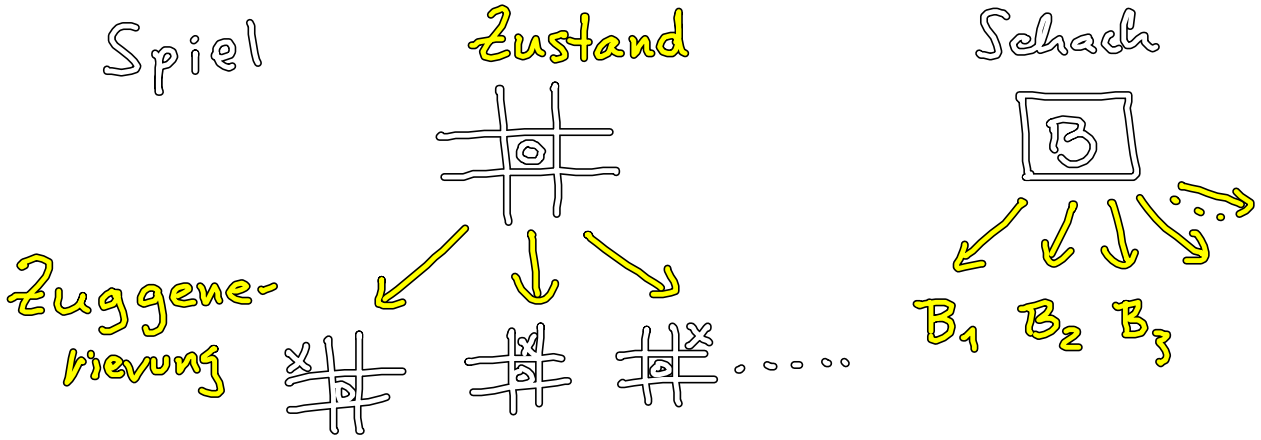
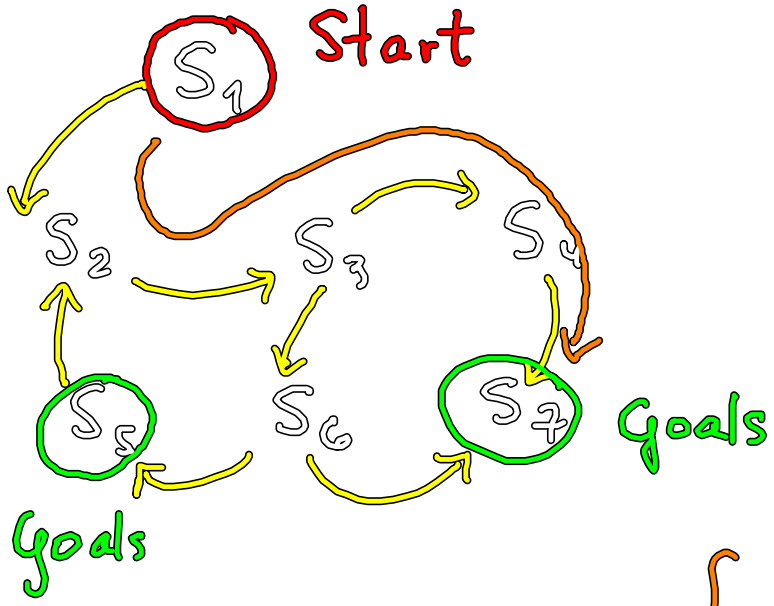
Kap. 3

