Lógica para Programação
LEIC-Alameda
2022

Ana Paiva

Programação em Lógica

(S3: A 1-2)

(estes slides são fortemente baseados nos slides gentilmente cedidos pelas Professoras Inês Lynce e Luísa Coheur, e qualquer gralha é da minha responsabilidade)

Programa

- ~~Conceitos Básicos (Livro: 1.1)~~
- ~~Lógica Proposicional – sistema dedutivo (2.1, 2.2.1, 2.2.2 e 2.2.4)~~
- ~~Lógica Proposicional (ou Cálculo de Predicados) – resolução(3.1)~~
- ~~Lógica de Primeira Ordem – sistema dedutivo (4.1, 4.2)~~
- ~~Lógica de Primeira Ordem – resolução (5.1 e 5.2)~~
- Programação em Lógica (6)
- Prolog (7 + Apêndice A: manual de sobrevivência em Prolog)
- Lógica Proposicional (ou de Predicados) – sistema semântico (2.3, 2.4, 3.2)

Programa de festas

- Cláusulas de Horn
- Programas
- Resolução SLD
- Árvores SLD

Especificação declarativa

- Os programas em lógica são especificados de uma forma "declarativa"
- **Programador**: descreve as propriedades lógicas que caracterizam o problema a resolver
- **Sistema**: utiliza a descrição e infere uma solução para o problema

1. **Programador**: representação do programa de um subconjunto de formulas da lógica de primeira ordem (Cláusulas de Horn)

2. **Sistema**: utiliza a estratégia de resolução designada por resolução-SLD

E o que é uma Cláusula de Horn?

Cláusulas de Horn

Definição: Cláusula de Horn é uma cláusula que contém no máximo um literal positivo

Sejam C , P1 e P2 predicados, são exemplos de cláusulas de Horn

- {C, ¬P1, ¬P2} ✓
- {C} ✓
- {C, P1, ¬P2} ✗
- {} ✓
- {¬P1, ¬P2} ✓

Cláusulas de Horn

Definição: Cláusula de Horn é uma cláusula que contém no máximo um literal positivo

Representação: Cabeça ← Corpo

Ou seja: o literal positivo, a existir, fica na cabeça da cláusula;
os restantes no corpo)

Exemplos:

$\{C, \neg P1, \neg P2\}$	representa-se por: $C \leftarrow P1, P2$
$\{C\}$	representa-se por: $C \leftarrow$
$\{\neg P1, \neg P2\}$	representa-se por: $\leftarrow P1, P2$

- $\square$ representa a Cláusula vazia

Tipos de Cláusulas de Horn

1. Regras ou implicações

   Cláusulas em que a cabeça e o corpo contêm literais

   Ex: C ← P1, P2

2. Afirmações ou factos

   Cláusulas em que o corpo não contém literais (é vazio)

   Ex: C ←

3. Objectivos

   Cláusulas em que a cabeça não contém literais (é vazio)

   Ex: ← P1, P2

As cláusulas do tipo 1 e 2 são chamadas cláusulas determinadas (do Inglês, definite clauses)

Exemplo da aula passada LPO

- Os antepassados dos ascendentes directos (Pai ou Mãe) são antepassados:

  $\forall_{x,y,z}$ [Ant(x , y) ∧ AD(y , z) → Ant(x , z)

  {¬Ant(x , y), ¬AD(y , z), Ant(x , z)}

- Os ascendentes directos são antepassados:

  ∀ x,y [AD(x , y) → Ant(x , y)]

  {¬AD(x , y), Ant(x , y)}

- O Pedro é um ascendente directo da Luisa:

  AD(Pedro, Luisa)

  {AD(Pedro, Luisa)}

- O Rui é um ascendente directo do Pedro:

  AD(Rui , Pedro)

  {AD(Rui , Pedro)}

{C, ¬P1, ¬P2} é C ← P1, P2

- Os antepassados dos ascendentes directos (Pai ou Mãe) são antepassados:

  {¬Ant(x , y), ¬AD(y , z), Ant(x , z)}

  Ant(x , z) ← Ant(x , y), AD(y , z)

- Os ascendentes directos são antepassados:

  {¬AD(x , y), Ant(x , y)}

  Ant(x , y) ← AD(x , y)

- O Pedro é um ascendente directo da Luisa:

  {AD(Pedro, Luisa)}

  AD(Pedro, Luisa) ←

- O Rui é um ascendente directo do Pedro:

  {AD(Rui , Pedro)}

  AD(Rui , Pedro) ←

Resolução com Cláusulas Horn: exemplo

Anteriormente tínhamos: (O Rui é antepassado da Luísa?)

$\{\neg AD(x, y), Ant(x, y)\}$

$\{\neg Ant(x, y), \neg AD(y, z), Ant(x, z)\}$

$\{AD(Pedro, Luísa)\}$

$\{AD(Rui, Pedro)\}$

$\{\neg Ant(Rui, Luísa)\}$

Agora

$Ant(x, y) \leftarrow AD(x, y)$	regra (e cláusula determinada)
$Ant(x, z) \leftarrow Ant(x, y), AD(y, z)$	regra (e cláusula determinada)
$AD(Pedro, Luísa) \leftarrow$	afirmação (e cláusula determinada)
$AD(Rui, Pedro) \leftarrow$	afirmação (e cláusula determinada)
$\leftarrow Ant(Rui, Luísa)$	objectivo

Relembrar: Resolução usando cláusulas com variáveis

Princípio da resolução para os casos em que as cláusulas contêm variáveis

- Sejam Ψ (Psi) e Φ (Phi) duas fórmulas
- Sejam α e β dois literais tais que $\alpha \in \Psi$ e $\neg\beta \in \Phi$
- Seja s = mgu(α, β) (unificador mais geral)

Então usando resolução podemos inferir a cláusula

$$((\Psi - \{\alpha\}) \cup (\Phi - \{\neg\beta\})) \cdot s$$

-> A clausula obtida é chamada **resolvente** de Ψ (Psi) e Φ (Phi)
-> α e β são os literais em conflito

Resolução com Cláusulas de Horn

Usamos sempre um objectivo e uma cláusula determinada para gerar cada resolvente

- Seja s o unificador mais geral para α e β_i
  - Regra da resolução

$$\frac{\begin{array}{l}\alpha \leftarrow \gamma_1, \ldots, \gamma_n \\ \delta \leftarrow \beta_1, \ldots, \underline{\beta_{i-1}}, \beta_i, \beta_{i+1}, \ldots, \beta_n\end{array}}{(\delta \leftarrow \beta_1, \ldots, \beta_{i-1}, \gamma_1, \ldots, \gamma_n, \beta_{i+1}, \ldots, \beta_n) \cdot s} \quad Res$$

Regra da resolução (adaptada – há sempre um objectivo na unificação)

$$\frac{\begin{array}{l}\alpha \leftarrow \gamma_1, \ldots, \gamma_n \\ \leftarrow \beta_1, \ldots, \beta_{i-1}, \underline{\beta_i}, \beta_{i+1}, \ldots, \beta_n\end{array}}{\leftarrow (\beta_1, \ldots, \beta_{i-1}, \gamma_1, \ldots, \gamma_n, \beta_{i+1}, \ldots, \beta_n) \cdot s} \quad Res$$

Resolução com Cláusulas Horn: exemplo (2)

1.	Ant(x, y) ← AD(x, y)	Prem
2.	Ant(x, z) ← Ant(x, y), AD(y, z)	Prem
3.	AD(Pedro, Luísa) ←	Prem
4.	AD(Rui, Pedro) ←	Prem
5.	← Ant(Rui, Luísa)	Prem
6.	Ant(Rui , Pedro) ←	Res, (1,4), {Rui/x,Pedro/y}
7.	Ant(Rui , z) ← AD(Pedro , z)	Res, (2,6), {Rui/x,Pedro/y}
8.	Ant(Rui , Luísa) ←	Res (3,7), {Luísa/z}
9.	□	Res (5,8)

Mas afinal, o que é que isto tem a ver com programação?

