Fundamentele limbajelor de programare

Programare Logică. Rezoluție.

Traian Florin Șerbănuță și Andrei Sipoș

Facultatea de Matematică și Informatică, DL Info

Anul II, Semestrul II, 2024/2025

Program în Prolog = mulțime de predicate

Un exemplu de program în Prolog din cursul trecut:

```
father(peter,meg).
father(peter,stewie).

mother(lois,meg).
mother(lois,stewie).

griffin(peter).
griffin(lois).

griffin(X) :- father(Y,X),
griffin(Y).
```

Predicate:
father/2
mother/2
griffin/1

Sintaxa Prolog

- Sintaxa Prolog nu face diferență între simboluri de operații și simboluri de predicate!

- Dar este important când ne uităm la teoria corespunzătoare programului în logică să facem această distincție.

- În sintaxa Prolog
  - termenii compuși sunt predicate: `father(peter, meg)`
  - operatorii sunt funcții: `+`, `*`, `mod`

Exerciţiu

Fie alfabetul definit prin $\mathbf{R} = \{<\}$, $\mathbf{F} = \{s, +, 0\}$ și
$r(0) = 0$, $r(s) = 1$, $r(+) = r(<) = 2$.
Dați exemple de 3 termeni, 3 formule atomice și 3 enunțuri.

Exercițiu

Fie alfabetul definit prin $\mathbf{R} = \{<\}$, $\mathbf{F} = \{s, +, 0\}$ și
$r(0) = 0$, $r(s) = 1$, $r(+) = r(<) = 2$.
Dați exemple de 3 termeni, 3 formule atomice și 3 enunțuri.

Exemple de termeni:

0, x, $s(0)$, $s(s(0))$, $s(x)$, $s(s(x))$, $\ldots$,
$+(0,0)$, $+(s(s(0)), +(0, s(0)))$, $+(x, s(0))$, $+(x, s(x))$, $\ldots$,

Exemple de formule atomice:

$<(0,0)$, $<(x,0)$, $<(s(s(x)), s(0))$, $\ldots$

Exemple de enunțuri:

$\forall x \forall y \; <(x, +(x,y))$, $\forall x \; <(x, s(x))$

Clauze

- O clauză este o disjuncție de literali.
- Dacă $L_1, \ldots, L_n$ sunt literali atunci putem reprezenta clauza $L_1 \vee \ldots \vee L_n$ ca mulțimea $\{L_1, \ldots, L_n\}$

  clauză = mulțime de literali

- O clauză C este trivială dacă conține un literal și complementul lui.
- Când $n = 0$ obținem clauza vidă, care se notează $\square$.

Clauze Horn

Clauza Horn = clauză program definită sau clauză scop

Avem echivalența

$$\forall(\neg Q_1 \vee \ldots \vee \neg Q_n) \equiv \neg\exists(Q_1 \wedge \ldots \wedge Q_n)$$

Negația unei "întrebări" în PROLOG este clauză scop.
Limbajul PROLOG are la bază logica clauzelor Horn.

Un exemplu

Fie următoarele clauze definite:

$father(jon, ken).$
$father(ken, liz).$
$father(X, Y) \to ancestor(X, Y)$
$daugther(X, Y) \to ancestor(Y, X)$
$ancestor(X, Y) \wedge ancestor(Y, Z) \to ancestor(X, Z)$

Putem pune întrebările:

- $ancestor(jon, liz)$?
- $ancestor(ken, Z)$?
  (există Z astfel încât $ancestor(ken, Z)$)

Sistem de deducție pentru logica clauzelor Horn

Pentru un program logic definit KB avem Regula de deducție *backchain*:

$$\frac{\theta(Q_1) \quad \theta(Q_2) \quad \ldots \quad \theta(Q_n)}{\theta(Q)} \quad \text{unde } Q_1 \wedge Q_2 \wedge \ldots \wedge Q_n \to P \in KB$$
$$\text{iar } \theta \text{ cmgu pentru } Q \text{ și } P.$$

Regula *backchain* conduce la un sistem de deducție complet:

Pentru o mulțime de clauze KB și o țintă Q,
dacă Q are soluție,
atunci există o derivare a lui Q folosind regula *backchain*.

Cum răspundem la întrebări

Pentru o țintă Q, trebuie să găsim o clauză din KB

$$Q_1 \wedge \ldots \wedge Q_n \rightarrow P,$$

și un unificator θ pentru Q și P.
În continuare vom verifica $\theta(Q_1), \ldots, \theta(Q_n)$.

