Teoria da Computação

Frequência

Resolução

Universidade da Beira Interior

Quinta-Feira 10 de Janeiro de 2008

Relembramos que, na tradição da axiomática de Peano, a notação $\mathbb{N}$ refere-se ao conjunto dos naturais incluindo o 0. Referiremo-nos ao conjunto dos naturais sem o 0 ($\{1, 2, 3 \ldots\}$) por $\mathbb{N}^*$.

Os tempos de resolução presentes em cada alínea são meramente indicativos e tem por objectivo ajudar-vos a planear a resolução.

1 Princípios da Teoria da Computação

(10 minutos) As noções de indecidibilidade e de não computabilidade estão ligadas a noção de conjunto infinito. Explique esta relação.

Solução:

►

- De forma informal um problema é indecidível quando não se sabe dar um resposta positiva ou negativa a todas as suas instâncias (instância = um caso particular do problema). A computabilidade é uma extensão simples da decidabilidade onde não se pretende encontrar um valor booleano (verdade, falso) mas sim um valor dum conjunto qualquer. Um problema não é computável quando não existe uma função total sobre o domínio considerado que *calcule* (i.e. via uma máquina de Turing) o valor pretendido.
- Fundamentalmente, tal acontece quando a procura do resultado necessita uma exploração completa do conjunto dos candidatos a solução (por exemplo, dos valores para os quais a resposta é positiva ou negativa).
- Nos casos do conjunto ser infinito, se não existirem alternativas computacionais, a procura está simplesmente condenada.
- Imagine por exemplo o caso da *satisfação em lógica de primeira ordem* (determinar se uma fórmula lógica é universalmente válida ou não - ou seja sistematicamente verdadeira ou não). Determinar esta propriedade por algoritmo implica poder explorar sistematicamente e em tempo finito os valores possíveis para qualquer variável de qualquer fórmula lógica de primeira ordem e verificar se a fórmula em questão é válida. Os conjuntos de valores são potencialmente infinitos e não existe heurísticas que no caso geral permitam evitar este tipo de exploração. O problema é por isso indecidível.

◄

2 Técnicas de Demonstração e OCaml

Considere o tipo *expr* das expressões aritméticas simples seguintes:

```
(* tipo expr onde:
   I = constante inteira, V = variável, A = +, S = −, M = ∗, D = /
*)
type expr = I of int | V of string | A of expr*expr
          | S of expr*expr | M of expr*expr | D of expr*expr

(* tipo dos ambientes (associação variável−valor)*)
type env = (string*int) list
```

Considere igualmente a função *eval* seguinte:

```
let rec eval (e:expr) (ambiente: env)=
  match e with
      I i → i
    | V v → assoc v ambiente
    | A (e1,e2) → eval e1 ambiente + eval e2 ambiente
    | S (e1,e2) → eval e1 ambiente − eval e2 ambiente
    | M (e1,e2) → eval e1 ambiente * eval e2 ambiente
    | D (e1,e2) → eval e1 ambiente / eval e2 ambiente
```

onde *assoc* é a função que devolve o valor inteiro associado a parâmetro v no ambiente *ambiente*, se este existir.

- (2 minutos) Qual é o princípio de indução associada a definição indutiva de *expr*?

  Solução:
  ▶

  de uma forma compacta:

  $$\begin{aligned}&\forall i \in \mathbb{N} P(I\ i) \wedge \forall x\ \textit{variável}, P(V\ x) \wedge \\ &(\forall e_1, e_2 \in \textit{expr}, P(e_1) \wedge P(e_2) \Rightarrow P(A\ e_1\ e_2) \wedge P(S\ e_1\ e_2) \wedge P(M\ e_1\ e_2) \wedge P(D\ e_1\ e_2)) \\ &\Rightarrow \forall e \in \textit{expr}, P(e)\end{aligned}$$

  ou seja se temos $P(V\ x)$ e $P(I\ i)$ para qualquer variável x e inteiro i e se para todos e_1 e e_2 expressões, $P(e_1)$ $P(e_2)$ implicam $P(A\ e_1\ e_2)$, $P(S\ e_1\ e_2)$, $P(M\ e_1\ e_2)$ e $P(D\ e_1\ e_2)$ então P é válido para todo o elemento de *expr* (ou seja $\forall e \in \textit{expr}, P(e)$)

