Löschen und Einfügen in eine Liste

Löschen des ersten Elements einer Liste:

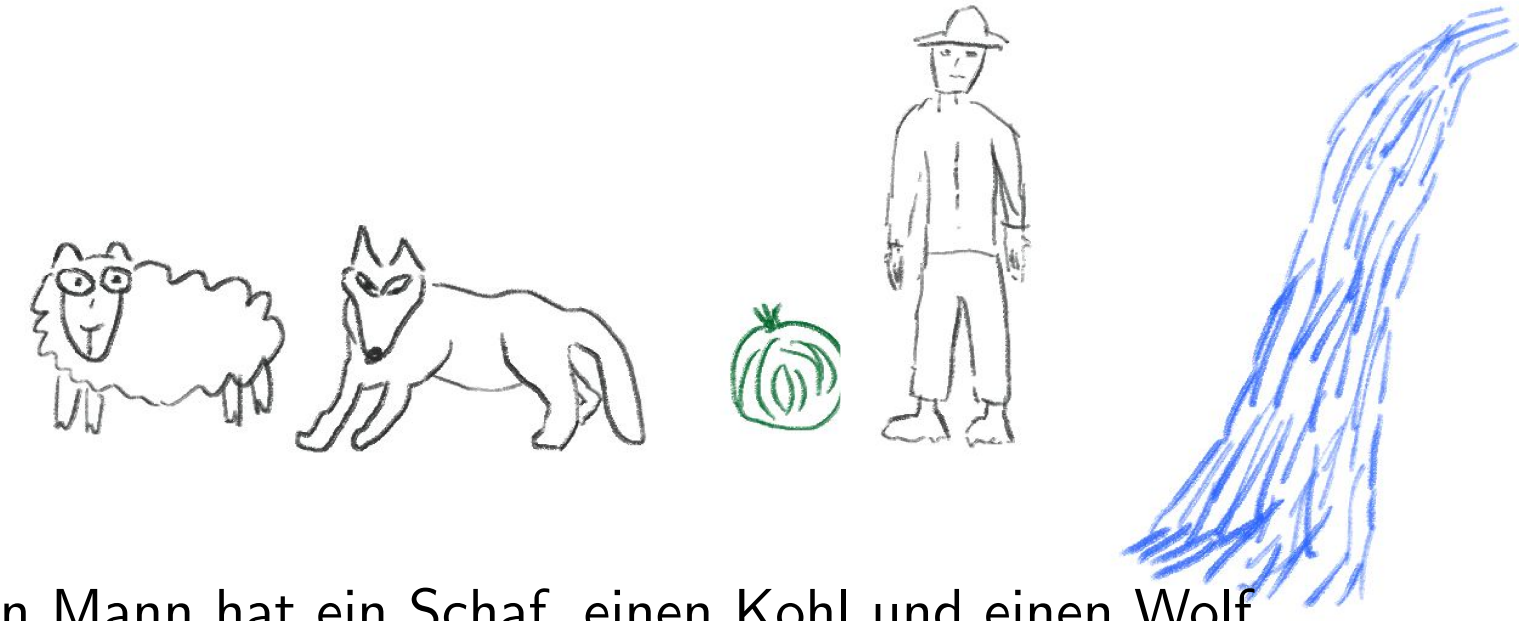
$$loesche\_erstes(X, [X|R], R).$$
$$loesche\_erstes(X, [Y|R], [Y|RR]) :- X \backslash = Y, loesche\_erstes(X, R, RR).$$

Löschen *eines* Elements:

$$loesche(X, [X|R], R).$$
$$loesche(X, [Y|R], [Y|RR]) :- loesche(X, R, RR).$$

Kann man so auch Elemente *einfügen*?

Das Fährproblem



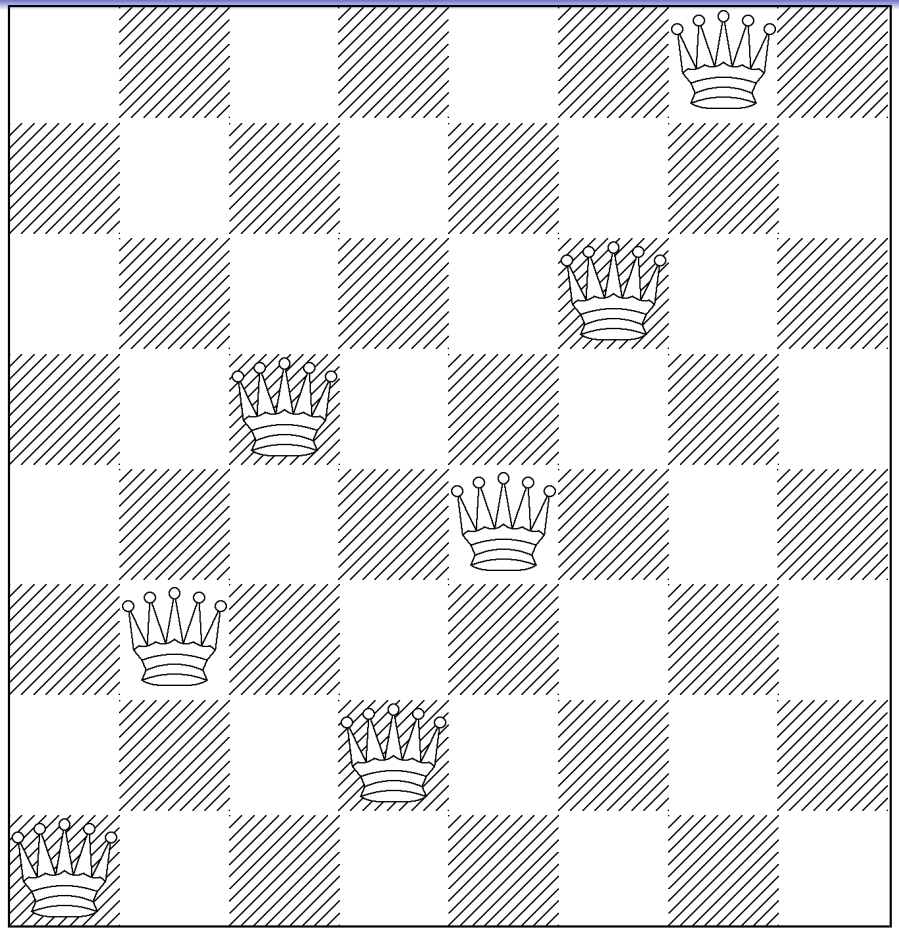
- Ein Mann hat ein Schaf, einen Kohl und einen Wolf.
- Er möchte sie auf die andere Flußseite bringen.
- Er darf das Schaf und den Wolf nicht allein lassen.
- Er darf auch das Schaf und den Kohl nicht allein lassen.
- Im Boot ist nur Platz für den Mann und einen der drei Dinge.

