Logische Programmierung

Vorlesung 9: Einbauprädikate. Komplexe Programme

Babeş-Bolyai Universität, Department für Informatik, Cluj-Napoca
csacarea@cs.ubbcluj.ro

Bestimmung des Termtyps

`var(Term):` `yes` falls der Term eine freie Variable ist
`nonvar(Term):` `yes` falls nicht
`integer(Term):` `yes` falls Term mit einer Integerzahl gebunden ist
`float(Term):` `yes` falls Term mit einer Floatingpointzahl gebunden ist
`number(Term):` `yes` falls Term entweder Integer oder Float
`atom(Term):` `yes` falls Term ein Atom ist e.g. john 'John'
`string(Term):` `yes` falls Term ein String ist
`atomic(Term):` `yes` falls Term ein Atom, eine Zahl oder ein String ist

Termvergleich

`Term1 == Term2:` `yes` falls Term1 gleich mit Term2
`Term1 \== Term2:` falls ungleich
`Term1 = Term2:` `yes` falls Unifikation der Terme erfolgreich
`Term 1 \= Term2`: falls Unifikation nicht erfolgreich

ZAHLENVERGLEICH

`between(Low,High,Value):`

`Low` und `High` sind ganze Zahlen und `High >= Low`. Wenn `Value` eine ganze Zahl ist wird getestet: `Low =< Value =< High`

```
?- between(1,3,X).
X = 1 ;X = 2 ;X = 3 ;
```

Wird `High` mit `inf` oder `infinite` angegeben, produziert das Prädikat Integerzahlen hochzählend von `Low` aus.

ZAHLENVERGLEICH

`succ(Int1,Int2):` `true` wenn `Int2 = Int1 +1` und `Int1 >= 0`

`plus(Int1,Int2,Int3):` `true` wenn `Int1+Int2 =:= Int3`

Vergleichsoperatoren

`<, >, =<, >=, =\=, =:=`

`Number is Expression:` `true` wenn `Number` erfolgreich mit der ganzzahligen Auswertung von Expression unifiziert wird

LISTENPRÄDIKATE

`append(List1,List2,List3)` hängt an die erste Liste die zweite an

`member(Element, List)` liefert `yes` falls `Element` in `List` ist

`nextto(X,Y,List):` `yes` falls `Y` hinter `X` in `List` kommt

`delete(List1, X, List2)`: alle `X` in `List1` löschen

`select(Element,List,Rest)`: `Element` in `List` löschen, `Rest` ausgeben; kann als `insert` benutzt werden

`nth0(Index,List,Element):` `yes` wenn Element das `Index`-te in `List` ist; Zählen startet bei 0

`nth1(Index,List,Element):` wie vorher, Zählen startet bei 1

Listenprädikate

`reverse(List1,List2)`: Ordnung der Elemente umkehren
`permutation(List1,List2)`: Elemente anders anordnen
`flatten(List1,List2)`: flache Liste rekursiv anordnen
`sumlist(List,Sum)`: summieren der Elemente
`numlist(Low,High,List)`: Integerliste zwischen `Low` und `High`

Mengenprädikate

`is_set(List):` `yes` wenn Liste ohne Duplikate ist

`list_to_set(List)`: Duplikate werden gelöscht, nur das erste Auftreten wird behalten

`intersection(Set1,Set2,Set3)`: Elemente, die sowohl in Menge1 als auch in Menge2 vorkommen

`subtract(Set,Delete,Result)`: lösche aus `Set` alle Elemente aus `Delete`

`union(Set1,Set2,Set3)`: `Set1,Set2` sind Listen ohne Duplikate, `Set3` ist die Vereinigung

`subset(Subset, Set)`: Alle Elemente aus `Subset` sind in `Set` enthalten

Stringmanipulation

In SWI Prolog sind single-quoted Zeichenketten Atome, doublequoted Zeichenketten dagegen Listen der Einzelbuchstaben

```
?-atom(''abc''). no
?-atom('abc'). yes.
?- append(''Max'',''moni'',L).
L=[77,97,120,109,111,105]
?- append('Max','moni',L). no.
```

STRINGMANIPULATION

`string_to_atom(String,Atom)`: eines der Argumente muss instantiert sein:

`?- string_to_atom(X,abc). X = ''abc''`

`string_to_list(String,List)`: Die einzelnen Buchstaben von `String` werden mit ihren Zeichencodes in `List` geschrieben

`string_length(String,Length)`: gibt die Zeichenzahl eines Strings aber auch eines Atoms an.

`string_concat(S1,S2,S3)`: Zusammenhängen zweier Strings

`sub_string(String,Start,Length,After,Sub)`: `Sub` ist ein Substring von `String`, der bei Start anfängt mit einer Länge von `Length`, dann sind noch `After` Buchstaben übrig

Sonstige Einbauprädikate

`Term =.. List`: zerlegt einen Term in eine Liste

```
?- f(1,2) =..X.
X= [f,1,2]
```

`functor(Term,Functor,Arity)`: gibt den Funktor und die Stelligkeit aus

`help(Prädikat)`: zeigt den Hilfstext an

`listing(Prädikat)`: zeigt die Implementierung an

```
?- listing(member)
lists:member(A,[A|_]).
lists:member(A,[_|B]):- member(A,B).
true.
```

Beispielprogramm zu =..

