Segunda Prova de Teoria da Computação — Prof. José de Oliveira Guimarães
DComp — Campus de Sorocaba da UFSCar

Ao final da prova, entregue APENAS a folha de respostas. A folha de questões não será considerada. Utilize quantas folhas de reposta forem necessárias. Não é necessário entregar as questões em ordem. Utilize lápis ou caneta, mas faça letra legível. Na correção, símbolos ou palavras ilegíveis não serão considerados. **Justifique** todas as respostas.

Se você estudou com algum colega para esta prova, não se sente ao lado dele pois é possível que acidentalmente vocês produzam respostas semelhantes para alguma questão. Provas de alunos que fizeram a prova próximos uns dos outros com respostas semelhantes caracterizam cópia de questões. Lembro-os de que todos os envolvidos na utilização de métodos ilegais na realização desta prova receberão zero de nota final da disciplina (e não apenas nesta prova).

Coloque o seu nome na folha de resposta, o mais acima possível na folha, seguido do número da sua coluna de carteiras. A primeira carteira é a mais perto da porta e a última a mais perto das janelas. Não precisa colocar o RA.

1. (2,5) Uma MTU (Máquina de Turing Universal) U toma um único parâmetro da forma $\langle M\rangle 2x$. Baseado em uma MTU U, responda:

(a) o que é $\langle M\rangle$? Responda suscintamente, não é necessário dar todos os detalhes. Dica: isto é uma MT? uma sequência de símbolos? Um número real?

(b) se $\langle M\rangle = \langle N\rangle$ então $M = N$?

(c) o que $U(\langle M\rangle 2x)$ retorna? Ela sempre retorna?

(d) pode-se chamar $U(\langle U\rangle 2\langle U\rangle)$? Responda apenas sim ou não com uma justificativa sumária.

2. (2,0) Responda:

(a) (1,0) cite a tese de Church-Turing, não é necessário explicá-la;

(b) (1,0) cite um tipo de máquina de Turing com limitações que não tem todo o poder de uma MT convencional mas tem pelo menos o poder de decisão de um autômato finito. Há inúmeras destas máquinas no exercício 1 da lista de exercícios. Justifique suscintamente (uma frase ou duas).

3. (2,0) Em uma linguagem CST (C Sem Tipos), cada função pode ser mapeada em um número inteiro. Suponha que exista, nesta linguagem, uma função `Para(x, y)` que retorna `1` se a função correspondente a `x` pára quando toma a entrada `y`. E `Para(x, y)` retorna `0` se a função correspondente a `x` não pára quando toma a entrada `y`. Usando a função `Q` dada abaixo, encontre uma contradição quando se chama `Q` passando o número de `Q` como parâmetro.

```
void Q( T ) {
   while ( Para(T, T) )
      ;
}
```

Escolha e faça DUAS e apenas DUAS das questões abaixo.

4. (2,0) Uma MT M_2 de duas fitas infinitas em ambas as direções pode ser simulada por uma MT M_1 de uma única fita que é infinita em apenas uma das direções. Mostre como as duas fitas de M_2 podem ser mapeadas na única fita de M_1. Nota: se você usar M_1 com uma fita infinita em ambas as direções, esta questão valerá 1,5 pontos. Você pode responder baseado em um desenho das três fitas envolvidas com uma explicação curta. Mas deve ficar claro exatamente qual o mapeamento que você está usando.

5. (2,0) Explique como funciona uma MTND. Na sua resposta, cite a diferença entre o conjunto de instruções de uma MTD e uma MTND, faça uma comparação de quem é mais poderosa (MTND ou MTD ou equivalentes), cite um exemplo de uma MTND (bem simples).

6. (2,0) Prove que, se linguagens L e K são computáveis, então $L \cup K$ e $L \cap K$ são computáveis. Faça explicitamente as MT´s que garantem a resposta.

7. (2,0) Prove que, se as linguagens L e L^c são computavelmente enumeráveis, então L e L^c são computáveis.

8. (2,0) Defina o problema $P = NP$ e cite uma linguagem $L \in NP$ tal que não sabemos se $L \in P$.

