Lógica Computacional

2^o Ano LCC

2007/2008

1 Lógica Proposicional

1.1 Sintaxe e Semântica

1. Defina *tautologia* e *fórmula refutável*.

2. Para cada uma das seguintes fórmulas apresente a respectiva árvore sintáctica.

   (a) $p \vee \neg p \wedge q \wedge r$

   (b) $\neg(p \wedge q)$

   (c) $p \Rightarrow q \Rightarrow r$

   (d) $p \Rightarrow \neg q \vee r \wedge s$

3. Mostre que $p \Rightarrow q \Rightarrow r \not\equiv (p \Rightarrow q) \Rightarrow r$.

4. Considere os seguintes modelos de $\mathcal{L}_{\mathcal{P}}$:

   $$M_1 = \{p, q\} \quad \text{e} \quad M_2 = \{q\}$$

   Determine a validade de cada uma das proposições seguintes no modelo M_1 e no modelo M_2:

   (a) $p \Rightarrow (q \wedge p)$

   (b) $(p \vee q) \Rightarrow \neg q$

5. Para cada uma das proposições seguintes apresente (se possível) um modelo que a valide e um que a refute.

   (a) $p \Rightarrow r$

   (b) $\neg(p \wedge r)$

   (c) $p \wedge \neg p$

   (d) $p \vee (p \Rightarrow r)$

(e) $(p \Rightarrow q \Rightarrow r) \Rightarrow (p \Rightarrow q) \Rightarrow (p \Rightarrow r)$

(f) $((p \wedge \neg q) \Rightarrow r) \Rightarrow q$

(g) $((p \Rightarrow q)) \vee (r \Rightarrow s) \Rightarrow ((p \wedge r) \Rightarrow (q \wedge s))$

6. Demonstre que:

   (a) A teoria $\{\varphi, \neg\varphi\}$ é uma teoria inconsistente.

   (b) Teorema da dedução: $\Gamma \models \psi \Rightarrow \varphi$ sse $\Gamma, A \models B$.

   (c) Para uma qualquer fórmula φ e teoria Γ,

   $$\Gamma \models \varphi \qquad \text{sse} \qquad \Gamma \cup \{\neg\varphi\} \models \bot.$$

   (d) Para qualquer teoria Γ e fórmulas φ e ψ,

   $$\Gamma \models \varphi \Rightarrow \psi \qquad \text{sse} \qquad \Gamma, \varphi \models \psi.$$

7. Quais das seguintes consequências semânticas se verificam? Justifique.

   (a) $\{p \Rightarrow r\} \models p \wedge r$

   (b) $\{p, r\} \models p \Rightarrow r$

   (c) $\models p \vee \neg p$

   (d) $\{p \wedge \neg p\} \models p \vee r$

8. Verifique que $((p \Rightarrow q) \Rightarrow p) \Rightarrow p$ é uma tautologia.

9. Seja $A \downarrow B$ a abreviatura de $\neg(A \vee B)$ (*Peirce's dagger*). Exprima as conectivas $\wedge, \vee, \neg, \Rightarrow$ recorrendo unicamente a essa conectiva.

10. Quais das seguintes teorias são consistentes? Jutifique.

    (a) $\{\neg p \wedge q \Rightarrow r, p \Rightarrow (\neg p \Rightarrow q), r \Rightarrow \neg q\}$

    (b) $\{p \Rightarrow q, q \Rightarrow r, r \Rightarrow \neg q\}$

    (c) $\{p_0 \Rightarrow p_1, p_0 \wedge p_2 \Rightarrow p_1 \wedge p_4, p_0 \wedge p_2 \wedge p_4 \Rightarrow p_1 \wedge p_3 \wedge p_5, \cdots\}$

11. Como poderia construir um algoritmo que, dado um modelo, verificasse a validade de uma fórmula no modelo.

1.2 Relações de Dedução e Sistema de Dedução Natural

1. Mostre que as seguintes relações constituem *relações de dedução*:
   (a) Relação de *Consequência Semântica* ($\models$);
   (b) Relaçao $\ni$ (i.e. $\Gamma \ni \varphi$ sse $\varphi \in \Gamma$);
   (c) Relação $\vdash^{ND}$.
2. Mostre que $\Gamma, A \vdash^{ND} B$ sse $\Gamma \vdash^{ND} A \Rightarrow B$.
3. Mostre que $\Gamma \vdash^{ND} \neg A$ sse $\Gamma, A \vdash^{ND} \bot$.
4. Mostre que, se $\Gamma \vdash^{ND} \neg A$ e $\Gamma \vdash^{ND} \neg B$ então $\Gamma \vdash^{ND} \neg(A \vee B)$.
5. Mostre que, se $\Gamma, A \vdash^{ND} B$ e $\Gamma, \neg A \vdash^{ND} B$ então $\Gamma \vdash^{ND} B$.