=.. Term umwandeln zu Funktor, Argumenten in einer Liste

Stellen uns vor, dass wir ein Programm haben, das geometrische Figuren manipuliert. Funktor gibt den Typ der Figur an, Argumente spezifizieren die Größe.

```
Figuren
square(Side)
square(Side1,Side2,Side3)
circle(Radius)
```

Mögliche Operation `enlarge(Figure,Factor,Figure1)`

Das Einsteinrätsel

Dieses Rätsel wurde wahrscheinlich von Albert Einstein [1879-1955] entwickelt. Er versah es mit dem Vermerk, dass nur 2% der Bevölkerung in der Lage seien, es zu lösen. Es ist tatsächlich durch reine Logik lösbar.

Fünf Häuser stehen nebeneinander. In ihnen wohnen Menschen von fünf unterschiedlichen Nationalitäten, die fünf unterschiedliche Getränke trinken, fünf unterschiedliche Zigarettenmarken rauchen und fünf unterschiedliche Haustiere haben.

1. Der Brite lebt im roten Haus.
2. Der Schwede hält sich einen Hund.
3. Der Däne trinkt gern Tee.
4. Das grüne Haus steht (direkt) links neben dem weißen Haus.
5. Der Besitzer des grünen Hauses trinkt Kaffee.

Das Einsteinrätsel

6. Die Person, die Pall Mall raucht, hat einen Vogel.
7. Der Mann im mittleren Haus trinkt Milch.
8. Der Bewohner des gelben Hauses raucht Dunhill.
9. Der Norweger lebt im ersten Haus.
10. Der Marlboro-Raucher wohnt neben der Person mit der Katze.
11. Der Mann mit dem Pferd lebt neben der Person, die Dunhill raucht.
12. Der Winfield-Raucher trinkt gern Bier.
13. Der Norweger wohnt neben dem blauen Haus.
14. Der Deutsche raucht Rothmanns.
15. Der Marlboro-Raucher hat einen Nachbarn, der Wasser trinkt.

Wem gehört der Fisch?

Prolog Lösung

Jedes Haus ist eine Liste der Form [Farbe, Nationalitaet, Getraenk, Zigarettenmarke, Haustier].

Hilfsprädikate zur Listenbearbeitung:

```
erstes(E,[E|_]).
mittleres(M,[_,_,M,_,_]).

links(A,B,[A,B|_]).
links(A,B,[_|R]) :- links(A,B,R).

neben(A,B,L) :- links(A,B,L);links(B,A,L).
```

Prolog Lösung

Lösungsprädikat:

```
run :-
X = [_,_,_,_,_], % Es gibt (nebeneinander) 5
(noch unbekannte) Häuser
```

Prolog Lösung

Lösungsprädikat:

```
run :-
X = [_,_,_,_,_], % Es gibt (nebeneinander) 5
(noch unbekannte) Häuser
member([rot,brite,_,_,_],X), % Der Brite lebt
im roten Haus
```

Prolog Lösung

Lösungsprädikat:

```
run :-
X = [_,_,_,_,_], % Es gibt (nebeneinander) 5
(noch unbekannte) Häuser
member([rot,brite,_,_,_],X), % Der Brite lebt
im roten Haus
member([_,schwede,_,_,hund],X), % Der Schwede
hält einen Hund
```

Prolog Lösung

Lösungsprädikat:

```
run :-
X = [_,_,_,_,_], % Es gibt (nebeneinander) 5
(noch unbekannte) Häuser
member([rot,brite,_,_,_],X), % Der Brite lebt
im roten Haus
member([_,schwede,_,_,hund],X), % Der Schwede
hält einen Hund
member([_,daene,tee,_,_],X), % Der Däne trinkt
gern Tee
```

Prolog Lösung

Lösungsprädikat:

```
run :-
X = [_,_,_,_,_], % Es gibt (nebeneinander) 5
(noch unbekannte) Häuser
member([rot,brite,_,_,_],X), % Der Brite lebt
im roten Haus
member([_,schwede,_,_,hund],X), % Der Schwede
hält einen Hund
member([_,daene,tee,_,_],X), % Der Däne trinkt
gern Tee
links([gruen,_,_,_,_],[weiss,_,_,_,_],X), % Das
grüne Haus steht links vom weißen Haus
```

