
Ciências de Engenharia Informática e de Computadores

# Lógica para Programação

Solução do Segundo Teste

9 de Junho de 2008

15:00–16:30

Nome: _____ Número: _____

1. Escolha a *única* resposta *correcta* para as seguintes questões. Cada resposta certa vale 1 valor e *cada resposta errada desconta 0.3 valores*.

   (a) **(1.0)** Uma cláusula de Horn

   A. contém no máximo um literal negativo;
   B. contém no máximo um literal positivo;
   C. não contém nenhum literal negativo;
   D. não contém nenhum literal positivo.

   **Resposta:** B

   (b) **(1.0)** Dizem-se *cláusulas determinadas*

   A. as regras e os objectivos;
   B. as regras e os factos;
   C. os objectivos e os factos;
   D. as regras, os objectivos e os factos.

   **Resposta:** B

   (c) **(1.0)** Uma função de selecção

   A. é uma regra para escolher um literal numa cláusula objectivo como candidato à aplicação do princípio da resolução;
   B. é uma regra para escolher um literal numa cláusula determinada como candidato à aplicação do princípio da resolução;
   C. é uma função do conjunto de literais e do conjunto das cláusulas determinadas para o conjunto dos programas;
   D. é uma função do conjunto de literais e do conjunto dos programas para o conjunto das cláusulas determinadas.

   **Resposta:** A

2. Escolha a *única* resposta *incorrecta* para as seguintes questões. Cada resposta certa vale 1 valor e *cada resposta errada desconta 0.3 valores*.

   (a) **(1.0)** Numa árvore SLD

   A. o rótulo de cada nó é um objectivo;
   B. um ramo cuja folha tem o rótulo $\square$ diz-se um nó falhado;
   C. um ramo que não seja um nó falhado nem um nó bem sucedido é um ramo infinito;

D. a um nó bem sucedido corresponde uma resposta.

**Resposta:** B

(b) **(1.0)** São exemplos de literais em PROLOG

A. `4 < 3`

B. `aluno(X, lp)`

C. `aluno`

D. `X`

**Resposta:** D

(c) **(1.0)** O operador de corte

A. é sempre avaliado pelo PROLOG como verdadeiro;

B. altera a semântica procedimental do PROLOG;

C. reduz sempre o espaço de procura de soluções;

D. compromete o PROLOG com todas as escolhas que foram feitas desde a unificação com a cláusula que contém o corte até ao operador de corte.

**Resposta:** C

3. **(0.5)** Diga o que é a base de Herbrand para um conjunto de cláusulas.

**Resposta:**

Sendo $\Delta$ um conjunto de cláusulas, o conjunto de todas as *fbfs* atómicas da forma $P^n(t_1, \ldots, t_n)$ para todos os predicados $n$-ários, $P^n$ ($n \geq 0$), existentes em $\Delta$, em que $t_1, \ldots, t_n$ pertencem ao universo de Herbrand para $\Delta$ tem o nome de *base de Herbrand* para $\Delta$. A base de Herbrand para um conjunto de cláusulas $\Delta$ é o conjunto de todas as *fbfs* atómicas fechadas que é possível construir com as letras de predicado em $\Delta$ e com os termos em $U_\Delta$.

4. **(0.5)** O que é uma regra de procura? Qual a regra de procura utilizada em PROLOG?

**Resposta:**

Uma regra de procura é uma função do conjunto dos literais e do conjunto dos programas para o conjunto das cláusulas definidas, tal que $P(\alpha, \Delta) \in \Delta$. Uma *regra de procura*, $P$, é uma regra que, dado um literal correspondente a um objectivo, escolhe uma cláusula definida num programa para aplicar o princípio da resolução com o objectivo dado.

O PROLOG escolhe a primeira cláusula que aparece no programa.

5. **(1.0)** Considere a seguinte avaliação em PROLOG: `V is Exp`. Qual a semântica desta expressão? Qual a razão por que a utilização da avaliação "estraga" a propriedade dos predicados em PROLOG poderem ser utilizados com qualquer argumento como dado ou como resultado?

**Resposta:**

Ao avaliar um literal da forma `V is Exp`, se a expressão `Exp` é avaliada sem erros, produzindo um valor, então se este valor é unificável com `V` a avaliação tem sucesso devolvendo a substituição adequada; em caso contrário, a avaliação falha.

