MAC0444 — Sistemas Baseados em Conhecimento

Departamento de Ciência da Computação

Prof.ª Renata Wassermann **Lista 2** 30 de setembro de 2019

Aluno Vitor Santa Rosa Gomes, 10258862, vitorssrg@usp.br
Curso Bacharelado em Ciência da Computação, IME-USP

1. Formalize as seguintes sentenças usando os predicados unários `fezEx`, `vaiBem`, `mediaAlta` e o predicado binário `aprovado`:

"Quem fez os exercícios vai bem na prova."

"Quem vai bem na prova fica com média alta."

"Quem fica com média alta é aprovado em mac444."

"João fez os exercícios."

"Maria foi bem na prova."

Use **resolução SLD** para mostrar que João e Maria foram aprovados em mac444.

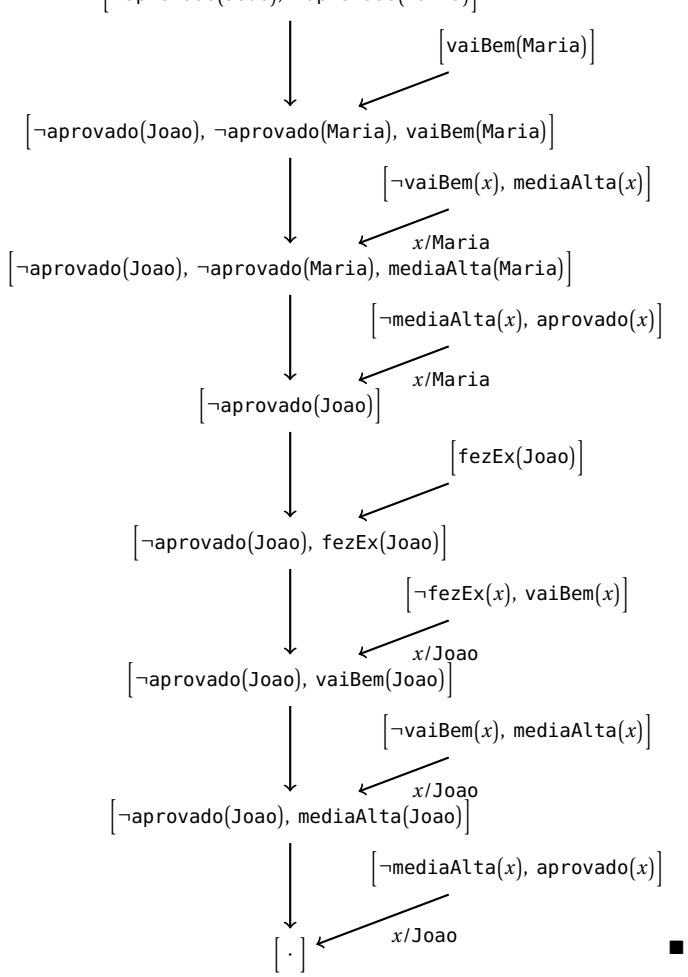
Formalização:

1. $\forall x(\texttt{fezEx}(x) \rightarrow \texttt{vaiBem}(x))$
2. $\forall x(\texttt{vaiBem}(x) \rightarrow \texttt{mediaAlta}(x))$
3. $\forall x(\texttt{mediaAlta}(x) \rightarrow \texttt{aprovado}(x))$
4. $\texttt{fezEx}(\texttt{Joao})$
5. $\texttt{vaiBem}(\texttt{Maria})$

Em horn:

1. $\left[\neg\texttt{fezEx}(x),\ \texttt{vaiBem}(x)\right]$
2. $\left[\neg\texttt{vaiBem}(x),\ \texttt{mediaAlta}(x)\right]$
3. $\left[\neg\texttt{mediaAlta}(x),\ \texttt{aprovado}(x)\right]$
4. $\left[\texttt{fezEx}(\texttt{Joao})\right]$
5. $\left[\texttt{vaiBem}(\texttt{Maria})\right]$

Adicionando a pergunta $\texttt{aprovado}(\texttt{Joao}) \wedge \texttt{aprovado}(\texttt{Maria})$, usa-se resolução SLD com extração de resposta, para mostrar que tanto João quanto Maria foram aprovados.