PROLOG LÖSUNG

Lösungsprädikat:

```
run :-
X = [_,_,_,_,_], % Es gibt (nebeneinander) 5
(noch unbekannte) Häuser
member([rot,brite,_,_,_],X), % Der Brite lebt
im roten Haus
member([_,schwede,_,_,hund],X), % Der Schwede
hält einen Hund
member([_,daene,tee,_,_],X), % Der Däne trinkt
gern Tee
links([gruen,_,_,_,_],[weiss,_,_,_,_],X), % Das
grüne Haus steht links vom weißen Haus
member([gruen,_,kaffee,_,_],X), % Der Besitzer
des grünen Hauses trinkt Kaffee
```

Prolog Lösung

Lösungsprädikat:

```
member([_,_,_,pallmall,vogel],X), % Die Person,
die Pall Mall raucht, hält einen Vogel
```

Prolog Lösung

Lösungsprädikat:

```
member([_,_,_,pallmall,vogel],X), % Die Person,
die Pall Mall raucht, hält einen Vogel
mittleres([_,_,milch,_,_],X), % Der Mann, der im
mittleren Haus wohnt, trinkt Milch
```

PROLOG LÖSUNG

Lösungsprädikat:

```
member([_,_,_,pallmall,vogel],X), % Die Person,
die Pall Mall raucht, hält einen Vogel
mittleres([_,_,milch,_,_],X), % Der Mann, der im
mittleren Haus wohnt, trinkt Milch
member([gelb,_,_,dunhill,_],X), % Der Besitzer
des gelben Hauses raucht Dunhill
```

Prolog Lösung

Lösungsprädikat:

```
member([_,_,_,pallmall,vogel],X), % Die Person,
die Pall Mall raucht, hält einen Vogel
mittleres([_,_,milch,_,_],X), % Der Mann, der im
mittleren Haus wohnt, trinkt Milch
member([gelb,_,_,dunhill,_],X), % Der Besitzer
des gelben Hauses raucht Dunhill
erstes([_,norweger,_,_,_],X), % Der Norweger
wohnt im 1. Haus
```

Prolog Lösung

Lösungsprädikat:

```
member([_,_,_,pallmall,vogel],X), % Die Person,
die Pall Mall raucht, hält einen Vogel
mittleres([_,_,milch,_,_],X), % Der Mann, der im
mittleren Haus wohnt, trinkt Milch
member([gelb,_,_,dunhill,_],X), % Der Besitzer
des gelben Hauses raucht Dunhill
erstes([_,norweger,_,_,_],X), % Der Norweger
wohnt im 1. Haus
neben([_,_,_,marlboro,_],[_,_,_,_,katze],X), %
Der Marlboro-Raucher wohnt neben dem, der eine
Katze hält
```

Prolog Lösung

Lösungsprädikat:

```
member([_,_,_,pallmall,vogel],X), % Die Person,
die Pall Mall raucht, hält einen Vogel
mittleres([_,_,milch,_,_],X), % Der Mann, der im
mittleren Haus wohnt, trinkt Milch
member([gelb,_,_,dunhill,_],X), % Der Besitzer
des gelben Hauses raucht Dunhill
erstes([_,norweger,_,_,_],X), % Der Norweger
wohnt im 1. Haus
neben([_,_,_,marlboro,_],[_,_,_,_,katze],X), %
Der Marlboro-Raucher wohnt neben dem, der eine
Katze hält
neben([_,_,_,_,pferd],[_,_,_,dunhill,_],X), % Der
Mann, der ein Pferd hält, wohnt neben dem, der
Dunhill raucht
```

PROLOG LÖSUNG

Lösungsprädikat:

```
member([_,_,bier,winfield,_],X), % Der
Winfield-Raucher trinkt gern Bier
```

Prolog Lösung

Lösungsprädikat:

```
member([_,_,bier,winfield,_],X), % Der
Winfield-Raucher trinkt gern Bier
neben([_,norweger,_,_,_],[blau,_,_,_,_],X), % Der
Norweger wohnt neben dem blauen Haus
```

Prolog Lösung

Lösungsprädikat:

```
member([_,_,bier,winfield,_],X), % Der
Winfield-Raucher trinkt gern Bier
neben([_,norweger,_,_,_],[blau,_,_,_,_],X), % Der
Norweger wohnt neben dem blauen Haus
member([_,deutsche,_,rothmans,_],X), % Der
Deutsche raucht Rothmans
```

PROLOG LÖSUNG

Lösungsprädikat:

```
member([_,_,bier,winfield,_],X), % Der
Winfield-Raucher trinkt gern Bier
neben([_,norweger,_,_,_],[blau,_,_,_,_],X), % Der
Norweger wohnt neben dem blauen Haus
member([_,deutsche,_,rothmans,_],X), % Der
Deutsche raucht Rothmans
neben([_,_,_,marlboro,_],[_,_,wasser,_,_],X), %
Der Marlboro-Raucher hat einen Nachbarn, der
Wasser trinkt
```

