Representação do Conhecimento e Raciocínio
Programação Lógica e Algoritmos de Inferência

Inteligência Artificial - 2020/1

Programas Prolog e Algoritmos de Inferência

- Semelhanças e diferenças entre sentenças Prolog e sentenças da lógica

- Programas Prolog são conjuntos de sentenças na forma de cláusulas definidas escritas em uma notação diferente daquela da Lógica de Predicados

- Prolog usa letras maiúsculas para variáveis e minúsculas para constantes e predicados, que é o oposto da convenção adotada majoritariamente para a lógica.

- Conjunções são separadas por virgulas e a cláusula é escrita "de trás para frente":
  - Em vez de escrever $A \wedge B \rightarrow C$
  - Escrevemos C :- A, B.

Programas Prolog e Algoritmos de Inferência

- Execução de programas em Prolog

- A execução de programas Prolog é feita por encadeamento para trás com busca em profundidade em que as cláusulas são tentadas na ordem em que foram escritas na base de conhecimento.

- Prolog busca um compromisso entre aspectos declarativos da lógica e eficiência de execução, necessária para uma linguagem de programação.

Programas Prolog e Algoritmos de Inferência

- Principais características do Prolog que diferem da inferência padrão da lógica de predicados:

- Prolog usa a semântica de base de dados e isso fica aparente na forma em que trata igualdade e negação:
- Suposição de unicidade de nomes – Cada símbolo de constante se refere a um objeto distinto;
- Suposição do mundo fechado – sentenças atômicas que não são verdadeiras (não aparecem na base de conhecimento) ou não são consequência lógica da base de conhecimento, são consideradas falsas;
- Suposição de domínio fechado – considera que o modelo não contem objetos além dos que são nomeados na base de conhecimento.

Programas Prolog e Algoritmos de Inferência

- Exemplos:

Se a base de conhecimento contém:

```
pai_de(joao, pedro).
pai_de(joao, paulo).
```

A consulta abaixo dá resposta falsa:

```
?- pai_de(joao, jose).
false.
```

A consulta abaixo dá resposta verdadeira:

```
?- not(pai_de(joao,jose)).
true
```

Programas Prolog e Algoritmos de Inferência

- Existe um conjunto de funções aritméticas pré-definidas. Os literais que usam essas funções são provados executando código e não fazendo inferências adicionais.

```
?- X is 3+5.
X = 8.
```

- Existem predicados que provocam efeitos colaterais quando são executados como, por exemplo, os predicados de entrada e saída, o corte e outros predicados que alteram a base de conhecimento.

```
?- read(X), read(Y), Z is X - Y.
|: 10.
|: 3.

X = 10,
Y = 3,
Z = 7.
```

Programas Prolog e Algoritmos de Inferência

- Prolog não faz verificação de recursão infinita;
- é rápido quando o conjunto de cláusulas está na ordem correta e
- é incompleto quando o conjunto de sentenças está na ordem "errada"

- Considere o programa que verifica se existe um caminho entre nós de um grafo:

```
caminho(X,Y) :- arco(X,Y).
caminho(X,Y) :- arco(X,Z), caminho(Z,Y).
arco(a,b).
arco(b,c).
```

- A consulta:

```
?- caminho(a,c).
true ;
false.  (Resposta correta e para, embora tente encontrar outra solução)
```

Programas Prolog e Algoritmos de Inferência

- Considere o mesmo programa com as premissas da regra recursiva na ordem trocada (predicado recursivo aparece antes):

```
caminho(X,Y) :- arco(X,Y).
caminho(X,Y) :- caminho(X,Z), arco(Z,Y).
arco(a,b).
arco(b,c).
```

- A consulta:

```
?- caminho(a,c).
true;
Erro
Erro..... (Resposta correta, mas quando tenta encontrar outra solução dá
                erro-estouro de pilha)
```

Programas Prolog e Algoritmos de Inferência

- Considere agora o mesmo programa com as sentenças na ordem trocada (regra recursiva aparece antes):

```
caminho(X,Y) :- arco(X,Z), caminho(Z,Y).
caminho(X,Y) :- arco(X,Y).
arco(a,b).
arco(b,c).
```

- A consulta:

```
?- caminho(a,c).
Erro
Erro
Erro.....  (Não encontra a resposta, só erros)
```

Programas Prolog e Algoritmos de Inferência

Exemplo – Extraído de Russell& Norvig, Inteligência Artificial, 2ª. Ed., capítulo 9, seções 9.2 e 9.3