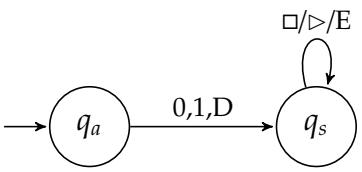
9. (2,0) Porque a complexidade em tempo de uma MT é sempre maior ou igual à sua complexidade em espaço? Descreva uma MT que tem complexidade em tempo $f(n) = 4n$ e complexidade em espaço $g(n) = n$. Apenas descreva como ela funciona, não é necessário fazer a representação gráfica.

10. (2,0) O que é Computabilidade? Explique as relações deste conceito com linguagens, subconjuntos de $\mathbb{N}$ e números reais. E cite pelo menos um teorema de Computabilidade.

11. (2,0) Prove que existe uma linguagem L que não pode ser decidida por uma MT.

12. (2,0) Prove que $H = \{\langle M \rangle 2x : M(x) \downarrow\}$ é computacionalmente enumerável.

13. (2,0) Mostre como é feita a codificação da seguinte MT:



Resumo:

Uma máquina de Turing determinística (MT ou MTD) é uma quadrupla (Q, Σ, I, q) na qual Q e Σ são conjuntos chamados de conjuntos de estados e de símbolos, I é um conjunto de instruções, $I \subset Q \times \Sigma \times Q \times \Sigma \times D$, $D = \{-1, 0, 1\}$ e $q \in Q$ é chamado de estado inicial. Há dois estados especiais: q_s e q_n, todos elementos de Q e todos diferentes entre si. Neste texto convenciona-se que o estado inicial será q_0 a menos de menção em contrário. Exige-se que $\{0, 1, \triangleright, \sqcup, \square\} \subset \Sigma$. Uma instrução é da forma (q_i, s_j, q_l, s_k, d) na qual $s_k \neq \square$ e $q_i \notin \{q_s, q_n\}$. Se $(q, s, q'_0, s'_0, d_0), (q, s, q'_1, s'_1, d_1) \in I$, então $q'_0 = q'_1$, $s'_0 = s'_1$ e $d_0 = d_1$. Q, Σ e I são conjuntos finitos.

O símbolo $\square$ é o branco utilizado para as células ainda não utilizadas durante a execução e $\sqcup$ é utilizado para separar dados de entrada e saída.

Símbolos: Máquina de Turing Não Determinística (MTND), Máquina de Turing (MT), Máquina de Turing Determinística (MTD). A menos de menção em contrário, uma MT é uma MTD. Todas as linguagens utilizadas são subconjuntos de $\{0, 1\}^\star$.

Uma linguagem L é computável (recursiva) se existe uma MT M de decisão tal que

$$x \in L \text{ sse } M(x) = 1$$

Isto é, M decide L.

Uma linguagem L é ou computacionalmente enumerável, c.e. (recursivamente enumerável, r.e.) se existe uma MT M tal que se $x \in L$ então $M(x) = 1$. Se $x \notin L$, $M(x) \uparrow$. Dizemos que M *aceita* L.

Uma linguagem L pertence a TIME(f) se existe uma MT M de decisão que toma uma entrada x e que termina a sua execução depois de um número de passos menor ou igual a $cf(n)$ tal que $x \in L$ sse $M(x) = 1$. Ou seja, M executa em tempo $cf(n)$. Uma linguagem L pertence a SPACE(f) se existe uma MT M que toma uma entrada x e que termina a sua execução depois de utilizar um número de células menor ou igual a $cf(n)$ nas fitas de trabalho (todas exceto a de entrada e a de saída). Ou seja, M executa em espaço $cf(n)$. NTIME(f) e NSPACE(f) têm definições análogas, mas que usam MTND´s.

$$\text{P} = \bigcup_{k \in \mathbb{N}} \text{TIME}(n^k)$$

$$\text{NP} = \bigcup_{k \in \mathbb{N}} \text{NTIME}(n^k)$$

Dizemos que uma linguagem L é polinomialmente redutível ou Karp-redutível a uma linguagem K se existe uma MT R que executa em tempo polinomial tal que

$$x \in L \text{ sse } R(x) \in K$$

Usaremos $L \leqslant_P K$ para "L é polinomialmente redutível a R". A MT R é chamada de redução de L para K.

Dada uma classe de linguagens C, d Dizemos que uma linguagem L é NP-completa se toda linguagem $K \in NP$ pode ser reduzida polinomialmente a L e $L \in$NP.