6. Fundamentos da Programação em Lógica

Programação em lógica:
O programador descreve as propriedades lógicas do problema a resolver.
Esta descrição é utilizada pelo sistema para *inferir* uma solução para o problema.

Por questões de eficiência utiliza-se

- um tipo especial de *cláusulas*: as cláusulas de Horn,
- uma estratégia de resolução particular: a resolução SLD.

Definição 6.1.1 (Cláusula de Horn)

Cláusula que contém, no máximo, um literal positivo.

Numa cláusula de Horn:

- O literal positivo (se existir) é chamado a *cabeça* da cláusula.
- Os literais negativos (se existirem) são chamados o *corpo* da cláusula.

Exemplos:

$\{C, \neg P_1, \neg P_2\}$
$\{C\}$
$\{\neg P_1, \neg P_2\}$
$\{\}$

Notação:

Dada a equivalência entre $(\alpha_1 \wedge \ldots \wedge \alpha_n) \rightarrow \beta$ e a cláusula de Horn $\{\neg\alpha_1, \ldots, \neg\alpha_n, \beta\}$, é vulgar escrever esta cláusula como $\beta \leftarrow \alpha_1, \ldots, \alpha_n$. Usando esta notação, a cláusula vazia é representada pelo símbolo $\square$.

Exemplos:

$C \leftarrow P_1, P_2$

$C \leftarrow$

$\leftarrow P_1, P_2$

$\square$

Tipos de cláusulas de Horn:

1. *Regras* ou *implicações*: tanto a cabeça como o corpo contêm literais.
2. *Afirmações* ou *factos*: o corpo é vazio, mas a cabeça contém um literal.
3. *Objetivos*: a cabeça é vazia, mas o corpo não.
   Cada um dos literais que constituem um objetivo tem o nome de *sub-objetivo*.
4. A cláusula vazia.

Definição 6.1.2 (Cláusula determinada)

As regras e as afirmações, isto é, as cláusulas com cabeça, têm o nome de *cláusulas determinadas*.

Exemplo:

$Ant(x, y) \leftarrow AD(x, y)$	Regra
$Ant(x, z) \leftarrow Ant(x, y),\ AD(y, z)$	Regra
$AD(Marge, Bart) \leftarrow$	Afirmação
$AD(Sr.B, Marge) \leftarrow$	Afirmação
$\leftarrow Ant(Sr.B, Bart)$	Objetivo

As 4 primeiras cláusulas são cláusulas determinadas.

Usando a resolução com cláusulas de Horn, um dos resolventes é obrigatoriamente uma cláusula determinada, pois só estas contêm literais positivos.

Assumindo que s é o unificador mais geral dos literais α e β_i, a regra da resolução com cláusulas de Horn é:

n	$\alpha \leftarrow \gamma_1, \ldots, \gamma_m$	
m	$\delta \leftarrow \beta_1, \ldots, \beta_{i-1}, \ \beta_i, \ \beta_{i+1}, \ldots, \beta_n$	
$m+1$	$(\delta \leftarrow \beta_1, \ldots, \beta_{i-1}, \gamma_1, \ldots, \gamma_m, \beta_{i+1}, \ldots, \beta_n) \cdot s$	Res, (n, m)

Exemplo:

1	$Ant(x, y) \leftarrow AD(x, y)$	Prem
2	$Ant(x, z) \leftarrow Ant(x, y),\ AD(y, z)$	Prem
3	$AD(Marge, Bart) \leftarrow$	Prem
4	$AD(Sr.B, Marge) \leftarrow$	Prem
5	$\leftarrow Ant(Sr.B, Bart)$	Prem
6	$Ant(Sr.B, Marge) \leftarrow$	Res, (1, 4), $\{Sr.B/x, Marge/y\}$
7	$Ant(Sr.B, z) \leftarrow AD(Marge, z)$	Res, (2, 6), $\{Sr.B/x, Marge/y\}$
8	$Ant(Sr.B, Bart) \leftarrow$	Res, (3, 7), $\{Bart/z\}$
9	$\square$	Res, (5, 8), ε

Em programação em lógica:

- informação disponível (premissas) é representada por um conjunto de cláusulas determinadas.
- cláusula cujas instâncias se pretendem derivar a partir desse programa é representada por um objetivo.