1.3 Formas Normais e Clausais

1. Determine a *forma normal negativa* equivalente a cada uma das seguintes proposições:
   (a) $(p \vee t) \Rightarrow (p \wedge t)$
   (b) $(p \wedge (p \Rightarrow t)) \Rightarrow \neg p$
   (c) $((p \Rightarrow t) \Rightarrow p) \Rightarrow p$
   (d) $(\neg p \Rightarrow t) \Rightarrow (\neg p \Rightarrow \neg t) \Rightarrow p$
   (e) $(a \Rightarrow b \vee c) \Rightarrow a \wedge \neg b \Rightarrow c$
2. Construa a *forma normal conjuntiva* e a *forma normal disjuntiva* equivalente a cada uma das proposições da alínea anterior.
3. Construa uma forma normal conjuntiva que é tautologia se e só se for válido

$$\{p \vee q, p \Rightarrow r, q \Rightarrow r\} \models r$$

1.4 Calculo de Sequentes e método de *Tableaux*

1. Construa derivações no cálculo de sequentes para as seguintes fórmulas:
   (a) $p \Rightarrow (q \Rightarrow p \wedge q)$
   (b) $(p \Rightarrow \neg q) \Rightarrow \neg q$
   (c) $(p \Rightarrow q) \Rightarrow ((p \Rightarrow q \Rightarrow r) \Rightarrow (p \Rightarrow r))$
2. Utilizando o método de *Tableaux*, verifique:
   (a) $(p \Rightarrow q) \wedge (p \Rightarrow \neg q) \Rightarrow \neg p$

(b) $p \vee q \Rightarrow p \wedge q$

3. Construa um tableaux que prove a validade de

$$\{p \vee q, p \Rightarrow r, q \Rightarrow r\} \models r.$$

1.5 Métodos de Verificação

1. Considere o algoritmo de resolução de Robinson para a Lógica Proposicional aplicado a formas normais conjuntiva. Mostre a validade das seguintes observações:

   (a) Se φ contém um par de cláusulas $(\alpha \vee p)$ e $(\beta \vee \neg p)$ então φ é semanticamente equivalente a $\varphi \wedge (\alpha \vee \beta)$.

   (b) Se $\neg p$ não ocorre em nenhuma cláusula de φ então, φ será inconsistente se e só se $\varphi \wedge (\alpha \vee p)$ for inconsistente.

2. Considere o seguinte conjunto de cláusulas $\Gamma = \{\ \neg p \vee \neg q \vee r,\ p \vee \neg q,\ q\ \}$. Use a resolução para demonstrar que $\Gamma \vdash r$.

3. Usando o *algoritmo de resolução de Robinson*, verifique se os seguintes conjuntos de cláusulas (na FNC) são inconsistentes:

   (a) $\{\{a, b, \neg c\}, \{\neg a, c\}, \{\neg b, e\}\}$

   (b) $\{\{\neg a\}, \{a, \neg b\}, \{a, \neg c\}, \{b, c\}, \{\neg b\}\}$

   (c) $\{\{a, \neg b\}, \{\neg a, b\}\}$

   (d) $\{\{a, \neg b\, c\}, \{a, b\}, \{\neg c, \neg a\}\}$

   (e) $\{\{a, b\}, \{\neg a, b\}, \{a, \neg b\}, \{\neg a, \neg b\}$

4. Use o *algoritmo de resolução de Robinson*, para verificar se as seguintes relações de consequência se verificam:

   (a) $\{r \wedge s \Rightarrow p, r, s\} \models p$

   (b) $\{c \wedge d \Rightarrow a, d, b \Rightarrow a\} \models a$

5. Use o *método de fraccionamento de Davis Putnam* para testar se as seguintes fórmulas são ou não tautologias. Com base na árvore gerada indique (se possível) modelos que validem a fórmula e modelos que a refutem.

