Reguły wnioskowania

- reguła odrywania (modus ponens) $\frac{F,\, F \rightarrow G}{G}$
- reguła odrywania dla równoważności
- reguła eliminacji kwantyfikatora ogólnego $\frac{\forall x\, F}{F\{x/t\}}$

Reguła rezolucji (J.A.Robinson, 1965)

– oryginalna postać:

$$\frac{\neg\alpha \vee \beta, \alpha \vee \gamma}{\beta \vee \gamma}$$

– szczególne przypadki:

$$\frac{\neg\alpha \vee \beta, \alpha}{\beta} \qquad \frac{\neg\alpha, \alpha}{\Box}$$

SLD-rezolucja

(metoda dowodzenia twierdzeń dla programów w logice)

- mechanizm zastępowania (wyprowadzania)
- mechanizm uzgadniania

SLD – **S**election rule-driven **L**inear resolution for **D**efinite clauses

SLD-rezolucja — mechanizm wyprowadzania

Niech P – program w logice, wszystkie klauzule ustalone

$G = \leftarrow A_1, \ldots, A_n$ – klauzula negatywna ($n \geq 1$)
(ustalona)

$C = A_i \leftarrow B_1, \ldots, B_k$ – klauzula z P
($1 \leq i \leq n,\ k \geq 0$)

Wówczas:

$$G' = \leftarrow A_1, \ldots, A_{i-1},\ B_1, \ldots, B_k,\ A_{i+1}, \ldots, A_n$$

jest wynikiem zastąpienia A_i z G przez $B_1, \ldots, B_k$.

G' — **rezolwenta** G i C (klauzula wyprowadzona z G i C)

A_i — **atom wybrany** (z klauzuli G)

wywód — maksymalny (skończony lub nie) ciąg rezolwent
$G_0 = G, G_1, G_2, \ldots$

dowód — wywód, którego ostatnia klauzula jest pusta
$G_0 = G, G_1, G_2, \ldots, G_l = \Box$

z założenia, że G zachodzi dla P wynika sprzeczność, a zatem uzyskaliśmy dowód negacji formuły G, czyli dowód $A_1 \wedge \ldots \wedge A_n$

SLD-rezolucja

Niech P — program w logice

$$G = \leftarrow A_1, \ldots, A_n \text{ – klauzula negatywna}$$

$$C = A \leftarrow B_1, \ldots, B_k \text{ – klauzula P (wariant)}$$

jeśli dla pewnego i $(1 \leq i \leq n)$, A_i oraz A są uzgadnialne, przy użyciu mgu θ, to klauzula postaci:

$$G' = \leftarrow (A_1, \ldots, A_{i-1}, B_1, \ldots, B_k, A_{i+1}, \ldots, A_n)\theta$$

jest **rezolwentą** G i C przy użyciu podstawienia θ

Sposób otrzymywania rezolwenty:

- wybór atomu A_i
- wybór klauzuli programu
- przemianowanie zmiennych w klauzuli programu
- próba uzgodnienia A_i z A (nagłówkiem wybranej klauzuli)
- zastąpienie A_i w G przez treść wybranej klauzuli, zastosowanie do powstałej klauzuli podstawienia uzgadniającego

Przemianowywanie zmiennych w wybranych klauzulach programu

P: p(f(x)) $\leftarrow$ (czyli $\forall x\ p(f(x))$)

$\leftarrow p(x)$ (zapytanie: $\exists x\ p(x)$)

SLD-wywód dla $P \cup \{G\}$

— ciąg klauzul negatywnych $G_0, G_1, \ldots$, gdzie $G_0 = G$
— ciąg $C_1, C_2, \ldots$ wariantów klauzul programu P
— ciąg podstawień mgu $\theta_1, \theta_2, \ldots$

takich, że dla każdego $i = 0, 1, \ldots$

— G_{i+1} jest rezolwentą G_i oraz C_{i+1} przy użyciu θ_{i+1}
— C_i nie zawiera zmiennych występujących w $G_0, C_1, \ldots, C_{i-1}$, $\theta_1, \ldots, \theta_{i-1}$

