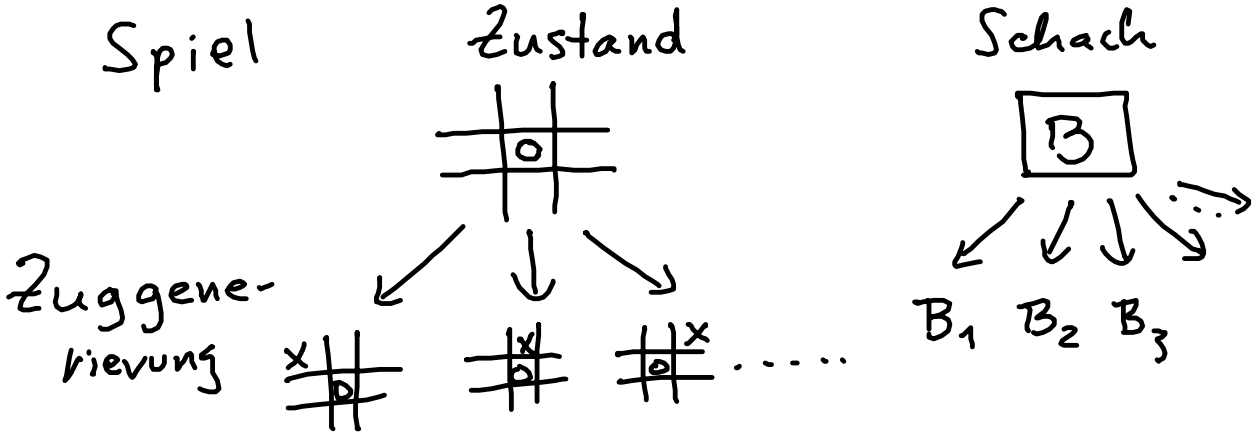
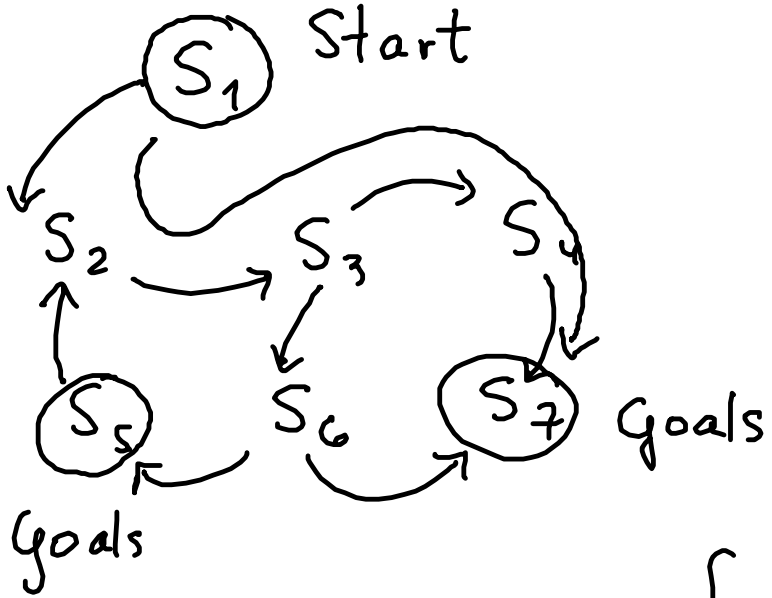
Kap. 3

