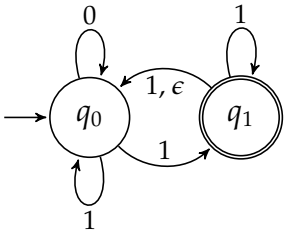
Prova Substitutiva de Teoria da Computação — Prof. José de Oliveira Guimarães
DComp — Campus de Sorocaba da UFSCar, 2015

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. (2,5) Explique o funcionamento de um autômato finito não determinístico (AFND) utilizando o exemplo abaixo. Nesta explicação, obviamente, deve ser mostrada a diferença para um autômato finito determinístico (AFD). Converta o autômato abaixo para um AFD utilizando as regras do Sipser (obrigatoriamente). Estas regras dizem para criar, no AFD, um estado para cada subconjunto de estados do AFND. O estado inicial do AFD é o estado inicial do AFND e os estados finais do AFD são os estados que contém algum estado final do AFND.



2. (3,75) Nesta questão serão utilizadas as linguagens $L_1 = \{0^n1 | n \in \mathbb{N}\}$, $L_2 = \{a^n0b^n | n \in \mathbb{N}\}$, $L_3 = \{a^nwb^n | n \in \mathbb{N} \text{ e } w \in L_1\}$. Responda às questões abaixo sobre gramáticas, expressões regulares e linguagens.

(a) (2,25) Faça gramáticas G_1, G_2 e G_3 tal que $L(G_1) = L_1$, $L(G_2) = L2$ e $L(G_3) = L_3$.

(b) (0,5) Explique, informalmente mas precisamente, por que um autômato finito não pode reconhecer L_2.

(c) (1,0) Faça expressões regulares (e.r.) que reconheçam L_1 e L_2 (duas e.r. diferentes) ou explique porque isto não é possível (há quatro possibilidades: as duas são possíveis, apenas uma ou nenhuma).

3. (2,5) Esta é uma questão diferente pois já possui, além da pergunta, a resposta da questão. Contudo, a resposta está errada (queria o que?). A sua tarefa, na folha de respostas, é apontar TODOS os erros da resposta.

Questão: Seja M_0 uma MT que, dada uma entrada x, percorre x da esquerda (onde inicialmente está a cabeça de leitura/gravação) para a direita e, ao atingir o □ após o símbolo de x mais à direita, substitui este □ por ▷ e volta uma célula. Se esta célula contiver 0, imprime 1 e vai para o estado q_s e, se contiver 1 imprime 0 e vai para o estado q_n. Pergunta-se:
(a) qual a complexidade em tempo de M_0 e a qual classe de complexidade ela pertence?

(b) sendo K a linguagem decidida por M_0, $K \in P$? P está definida ao final da prova, nos resumos.

Respostas:
(a) A complexidade em tempo de M_0 é $O(n)$ pois M_0 percorre todas as x células da entrada mais uma constante c_1 de células (nota do professor: c_1 pode ser deixado não especificado mesmo, isto NÃO é um erro). Então $M_0 \in \text{TIME}(f(n))$.

(b) K é decidida por uma MT em tempo n, $|n| = x$. Então $M_0 \in \text{TIME}(n)$, $K \subset \text{TIME}(n)$ e $K \in P$.

4. (1,5) Sobre MT universais, responda:

(a) cite a Tese de Church-Turing.

(b) esta Tese afirma que qualquer dispositivo que faz computações e que possa ser fisicamente implementado (não apenas imaginado) é tão poderoso quanto um computador atual (a menos na capacidade de memória)?

Escolha e faça UMA e apenas UMA das questões abaixo.

5. (2,25) Seja K uma linguagem decidida por uma MT M_k e L uma linguagem decidida por uma MT M_L. $L \subset K$ e $L \neq K$. Prove que $K - L$ é uma linguagem decidível construíndo uma MT M que a decide.

6. (2,25) Sejam L_1, L_2 e L_3 linguagens computacionalmente enumeráveis por MTs M_1, M_2 e M_3. Sabendo-se que $L_1 \cup L_2 \cup L_3 = \Sigma^\star$, prove que L_1 é computável. Se fossem apenas duas linguagens L e L^c, a prova seria: construa uma MT M que executa as MTs ... um passo por vez, alternadamente ... como x (a entrada) pertence ou a L ou a L^c, pelo menos uma ... iria parar ...

7. (2,25) Explique como se faz a codificação de uma MT. Dê um pequeno exemplo com dois estados. Você não necessariamente precisa seguir as instruções da apostila. Apenas dê uma forma de codificação que seja realmente uma codificação. Então a função de codificação $f : T \longrightarrow \mathbb{N}$, sendo T o conjunto de todas as MTs, deve ser injetora e, dado $n \in \mathbb{N}$, deve ser possível

saber se n corresponde a alguma MT e, se corresponder, que MT é esta.

Resumo:

Uma expressão regular sobre um alfabeto Σ é descrita como: a) x é uma e.r. (expressão regular) se $x \in \Sigma$; b) ϵ é uma e.r. c) $\emptyset$ é uma e.r. d) $x?$, $(x \cup y)$, (xy) (concatenação de x e y) e $(x^\star)$ são e.r. se x, y são e.r. Assume-se que a ordem de precedência dos operadores seja, do maior para o menor: $\star$, concatenação e união ($\cup$). Parenteses podem ser removidos se forem redundantes. A linguagem associada a uma expressão regular é definida indutivamente como $L(x) = \{x\}$ se $x \in \Sigma$, $L(\epsilon) = \{\epsilon\}$, $L(\emptyset) = \emptyset$, $L(R_1 \cup R_2) = L(R_1) \cup L(R_2)$, $L(R_1 \circ R_2) = L(R_1) \circ L(R_2)$, $L(R_1^\star) = (L(R_1))^\star$, $L(R?) = L(R) \cup \{\epsilon\}$.

A concatenação de duas linguagens L e K é $L \circ K = \{vw : v \in L \text{ e } w \in K\}$.

A definição de uma gramática pode ser feita com a seguinte sintaxe:

$A \longrightarrow 0A$

$A \longrightarrow BC$

$B \longrightarrow bB|\epsilon$

$C \longrightarrow c$

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. A saída de uma MT é o conjunto de símbolos da célula corrente até o $\triangleright$ que está à direita (deve existir alguma célula à direita da célula corrente com este símbolo).

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.

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