  ◀

- (15 minutos) Defina uma função `simplify : expr →expr` que execute sobre toda a estrutura do seu parâmetro as transformações seguintes:

  para uma qualquer expressão e, $\begin{array}{ll} e+0=e & e-0=e \\ e*1=e & e/1=e \\ e+e=2*e & e-e=0 \end{array}$

  Por exemplo a expressão $\frac{((x+0)+x)}{1}$ se simplifica em $2*x$ porque, pelas regras definidas, $\frac{((x+0)+x)}{1}$ se transforma em $((x+0)+x)$, $x+0$ se transforma em x e $x+x$ se transforma em $2*x$. Repare que a ordem de aplicação destas simplificações é irrelevante se todas elas são de facto executadas.

  Solução:

▶

```
let rec simplify e =
match e with
     I i → I i
   | V v → V v
   | A (e1,e2) → let e'1,e'2 =  simplify e1, simplify e2 in
       if e'1 = I 0 then  e'2
       else if e'2 = I 0 then e'1
       else   if e'1=e'2
              then if e'1 = I 1 then I 2 (* uma pequena optimização não requerida...*)
                   else (M(I 2,e'1)))
              else  A(e'1,e'2)
   | S (e1,e2) → let e'1,e'2 =  simplify e1, simplify e2 in
       if e'2 = I 0 then e'1
       else if e'1 = e'2 then I 0 else S(e'1,e'2)
   | M (e1,e2) → let e'1,e'2 =  simplify e1, simplify e2 in
       if e'1= I 1 then  e'2
       else if e'2 = I 1 then e'1
       else if (e'1 = I 0) || (e'2 = I 0) then  I 0 (*não exigido....*)
       else M(e'1,e'2)
   | D (e1,e2) → let e'1,e'2 =  simplify e1, simplify e2 in
       if e'2= I 1 then e'1
       else if e'1 = e'2 then (I 1)  (*não exigido....*)
       else if e'2 = I 0 then failwith "divisão por zero" (*não exigido....*)
       else D (e'1,e'2)
```

◀

- (15 minutos) Demonstre por indução que $\forall e : expr, \forall a : env, eval\ (simplify\ e)\ a = eval\ e\ a$. Assuma para esse efeito e *se necessário* que o ambiente a tem todas as propriedades desejadas. Por exemplo, o ambiente tem todas as variáveis presentes na expressão considerada.

Solução:
▶
Provemos esta enunciado por indução sobre a estrutura do parâmetro e.

- Casos de base. Consideremos um inteiro i e uma variável x. É trivial verificar que por definição de $simplify$, $eval(simplify\ (V\ x)) = eval(V\ x)$ e $eval(simplify\ (I\ i)) = eval(I\ i)$.
- Passo indutivo. Em moldes gerais, as operações e simplificações operadas não alteram o resultado. É esse facto que vamos verificar. Vamos somente resolver o caso da soma, sendo os outros casos muito semelhantes (fica em exercício). Consideremos então e_1 e e_2 duas expressões.

  Hipótese de Indução: $eval(simplify\ e_1) = eval\ e_1$ e $eval(simplify\ e_2) = eval\ e_2$.

  Objectivo: provar que sob estas hipóteses temos necessariamente $eval(simplify\ (A\ e_1\ e_2) = eval\ (A\ e_1\ e_2)$.

  A parte do código que nos interessa aqui é:

```
   (...)
| A (e1,e2) → let e'1,e'2 =  simplify e1, simplify e2 in
        if e'1 = I 0 then  e'2
        else if e'2 = I 0 then e'1
        else
        if e'1=e'2 then
             if e'1 = I 1 then I 2
             else (M(I 2,e'1))
        else  A(e'1,e'2)
```