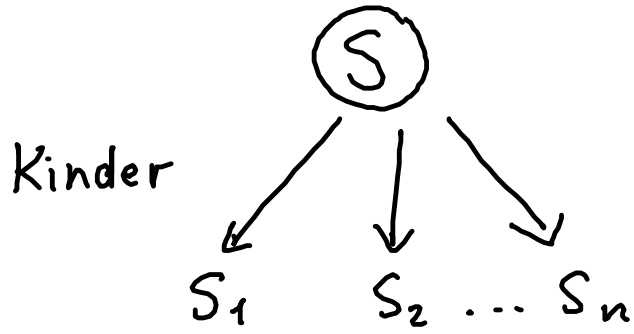
State Space Search

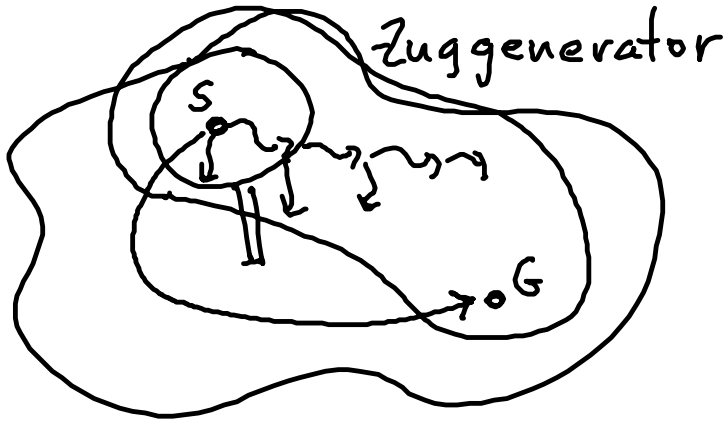




GwA

Zustandsraum { Schachbrett
Tic-Tac-Toe





Graph

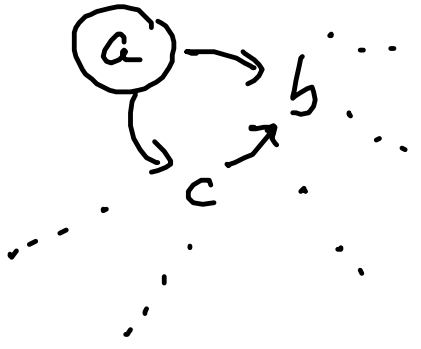
Knoten + Kanten

$G = (V, E)$

$V = \{ s_1, s_2, \dots \}$

$E = \{ (s_1, s_2), (s_2, s_3) \dots \}$

```
con(a,b).
con(a,c).
con(c,b).
⋮
```



2 con(a, X).

X = b;

X = c

?

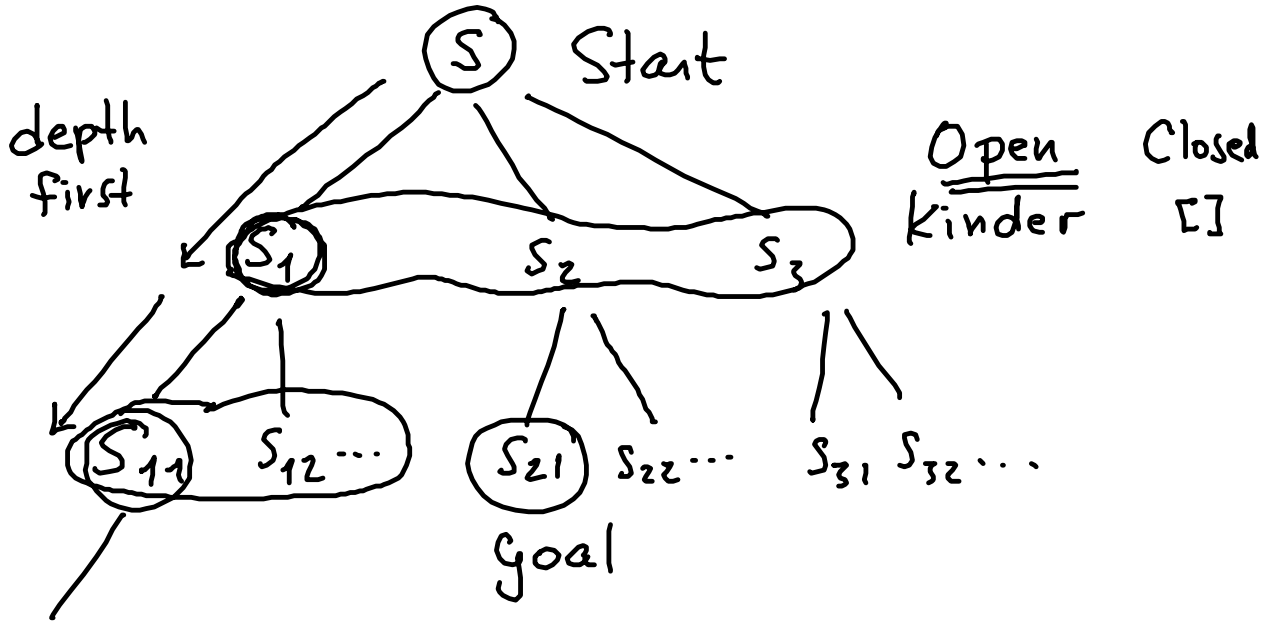
? bagof(X, con(a,X), L).

