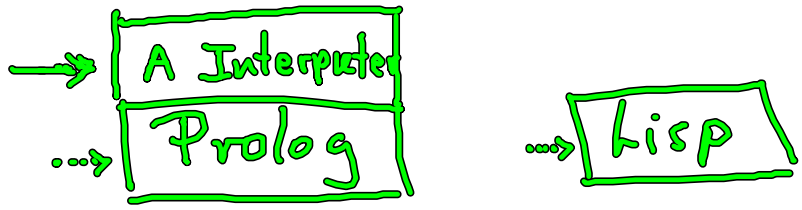
Metainterpreter



Symptome → Diagnose

Schnupfen → (Erkältung, 0,80)

⋮

Prolog in Prolog

Systemprädikate

```
system(call(_)).
system(nl).
system(=(_,_)).
system(==(_,_)).
   ⋮
system(clause(_,_)).
```

Syntax

```
X = a
=(X, a)
X == Y
```

Bsp:

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

```
?- clause(a, X).

X = ';'(b, ';'(c,d))

yes ;

X = ';'(d,e)

yes
?-
```

Prolog Metainterpreter (ohne !)

```
?- solve(a).
   ⋮
```

```
? solve (;(a, b)).
```

```
a :- b, true, c.

solve (true).

solve ((A, B)) :- solve(A), solve(B).

solve ((A; B)) :- solve(A); solve(B).

solve (A) :- system(A), call(A).

solve (A) :- clause(A, B),
             solve(B).
```

```
a :- b, true, c.
b :- true.
c :- true.
------
? solve (a).
  ↳ solve ((b, (true, c)))
    ↳ solve(b), solve((true, c))
      ↳ solve(true), solve((true, c))
```

clause(a, B)

↳ solve((true,c))

↳ solve(true), solve(c)

⋮

yes

?

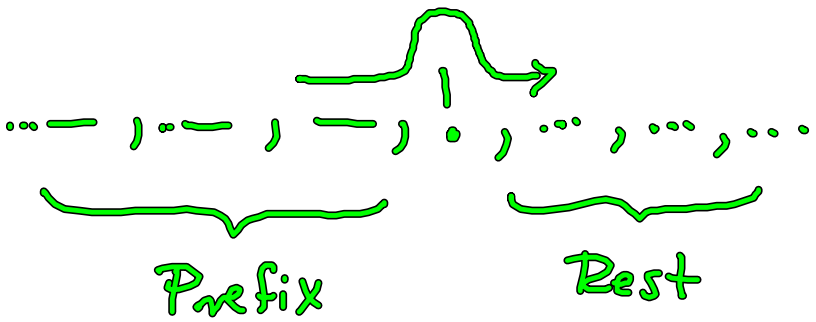
Metainterpreter mit cut

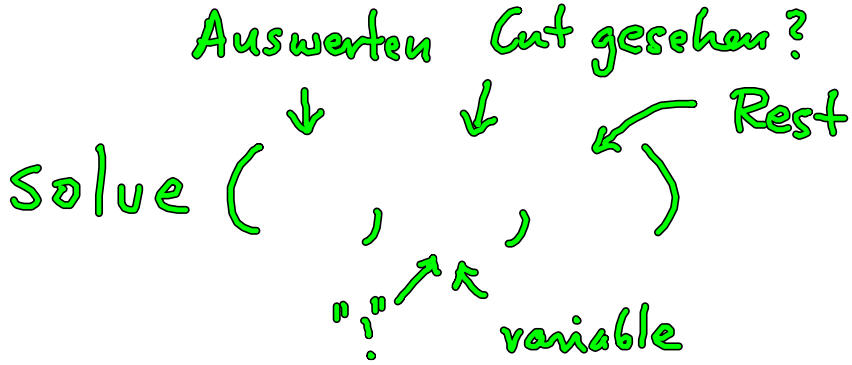
'&'(a,b)

solve('&'(A,B)) :- solve(A), solve(B).

infix('&',...) A & B

Cut






```
solve(X) :- solve(X,C,R),
            (C == !, !, solve(R); true).
```

```
solve(X) :- solve(X,C,R),
            C == !, !, solve(R).
```

CP ↘

```
solve(X) :- solve(X,C,R).
```

```
? solve(solve(solve(solve(...)))).
```

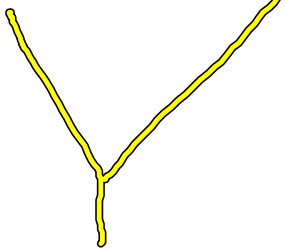
```
solve(two,_,_):-!.
solve(!,!,two):-!.
solve((A,B),X,Y):-!,
    solve(A,X,R),
    (X==!,!,Y=(R,B);
     solve(B,X,Y)).
solve((A;B),X,Y):-!,
    (solve(A,X,Y),
     (X==!,!;two);
     solve(B,X,Y)).
```