Uma vez que numa avaliação se avalia o termo antes da ligação do seu valor à variável, as variáveis que eventualmente existam no termo devem estar instanciadas no momento da sua avaliação. Isto significa que com a introdução da avaliação, perdemos a possibilidade de utilizar qualquer dos argumentos como variável.

6. Considere a seguinte conceptualização $C = (D, F, R)$:

   - $D = \{\boxplus, \boxminus, \boxtimes, \boxdot\}$
   - $F = \{\{(\boxplus, \boxtimes)\}, \{(\boxminus, \boxdot)\}, \{(\boxminus, \boxdot, \boxtimes), (\boxminus, \boxtimes, \boxdot)\}\}$
   - $R = \{\{(\boxplus, \boxdot)\}, \{(\boxtimes, \boxdot)\}, \{(\boxplus), (\boxminus), (\boxtimes), (\boxdot)\}\}$

   e a seguinte interpretação:

   - $I(a) = \boxplus$
   - $I(b) = \boxminus$
   - $I(c) = \boxtimes$
   - $I(d) = \boxdot$
   - $I(f1) = \{(\boxplus, \boxtimes)\}$
   - $I(f2) = \{(\boxminus, \boxdot)\}$
   - $I(R1) = \{(\boxplus, \boxdot)\}$
   - $I(R2) = \{(\boxtimes, \boxdot)\}$
   - $I(R3) = \{(\boxplus), (\boxminus), (\boxtimes), (\boxdot)\}$

   (a) **(1.5)** Diga, justificando, se esta interpretação para esta conceptualização é um modelo do seguinte conjunto de fórmulas:

   $$\{\exists x, y[R3(x) \wedge R1(x, f2(y))], R2(f1(a), d) \vee \neg R3(b)\}.$$

   **Resposta:**
   Uma interpretação para uma conceptualização é um modelo de um conjunto de fórmulas se satisfizer todas as fórmulas desse conjunto, isto é, se as tornar todas verdadeiras.

   - A fbf $\exists x, y[R3(x) \wedge R1(x, f2(y))]$ é satisfeita sse existir uma substituição $\{c1/x, c2/y\}$, em que $c1$ e $c2$ são constantes individuais, tal que $I$ satisfaz $R3(c1) \wedge R1(c1, f2(c2))$. Sejam $c1 = a$ e $c2 = b$. Neste caso, a fbf $R3(a) \wedge R1(a, f2(b))$ é satisfeita sse $(I(a)) \in I(R3)$ e $(I(a), I(f2)(I(b))) \in I(R1)$, ou seja, sse $(\boxplus) \in \{(\boxplus), (\boxminus), (\boxtimes), (\boxdot)\}$ e $(\boxplus, \boxdot) \in \{(\boxplus, \boxdot)\}$. Como ambas as condições se verificam, a conjunção é satisfeita para esta substituição e a fórmula quantificada existencialmente também é satisfeita.
   - A fbf $R2(f1(a), d) \vee \neg R3(b)$ é satisfeita sse $R2(f1(a), d)$ for satisfeita ou $\neg R3(b)$ for satisfeita. A fbf $R2(f1(a), d)$ é satisfeita sse $(I(f1)(I(a)), I(d)) \in I(R2)$, ou seja, sse $(\boxtimes, \boxdot) \in \{(\boxtimes, \boxdot)\}$, o que se verifica. Como um dos elementos da disjunção é satisfeito, a disjunção inicial é satisfeita.

   Como ambas as fbfs são satisfeitas, esta interpretação para esta conceptualização é modelo deste conjunto de fórmulas.

(b) **(1.0)** Explique porque é que o seguinte não pode ser uma interpretação para esta conceptualização, mencionando *cinco* dos erros que foram cometidos.

- $I(a) = \boxplus$
- $I(b) = \boxplus$
- $I(c) = \boxplus$
- $I(c) = \boxminus$
- $I(d) = \circledast$
- $I(f1) = \{(\boxplus, \boxminus), (\boxplus, \boxtimes)\}$
- $I(R1) = \{(\boxplus), (\boxtimes), (\boxdot)\}$

**Resposta:**

Não pode ser uma interpretação para esta conceptualização porque:

- A constante $c$ tem duas interpretações diferentes.
- A interpretação da constante $d$ não pertence ao domínio da conceptualização.
- A função $f1$ tem dois resultados diferentes quando o seu argumento é $\boxplus$.
- A interpretação da função $f1$ não pertence ao conjunto de funções da conceptualização.
- A interpretação da relação $R1$ não pertence ao conjunto de relações da conceptualização.