* Temos $simplify\ e_1 = e'_1$ e $simplify\ e_2 = e'_2$. Assim, pelas hipóteses de indução, sabemos que $eval\ e_1 = eval\ (simplify\ e_1) = eval\ e'_1$ e que $eval\ e_2 = eval\ (simplify\ e_2) = eval\ e'_2$.
* Por consequência $eval(A\ e_1\ e_2) = eval\ e_1 + eval\ e_2 = eval\ e'_1 + eval\ e'_2$. Resta agora saber se é igual a $eval\ (simplify\ (A\ e_1\ e_2))$.
* De facto o resultado de $(simplify\ (A\ e_1\ e_2))$ depende da forma de e'_1 e de e'_2.
  - Se $e'_1 = I\ 0$ então, por definição de $simplify$, $(simplify\ (A\ e_1\ e_2)) = e'_2$. Ora acontece que se $e'_1 = I\ 0$ então $eval\ e'_1 = I\ 0 = eval\ e_1$, por consequência $eval(A\ e_1\ e_2) = eval\ e_2 = eval\ e'_2 = eval\ (simplify\ (A\ e_1\ e_2))$. Caso resolvido.
  - Caso $e'_2 = I\ 0$. Idêntico ao caso anterior.
  - Caso $e'_1 = e'_2$. Neste caso $eval\ e'_1 = eval\ e'_2$ e $eval(A\ e_1\ e_2) = eval\ e'_1 + eval\ e'_2 = 2 \times eval\ e'_1$. que coincide com o resultado de $eval\ (simplify\ (A\ e_1\ e_2))$. No caso de $eval\ e'_1 = I\ 1$ esta propriedade mantém-se. Caso resolvido.
  - No caso geral (nenhum destes casos particulares ocorrem), $eval\ (simplify\ (A\ e_1\ e_2)) = eval\ (A\ e'_1\ e'_2) = eval\ e'_1 + eval\ e'_2 = eval\ (A\ e_1\ e_2)$. Caso resolvido.

QED.

◀

3 Autómatos Finitos e Linguagens regulares

1. (5 minutos) Considere o alfabeto $\Sigma = \{a, b\}$, defina um autómato *determinista* que reconheça a linguagem $\{w_1abw_2 \mid w_1, w_2 \in \{a, b\}^*\}$.

   Solução:
   ▶

   ver figura 1.

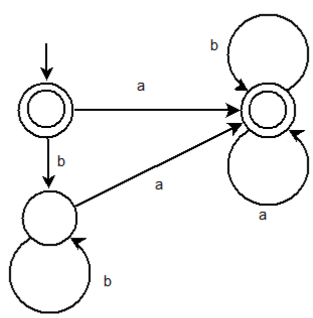

Figura 1: Autómato determinista reconhecendo a linguagem $\{w_1abw_2 \mid w_1, w_2 \in \{a, b\}^*\}$

◀

2. Considere o alfabeto $\Sigma = \{a, b\}$ e o autómato A_1 da figura 2:

   (a) (5 minutos) Minimize o autómato A_1.

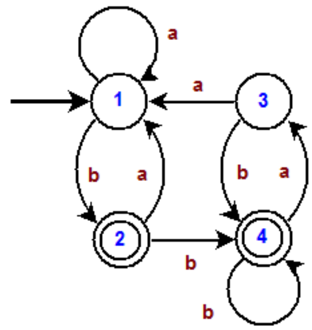

Figura 2: Autómato A_1

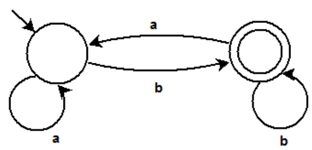

Figura 3: Versão minimal do autómato A_1

Solução:
▶
ver figura 3.
◀

(b) (1 minuto) Olhando para o autómato minimal resultante, descreva de forma sucinta e informal (i.e. em português) a linguagem aceite por A_1.

Solução:
▶
A linguagem é: qualquer sequência de a's e de b's terminada por uma sequência de pelo menos uma unidade de b's. Ou seja $(a+b)^*b^+$
◀

3. (15 minutos) Considere o autómato A_2 da figura 4.

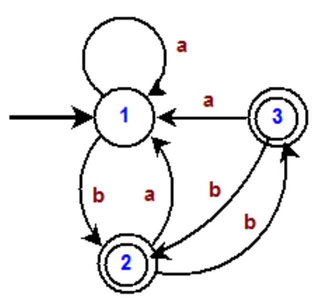

Figura 4: Autómato A_2

Utilize o algoritmo de *Mac Naughton-Yamada* para inferir que expressão/linguagem regular aceita este autómato. Pretende-se aqui que apresente *somente* o resultado final (a expressão regular calculada) e o valor de $R(1,3,2)$.