L = [b, c] L = [b,c,b]

? setof(X, con(a,X), L).

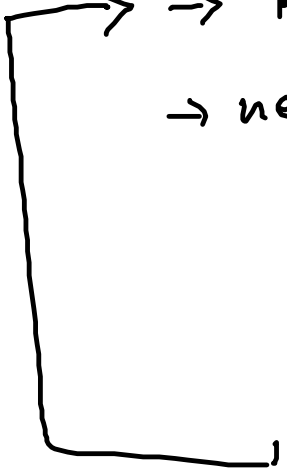
L = [b, c]

con(a,b).
con(a,c).
con(a,b).



S_{111} ...

Schritte



→ Kopf ∈ Goals ? ja, fertig.

→ nein

Kopf raus aus Open
Kopf rein in Closed
Generiere die Liste der Kinder
Tue Kinder → Open

Bsp.

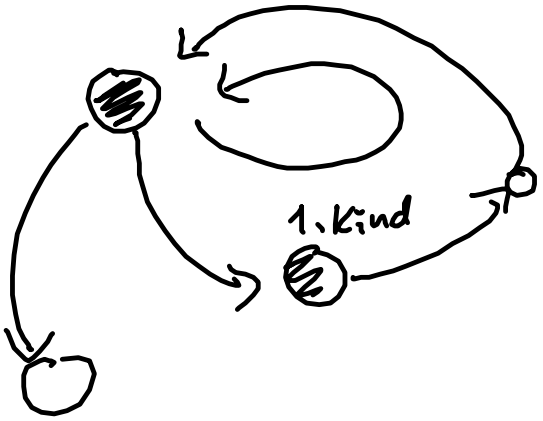
1. { Open $[S_1\ S_2\ S_3]$; Closed $[]$

Tiefensuche

2. { Open $[(S_{11}), S_{12}, \dots S_2, S_3]$; Closed $[S_1]$

Kinder $[S_{11}, S_{12} \dots]$

3. { Open $[S_{111}, S_{112} \dots, S_{12} \dots S_2, S_3]$; Closed $[S_{11}, S_1]$



In Prolog:

Argumente

```
depth ( Goals, Open, Closed)
```

Aufruf:

```
? depth ( [b,c], [a], []).
```

⋮

```
write ( 3)
```

⋮

← hoch problematisch unlogisch

Seiteneffekt

```
depth ( Goals, [X|_], _) :-
    member(X, Goals), !,
    write(X).
```

```
depth ( Goals, [X|RestOpen], Closed):-
        member (X, Closed), !,
        depth ( Goals, RestOpen , Closed).
depth ( Goals, [X|RestOpen], Closed):-
        write (X),
        kinder (X, L),
        appendm ( L, RestOpen, Open),
        depth ( Goals, Open, [X|Closed]).
```

---

kinder

depth (Goals, Open, Closed, Direkterweg)

—·—

```
kinder(X, L) :- setof(Y, con(X,Y), L).

kinder(_, []).
```

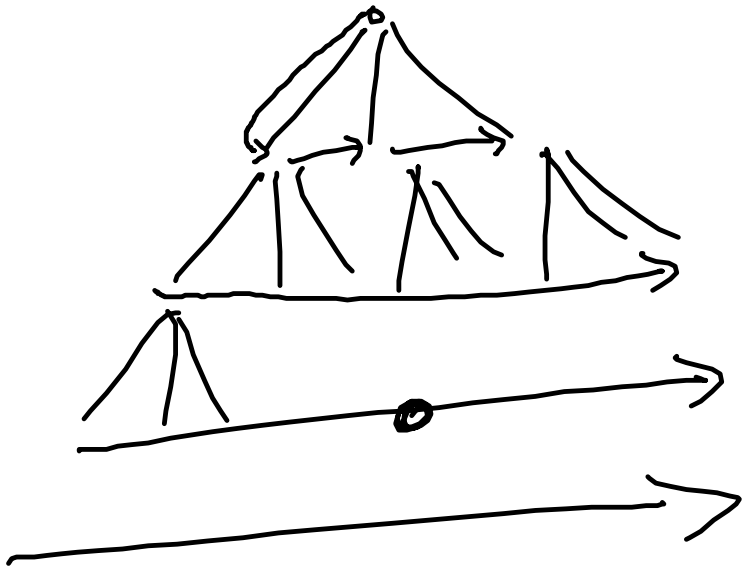
```
appendm([], L, L) :- !.
appendm([X|Y], L, [X|R]) :-
        appendm(Y, L, R).
```

