Kap. 2 Prädikatenlogik

↓

Kap. 3 Suchen in Entscheidungsbäumen

Aussagenlogik

Konstante Atomare Aussagen

1	true	true
0	false	fail

Operatoren $\wedge, \vee, \neg$

↑ AND ↑ OR ↖ NEGATION

Prolog

and(1, 0) $1 \wedge 0$

↑

Struktur

and (and (1, 0), 1) $(1 \wedge 0) \wedge 1$

↑ ↑

S_1 S_2

and (or (1, 0), or (and (1, 1), 1))

$(1 \vee 0) \vee ((1 \wedge 1) \vee 1)$

? eval (and (1, 1)).

yes

```
eval ( 1 ) :-  true.
eval ( and ( A, B ) ) :-  eval(A), eval(B).
eval ( or ( A, B ) ) :-  eval(A); eval(B).
eval ( not ( A ) ) :-  not ( eval(A) ).
```

Struktur (not (A))

Prädikat not

Strukturen

```
? eval( and( not(1), 1)).
```

```
f(g(h(x)))
```

```
? eval(0).
  fail.
```

Negation als Failure

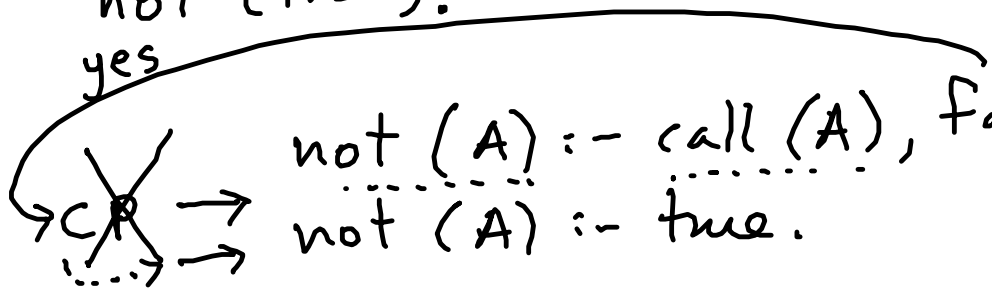
cut

meta prädikat

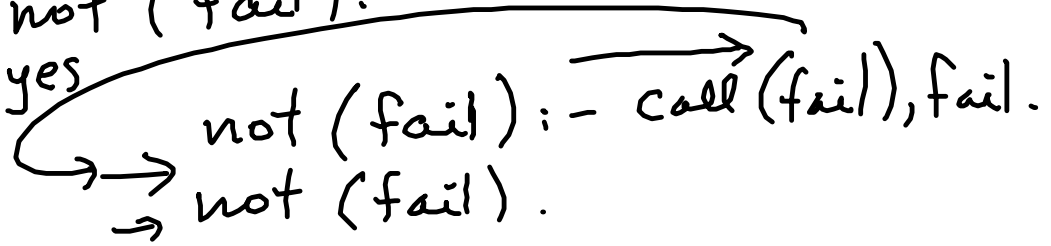
```
not(A) :- call(A), !, fail.
not(A) :- true.
```

! ← entfernt der choice point

```
? not(true).
```




```
? not(fail).
```



? not (true).

fail → ..... ↓

not (true) :- call (true), !, fail.

~~CP → not (true) :- true.~~

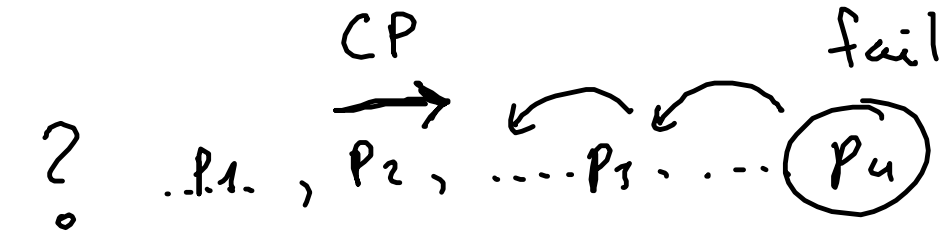
---

eval (1) :- true.

eval (and (A, B)) :- !, eval(A), eval(B).

eval (not(A)) :- !, not (eval (A))

eval (or (A, B)) :- !, eval (A); eval(B).



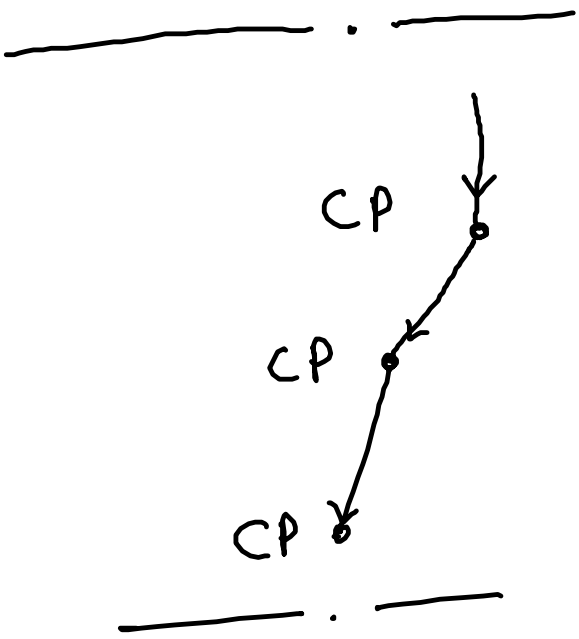
4. Zeile

eval (or (A, B)) :- eval (A).

