Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação

SCC-0605
TEORIA DA COMPUTAÇÃO E COMPILADORES

Lista de Exercícios do Capítulo 3

Gramáticas e Linguagens Sensíveis ao Contexto

1. Seja o seguinte conjunto de produções de uma gramática livre de contexto G:

$$P = \{S \rightarrow AB,\ A \rightarrow 0,\ A \rightarrow 1,\ A \rightarrow \lambda,\ B \rightarrow 1\}$$

a) Descreva $L(G)$;
b) $L(G)$ é sensível ao contexto?
c) Se possível, ache um autômato finito que processe $L(G)$;

Autômatos Limitados Linearmente e Máquinas de Turing

2. Construa uma MT com fita limitada (ALL) que reconheça a linguagem $a^{2n}bc^n$, $n >= 1$. Ilustre sua operação com o reconhecimento da sentença aabc.

3. Especifique um ALL que reconheça a linguagem 0*1+2*. Mostre os movimentos que conduzem a aceitação da cadeia 0012 e à rejeição da cadeia 022.

4. Construa ALLs que reconheçam as seguintes linguagens:
   a. a(bc)*a* | a*(b*c | bc*)a
   b. o conjunto de anagramas que podem ser obtidos a partir de uma palavra qualquer, construída sobre o alfabeto {a,b,...,z}, que é fornecida como entrada. Considere que a cadeia a ser analisada possui a forma palavra1-palavra2. A cadeia deve ser aceita se palavra2 for anagrama de palavra1. Caso contrário, deverá ser rejeitada
   c. o conjunto das expressões aritméticas que podem ser definidas sobre o alfabeto {a,b,c,+,*, -,/,(,)}
   d. $a^m b^n c^m d^n$, m >=1, n <> m
   e. $a^m b^n c^p d^q$, m >= 1, n > m, p > n, q > p

5. Considere a linguagem $L = \{w \mid w \in (a + b)^*$ com número par de *a*'s$\}$. Por exemplo, a cadeia *abbabaa* seria aceita, enquanto que a cadeia *baabba* não.

a) Se possível, escreva um autômato limitado linearmente (ALL) que processe *L*. Caso não seja possível, explique o porquê.

b) Se possível, escreva um autômato de pilha (APN) que processe L. Caso não seja possível, explique o porquê.
c) Qual é o tipo de L? Comente a sua resposta.

6. Seja o seguinte conjunto de produções da gramática G:

$S \rightarrow aSBC|aBC$
$CB \rightarrow BC$
$aB \rightarrow ab$
$bB \rightarrow bb$
$bC \rightarrow bc$
$cC \rightarrow cc$

a) Qual o processador de linguagem de menor poder computacional capaz de processar $L(G)$ (AFN, APD, ALL ou MT)? Por que?
b) Escreva este processador.

7. Seja o seguinte conjunto de produções da gramática livre de contexto GA:

$S \rightarrow aaZcc$
$Z \rightarrow aZc$
$Z \rightarrow b$

Observe agora o seguinte conjunto de produções da gramática linear a direita G_B:

$S \rightarrow aA$
$A \rightarrow aB$
$B \rightarrow aB \mid bC$
$C \rightarrow cC \mid cD$
$D \rightarrow c$

Qual a relação entre G_A e G_B? São equivalentes? Por que? Escreva a máquina de Turing que processa $L(G_A)$.

8. Considere a gramática G = ({a,b}, {S,A,B}, S, P}, onde P é o conjunto de produções:

$S \rightarrow aAa \mid bBb$
$A \rightarrow b$
$B \rightarrow aA$

a) Ache o autômato limitado linearmente que processe $L(G)$, se possível. Se não for possível, explique o porquê.
b) Ache a máquina de Turing de uma cabeça que processe $L(G)$, se possível. Se não for possível, explique o porquê.