   (a) $(p \vee t) \Rightarrow (p \wedge t)$

   (b) $(\neg p \Rightarrow t) \Rightarrow (\neg p \Rightarrow \neg t) \Rightarrow p$

   (c) $((\neg a \vee \neg b) \wedge \neg(a \wedge \neg b)) \Rightarrow \neg a$

   (d) $((p \Rightarrow t) \Rightarrow p) \Rightarrow p$

6. Relembre o *algoritmo de Davis Putnam* para testar se uma FNC é inconsistente (trabalhando com FNC representadas como conjuntos de conjuntos de literais), e tenha em consideração os aspectos de simplificação do algoritmo. Use este algoritmo para verificar se:

   (a) $\{\{a, \neg b\}, \{\neg a, \neg c, d\}, \{a, \neg d\}, \{\neg a, c, b\}, \{c\}\}$ é inconsistente.

   (b) $\{a \Rightarrow b \wedge c, b \Rightarrow \neg d, a \wedge (\neg b \vee \neg e \vee d), c \Rightarrow e\}$ é inconsistente.

   (c) $\{p \Rightarrow s \wedge r, \neg q \vee s, \} \models p \wedge q \Rightarrow r$

7. O seguinte programa Prolog constroi a árvore de fraccionamento de Davis Putnam pelo próprio processo de prova.

```
inconsistente([]) :- fail.
inconsistente(P) :- member([],P).
inconsistente(P) :- fracciona(P,P1,P2),
                    inconsistente(P1), inconsistente(P2).
```

   (a) Defina o predicado `fracciona/3`.

   (b) Indique o que teria que fazer para contemplar os aspectos de simplificação do algoritmo.

8. Para as fórmulas:

   - $(b \wedge c) \Rightarrow (a \vee b)$
   - $(a \vee b) \Rightarrow (b \wedge c)$

   determine:

   (a) A BDD respectiva obtida por redução da árvore binária de decisão.

   (b) A construção *bottom-up* da BDD.

2 Lógica de Predicados

2.1 Sintaxe e Semântica

1. Exiba um modelo que valide e um que refute cada uma das fórmulas apresentadas:

   (a) $\forall_X.\exists_Y.P(X, f(Y))$

   (b) $\exists_X.\forall_Y.P(X, f(Y))$

2. Mostre que:

   (a) $\models \exists_X.\forall_Y.\varphi \Rightarrow \forall_Y.\exists_X.\varphi$,

(b) $\not\models \forall_X.\exists_Y.\varphi \Rightarrow \exists_Y.\forall_X.\varphi$,

(c) $\forall_X.(\varphi \Rightarrow \psi) \Rightarrow (\exists_X.\varphi \Rightarrow \exists_X.\psi)$.

3. Indique se as fórmulas apresentadas abaixo são válidas, inconsistentes ou não são uma coisa nem outra.

(a) $(\forall_X.\forall_Y.(P(X) \vee Q(Y))) \vee (\exists_X.\exists_Y.(\neg P(X) \wedge \neg Q(Y)))$,

(b) $(\neg\exists_X.\exists_Y.P(X,Y)) \vee (\exists_X.P(X,X))$,

(c) $(\exists_X.\exists_Y.P(X,Y)) \vee (\exists_X.P(X,X))$,

(d) $\forall_X.\forall_Y.(P(X,Y) \vee \neg P(Y,X))$,

(e) $(\forall_X.P(X,X)) \Rightarrow (\exists_X.\forall_Y.(P(X,Y) \Rightarrow P(Y,X)))$,

(f) $(\forall_X.P(X,X)) \Rightarrow (\forall_Y.\exists_X.(P(X,Y) \Rightarrow P(Y,X)))$,

(g) $\exists_X.\forall_Y.(p(X,Y) \Rightarrow \neg P(Y,Y))$.