Programas

- Programa = conjunto finito de cláusulas determinadas
  - Premissas do exemplo anterior constituem um programa
    - $\mathtt{Ant}(\mathtt{x}, \mathtt{y}) \leftarrow \mathtt{AD}(\mathtt{x}, \mathtt{y})$
    - $\mathtt{Ant}(\mathtt{x}, \mathtt{z}) \leftarrow \mathtt{Ant}(\mathtt{x}, \mathtt{y}), \mathtt{AD}(\mathtt{y}, \mathtt{z})$
    - $\mathtt{AD}(\mathrm{Pedro}, \mathrm{Luísa}) \leftarrow$
    - $\mathtt{AD}(\mathrm{Rui}, \mathrm{Pedro}) \leftarrow$

Conceito: definição do predicado P

Definição do predicado P = conjunto de todas as cláusulas cuja cabeça corresponde ao predicado P (o conceito de P!)

Exemplo: a definição de Ant no exemplo anterior

$$\mathrm{Ant}(x, z) \leftarrow \mathrm{Ant}(x, y), \mathrm{AD}(y, z)$$

$$\mathrm{Ant}(x, y) \leftarrow \mathrm{AD}(x, y)$$

Ou a definição de AD no exemplo anterior:

$$\mathrm{AD}(\mathrm{Pedro}, \mathrm{Luísa}) \leftarrow$$

$$\mathrm{AD}(\mathrm{Rui}, \mathrm{Pedro}) \leftarrow$$

Conceito: Base de dados

- Base de dados = definição de um predicado que contém apenas cláusulas sem variáveis (ditas chãs-grounded)
  - Base de dados para AD no exemplo anterior

$$\mathrm{AD}(\mathrm{Pedro}, \mathrm{Luísa}) \leftarrow$$
$$\mathrm{AD}(\mathrm{Rui}, \mathrm{Pedro}) \leftarrow$$

TÉCNICO LISBOA

Conceito: Resposta de um programa a um objectivo

Seja Δ um programa e α um objectivo

Uma resposta de Δ a α é uma substituição s (eventualmente vazia) para as variáveis de α

Nota: a restrição de uma substituição **s** ao conjunto de variáveis $\{x_1, \ldots, x_m\}$ ´e dada por

- $s\,|\{x_1,\ldots,x_m\} = \{t_i/x_i \in s : x_i \in \{x_1, \ldots, x_m\}\}$

Logo, uma resposta de Δ a α ´e dada por $s\,|_{\nu(\alpha)}$ em que $\nu(\alpha)$ devolve as variáveis de α

Uma resposta s de um programa Δ a um objectivo α diz-se correcta se $\Delta \models (\alpha \cdot s)$

- s é uma resposta correcta se $\Delta \cup \{\neg\alpha \cdot s\}$ for contraditório
  - Para o exemplo anterior e dado o objectivo
    $\leftarrow Ant(Rui, Luísa)$ a resposta correcta é a substituição vazia (representada por ε)

Ok, mas a resolução era o caos... tínhamos que ter "intuição", e íamos aplicado a resolução por "olhómetro".... E no teste até correu ... menos bem.... Não há forma de garantirmos o determinismo?????

Exemplo anterior: Resolução com Cláusulas Horn

1. $Ant(x, y) \leftarrow AD(x, y)$	Prem
2. $Ant(x, z) \leftarrow Ant(x, y), AD(y, z)$	Prem
3. $AD(Pedro, Luísa) \leftarrow$	Prem
4. $AD(Rui, Pedro) \leftarrow$	Prem
5. $\leftarrow Ant(Rui, Luísa)$	Prem
6. $Ant(Rui, Pedro) \leftarrow$	Res, (1,4), {Rui/x,Pedro/y}
7. $Ant(Rui, z) \leftarrow AD(Pedro, z)$	Res, (2,6), {Rui/x,Pedro/y}
8. $Ant(Rui, Luísa) \leftarrow$	Res (3,7), {Luísa/z}
9. $\square$	Res (5,8)

Porquê?

