

Inhalt

1. Einführung in die Informatik
2. Algorithmen
3. Imperative Programmierung
4. Verarbeitung externer Datenquellen
- 5. Exkurs: Deklarative, Logische Programmierung**

Funktional-logische Programmierung in Prolog

Prolog – Programming in Logic

Prolog ist eine deklarative Programmiersprache, dabei wird das Problem beschrieben, nicht aber die Suche nach Lösungen.

Prolog Programm: Menge von Klauseln

Klauseln können ausdrücken:

- Fakten – Wissensbasis
- Regeln – Ableitung von neuem Wissen aus bekanntem Wissen

SWI-Prolog (UNI Amsterdam),

Web-Prolog-Interpreter: *swish.swi-prolog.org*

Grundzüge der Sprache Prolog

Fakten:

wetter(sommer,heiss).

wetter(winter,kalt).

wetter(herbst, kuehl).

wetter(fruehling,warm).

kleingeschriebene Bezeichner sind konkrete Werte (*sommer, winter, warm, kalt*)

Regeln:

freibadzeit(X):-wetter(X,heiss);wetter(X,warm).

saunazeit(X):-wetter(X,kalt); wetter(X,kuehl).

Anfragen (hier in einer interaktiven Sitzung gestellt):

?- wetter(F,warm).

F=fruehling.

?-saunazeit(fruehling).

false.

Grundzüge der Sprache Prolog

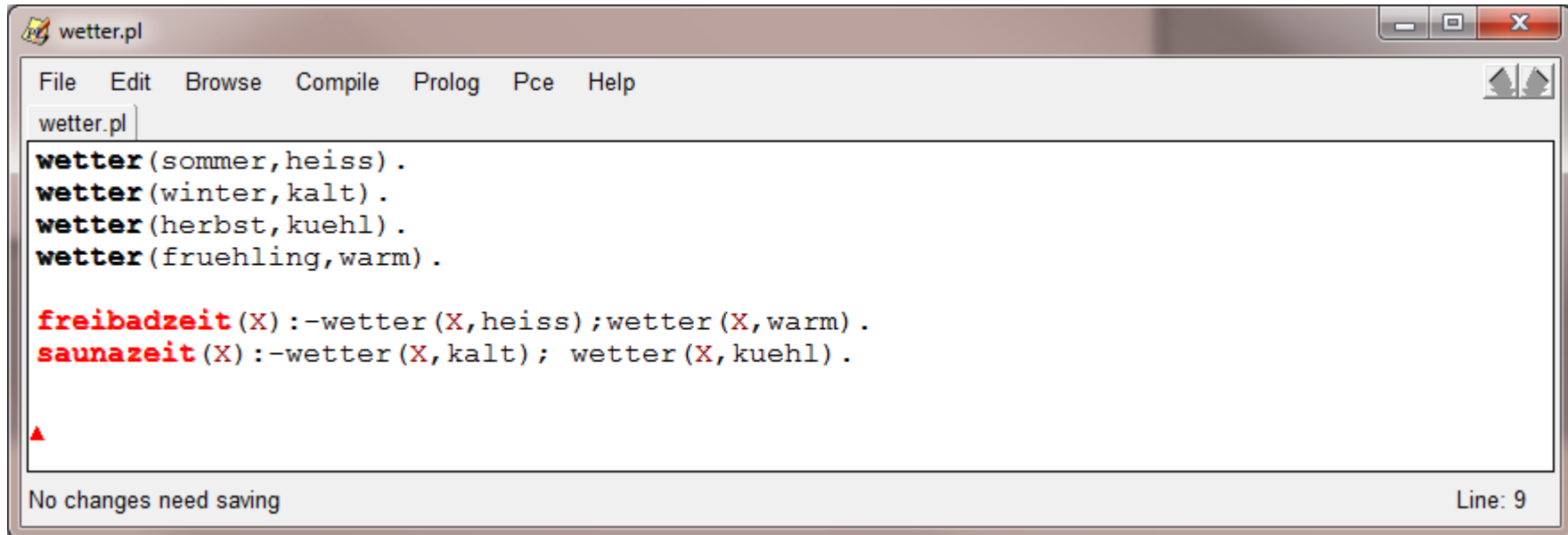
Ungebundene Variable und gebundene Variable

saunazeit(X):-wetter(X,kalt); wetter(X,kuehl).

Großgeschriebene Bezeichner, zum Beispiel X sind **ungebundene Variablen**, deren Wert zur Ausführungszeit des Programms einmal oder auch mehrmals mit unterschiedlichen Werten gebunden werden kann.