4. Determine a forma *Prenex* e *Skolen* de cada uma das seguintes frases lógicas:

(a) $\exists_X.(\forall_Y.\exists_Z.P(X,Y,Z) \wedge \exists_Z.\forall_Y.\neg P(X,Y,Z))$

(b) $\forall_X.(\exists_Y.Q(X,Y) \vee \forall_Y.\exists_Z.R(X,Y,Z))$

(c) $\forall_X.(\exists_Y.P(X,Y) \Rightarrow \exists_Y.Q(X,Y))$

2.2 Cálculo de Sequentes

1. Estabeleça as seguintes asserções utilizando o cálculo de sequentes.

(a) $\neg\forall_X.P(X) \vdash \exists_X.\neg P(X)$

(b) $(\forall_X.P(X)) \wedge Q(X) \vdash \forall'_X.(P(X') \wedge Q(X))$

(c) $\forall'_X.(P(X') \wedge Q(X)) \vdash (\forall X.P(X)) \wedge Q(X)$

2. Considere as seguintes fórmulas

(a) $\exists_X.\forall_Y.\varphi \wedge \psi \Rightarrow \forall_Y.(\exists_X.\varphi) \wedge (\exists_X.\psi)$

(b) $\exists_X.(\forall_Y.\varphi) \vee (\forall_Y.\psi) \Rightarrow \forall_Y.\exists_X.\varphi \vee \psi$

(a) Construa árvores de prova em cálculo de sequentes.

(b) Construa contra-exemplos que mostre que as implicações inversas não são válidas.

2.3 Resolução

1. Considere a fórmula universal

   $$\forall_X.(R(\mathsf{c})\wedge\neg R(f(\mathsf{c}))\wedge R(f(f(f(\mathsf{c}))))\wedge(R(X)\vee\neg R(f(f(f(X)))))\wedge(\neg R(X)\vee R(f(f(X)))))$$

   mostre que é uma contradição utilizando resolução proposicional.

2. Determine, se existir, os seguintes unificadores mais gerais para os seguintes conjuntos:

   (a) $\{f(X, g(X)), f(Y, Z)\}$,
   (b) $\{P(X, f(X)), P(f(X), Z\}$,
   (c) $\{P(X, g(X), Y), P(Z, U, g(U))\}$,
   (d) $\{P(g(X), Y), P(Y, Y), P(U, f(W))\}$,

3. Use a resolução de primeira ordem para verificar a inconsistência das seguintes conjuntos de cláusulas:

   (a) $\{\{R(X, Y), R(Y, X)\}, \{\neg R(X, f(X))\}\}$,
   (b) $\{\{\neg P(X), P(f(f(X)))\}, \{\neg Q(X), Q(f(f(X)))\}, \{P(X), Q(X)\}, \{\neg P(X), \neg Q(X)\}\}$,

4. Considere o algoritmo de resolução de Robinson para a Lógica de Primeira Ordem aplicado ao seguinte conjunto de clausulas disjuntivas (variáveis $\{X, Y, Z\}$,

   $$\Gamma = \{P(X, f(X)) \vee \neg Q(X, Z), Q(Y, Y), \neg P(g(Y), Y)\}$$

   (a) Calcule os resolventes da primeira e da segunda clausula e da primeira e da terceira cláusula.
   (b) Construa o grafo de resolventes resultante desta teoria e verifique se Γ é ou não inconsistente.

5. Verifique se $\perp$ é derivável dos seguintes conjuntos de cláusulas. Se sim, apresente a derivação. Se não, justifique informalmente.

   (a) $\{\{R(X, X)\}, \{\neg R(X, f(X))\}, \{R(X, f(X)), \neg R(f(X), X)\}, \{R(X, Y), \neg R(Y, Z)\}\}$,
   (b) $\{\{R(X, X)\}, \{\neg R(X, f(X))\}, \{R(X, f(X)), \neg R(f(X), X)\}, \{\neg R(X, f(Y)), R(X, Z), \neg R(Y, Z)\}\}$.

6. Use a resolução para mostrar que da assunção $\forall X.\neg gosta(X, ana) \Rightarrow gosta(ana, X)$ se pode provar que $gosta(ana, ana)$.

7. Considere o seguinte conjunto de cláusulas na forma condicional (regras):

   $$member(X, [X|R])$$
   $$member(X, [H|T]) \Leftarrow member(X, T)$$

(a) Converta cada uma destas fórmulas, em cláusulas na forma disjuntiva.