7. **(1.5)** Considere o seguinte conjunto de cláusulas de Horn:

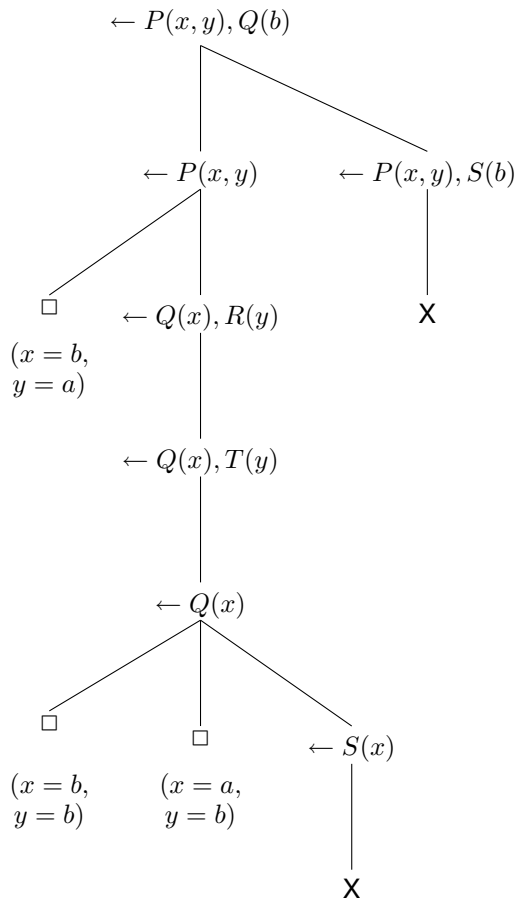
- $P(x, y) \leftarrow Q(x), R(y)$
- $Q(x) \leftarrow S(x)$
- $R(x) \leftarrow T(x)$
- $P(b, a)$
- $Q(b)$
- $Q(a)$
- $T(b)$

Usando uma árvore de resolução SLD e uma função de selecção que escolha para unificar o *último* literal do objectivo, mostre todas as soluções para o seguinte objectivo:

$$\leftarrow P(x, y), Q(b).$$

Pode usar a estratégia de procura que preferir. No final indique explicitamente todas as soluções.

**Resposta:**




Soluções encontradas: $x = b, y = a$;
$x = b, y = b$;
$x = a, y = b$.

8. Considere a definição dos *números de Fibonacci*:

$$fib(n) = \begin{cases} 0 & \text{se } n = 0 \\ 1 & \text{se } n = 1 \\ fib(n-1) + fib(n-2) & \text{se } n > 1 \end{cases}$$

Seja `fib` o predicado com o seguinte significado: `fib(N, V)` afirma que o `N`-ésimo número de Fibonacci é `V`.

(a) **(1.0)** Escreva um programa em PROLOG que implementa o predicado `fib`.
**Resposta:**

```
fib(0, 0).
fib(1, 1).
fib(X, Fib_X) :-
      X_menos_1 is X - 1,
      fib(X_menos_1, Fib_X_menos_1),
      X_menos_2 is X - 2,
      fib(X_menos_2, Fib_X_menos_2),
      Fib_X is Fib_X_menos_1 + Fib_X_menos_2.
```

(b) **(0.5)** Qual a resposta do seu programa ao objectivo `fib(X, 21)`? Justifique a sua resposta.
**Resposta:**

```
?- fib(X, 21).
ERROR: is/2: Arguments are not sufficiently instantiated
^  Exception: (8) _L134 is _G180-1
```

O operador `is` necessita que todas as variáveis da expressão a avaliar estejam instanciadas.