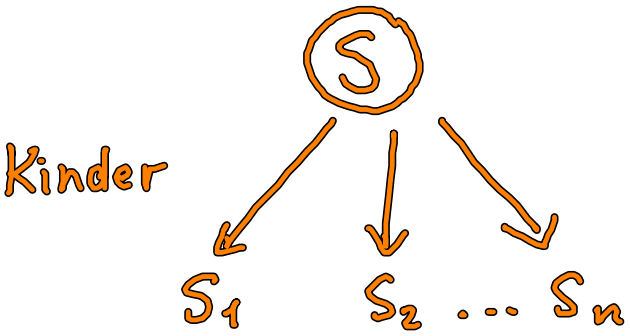
State Space Search

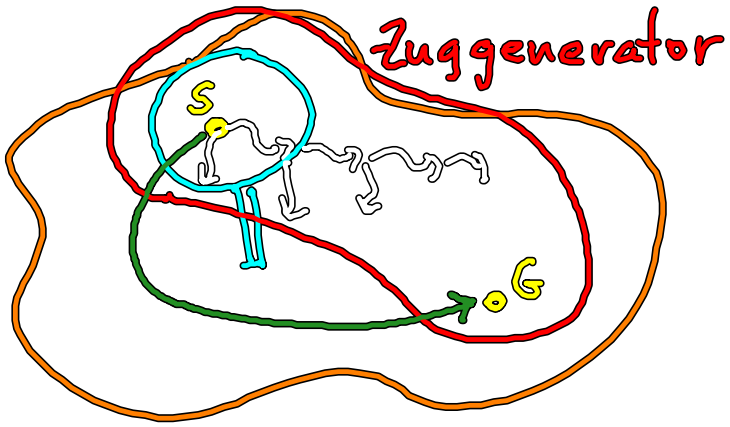




Zustandsraum { Schachbrettler, Tic-Tac-Toe

Graf





Graph

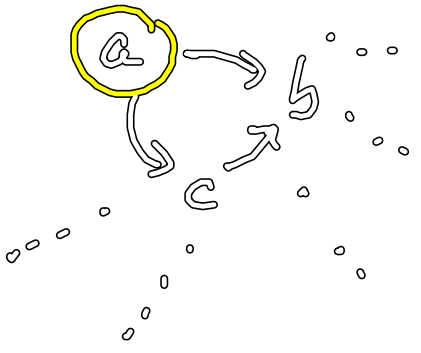
Knoten + Kanten

$G = (V, E)$

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

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

con (a,b).
con (a,c).
con (c,b).
⋮



? con(a, x).

x = b;

x = c

?

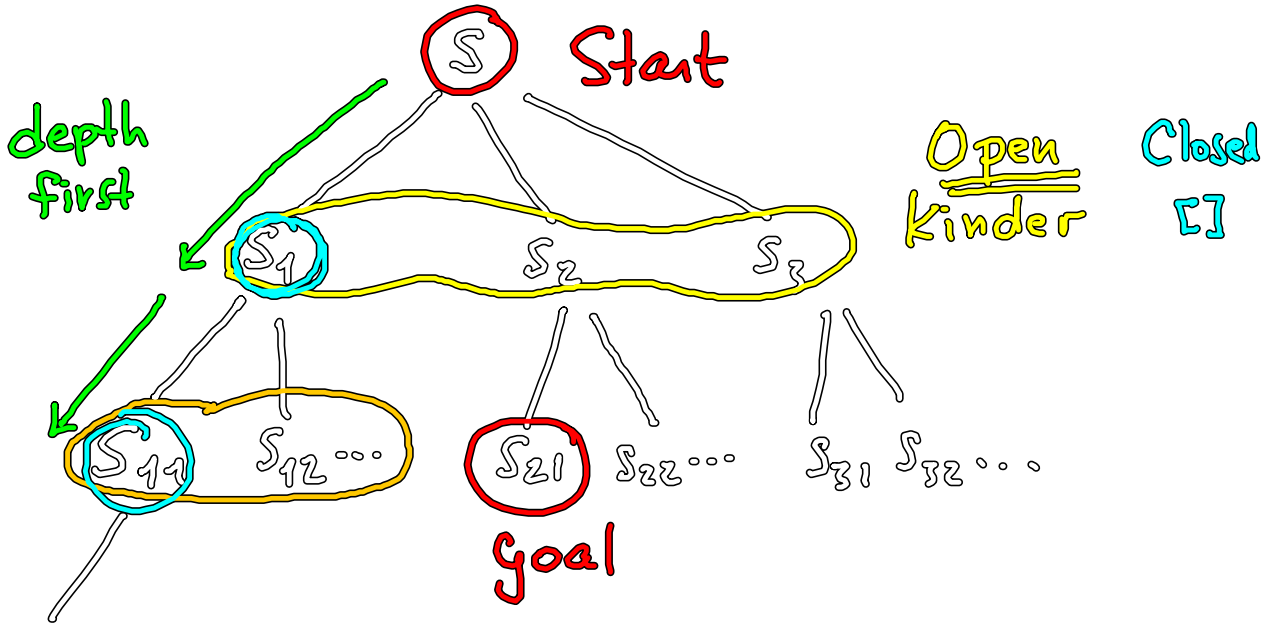
? bagof(x, con(a,x), L).