Definição 6.2.1(Programa)

Um *programa* é qualquer conjunto finito de cláusulas determinadas.

Exemplo de programa:

$$Ant(x, y) \leftarrow AD(x, y)$$

$$Ant(x, z) \leftarrow Ant(x, y),\ AD(y, z)$$

$$AD(Marge, Bart) \leftarrow$$

$$AD(Sr.B, Marge) \leftarrow$$

Exemplo de objetivo:

$$\leftarrow Ant(Sr.B, Bart)$$

Definição 6.2.2 (Definição de um predicado)

Num programa, o conjunto de todas as cláusulas cuja cabeça corresponde a um literal contendo a letra de predicado P, diz-se a *definição* de P.

Exemplo:

No programa anterior, a definição de Ant é dada pelo conjunto:

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

No mesmo programa, a definição de AD é dada pelo conjunto:

$$\{AD(Marge, Bart) \leftarrow, \quad AD(Sr.B, Marge) \leftarrow\}.$$

Definição 6.2.3 (Base de dados)

Uma definição de um predicado que contenha apenas cláusulas fechadas chama-se uma *base de dados* para esse predicado.

Exemplo:

No programa anterior, o conjunto

$$\{AD(Marge, Bart) \leftarrow, \quad AD(Sr.B, Marge) \leftarrow\}$$

é uma base de dados para AD.

Definição 6.2.4 (Resposta de um programa a um objetivo)

Sendo Δ um programa e α um objetivo, uma *resposta* de Δ ao objetivo α é uma substituição s para as variáveis de α.

Definição 6.2.5 (Restrição de uma substituição a variáveis)

Sendo s uma substituição e $\{x_1, \ldots, x_m\}$ um conjunto de variáveis, define-se a *restrição de s ao conjunto de variáveis* $\{x_1, \ldots, x_m\}$, escrita $s\,|_{\{x_1,\ldots,x_m\}}$, como sendo o conjunto

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

Definição 6.2.6 (Resposta correta de um programa)

Uma resposta s de Δ ao objetivo α diz-se *correta* se $\Delta \models (\alpha \cdot s)$.

Exemplos:

Dado o programa

$$Ant(x, y) \leftarrow AD(x, y)$$

$$Ant(x, z) \leftarrow Ant(x, y),\ AD(y, z)$$

$$AD(Marge, Bart) \leftarrow$$

$$AD(Sr.B, Marge) \leftarrow$$

e o objetivo $\leftarrow Ant(x, Bart)$,
tanto $s_1 = \{Homer/x\}$, como $s_2 = \{Sr.B/x\}$ são respostas do programa ao objetivo.

No entanto, só s_2 é uma resposta correta.

A utilização do princípio da resolução origina um processo não determinístico: não estão definidas quais a cláusulas a resolver, nem estão determinados quais os literais a resolver depois de escolhidas duas cláusulas para resolver.

As estratégias de resolução apresentadas resolvem parcialmente este problema.
Para o resolver na totalidade precisamos de introduzir os seguintes conceitos:

- *função de seleção*
- *regra de procura*

Definição 6.3.1 (Função de seleção)

Uma *função de seleção*, S, é uma regra para escolher um literal num objetivo: $S(\leftarrow \alpha_1, \ldots, \alpha_n) \in \{\alpha_1, \ldots, \alpha_n\}$.
O literal escolhido será o candidato à aplicação do princípio da resolução.

