Ablauf

Übersicht

- Wie wird eine Prolog-Anfrage beantwortet?
  - Anfrage und Wissensbasis
  - Beispiel: Ist Sabrina eine Frau?
  - Beispiel: Wer alles sind Frauen?
  - Backtracking
- Beweisbäume

Ziel

- Verstehen, wie Prolog arbeitet

Wie wird eine Anfrage beantwortet?

Beispiel:

```
frau(sabrina).
```

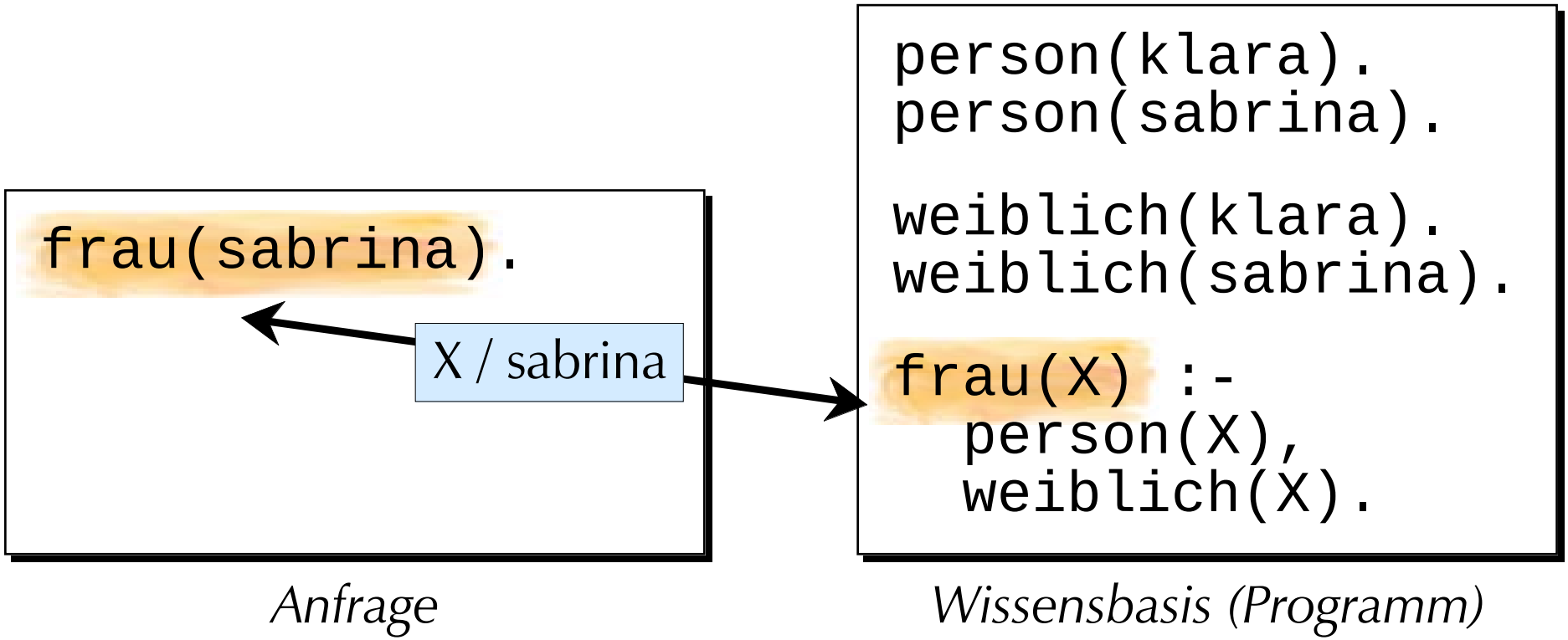
Anfrage

```
person(klara).
person(sabrina).

weiblich(klara).
weiblich(sabrina).

frau(X) :-
  person(X),
  weiblich(X).
```

Wissensbasis (Programm)

Wie wird eine Anfrage beantwortet?

Prolog versucht, den ersten Term der Anfrage mit einem Fakt oder mit dem Kopf einer Regel zu unifizieren.



Wie wird eine Anfrage beantwortet?