Resolução – problema

De facto a resolução origina um processo não determinístico: podem existir diferentes outputs para o mesmo input

Porquê:

- Não estão definidas quais as cláusulas a resolver...
- Não estão definidos quais os literais a resolver depois de escolhidas duas cláusulas...
- Algoritmo de unificação também não é determinístico...

Solução: utilização de estratégias de resolução para eliminar o não determinismo

Resolução SLD (do Inglês "Selective Linear Definite clause resolution")

Estratégia =

- (1) Função de selecção S (dos literais)
- (2) + Regra de procura P (das cláusulas):

(1) **Função de selecção S** (ou regra de computação): escolhe de um modo determinístico um literal numa cláusula objectivo:

$$S(\leftarrow \alpha 1, \ldots, \alpha n) \in \{\alpha 1, \ldots, \alpha n\}$$

- Exemplo de uma função de selecção:

  $S(\leftarrow \alpha 1, \ldots, \alpha n) = \alpha 1$ quer dizer que se escolhe sempre o primeiro literal dos objectivos

(2) **Regra de procura P**: dado o literal (α) escolhido pela função de seleção, escolhe uma cláusula determinada (de Δ) de um programa (para se poder aplicar o princípio da resolução): $(P(\alpha, \Delta) \in \Delta)$

TÉCNICO LISBOA

Resolução SLD (cont.)

- Resolução linear: para gerar cada resolvente, usar sempre uma cláusula inicial ou um dos sucessores dessa cláusula (cláusulas centrais)

- Nota: Em resolucão-SLD, cláusula inicial = objectivo

1. Ant(x, y) ← AD(x, y)	Prem
2. Ant(x, z) ← Ant(x, y), AD(y, z)	Prem
3. AD(Pedro, Luísa) ←	Prem
4. AD(Rui, Pedro) ←	Prem
5. ← Ant(Rui, Luísa)	Prem objectivo
6. Ant(Rui , Pedro) ←	Res, (1,4), {Rui/x,Pedro/y}
7. Ant(Rui , z) ← AD(Pedro , z)	Res, (2,6), {Rui/x,Pedro/y}
8. Ant(Rui , Luísa) ←	Res (3,7), {Luísa/z}
9.	Res (5,8)

Ou seja, não deveríamos ter começado por 6...

Resolução SLD (cont.)

Processo: encontra a resposta de um programa a um determinado objectivo que que vai:

(1) Substituindo sucessivamente cada literal no objectivo pelo corpo de uma cláusula cuja cabeça seja unificável com o objectivo

(2) Processo é repetido até que (condições de paragem):

- Não existem mais objectivos (devido à geração da cláusula vazia)
- Nenhum dos objectivos é unificável com a cabeça de uma cláusula do programa

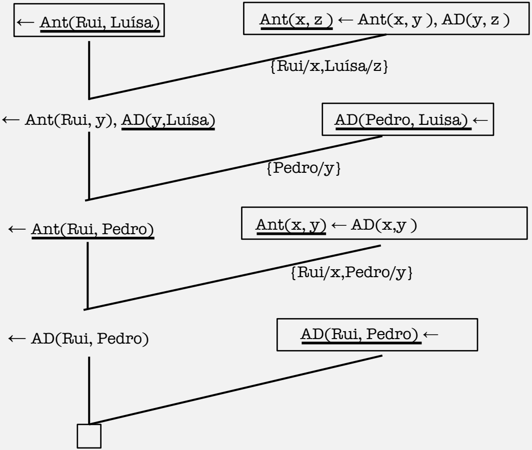
Resolução SLD: exemplo

Programa

1. Ant(x, y) ← AD(x, y)
2. Ant(x, z) ← Ant(x, y), AD(y, z)
3. AD(Pedro, Luísa) ←
4. AD(Rui, Pedro) ←

Objectivo

← Ant(Rui, Luísa)