Exemplu. Pentru ținta

$$ancestor(ken, Z),$$

putem folosi clauza

$$father(X, Y) \rightarrow ancestor(X, Y)$$

cu unificatorul

$$\{X \mapsto ken, Y \mapsto Z\}$$

pentru a obține o nouă țintă

$$father(ken, Z).$$

Sistem de deducție

Regula backchain

$$\frac{\theta(Q_1) \quad \theta(Q_2) \quad \ldots \quad \theta(Q_n)}{\theta(Q)} \quad \text{unde } Q_1 \wedge Q_2 \wedge \ldots \wedge Q_n \to P \in KB$$

iar θ cmgu pentru Q și P.

Exemplu. Presupunem că în KB avem:

$father(ken, liz)$.

$father(X, Y) \to ancestor(X, Y)$

$$\cfrac{\cfrac{}{father(ken, Z)}\; father(ken, liz)}{ancestor(ken, Z)}\; father(X, Y) \to ancestor(X, Y)$$

Puncte de decizie în programarea logică

Având doar această regulă, care sunt punctele de decizie în căutare?

- Ce clauză să alegem.
  - Pot fi mai multe clauze a căror parte dreaptă se potrivește cu o țintă.
  - Aceasta este o alegere de tip **SAU**: este suficient ca oricare din variante să reușească.
- Ordinea în care rezolvăm noile ținte.
  - Aceasta este o alegere de tip **ȘI:** trebuie arătate toate țintele noi.
  - Ordinea în care le rezolvăm poate afecta găsirea unei derivări, depinzând de strategia de căutare folosită.

Strategia de căutare din Prolog

Strategia de căutare din Prolog este de tip *depth-first*

- de sus în jos
  - pentru alegerile de tip **SAU**
  - alege clauzele în ordinea în care apar în program
- de la stânga la dreapta
  - pentru alegerile de tip **ȘI**
  - alege noile ținte în ordinea în care apar în clauza aleasă

Regula backchain și rezoluția SLD

- Regula *backchain* este implementată în programarea logică prin rezoluția SLD (Selected, Linear, Definite).
- Prolog are la bază rezoluția SLD.

Rezoluția SLD

Fie KB o mulțime de clauze definite.

$$\text{SLD} \quad \boxed{\dfrac{\neg Q_1 \vee \cdots \vee \neg Q_i \vee \cdots \vee \neg Q_n}{\theta(\neg Q_1 \vee \cdots \vee \neg P_1 \vee \cdots \vee \neg P_m \vee \cdots \vee \neg Q_n)}}$$

unde

- $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ este o clauză definită din KB în care toate variabilele au fost redenumite cu variabile noi
- θ este cmgu pentru Q_i și Q

Rezoluția SLD - exemplu

```
father(eddard,sansa).
father(eddard,jonSnow).

stark(eddard).
stark(catelyn).

stark(X) :- father(Y,X), stark(Y).
```

```
?- stark(jonSnow)
```

$$\text{SLD}\quad \boxed{\frac{\neg Q_1 \vee \cdots \vee \neg Q_i \vee \cdots \vee \neg Q_n}{\theta(\neg Q_1 \vee \cdots \vee \neg P_1 \vee \cdots \vee \neg P_m \vee \cdots \vee \neg Q_n)}}$$

- $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ este o clauză definită din *KB*
  variabilele din $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ se redenumesc
- θ este cmgu pentru Q_i și Q.

Rezoluția SLD - exemplu

$father(eddard, sansa)$
$father(eddard, jonSnow)$

$stark(eddard)$
$stark(catelyn)$

$stark(X) \vee \neg father(Y, X) \vee \neg stark(Y)$

$$\frac{\neg stark(jonSnow)}{}$$

$$\theta(X_1) = jonSnow$$

$$\text{SLD} \quad \boxed{\frac{\neg Q_1 \vee \cdots \vee \neg Q_i \vee \cdots \vee \neg Q_n}{\theta(\neg Q_1 \vee \cdots \vee \neg P_1 \vee \cdots \vee \neg P_m \vee \cdots \vee \neg Q_n)}}$$

- $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ este o clauză definită din *KB* variabilele din $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ se redenumesc
- θ este cmgu pentru Q_i și Q.

