Fundamentele limbajelor de programare

C10

Denisa Diaconescu
Traian Șerbănuță

Departamentul de Informatică, FMI, UB

Programare Logică
Logica Clauzelor Horn

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

Spre logica din spatele Prologului

Pentru a putea modela universul programului, fixăm un alfabet/signatură/vocabular:

- o mulțime $\mathbf{R}$ de simboluri de relații/predicate
- o mulțime $\mathbf{F}$ de simboluri de operații/funcții
- o funcție aritate $ar : \mathbf{F} \cup \mathbf{R} \to \mathbb{N}$
- Fie $\mathbf{C} \subseteq \mathbf{F}$ mulțimea simbolurilor de operații de aritate 0, numite și simboluri de constante.

Notăm cu $\mathbf{R}_n$/$\mathbf{F}_n$ mulțimea simbolurilor de relații/operații de aritate n.
Observație: $\mathbf{F}_0 = C$

Exemplu.

- $\mathbf{R} = \mathbf{R}_1 \uplus \mathbf{R}_2$, unde $\mathbf{R}_1 = \{P\}$ și $\mathbf{R}_2 = \{R\}$
- $\mathbf{F} = \mathbf{F}_0 \uplus \mathbf{F}_2$, unde $\mathbf{F}_0 = C = \{c\}$ și $\mathbf{F}_2 = \{f\}$

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`

Termeni

Fixăm o mulțime numărabilă de variabile V.

Definim termenii inductiv astfel:

- orice variabilă este un termen
- orice simbol de constantă este un termen
- dacă $f \in \mathbf{F}_n$, $n > 0$ și $t_1, \ldots, t_n$ sunt termeni, atunci $f(t_1, \ldots, t_n)$ este termen.

Exemple: $c,\ x_1,\ f(x_1, c),\ f(f(x_2, x_2), c)$
unde $c \in \mathbf{C}$ reprezintă o constantă, $x_1, x_2 \in V$ sunt variabile, iar $f \in \mathbf{F}_2$ reprezintă un simbol de funcție de aritate 2.

Formule atomice

Formulele atomice sunt definite astfel:

dacă $R \in \mathbf{R}_0$, atunci R este formulă atomică

dacă $R \in \mathbf{R}_n$, $n > 0$ și $t_1, \ldots, t_n$ sunt termeni,
atunci $R(t_1, \ldots, t_n)$ este formulă atomică.

Exemple: $P(f(x_1, c)),\ R(c, x_2)$
unde $c \in \mathbf{C}$, $x_1, x_2 \in V$, $f \in \mathbf{F}_2$, $P \in \mathbf{R}_1$, iar $R \in \mathbf{R}_2$.

Formulele logicii de ordinul I

Definim formulele astfel:

- orice formulă atomică este o formulă
- dacă φ este o formulă, atunci $\neg\varphi$ este o formulă
- dacă φ și ψ sunt formule, atunci $\varphi \vee \psi$, $\varphi \wedge \psi$, $\varphi \rightarrow \psi$ sunt formule
- dacă φ este o formulă și x este o variabilă, atunci $\forall x\, \varphi$, $\exists x\, \varphi$ sunt formule

Formulele logicii de ordinul I

Definim formulele astfel:

- orice formulă atomică este o formulă
- dacă φ este o formulă, atunci $\neg\varphi$ este o formulă
- dacă φ și ψ sunt formule, atunci $\varphi \vee \psi$, $\varphi \wedge \psi$, $\varphi \rightarrow \psi$ sunt formule
- dacă φ este o formulă și x este o variabilă, atunci $\forall x\, \varphi$, $\exists x\, \varphi$ sunt formule

Exemplu:

$$P(f(x_1, c)), \quad P(x_1) \vee P(c), \quad \forall x_1\, P(x_1), \quad \forall x_2\, R(x_2, x_1)$$

unde $c \in \mathbf{C}$, $x_1, x_2 \in V$, $f \in \mathbf{F}_2$, $P \in \mathbf{R}_1$, iar $R \in \mathbf{R}_2$.

Exercițiu

Fie alfabetul definit prin $\mathbf{R} = \{<\}$, $\mathbf{F} = \{s, +\}$, $\mathbf{C} = \{0\}$ și $ari(s) = 1$, $ari(+) = ari(<) = 2$.

Dați exemple de 3 termeni, 3 formule atomice și 3 formule.

Exerciţiu

Fie alfabetul definit prin $\mathbf{R} = \{<\}$, $\mathbf{F} = \{s, +\}$, $\mathbf{C} = \{0\}$ şi $ari(s) = 1$, $ari(+) = ari(<) = 2$.

