7. PROLOG: PROGRAMAÇÃO EM LÓGICA EM PRÁTICA

Eficiência implica a existência de alguns compromissos.

PROGRAMA = LÓGICA + CONTROLO

7.1 Componentes básicos

7.1.1Termos

Tal como em LPO, um termo pode ser:

- Uma constante.
- Uma variável.
- A aplicação de uma função a termos.

Constantes

Exemplos:
`nemo, 'nemo', moby_dick, 'Moby Dick', 123`

`nemo` e `'nemo'` representam a mesma constante.

Variáveis

Exemplos: `X, _x, X_mais_1, _`
Variável anónima: `_`. O seu valor não interessa.
Exemplos:
Objectivo: `<- peixe(X)` Resposta: `X = nemo`
Objectivo: `<- peixe(_)` Resposta: `true`
Objectivo: `<- mae(_,_)` Resposta: `true`

Termos compostos

Aplicação de uma letra de função (functor) a termos:

Exemplos:

```
mae(bart), mae(X),mae(mae(bart)),
+(X,5), X + 5, f(a),f(a,b)
```

Aplicação de uma letra de predicado (átomo) a termos:
Exemplos:
`mae(marge,bart), ad(X,bart)`

Sequência de cláusulas determinadas.

Exemplo:

```
ad(marge, bart).
ad(srB, marge).
ant(X,Y) :- ad(X,Y).
ant(X,Y) :- ad(X, Z), ant(Z, Y).
```

Objectivo: `?- <literais>.`

`?-` é a "prompt" do PROLOG.

Exemplos:

```
?- ant(X,bart).
?- ant(X,bart), ant(X,marge).
```

7.2 Unificação de termos

Operadores de unificação: `=` e `\=`

Exemplos:

```
?- X = a.
 X = a.

 ?- X + 5 = +(8,5).
 X = 8.
```

7.2 Unificação de termos

```
?- X + 5 = +(5,8).
false.

?- f(a,X) = f(Y,b).
X = b,
Y = a.

?- f(a,_) = f(Y,b).
Y = a.

?- f(X,X) = f(a,b).
false.

?- f(_,_) = f(a,b).
true.
```

7.3 Comparação de termos

Operadores de comparação: == e \==

Exemplos:

```
?- b == a.
false.

?- 'a'  == a.
true.

?- X == a.
false.

?- X = a, X == a.
X = a.
```

7.5 A semântica do PROLOG

Para provar um objectivo, PROLOG usa uma refutação SLD com:

Função de selecção: escolhe o primeiro literal do objectivo.

Regra de procura: escolhe a primeira cláusula, cuja cabeça é unificável com o literal escolhido.

7.5 A semântica do PROLOG

Semântica declarativa

Diz unicamente respeito ao significado das cláusulas do programa, vistas como fórmulas da lógica.

Exemplo:

```
p(X) :- q(X), r(X).
```

significa

para provar `p(X)`,

provar `q(X)` e provar `r(X)`, para certo valor de `X`.

7.5 A semântica do PROLOG

Semântica procedimental

Diz também respeito ao modo de utilizar as regras do programa.