PROLOG LÖSUNG

Lösungsprädikat:

```
member([_,_,bier,winfield,_],X), % Der
Winfield-Raucher trinkt gern Bier
neben([_,norweger,_,_,_],[blau,_,_,_,_],X), % Der
Norweger wohnt neben dem blauen Haus
member([_,deutsche,_,rothmans,_],X), % Der
Deutsche raucht Rothmans
neben([_,_,_,marlboro,_],[_,_,wasser,_,_],X), %
Der Marlboro-Raucher hat einen Nachbarn, der
Wasser trinkt
member([_,N,_,_,fisch],X), % Der mit der
Nationalität N hat einen Fisch
```

Prolog Lösung

Lösungsprädikat:

```
member([_,_,bier,winfield,_],X), % Der
Winfield-Raucher trinkt gern Bier
neben([_,norweger,_,_,_],[blau,_,_,_,_],X), % Der
Norweger wohnt neben dem blauen Haus
member([_,deutsche,_,rothmans,_],X), % Der
Deutsche raucht Rothmans
neben([_,_,_,marlboro,_],[_,_,wasser,_,_],X), %
Der Marlboro-Raucher hat einen Nachbarn, der
Wasser trinkt
member([_,N,_,_,fisch],X), % Der mit der
Nationalität N hat einen Fisch
write(X),nl, % Ausgabe aller Häuser
```

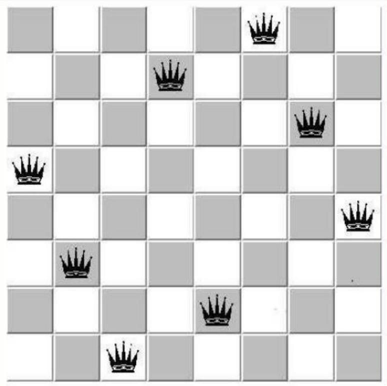
Prolog Lösung

Lösungsprädikat:

```
member([_,_,bier,winfield,_],X), % Der
Winfield-Raucher trinkt gern Bier
neben([_,norweger,_,_,_],[blau,_,_,_,_],X), % Der
Norweger wohnt neben dem blauen Haus
member([_,deutsche,_,rothmans,_],X), % Der
Deutsche raucht Rothmans
neben([_,_,_,marlboro,_],[_,_,wasser,_,_],X), %
Der Marlboro-Raucher hat einen Nachbarn, der
Wasser trinkt
member([_,N,_,_,fisch],X), % Der mit der
Nationalität N hat einen Fisch
write(X),nl, % Ausgabe aller Häuser
write('Der '),write(N),write(' hat einen Fisch
als Haustier.'),nl. % Antwort auf die Frage
```

Das 8-Damen Problem

Man platziere acht Damen so auf einem Schachbrett, dass sie sich gegenseitig nicht bedrohen.



Das 8-Damen Problem

Beschreibung des Problems

Für jedes Feld des Schachbretts eine aussagenlogische Variable D_{ij} die den Wert wahr hat, wann immer auf dem Feld (i, j) eine Dame steht.

Das 8-Damen Problem

Beispiel

Auf dem Feld $(5, 7)$ steht eine Dame $\mapsto D_{57}$ wahr.

Einschränkungen pro Feld F_{ij}

Falls auf dem Feld $(5, 7)$ eine Dame steht:

- keine andere Dame auf Feld (5,1), (5,2), (5,3), (5,4),(5,5), (5,6), (5,8);
- keine andere Dame auf Feld (1,7), (2,7), (3,7), (4,7),(6,7), (7,7), (8,7);
- keine andere Dame auf Feld (6,8), (4,6), (3,5), (2,4), (1,3);
- keine andere Dame auf Feld (4,8), (6,6), (7,5), (8,4);
- (ähnliche Bedingungen für alle Felder (i, j)).

Das 8-Damen Problem

Einschränkungen pro Feld F_{ij}

- keine andere Dame auf Feld (5,1), (5,2), (5,3), (5,4),(5,5), (5,6), (5,8);

Das 8-Damen Problem

Einschränkungen pro Feld F_{ij}

- keine andere Dame auf Feld (5,1), (5,2), (5,3), (5,4),(5,5), (5,6), (5,8);
- $D_{5,7} \rightarrow \neg D_{5,8} \wedge \neg D_{5,6} \wedge \neg D_{5,5} \wedge \neg D_{5,4} \wedge \neg D_{5,3} \wedge \neg D_{5,2} \wedge \neg D_{5,1}$;

Das 8-Damen Problem

Einschränkungen pro Feld F_{ij}

- keine andere Dame auf Feld (5,1), (5,2), (5,3), (5,4),(5,5), (5,6), (5,8);
- $D_{5,7} \rightarrow \neg D_{5,8} \wedge \neg D_{5,6} \wedge \neg D_{5,5} \wedge \neg D_{5,4} \wedge \neg D_{5,3} \wedge \neg D_{5,2} \wedge \neg D_{5,1}$;
- keine andere Dame auf Feld (1,7), (2,7), (3,7), (4,7),(6,7), (7,7), (8,7);