Daţi exemple de 3 termeni, 3 formule atomice şi 3 formule.

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 formule:

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

Literali

Un literal este o formulă atomică sau negația unei formule atomice.

O formulă este în formă normală conjunctivă (FNC) dacă este o conjuncție de disjuncții de literali.

Exemplu:

$(P(f(x_1, c)) \vee R(c, x_2)) \ \wedge \ \neg R(x_1, x_2) \ \wedge \ (R(x_1, x_1) \vee \neg P(c))$

unde $c \in \mathbf{C}$, $x_1, x_2 \in V$, $f \in \mathbf{F}_2$, $P \in \mathbf{R}_1$, iar $R \in \mathbf{R}_2$.

Clauze

- O clauză este o disjuncție de literali.
- Dacă $L_1, \ldots, L_n$ sunt literali atunci vom 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

Putem reprezenta o clauză prin mulțimea

$$\{\neg Q_1, \ldots, \neg Q_n, P_1, \ldots, P_k\}$$

unde $n, k \geq 0$ și $Q_1, \ldots, Q_n, P_1, \ldots, P_k$ sunt formule atomice.

Formula corespunzătoare este

$$\forall x_1 \ldots \forall x_m(\neg Q_1 \vee \ldots \vee \neg Q_n \vee P_1 \vee \ldots \vee P_k)$$

unde $x_1, \ldots, x_m$ sunt toate variabilele care apar în clauză

Echivalent, putem scrie

$$\forall x_1 \ldots \forall x_m(Q_1 \wedge \ldots \wedge Q_n \to P_1 \vee \ldots \vee P_k)$$

Presupunem cuantificarea universală a clauzelor implicită:

$$Q_1 \wedge \ldots \wedge Q_n \to P_1 \vee \ldots \vee P_k$$

Clauze program definite

- Clauză $Q_1 \wedge \ldots \wedge Q_n \rightarrow P_1 \vee \ldots \vee P_k$
  unde $n, k \geq 0$ și $Q_1, \ldots, Q_n, P_1, \ldots, P_k$ sunt formule atomice.

- Dacă $k = 1$, atunci avem o clauză program definită:
  - cazul $n > 0$: $Q_1 \wedge \ldots \wedge Q_n \rightarrow P$
  - cazul $n = 0$: $\top \rightarrow P$ (clauză unitate, fapt)
    $\top$ este simbol pentru o formula mereu adevărată

- Program logic definit = mulțime finită de clauze definite

Clauze Horn

- Clauză $Q_1 \wedge \ldots \wedge Q_n \to P_1 \vee \ldots \vee P_k$
  unde $n, k \geq 0$ și $Q_1, \ldots, Q_n, P_1, \ldots, P_k$ sunt formule atomice.

- Dacă $k = 0$, atunci avem o clauză scop definită (țintă, întrebare):

$$Q_1 \wedge \ldots \wedge Q_n \to \bot$$

  $\bot$ este simbol pentru o formula mereu falsă

  Vom scrie o clauza scop definită ca $Q_1, \ldots, Q_n$.

- În plus, dacă $n = k = 0$, atunci avem clauza vidă $\square$

Clauză Horn = clauză program definită sau clauză scop ($k \leq 1$)

Limbajul PROLOG are la bază logica clauzelor Horn.

Clauze Horn

Clauza Horn = clauză program definită sau clauză scop ($k \leq 1$)

- Clauză $Q_1 \wedge \ldots \wedge Q_n \rightarrow P_1 \vee \ldots \vee P_k$ unde $n, k \geq 0$ și $Q_1, \ldots, Q_n, P_1, \ldots, P_k$ sunt formule atomice.

- Fie $x_1, \ldots, x_m$ toate variabilele care apar într-o clauza scop $Q_1, \ldots, Q_n$. Atunci avem echivalența

$$\forall x_1 \ldots \forall x_m (\neg Q_1 \vee \ldots \vee \neg Q_n) \equiv \neg \exists x_1 \ldots \exists x_m (Q_1 \wedge \ldots \wedge Q_n)$$

Negația unei "întrebări" în PROLOG este clauză scop.

Programare logica

- Logica clauzelor Horn: un fragment al logicii de ordinul I în care singurele formule admise sunt clauze Horn
  - formule atomice: $P(t_1, \ldots, t_n)$
  - $Q_1 \land \ldots \land Q_n \to P$
    unde toate Q_i, P sunt formule atomice, $\top$ sau $\bot$

- Problema programării logice: reprezentăm cunoștințele ca o mulțime de clauze definite KB și suntem interesați să aflăm răspunsul la o întrebare de forma $Q_1 \land \ldots \land Q_n$, unde toate Q_i sunt formule atomice

$$KB \models Q_1 \land \ldots \land Q_n$$

  - Variabilele din KB sunt cuantificate universal.
  - Variabilele din $Q_1, \ldots, Q_n$ sunt cuantificate existențial.