Solução:

▶

É preciso calcular $R(1,2,4) \cup R(1,3,4)$

Relembremos as seguintes propriedades $L \cup \emptyset = L$, $L.\emptyset = \emptyset$, $\emptyset^* = \emptyset$

$R(1,2,4) = R(1,2,3) \cup R(1,3,3).R(3,3,3)^*.R(3,2,3) = ((b+(\epsilon+a)(\epsilon+a)^*b)+(b+(\epsilon+a)(\epsilon+a)^*b)(\epsilon+a(\epsilon+a)^*b)^*(\epsilon+a(\epsilon+a)^*b))+((b+(\epsilon+a)(\epsilon+a)^*b)(\epsilon+a(\epsilon+a)^*b)^*b)(\epsilon+(b+a(\epsilon+a)^*b)(\epsilon+a(\epsilon+a)^*b)^*b)^*((b+a(\epsilon+a)^*b)+(b+a(\epsilon+a)^*b)(\epsilon+a(\epsilon+a)^*b)^*(\epsilon+a(\epsilon+a)^*b))$

$R(1,2,3) = R(1,2,2) \cup R(1,2,2).R(2,2,2)^*.R(2,2,2) = (b+(\epsilon+a)(\epsilon+a)^*b)+(b+(\epsilon+a)(\epsilon+a)^*b)(\epsilon+a(\epsilon+a)^*b)^*(\epsilon+a(\epsilon+a)^*b)$

$R(1,3,3) = R(1,3,2) \cup R(1,2,2).R(2,2,2)^*.R(2,3,2) = (b+(\epsilon+a)(\epsilon+a)^*b)(\epsilon+a(\epsilon+a)^*b)^*b$

$R(3,3,3) = R(3,3,2) \cup R(3,2,2).R(2,2,2)^*.R(2,3,2) = \epsilon+(b+a(\epsilon+a)^*b)(\epsilon+a(\epsilon+a)^*b)^*b$

$R(3,2,3) = R(3,2,2) \cup R(3,2,2).R(2,2,2)^*.R(2,2,2) = (b+a(\epsilon+a)^*b)+(b+a(\epsilon+a)^*b)(\epsilon+a(\epsilon+a)^*b)^*(\epsilon+a(\epsilon+a)^*b)$

$R(1,2,2) = R(1,2,1) \cup R(1,1,1).R(1,1,1)^*.R(1,2,1) = b+(\epsilon+a)(\epsilon+a)^*b$
$R(2,2,2) = R(2,2,1) \cup R(2,1,1).R(1,1,1)^*.R(1,2,1) = \epsilon+a(\epsilon+a)^*b$
$R(1,3,2) = R(1,3,1) \cup R(1,1,1).R(1,1,1)^*.R(1,3,1) = \emptyset$
$R(2,3,2) = R(2,3,1) \cup R(2,1,1).R(1,1,1)^*.R(1,3,1) = b$
$R(3,3,2) = R(3,3,1) \cup R(3,1,1).R(1,1,1)^*.R(1,3,1) = \epsilon$
$R(3,2,2) = R(3,2,1) \cup R(3,1,1).R(1,1,1)^*.R(1,2,1) = b+a(\epsilon+a)^*b$
$R(1,2,1) = \{b\}$
$R(1,1,1) = \{\epsilon, a\} = \epsilon+a$
$R(2,2,1) = \{\epsilon\}$
$R(2,1,1) = \{a\}$
$R(1,3,1) = \emptyset$
$R(2,3,1) = \{b\}$
$R(3,3,1) = \{\epsilon\}$
$R(3,1,1) = \{a\}$
$R(3,2,1) = \{b\}$

$R(1,3,4) = R(1,3,3) \cup R(1,3,3).R(3,3,3)^*.R(3,3,3) = ((b+(\epsilon+a)(\epsilon+a)^*b)(\epsilon+a(\epsilon+a)^*b)^*b)+((b+(\epsilon+a)(\epsilon+a)^*b)(\epsilon+a(\epsilon+a)^*b)^*b)(\epsilon+(b+a(\epsilon+a)^*b)(\epsilon+a(\epsilon+a)^*b)^*b)^*(\epsilon+(b+a(\epsilon+a)^*b)(\epsilon+a(\epsilon+a)^*b)^*b)$

◀

4. (10 minutos) Demonstre que a linguagem $\{a^{3^n} \mid n \in \mathbb{N}\}$ não é regular.