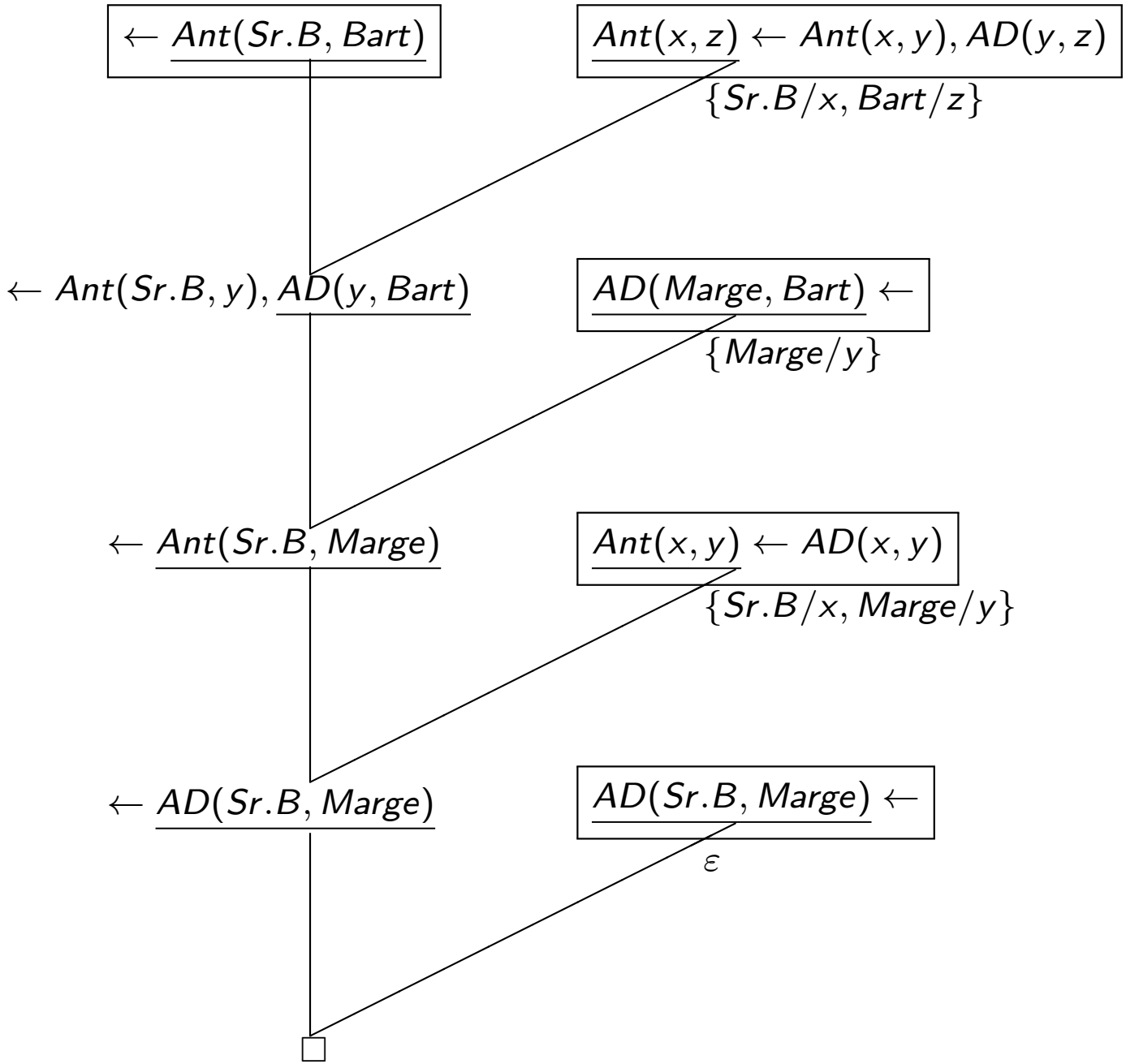
Uma vez escolhido o literal, é escolhida uma cláusula do programa cuja cabeça unifique com o literal escolhido.

Definição 6.3.5 (Regra de procura)

Dado um programa Δ e um literal α (correspondente a um sub-objetivo), uma regra de procura escolhe uma cláusula de Δ cuja cabeça unifique com α, se tal cláusula existir: $P(\alpha, \Delta) \in \Delta$.