Un exemplu

Fie următoarele clauze definite:

$$father(jon, ken).$$

$$father(ken, liz).$$

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

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

$$ancestor(X, Y) \wedge ancestor(Y, Z) \rightarrow ancestor(X, Z)$$

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

- Axiome: orice clauză din KB
- Regula de deducție *backchain*:

$$\frac{\theta(Q_1) \quad \theta(Q_2) \quad \ldots \quad \theta(Q_n) \quad (Q_1 \land Q_2 \land \ldots \land Q_n \to P)}{\theta(Q)}$$

unde $Q_1 \land Q_2 \land \ldots \land Q_n \to P \in KB$, iar θ cmgu pentru Q și P.

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

Pentru o mulțime de clauze KB și o țintă Q,
dacă $KB \models Q$,
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 clauză

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

cu unificatorul

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

pentru a obține o nouă țintă

$$father(ken, Z).$$

Sistem de deducție

Rergula backchain

$$\frac{\theta(Q_1) \quad \theta(Q_2) \quad \dots \quad \theta(Q_n) \quad (Q_1 \wedge Q_2 \wedge \dots \wedge Q_n \to P)}{\theta(Q)}$$

unde $Q_1 \wedge Q_2 \wedge \dots \wedge Q_n \to P \in KB$, iar θ este cgu pentru Q și P.

Exemplu. Presupunem că în KB avem:

$father(ken, liz)$.

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

$$\frac{\dfrac{father(ken, liz)}{father(ken, Z)} \quad father(X, Y) \to ancestor(X, Y)}{ancestor(ken, Z)}$$

Puncte de decizie în programarea logica

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.

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

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)
```

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

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

Rezoluția SLD - exemplu

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

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

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

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

$\theta(X) = jonSnow$

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

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

Rezoluția SLD - exemplu

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

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

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

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

$$\theta(X) = jonSnow$$

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

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

Rezoluția SLD - exemplu

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

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

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

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

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

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

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

- $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ este o clauză definită din *KB*
- variabilele din $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ și Q_i 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 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 $Q_1 \wedge \ldots \wedge Q_m$ o întrebare, unde Q_i sunt formule atomice.

- O derivare din KB 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.

Teoremă. Sunt echivalente:

1. există o SLD-respingere a lui $Q_1 \wedge \ldots \wedge Q_m$ din KB,
2. $KB \models Q_1 \wedge \cdots \wedge Q_m$.

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.

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

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

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

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

- Am arătat 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 căutate 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:

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

$$
\dfrac{\dfrac{\mathit{carbonInc.} \qquad \mathit{carbonInc.} \to \mathit{warmerClim.}}{\mathit{warmerClim.}} \quad \mathit{warmerClim.} \to \mathit{iceMelts}}{\mathit{iceMelts}}
$$

Exercițiu Desenați arborele SLD pentru programul Prolog de mai jos și ținta

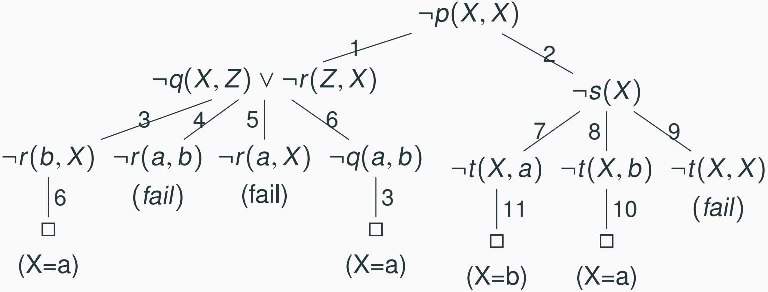
```
?- p(X,X).
```

```
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) :- 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).
```

$p(X,Y) \vee \neg q(X,Z) \vee \neg r(Z,Y)$
$p(X,X) \vee \neg s(X)$
$q(X,b)$
$q(b,a)$
$q(X,a) \vee \neg r(a,X)$
$r(b,a)$
$s(X) \vee \neg t(X,a)$
$s(X) \vee \neg t(X,b)$
$s(X) \vee \neg t(X,X)$
$t(a,b)$
$t(b,a)$



Pe data viitoare!