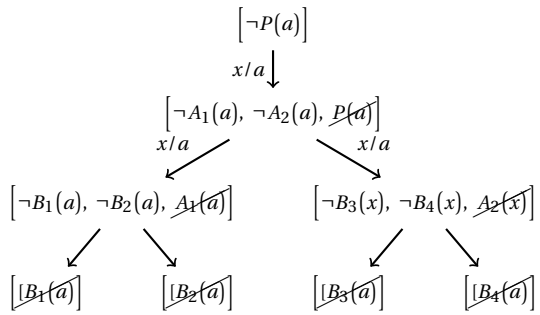


■

2. **(a)** Mostre passo a passo que o procedimento de **encadeamento para trás (*backward chaining*)** produz resposta SIM com objetivo $P(a)$ e uma base de conhecimento formada pelas seguintes cláusulas:

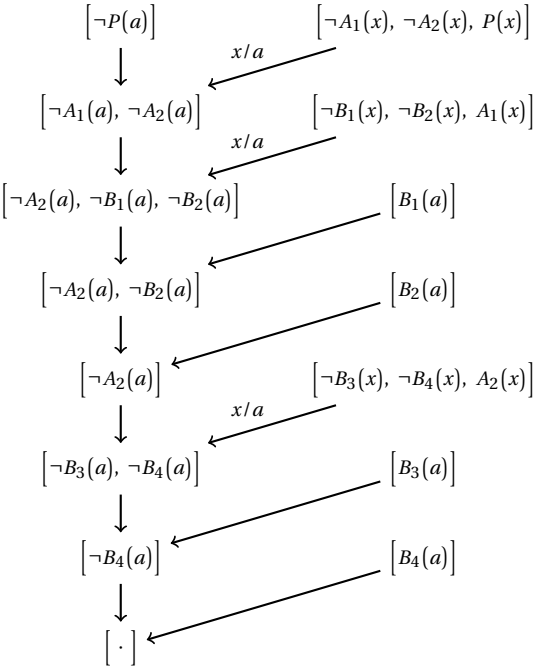
1. $\left[\neg A_1(x),\ \neg A_2(x),\ P(x)\right]$
2. $\left[\neg B_1(x),\ \neg B_2(x),\ A_1(x)\right]$
3. $\left[\neg B_3(x),\ \neg B_4(x),\ A_2(x)\right]$
4. $\left[B_1(a)\right]$
5. $\left[B_2(a)\right]$
6. $\left[B_3(a)\right]$
7. $\left[B_4(a)\right]$

É possível chegar na cláusula vazia em todos os ramos.



(b) Utilize **resolução SLD** para provar que $P(a)$ é consequência desta base de conhecimento.

Adicionando a negação da pergunta $P(a)$ à base, chegamos na cláusula vazia por resolução SLD, obtendo resposta positiva.



■

3. Considere o seguinte programa em Prolog:

```
result([_, E | L], [E | M]) :- !, result(L, M).
result(_, []).
```

(a) Após ter carregado este programa, qual seria a resposta do Prolog para a seguinte consulta?

```
?- result([a, b, c, d, e, f, g], X).
```

Consulta na base:

```
?- forall(result([a, b, c, d, e, f, g], X),
          writef("%t\n", [X])),
[b,d,f]
```

Rastro da recursão:

```
result([a, b | [c,d,e,f,g]], [b | _5688])
result([c, d | [e,f,g]], [d | _5738])
result([e, f | [g]], [f | _5788])
result([g], [])
```

(b) Descreva brevemente **o que** o programa faz e **como** ele o faz quando o primeiro argumento do predicado `result/2` é instanciado com uma lista e uma variável é passada no segundo argumento, assim como no item **(a)**. Suas explicações devem incluir respostas às seguintes perguntas:

1. Qual(is) caso(s) é(são) coberto(s) pelo fato?

O fato valida-se fazendo `X` conter os elementos em posição par da lista passada como primeiro argumento,

2. Qual efeito tem o corte na primeira linha do programa?

O corte na segunda linha serve para forçar que a recursão de `result([_, E | L], [E | M])` seja de calda com `result(_, []).` como caso base. Isto é, o corte impede que, quando a primeira regra é válida, a segunda (que é sempre válida) também não seja. Sem ele, haveria muitas respostas:

```
?- forall(result([a, b, c, d, e, f, g], X),
          writef("%t\n", [X])),
[b,d,f]
[b,d]
[b]
[]
```

3. Por que foi utilizada a variável anônima?

Por que não temos interesse em seu valor; ele não é usado em outras partes do código. ■

4. Suponha a seguinte base de conhecimento:

```
homem(americo).
homem(daniel).
homem(paulo).
homem(carlos).
homem(joaquim).
homem(filipe).

mulher(teresa).
mulher(sonia).
mulher(ana).
mulher(carla).
mulher(barbara).
mulher(maria).

idade(americo, 18).
idade(daniel, 60).
idade(paulo, 25).
idade(carlos, 37).
idade(joaquim, 80).
idade(felipe, 32).
idade(teresa, 18).
idade(sonia, 28).
idade(ana, 17).
idade(carla, 26).
idade(barbara, 51).
idade(maria , 79).

irmaos(americo, paulo).
irmaos(carlos, sonia).

pai(carlos, teresa).
pai(daniel, americo).
pai(daniel, paulo).
pai(joaquim, daniel).

mae(maria, daniel).
mae(barbara, ana).

casados(filipe, carla).
casados(americo, teresa).
casados(joaquim, maria).
```