Rezoluția SLD - exemplu

$father(eddard, sansa)$
$father(eddard, jonSnow)$

$stark(eddard)$
$stark(catelyn)$

$stark(X) \vee \neg father(Y, X) \vee \neg stark(Y)$

$$\frac{\neg stark(jonSnow)}{\neg father(Y_1, jonSnow) \vee \neg stark(Y_1)}$$

$$\theta(X_1) = jonSnow$$

$$\text{SLD} \quad \boxed{\frac{\neg Q_1 \vee \cdots \vee \neg Q_i \vee \cdots \vee \neg Q_n}{\theta(\neg Q_1 \vee \cdots \vee \neg P_1 \vee \cdots \vee \neg P_m \vee \cdots \vee \neg Q_n)}}$$

- $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ este o clauză definită din *KB*
  variabilele din $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ se redenumesc
- θ este cmgu pentru Q_i și Q.

Rezoluția SLD - exemplu

$father(eddard, sansa)$
$father(eddard, jonSnow)$

$stark(eddard)$
$stark(catelyn)$

$stark(X) \vee \neg father(Y, X) \vee \neg stark(Y)$

$$\frac{\neg stark(jonSnow)}{\neg father(Y_1, jonSnow) \vee \neg stark(Y_1)}$$

$$\frac{\neg father(Y_1, jonSnow) \vee \neg stark(Y_1)}{\neg stark(eddard)}$$

$$\frac{\neg stark(eddard)}{\square}$$

$$\text{SLD} \quad \boxed{\frac{\neg Q_1 \vee \cdots \vee \neg Q_i \vee \cdots \vee \neg Q_n}{\theta(\neg Q_1 \vee \cdots \vee \neg P_1 \vee \cdots \vee \neg P_m \vee \cdots \vee \neg Q_n)}}$$

- $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ este o clauză definită din KB
- variabilele din $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ se redenumesc
- θ este cmgu pentru Q_i și Q.

Rezoluția SLD

Fie KB o mulțime de clauze definite și $Q_1 \wedge \ldots \wedge Q_m$ o întrebare, unde Q_i sunt formule atomice.

- O derivare din KB a întrebării prin rezoluție SLD este o secvență

  $$G_0 := \neg Q_1 \vee \ldots \vee \neg Q_m, \quad G_1, \quad \ldots, \quad G_k, \ldots$$

  în care G_{i+1} se obține din G_i prin regula SLD.
- Dacă există un k cu $G_k = \square$ (clauza vidă), atunci derivarea se numește SLD-respingere.

Rezoluția SLD

Fie KB o mulțime de clauze definite și $Q_1 \land \ldots \land Q_m$ o întrebare, unde Q_i sunt formule atomice.

- O derivare din KB a întrebării prin rezoluție SLD este o secvență
$$G_0 := \neg Q_1 \lor \ldots \lor \neg Q_m, \quad G_1, \quad \ldots, \quad G_k, \ldots$$
în care G_{i+1} se obține din G_i prin regula SLD.
- Dacă există un k cu $G_k = \square$ (clauza vidă), atunci derivarea se numește SLD-respingere.

Teoremă. Sunt echivalente:

1. există o SLD-respingere a lui $Q_1 \land \ldots \land Q_m$ din KB,
2. Există soluție pentru întrebarea $Q_1 \land \cdots \land Q_m$.

Rezoluția SLD - arbori de căutare

Arbori SLD

- Presupunem că avem o mulțime de clauze definite KB și o țintă $G_0 = \neg Q_1 \vee \ldots \vee \neg Q_m$
- Construim un arbore de căutare (arbore SLD) astfel:
  - Fiecare nod al arborelui este o țintă (posibil vidă)
  - Rădăcina este G_0
  - Dacă arborele are un nod G_i, iar G_{i+1} se obține din G_i folosind regula SLD folosind o clauză $C_i \in KB$, atunci nodul G_i are copilul G_{i+1}.
    Muchia dintre G_i și G_{i+1} este etichetată cu C_i.
- Dacă un arbore SLD cu rădăcina G_0 are o frunză $\square$ (clauza vidă), atunci există o SLD-respingere a lui G_0 din KB.

Rezoluția SLD - arbori de căutare

Exemplu.