Prolog nimmt als erstes die oberste passende Klausel:

- Variablensubstitution (hier: X / sabrina)
- Ersetzen des ersten Terms der Anfrage durch den Klauselrumpf

```
person(sabrina),
weiblich(sabrina).
```

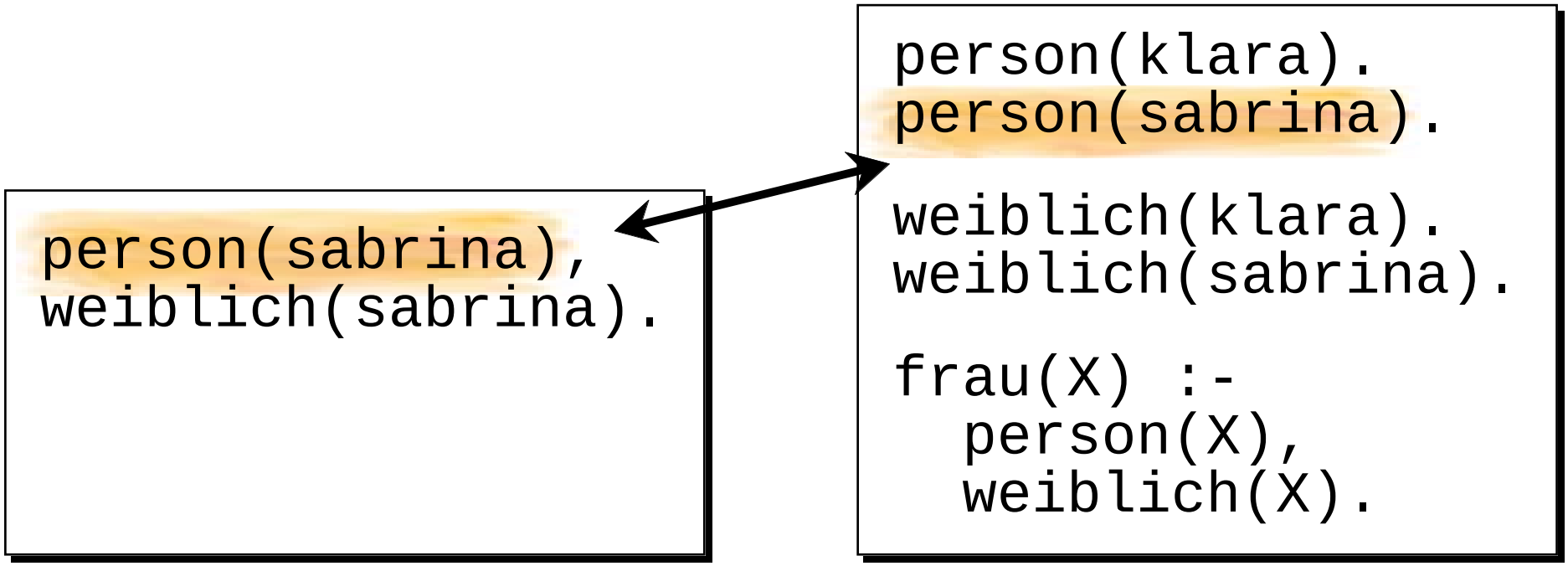
Anfrage

```
person(klara).
person(sabrina).

weiblich(klara).
weiblich(sabrina).

frau(X) :-
  person(X),
  weiblich(X).
```

Wissensbasis (Programm)

Wie wird eine Anfrage beantwortet?

Prolog versucht, den ersten Term der Anfrage mit einem Fakt oder mit dem Kopf einer Regel zu unifizieren.



Wie wird eine Anfrage beantwortet?

