Przykład:

$$\begin{aligned}
&\mathrm{unify}(\{f(x, g(a,b)) = f(g(y,b), x)\}) \\
&= \mathrm{unify}(\{x = g(y,b),\ g(a,b) = x\}) \\
&= \mathrm{unify}(\{g(a,b) = g(y,b)\}) \cup [x/g(y,b)] \\
&= \mathrm{unify}(\{a = y,\ b = b\}) \cup [x/g(y,b)] \\
&= \mathrm{unify}(\{a = y\}) \cup [x/g(y,b)] \\
&= \mathrm{unify}(\{\}) \cup [y/a; x/g(y,b)] \\
&= [y/a; x/g(y,b)]
\end{aligned}$$

czyli $\sigma = [y/a,\ x/g(a,b)]$

Przykład:

$$A \equiv \neg \exists y \forall z\, (p(z,y) \longleftrightarrow \neg \exists x\, (p(z,x) \wedge p(x,z)))$$

$$\begin{aligned}
\neg A &= \exists y \forall z\, (p(z,y) \to \neg \exists x\, (p(z,x) \wedge p(x,z))) \wedge \\
&\qquad ((\neg \exists x\, (p(z,x) \wedge p(x,z)) \to p(z,y)) \\
&= \exists y \forall z\, (\neg p(z,y) \vee \forall x\, (\neg p(z,x) \vee \neg p(x,z))) \wedge \\
&\qquad ((\exists u\, (p(z,u) \wedge p(u,z))) \vee p(z,y)) \\
&= \forall z\, (\neg p(z,a) \vee \forall x\, (\neg p(z,x) \vee \neg p(x,z))) \wedge \\
&\qquad ((p(z, f(z)) \wedge p(f(z), z))) \vee p(z,a))
\end{aligned}$$

$$= \forall z \forall x \, (\neg p(z,a) \vee \neg p(z,x) \vee \neg p(x,z)) \wedge$$
$$(p(z, f(z)) \vee p(z,a)) \wedge$$
$$(p(f(z), z) \vee p(z,a))$$

$$K_{\neg A} = \{ \neg p(z,a) \vee \neg p(z,x) \vee \neg p(x,z),$$
$$p(z, f(z)) \vee p(z,a),$$
$$p(f(z),z) \vee p(z,a) \}$$

$\neg p(z_1,a) \vee \neg p(z_1,x) \vee \neg p(x,z_1)$ $\qquad$ $p(z_2, f(z_2)) \vee p(z_2,a)$

$\sigma(x) = a$

$\sigma(z_2) = z_1$

$\neg p(a,z_1) \vee p(z_1, f(z_1))$ $\qquad$ $p(z_2, f(z_2)) \vee p(z_2,a)$

$\sigma(z_1) = \sigma(z_2) = a$

$p(a, f(a)$

$\neg p(z_1,a) \vee \neg p(z_1,x) \vee \neg p(x,z_1)$ $\qquad$ $p(f(z_3), z_3) \vee p(z_3,a)$

$\sigma(x) = a$

$\sigma(z_3) = z_1$

$\neg p(a,z_1) \vee p(f(z_1), z_1)$ $\qquad$ $p(f(z_3), z_3) \vee p(z_3,a)$

$\sigma(z_1) = \sigma(z_3) = a$

$p(f(a), a)$

$\vdots$

$p(f(a),a)$ $\qquad$ $\neg p(z_1,a) \vee \neg p(z_1,x) \vee \neg p(x,z_1)$

$\sigma(z_1) = f(a)$

$\neg p(f(a),x) \vee \neg p(x,f(a))$ $\qquad$ $p(a,f(a))$

$\sigma(x) = a$

$p(f(a),a)$ $\qquad$ $\neg p(f(a),a)$

$\square$

2.4 Rezolucja w Prologu

w Prologu nie używamy wszystkich formuł w postaci klauzuli:

Def:

(i) Definite Clause

Klauzula, w której dokładnie jeden literał jest pozytywny.