Exemplo:

```
p(X) :- q(X), r(X).
```

significa

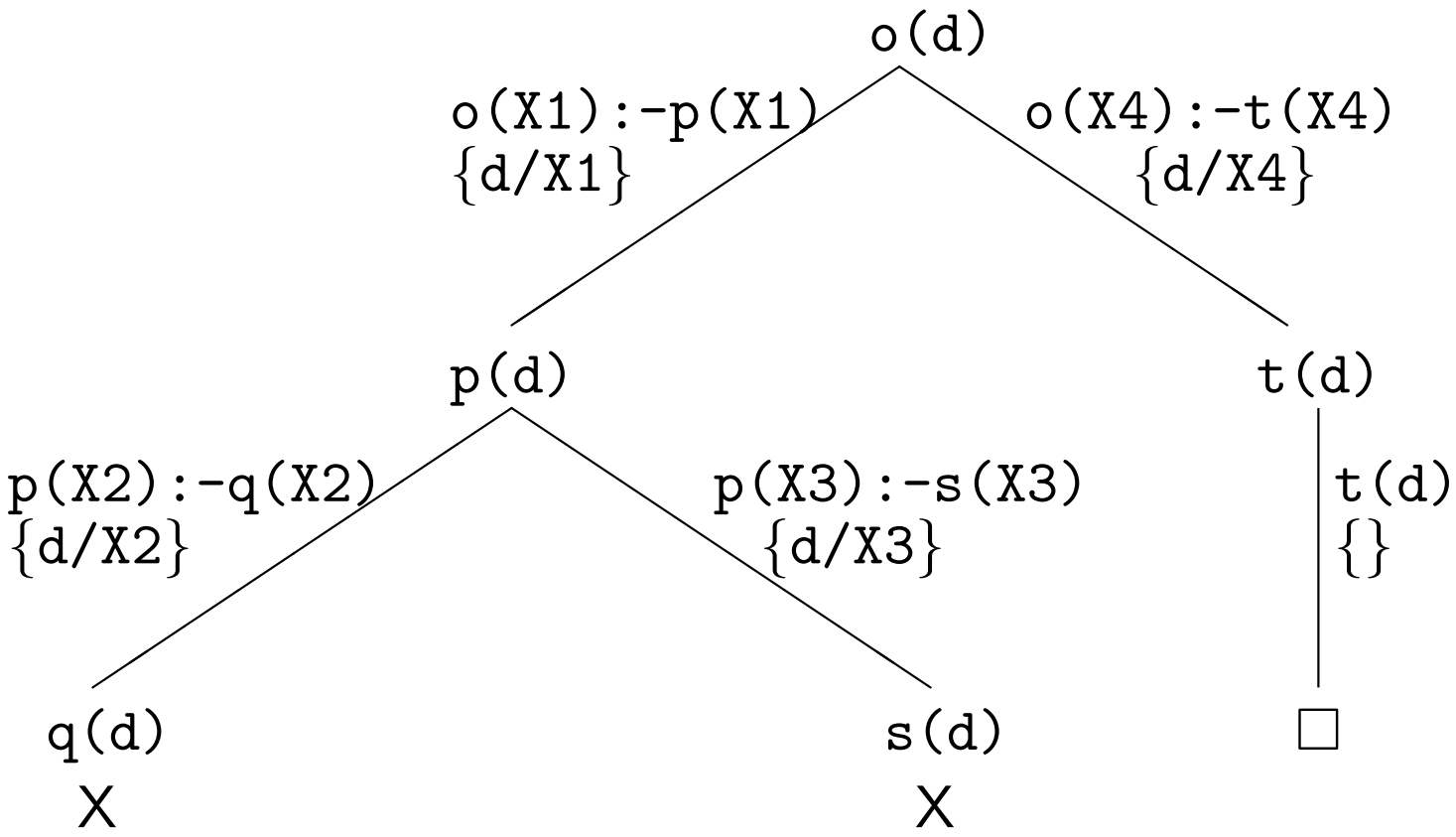
para provar `p(X)`,

provar PRIMEIRO `q(X)` e DEPOIS provar `r(X)`, para certo valor de `X`.

7.5 A semântica do PROLOG

Programa:

```
o(X):-p(X).
o(X):-t(X).
p(X):-q(X).
p(X):-s(X).
q(a).
s(b).
t(d).
```



Exemplo 7.6.4:

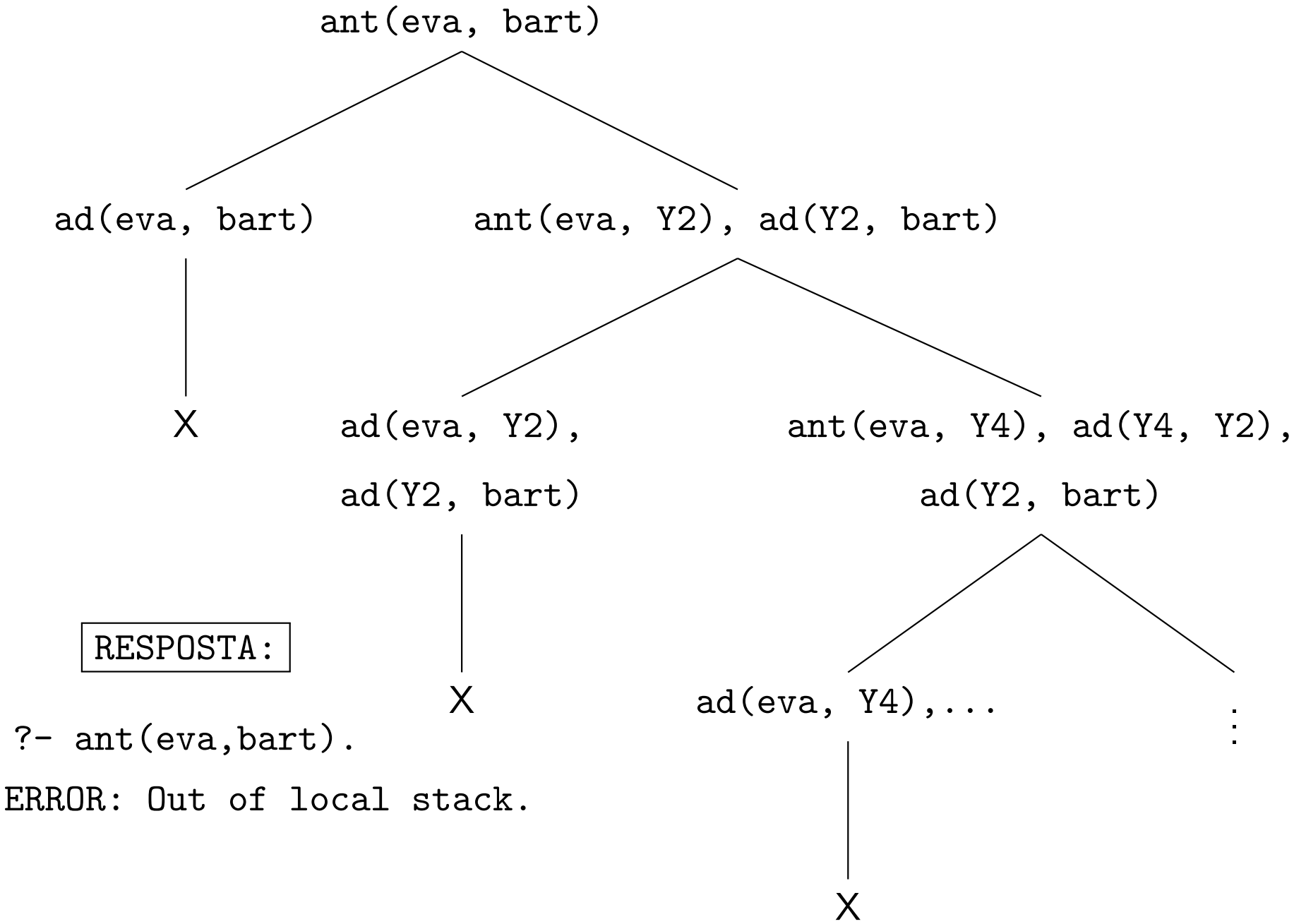
Programa:

```
ad(marge, bart).
ad(srB, marge).

ant(X, Y) :- ad(X, Y).
ant(X, Z) :- ant(X, Y), ad(Y, Z).
```