Das 8-Damen Problem

Einschränkungen pro Feld F_{ij}

- keine andere Dame auf Feld (5,1), (5,2), (5,3), (5,4),(5,5), (5,6), (5,8);
- $D_{5,7} \rightarrow \neg D_{5,8} \wedge \neg D_{5,6} \wedge \neg D_{5,5} \wedge \neg D_{5,4} \wedge \neg D_{5,3} \wedge \neg D_{5,2} \wedge \neg D_{5,1}$;
- keine andere Dame auf Feld (1,7), (2,7), (3,7), (4,7),(6,7), (7,7), (8,7);
- $D_{5,7} \rightarrow \neg D_{1,7} \wedge \neg D_{2,7} \wedge \neg D_{3,7} \wedge \neg D_{4,7} \wedge \neg D_{6,7} \wedge \neg D_{7,7} \wedge \neg D_{8,7}$;

Das 8-Damen Problem

Einschränkungen pro Feld F_{ij}

- keine andere Dame auf Feld (5,1), (5,2), (5,3), (5,4),(5,5), (5,6), (5,8);
- $D_{5,7} \rightarrow \neg D_{5,8} \wedge \neg D_{5,6} \wedge \neg D_{5,5} \wedge \neg D_{5,4} \wedge \neg D_{5,3} \wedge \neg D_{5,2} \wedge \neg D_{5,1}$;
- keine andere Dame auf Feld (1,7), (2,7), (3,7), (4,7),(6,7), (7,7), (8,7);
- $D_{5,7} \rightarrow \neg D_{1,7} \wedge \neg D_{2,7} \wedge \neg D_{3,7} \wedge \neg D_{4,7} \wedge \neg D_{6,7} \wedge \neg D_{7,7} \wedge \neg D_{8,7}$;
- keine andere Dame auf Feld (6,8), (4,6), (3,5), (2,4), (1,3);
- $D_{5,7} \rightarrow \neg D_{6,8} \wedge \neg D_{4,6} \wedge \neg D_{3,5} \wedge \neg D_{2,4} \wedge \neg D_{1,3}$
- keine andere Dame auf Feld (4,8), (6,6), (7,5), (8,4);
- $D_{5,7} \rightarrow \neg D_{4,8} \wedge \neg D_{6,6} \wedge \neg D_{7,5} \wedge \neg D_{8,4}$;
- (ähnliche Bedingungen für alle Felder (i, j)).

Das 8-Damen Problem

$F_{5,7}$:

$D_{5,7} \rightarrow \neg D_{5,8} \wedge \neg D_{5,6} \wedge \neg D_{5,5} \wedge \neg D_{5,4} \wedge \neg D_{5,3} \wedge \neg D_{5,2} \wedge \neg D_{5,1}$
$D_{5,7} \rightarrow \neg D_{1,7} \wedge \neg D_{2,7} \wedge \neg D_{3,7} \wedge \neg D_{4,7} \wedge \neg D_{6,7} \wedge \neg D_{7,7} \wedge \neg D_{8,7}$
$D_{5,7} \rightarrow \neg D_{6,8} \wedge \neg D_{4,6} \wedge \neg D_{3,5} \wedge \neg D_{2,4} \wedge \neg D_{1,3}$
$D_{5,7} \rightarrow \neg D_{4,8} \wedge \neg D_{6,6} \wedge \neg D_{7,5} \wedge \neg D_{8,4}$

Das 8-Damen Problem

Globale Einschränkungen

Das 8-Damen Problem

Globale Einschränkungen

Für jedes k mit $1 \leq k \leq 8$:

$$R_k := D_{1,k} \vee D_{2,k} \vee D_{3,k} \vee D_{4,k} \vee D_{5,k} \vee D_{6,k} \vee D_{7,k} \vee D_{8,k}.$$

DAS 8-DAMEN PROBLEM

Struktur: Wahrheitswerte für die atomaren Aussagen $D_{i,j}$
Modell für $F_{i,j}(R_k)$: Wahrheitswerte fur die atomaren Aussagen $D_{i,j}$ so dass $F_{i,j}$ wahr (bzw. R_k wahr).
Lösung des 8-Damen Problems: Eine aussagenlogische Struktur beschreibt eine Lösung des 8-DamenProblems genau dann, wenn sie ein Modell der Formeln

- $F_{i,j}$ für alle $1 \leq i, j \leq 8$;
- R_k für alle $1 \leq k \leq 8$ ist.