Resposta calculada= $(\{Rui/x, Luisa/z\} \circ \{Pedro/y\} \circ \{Rui/x, Pedro/y\} \circ \epsilon) \mid_{\{\}} = \epsilon$

- Seja Δ um programa, α um objectivo e S uma função de selec¸c˜ao
- Uma prova SLD para Δ é uma sequência $[\gamma_0, \gamma_1, \ldots, \gamma_n]$ de objectivos satisfazendo as seguintes propriedades
  1. $\gamma_0 = \alpha$
  2. Para cada γ_i da sequência $(0 \leq i)$ se
     - $\gamma_i = \leftarrow \beta_1, \ldots, \beta_{k-1}, \beta_k, \beta_{k+1}, \ldots, \beta_j$
     - $\beta_k = S(\leftarrow \beta_1, \ldots, \beta_{k-1}, \beta_k, \beta_{k+1}, \ldots, \beta_j)$

     e existe uma cláusula em Δ

     - $\alpha \leftarrow \delta_1, \ldots, \delta_p$

     tal que β_k e α são unificáveis, sendo s_i o seu mgu então

     - $\gamma_{i+1} = \leftarrow (\beta_1, \ldots, \beta_{k-1}, \delta_1, \ldots, \delta_p, \beta_{k+1}, \ldots, \beta_j) \cdot s_i$

Resposta calculada

Se uma prova SLD [$\gamma 0, \gamma 1, \ldots, \gamma n$] é finita, então uma resposta calculada de Δ a α é dada pela restrição da composição de substituições $s_0, s_1, \ldots, s_n$ ao conjunto de variáveis que ocorrem em α

– $(s0 \circ s1 \circ \ldots \circ sn) \mid_{\nu(\alpha)}$

E n é o comprimento da prova SLD

Refutação SLD

Uma prova SLD é uma refutação SLD se e só se a sequência [$\gamma 0, \gamma 1, \ldots, \gamma n$] é finita e o seu último elemento γn corresponde à clausula vazia ($\gamma n = \square$)

Outro exemplo: Refutação SLD

Programa

1. $R(a,b) \leftarrow$
2. $R(b,c) \leftarrow$
3. $R(x,y) \leftarrow R(x,z), R(z,y)$

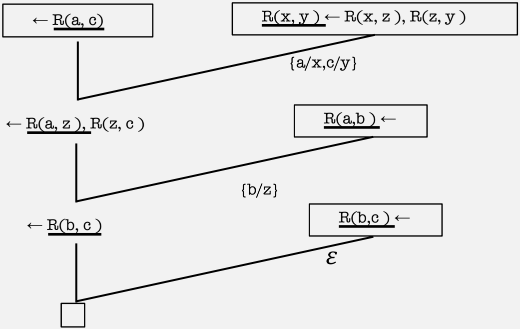
Objectivo

$\leftarrow R(a,c)$

Cenário 1. considere-se que:

(a) A função de selecção escolhe o primeiro literal no objectivo

(b) Regra de procura escolhe aleatoriamente uma cláusula determinada



Refutação SLD:

$\gamma_0 = \leftarrow R(a,c)$

$\gamma_1 = \leftarrow R(a,z), R(z,c)$ $\quad \{a/x, c/y\}$

$\gamma_2 = \leftarrow R(b,c)$ $\quad \{b/z\}$

$\gamma_3 = \square$ $\quad \{\ \}$

- Resposta dada por $(\{a/x, c/y\} \circ \{b/z\} \circ \epsilon)|_{\{\}} = \epsilon$

Outro exemplo: Refutação SLD

Programa

1. $R(a,b) \leftarrow$
2. $R(b,c) \leftarrow$
3. $R(x,y) \leftarrow R(x,z), R(z,y)$