Prolog nimmt als erstes die oberste passende Klausel:

- Variablensubstitution (hier: keine Variablen)
- Ersetzen des ersten Terms der Anfrage durch den Klauselrumpf (bei Fakten ist der Rumpf leer)

```
weiblich(sabrina).
```

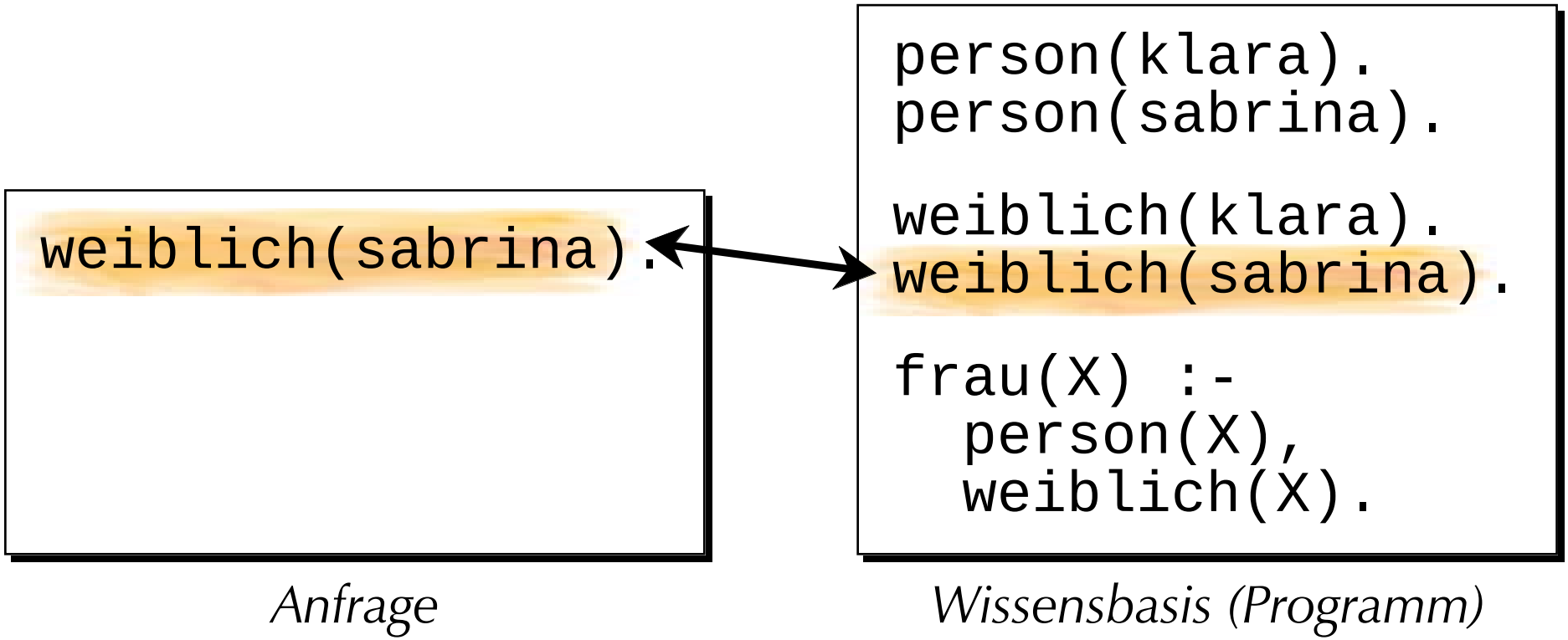
Anfrage

```
person(klara).
person(sabrina).

weiblich(klara).
weiblich(sabrina).

frau(X) :-
  person(X),
  weiblich(X).
```

Wissensbasis (Programm)

Wie wird eine Anfrage beantwortet?

Prolog versucht, den ersten Term der Anfrage mit einem Fakt oder mit dem Kopf einer Regel zu unifizieren.



Wie wird eine Anfrage beantwortet?

Prolog nimmt als erstes die oberste passende Klausel:

- Variablensubstitution (hier: keine Variablen)
- Ersetzen des ersten Terms der Anfrage durch den Klauselrumpf (bei Fakten ist der Rumpf leer)

```
.
```

Anfrage

```
person(klara).
person(sabrina).

weiblich(klara).
weiblich(sabrina).

frau(X) :-
  person(X),
  weiblich(X).
```

Wissensbasis (Programm)