- A lei diz que é crime um americano vender armas a nações hostis. O país Nono, inimigo da América, tem alguns mísseis, e todos foram vendidos pelo Coronel West, um americano.
- West é criminoso?
- Representar os fatos como clausulas definidas de primeira ordem

Programas Prolog e Algoritmos de Inferência

Base de Conhecimento

1. $\text{Americano}(x) \land \text{Arma}(y) \land \text{Vende}(x,y,z) \land \text{Hostil}(z) \rightarrow \text{Criminoso}(x)$
2. $\text{Possui}(\text{Nono}, \text{M1})$
3. $\text{Míssil}(\text{M1})$
4. $\text{Míssil}(x) \land \text{Possui}(\text{Nono}, x) \rightarrow \text{Vende}(\text{West}, x, \text{Nono})$
5. $\text{Míssil}(x) \rightarrow \text{Arma}(x)$
6. $\text{Inimigo}(x, \text{América}) \rightarrow \text{Hostil}(x)$
7. $\text{Americano}(\text{West})$
8. $\text{Inimigo}(\text{Nono}, \text{América})$

Esta base de conhecimento não tem nenhum símbolo de função (base de conhecimento Datalog).
A inferência é mais simples sem símbolos de função.

Programas Prolog e Algoritmos de Inferência

1. Americano(x) $\wedge$ Arma(y) $\wedge$ Vende(x,y,z) $\wedge$ Hostil(z) $\rightarrow$ Criminoso(x)
2. Possui(Nono, M1)
3. Míssil(M1)
4. Míssil(x) $\wedge$ Possui(Nono, x) $\rightarrow$ Vende(West, x, Nono)
5. Míssil(x) $\rightarrow$ Arma(x)
6. Inimigo(x,América) $\rightarrow$ Hostil(x)
7. Americano(West)
8. Inimigo(Nono,América)

%programa Prolog

```
criminoso(X):- americano(X), arma(Y), vende(X,Y,Z), hostil(Z).
vende(west, X, nono) :- míssil(X), possui(nono, X).
arma(X) :- missil(X).
hostil(X) :- inimigo(X, america).
possui(nono, m1).
missil(m1).
americano(west).
inimigo(nono,america).
```

Programas Prolog e Algoritmos de Inferência

1. Americano(x) $\wedge$ Arma(y) $\wedge$ Vende(x,y,z) $\wedge$ Hostil(z) $\rightarrow$ Criminoso(x)
2. Possui(Nono, M1)
3. Míssil(M1)
4. Míssil(x) $\wedge$ Possui(Nono, x) $\rightarrow$ Vende(West, x, Nono)
5. Míssil(x) $\rightarrow$ Arma(x)
6. Inimigo(x,América) $\rightarrow$ Hostil(x)
7. Americano(West)
8. Inimigo(Nono,América)

%programa Prolog

```
criminoso(X):- americano(X), arma(Y), vende(X,Y,Z), hostil(Z).
vende(west, X, nono) :- míssil(X), possui(nono, X).
arma(X) :- missil(X).
hostil(X) :- inimigo(X, america).
possui(nono, m1).
missil(m1).
americano(west).
inimigo(nono,america).
```