Durch Konstanten (Begriffe, Zahlenkonstanten) **gebundene Variable** werden mit kleinen Buchstaben bzw. als Zahlenwert geschrieben.

Grundzüge der Sprache Prolog



```
wetter.pl  
File Edit Browse Compile Prolog Pce Help  
wetter.pl  
wetter(sommer,heiss).  
wetter(winter,kalt).  
wetter(herbst,kuehl).  
wetter(fruehling,warm).  
  
freibadzeit(X):-wetter(X,heiss);wetter(X,warm).  
saunazeit(X):-wetter(X,kalt);wetter(X,kuehl).  
  
▲  
No changes need saving Line: 9
```

Anfrage: *?- wetter(winter,W).*

Antwort: *W=kalt.*

Anfrage: *?- wetter(herbst, warm).*

Antwort: *false.*

Anfrage: *?- saunazeit(herbst).*

Antwort: *true.*

Die Anfragen werden als Ziele aufgefasst. Prolog versucht nun:

- diese Ziele durch eine gültige Bindung der Variablen zu realisieren,
- deren Gültigkeit zu beweisen (Antwort ergibt dann *true*)
- oder deren Unerfüllbarkeit zu zeigen (Antwort: *false*).

Grundzüge der Sprache Prolog

Prolog ist eine **deklarative Sprache**. Die Reihenfolge der Klauseln ist nicht notwendigerweise die Abarbeitungsreihenfolge. Die Problemlösungsalgorithmen werden durch den Prolog-Interpreter bereitgestellt.

Prinzipien der Problemlösung:

- Suchen passender Klauseln (Unifikation).
- Bindung ungebundener Variablen mit den Werten in den Fakten und bereits gebundenen Zwischenergebnissen innerhalb der Regeln.
- Backtracking: Wenn eine spezielle Bindung von Variablen nicht zum Ziel führt, werden automatisch andere Variablenbindungen versucht.

Grundzüge der Sprache Prolog

Prinzipien der Problemlösung (Fortsetzung):

Teilziele in Regeln werden

UND-verknüpft, wenn mit Komma aufgezählt:

alle_gleich(A,B,C):-A==B, B==C.

ODER-verknüpft, wenn mit Semikolon aufgezählt

ungleich(A, B):- A >B; A<B.

UND bindet stärker als ODER:

mehrheit(A,B,C,M):- A==B, M is A; B==C, M is B; A==C, M is A.

Vergleichs-Operatoren:

Gleichheit $A == B$

Ungleichheit $A \neq B$

Zuweisung mit Bindung einer neuen Variablen: *R is S.*

Grundzüge der Sprache Prolog

Ein Beispiel mit einfachen Rechenoperationen:

Anfrage: *?- quadriere(5,Q).*

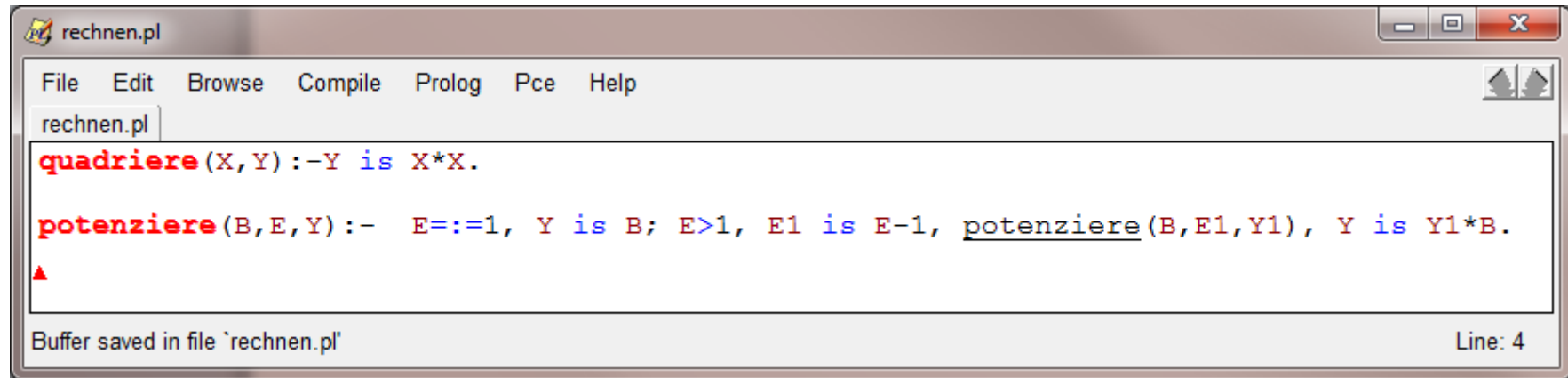
Antwort: *Q=25*

Das dazugehörige Programm muss eine Regel wie folgt enthalten:

*quadrriere(X,Y): - Y is X*X.*

Die Anfrage *quadrriere(5,Q)* wird auf die Regel *quadrriere(X,Y)* abgebildet, dabei *X* mit 5 ‚gebunden‘ und die Regel abgearbeitet. *Y* wird berechnet und *Q* wird mit dem Wert von *Y* ‚gebunden‘.