Backtracking

Noch ein Beispiel:

```
frau(Wer).
```

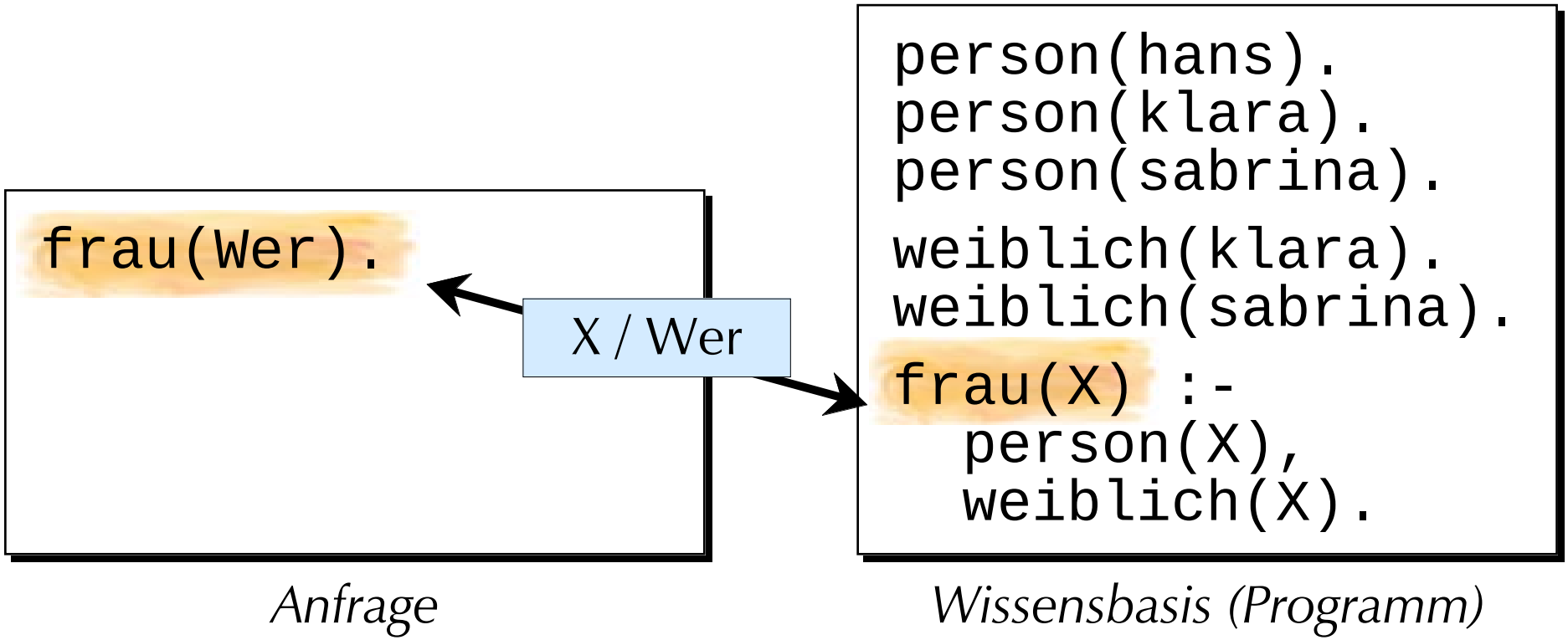
Anfrage

```
person(hans).
person(klara).
person(sabrina).

weiblich(klara).
weiblich(sabrina).

frau(X) :-
  person(X),
  weiblich(X).
```

Wissensbasis (Programm)

Backtracking

Prolog versucht, den ersten Term der Anfrage mit einem Fakt oder mit dem Kopf einer Regel zu unifizieren.



Backtracking

Prolog nimmt als erstes die oberste passende Klausel:

- Variablensubstitution (hier: X / Wer)
- Ersetzen des ersten Terms der Anfrage durch den Klauselrumpf

```
person(Wer),
weiblich(Wer).
```

