Prova de SAC de Teoria da Computação — 2016/2
Prof. José de Oliveira Guimarães — DComp — Campus de Sorocaba da UFSCar

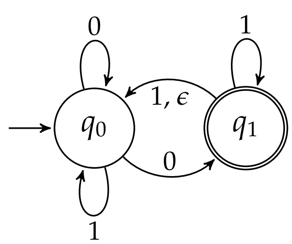
Ao final da prova, entregue APENAS a folha de respostas. A folha de questões não será considerada. Utilize quantas folhas de reposta forem necessárias. Não é necessário entregar as questões em ordem. Utilize lápis ou caneta, mas faça letra legível. Na correção, símbolos ou palavras ilegíveis não serão 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. (1,5) Esta questão utiliza o seguinte autômato finito:



Baseado nele,

(a) (0,75) faça a árvore de computação deste autômato com a entrada 101;

(b) (0,25) explique porque este autômato não é determinístico. Um motivo é suficiente. Apenas cite o motivo, não é preciso justificar;

(c) (0,5) cite as condições necessárias e suficientes para que um autômato finito não determinístico aceite uma entrada.

2. (2,5) Seja E a expressão regular $(a^\star \cup bb) * c?ab$ e $H = \{a^n c^m 0 d^m a^n \mid n \geqslant 0, m \geqslant 1\}$. Baseado nestes dados,

(a) (1,0) faça uma gramática G_E tal que $L(G_E) = L(E)$ e uma gramática G_H tal que $L(G_H) = H$. Não é necessário justificar;

(b) (0,5) usando as gramáticas já feitas no item anterior, faça gramáticas G_1 e G_2 tal que $L(G_1) = L(G_E) \cup L(G_H)$ e $G_2 = L(G_E) \cdot L(G_H)$. O $\cdot$ é a concatenação de linguagens;

(c) (1,0) explique, intuitivamente, porque H não pode ser reconhecida por um autômato finito;

3. (1,0) Sejam M e M' dois autômatos finitos. Mostre como obter um autômato que reconheça $L(M) \cup L(M')$. Faça apenas o desenho como está no Sipser, com retângulos com três círculos representando autômatos.

4. (1,5) Faça os seguintes itens sobre máquinas de Turing (MT).

(a) (0,75) Escreva a tese de Church-Turing.

(b) (0,75) Explique porque $TIME(n^2) \subset SPACE(n^2)$. A explicação pode ser intuitiva, mas precisa.

5. (1,5) Faça uma MT S que produz 0 como saída se a sua entrada tiver mais de um dígito e 1 caso a entrada tenha apenas um dígito. Então $S(0) = S(1) = 1$ e $S(00) = S(001) = S(10) = S(00000) = 0$. Assuma que a entrada desta MT seja uma sequência de zeros e uns.

6. (2,0) M é uma MT que toma dois números como entrada: a codificação de uma MT R e uma entrada x para R. M produz 0 como saída se R com entrada x produz 0 como saída e M produz 1 como saída se R com entrada x produz qualquer outro número como saída. Isto é,

$$M(\langle R \rangle, x) = 0 \text{ se } R(x) = 0$$

$$M(\langle R \rangle, x) = 1 \text{ se } R(x) > 0$$

Observe que não necessariamente M utiliza uma MT Universal U para simular a execução de R. M poderia apenas analisar as instruções de R e prever o que ela faria com a entrada x.

Prove que M não pode existir. Dica: faça uma MT P que invoca M e faz o contrário do que M diz que P fará. Mas dê detalhes, seja preciso na sua definição. Será necessário chamar P passando $\langle P \rangle$ como parâmetro.

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.

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