ACHT DAMEN PROBLEM: PROLOG LÖSUNG

Repräsentation der acht Damen auf dem Brett:

Acht Objekte, wobei jedes eine Dame auf einem Feld des Schachbretts repräsentiert.
Jedes Feld kann durch ein Paar von Koordinaten `X,Y` dargestellt werden.
Die Koordinaten sind Integer zwischen 1 und 8.
Im Programm schreiben wir `X/Y`. '/' steht hier nicht für Division sondern kombiniert die Koordinaten eines Feldes.
Das Problem ist nun also, eine Liste zu finden wo keine Dame die andere bedroht.

```
[X1/Y1,X2/Y2,X3/Y3,X4/Y4,X5/Y5,X6/Y6,X7/Y7,X8/Y8
]
```

Acht Damen Problem: Prolog Lösung

Eine Dame bedroht die andere:

vertikal, horizontal, diagonal nach oben, diagonal nach unten.
Die Bedrohung ist symmetrisch wir können nach rechts auf dem Brett vorgehen.
Ein Teil der Lösung kann schon in der Repräsentation der Lösung formuliert werden. Alle Damen müssen offensichtlich in verschiedenen Spalten sein, damit keine vertikale Bedrohung eintritt. Wir halten in der Datenrepräsentation die X Koordinate fest.

```
[1/Y1,2/Y2,3/Y3,4/Y4,5/Y5,6/Y6,7/Y7,8/Y8 ]
```

Acht Damen Problem: Prolog Lösung

Generalisierung zur Vereinfachung des Problems:

Finde die Lösung für 0-8 Damen. Das bedeutet die Liste der Damen kann $0-8$ Objekte der Form `X/Y` enthalten.

Fall1:

Die Liste der Damen ist leer. Dies ist eine Lösung, da hier keine Dame die andere bedroht.

Acht Damen Problem: Prolog Lösung

Fall2:

Die Liste ist nicht leer `[X/Y|Others]`. Die erste Dame steht also auf einem Feld `X/Y` und die anderen Damen stehen auf anderen Feldern, die in `Others` spezifiziert sind. Bedingungen einer Lösung:

1. Keine Bedrohung innerhalb der Liste `Others`, `Others` ist Lösung
2. `X` und `Y` sind integer zwischen 1 und 8
3. Die Dame auf `X/Y` darf keine der Damen in `Others` bedrohen

Acht Damen Problem: Prolog Lösung

Fall 1:

```
solution([ ]).
```

Fall 2:

1. Aufteilung des Problems durch rekursiven Aufruf.
2. `Y` muss `member` von `[1,2,3,4,5,6,7,8]` sein. `X` wird dadurch beschränkt, dass jede Lösung dem Template entsprechen muss, wo die `X` Koordinaten schon festgelegt sind.
3. Nicht-Bedrohung wird durch ein Prädikat `noattack` implementiert

Acht Damen Problem: Prolog Lösung

Fall 2:

```
solution([X/Y|Others]) :-
solution(Others),
member(Y,[1,2,3,4,5,6,7,8]),noattack(X/Y,Others).
```

Bleibt: `noattack` Prädikat `noattack(Q,QList).`

1. Wenn `QList` leer ist, ist `noattack` erfüllt. Es gibt nur eine Dame auf dem Brett.
2. Wenn `QList` nicht leer ist, hat sie die Form `[Q1|Qlist1]`
   1. Die Dame `Q1` darf die Damen in `QList1` nicht bedrohen
   2. Die Dame `Q` darf nicht `Q1` bedrohen und auch keine der Damen in `Qlist1`.

ACHT DAMEN PROBLEM: PROLOG LÖSUNG

Implementation der Nichtbedrohung der Damen: `X/Y` und `X1/Y1` horizontal, vertikal, diagonal

Vertikal: `X` Koordinaten müssen unterschiedlich sein, gewährleistet schon das Template

Horizontal: `Y` Koordinaten müssen unterschiedlich sein `Y =\= Y1`

Diagonal: weder aufwärts noch abwärts darf der Abstand der Felder in `X` Richtung gleich demjenigen in der `Y` Richtung sein.

`Y1 - Y =\= X1 - X` aufwärts (`X1 - X` positiv)

`Y1 -Y =\= X - X1` abwärts (`X - X1` negativ)

Acht Damen Problem. Komplette Lösung

```
solution([ ]).
solution([X/Y|Others]):- solution(Others),
member(Y,[1,2,3,4,5,6,7,8]),
noattack(X/Y,Others).
noattack(_,[ ]).
noattack(X/Y,[X1/Y1|RestOthers]):-
Y=\=Y1,
Y1 - Y =\= X1 - X, %aufwärts diagonal
Y1 - Y =\= X - X1, %abwärts
noattack(X/Y,RestOthers).
```

Acht Damen Problem. Komplette Lösung

Hilfsprädikat:

```
member(Item,[Item|Rest]).
member(Item,[First|Rest]):- member(Item,Rest).
```

Lösungstemplate:

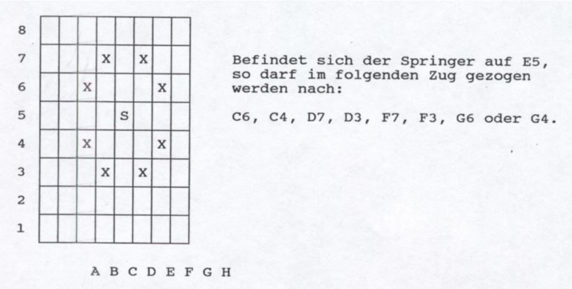
```
template([1/Y1,2/Y2,3/Y3,4/Y4,5/Y5,6/Y6,7/Y7,8/Y8])
```