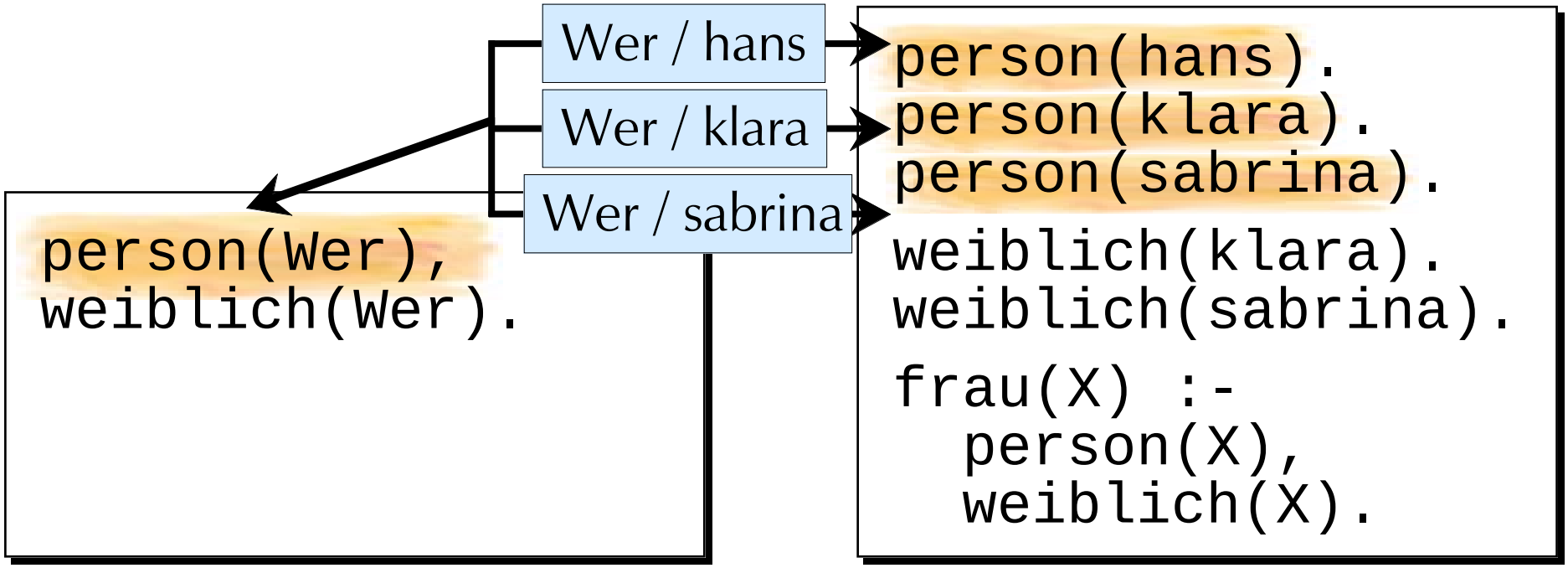
Anfrage

```
person(hans).
person(klara).
person(sabrina).

weiblich(klara).
weiblich(sabrina).

frau(X) :-
  person(X),
  weiblich(X).
```

Wissensbasis (Programm)

Backtracking

Prolog versucht, den ersten Term der Anfrage mit einem Fakt oder mit dem Kopf einer Regel zu unifizieren.



Backtracking

Prolog nimmt als erstes die oberste passende Klausel:

- Variablensubstitution (hier: *Wer* / hans)
- Ersetzen des ersten Terms der Anfrage durch den Klauselrumpf (bei Fakten ist der Rumpf leer)

```
weiblich(hans).
```

Anfrage

```
person(hans).
person(klara).
person(sabrina).

weiblich(klara).
weiblich(sabrina).

frau(X) :-
  person(X),
  weiblich(X).
```

Wissensbasis (Programm)

Backtracking

Nun kann der erste Term der Anfrage mit keinem Klauselkopf unifiziert werden.

Daher:

- Zurückgehen zum letzten Punkt, wo Prolog sich für eine Möglichkeit entscheiden musste (Entscheidungspunkt, *Decision Point*)
- Rückgängigmachen der Variablenbindungen
- Weitermachen mit der nächsten Klausel, die in Frage kommt

Dieser Vorgang heisst Backtracking.

Backtracking

Prolog nimmt also die nächste passende Klausel:

- Variablensubstitution (hier: *Wer* / klara)
- Ersetzen des ersten Terms der Anfrage durch den Klauselrumpf (bei Fakten ist der Rumpf leer)

```
weiblich(klara).
```

Anfrage

```
person(hans).
person(klara).
person(sabrina).

weiblich(klara).
weiblich(sabrina).

frau(X) :-
  person(X),
  weiblich(X).
```

Wissensbasis (Programm)

Wie wird eine Anfrage beantwortet?