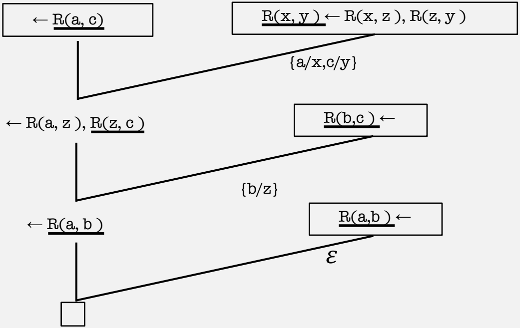
Objectivo

$\leftarrow R(a,c)$

Cenário 2. considere-se que:

(a) **A função de selecção** escolhe o **último** literal no objectivo

(b) **Regra de procura** escolhe aleatoriamente uma cláusula determinada



Refutação SLD:

$\gamma_0 = \leftarrow R(a,c)$

$\gamma_1 = \leftarrow R(a,z), R(z,c) \quad \{a/x, c/y\}$

$\gamma_2 = \leftarrow R(a,b) \quad \{b/z\}$

$\gamma_3 = \square \quad \{\ \}$

- Resposta dada por $(\{a/x, c/y\} \circ \{b/z\} \circ \epsilon)\,|_{\{\}} = \epsilon$

Outro exemplo: Refutação SLD

Programa

1. $R(a, b) \leftarrow$
2. $R(b, c) \leftarrow$
3. $R(x, y) \leftarrow R(x, z), R(z, y)$

Objectivo

$\leftarrow R(a, c)$

Cenário 3. considere-se que:

(a) A função de selecção escolhe o último literal no objectivo

(b) Regra de procura escolhe a clausula determinada com maior número de literais no corpo

Refutação SLD:

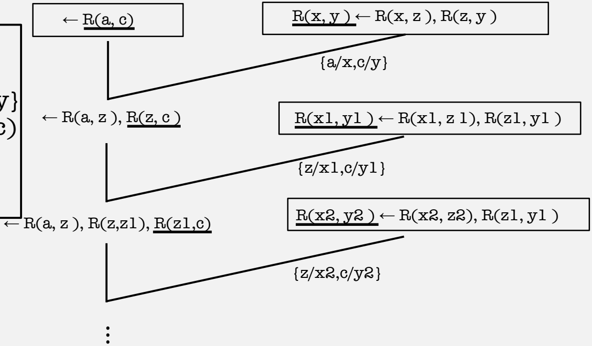
$\gamma_0 = \leftarrow R(a, c)$

$\gamma_1 = \leftarrow R(a, z), R(z, c) \quad \{a/x, c/y\}$

$\gamma_2 = \leftarrow R(a, z), R(z, z1), R(z1, c)$

$\{z/x1, c/y1\}$

$\gamma_3 =$ ……



Ideia: procurar todos....

Como resolver?????

Como sabemos qual é o melhor caminho na procura???

Árvores SLD: definição

- Seja Δ um programa, α um objectivo e S uma função de seleção
- A árvore SLD de Δ via α é uma árvore rótulada construída do seguinte modo
  1. O rótulo de cada nó é um objectivo
  2. O rótulo da raiz é α
  3. Cada nó com rótulo $\leftarrow \beta_1, \ldots, \beta_n$ tem um ramo por cada cláusula $\delta \leftarrow \gamma_1, \ldots, \gamma_p$ de Δ cuja cabeça δ seja unificável com $S(\leftarrow \beta_1, \ldots, \beta_n)$
     - O rótulo da raiz deste ramo corresponde ao resolvente entre as duas cláusulas

  – Um ramo cuja folha tem o rótulo Q diz-se um ramo bem sucedido
  – Um ramo cuja folha não tem o rótulo Q diz-se um ramo falhado
  – Os restantes ramos dizem-se ramos infinitos