Breitensuche

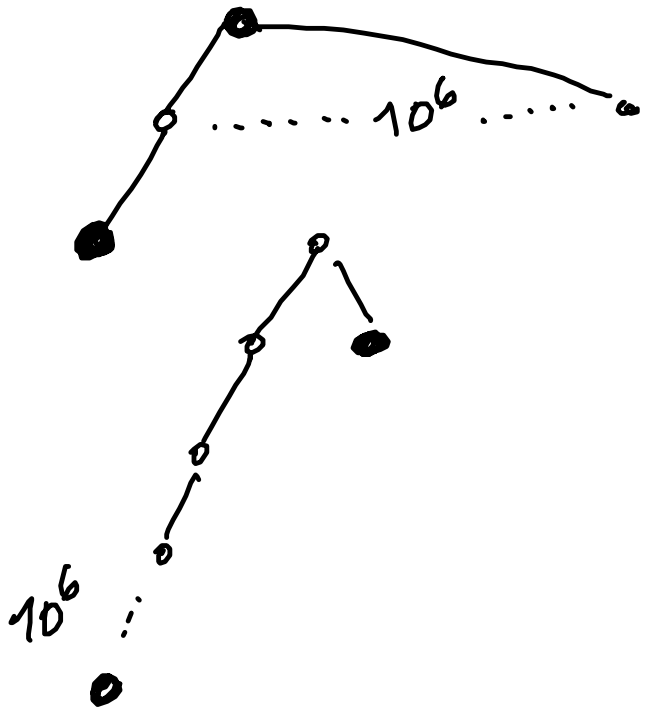
```
breadth(Goals, [X|_], _) :-
        .....
breadth(Goals, [X|RestOpen], Closed) :-
        .....
        breadth(.... )
```

```
breadth(Goals, [X|RestOpen], Closed) :-
        write(X),
```

kinder(X, L),
append(RestOpen, L, Open),
breadth(Goals, Open, [X|Closed]).



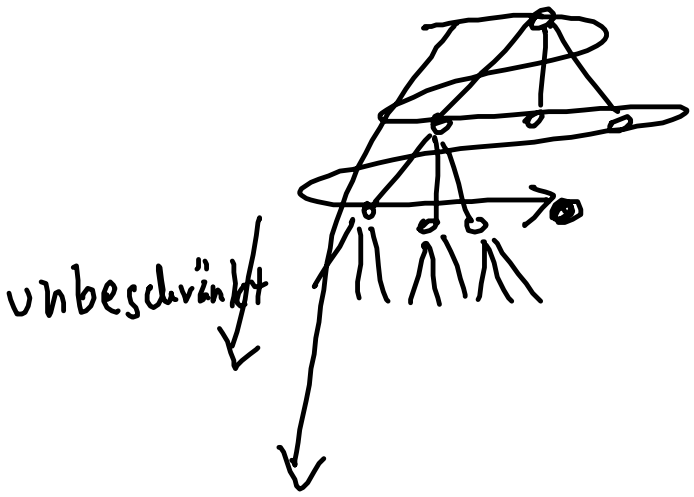
Besser?



Prolog?

1) Prolog ?

2) Terminiert ?



In einem unendlichen Baum (Tiefe) findet Breitensuche immer eine Lösung

a ← b, c, d.

b :- e, f, g.

e :- h, i, j.

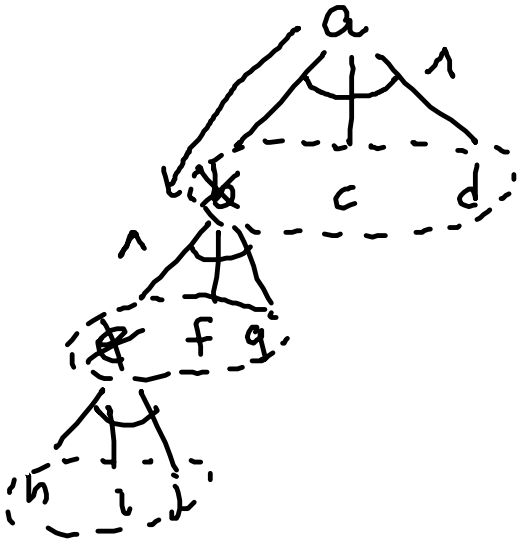
c :- a, b.