Fie *KB* următoarea mulțime de clauze definite:

1. $grandfather(X, Z) :- father(X, Y), parent(Y, Z)$
2. $parent(X, Y) :- father(X, Y)$
3. $parent(X, Y) :- mother(X, Y)$
4. $father(ken, diana)$
5. $mother(diana, brian)$

Găsiți o respingere din *KB* pentru

$$? - grandfather(ken, Y)$$

Rezoluția SLD - arbori de căutare

Exemplu.
Fie *KB* următoarea mulțime de clauze definite:

1. $grandfather(X, Z) \vee \neg father(X, Y) \vee \neg parent(Y, Z)$
2. $parent(X, Y) \vee \neg father(X, Y)$
3. $parent(X, Y) \vee \neg mother(X, Y)$
4. $father(ken, diana)$
5. $mother(diana, brian)$

Găsiți o respingere din *KB* pentru

$$\neg grandfather(ken, Y)$$

Rezoluția SLD - arbori de căutare

Exemplu.

1. $grandfather(X,Z) \lor \neg father(X,Y) \lor \neg parent(Y,Z)$
2. $parent(X,Y) \lor \neg father(X,Y)$
3. $parent(X,Y) \lor \neg mother(X,Y)$
4. $father(ken, diana)$
5. $mother(diana, brian)$

$$
\begin{tikzcd}
& \neg grandfather(ken,Y) \arrow[d, "1"] & \\
& \neg father(ken,V) \lor \neg parent(V,Y) \arrow[d, "4"] & \\
& \neg parent(diana,Y) \arrow[dl, "2"'] \arrow[dr, "3"] & \\
\neg father(diana,Y) & & \neg mother(diana,Y) \arrow[d, "5"] \\
& & \square
\end{tikzcd}
$$

Limbajul Prolog

- Am afirmat că sistemul de inferență din spatele Prolog-ului este complet.
  - Dacă o întrebare este consecință logică a unei mulțimi de clauze, atunci există o derivare a întrebării.
- Totuși, strategia de căutare din Prolog este incompletă!
  - Chiar dacă o întrebare este consecință logică a unei mulțimi de clauze, Prolog nu găsește mereu o derivare a întrebării.

Limbajul Prolog - exemplu

```
warmerClimate :- albedoDecrease.
warmerClimate :- carbonIncrease.
iceMelts :- warmerClimate.
albedoDecrease :- iceMelts.
carbonIncrease.

?- iceMelts.
! Out of local stack
```

Limbajul Prolog - exemplu

```
warmerClimate :- albedoDecrease.
warmerClimate :- carbonIncrease.
iceMelts :- warmerClimate.
albedoDecrease :- iceMelts.
carbonIncrease.

?- iceMelts.
! Out of local stack
```

Limbajul Prolog - exemplu

Totuși, există o derivare a lui *iceMelts* în sistemul de deducție din clauzele:

$$
\begin{array}{rcl}
\mathit{albedoDecrease} & \to & \mathit{warmerClimate} \\
\mathit{carbonIncrease} & \to & \mathit{warmerClimate} \\
\mathit{warmerClimate} & \to & \mathit{iceMelts} \\
\mathit{iceMelts} & \to & \mathit{albedoDecrease} \\
\top & \to & \mathit{carbonIncrease}
\end{array}
$$

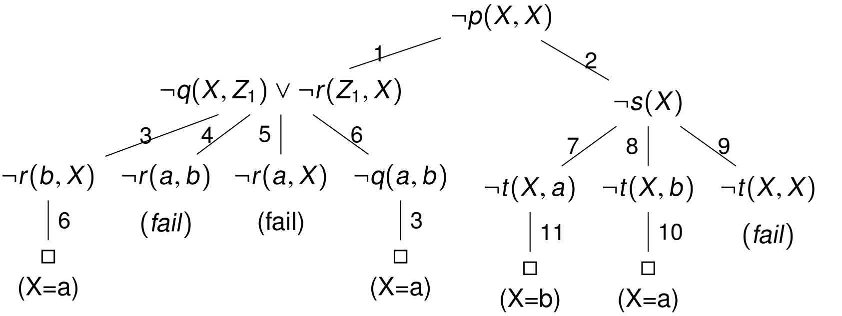
$$
\dfrac{\dfrac{\mathit{carbonInc}}{\mathit{warmerClim}}}{\mathit{iceMelts}} \quad \begin{array}{l} \mathit{carbonInc} \to \mathit{warmerClim} \\ \mathit{warmerClim} \to \mathit{iceMelts} \end{array}
$$