A: a, b, !, c, d

B: e, f

A: a, !, c

B: d



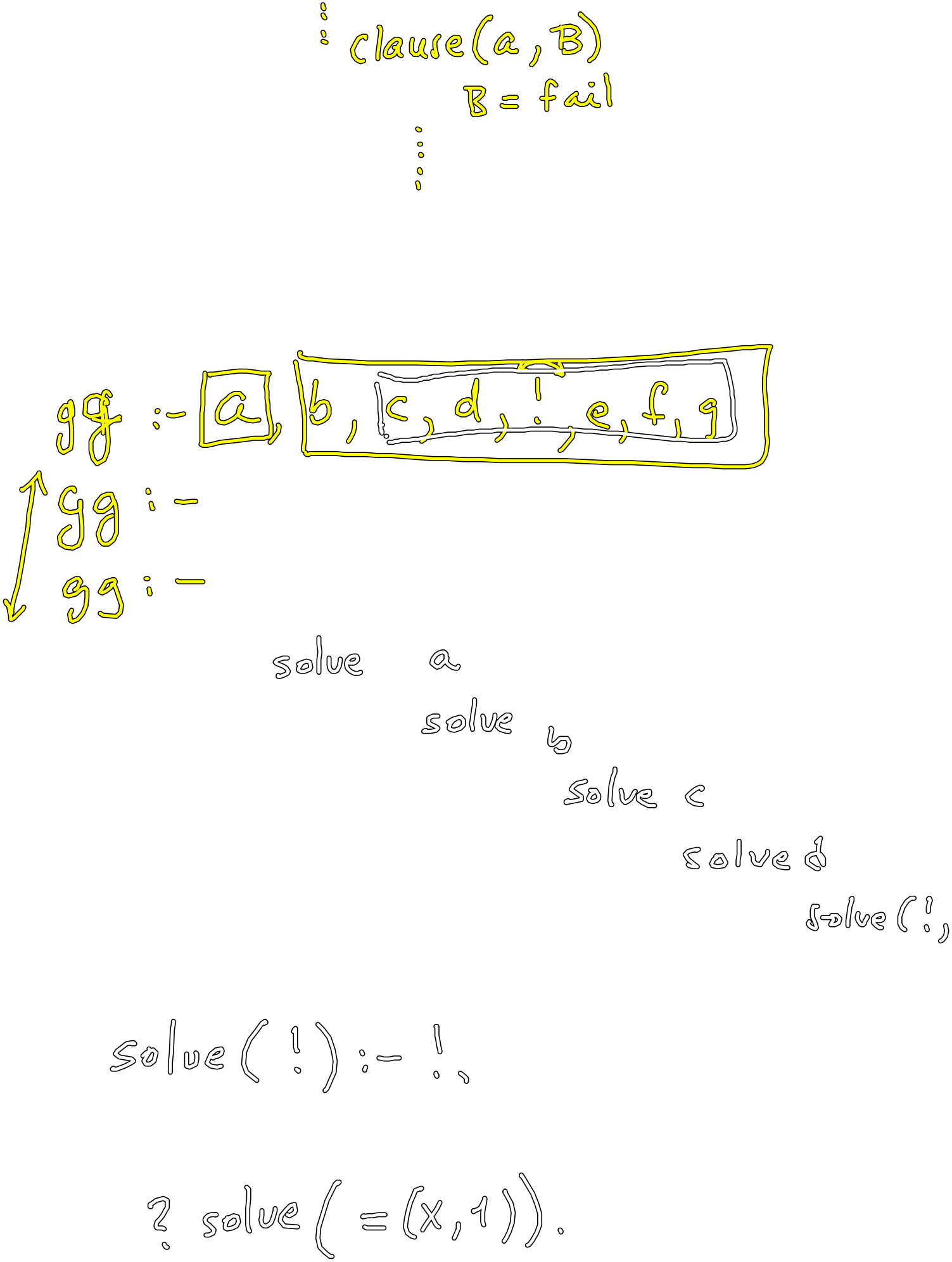
```
solve(A,_,_):- system(A),!,
                call(A).

solve(A,_,_):-
    clause(A,B),
    solve(B,X,R),
    (X==!,!,solve(R); true).
```

⋮

```
a:- fail.
a:- fail,
a:- !,fail,
a:- true.
```

```
system(fail):-!,fail.
```

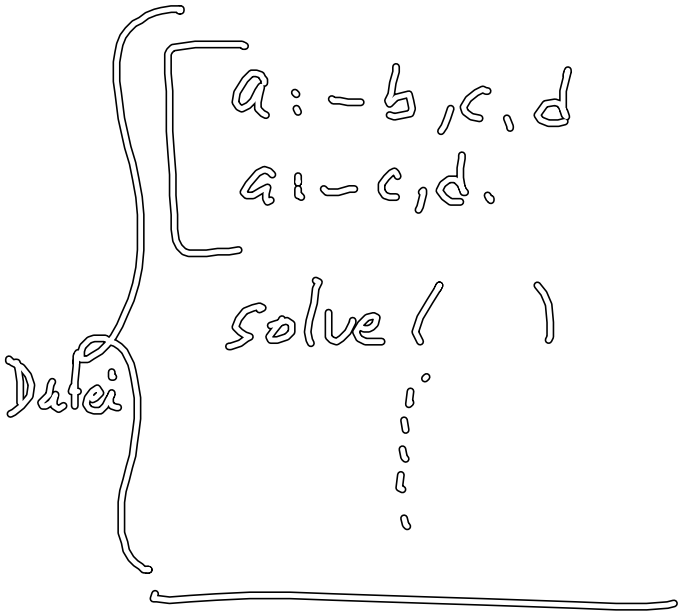
```
? solve(a,X,Y).
.
```


clause(a, B)
B = fail
gg :- a, b, c, d, !, e, f, g
gg :-
gg :-
solve a
solve b
solve c
solve d
solve(!).
solve(!) :- !.
? solve(=(X, 1)).

X = 1

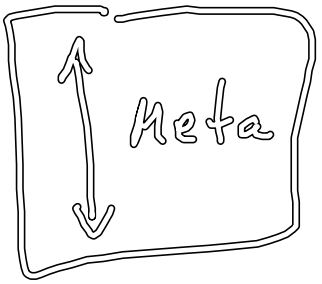
yes

?.



? solve (a).

system (= (_)).



unify (_ , _)

unify (x , y) :-

atom(x), atom(y),

```
X == Y.

unify(X, Y) :-
    atom(X), var(Y),
    Y = X.
    ⋮
Variablen [a, _]
          (a, _)
```