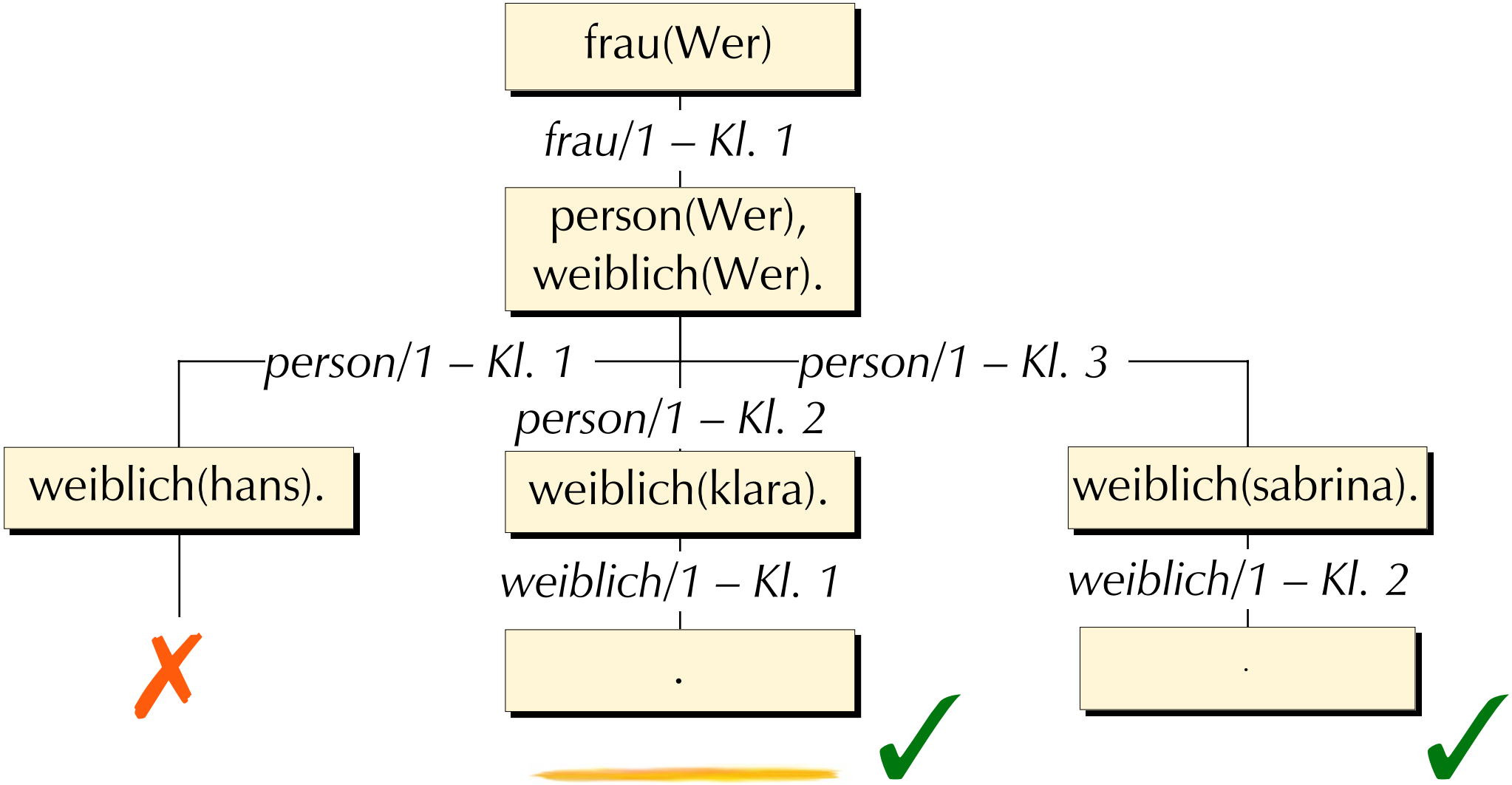
Wenn die Anfrage leer ist, ist der Beweis gelungen:

- Wenn Variablen *aus der Anfrage* gebunden wurden: Ausgabe der Bindungen
- Sonst: Ausgabe von »yes«.