Exercițiu Desenați arborele SLD pentru programul Prolog de mai jos și ținta
`?- p(X,X).`

```
1.  p(X,Y) :- q(X,Z), r(Z,Y).
2.  p(X,X) :- s(X).
3.  q(X,b).
4.  q(b,a).
5.  q(X,a) :- r(a,X).
6.  r(b,a).
7.  s(X) :- t(X,a).
8.  s(X) :- t(X,b).
9.  s(X) :- t(X,X).
10.  t(a,b).
11.  t(b,a).
```

Rezoluția SLD - arbori de căutare

1. $p(X, Y) \vee \neg q(X, Z) \vee \neg r(Z, Y)$
2. $p(X, X) \vee \neg s(X)$
3. $q(X, b)$
4. $q(b, a)$
5. $q(X, a) \vee \neg r(a, X)$
6. $r(b, a)$
7. $s(X) \vee \neg t(X, a)$
8. $s(X) \vee \neg t(X, b)$
9. $s(X) \vee \neg t(X, X)$
10. $t(a, b)$
11. $t(b, a)$



Un exemplu inspirat din Minecraft

```
activate_structure(power_module(X), Y) :-
  verify_component(X),
  match_resource(Y, X).

verify_component(X) :-
  combine_elements(craft_token(Y), X),
  match_resource(Y, Y).

match_resource(oak_log, oak_log).
match_resource(redstone_block, redstone_block).

combine_elements(craft_token(oak_log), redstone_block).

?- activate_structure(X, Y)
```

Un exemplu inspirat din Minecraft

```
activate_structure(power_module(X), Y) :-
  verify_component(X),
  match_resource(Y, X).

verify_component(X) :-
  combine_elements(craft_token(Y), X),
  match_resource(Y, Y).

match_resource(oak_log, oak_log).
match_resource(redstone_block, redstone_block).

combine_elements(craft_token(oak_log), redstone_block).

?- activate_structure(X, Y)
```

Clauze asociate

1. $\forall as(pm(X), Y) \vee \neg vc(X) \vee \neg mr(Y, X)$
2. $\forall vc(X) \vee \neg ce(ct(Y), X) \vee \neg mr(Y, Y)$
3. $\forall mr(ol, ol)$
4. $\forall mr(rb, rb)$
5. $\forall ce(ct(ol), rb)$

Scop: $\forall \neg as(X, Y)$

Rezolvare prin rezoluție

Clauze asociate

1. $\forall as(pm(X), Y) \vee \neg vc(X) \vee \neg mr(Y, X)$
2. $\forall vc(X) \vee \neg ce(ct(Y), X) \vee \neg mr(Y, Y)$
3. $\forall mr(ol, ol)$
4. $\forall mr(rb, rb)$
5. $\forall ce(ct(ol), rb)$

Scop: $\forall \neg as(X, Y)$

Rezolvare prin rezoluție

Clauze asociate

1. $\forall as(pm(X), Y) \vee \neg vc(X) \vee \neg mr(Y, X)$
2. $\forall vc(X) \vee \neg ce(ct(Y), X) \vee \neg mr(Y, Y)$
3. $\forall mr(ol, ol)$
4. $\forall mr(rb, rb)$
5. $\forall ce(ct(ol), rb)$

Scop: $\forall \neg as(X, Y)$

Scop: $\forall \neg as(X, Y)$
Alegem clauza 1 și redenumim variabilele
$\forall as(pm(X_1), Y_1) \vee \neg vc(X_1) \vee \neg mr(Y_1, X_1)$
unificăm $as(X, Y)$ cu $as(pm(X_1), Y_1)$ și obținem substituția
$X \mapsto pm(X_1), Y_1 \mapsto Y$
Inlocuim în scop $\neg as(X, Y)$ cu $\neg vc(X_1) \vee \neg mr(Y_1, X_1)$
și aplicăm substituția.
Noul scop: $\forall \neg vc(X_1) \vee \neg mr(Y, X_1)$

Rezolvare prin rezoluție

Clauze asociate

1. $\forall as(pm(X), Y) \vee \neg vc(X) \vee \neg mr(Y, X)$
2. $\forall vc(X) \vee \neg ce(ct(Y), X) \vee \neg mr(Y, Y)$
3. $\forall mr(ol, ol)$
4. $\forall mr(rb, rb)$
5. $\forall ce(ct(ol), rb)$