eval (or (A, B)) :- eval (B).

```
p1(   ) :- !,
p1(   ) :- (a, b); (c, d)
p1(   ) :-
p1(   ) :-
```

a, (b; c), d



eval ← Metainterpeter

Prädikatenlogik = Aussagenlogik
+
Quantoren ∀, ∃
+

Variablen
+
Funktionen

Bsp:

$$\forall x \quad A(x) \wedge B(x)$$

Metainterpreter für PL

forall (X, P)

↑ Var ↑ Aussage

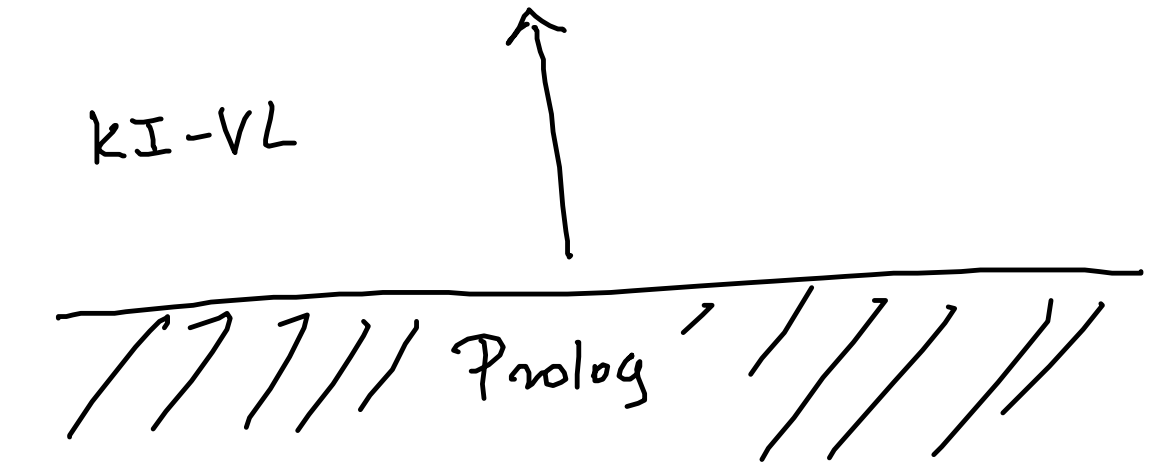
exists (X, P)

Variablen: x(1)
x(2)
x(3)
⋮

father (x(1))
mother (x(2))
⋮

$\exists y$ friend (y, peter)

friend (thomas, peter).



? verify (.....). ← überprüft Syntax

yes

verify (1) :- !, true.

```
verify(0) :- !, true.
verify(exists(x(N), P)) :- !, verify(P).
verify(forall(x(N), P)) :- !, verify(P).
verify(and(A, B)) :- !, verify(A),
                        verify(B).
verify(or(A, B)) :- !, verify(A),
                       verify(B).
verify(not(A)) :- !, verify(A).
```

(Funktionen kommen noch)

```
? verify(and(1, forall(x(1), 1))).
```

KNF ≡ Konjuntive Normalform

$$S \equiv S_1 \wedge S_2 \wedge \ldots \wedge S_n$$

$$S_i \equiv S_i' \vee S_i'' \vee \ldots$$

$$S_i' \equiv S \mid \neg S$$

KNF

$$(a \vee b) \wedge (\neg c \vee d) \wedge e$$

```
knf(A,B) :- negation(A,C),
            reorder(C,B).
```

$$\neg(a \vee b) \equiv (\neg a) \wedge (\neg b)$$

$$a \vee (b \wedge c) \equiv (a \vee b) \wedge (a \vee c)$$

```
reorder(or(and(A,B),C), and(or(A,C),
                            or(B,C))).

reorder(or(and(A,B),C), reorder(A,A2),
                        reorder(B,B2),
                        reorder(C,C2),
                        ...
                        ).

reorder(or(and(A,B),C), and(X,Y)):-!,
                        reorder(or(A,C),X),
                        reorder(or(B,C),Y).

reorder(or(A,and(B,C)), and(X,Y)):-!,
```

```
reorder(or(A,B),X),
reorder(or(A,C),Y).
reorder(and(A,B),and(X,Y)):-!,
        reorder(A,X),
        reorder(B,Y),
( reorder(or(A,B),or(X,Y)):.... ? )
  reorder(A,A).
```

Unifikation

```
? unify(x(1),1).
   yes
```

⋮

Metaprädikate

```
? var(X).
   yes
? var(2).
   fail
```

Wahr wenn Argument eine Variable

```
? s(1,2,3) =.. L.
```

L = [s, 1, 2, 3]

?

(Syntax von Prolog)

Unifikation (Versuch 1)

1) `unify(X,Y) :- var(X), !, true.`

? `unify(T, 1).`

yes

2) `unify(X,Y) :- var(Y), !, true.`

3) `unify(X,Y) :- X == Y, !.`

4) `unify(X,Y) :-`

? `unify(s(X), s(1)).`

`unify(X,Y) :- X =.. L1,`
`Y =.. L2,`
`uniflist(L1,L2).`

? `1 =.. L.`

`L = [1].`