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

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.

(2,0) Faça UMA e apenas UMA das próximas três questões

1. Seja LREG o conjunto das linguagens regulares, LLC o conjunto das linguagens livres de contexto, LR o conjunto das linguagens decididas por máquinas de Turing, LRE o conjunto das linguagens recursivamente enumeráveis, LAF o conjunto das linguagens reconhecidas por autômatos finitos, LAP o conjunto das linguagens reconhecidas por autômatos com pilha e K uma linguagem decidida por um autômato não determinístico. Faça todas relações ($\in$, $=$ e $\subset$) possíveis entre os conjuntos citados acima. Justifique.

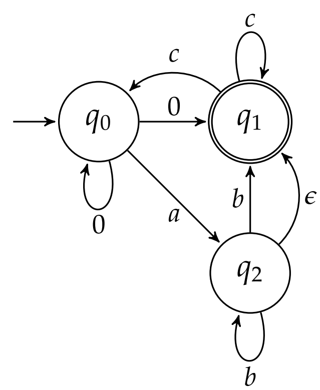
2. A função δ de um autômato finito não determinístico $N = (\{q_0, q_1, q_2, q_3\}, \{0, 1, 2, 3\}, \delta, q_0, \{q_3\})$ é dada abaixo.
$\delta(q_0, 1) = \{q_0, q_1\}$
$\delta(q_0, 0) = \{q_1, q_2\}$
$\delta(q_1, 2) = \{q_1\}$
$\delta(q_1, 1) = \{q_3\}$
$\delta(q_2, 3) = \{q_3\}$

Baseado neste autômato, faça:

(a) a árvore de computação de N com entrada 1121;

(b) uma gramática G tal que $L(G) = L(N)$.

3. Esta questão utiliza o seguinte autômato finito N:



Baseado neste autômato,

(a) (1,0) faça a árvore de computação da cadeia $abc0$. Não é necessário justificar;

(b) (1,0) se $N = (Q, \Sigma, \delta, q_0, F)$, defina Q, Σ, F e $\delta(q_2, b)$, $\delta(q_1, c)$;

Terminaram as questões opcionais. A partir de agora faça todas as questões a menos de menção em contrário

4. (1,5) Prove que, se A e B são linguagens regulares, também o são $A \cup B$ e $A^\star$. Você pode utilizar diagramas na prova; isto é, a prova pode ser informal. Um diagrama de um autômato finito (porquê AF e não MT? justifique) é um retângulo contendo uma bolinha representando o estado inicial e duas bolinhas representando estados finais.

5. (1,5) Faça gramáticas que gerem as seguintes linguagens:

(a) $L = \{a^n 00 b^n | n \geqslant 1\}$

(b) $K = L \cup \{1^n w | w \in L \text{ e } n \geqslant 1\}$

6. (2,0) Explique o que é uma MT Universal. A sua resposta deve responder às seguintes questões:

(a) o que uma MT universal U toma como entrada?

(b) o que produz como saída? Ela sempre produz uma saída?

(c) A codificação de uma máquina de Turing é utilizada em algum lugar? O que é mesmo codificação? Não é preciso explicar com detalhes esta última pergunta, apenas defina o que é codificação.

(1,5 cada) Escolha e faça DUAS e apenas DUAS das questões abaixo.

7. Seja S o conjunto das linguagens sobre $\{0, 1\}$ e T o conjunto de todas as máquinas de Turing (ou codificações no conjunto $\mathbb{N}$ de todas as MTs). Pergunta-se:

(a) S é equipotente a qual conjunto? Não é preciso justificar;

(b) T é equipotente a qual conjunto? Não é preciso justificar;

(c) $S \sim T$? Justifique. O que isto significa?

8. Suponha que $SAT \leqslant_P K$ para certa linguagem $K \in$ NP. Prove que K é NP-completa. Se houver um algoritmo polinomial para decidir SAT então haverá um algoritmo polinomial que decide K? E o contrário?

9. Faça uma MT que toma um número binário como entrada (terminando com $\square$) e retorna 1, terminando no estado q_s, se a entrada é um palíndromo e retorna 0, terminando no estado q_n, se a entrada não é um palíndromo. Diga qual é o tempo de execução deste algoritmo. Justifique a resposta desta última pergunta.

10. Prove que, se L é decidível, L é computacionalmente (recursivamente) enumerável. Uma linguagem L é recursivamente enumerável se existe uma MT M tal se, se $x \in L$, $M(x) = 0$ e, se $x \notin L$, $M(x) \uparrow$.

11. (2,0) Prove que, se as linguagens L e L^c são computavelmente enumeráveis, então L e L^c são computáveis.

12. (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) )
      ;
}
```

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.

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