Wie kann er das schaffen?

Das Fährproblem

$tausch([K, W, S, o], [K, W, S, w]).$
$tausch([o, W, S, o], [w, W, S, w]).$
$tausch([K, o, S, o], [K, w, S, w]).$
$tausch([K, W, o, o], [K, W, w, w]).$
$fahrt(X, Y) :- tausch(X, Y).$
$fahrt(X, Y) :- tausch(Y, X).$
$sicher(S) :- kohl\_sicher(S), schaf\_sicher(S).$
$kohl\_sicher([K, \_, \_, K]).$
$kohl\_sicher([w, \_, o, \_]).$
$kohl\_sicher([o, \_, w, \_]).$
$schaf\_sicher([\_, \_, S, S]).$
$schaf\_sicher([\_, w, o, \_]).$
$schaf\_sicher([\_, o, w, \_]).$
$erreichbar(0, [o, o, o, o], [\,]).$
$erreichbar(N, T, [S|L]) :- N > 0, fahrt(S, T), sicher(T),$
$N1\ is\ N - 1, erreichbar(N1, S, L).$

Das Acht-Damen-Problem



Löse das Problem rekursiv für die ersten k Spalten.

Dann versuche die $k + 1$te Spalte zu besetzen.

Das Acht-Damen-Problem

$$
\begin{aligned}
&sol(0, [\,]).\\
&sol(N, [Y|L]) :-\\
&\quad N > 0, N1 \ is \ N - 1, member(Y, [1,2,3,4,5,6,7,8]),\\
&\quad sol(N1, L),\\
&\quad not\_same\_row(Y, L), not\_same\_diag1(Y, L), not\_same\_diag2(Y, L).\\
\\
&not\_same\_row(Y, [\,]).\\
&not\_same\_row(Y, [Y1|R]) :- Y \backslash = Y1, not\_same\_row(Y, R).\\
\\
&not\_same\_diag1(Y, [\,]).\\
&not\_same\_diag1(Y, [Y1|R]) :- YY \ is \ Y - 1, YY \backslash = Y1,\\
&\qquad\qquad not\_same\_diag1(YY, R).\\
&not\_same\_diag2(Y, [\,]).\\
&not\_same\_diag2(Y, [Y1|R]) :- YY \ is \ Y + 1, YY \backslash = Y1,\\
&\qquad\qquad not\_same\_diag1(YY, R).
\end{aligned}
$$

Hinzufügen von Fakten und Regeln

Durch **assert** (und *asserta*, *assertz*) können wir Fakten und Regeln zur Wissensbasis hinzufügen.

Hier ein Beispiel zu Memoization:

$:- \mathit{dynamic}\ \mathit{fib}/2.$
$\mathit{fib}(0,0).$
$\mathit{fib}(1,1).$
$\mathit{fib}(N,M) :-$
$\quad N > 1,$
$\quad N1\ \mathit{is}\ N-1, N2\ \mathit{is}\ N-2,$
$\quad \mathit{fib}(N1,M1),$
$\quad \mathit{fib}(N2,M2),$
$\quad M\ \mathit{is}\ M1+M2,$
$\quad \mathit{asserta}(\mathit{fib}(N,M)).$

Die erste Zeile „erlaubt“ *asserta*.

Ausgabe

Das Ziel *write*(*X*) ist immer erfüllt.

Als Seiteneffekt wird *X* auf den Bildschirm ausgegeben.

Durch *nl* wird eine neue Zeile angefangen.

Türme von Hanoi als Beispiel:

$$
\begin{aligned}
&\mathit{hanoi}(0, A, B, C).\\
&\mathit{hanoi}(N, A, B, C) :-\\
&\quad N > 0,\\
&\quad \mathit{N1}\ \mathit{is}\ N - 1,\\
&\quad \mathit{hanoi}(\mathit{N1}, A, C, B),\\
&\quad \mathit{write}(\texttt{'Nimm Scheibe von '}), \mathit{write}(A),\\
&\quad \mathit{write}(\texttt{' und setze sie auf '}), \mathit{write}(C), \mathit{nl},\\
&\quad \mathit{hanoi}(\mathit{N1}, C, B, A).
\end{aligned}
$$