?a.

b ∧ c ∧ d ?

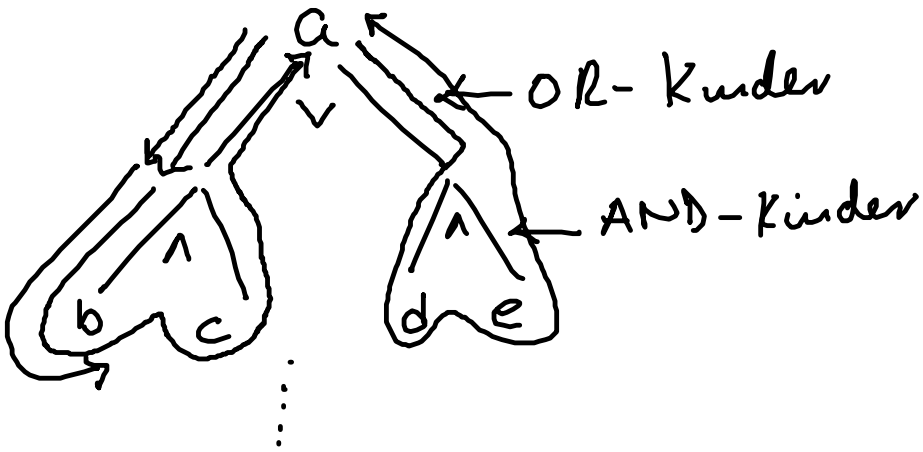
(e ∧ f ∧ g) ∧ c ∧ d ?