(b) Usando a resolução, demonstre que deste conjunto de cláusulas se infere $member(X, [a, b])$, indicando explicitamente as unificações necessárias. Ou seja, demontre que

$$\{\ \forall X, R, T\ .\ member(X, [X|R]),\ member(X, T) \Rightarrow member(X, [H|T])\ \}\ \models\ \exists X.member(X, [a, b])$$

8. Considere o seguinte conjunto de cláusulas na forma condicional (regras):

$$soma(0, X, X)$$
$$soma(s(X), Y, s(Z)) \Leftarrow soma(X, Y, Z)$$
$$mult(0, X, 0)$$
$$mult(s(X), Y, Z) \Leftarrow mult(X, Y, W) \wedge soma(W, Y, Z)$$

(a) Converta cada uma destas fórmulas, em cláusulas na forma disjuntiva.

(b) Usando a resolução, prove que deste conjunto de cláusulas se infere (indique explicitamente as unificações necessárias):

i. $soma(s(s(0)), s(X), Y)$

ii. $mult(s(0), s(s(0)), M)$

3 PROLOG

1. Considere o seguinte programa em *Prolog*:

```
concatena([],L,L).
concatena([H|L1],L2,[H|L3]):-concatena(L1,L2,L3).

f1(X,L) :- concatena(X,_,L).

f2([],[]).
f2([Y|T],R) :- f2(T,Z), concatena(Z,[Y],R).
```

(a) Caracterize as árvores de derivação dos seguintes objectivos:

i. `f1([2,3],[2,3,4]).`

ii. `f2([2,3,4],X).`

(b) Descreva o que fazem $f1$ e $f2$.

(c) Defina os predicados $f1$ e $f2$ de forma alternativa.

(d) Como poderia usar $f1$ para obter todos os prefixos de uma lista dada?

2. Considere uma representação de árvores binárias em Prolog.

(a) Defina um predicado que verifique se uma dada árvore é uma árvore binaria de procura.

(b) Defina um predicado que permita fazer uma travessia "inorder"na árvore binária.

(c) Defina o predicado `subArv/2` que verifique se uma árvore é uma sub-árvore de outra.

(d) Defina o predicado `altura(+Arv, ?N)` que devolve em `N` a altura da árvore `Arv`.

(e) Relembre que uma árvore diz-se balanceada quando for vazia ou quando ambas as sub-árvores forem balanceadas e as respectivas alturas não diferem em mais do que uma unidade. Defina o predicado `balanceada(+Arv)` que verifique se a árvore `Arv` está balanceada.

3. Considere a seguinte base de conhecimento de um programa *Prolog* que manipula números naturais unários:

```
a(zero, Y, Y).
a(suc(X), Y, suc(Z)) :- a(X,Y,Z).

m1(zero, _, zero).
m1(suc(X), Y, Z) :- a(Y, W, Z), m1(X,Y,W).

m2(zero, _, zero).
m2(suc(X), Y, Z) :- m2(X,Y,W), a(Y, W, Z).
```

(a) Caracterize as árvores de derivação dos seguintes objectivos:

i. `m1(X, suc(suc(zero)), suc(suc(zero))).`

ii. `m2(X, suc(suc(zero)), suc(suc(zero))).`

(b) Defina um predicado que permita calcular a exponenciação de dois números naturais.

(c) Será que poderia utilizar o predicado definido na alínea anterior para calcular o logaritmo? Justifique.

4. Considere a seguinte base de conhecimento de um programa *Prolog*:

```
q(0,X,Y) :- p(X), !, p(Y).
q(_,X,b) :- p(X).
q(_,c,Y) :- p(Y), !.
p(a). p(b).
```

(a) Quais são (todas) as soluções admissíveis para os seguintes objectivos:

i. `q(1,X,Y).`

ii. `q(0,X,Y).`

(b) Apresente as *árvores de derivação* para os objectivos da alínea anterior.

5. Defina o predicado `posFirst(+Elem, +List, ?Pos)` que, dado um elemento `Elem` e uma lista `List`, determine a primeira posição onde ocorre `Elem` em `List` (0 se não ocorrer).

6. Uma forma simples de descrever um algorito de ordenação é: *encontrar uma permutação da lista dada que esteja ordenada*. Essa descrição pode ser directamente codificada em *Prolog* usando a estratégia *generate-and-test*. Codifique o predicado `slowSort/2` que implemente essa estratégia.

7. Considere o seguinte programa em Prolog:

```
s(X,Y):- q(X,Y).
s(0,0).
q(X,Y):-r(X),t(Y).
r(1). r(2).
t(1). t(2). t(3).
```

   (a) Indique todas as soluções para a seguinte questão: `?- s(X,Y).`

   (b) Construa a árvore de derivação que explique como o Prolog encontra as soluções acima indicadas.