Solução:

▶

As palavras da linguagem em questão são sequências de a's, de comprimento igual a 3^n para um n qualquer. Informalmente, é preciso mostrar que não há, para palavras suficientemente grandes, padrão que se possa repetir sem quebrar esta restrição sobre o comprimento da sequência.

Vamos proceder como é habitual: Demonstração por contradição usando o lema de bombeamento sobre as linguagens regulares.

A hipótese da demonstração por contradição é $L = \{a^{3^n} \mid n \in \mathbb{N}\}$ é uma linguagem regular. Seja N o limite de tamanho de palavra referido no lema de bombeamento (como é usual, o numero de estados do autómato mínimo que reconhece L).

Neste caso, consideremos uma palavra w suficientemente grande ($|w| = m \geq N$). Consideremos $x, y \in \{a\}^*$ e $u \in \{a\}^+$ tais que $w = \underbrace{aa\ldots a}_{m=3^i} = xuy$ com $|xu| \leq N$.

Verifiquemos que nenhuma escolha adequada de x, u, y permite satisfazer a conclusão do lema de bombeamento.

Temos a situação seguinte: $\overbrace{\underbrace{a\dots a}_{3^i}}^{w} = \overbrace{\underbrace{a\dots a}_{j}}^{x}\overbrace{\underbrace{a\dots a}_{k}}^{u}\overbrace{\underbrace{a\dots a}_{l}}^{y}$ com $|x| = j$, $|u| = k > 0$ e $|y| = l$, $j + k \leq N$.

Ou seja, dado $i \in \mathbb{N}$ tal que $|w| = 3^i$, temos $3^i = j + k + l$. A questão é saber se existe uma escolha adequada de (j, k, l) tal que $\forall o \in \mathbb{N}, \exists p \in \mathbb{N}, j + o \times k + l = 3^p$.

Vamos considerar o caso de $o > 0$, em particular o caso $o = 2$. Se $|w| = 3^i$, a próxima palavra de L é $w' = www$ e $|w'| = 3^{i+1}$. Ou seja, visto que $w = xuy$ ($u \neq \epsilon$ e $|u| \leq |w|$), não há hipótese nenhuma de consegui produzir uma palavra de L quer seja w' quer seja as palavras seguintes de L só repetindo o padrão u duas vezes (no melhor dos casos obtemos ww). Contradição.

L não é regular. QED.

◀

4 Gramáticas

1. (10 minutos) Defina uma gramática livre de contexto que gere a linguagem seguinte $\{a^n b^m \mid n, m \in \mathbb{N}, n \neq m\}$.

   Solução:
   ▶
   Uma forma possível de resolução passa pela seguinte constatação:

   $\{a^n b^m \mid n, m \in \mathbb{N}, n \neq m\} = \{a^n b^m \mid n, m \in \mathbb{N}, n < m\} \cup \{a^n b^m \mid n, m \in \mathbb{N}, n > m\}$

   Para a primeira componente podemos propor a seguinte gramática:

   $$S_1 \quad \to aS_1b \mid bS_1 \mid b$$

   Para segunda componente, temos:

   $$S_2 \quad \to aS_2b \mid aS_2 \mid a$$

   Ou seja no final temos:

   $$\begin{array}{ll} S & \to S_1 \mid S_2 \\ S_1 & \to aS_1b \mid bS_1 \mid b \\ S_2 & \to aS_2b \mid aS_2 \mid a \end{array}$$

   ◀

2. Considere a gramática cujas as produções são as seguintes:

$$\begin{array}{lcl} S & \to & AB \\ A & \to & a \\ A & \to & AS \\ B & \to & CD \\ C & \to & b \\ D & \to & c \end{array}$$

e a palavra $w = aabcbc$.

(a) (10 minutos) Utilize o algoritmo CYK para verificar que a palavra w é reconhecida pela gramática.