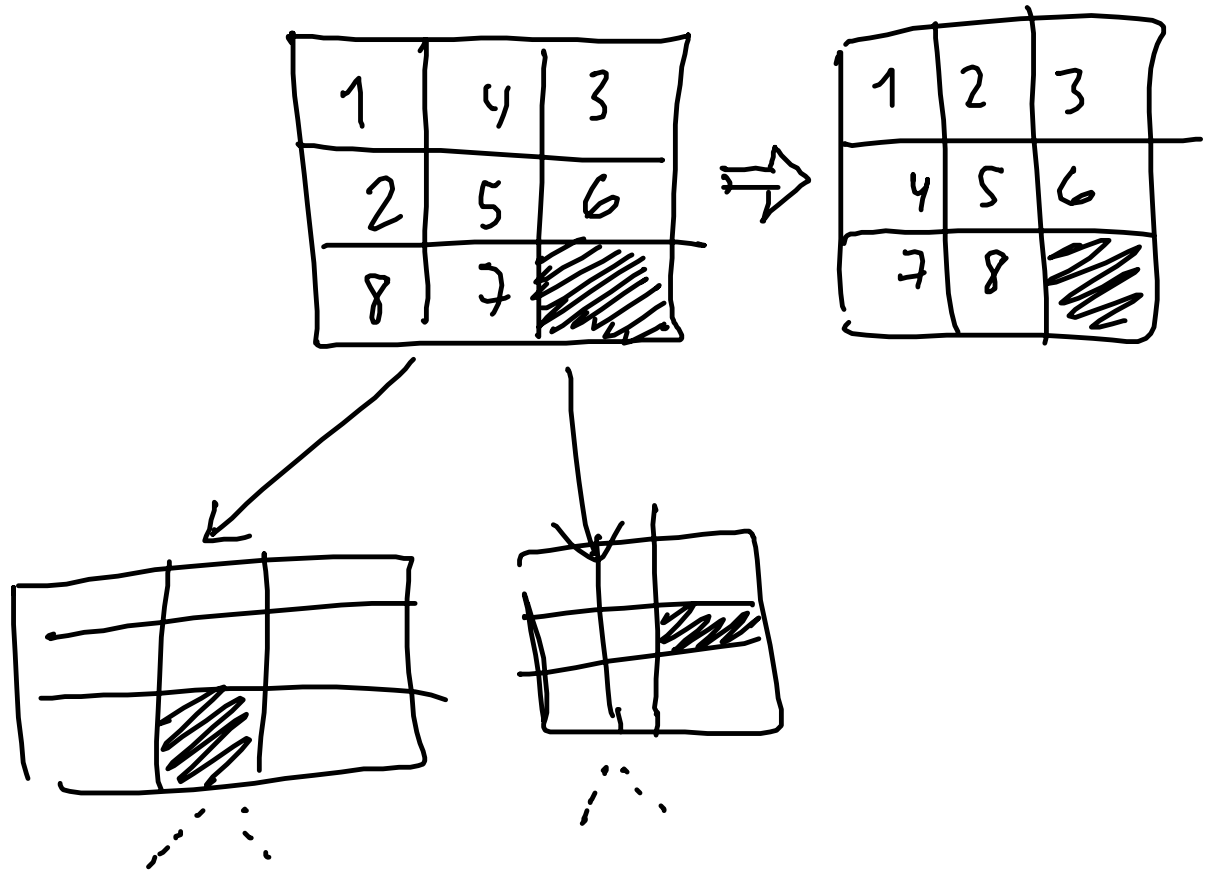
⋮



AND-OR-Graph

```
a :- b, c.
a :- d, e.
b :- f, h.
```

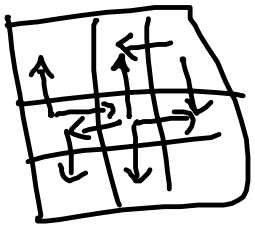




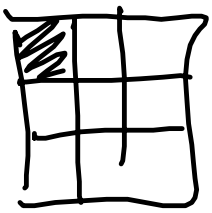
connect(bd(1,4,3,2,5,6,8,7,b),
bd(1,4,3,2,5,6,8,b,7).

alle

[path(bd(.....), bd(1,2,3,4,5,6,7,8,b)



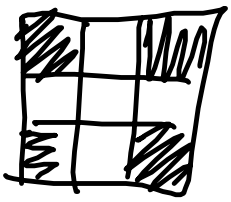
Max 4 Züge



9 Stellen

$$9 \times \frac{8!}{2} \times 4 \approx \underline{2 \cdot 9!}$$

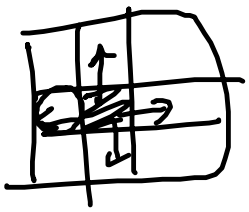
move (bd(X1, X2, X3, X4, X5, X6, X7, X8, b),
bd(X1, X2, X3, X4, X5, X6, X7, b, X8).
move (bd(.....),
bd(X1, X2, X3, X4, X5, b, X7, X8, X6)).



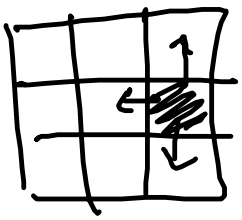
$2 \times 4 = 8$

move (bd(X1, X2, X3, X4, b, X5, X6, X7, X8),
bd(X1, b, X3, X4, X2, X5, X6, X7, X8)).

4 { ⋮

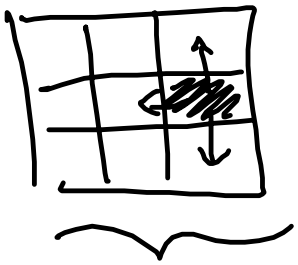


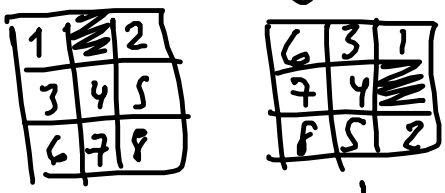
4



4 x 3 = 12

24 Prädikate





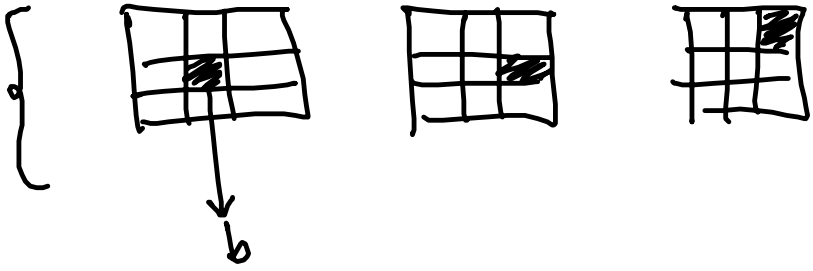
↓

Zug

↓

anti-rotation

Prototypen



24 → 6 Prädikate