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.

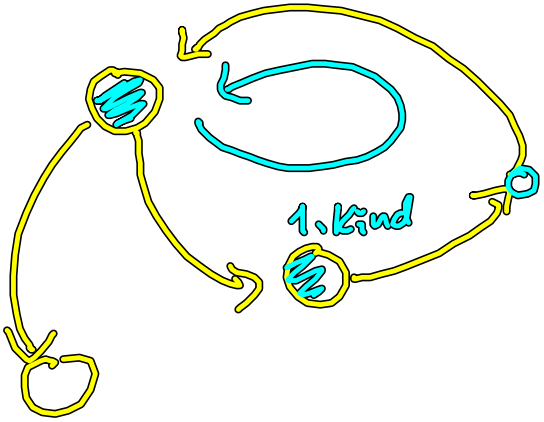
1. { Open [S_1 S_2 S_3] ; Closed [] }

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

Tiefensuche

Kinder [S_{11}, S_{12} ...]

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



In Prolog:

Argumente

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

Aufruf:

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

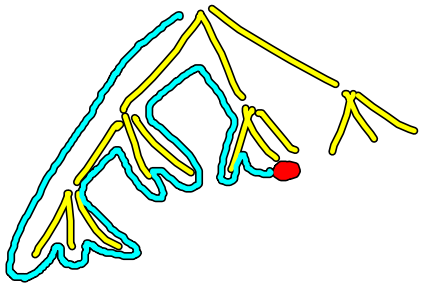
```
...
write (3)   ← hoch problematisch unlogisch
...
```