   (c) Considere agora que se substitui a definição de `q/2` por: `q(X,Y):-r(X),!,t(Y).` Indique quais as soluções agora encontradas para: `?- s(X,Y).` Justifique apresentando a árvore de derivação.

8. Considere a seguinte base de conhecimento:

```
q(X,Y) :- /+ p(X), !.
q(X,Y) :- p(Y).
p(a). p(b).
```

   Indique todas as soluções retornadas o interpretador de prolog à questão `q(X,X)`? Construa a árvore de derivação correspondente.

9. Defina um predicado que verifique se uma árvore binária é uma árvore binária de procura.

10. Considere o predicado `insert(Elem, List1, List2)` de inserção ordenada (`List2` resulta da inserção de `Elem` em `List1`).

```
insert(X,[H|T],[H|T1]) :- X>H, !, insert(X,T,T1).
insert(X,L,[X|L]).
```

Mostre, com um *goal* apropriado, que o predicado não satisfaz (sempre) a descrição apresentada. Como o poderia corrigir?

11. Defina um predicado que permita "calcular" a junção ordenada (*merge*) de duas listas ordenadas de inteiros.

12. Pode utilizar o predicado definido na alínea anterior instanciando unicamente a variável do ressultado? Justifique.

13. Utilizando o predicado `merge` definido atrás, defina o predicado `mergeSort/2` que permita "calcular" a ordenação de uma lista de inteiros. (obs.: pode definir predicados auxiliares...)

14. Considere o seguinte programa Prolog:

```
ap([], X, X).
ap([X|R], Y, [X|T]) :- ap(R, Y, T).
ap1([], X, X) :- !.
ap1([X|R], Y, [X|T]) :- ap1(R, Y, T).
```

Desenhe as árvores de procura para as seguintes questões:

- `ap(X, Y, [a,b]).`
- `ap1(X, Y, [a,b]).`

15. Considere o seguinte programa em Prolog:

```
gosta(joao,Y):- gelado_morango(Y),!,fail.
gosta(joao,Y):- gelado(Y).
gelado(X):-gelado_chocolate(X).
gelado(X):-gelado_morango(X).
gelado(X):-gelado_baunilha(X).
gelado_chocolate(magnun). gelado_baunilha(magnun).
gelado_morango(pernapau). gelado_morango(corneto).
```

(a) Explique o funcionamento do predicado `gosta/2`.

(b) O programa apresentado usa um red cut. Justifique esta afirmação.

(c) Para além do problema acima indicado, o comportamento do programa depende da ordem das cláusulas. Apresente um programa alternativo que não exiba estes problemas.

(d) Apresente um exemplo que mostre que a negação por falha tem um comportamento diferente da negação lógica.

(e) Como poderia obter a informação acerca do número de gelados de que o `joao` gosta?

16. Considere o seguinte programa Prolog:

```
q(a).
q(b).
r(b,1).
r(a,2).
r(a,3).
p(X,Y) :- q(X), r(X,Y).
p(c,4).
pp(X,Y) :- q(X), !, r(X,Y).
pp(c,4).
```

Indique a sequência de respostas a cada uma das questões abaixo indicadas. Justifique a sua resposta desenhando as árvores de procura para cada questão:

(a) `p(A,B).`

(b) `pp(A,B).`

17. Uma empresa, para a produção dos seus artigos, necessita de determinadas quantidades de diferentes matérias primas. O preço de cada matéria prima pode diferir de fornecedor para fornecedor. Admitindo que esta informação está representada na base de conhecimento através de factos com a forma

```
artigo( Ref_art , [ (Mat_prima, Quantidade) , ... ] ).
fornecedor( Nome_forn , Mat_prima , Preco ).
```

defina:

(a) O predicado `minMat(+MatPrima, ?MinPreco)` que dada uma matéria prima, `MatPrima`, calcule o preço mínimo, `MinPreco`, dessa matéria prima.

(b) Usando o predicado da alínea anterior, escreva o predicado `custoMinimo(+Ref, ?CM)` que dado um artigo com referência `Ref`, calcule o custo mínimo em matéria prima, `CM`, que esse artigo pode ter.

18. Assuma que os resultados de jogos de futebol estão registados na base de conhecimento em factos da forma: `jogo(EquipaA, NumGolosA, EquipaB, NumGolosB)`.