Grundzüge der Sprache Prolog



```
rechnen.pl
File Edit Browse Compile Prolog Pce Help
rechnen.pl
quadriere(X,Y):-Y is X*X.
potenziere(B,E,Y):- E==1, Y is B; E>1, E1 is E-1, potenziere(B,E1,Y1), Y is Y1*B.
▲
Buffer saved in file `rechnen.pl`
Line: 4
```

Das Programm oben wird mit *consult('rechnen.pl')*. geladen.

Die für die Anfrage (z.B. *quadriere(2,Q).*) wird zur Laufzeit automatisch die passende Regel angewendet. Das Programm wird interpretiert (Zum Vergleich: ein C-Programm wurde übersetzt).

Weitere Beispiele:

?- *potenziere(5,3,Y).*

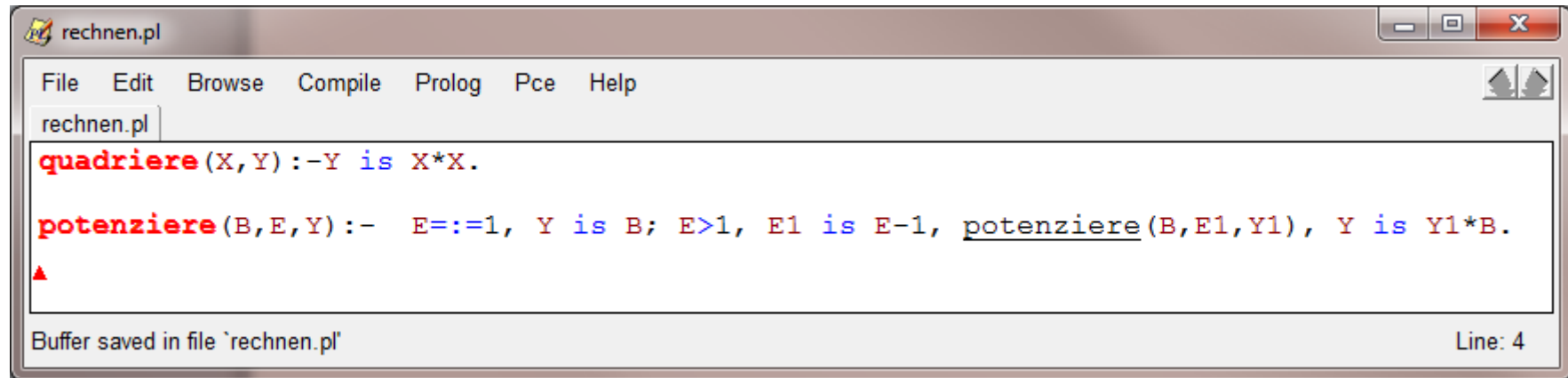
Y = 125

?- *qwurzel(25,W).*

ERROR: toplevel: Undefined procedure: qwurzel/2

Nicht erfüllbar, da keine passende Regel gefunden wird.

Grundzüge der Sprache Prolog



```
rechnen.pl
File Edit Browse Compile Prolog Pce Help
rechnen.pl
quadriere(X,Y):-Y is X*X.
potenziere(B,E,Y):- E==1, Y is B; E>1, E1 is E-1, potenziere(B,E1,Y1), Y is Y1*B.
▲
Buffer saved in file `rechnen.pl`
Line: 4
```

Weitere Beispiele:

*?- quadriere(5,25).
true.*

Man kann prüfen, ob Klauseln erfüllbar sind, d.h. bewiesen werden können.