Consulta:

```
?-criminoso(west).
true.
```

Programas Prolog e Algoritmos de Inferência

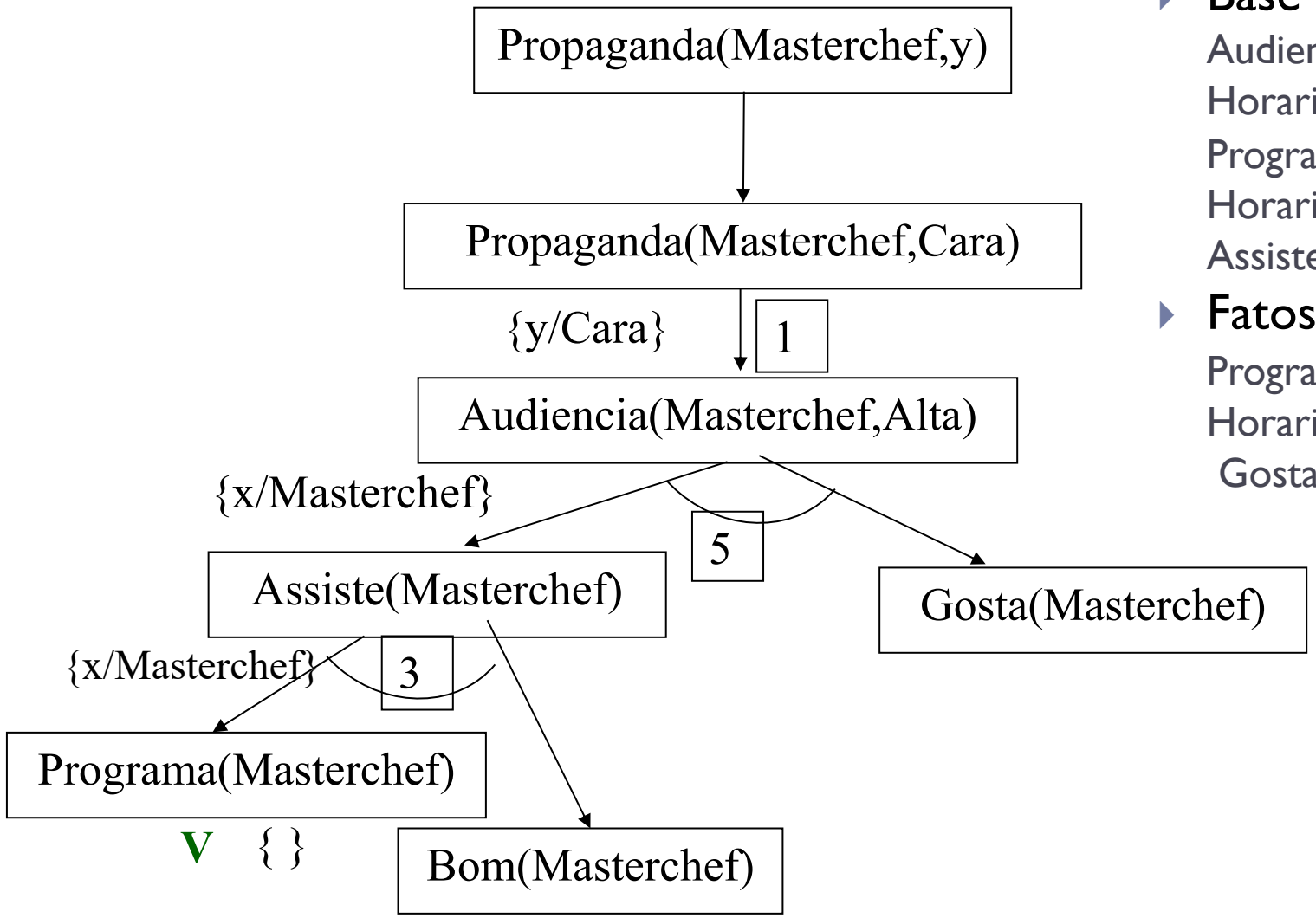
Cuidado com predicados não definidos!

- Base de Conhecimento (em sentenças da lógica):
  - Audiencia(x, Alta) $\rightarrow$ Propaganda(x, Cara)
  - Horario(x, Madrugada) $\rightarrow$ Propaganda(x, Barata)
  - Programa(x) $\wedge$ Bom(x) $\rightarrow$ Assiste(x)
  - Horario(x, Nobre) $\rightarrow$ Assiste(x)
  - Assiste(x) $\wedge$ Gosta(x) $\rightarrow$ Audiencia(x, Alta)
- Fatos:
  - Programa(Masterchef)
  - Horario(Masterchef, Nobre)
  - Gosta(Masterchef)

Consulta:

Propaganda(Masterchef, y).

Programas Prolog e Algoritmos de Inferência



- Base de Conhecimento:

  Audiencia(x, Alta) → Propaganda(x, Cara)
  Horario(x, Madrugada) → Propaganda(x, Barata)
  Programa(x) ∧ Bom(x) → Assiste(x)
  Horario(x, Nobre) → Assiste(x)
  Assiste(x) ∧ Gosta(x) → Audiencia(x, Alta)