(a) Defina os predicados `vencedor(?Equipa)` e `empate(?EquipaA,?EquipaB)` para gerar, por backtracking, as equipas que venceram jogos e as equipas que empataram jogos, respectivamente.

(b) Sabendo que uma vitória corresponde a 3 pontos para a equipa vencedora (e 0 pontos para a perdedora), e que um empate corresponde a 1 ponto para cada equipa, defina o predicado `totalPontos(-Tabela)` que produz em Tabela a lista de pares com o total de pontos obtidos por cada equipa.

19. Defina em Prolog os seguintes predicados sobre números primos (relembre que os operadores de divisão inteira e resto da divisão inteira são, respectivamente, `//` e `mod`).

    (a) Defina um predicado `geraPrimos(+N,?P)` para gerar, por backtracking, números primos sucessivos até `N`.

    (b) Use o predicado anterior para definir o predicado `factoriza(+N,?Factores)` que sucede se `Factores` é a lista de números primos que representa a factorização do número natural `N` (que deve ser superior a 1). Por exemplo:

    ```
    | ?- factoriza(520,L).
    L = [2,2,2,5,13] ?
    yes
    ```

20. Considere o seguinte programa Prolog:

    ```
    q(a).
    q(b).
    r(a,2).
    r(a,3).
    r(b,1).

    h(c,4).
    h(X,Y) :- q(X), r(X,Y).

    f(X,Y) :- !, r(X,Y), q(X).
    f(d,5).
    ```

    Indique a sequência de respostas a cada uma das questões abaixo indicadas. Justifique a sua resposta desenhando as árvores de procura para cada questão:

    (a) `h(A,B).`

    (b) `f(A,B).`

21. Assuma que temos na base de conhecimento factos da forma

    `class(`*Corrida*, *ListaPilotos*`)`

    que registam a ordem de chegada à meta dos pilotos numa determinada corrida. Por exemplo:

    ```
    class(monaco, [alonso, montoya, coulthard, barrichello, schumacher,
                   fisichelella, massa, albers]).
    class(unitedStates, [schumacher, barrichello, monteiro, karthikeyan,
    ```

```
                    albers, frisacher]).
class(imola, [massa, schumacher, barrichello, button, trulli, albers,
              montoya, alonso, webber, monteiro, speed, sato]).
```

(a) Defina o predicado `pontos(-Lista)` que produz a lista com a pontuação total de cada piloto. Note que, em cada corrida, o 1º calssificado recebe 10 pontos, o 2º recebe 8 pontos, o 3º recebe 6 pontos, e o 4º, 5º, 6º, 7º e 8º recebem, respectivamente, 5, 4, 3, 2 e 1 pontos. Os restantes pilotos recebem 0 pontos.

(b) Defina um programa que apresente no ecrã a tabela de pontuação (nº de pontos - piloto) ordenada por ordem decrescente de pontos (obs.: o predicado `print/1` imprime o argumento dado no ecrã).

22. Escreva um predicado Prolog `conta_termos(+Tree, +Term, ?N)` que sucede se, sendo `Tree` uma árvore binária de termos e `Term` um termo, `N` é o número de termos da árvore `Tree` que unificam com `Term`.

23. Assuma que temos na base de conhecimento factos da forma `nasc(Nome,Data)` indicando as datas de nascimento (dia, mês e ano) de diversas pessoas.

    (a) Defina o predicado `idade(?Nome,+Data,?Idade)` que sucede se `Idade` é a idade de `Nome`, na data `Data` (constituida por dia mês e ano).

    (b) Defina o programa `listaNomes(+A,+B,+Data)`, que escreve no ecrã o nome das pessoas com idades compreendidas entre `A` e `B` (calculadas na data `Data`), por ordem crescente de idades.

24. Escreva um predicado Prolog `conta_termos(+Tree, +Term, ?N)` que sucede se, sendo `Tree` uma árvore binária de termos e `Term` um termo, `N` é o número de termos da árvore `Tree` que unificam com `Term`.

25. Dado um mapa de ligações directas entre cidades, sob a forma de um conjunto de factos Prolog da forma `ligacao(Cidade1, Cidade2)`, escreva um programa Prolog para determinar (por backtracking) os caminhos, directos ou não, entre duas cidades. O programa deve evitar os ciclos existentes no mapa.