TEORIA DA COMPUTAÇÃO

Prova 3 – 08/08/2016 – Prof. Marcus Ramos

1. (1,5 ponto) Defina:
   a) Classe P;

      Conjunto de todas as linguagens que podem ser decididas por Máquinas de Turing determinísticas de tempo polinomial.

   b) Classe NP;

      Conjunto de todas as linguagens que podem ser decididas por Máquinas de Turing não-determinísticas de tempo polinomial, ou então verificadas em tempo polinomial.

   c) Classe NPC.

      Conjunto de todas as linguagens de NP para as quais existem reduções de tempo polinomial de todas as demais linguagens pertencentes à classe NP.

2. (1,0 ponto) Defina "verificador para uma linguagem A".

   Um verificador é um algoritmo que recebe como entrada a descrição de um problema e uma proposta de solução. Ele então determina se a proposta é uma solução de fato para o problema. Um verificador polinomial faz isso em tempo polinomial com o tamanho da entrada. Um verificador aceita a sua entrada quando a proposta é uma solução para o problema em questão, ou a rejeita em caso contrário.

3. (2,0 pontos) Dada uma linguagem decidível L, como determinar que:
   a) $L \in P$?

      Apresentando uma MT determinística de tempo polinomial que decide L, ou uma redução de tempo polinomial de L para outra linguagem pertencente à P.

   b) $L \in NP$?

      Apresentando uma MT não-determinística de tempo polinomial que decide L, ou então um verificador de tempo polinomial para L.

   c) $L \in NPC$?

      Provando que todas as linguagens de NP reduzem em tempo polinomial para L, ou então apresentando uma redução de tempo polinomial de alguma outra linguagem de NPC para L.

   d) $L \notin P$?

      Não se sabe. Existem apenas evidências (por exemplo quando uma linguagem pertence à NPC), mas não existe um método conhecido para isto até o momento.

4. (1,0 ponto) Em que consiste e para que são usadas reduções de tempo polinomial?

Uma redução de tempo polinomial entre duas linguagens L_1 e L_2 é uma função f tal que $w \in L_1 \Leftrightarrow f(w) \in L_2$ e cujo tempo de execução é polinomial com o comprimento de w. A importância das reduções de tempo polinomial reside no fato de que elas podem ser usadas para provar que uma linguagem L_1 pertence à P. Para isto, basta obter uma redução de tempo polinomial de L_1 para alguma outra linguagem L_2 pertencente à P. A combinação da solução de tempo polinomial de L_2 com a redução de tempo polinomial produz uma solução de tempo polinomial para L_1.

5. (1,5 ponto) Obtenha formas normais para os seguintes termos lambda:
   a) (λz.z)(λy.yy)(λx.xa)

      (λz.z) (λy.y y) (λx.x a)
      (λy.y y) (λx.x a)
      (λx.x a) (λx.x a)
      (λx.x a) a
      aa

   b) (λxy.xyy)(λa.a)b

      (λx.λy.x y y) (λa.a) b
      (λy.(λa.a) y y) b
      (λa.a) b b
      bb

   c) (((λxy.(xy))(λy.y))w)

      (((λx. λy.(x y))(λy.y)) w)
      (((λx. λa.(x a))(λy.y)) w)
      ((λa.((λy.y) a)) w)
      (λy.y) w
      w

6. (1,5 ponto) Considere a definição recursiva apresentada a seguir para a operação de exponenciação, e obtenha um termo lambda que represente tal operação. Considere dados os termos que efetuam a subtração e a multiplicação, assim como o operador lógico que testa se um número é zero e o comando condicional (representados respectivamente por <u>sub</u>, <u>mult</u>, <u>zero</u> e <u>if</u>).

   exp (a,b) = a^b =
   1 se b = 0
   a * a^{b-1} se b > 0

   Inicialmente, devemos representar a operação na forma de uma igualdade de termos lambda (supondo que a e b sejam Numerais de Church):

   xyz $=_\beta$ <u>if</u> (<u>zero</u> z) <u>1</u> (<u>mult</u> y (x y (<u>sub</u> z <u>1</u>)))

   Agora, basta usar o operador de ponto fixo Y e construir o termo que representa a solução de exp:

   <u>exp</u> ≡ Y (λxyz. <u>if</u> (<u>zero</u> z) <u>1</u> (<u>mult</u> y (x y (<u>sub</u> z <u>1</u>))))

7. (1,5 ponto) Cite dois corolários do Teorema de Soctt-Curry e faça considerações acerca da importância dos mesmos para a teoria da computação.

O Teorema de Scott-Curry diz que nenhum par de conjuntos não-vazios de termos lambda que sejam fechados em relação à operação de conversão são recursivamente separáveis. Os corolários apresentados em sala de aula foram:

i. Não existe algoritmo para determinar se um certo termo pertence ou não pertence à um conjunto de termos lambda fechados em relação à operação de conversão;
ii. Não existe algoritmo para determinar se um termo lambda possui ou não possui forma normal;
iii. Não existe algoritmo para determinar se dois termos lambda representam a mesma operação.

Estes foram os primeiros problemas indecídiveis provados na história. No primeiro caso, que o problema do pertencimento não tem solução. No segundo caso, que não é possível determinar se um programa produz ou não um resultado. No terceiro caso, que não é possível determinar se dois programas computam ou não a mesma função.