. ← 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(_, []).
```

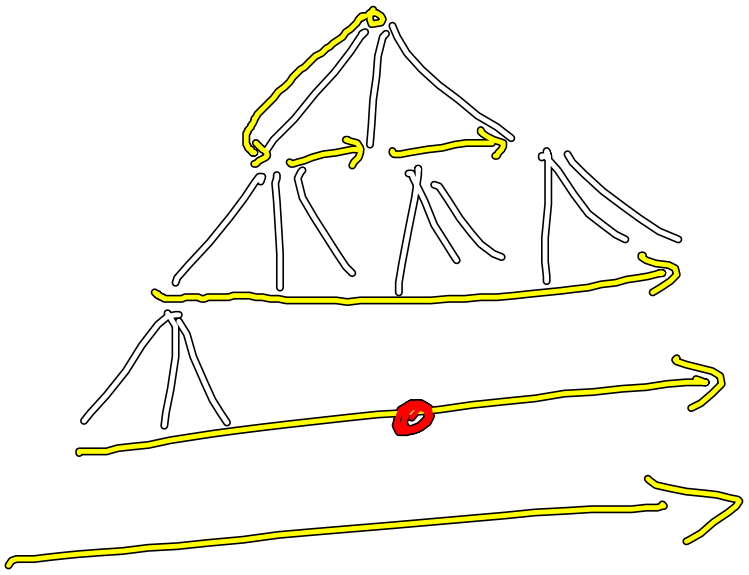
```
appendm([], L, L) :- !.
appendm([X|Y], L, [X|R]) :-
        appendm(Y, L, R).
```

Breitensuche

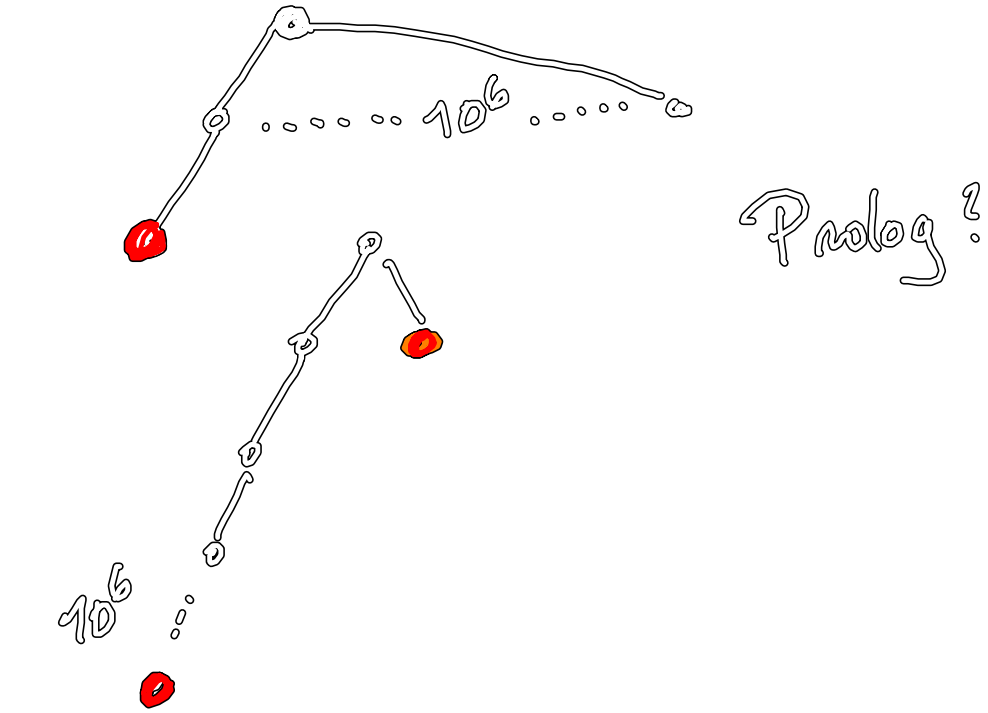
```
breadth(Goals, [X|_], _) :-
        .......
breadth(Goals, [X|RestOpen], Closed) :-
        .....
        breadth(.... )
```

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

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



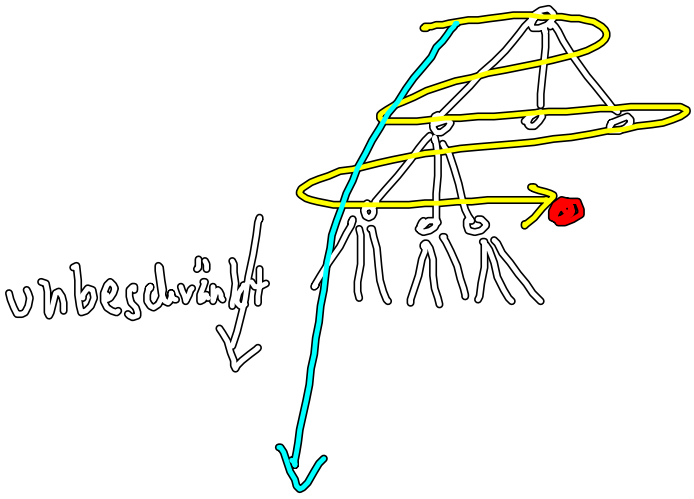
Besser?



Prolog?

1) Prolog ?

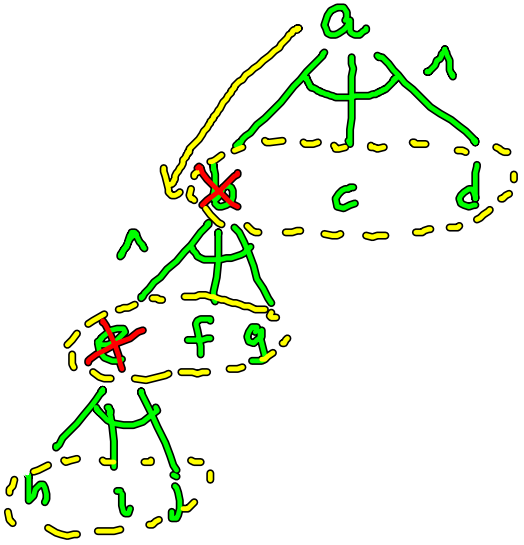
2) Terminiert ?



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 ?
      ⋮
```