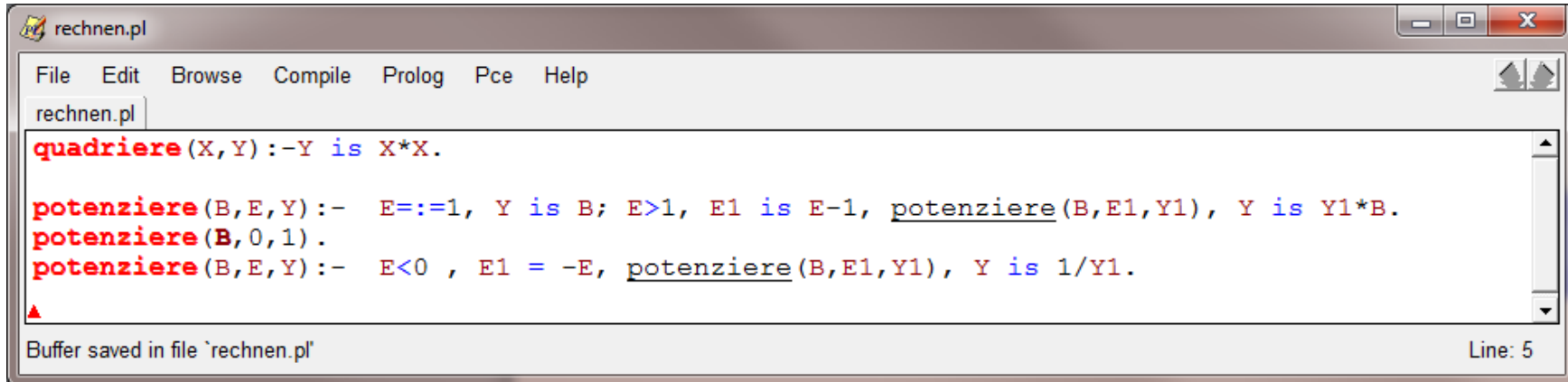
*?-quadriere(1,2).
false.*

Man kann auch zeigen, dass Beziehungen ungültig sind.

*?- potenziere(5,3,Y).
Y = 125 .
?- potenziere(5,0,Y).
false.*

potenziere ist für Exponenten gleich und größer als 1 definiert.
Für einen Exponenten 0 gibt es (noch) keinen Berechnungsweg im Programm oben.

Grundzüge der Sprache Prolog



```
File Edit Browse Compile Prolog Pce Help
rechnen.pl
quadrriere(X,Y):-Y is X*X.

potenziere(B,E,Y):- E:=1, Y is B; E>1, E1 is E-1, potenziere(B,E1,Y1), Y is Y1*B.
potenziere(B,0,1) .
potenziere(B,E,Y):- E<0 , E1 = -E, potenziere(B,E1,Y1), Y is 1/Y1.
```

Buffer saved in file 'rechnen.pl' Line: 5

Nach Erweiterung der Regelbasis um weitere Klauseln kann nun auch ein Exponent 0, oder ein negativer Exponent gehandhabt werden. Es verbleibt nur die Einschränkung, dass der Exponent ganzzahlig sein muss.

?- potenziere(5,3,Y).

Y = 125 .

?- potenziere(5,0,Y).

false.

?- potenziere(5,-2,P).

P = 0.04

Grundzüge der Sprache Prolog

Listenverarbeitung (als Ersatz für Felder):

Eine Liste wird in Eckigen Klammern notiert

[dresden, berlin,hamburg]