Scop: $\forall \neg as(X, Y)$

Scop: $\forall \neg vc(X_1) \vee \neg mr(Y, X_1)$
Alegem clauza 2 și redenumim variabilele
$\forall vc(X_2) \vee \neg ce(ct(Y_2), X_2) \vee \neg mr(Y_2, Y_2)$
unificăm $vc(X_1)$ cu $vc(X_2)$ și obținem substituția
$X_2 \mapsto X_1$
Inlocuim în scop $\neg vc(X_1)$ cu $\neg ce(ct(Y_2), X_2) \vee \neg mr(Y_2, Y_2)$
și aplicăm substituția.
Noul scop: $\forall \neg ce(ct(Y_2), X_1) \vee \neg mr(Y_2, Y_2) \vee \neg mr(Y, X_1)$

Rezolvare prin rezoluție

Clauze asociate

1. $\forall as(pm(X), Y) \lor \neg vc(X) \lor \neg mr(Y, X)$
2. $\forall vc(X) \lor \neg ce(ct(Y), X) \lor \neg mr(Y, Y)$
3. $\forall mr(ol, ol)$
4. $\forall mr(rb, rb)$
5. $\forall ce(ct(ol), rb)$

Scop: $\forall \neg as(X, Y)$

Scop: $\forall \neg ce(ct(Y_2), X_1) \lor \neg mr(Y_2, Y_2) \lor \neg mr(Y, X_1)$
Alegem clauza 5 și redenumim variabilele
$\forall ce(ct(ol), rb)$
unificăm $ce(ct(Y_2), X_1)$ cu $ce(ct(ol), rb)$ și obținem substituția
$X_1 \mapsto rb, Y_2 \mapsto ol$
Inlocuim în scop $\neg ce(ct(Y_2), X_1)$ cu nimic
și aplicăm substituția.
Noul scop: $\forall \neg mr(ol, ol) \lor \neg mr(Y, rb)$

Rezolvare prin rezoluție

Clauze asociate

1. $\forall as(pm(X), Y) \vee \neg vc(X) \vee \neg mr(Y, X)$
2. $\forall vc(X) \vee \neg ce(ct(Y), X) \vee \neg mr(Y, Y)$
3. $\forall mr(ol, ol)$
4. $\forall mr(rb, rb)$
5. $\forall ce(ct(ol), rb)$

Scop: $\forall \neg as(X, Y)$

Scop: $\forall \neg mr(ol, ol) \vee \neg mr(Y, rb)$
Alegem clauza 3 și redenumim variabilele
$\forall mr(ol, ol)$
unificăm $mr(ol, ol)$ cu $mr(ol, ol)$ și obținem substituția identitate
Inlocuim în scop $\neg mr(ol, ol)$ cu nimic
și aplicăm substituția.
Noul scop: $\forall \neg mr(Y, rb)$

Rezolvare prin rezoluție

Clauze asociate

1. $\forall as(pm(X), Y) \lor \neg vc(X) \lor \neg mr(Y, X)$
2. $\forall vc(X) \lor \neg ce(ct(Y), X) \lor \neg mr(Y, Y)$
3. $\forall mr(ol, ol)$
4. $\forall mr(rb, rb)$
5. $\forall ce(ct(ol), rb)$

Scop: $\forall \neg as(X, Y)$

Scop: $\forall \neg mr(Y, rb)$
Alegem clauza 4 și redenumim variabilele
$\forall mr(rb, rb)$
unificăm $mr(Y, rb)$ cu $mr(rb, rb)$ și obținem substituția
$Y \mapsto rb$
Inlocuim în scop $\neg mr(Y, rb)$ cu nimic
și aplicăm substituția.
Noul scop: clauza vidă

Rezolvare prin rezoluție

Clauze asociate

1. $\forall as(pm(X), Y) \vee \neg vc(X) \vee \neg mr(Y, X)$
2. $\forall vc(X) \vee \neg ce(ct(Y), X) \vee \neg mr(Y, Y)$
3. $\forall mr(ol, ol)$
4. $\forall mr(rb, rb)$
5. $\forall ce(ct(ol), rb)$

Scop: $\forall \neg as(X, Y)$

Deci negația interogării nu este validă pentru substituția calculată:
$[Y \mapsto rb] \circ 1 \circ [X_1 \mapsto rb, Y_2 \mapsto ol] \circ [X_2 \mapsto X_1] \circ [X \mapsto pm(X_1), Y_1 \mapsto Y]$

Care prin simplificare devine: $[X \mapsto pm(rb), Y_1 \mapsto rb, X_2 \mapsto rb, Y_2 \mapsto ol]$

Așadar interogarea are soluție, dată de substituția calculată:

$$X \mapsto pm(rb), Y \mapsto rb$$