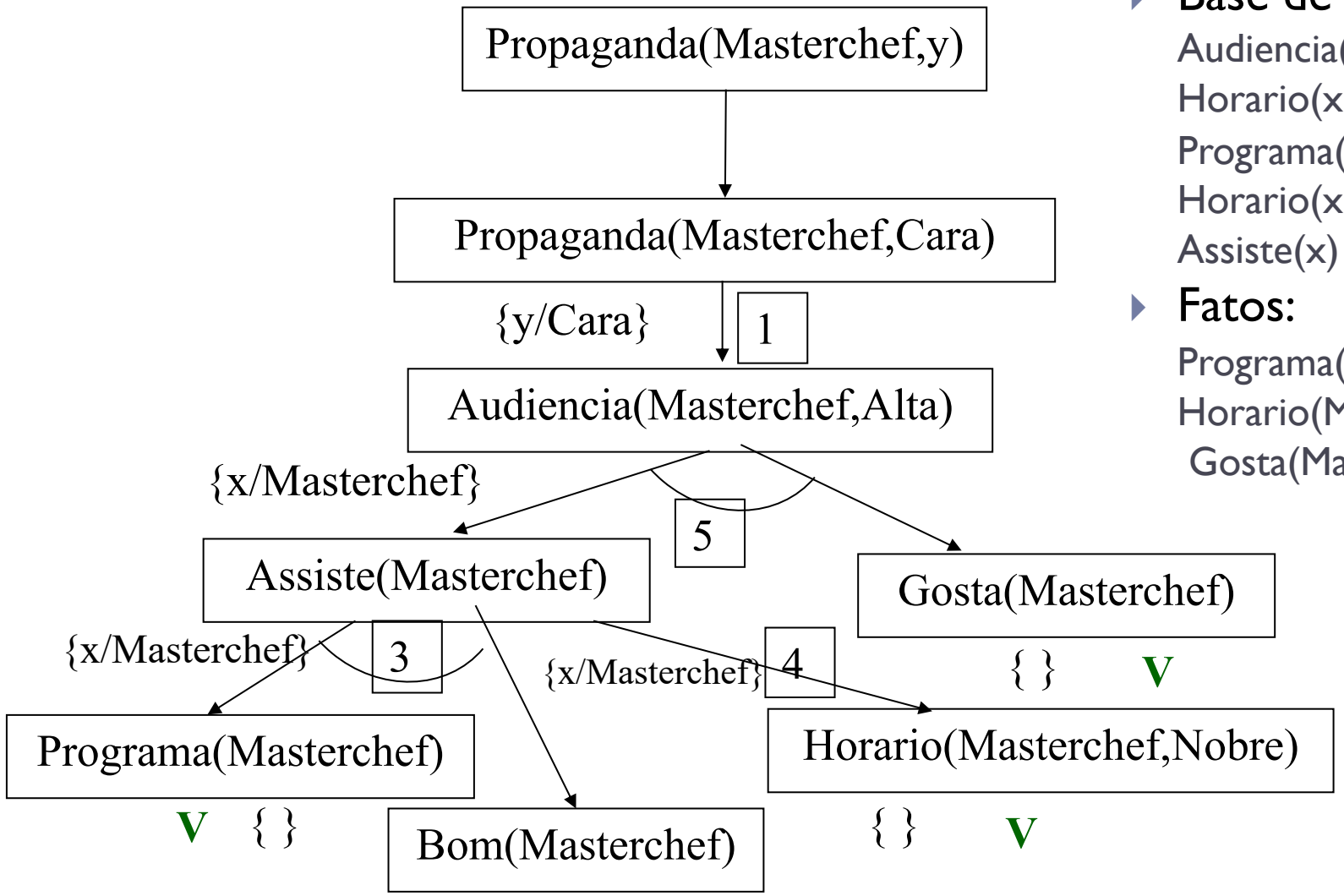
- Fatos:

  Programa(Masterchef)
  Horario(Masterchef, Nobre)
  Gosta(Masterchef)

Consulta:

Propaganda(Masterchef, y).

Programas Prolog e Algoritmos de Inferência



- Base de Conhecimento:

  Audiencia(x, Alta) → Propaganda(x, Cara)
  Horario(x, Madrugada) → Propaganda(x, Barata)
  Programa(x) ∧ Bom(x) → Assiste(x)
  Horario(x, Nobre) → Assiste(x)
  Assiste(x) ∧ Gosta(x) → Audiencia(x, Alta)