Aufruf: `?- template(S),solution(S).`

```
S=[1/4,2/2,3/7,4/3,5/6,6/8,7/5,8/1]
```

Das Springer-Problem

Auf einem Schachbrett soll ein Springer, ausgehend von einem festzulegenden Feld (häufig beschränkt man sich auf ein Eckfeld), über eine geeignete Zugfolge alle Felder durchlaufen, so dass jedes Feld genau einmal besetzt wird. Sofern nicht Sprünge wegen der Randnähe unmöglich sind, stehen dem Springer jeweils acht Zugmöglichkeiten offen



DAS SPRINGER-PROBLEM

Nummeriert man die Felder in der Reihenfolge des Durchlaufs und beginnt im linken oberen Eckfeld, so ergibt sich eine Lösung durch folgende Anordnung:

8	1	16	27	22	3	18	47	56
7	26	23	2	17	46	57	4	19
6	15	28	25	62	21	48	55	58
5	24	35	30	45	60	63	20	5
4	29	14	61	34	49	44	59	54
3	36	31	38	41	64	53	6	9
2	13	40	33	50	11	8	43	52
1	32	37	12	39	42	51	10	7
	A	B	C	D	E	F	G	H
	1	2	3	4	5	6	7	8

Das Springer-Problem

Das Springerproblem kann für jedes quadratische Feld mit mindestens fünf Zeilen gelöst werden. Auch für nichtquadratische Felder kann es Lösungen geben, wie das folgende Beispiel zeigt:

8	1	4	35	30	25	6	23	46
7	34	31	2	5	36	45	26	7
6	3	42	33	52	29	24	47	22
5	32	51	40	37	44	49	8	27
4	41	38	43	50	53	28	21	48
3				39	16	19	12	9
2				54	11	14	17	20
1				15	18	55	10	13
	A	B	C	D	E	F	G	H
	1	2	3	4	5	6	7	8

Das Springer-Problem

Zuerst ist ein Prädikat zur Realisierung der Springerzüge zu formulieren:

```
springer_zug(Von,Nach)
```

Hierbei sind Von und Nach Variablen für die Stellungen vor und nach dem durchgeführten Springerzug. Die Stellungen des Springers werden wie beim Acht-Damen-Problem durch Paare beschrieben. Ein Springerzug ist erlaubt, wenn eine der acht oben angegebenen Wahlmöglichkeiten vorkommt und die Koordinaten im richtigen Intervall liegen:

Das Springer-Problem

```
springer_zug([X_alt,Y_alt],[X_neu,Y_neu]):-
X_neu is X_alt+2, Y_neu is Y_alt+1, intervall(X_neu,Y_neu).
springer_zug([X_alt,Y_alt], [X_neu,Y_neu]):-
X_neu is X_alt+2, Y_neu is Y_alt - 1, intervall(X_neu,Y_neu).
springer_zug([X_alt,Y_alt], [X_neu,Y_neu]):-
X_neu is X_alt - 2, Y_neu is Y_alt+1, intervall(X_neu,Y_neu).
springer_zug([X_alt,Y_alt], [X_neu,Y_neu]):-
X_neu is X_alt - 2, Y_neu is Y_alt - 1, intervall(X_neu,Y_neu).
springer_zug([X_alt,Y_alt], [X_neu,Y_neu]):-
X_neu is X_alt+1, Y_neu is Y_alt+2, intervall(X_neu,Y_neu).
springer_zug([X_alt,Y_alt],[X_neu,Y_neu]):-
X_neu is X_alt - 1, Y_neu is Y_alt+2, intervall(X_neu,Y_neu).
springer_zug([X_alt,Y_alt], [X_neu,Y_neu]):-
X_neu is X_alt+1, Y_neu is Y_alt - 2, intervall(X_neu,Y_neu).
springer_zug([X_alt,Y_alt], [X_neu,Y_neu]):-
X_neu is X_alt - 1, Y_neu is Y_alt - 2, intervall(X_neu,Y_neu).
```

Das Springer-Problem

Nachdem geklärt ist, welche Springerzüge erlaubt sind, können wir an die Formulierung der Klauseln zur Bestimmung einer zulässigen Zugfolge des Springers gehen. Hierzu verwenden wir das zweistellige Prädikat

```
zugfolge(X,Y)
```

Hierbei soll X die (geordnete) Liste der durchlaufenen Felder (also Paare von Koordinaten) und Y die Menge der hierzu zur Verfügung stehenden Felder sein. X und Y enthalten also im wesentlichen die gleichen Elemente (unten wird das Startfeld in Y weggelassen), Y in beliebig vorgegebener, X in korrekter Reihenfolge der Züge des Springers.

