Segunda Prova de Teoria da Computação — Prof. José de Oliveira Guimarães
DComp — Campus de Sorocaba da UFSCar, 2014

Ao final da prova, entregue APENAS a folha de respostas. A folha de questões não será considerada. Utilize quantas folhas de respostas 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. (3,0) Uma "variação de uma MT" é uma forma de se definir uma MT de maneira diferente da original. Por exemplo, podemos ter variações em que:

(a) a fita é infinita apenas à direita;

(b) a cabeça de leitura/gravação só pode se movimentar para a direita. As variações (a) e (b) são independentes.

Baseado nisto, prove ou mostre evidências de que:

(a) (1,5) a variação (a) possui um número de células na fita equivalente ao número de células de uma MT normal. Não é necessário provar que tudo o que pode ser feito em uma pode ser feito na outra. A resposta pode ser um desenho das duas fitas com a relação, dada por setas, entre as células;

(b) (1,5) a variação (b) tem menos poder computacional do que as MT´s normais. Na sua resposta, deixe claro qual tipo de máquina a variação (b) é equivalente (AF, AF com pilha) em relação à decisão de linguagens.

2. (3,0) Sobre máquinas de Turing universais (MTU´s), responda:

(a) (1,5) assumindo que exista **uma** MTU U, prove que existem infinitas MTU´s;

(b) (1,5) pelo teorema da recursão, pode-se assumir que uma máquina de Turing M tem acesso à sua própria codificação $\langle M \rangle$. Suponha que uma MT M tenha dentro dela todas as instruções de uma MTU e faça o seguinte, dada uma entrada x:

- testa se x é 0. Se for, retorna 1;

- senão faz o seguinte: M escreve $\langle M\rangle$ na fita (pode fazer isto pelo teorema da recursão), $\sqcup$ e $x-1$, nesta ordem. Então M passa o controle para as instruções de M que simulam uma MTU. Após a simulação da MTU terminar, o resultado, que foi colocado na fita, é multiplicado por x. Este valor é retornado por M.

Pergunta-se: o que M faz? Justifique detalhadamente.

3. (2,0) Prove que, se L e L^c são computacionalmente enumeráveis, L é computável; isto é, decidível. Dê todos os detalhes possíveis na sua resposta.

4. (2,5) Sobre complexidade, responda:

(a) (1,0) por que $P \subset NP$?

(b) (1,5) todas as linguagens da classe NP são redutíveis a SAT. E SAT é redutível à linguagem H, que contém todas as codificações de grafos Hamiltonianos (assuma isto). Prove que, se $L \in NP$, L é redutível a H usando apenas as informações dadas neste item. Isto é, você pode assumir apenas que todas as linguagens de NP são redutíveis a SAT e que SAT é redutível a H.

Dizemos que K_1 é redutível a K_2 se existe uma MT R que executa em tempo polinomial tal que $x \in K_1$ sse $R(x) \in K_2$.

5. (2,5) Há uma proposição que diz que a linguagem $H = \{\langle M\rangle 2x : M(x)\downarrow\}$ não é computável; isto é, não pode existir uma MT que a decide. Parte da prova desta proposição é a seguinte, em itálico:

Provaremos que não pode existir uma máquina de Turing que toma $\langle M\rangle \sqcup x$ como entrada e retorna 1 *de $M(x)\downarrow$ ou* 0 *se $M(x)\uparrow$. Em resumo, provaremos que não pode existir uma MT que toma a codificação de uma MT M e um x como entrada e diz se M irá parar ou não com a entrada x.*

Provaremos por contradição. Assumiremos que exista uma MT P que decida H e chegaremos a uma contradição. Isto é, suponha que $P(\langle M\rangle \sqcup x)$ retorna 1 *se M pára a sua execução com entrada x ou retorna* 0 *se $M(x)\uparrow$. Máquinas de Turing que decidem linguagens sempre param. Então P sempre pára a sua execução.*

Usando P, podemos construir uma MT Q que toma uma entrada $\langle M\rangle$ e faz o seguinte:

(a) a partir da entrada $\langle M\rangle$, produz $\langle M\rangle \sqcup \langle M\rangle$;

(b) P é simulado passando-se $\langle M\rangle \sqcup \langle M\rangle$ como entrada. A simulação pode ser feita colocando-se as instruções de P em Q ou simulando-se P como é feito em uma MT Universal;

(c) se P retornar 1*, isto significa que $\langle M\rangle \sqcup \langle M\rangle \in H$. Ou seja, a MT M pára a sua execução com a entrada $\langle M\rangle$. Neste caso, Q entra em um laço infinito, nunca parando a sua execução;*

(d) se P retornar 0*, isto significa que $\langle M\rangle \sqcup \langle M\rangle \notin H$. Ou seja, a MT M não pára a sua execução com a entrada $\langle M\rangle$. Neste caso, Q pára a sua execução.*

Este é o fim da prova parcial de que H não é computável. Complete o resto desta prova.

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 $\{0, 1, \rhd, \sqcup, \square\} \subset \Sigma$. 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.

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

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

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, dizemos que uma linguagem L é NP-completa se toda linguagem $K \in NP$ pode ser reduzida polinomialmente a L e $L \in$NP.