SLD-dowód

skończony SLD-wywód, którego ostatnia rezolwenta G_n (dla pewnego n, $n \geq 1$) jest klauzulą pustą

n – długość dowodu

postać reguły wnioskowania:

$$\frac{\forall\neg(A_1 \wedge \ldots \wedge A_{i-1} \wedge A_i \wedge A_{i+1} \wedge \ldots A_n), \forall(A \leftarrow B_1 \wedge \ldots \wedge B_k)}{\forall\neg(A_1 \wedge \ldots A_{i-1} \wedge B_1 \wedge \ldots \wedge B_k \wedge A_{i+1} \wedge \ldots A_n)\theta}$$

gdzie

θ – najbardziej ogólne podstawienie uzgadniające dla A_i oraz A

odpowiedź obliczona na $P \cup \{G\}$:

podstawienie θ uzyskane ze złożenia podstawień $\theta_1, \ldots, \theta_n$ z SLD-dowodu $P \cup \{G\}$ ograniczone do zmiennych występujących w klauzuli G ($\theta = \theta_1 \ldots \theta_n | Var(G)$)

funkcja wyboru (reguła obliczeniowa) (ang. selection rule)

funkcja, która dla początku SLD-wywodu $G_0, G_1, \ldots, G_k$, gdzie $G_k = \leftarrow A_1, \ldots, A_l$ przyporządkowuje atom A_i $(1 \leq i \leq l)$

R – funkcja wyboru
SLD-wywód $P \cup \{G\}$ uzyskany przy użyciu reguły R – jeśli do wyboru wszystkich wybranych atomów użyto funkcji R

Właściwości SLD-dowodów

1. Niezależność od wyboru nazw zmiennych oraz wyboru mgu

Lemat. Niech θ i η — odpowiedzi obliczone dla P $\cup$ $\{\leftarrow Q\}$ uzyskane w podobnych SLD-dowodach (w każdym kroku obu dowodów: wybierany atom z tej samej pozycji, użyta ta sama klauzula programu). Wówczas Qθ i Qη są swoimi wariantami.

2. SLD-dowody dla mniej/bardziej ogólnych zapytań

Lemat. Dla każdego SLD-dowodu dla P $\cup$ $\{\leftarrow Q\eta\}$ z odpowiedzią obliczoną θ istnieje SLD-dowód dla P $\cup$ $\{\leftarrow Q\}$ z odpowiedzią obliczoną θ' taki, że:
— oba dowody mają jednakową długość
— wybór atomów z tych samych pozycji
— wybór tych samych klauzul programu
— Q$\eta\theta$ jest ukonkretnieniem Qθ'

3. Niezależność od funkcji wyboru

Lemat. Dla każdego SLD-dowodu dla P $\cup$ {G} oraz dowolnej funkcji wyboru R istnieje SLD-dowód dla P $\cup$ {G} uzyskany przy użyciu R o tej samej długości i identycznej odpowiedzi obliczonej.

Przykład

```
knows_logic( x ) <-
     good_student( x ),
     teacher( y, x ),
     logician( y )
good_student( tom ) <-
logician( peter ) <-
teacher( peter, tom ) <-
```

zapytanie: $\leftarrow$ knows_logic(z) (czyli: kto zna logikę?)

G_0: $\leftarrow$ knows_logic(z)

C_1: knows_logic(x_1) $\leftarrow$ good_student(x_1),
teacher(y_1, x_1), logician(y_1)

$\theta_1 = \{ z/x_1 \}$

G_1: $\leftarrow$ good_student(x_1), teacher(y_1, x_1), logician(y_1)

(zał.: funkcja wyboru – wybór skrajnie lewego atomu)

C_2: good_student(tom) $\leftarrow$

$\theta_2 = \{ x_1/tom \}$

G_2: $\leftarrow$ teacher(y_1, tom), logician(y_1)

C_3: teacher(peter, tom) $\leftarrow$

$\theta_3 = \{ y_1/peter \}$

G_3: $\leftarrow$ logician(peter)

C_4: logician(peter) $\leftarrow$

$\theta_4 = \varepsilon$

G_4: $\square$

obliczone podstawienie:

$$\begin{aligned}\theta_1\theta_2\theta_3\theta_4 = {} & \{\mathrm{z/x_1}\}\ \{\mathrm{x_1/tom}\}\ \{\mathrm{y_1/peter}\}\ \varepsilon = \\ & \{\mathrm{z/tom,\ x_1/tom,\ y_1/peter}\}\end{aligned}$$

SLD-dowód długości 4

odpowiedź obliczona:

$$\theta = \{\ \mathrm{z/tom}\ \}$$

$\square$

Strategie poszukiwania dowodów

przestrzeń poszukiwań tworzy pewnego rodzaju drzewo

P – program, G – klauzula negatywna, R – reguła obliczeniowa

SLD-drzewo dla P $\cup$ {G} — drzewo spełniające warunki:

- korzeniem drzewa jest klauzula G
- każdy węzeł drzewa jest rezolwentą G_i
- węzeł G_{i+1} jest dzieckiem węzła G_i wiw gdy istnieje klauzula C_{i+1}, będąca wariantem pewnej klauzuli programu P taka, że G_{i+1} można wyprowadzić z G_i oraz C_{i+1} przy użyciu reguły obliczeniowej R
- drzewo jest etykietowane klauzulami C_i

każda gałąź SLD-drzewa reprezentuje SLD-wywód dla $P \cup G$

SLD-dowód: skończona gałąź SLD-drzewa zakończona klauzulą pustą

Przykład

program P:

1. $p(x, z) \leftarrow q(x, y), p(y, z)$
2. $p(x, x) \leftarrow$
3. $q(a, b) \leftarrow$

SLD-drzewo dla $P \cup \{\leftarrow p(x, b)\}$

reguła obliczeniowa: wybór lewego atomu

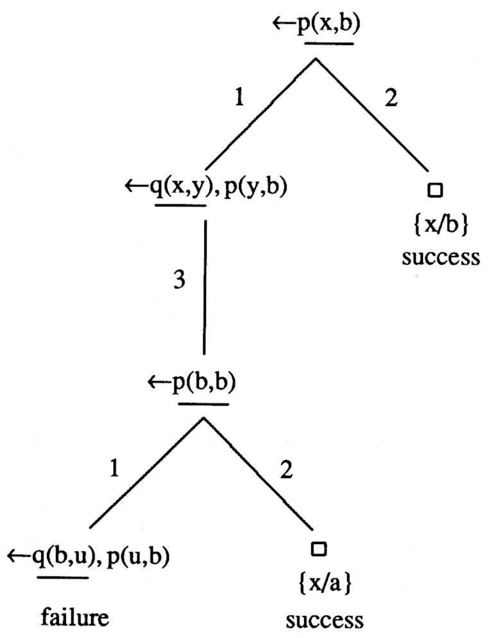

Fig. 2. A finite SLD-tree

(Rysunek pochodzi z książki Johna Lloyda „Foundations of Logic Programming".)

program P:

1. $p(x, z) \leftarrow q(x, y), p(y, z)$
2. $p(x, x) \leftarrow$
3. $q(a, b) \leftarrow$

SLD-drzewo dla $P \cup \{\leftarrow p(x, b)\}$

reguła obliczeniowa: wybór prawego atomu

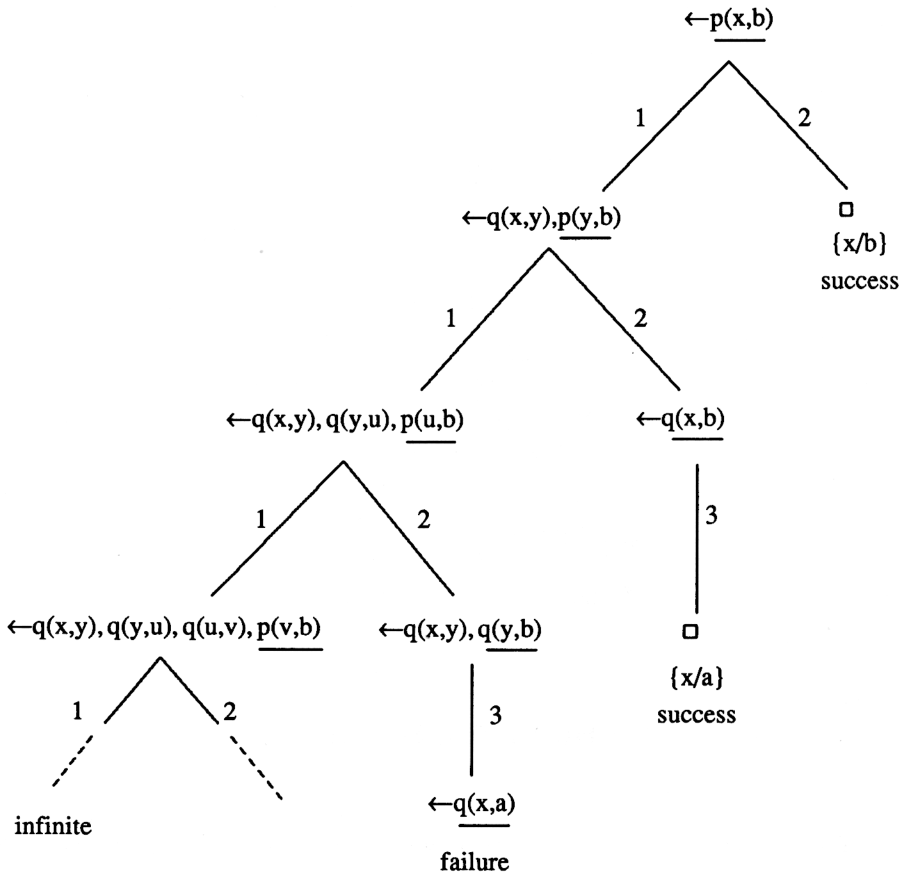

Fig. 3. An infinite SLD-tree

(Rysunek pochodzi z książki Johna Lloyda „Foundations of Logic Programming”.)