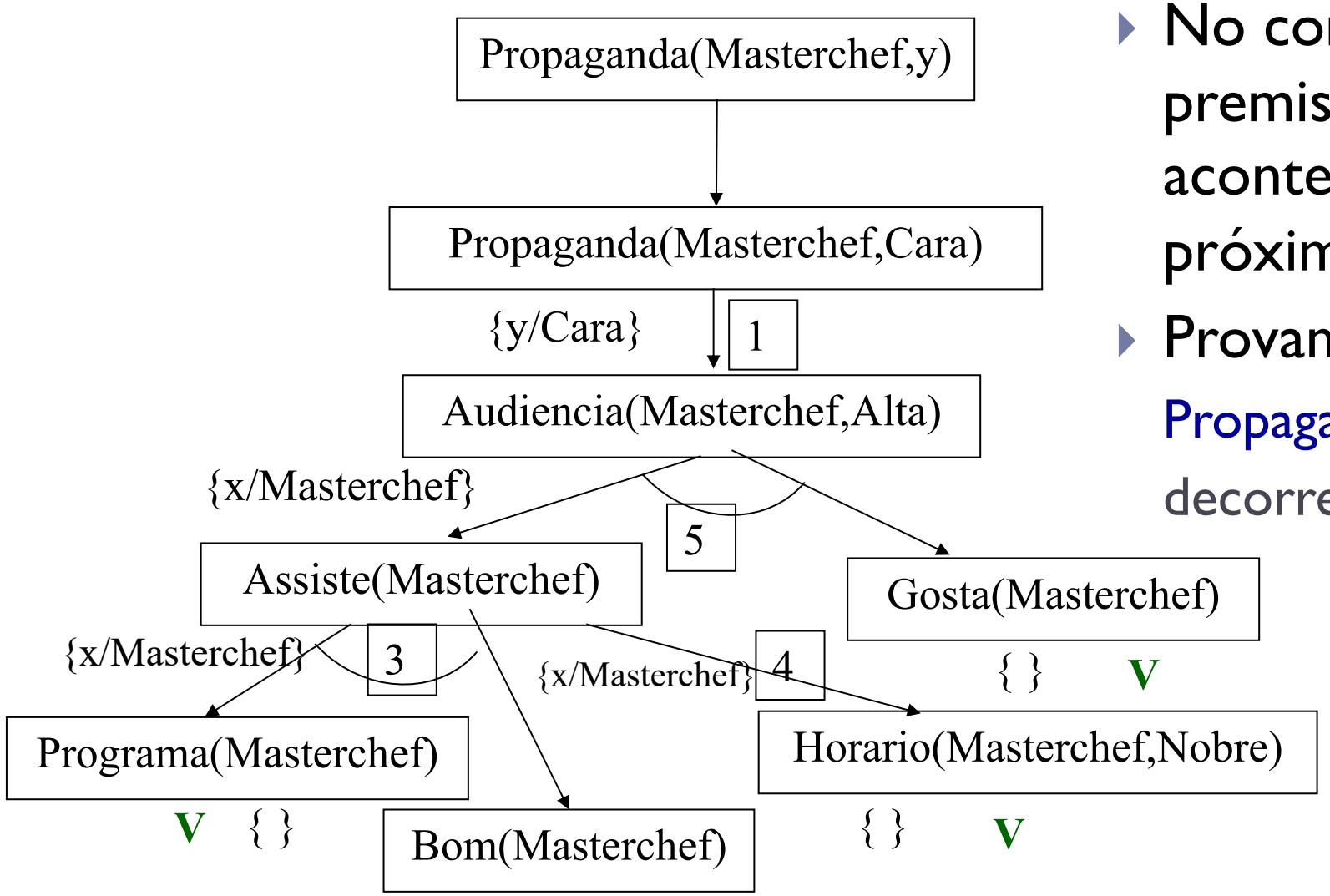
- Fatos:

  Programa(Masterchef)
  Horario(Masterchef, Nobre)
  Gosta(Masterchef)

Consulta:

Propaganda(Masterchef, y).

Programas Prolog e Algoritmos de Inferência



- No contexto da lógica, quando a premissa Bom(Masterchef) falha, acontece o retrocesso e a próxima regra é tentada.
- Provamos que:
  Propaganda(Masterchef, Cara)
  decorre logicamente da BC

Consulta:

Propaganda(Masterchef, y).

Propaganda(Masterchef, Cara)

Programas Prolog e Algoritmos de Inferência

Cuidado com predicados não definidos!

