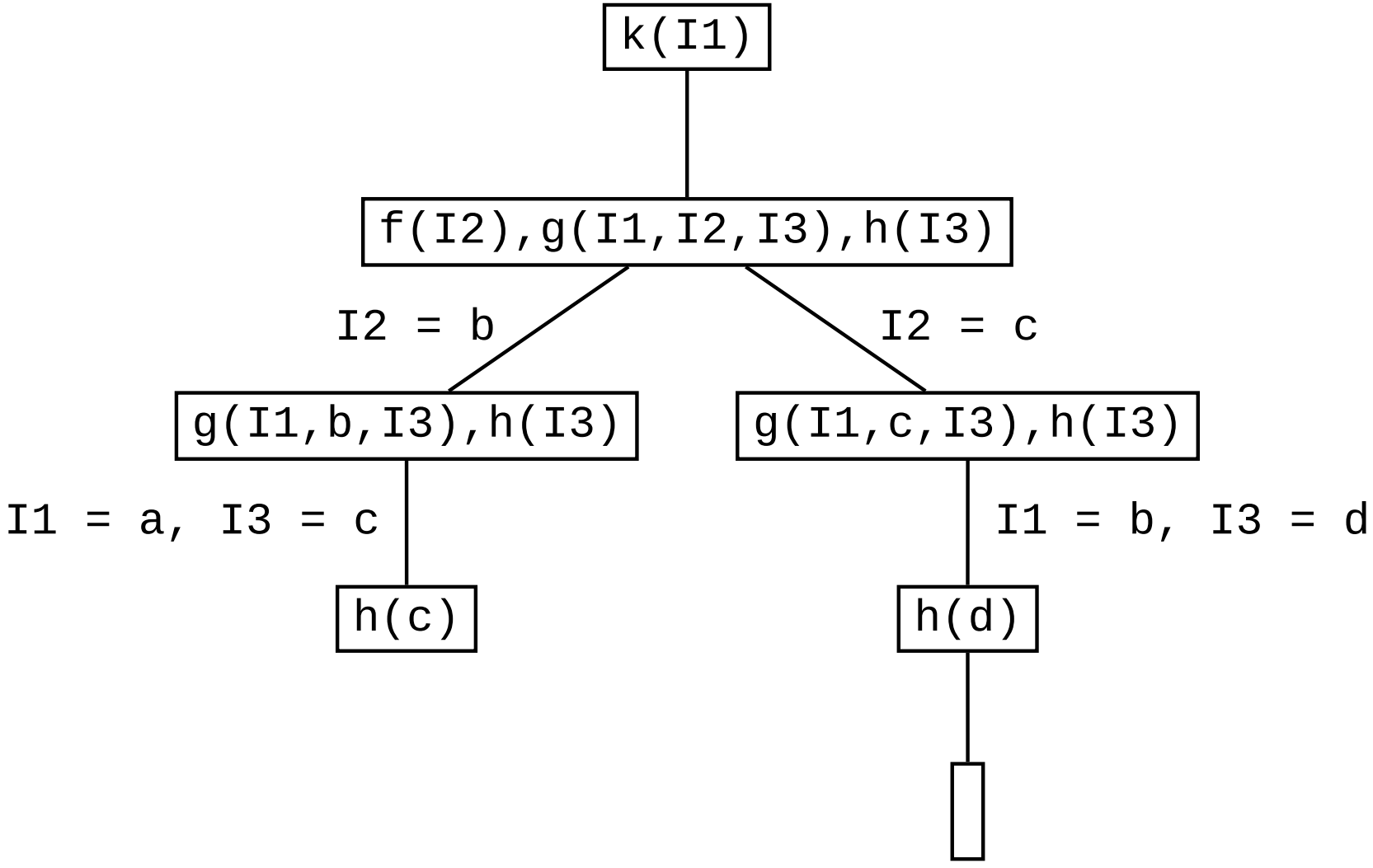
Suchbaum für Beispiel 2



Beweissuche: Beispiel 3

```
f(b).
f(c).

g(a,b,c).
g(b,c,d).

h(d).

k(X) :- f(Y), g(X,Y,Z), h(Z).


?- k(X).
```

Suchbaum für Beispiel 3



Aufgaben

```
loves(marcellus,mia).
loves(vincent,mia).
gives_footmassage(tony,mia).
jealous(X,Y) :- loves(X,Z),
                loves(Y,Z).
jealous(X,Y) :- loves(X,Z),
                gives_footmassage(Y,Z).
```

Zeichnet die Suchbäume für die folgenden Anfragen:

```
?- jealous(marcellus,tony).
```

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

Zusammenfassung

Wir haben gesehen,

- wie Matching (Unifi kation) in Prolog funktioniert, und
- wie Prolog bei der Beweissuche vorgeht.

Wichtige Begriffe: Matching, Choice Point, Backtracking

Nächste Woche: Rekursive Prädikate

Übungsaufgaben: Exercises aus Kapitel 2 von ‘LPN!’.