(a) `avof(Mul, Pess)` em que `Mul` seja avó de `Pess`.

```
progenitor(A, B) :-
    pai(A, B);
    mae(A, B).

avof(X, Y) :-
    mae(X, Z),
    progenitor(Z, Y).
```

Consulta na base:

```
?- forall(avof(X, Y), writef("%t %t\n", [X, Y]))
maria americo
maria paulo
```

(b) `avom(Hom, Pess)` em que `Hom` seja avô de `Pess`.

```
avom(X, Y) :-
    pai(X, Z),
    progenitor(Z, Y).
```

Consulta na base:

```
?- forall(avom(X, Y), writef("%t %t\n", [X, Y]))
joaquim americo
joaquim paulo
```

(c) `bisavom(Hom, Pess)` que é verdadeiro se Hom for bisavô de Pess.

```
bisavo(X, Y) :-
    avom(X, Z),
    progenitor(Z, Y).
```

Consulta na base:

```
?- forall(bisavo(X, Y), writef("%t %t\n", [X, Y]))
```

(d) `primo_1(P1, P2)` que é verdadeiro se `P1` e `P2` forem primos em primeiro grau.

```
primo_1(A, B) :-
    progenitor(C, A),
    progenitor(D, B),
    \+ progenitor(D, A),
    \+ progenitor(C, B),
    setof(E, (progenitor(E, C), progenitor(E, D)), _).
```

Consulta na base:

```
?- forall(primo_1(X, Y), writef("%t %t\n", [X, Y]))
```

(e) `primo(P1, P2)` que é verdadeiro se `P1` e `P2` forem primos em qualquer grau.

```
primo(A, B) :-
    primo_1(A, B).
primo(A, B) :-
    (progenitor(C, A), primo(C, B));
    (progenitor(C, B), primo(A, C)).
```

Consulta na base:

```
?- forall(primo(X, Y), writef("%t %t\n", [X, Y]))
```

(f) `maior_de_idade(Pess)` que é verdadeiro se `Pess` for maior de idade.

```
maior_de_idade(A) :-
    age(A, B),
    B >= 18.
```

Consulta na base:

```
?- forall((maior_de_idade(X), idade(X, Y)),
          writef("%t %t\n", [X, Y]))
americo 18
daniel 60
paulo 25
carlos 37
joaquim 80
filipe 32
teresa 18
sonia 28
carla 26
barbara 51
maria 79
```

(g) `pessoas(Lista)` que devolve a `Lista` de todas as pessoas existentes na base de conhecimento.

```
pessoas(L) :-
    setof(A, (homem(A); mulher(A)), L).
```

Consulta na base:

```
?- forall(pessoas(L), writef("%t\n", [L]))
[americo,ana,barbara,carla,carlos,daniel,felipe,
 joaquim,maria,paulo,sonia,teresa]
```

(h) `mais_velho(Pess)` que retorna a pessoa mais velha que consta na base de conhecimento.

```
mais_velho(A) :-
    idade(A, B),
    forall(idade(_, D), B >= D).
```

Consulta na base:

```
?- forall((mais_velho(X), idade(X, Y)),
          writef("%t %t\n", [X, Y]))
joaquim 80
```

(i) `lista_pessoas(Lista, Sexo)` que retorna uma `Lista` de todas as pessoas do `Sexo` indicado (`m`/`f`), incluindo as suas respectivas idades. Por exemplo, `lista_pessoas(Lista, m)` deveria retornar:

```
Lista=[[americo, 18], [daniel, 60], [paulo, 25],
       [carlos, 37], [joaquim, 80], [filipe, 32]].
```

```
lista_pessoas(L, m) :-
    findall([A, B], (homem(A), idade(A, B)), L).
lista_pessoas(L, f) :-
    findall([A, B], (mulher(A), idade(A, B)), L).
```

Consulta na base:

```
?- forall(lista_pessoas(L, m), writef("%t\n", [L]))
[[americo,18],[daniel,60],[paulo,25],
 [carlos,37],[joaquim,80],[filipe,32]]
?- forall(lista_pessoas(L, f), writef("%t\n", [L]))
[[teresa,18],[sonia,28],[ana,17],
 [carla,26],[barbara,51],[maria,79]]
```

(j) `adequados(Hom, Mul)` que é verdadeiro se `Hom` for um homem, `Mul` for uma mulher e o homem for (no máximo) 2 anos mais novo do que a mulher ou 10 anos mais velho do que ela e se ambos não tiverem nenhuma relação de parentesco nem nenhum deles for casado.

```
adequados(A, B) :-
    homem(A),
    mulher(B),
    idade(A, C),
    idade(B, D),
    (C >= D - 2),
    (D >= C - 10).
```

Consulta na base:

```
?- forall(adequados(X, Y), writef("%t %t\n", [X, Y]))
americo teresa
americo ana
daniel barbara
paulo teresa
paulo ana
paulo carla
carlos sonia
joaquim maria
filipe sonia
filipe carla
```

■