(ii) Definite Goal Clause

$\neg G_1 \vee \ldots\ldots \vee \neg G_m$

(czyli $\neg \exists x_1 \ldots \exists x_k \; G_1 \wedge \ldots \wedge G_m$)

(iii) Horn Clauses

Definite Clauses + Definite Goal Clause

Przykład:

$$\{ x + 0 = x, \; x + s(y) = s(x+y) \}$$
$$\models ss0 + ss0 = ssss0$$

```
add(X,0,X).
add(X,sY,sZ) :- add(X,Y,Z).
```

$\neg$ add(W, ss0, ssss0).

add(X,0,X)

$\neg$ add(X,Y,Z) $\vee$ add(X,sY,sZ)

$\neg add(W, ss0, ssss0)$

$\neg add(X, Y, Z) \vee add(X, sY, sZ)$

$\sigma(x) = W$
$\sigma(y) = s0$
$\sigma(z) = sss0$

$\neg add(W, s0, sss0)$

$\neg add(X, Y, Z) \vee add(X, sY, sZ)$

$\sigma(x) = W$
$\sigma(y) = 0$
$\sigma(z) = ss0$

$\neg add(W, 0, ss0)$

$add(X, 0, X)$

$\sigma(w) = X$
$\sigma(x) = ss0$

$\square$

$\sigma(w) = \underline{\underline{ss0}}$

Obserwacje:

- zaczynamy z celem
- wynik kroku rezolucji używamy w kolejnym kroku
- druga klauzula jest klauzulą ze zbioru startowego – albo przodkiem aktualnego.

Rezolucja SLD

Linear Resolution for Definite Clauses with Selection Function

Twierdzenie

Rezolucja SLD jest $\square$-zupełna dla klauzuli Horn'a, tzn. jeżeli M nie jest spełnialna, istnieje rezolucja SLD, taka że $M \vdash^{*}_{SLD} \square$.

Uwaga: Nie każda rezolucja musi się skończyć z $\square$ – nawet jeżli istnieje jedna taka.

$\to$ strategia wyboru

"drzewo odpowiedzi"

Strategia Prologu

- zastosuj pierwszą klauzulę z góry
- wybieraj cele od lewa

Przykład:

$p(X) :- q(X), r(X)$

$p(3).$

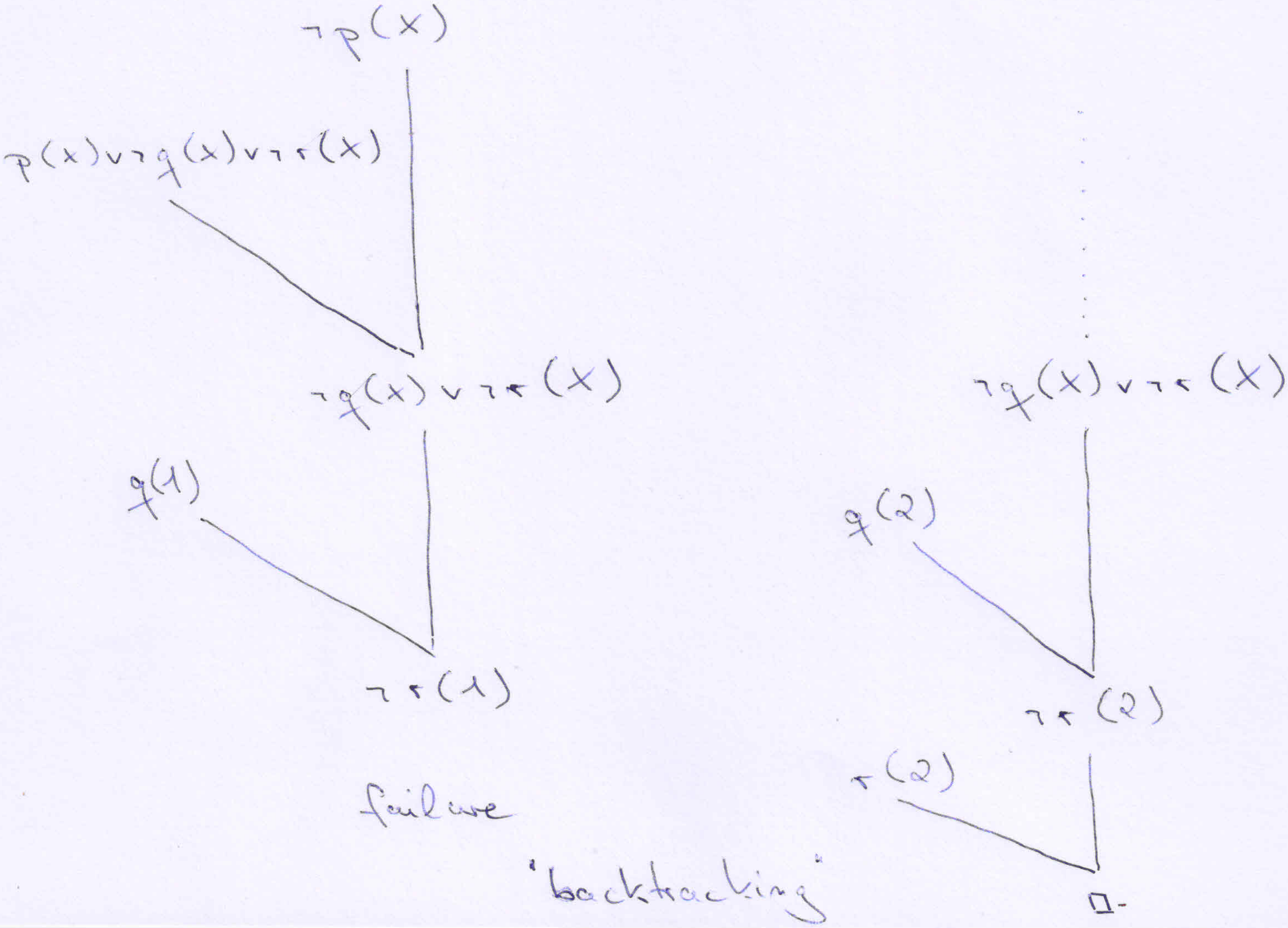
$q(1).$

$q(2).$

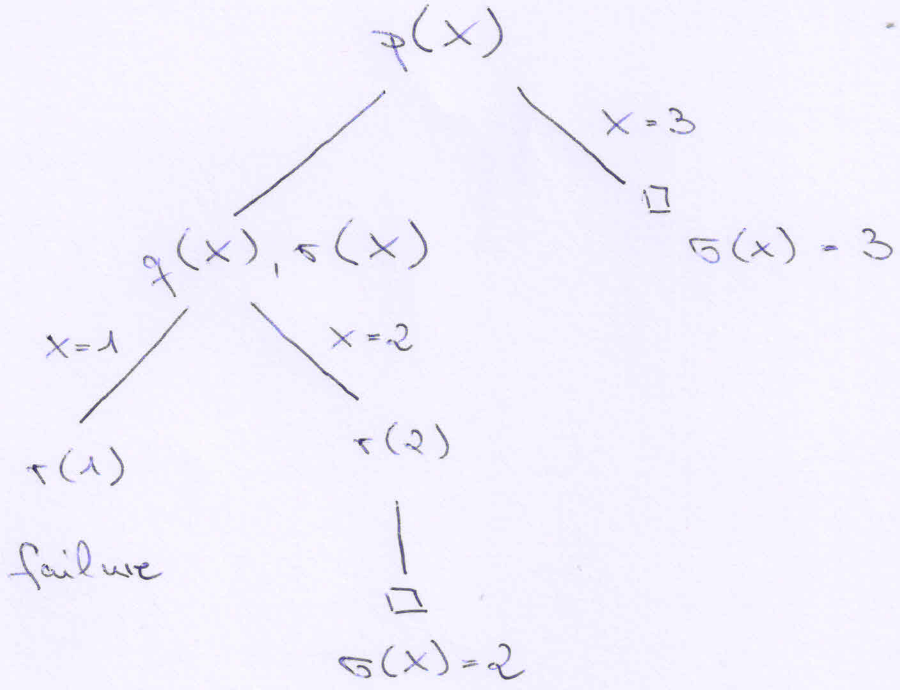
$r(2).$

$r(3).$

$?- p(X)$



Drzewo SLD



czyli

```
?- p(X)
X = 2;
X = 3;
false
```

Przykład:

```
p(X) :- p(X), r(X).
p(3).
r(3).
```

```
?- p(3).
```

oczywiście p(3) jest konsekwencją programu, ale p(3)

p(3), r(3)

p(3), r(3), r(3)