A resolução SLD é uma estratégia de resolução linear aplicável a um conjunto de cláusulas determinadas e um objetivo, juntamente com uma função de seleção.
(“*SLD-resolution*” - “*L*inear resolution with *S*election function and *D*efinite clauses”.)

Informalmente, a resolução SLD parte de um objetivo, substituindo sucessivamente, cada sub-objetivo pelo corpo de uma cláusula cuja cabeça seja unificável com o sub-objetivo.
Este processo é sucessivamente repetido até que não existam mais sub-objetivos ou quando o sub-objetivo escolhido não for unificável com a cabeça de nenhuma das cláusulas do programa.


$\leftarrow \underline{Ant(Sr.B, Bart)}$
$\underline{Ant(x,z)} \leftarrow Ant(x,y), AD(y,z)$
$\{Sr.B/x, Bart/z\}$
$\leftarrow Ant(Sr.B, y), \underline{AD(y, Bart)}$
$\underline{AD(Marge, Bart)} \leftarrow$
$\{Marge/y\}$
$\leftarrow \underline{Ant(Sr.B, Marge)}$
$\underline{Ant(x,y)} \leftarrow AD(x,y)$
$\{Sr.B/x, Marge/y\}$
$\leftarrow \underline{AD(Sr.B, Marge)}$
$\underline{AD(Sr.B, Marge)} \leftarrow$
ε
$\square$

Definição 6.3.3 (Resposta calculada)

Seja Δ um programa, α um objetivo e S uma função de seleção. Se a prova SLD para α usando Δ for finita, $[\gamma_0, \gamma_1, \ldots, \gamma_n]$, a composição das substituições s_0, s_1, $\ldots$, s_{n-1} restringida às variáveis que ocorrem em α, $(s_0 \circ s_1 \circ \ldots \circ s_{n-1})\,|_{vars(\alpha)}$, diz-se uma *resposta calculada de* Δ *a* α *via* S. Diz-se também que n é o *comprimento* da prova SLD.

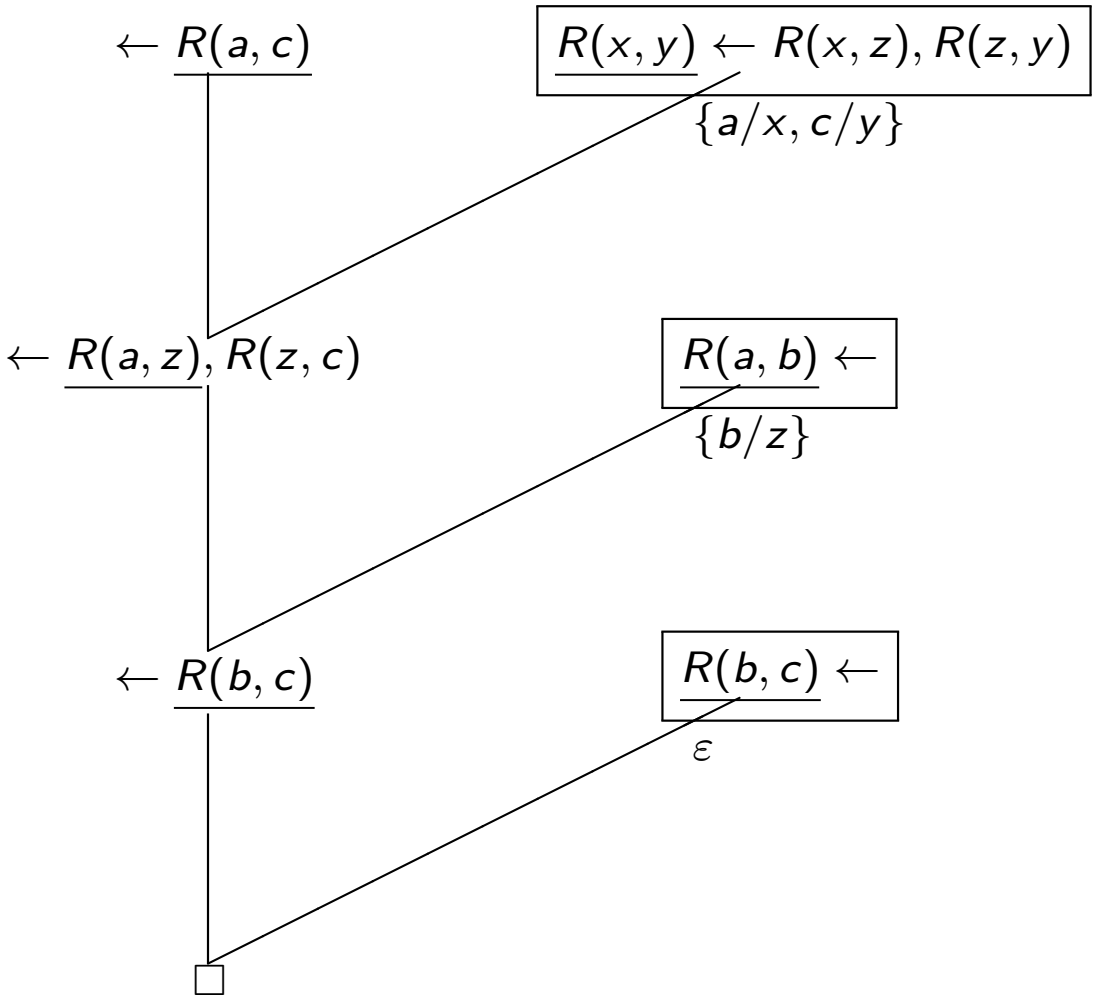
Definição 6.3.4 (Refutação SLD)