Solução:
▶

$$
\begin{aligned}
&N_{1,1} = \{A\}\\
&N_{2,2} = \{A\}\\
&N_{3,3} = \{C\}\\
&N_{4,4} = \{D\}\\
&N_{5,5} = \{C\}\\
&N_{6,6} = \{D\}\\
&N_{1,2} = \emptyset\\
&N_{2,3} = \emptyset\\
&N_{3,4} = \{B\}\\
&N_{4,5} = \emptyset\\
&N_{5,6} = \{B\}\\
&N_{1,3} = \emptyset\\
&N_{2,4} = \{S\}\\
&N_{3,5} = \emptyset\\
&N_{4,6} = \emptyset\\
&N_{1,4} = \{A\}\\
&N_{2,5} = \emptyset\\
&N_{3,6} = \emptyset\\
&N_{1,5} = \emptyset\\
&N_{2,6} = \emptyset\\
&N_{1,6} = \{S\}
\end{aligned}
$$

Conclusão: S pertence a $N_{1,6}$. Logo $aabcbc$ é reconhecido pela gramática.

Uma versão matricial da resolução encontra-se na figura 5

◀

(b) (2 minutos) Dê a árvore de derivação de w.

Solução:
▶

ver figura 6

◀

5 Autómatos de pilha

(10 minutos) Defina um autómato com pilha que reconheça a linguagem $\{a^n.b^{3n+m}.c^m \mid n, m \in \mathbb{N}\}$. Sugestão: define um autómato que utilize Z como símbolo inicial de pilha.

Figura 5: CYK

Solução:
►

ver figura 7

◄

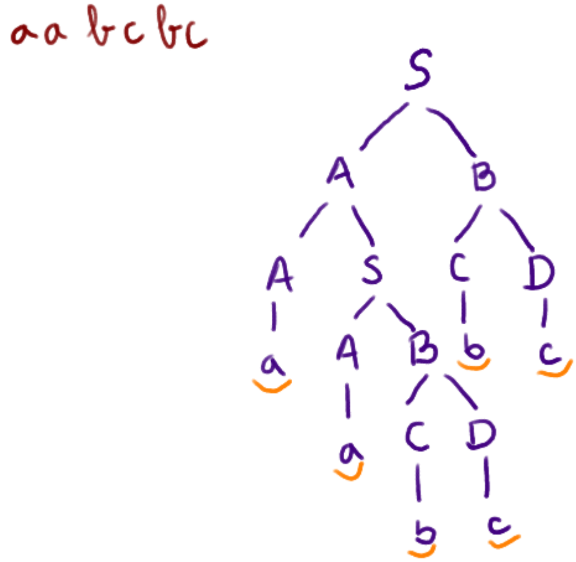

Figura 6: Árvore de derivação de *aabcbc*

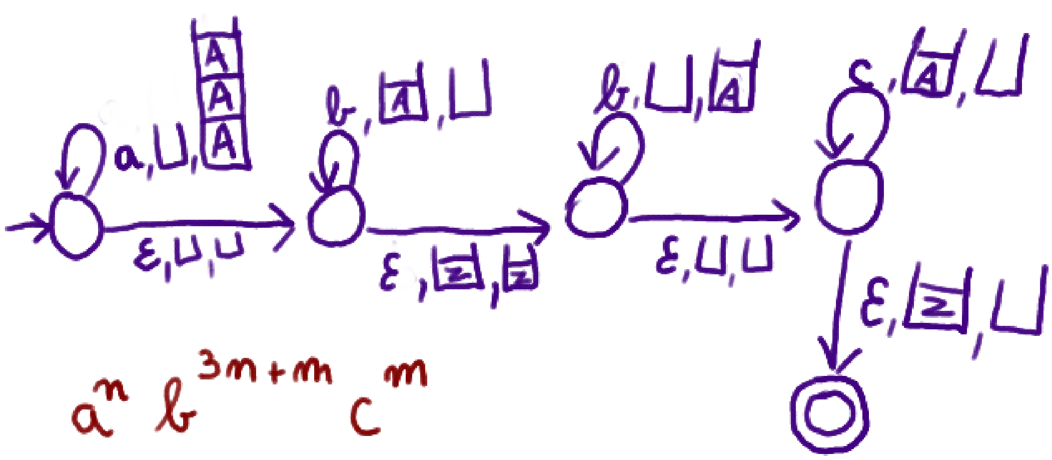

Figura 7: Autómato reconhecedor para $\{a^n.b^{3n+m}.c^m \mid n, m \in \mathbb{N}\}$