UNIVERSIDADE DE SÃO PAULO - ICMC
Departamento de Ciências de Computação e Estatística
SCC 205 Teoria da Computação e Linguagens Formais - TURMA B
PROFA: Sandra Aluisio

Prova 1 – 13/9/2011

Questão 1) (Valor 2.0) Considere a gramática G = ({R,S,T,X}, {a,b}, P, R)

P = { R -> XRX | S
S -> aTb | bTa
T -> XTX | X | λ
X -> a | b
}

a. Dê 3 exemplos de cadeias que pertencem a L(G)
b. Dê 3 exemplos de cadeias que **não** pertencem a L(G)
c. Responda Verdadeiro ou Falso para as afirmações abaixo:
    1. T => aba
    2. T =>* aba
    3. T => T
    4. T =>* T
    5. XXX =>* aba
    6. X =>* aba
    7. T =>*XX
    8. T =>* XXX
    9. S =>* λ
d. Dê a linguagem L(G)

Questão 2) (Valor 2.5) Um requisito importante de uma linguagem de programação, que é geralmente definida através de uma GLC, é que ela não seja ambígua, ou que, pelo menos, qualquer ambigüidade seja imediatamente evidente e facilmente evitada.

Nós deveríamos, portanto, ser capazes de examinar a definição de uma linguagem e decidir se ela é ou não ambígua. Entretanto, o problema de se decidir se uma GLC é ambígua é insolúvel. Mas, em casos particulares nós podemos reconhecer a ambigüidade nas gramáticas.

PEDÊ-SE:

a) Mostre que a gramática G1 abaixo é **ambígua**.
b) Construa uma **gramática equivalente a G1 não ambígua** cujos operadores * (fecho), concatenação e + são associados à esquerda e * tem a maior prioridade, a concatenação a segunda maior prioridade e + tem a menor prioridade.

G1 = ({R},{a,b,c,(,),+,*},P1,R)

P1 = {R → R + R | RR | R* | (R) | a| b | c }

c) Construa uma gramática (**em notação EBNF**) não ambígua que gere todas as expressões booleanas válidas, envolvendo aos seguintes operadores que são associados à esquerda (exceto negação que é associado à direita) e possuem ordem crescente de prioridade:

≡ (equivalência)

→ (implicação)

∨ (ou)

∧ (e)

¬ (negação)

os operandos **a**, **b**, **c**, e as **expressões parentizadas**.

Exemplo de sentença: **¬ (a ∨ b) ≡ ¬ a ∧ ¬b**

Questão 3) (Valor 2.5) Considere a seguinte linguagem L = {w | w ∈ {0,1}* e |w| >= 2}

a) Construa o **AFND** que reconhece a linguagem L.
b) Usando o **Teorema que prova a Equivalência entre AFD e AFND**, construa o **AFD** para a L(G), mostrando os passos.
c) Construa a **Expressão Regular** para a linguagem L, usando o Teorema 3.11 que prova a equivalência entre ER e AF, em **ANEXO. MOSTRE** todos os passos do teorema**.**

Questão 3) (Valor 2.0) Considere as 3 tarefas abaixo:

i) **Crie** uma gramática para a representação de números reais, na notação de ponto fixo e ponto flutuante, como nos seguintes exemplos:

5.35	**8 E 10**	**3.3 E 5**	**8 E - 10**
+ 53.3	**+ 5 E -10**	**3.3 E – 5**	**-8 E - 10**
– 029.9	**+ .3 E + 5**	**3.3 E + 5**	**-8 E 10**
-8 E +10	**– 3. E – 5**	**.3 E 5**	**8 E + 10**

isto é, números com ou sem sinais na notação de ponto fixo ou na notação de ponto flutuante, sejam eles começando com ponto, terminando com ponto ou tendo dígitos antes e depois do ponto. Inteiros somente **não** são considerados reais aqui (**56**, por exemplo, **não** deve ser gerado), mas pode fazer parte da expressão real como em -8 E +10

ii) **Classifique-a** segundo a hierarquia de Chomsky vista em classe, **justificando** sua resposta.
iii) **Mostre** a derivação da sentença: **- 5.8 E + 6**

Questão 4) (Valor 1.0) Sobre Expressões Regulares (ER):

a) Prove ser verdadeira ou falsa a afirmação sobre expressão regular: (R + S)* = R* + S*

b) **Sendo Vt = {a,b}**, para cada uma das ER abaixo apresente 2 cadeias que pertencem à linguagem denotada por elas e 2 cadeias que **não** pertencem à linguagem denotada por elas:

1) a* b*
2) (aaa)*
3) a* + b*
4) a(ba)*b
5) Vt*aVt*bVt*aVt*

Teorema 3.11 "Seja r uma expressão regular sobre o alfabeto Σ. Então existe um autômato finito M que aceita a linguagem definida por r." O autômato finito não-determinístico que aceita a linguagem definida por r pode ser obtido através da aplicação do Algoritmo 3.11, que especifica as regras de mapeamento **parciais** que abrangem casos triviais de sentenças (casos 1, 2 e 3) e cada um dos operadores de união (caso 4), concatenação (caso 5) e fechamento (caso 6), conforme a própria definição das expressões regulares (Ramos, Neto e Veja, 2009):

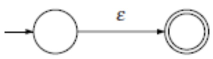
Algoritmo 3.11 *"Obtenção de um autômato finito a partir de uma expressão regular."*

- *Entrada: uma expressão regular r sobre um alfabeto Σ;*
- *Saída: um autômato finito M tal que $L(M) = r$;*
- *Método:*

Caso 1:

$r = \varepsilon$

r é aceita por M representado na Figura 27.



Caso 2:

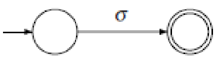
$r = \emptyset$

r é aceita por M representado na Figura 28.

Caso 3:

$r = \sigma, \sigma \in \Sigma$

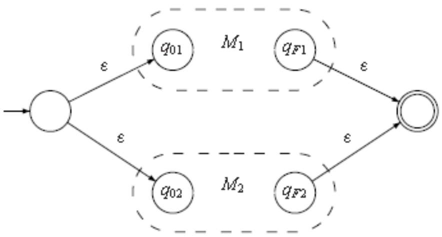
r é aceita por M representado na Figura 29.



Caso 4:

$r = r_1 \mid r_2$

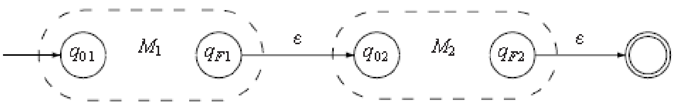
r é aceita por M representado na Figura 30.



Caso 5:

$r = r_1 r_2$

r é aceita por M representado na Figura 31.



Caso 6:

$r = r_1^*$

r é aceita por M representado na Figura 32.