Uma prova SLD diz-se uma *refutação SLD* se e só o seu último elemento for a cláusula vazia, $\square$.

Exemplo: Consideremos o seguinte programa:
$R(a, b) \leftarrow$
$R(b, c) \leftarrow$
$R(x, y) \leftarrow R(x, z), R(z, y)$

Suponhamos que a função de seleção escolhia o primeiro literal no objetivo, e que a regra de procura escolhia a cláusula com menos literais.

Prova por refutação para o objetivo $\leftarrow R(a, c)$:

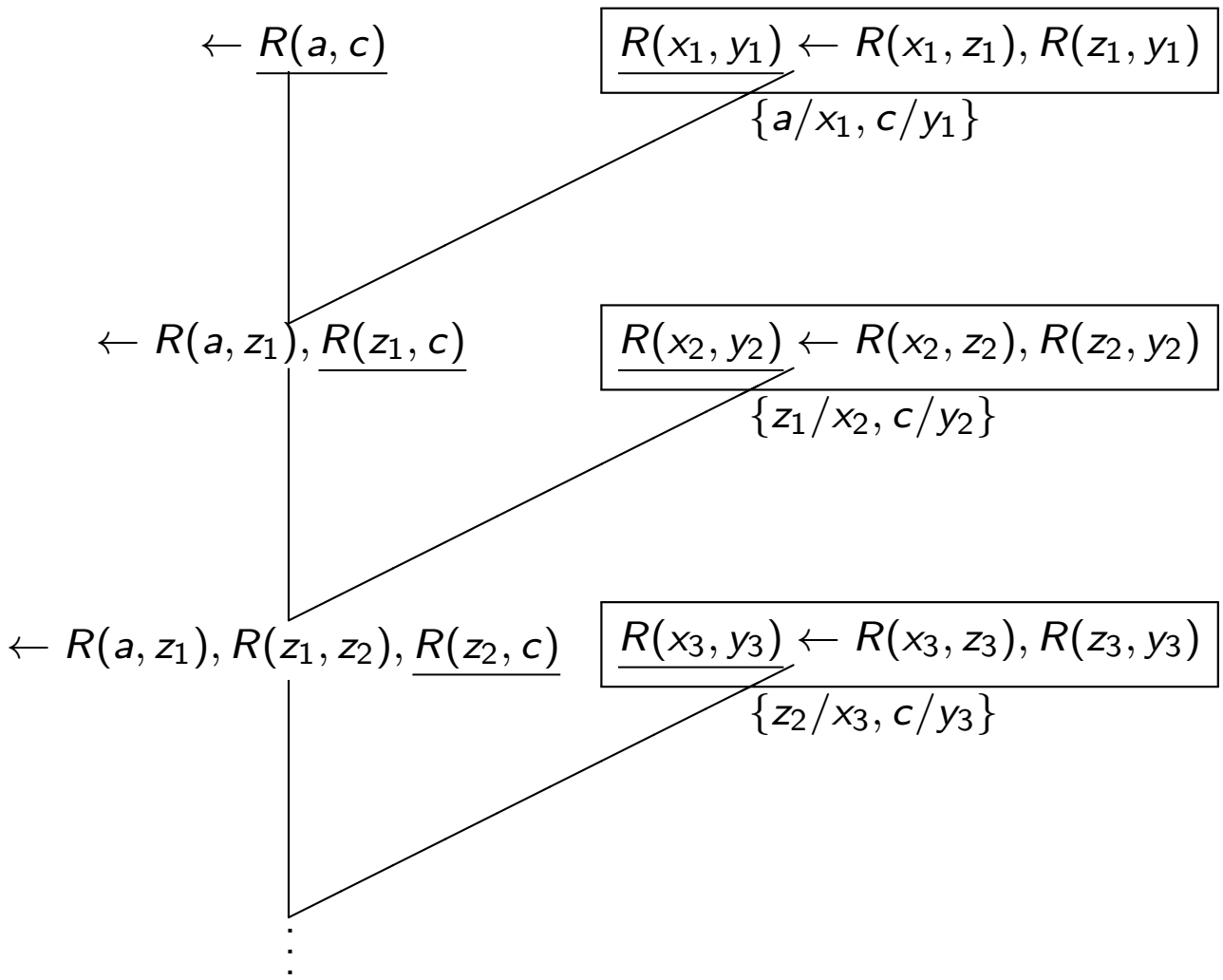


Resposta calculada: $(\{a/x, c/y\} \circ \{b/z\} \circ \varepsilon)\,|_{\{\}} = \varepsilon$.

Exemplo: Consideremos o seguinte programa:
$R(a, b) \leftarrow$
$R(b, c) \leftarrow$
$R(x, y) \leftarrow R(x, z), R(z, y)$

Suponhamos agora que a função de seleção escolhia o último literal no objetivo, e que a regra de procura escolhia a cláusula com mais literais.

Prova por refutação para o objetivo $\leftarrow R(a, c)$:



Este exemplo mostra a importância da escolha da cláusula definida (regra de procura) para aplicar o princípio da resolução.

6.4 Árvores SLD

A mesma função de selecção pode originar várias refutações SLD, dependendo da cláusula escolhida.

Uma árvore SLD mostra todas as alternativas para uma dada função de selecção.