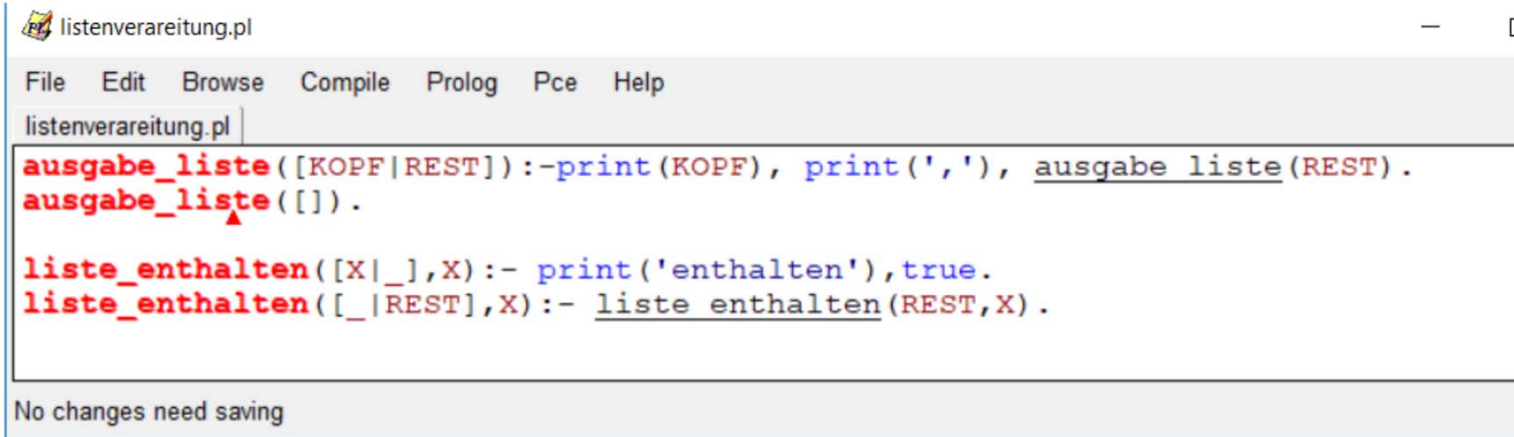
Leere Liste: *[]*

Listen können in Kopf und Rest (engl. Head und Tail) aufgeteilt werden: *[HEAD| TAIL]*

Listen werden rekursiv verarbeitet (es gibt keine Zyklen in Prolog), durch schrittweises ‚Abtrennen‘ des Kopfes und dessen Verarbeitung

Grundzüge der Sprache Prolog

Listenverarbeitung (Fortsetzung)



```
listenverarbeitung.pl
File Edit Browse Compile Prolog Pce Help
listenverarbeitung.pl
ausgabe_liste([KOPF|REST]):-print(KOPF), print(','), ausgabe_liste(REST).
ausgabe_liste([]).

liste_enthalten([X|_],X):- print('enthalten'),true.
liste_enthalten([_|REST],X):- liste_enthalten(REST,X).

No changes need saving
```

Anfragebeispiele:

?- ausgabe_liste([otto, emil, berta]).

otto , emil , berta

true.

?- liste_enthalten([dresden,berlin,hamburg], hamburg).

enthalten

true

?- liste_enthalten([dresden,berlin,hamburg], cottbus).

false.

Funktional-logische Programmierung in Prolog

Beispiel: Fakultät

fak(0,1).

fak(N,F):-

N>0,

N1 is N-1,

fak(N1,F1),

*F is N*F1.*

Anfrage: *fak(5,F).*

F=120.

Funktional-logische Programmierung in Prolog

Beispiel: Fibonacci-Zahlen

Programm:

fibonacci(0,0).

fibonacci(1,1).

fibonacci(N,F):-

A is N-1,

B is N-2,

fibonacci(A,FA),

fibonacci(B,FB),

F is FA+FB.

Anfragen:

?- fibonacci(8,X).

X = 21 .

?- fibonacci(9,X).

X = 34 .

?- fibonacci(10,X).

X = 55

Funktional-logische Programmierung in Prolog

Beispiel: Türme von Hanoi

The screenshot shows the SWISH web interface for Prolog. The browser address bar displays `swish.swi-prolog.org`. The SWISH logo and menu (File, Edit, Examples, Help) are at the top. The main editor contains a Prolog program for the Tower of Hanoi:

```
1 umsetz(1,X,Y,_):- write('Ziehe oberste Scheibe von '),
2                  write(X), write(' nach '), write(Y),nl.
3 umsetz(N,X,Y,Z):-N>1, M is N-1,
4                  umsetz(M,X,Z,Y),
5                  umsetz(1,X,Y,_),
6                  umsetz(M,Z,Y,X).
```

Below the code is a small circle icon. On the right, a console window titled `umsetz(3,a,b,c).` shows the execution output:

```
Ziehe oberste Scheibe von a nach b
Ziehe oberste Scheibe von a nach c
Ziehe oberste Scheibe von b nach c
Ziehe oberste Scheibe von a nach b
Ziehe oberste Scheibe von c nach a
Ziehe oberste Scheibe von c nach b
Ziehe oberste Scheibe von a nach b
true
```

Below the output are buttons for `Next`, `10`, `100`, `1,000`, and `Stop`. At the bottom right, there are buttons for `Examples`, `History`, `Solutions`, a checkbox for `table results`, and a `Run!` button.

Funktional-logische Programmierung in Prolog

Weiterführende Aspekte:

Nach Ausgabe einer Lösung für eine Prolog-Anfrage können weitere Lösungen erfragt werden.

→ Eingabe 'n' veranlasst Backtracking und eventuelle Berechnung weiterer Lösungen. Entspricht Next-Button in swish-Webanwendung

Werden keine weiteren Lösungen benötigt

→ Eingabe 'y' bricht weitere Verarbeitung ab, entspricht Stop-Button in swish-Webanwendung

Durch ! (s.g. Cut-Operation) kann Backtracking auf Klauseln auf linker Seite verhindert werden.