...

tzn. Prolog nie znajduje poprawnej odpowiedzi – nawet nie terminuje!

więc:

```
p(3).
r(3).
p(X) :- p(X), r(X).
```

2.4 Funkcje rozstrzygalne

Funkcje μ-rekursywne:

(i) funkcje bazowe

$$s(n) = n+1$$

$$pr^i_n(x_1, \dots x_n) = x_i$$

$$c^i_n(x_1, \dots x_n) = i$$

(ii) schematy

$$(f \circ \langle g_1, \dots g_k \rangle)(x_1, \dots x_m) = f(g_1(x_1, \dots x_m), \dots g_k(x_1, \dots x_m))$$

$$R_{g,f}(x_1, \dots x_m, y) = \begin{cases} g(x_1, \dots x_m) & ; y = 0 \\ f(x_1, \dots x_m, y, R_{g,f}(x_1, \dots x_m, y-1)) & ; y > 0 \end{cases}$$

$$\mu f(x_1, \dots x_m) = \begin{cases} \min M_{f, x_1, \dots x_m} & ; M_{f, x_1, \dots x_m} \neq \emptyset \\ \uparrow & ; M_{f, x_1, \dots x_m} = \emptyset \end{cases}$$

dla

$$M_{f, x_1, \dots x_m} = \{ y \geq 0 \mid f(x_1, \dots x_m, y) = 0 \wedge \forall k < y : f(x_1, \dots x_m, k) \downarrow \}$$

<u>Def:</u> Funkcje μ-rekursywne $\mathcal{R}_p$

(i) $\{ s, pr^i_m, c^i_m \} \subseteq \mathcal{R}_p$

(ii) $f, g_1, \dots g_k \in \mathcal{R}_p \rightsquigarrow f \circ \langle g_1, \dots g_k \rangle \in \mathcal{R}_p$

(iii) $f, g \in \mathcal{R}_p \rightsquigarrow R_{g,f} \in \mathcal{R}_p$

(iv) $f \in \mathcal{R}_p \rightsquigarrow \mu f \in \mathcal{R}_p$

(v) $\mathcal{R}_p$ jest najmnieszą klasą funkcji, która spełni (i) - (iv).

Przykład

$g(x) := x$, czyli $g = pr^1_1$

$f(x,y,z) := z+1$, czyli $f = s \circ \langle \pi^3_3 \rangle$

więc $R_{g,f} \in \mathcal{R}_P$

$R_{g,f}(x,0) = g(x) = x = x+0$

dla $y > 0$:

$$
\begin{aligned}
R_{g,f}(x,y) &= f(x,y,R_{g,f}(x,y-1)) \\
&= R_{g,f}(x,y-1) + 1 \\
&\overset{z.i.}{=} (x + (y-1)) + 1 \\
&= x+y
\end{aligned}
$$

wynik: "$x+y$" $\in \mathcal{R}_P$

Twierdzenie

$f \in \mathcal{R}_P$ wtw. istnieje maszyna Turinga, która obliczy f.

Def:

Język programowania P jest Turing-zupełny, jeżeli w P można programować wszystkie funkcje $f \in \mathcal{R}_P$.

w Prologu:

```
s(X,Z) :- Z is X+1.

pr^i_m(X1,....Xm,Xi).

c^i_m(X1,....Xm,i).

k(X1,....Xm,Z) :- g1(X1,....Xm,Z1),
                  :
                  gk(X1,...Xm,Zk),
                  f(Z1,...Zk,Z).

r(X1,...Xm,0,Z) :- g(X1,...Xm,Z).
r(X1,...Xm,Y,Z) :- Y > 0, Y1 is Y-1,
                   r(X1,...Xm,Y1,Z1),
                   f(X1,...Xm,Y,Z1,Z).

mf(X1,...Xm,Z) :- h(X1,....Xm,0,Z).
    h(X1,...Xm,Z,Z) :- f(X1,...Xm,Z,0).
    h(X1,...Xm,Y,Z) :-
             Y1 is Y+1,
             h(X1,...Xm,Y1,Z).
```

wynik: Prolog jest Turing-zupełny.