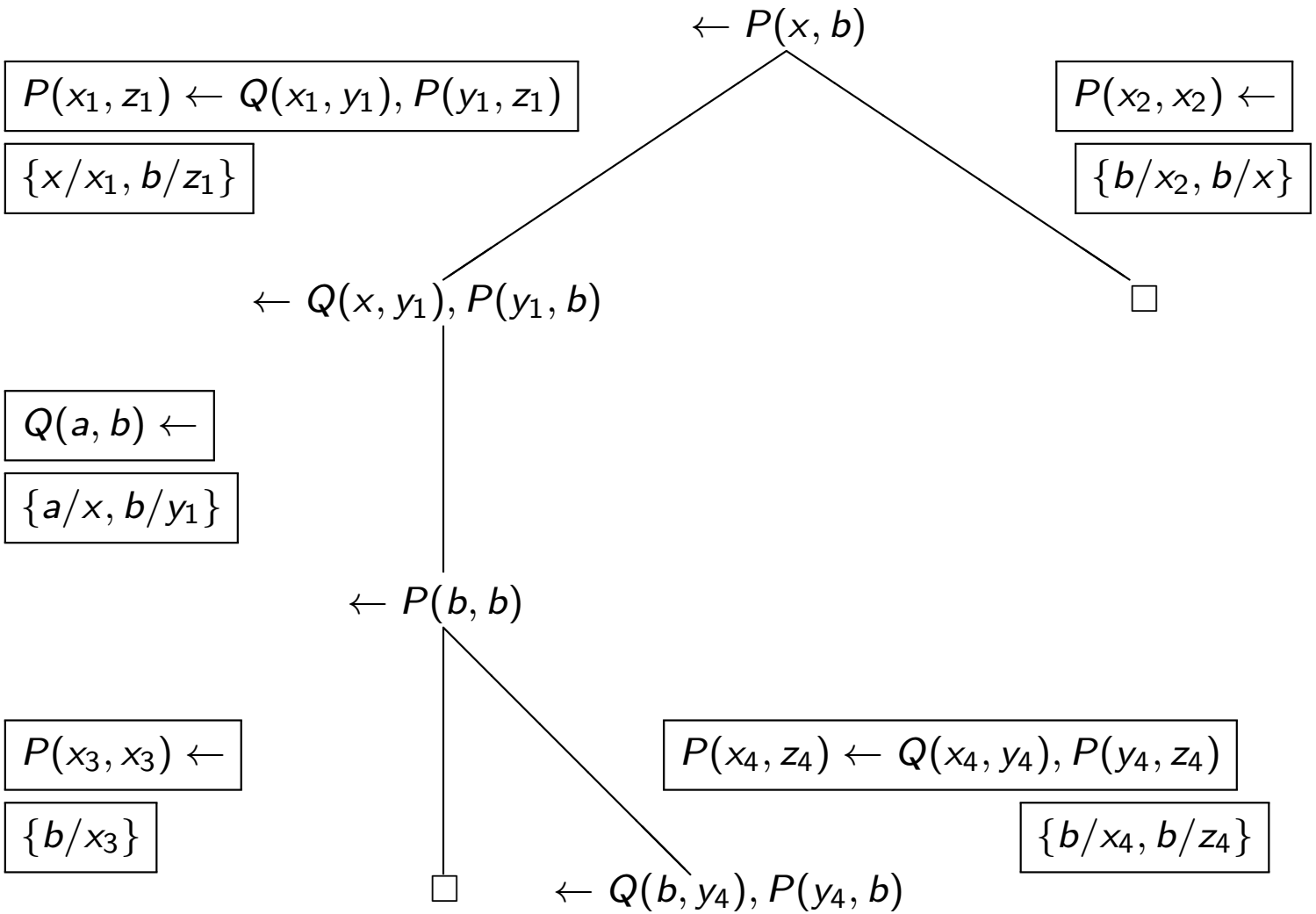
6.4 Árvores SLD

Consideremos o programa
$P(x, z) \leftarrow Q(x, y), P(y, z)$
$P(x, x) \leftarrow$
$Q(a, b) \leftarrow$

e o objetivo
$\leftarrow P(x, b)$.

Vamos usar a função de seleção $S(\leftarrow \alpha_1, \ldots, \alpha_n) = \alpha_1$.

6.4 Árvores SLD



6.4 Árvores SLD

Obtivemos:

- 2 ramos bem sucedidos, com respostas $\{a/x\}$ e $\{b/x\}$.
- 1 nó falhado.

Com a função de seleção $S(\leftarrow \alpha_1, \ldots, \alpha_n) = \alpha_n$, obteríamos:

- 2 ramos bem sucedidos, com respostas $\{a/x\}$ e $\{b/x\}$.
- infinitos nó falhados.
- ramos infinitos.

Teorema (Independência da função de seleção)

Seja Δ um programa e α um objetivo. Então, independentemente da função de seleção, todas as árvores SLD de Δ e α têm o mesmo número (finito ou infinito) de ramos bem sucedidos.