Objectivo:

```
?- ant(eva, bart).
```



RESPOSTA:

```
?- ant(eva,bart).
ERROR: Out of local stack.
```

7.6 Exemplos iniciais

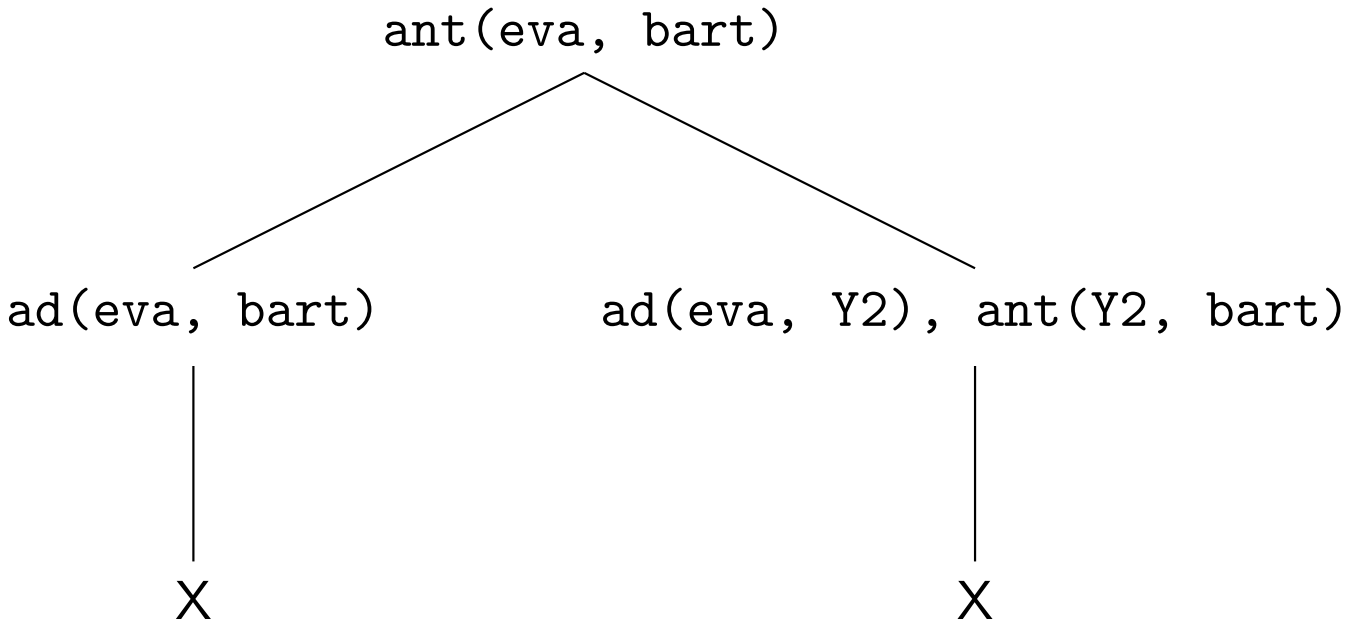
Exemplo 7.6.4:

Programa:

```
ad(marge, bart).
ad(srB, marge).
ant(X, Y) :- ad(X, Y).
ant(X, Z) :- ad(X, Y), ant(Y, Z).
```

Objectivo:

```
?- ant(eva, bart).
```



RESPOSTA:

```
?- ant(eva,bart).
false.
```

7.6 Exemplos iniciais

Resposta `false` do PROLOG deve ser interpretada como "não sei".

Hipótese do mundo fechado:

Assume que tudo o que não é derivável a partir de um programa é falso.

Regras empíricas:

- Na definição de um predicado, as afirmações devem aparecer antes das regras.
- Devemos evitar recursão à esquerda (recursão no 1º literal do corpo). Por exemplo, em vez de

```
ant(X,Z) :- ant(X,Y), ad(Y,Z).
```

devemos escrever

```
ant(X,Z) :- ad(Y,Z), ant(X,Y).
```

7.6 Exemplos iniciais

Múltiplas respostas

Podemos pedir múltiplas respostas usando ;.

Exemplo:

Programa:

```
p(a).
p(b).
q(a).
q(b).
r(X) :- p(X), q(X).
```

Objectivo:

```
?- r(X).
X = a;
X = b.
```

7.6 Exemplos iniciais