Das Springer-Problem

Eine mit den Feldern X und Y beginnende Zugfolge ist korrekt, wenn Y in der Menge der noch nicht benutzten Felder (d.h. noch zur Verfügung stehenden Felder) gelöscht werden kann, der Zug von X nach Y ein erlaubter Springerzug ist und die restliche mit Y beginnende Zugfolge korrekt ist.

Die Suche nach einer Zugfolge soll abbrechen, wenn keine Felder mehr benutzt werden können.

Die folgenden beiden Klauseln enthalten dann einen Vorschlag zur Realisierung:

```
zugfolge([X],[ ]).
zugfolge([X,Y|Rest],Z):- loesche(Y,Z,Z_Rest),
springer_zug(X,Y),
zugfolge([Y|Rest],Z_Rest).
```

Das Springer-Problem

Alle Lösungen, deren Zugfolge mit dem Feld (1,1) beginnt, können dann mit

```
loesung([[1,1]|X]):-
zugfolge([[1,1]|X], [ [1,2],[1,3],[1,4],[1,5],
[2,1],[2,2],[2,3],[2,4],[2,5],
[3,1],[3,2],[3,3],[3,4],[3,5],
[4,1],[4,2],[4,3],[4,4],[4,5],
[5,1],[5,2],[5,3],[5,4],[5,5]]).
```

beschrieben werden. Die Bestimmung aller Lösungen erfolgt dann über die Anfrage

```
?-loesung(X).
```

DAS SPRINGER-PROBLEM

Allerdings muss man wegen der sehr großen Anzahl von Möglichkeiten lange auf die erste Lösung warten. Kennt man eine Lösung, so kann damit zumindest das obige Programm ohne langes Warten getestet werden, indem man diese Lösung als Zugfolge in die Menge der noch nicht benutzten Felder eingibt:

```
loesung([[1,1]|X]):-
zugfolge([[1,1]|X], [ [3,2],[5,1],[4,3],[5,5],
[3,4],[1,5],[2,3],[3,1],[5,2],
[4,4],[2,5],[1,3],[2,1],[4,2],
[5,4],[3,5],[1,4],[2,2],[4,1],
[5,3],[4,5],[2,4],[1,2],[3,3]]).
```

Nach der Anfrage

```
?-loesung(X).
```

wird zuerst die schon vorgegebene Lösung wieder ausgegeben. Anschließend erhält man durch Backtracking aber relativ schnell weitere Lösungen.

Das Springer-Problem. Komplette Lösung

% Erlaubte Intervallgrenzen

`intervall(X,Y):-` $0 < X, X < 6, 0 < Y, Y < 6.$

```
% Erlaubte Springerzüge
springer_zug([X_alt,Y_alt],[X_neu,Y_neu]):-
X_neu is X_alt+2, Y_neu is Y_alt+1,
intervall(X_neu,Y_neu).
springer_zug([X_alt,Y_alt], [X_neu,Y_neu]):-
X_neu is X_alt+2, Y_neu is Y_alt - 1,
intervall(X_neu,Y_neu).
springer_zug([X_alt,Y_alt], [X_neu,Y_neu]):-
X_neu is X_alt - 2, Y_neu is Y_alt+1,
intervall(X_neu,Y_neu).
springer_zug([X_alt,Y_alt], [X_neu,Y_neu]):-
X_neu is X_alt - 2, Y_neu is Y_alt - 1,
intervall(X_neu,Y_neu).
```

DAS SPRINGER-PROBLEM. KOMPLETTE LÖSUNG

```
springer_zug([X_alt,Y_alt], [X_neu,Y_neu]):-
X_neu is X_alt+1, Y_neu is Y_alt+2,
intervall(X_neu,Y_neu).
springer_zug([X_alt,Y_alt],[X_neu,Y_neu]):-
X_neu is X_alt - 1, Y_neu is Y_alt+2,
intervall(X_neu,Y_neu).
springer_zug([X_alt,Y_alt], [X_neu,Y_neu]):-
X_neu is X_alt+1, Y_neu is Y_alt - 2,
intervall(X_neu,Y_neu).
springer_zug([X_alt,Y_alt], [X_neu,Y_neu]):-
X_neu is X_alt - 1, Y_neu is Y_alt - 2,
intervall(X_neu,Y_neu).
```

DAS SPRINGER-PROBLEM. KOMPLETTE LÖSUNG

% Lösung bestimmen

```
zugfolge([X],[ ]).
zugfolge([X,Y|Rest],Z):- loesche(Y,Z,Z_Rest),
springer_zug(X,Y),
zugfolge([Y|Rest],Z_Rest).

loesung([[1,1]|X]):-
zugfolge([[1,1]|X], [ [3,2],[5,1],[4,3],[5,5],
[3,4],[1,5],[2,3],[3,1],[5,2],
[4,4],[2,5],[1,3],[2,1],[4,2],
[5,4],[3,5],[1,4],[2,2],[4,1],
[5,3],[4,5],[2,4],[1,2],[3,3]]).
```