Wenn die Anfrage nicht leer ist, und der erste Term ist mit keinem der Klauselköpfe unifizierbar, wird

- zuerst versucht, im Beweis zurückzugehen und statt der ersten Klausel die zweite (bzw. dritte, vierte, ...) zu nehmen, wobei Variablenbindungen rückgängig gemacht werden **(Backtracking)**
- wenn das auch nicht geht: Beweis schlägt fehl, Ausgabe von »no«.

Visualisierung als Beweisbaum



Aufgaben: Strukturen/Ablauf

Programmiertechniken der Computerlinguistik 1 · Wintersemester 1998/99

1. Terme klassifizieren

Gib bei den folgenden Zeilen jeweils an, ob es sich um einen atomaren Term, um einen komplexen Term oder um eine Variable handelt, oder ob der Term überhaupt nicht zulässig ist. Bei komplexen Termen bestimme zusätzlich Funktor sowie Stelligkeit, und klassifiziere die einzelnen Argumente nach denselben Kriterien.

a) `hans`
b) `Klara`
c) `'Kevin'`
d) `s a b r i n a`
e) `' s a b r i n a '`
f) `s_a_b_r_i_n_a_`
g) `_s_a_b_r_i_n_a`
h) `S_a_b_r_i_n_a_`
i) `_S_a_b_r_i_n_a_`
j) `liebt(hans, klara)`
k) `liebt(hans, Klara)`
l) `bUCH`
m) `Buch`
n) `familie(hans, klara, sabrina, kevin)`
o) `beisst(schnurrli fido)`
p) `kratzt(Schnurrli, Fido)`
q) `gelogen(beisst(schnurrli, FIDO))`
r) `Gelogen(beisst(schnurrli, FIDO))`
s) `behauptung(hans, gelogen(beisst(schnurrli, 'Fido')))`
t) `muehsam(muessen(viele(terme), klassifizieren))`

2. Unifizieren

Finde heraus, welche von den folgenden Paaren von Termen unifizierbar sind. Gib bei den unifizierbaren Paaren an, welche Variablen an welchen Wert gebunden werden müssen.

Beispiel: `frisst(fido, X) = frisst(fido, gulasch)`
Antwort: ja, `X`/`gulasch`

a) `brot = brot`
b) `'Brot' = brot`
c) `'brot' = brot`
d) `Brot = brot`
e) `brot = wurst`
f) `essen(brot) = wurst`
g) `essen(brot) = X`
h) `essen(X) = essen(brot)`
i) `essen(brot, X) = essen(Y, wurst)`
j) `essen(brot, X, wasser) = essen(brot, Z, Z)`
k) `essen(brot, X, wasser) = essen(Y, Z, Z)`
l) `essen(brot, X, wasser) = essen(Z, Z, Z)`
m) `essen(X) = essen(brot, wasser)`
n) `mahlzeit(essen(brot), trinken(wasser)) = mahlzeit(X, Y)`
o) `X = X`
p) `X = Y`

3. Gesund ist, wer Früchte isst

Adam mag Äpfel. Friederike mag Frikadellen. Ottilie mag Orangen. Äpfel sind Früchte. Orangen sind Früchte. Wenn jemand etwas mag, und dieses Etwas ist eine Frucht, dann ist dieser Jemand gesund.

a) Formuliere die obenstehenden Angaben als Fakten bzw. Regeln in einem Prolog-Programm. Wenn Du mit dem Formulieren der Wenn-Dann-Regel Mühe hast, schaue Dir die Lösungen zu Aufgabe 3 aus der letzten Stunde an: In welcher Richtung gilt die Folgerung?

b) Suche jemanden, der gesund ist. Welche Anfrage musst Du stellen?

c) Wie findest Du heraus, wer ausser der ersten Person sonst noch gesund ist?

d) Überlege Dir — ohne Benutzung des Computers —, mit welchen einzelnen Schritten Prolog zu den Antworten findet.

e) Zeichne einen Beweisbaum.

f) Lasse Dir einen Trace ausgeben.