Prova de SAC de Teoria da Computação
Campus de Sorocaba da UFSCar
17 de dezembro de 2010
Prof. José de Oliveira Guimarães

Entregue apenas a folha de respostas. As questões não precisam estar em ordem e podem ser respondidas à lápis ou caneta. Na correção, símbolos ou palavras ilegíveis não serão considerados. Justifique todas as respostas a menos de menção em contrário.

Coloque o seu nome na folha de resposta, o mais acima possível na folha. Não é necessário colocar o RA.

No final da prova há um pequeno resumo da matéria.

1. (1,0) Enuncie o teorema de Church-Turing. Explique-o.

2. (3,0) Esta questão se baseia na seguinte gramática na qual as letras em minúscula e dígitos são terminais.

```
S ::= A          S ::= B
A ::= a A b      A ::= C
C ::= C a        C ::= c
B ::= 0 D        D ::= d D
D ::= d
```

Chamaremos esta gramática de G e a linguagem gerada por ela de L(G). Baseado em G, faça:

(a) a descrição de L(G) em termos de conjuntos, algo do tipo $\{0^n 1 a b^k : n > 0 \text{ e } k \geqslant 0\}$;

(b) a expressão regular que gera a mesma linguagem que esta gramática;

(c) o autômato com pilha que decide a linguagem L(G).

3. (2,5) Prove que o conjunto R que contém todas as linguagens regulares é fechado sobre as operações de: a) união e b) concatenação. Isto é, dadas L e K linguagens regulares, $L \cup K$ e $L \circ K = \{wt : w \in L \text{ e } t \in K\}$ pertencem a R. Dica: faça o desenho de autômatos "genéricos" que decidem as linguagens L e K e ...

4. (1,5) Uma MT executa por t passos. Então no máximo quantas células diferentes da fita ela utilizou? Isto é, qual, no máximo, o espaço utilizado pela MT? Diga qual o relacionamento, usando $\subset$, $\in$ e $\notin$, entre as classes TIME(n) e SPACE(n). Mais geralmente, qual o relacionamento entre TIME(f) e SPACE(f)? Justifique.

5. (2,5) Considere uma função `Para` que decide $H = \{<M> \sqcup x : M(x) \downarrow\}$; isto é, `Para(<M>, x)` retorna `1` se $<M> \sqcup x \in H$ e `0` caso contrário. Note que usamos dois parâmetros para `Para` — isto não trás nenhuma má consequência. Uma função `Q` utiliza `Para`:

```
int Q(X) {
   while ( Para(X, X) )
      ;
}
```

Baseado em `Q`, explique porquê `Para` não pode existir; isto é, H não é recursiva. Dica: `Q(<Q>)`.

Resumo

Um autômato finito M é uma 5-tupla $(Q, \Sigma, \delta, q_0, F)$ no qual Q é um conjunto finito de estados, Σ é o alfabeto, $\delta : Q \times \Sigma \longrightarrow Q$ é a função de transição, $q_0 \in Q$ é o estado final e $F \subset Q$ e o conjunto de estados de aceitação.

Sendo R_1 e R_2 expressões regulares sobre Σ, uma expressão regular (e.r.) sobre um alfabeto Σ é definida indutivamente como: a) x é e.r. para $x \in \Sigma$ b) ϵ é e.r. c) $\emptyset$ é e.r. d) $(R_1 \cup R_2)$ é e.r. e) $(R_1 \circ R_2)$ é e.r. f) $(R^\star$ é e.r.

Sendo Σ um conjunto finito de símbolos, uma cadeia sobre Σ é a concatenação de um conjunto finito de símbolos de Σ. Definimos $\Sigma^n = \{a_1 a_2 \ldots a_n : a_i \in \Sigma, 1 \leqslant i \leqslant n\}$, o conjunto de todas as cadeias sobre Σ de tamanho n. Usamos ϵ para a cadeia com zero elementos. Logo $\Sigma 0 = \{\epsilon\}$. Definimos $\Sigma^\star$ como

$$\bigcup_{n \geqslant 0} \Sigma^n$$

Uma linguagem L sobre Σ é um subconjunto de $\Sigma^\star$.

Uma máquina de Turing M é uma 7-tupla $(Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R)$ na qual Q, Σ, Γ são conjuntos finitos. Q é um conjunto de estados, Σ é o alfabeto de entrada ($\sqcup \in \Sigma$), Γ é o alfabeto da fita ($\sqcup \in \Gamma$ e $\Sigma \subset \Gamma$), $\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R, S\}$ é a função de transição, $q_0 \in Q$ é o estado inicial, q_A é o estado de aceitação e q_R é o estado de rejeição. Sempre que o estado corrente for q_A ou q_R a máquina pára (e estes são os únicos estados finais). Para facilitar as provas, assuma que sempre que o estado corrente for q_A o valor de retorno da máquina será 1. Idem para q_R e 0.

Uma MT de decisão sempre termina o seu processamento e retorna 0 ou 1. A menos de menção em contrário, todas as MT aceitam um inteiro em binário como entrada. Uma MT M decide uma linguagem L sobre Σ se M é uma MT de decisão e $x \in L$ se e somente se $M(x) = 1$. Uma linguagem L sobre Σ é recursivamente enumerável se existe uma MT M tal que

$$x \in L \Longrightarrow M(x) = 1$$

$$x \notin L \Longrightarrow M(x) \uparrow$$

Uma linguagem $L \in$ NP se existe uma MT não determinística N que decide L em tempo polinomial; isto é, se $x \in L$, então existe uma sequência de escolhas não determinísticas na computação $N(x)$ de tal forma que o resultado seja $N(x) =^1$. Uma linguagem $L \in$ P se existe uma MT M que decide L em tempo polinomial. Isto é, $L \in \text{TIME}(n^k)$ para algum $k \in \mathbb{N}$. SAT é a linguagem $\{< \varphi >: \varphi$ está na FNC e é satisfazível $\}$. SAT é NP-completa. Isto é, $SAT \in NP$ e para toda $L \in$ NP, $L \leqslant_P SAT$ ($L \leqslant_P K$ se existe uma MT R que executa em tempo polinomial tal que $x \in L$ sse $R(x) \in K$). A relação $\leqslant_P$ é transitiva: se $L_1 \leqslant_P L_2$ e $L_2 \leqslant_P L_3$ então $L_1 \leqslant_P L_3$.

$A \sim B$ se existe uma função bijetora entre A e B.