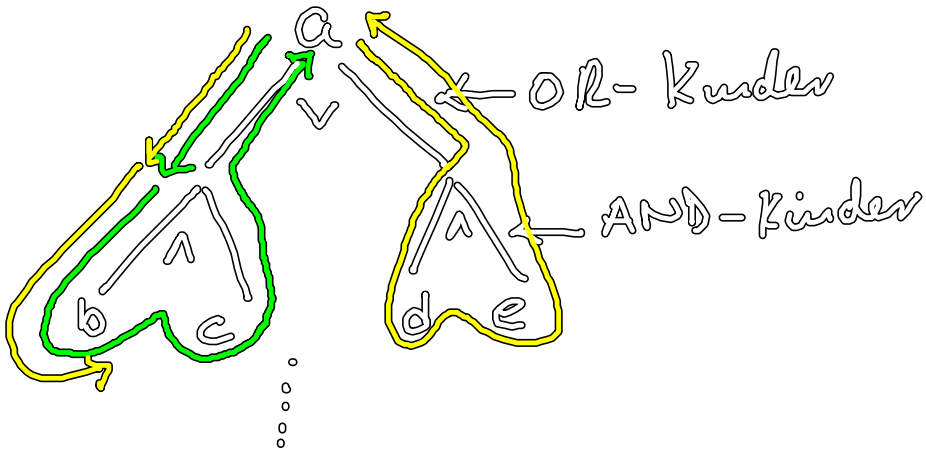


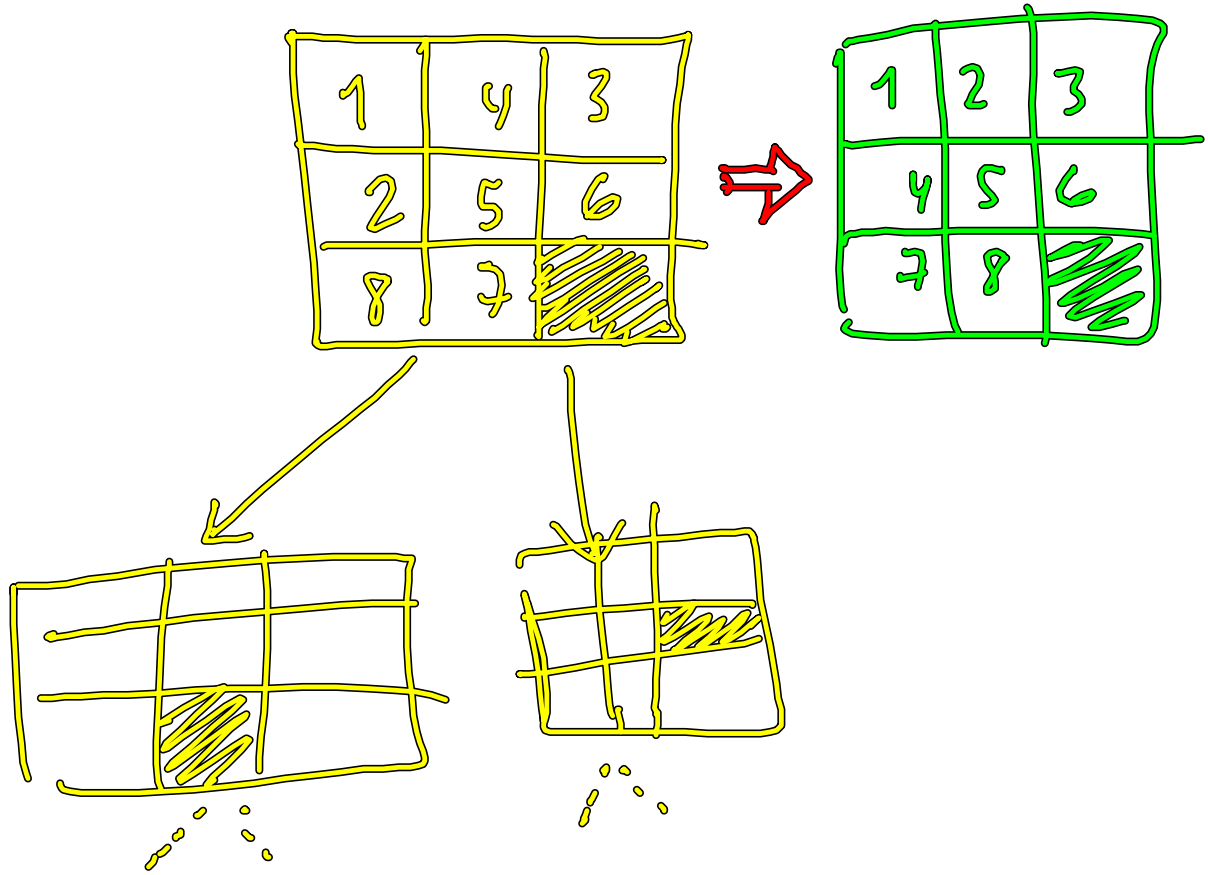
AND-OR-Graph

a :- b, c.

a :- d, e.

b :- f, h.

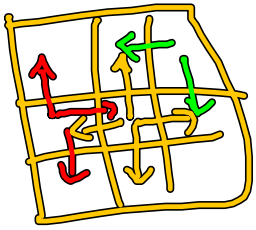




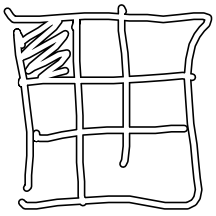
connect(bd(1,4,3,2,5,6,8,7,b),
bd(1,4,3,2,5,6,8,b,7).

alle

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



Max 4 Züge



9 Stellen

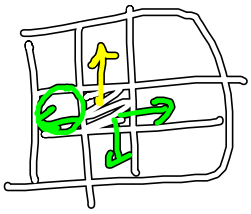
$$9 \times \frac{8!}{2} \times 4 \approx \underline{2 \cdot 9!}$$