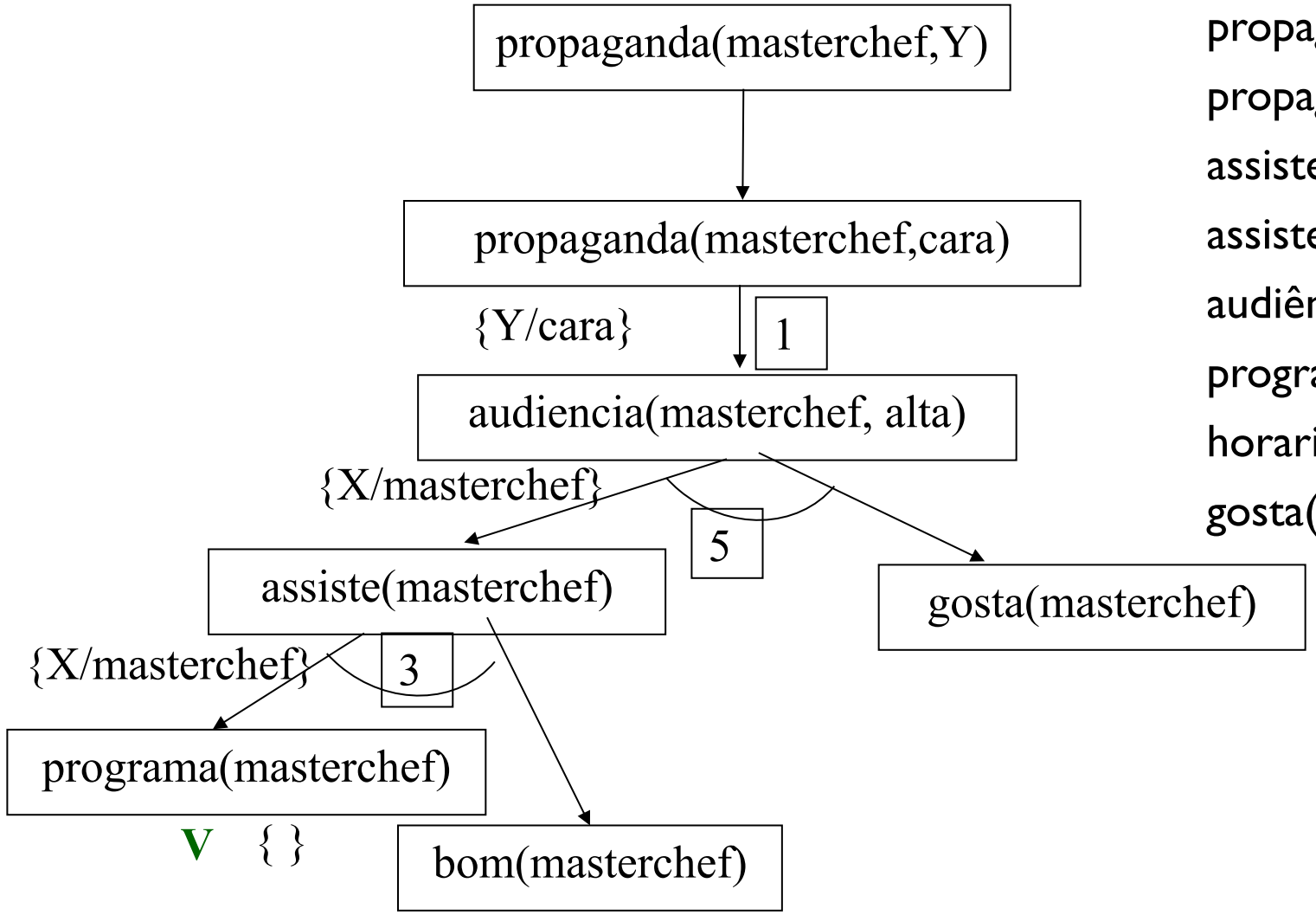
Base de Conhecimento (em Prolog):

```
propaganda(X, cara) :- audiencia(X, alta).
propaganda(X, barata) :- horario(X, madrugada).
assiste(X):- programa(X), bom(X).
assiste(X) :- horario(X, nobre).
audiencia(X, alta) :- assiste(X), gosta(X).
programa(masterchef).
horario(masterchef, nobre).
gosta(masterchef).
```

Consulta:

```
?-propaganda(masterchef,X).
```

Programas Prolog e Algoritmos de Inferência



propaganda(X,cara) :- audiencia(X, alta).

propaganda(X, barata) :- horário(X, madrugada).

assiste(X):- programa(X), bom(X).

assiste(X) :- horario(X, nobre).

audiência(X, alta) :- assiste(X), gosta(X).

programa(masterchef).