9. **(1.5)** Considere a seguinte base de conhecimento:

```
cao(bobi).
cao(fiel).
cao(guerreiro).
morde(guerreiro).
```

E as duas formas de representar que o Carlos gosta de todos os cães que não mordam. Repare que em termos lógicos não existem diferenças entre as duas.

```
gosta1(carlos,X) :- cao(X), \+(morde(X)).

gosta2(carlos,X) :- \+(morde(X)), cao(X).
```

Explique qual a resposta do PROLOG a cada um dos objectivos `gosta1(carlos,X)` e `gosta2(carlos,X)`. Se as repostas forem diferentes, explique a razão dessas diferenças.
**Resposta:**

```
?- gosta1(carlos,X).

X = bobi ;

X = fiel ;

No
?- gosta2(carlos,X).

No
?-
```

No primeiro caso, primeiro o PROLOG vai tentar encontrar os cães e só depois de estarem instanciados é que vai tentar provar que não mordem. Neste caso, os resultados são os esperados.

No segundo caso, uma vez que a variável `X` não está instanciada quando o PROLOG vai tentar provar `\+(morde(x))`, este objectivo falha porque existe algo que morde na base de conhecimento.

Convém notar que ambos os objectivos dariam origem a um erro se não existisse nada que mordesse nem nenhuma definição para o que significa morder na base de conhecimento, pois o predicado não estaria definido.

10. Considere o seguinte programa em PROLOG:

```
a_1(X, Y) :- b(X), c(Y).

a_2(X, Y) :- !, b(X), c(Y).

a_3(X, Y) :- b(X), !, c(Y).

a_4(X, Y) :- b(X), c(Y), !.

b(0).
b(1).

c(2).
c(3).
```

Diga quais as respostas fornecidas para os seguintes objectivos, considerando que o utilizador escreve `;` até esgotar todas as respostas.

(a) **(0.5)** `a_1(X, Y).`
**Resposta:**

```
X = 0,
Y = 2 ;
X = 0,
Y = 3 ;
X = 1,
Y = 2 ;
X = 1,
Y = 3 ;
No
```

(b) **(0.5)** `a_2(X, Y).`
**Resposta:**

```
X = 0,
Y = 2 ;
X = 0,
Y = 3 ;
X = 1,
Y = 2 ;
X = 1,
Y = 3 ;
No
```

(c) **(0.5)** `a_3(X, Y).`
**Resposta:**

```
X = 0,
Y = 2 ;
X = 0,
Y = 3 ;
No
```

(d) **(0.5)** `a_4(X, Y).`
**Resposta:**

```
X = 0,
Y = 2 ;
No
```

11. Considere as seguintes estruturas:

```
actor(NumA, NomeA, SexoMF, AnoNascimento)
filme(NumF, NomeF, AnoEstreia)
participa(NumA, NumF)
```

(a) Supondo disponível uma base de dados com os dados de cinema (BDCinema.pl), implemente em PROLOG as seguintes funcionalidades:

i. **(0.5)** `listaFilmes(NomeA, ListaResultado)` que, dado o nome de um actor (`NomeA`), guarda em `ListaResultado` a lista dos números dos filmes em que este participou.
**Resposta:**

```
listaFilmes(NomeA, ListaResultados) :-
   findall(NumF,
            (actor(NumA, NomeA, _, _), participa(NumA, NumF)),
            ListaResultados).
```

ii. **(1.0)** `listaPares(NomeF, ListaResultado)` que, dado o nome de um filme (`NomeF`), guarda em `ListaResultado` uma lista de pares (`NumA`, `NomeA`), com o número (`NumA`) e o nome (`NomeA`) de cada actor que participou no filme.
**Resposta:**

```
listaPares(NomeF, ListaResultados) :-
        findall((NumA, NomeA),
        (filme(NumF, NomeF, _),
         participa(NumA, NumF),
         actor(NumA, NomeA, _, _)),
        ListaResultados).
```

(b) **(1.5)** Supondo que um realizador queria convidar um conjunto de *actrizes* entre os *90 e os 95 anos*, para um "*casting*", escreva um pequeno programa em PROLOG que usa BDCinema.pl para gerar o convite, que deverá ter a seguinte forma:
Cara `NomeA`
Gostaria de convidá-la para ...

**Resposta:**

```
casting :- ['BDCinema.pl'],
           findall(NomeA,
                   (actor(_, NomeA, 'F', Data),
                    Data > 1913,
                    Data < 1918),
           ListaActrizes),
           convida(ListaActrizes).
```

```
convida([]).
convida([NomeA | Resto]) :-
      write('Cara '),
      write(NomeA),
      nl,
      write('Gostaria de convidá-la para...'),
      nl,
      convida(Resto).
```