$$\begin{cases} \text{move}(bd(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,b), \\ \qquad bd(x_1,x_2,x_3,x_4,x_5,x_6,x_7,b,x_8)). \\ \text{move}(bd(\ldots\ldots), \\ \qquad bd(x_1,x_7,x_3,x_4,x_5,b,x_7,x_8,x_6)). \end{cases}$$

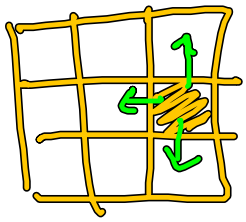
$2 \times 4 = 8$

move(bd(x1, x2, x3, x4, b, x5, x6, x7, x8),
      bd(x1, b, x3, x4, x2, x5, x6, x7, x8)).

4 { ... }

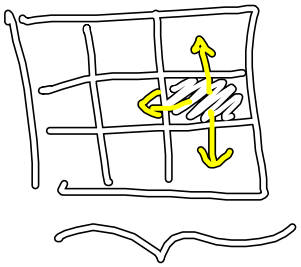


4

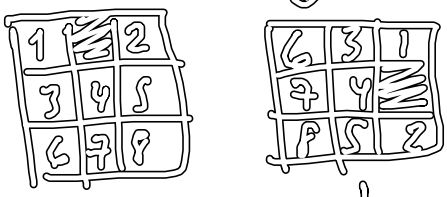


4 x 3 = 12

24 Prädikate



rot 90°



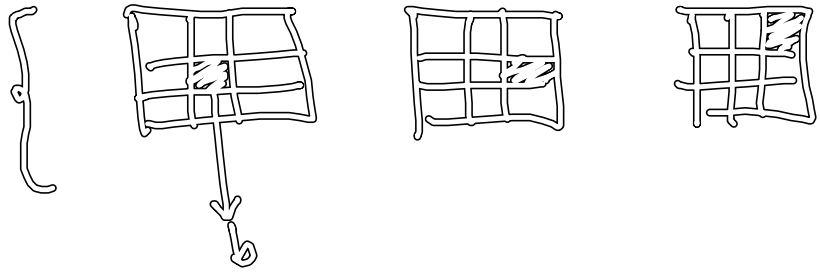
↓

zug

↓

anti-rotation

Prototypen



24 → 6 Prädikate