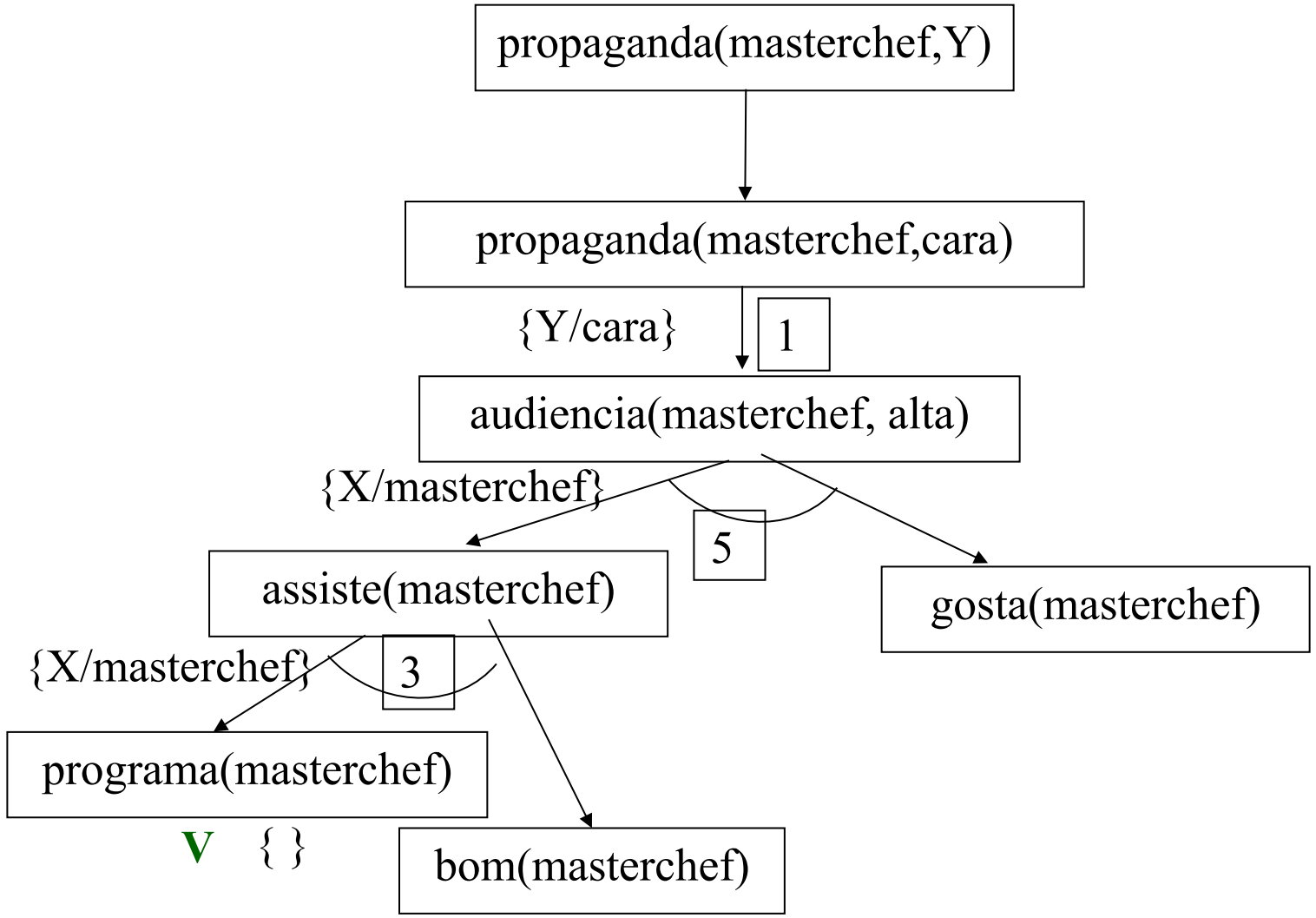
horario(masterchef, nobre).

gosta(masterchef).

Consulta:

?-propaganda(masterchef,X).

Programas Prolog e Algoritmos de Inferência



bom(X): predicado não definido!

Resposta: erro

Consulta:

?-propaganda(masterchef,X).