Árvore SLD: exemplo

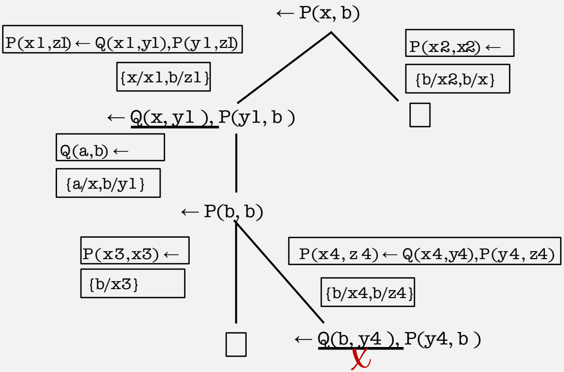
- Função de selecção
  - $S_1(\leftarrow \alpha_1, \ldots, \alpha_n) = \alpha_1$

Programa

1. $P(x,z) \leftarrow Q(x,y), P(y,z)$
2. $P(x,x) \leftarrow$
3. $Q(a,b) \leftarrow$

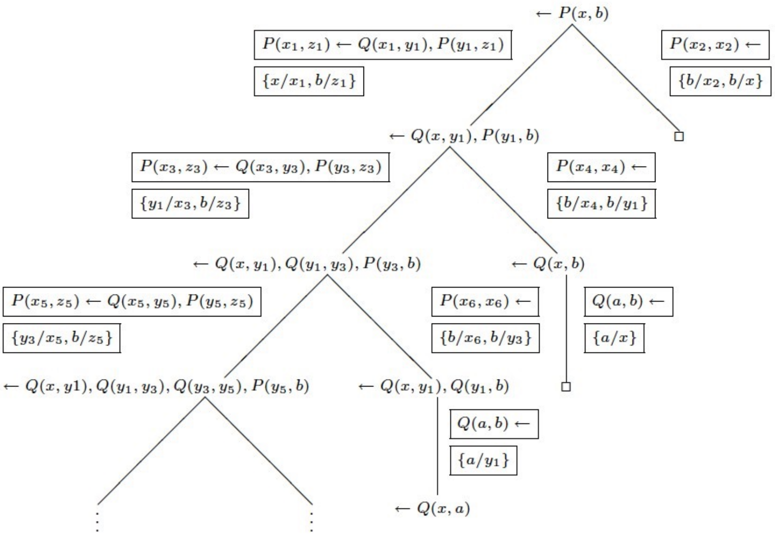
Objectivo

$\leftarrow P(x,b)$



Respostas: ramo da esquerda $\{a/x\}$; ramo da direita $\{b/x\}$

Árvore SLD: exemplo para $S_2(\leftarrow \alpha_1, \ldots, \alpha_n) = \alpha_n$



Respostas: ramo da esquerda $\ldots, \{a/x\}$; ramo da direita $\{b/x\}$

Então e o Prolog, o que tem a ver com isto?

Programação em Lógica vs. Prolog

- Programa
  - $P(x, z) \leftarrow Q(x, y), P(y, z)$
    Prolog: p(X, Z) :- q(X, Y), p(Y, Z).
  - $P(x, x) \leftarrow$
    Prolog: p(X, X).
  - $Q(a, b) \leftarrow$
    Prolog: q(a, b).
- Objectivo
  - $\leftarrow P(x, b)$
    Prolog: ?- p(X, b).
- Função de selecção do Prolog (o primeiro literal do objectivo):
  - $S_1(\leftarrow \alpha_1, \dots, \alpha_n) = \alpha_1$

Então e o Prolog, o que tem a ver com isto?

TPC

SWI Prolog

Robust, mature, free. **Prolog for the real world.**

HOME | DOWNLOAD | DOCUMENTATION | TUTORIALS | COMMUNITY | USERS | WIKI

SWI-Prolog offers a comprehensive free Prolog environment. Since its start in 1987, SWI-Prolog development has been driven by the needs of real world applications. SWI-Prolog is widely used in research and education as well as commercial applications. Join over a million users who have downloaded SWI-Prolog. more ...

Download SWI-Prolog

Get Started

Try SWI-Prolog online (SWISH)

Try SWI-Prolog in your browser (WASM)