ERRO! Predicado não definido

Programas Prolog e Algoritmos de Inferência

Solução: Acrescentar uma definição para o predicado bom

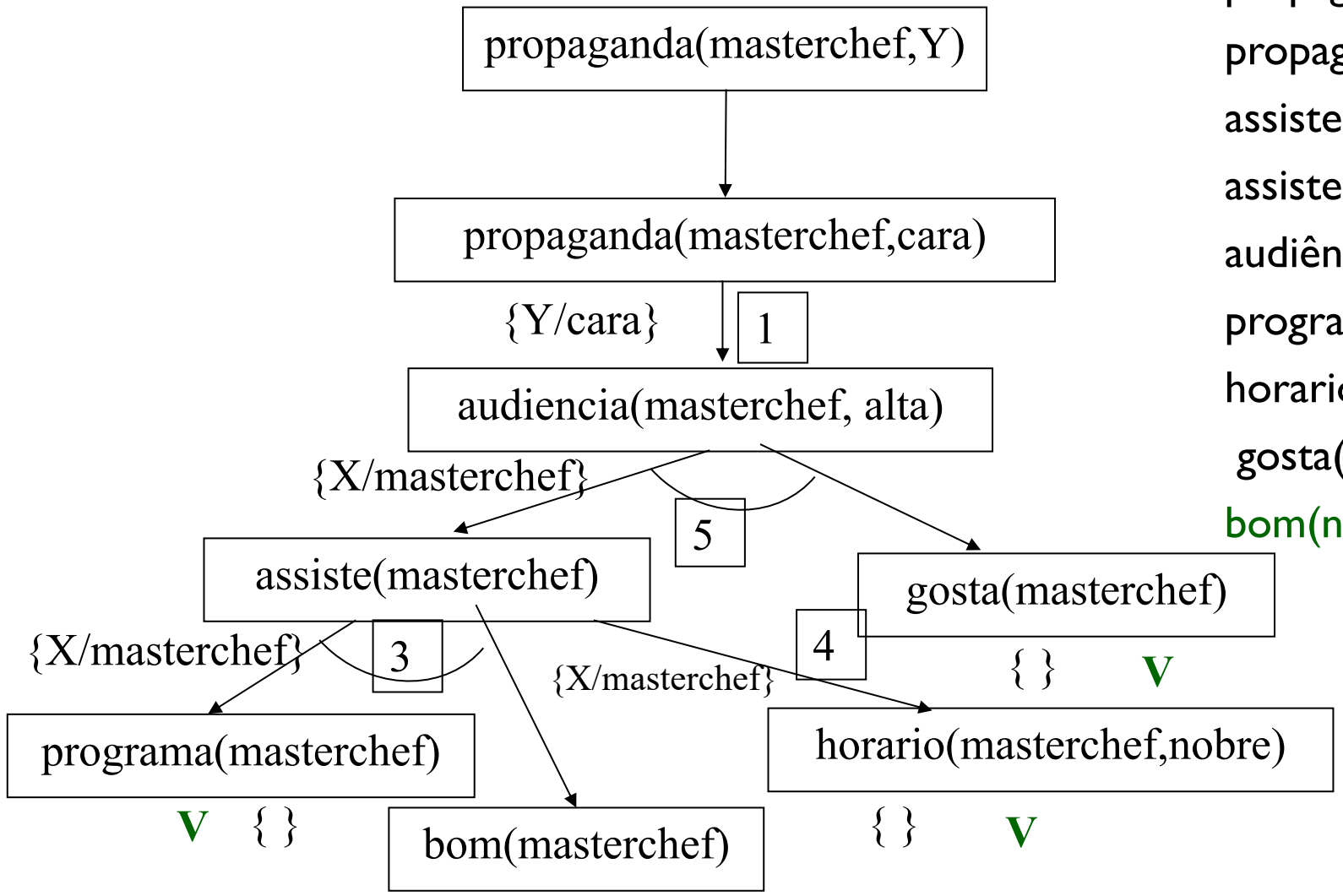
Base de Conhecimento (em Prolog):

```
propaganda(X,cara) :- audiencia(X, alta).
propaganda(X, barata) :- horario(X, madrugada).
assiste(X):- programa(X), bom(X).
assiste(X) :- horario(X, nobre).
audiencia(X, alta) :- assiste(X), gosta(X).
programa(masterchef).
horario(masterchef, nobre).
gosta(masterchef).
bom(nenhum).
```

Consulta:

```
?-propaganda(masterchef,X).
```

Programas Prolog e Algoritmos de Inferência



propaganda(X,cara) :- audiencia(X, alta).

propaganda(X, barata) :- horário(X, madrugada).

assiste(X):- programa(X), bom(X).

assiste(X) :- horario(X, nobre).

audiência(X, alta) :- assiste(X), gosta(X).

programa(masterchef).

horario(masterchef, nobre).

gosta(masterchef).

bom(nenhum).

Consulta:

?-propaganda(masterchef,X).
X = cara.

- Fim do Tópico

- Programação Lógica e Algoritmos de Inferência