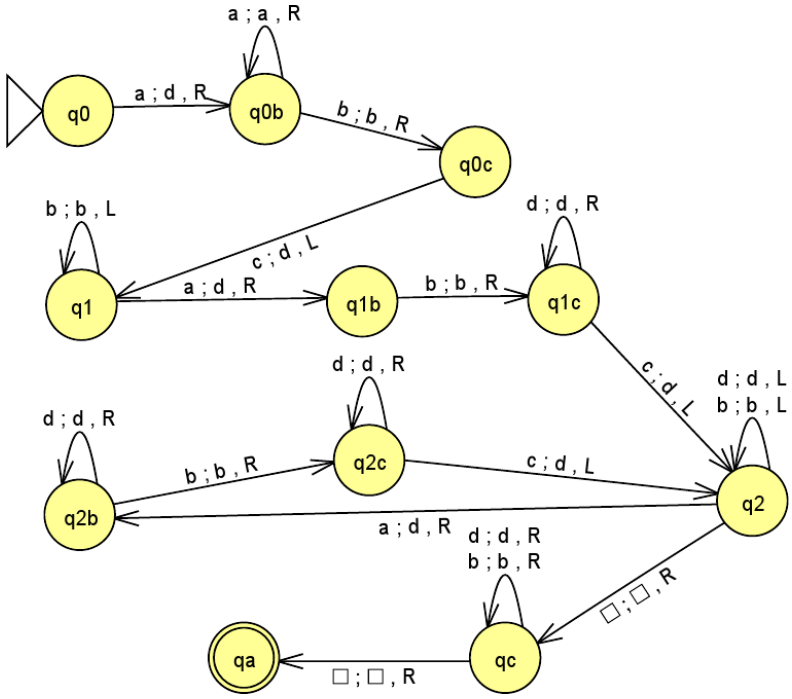
9. Seja o seguinte autômato finito ({q0, q1}, {0,1}, δ, q0, {q0}):

δ	**0**	**1**
$\boldsymbol{q_0}$	q_1	q_1
$\boldsymbol{q_1}$	q_1	q_0

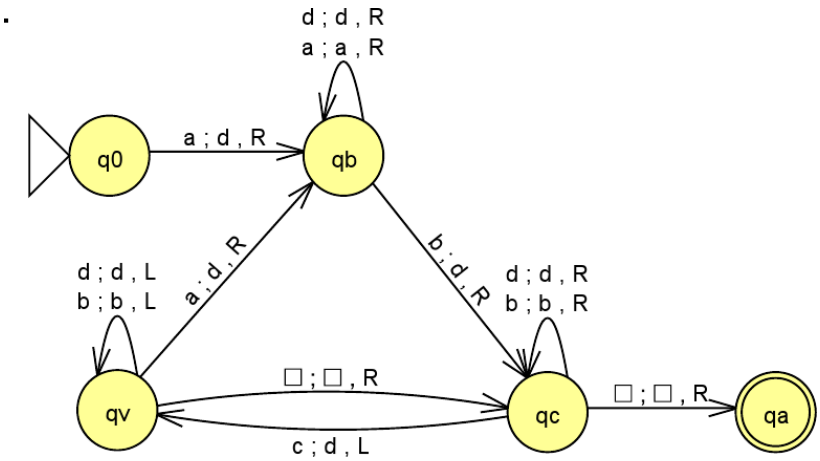
Escreva a máquina de Turing *T* equivalente. Se não for possível, explique o porquê.

10. Considere uma gramática G = (Σ, V, S, P), onde Σ = {0, 1}, V = {S, A, B}, P = {S → 0A | 1B | 0, A → 0A | 0S | 1B, B → 1B | 1 | 0}. Qual é a Máquina de Turing que processa L(G)?

11. Seja a Máquina de Turing MA, representada no JFlap:



Seja a Máquina de Turing M_B, representada no *JFlap*:



A partir do conjunto de instruções:

a) É possível afirmar que $T(M_A)$, ou seja, o conjunto de cadeias aceitas pela Máquina de Turing M_A, é regular?
b) E quanto à $T(M_B)$?
c) Se não forem regulares, quais os tipos das linguagens processadas por M_A e por M_B?
d) Escreva os processadores de menor poder computacional que processa $T(M_A)$ e $T(M_B)$.

12. Seja T a máquina de Turing:

$T = (\{q_0, q_1, q_2, q_3\}, \{a, [,], \#\}, q_0, \{q_3\}, \delta)$

onde δ é dado por:
$\delta(q_0, a) = (q_0, a, R)$
$\delta(q_0, \#) = (q_0, \#, R)$
$\delta(q_0,]) = (q_1, \#, R)$
$\delta(q_1,]) = (q_1, [, R)$
$\delta(q_1, \#) = (q_1, \#, R)$
$\delta(q_1, [) = (q_2, \#, L)$
$\delta(q_2, x) = (q_2, x, L)$ para todo $x \neq a$
$\delta(q_2, a) = (q_0, a, R)$
$\delta(q_0, B) = (q_3, \#, R)$

Quais palavras da forma *aw*, onde *w* está em {[,]}*, são aceitas? Você pode achar uma gramática para esta linguagem?

13. Escreva uma máquina de Turing que aceite a linguagem (a + b)*, na qual há menos a's do que b's.

14. Escreva uma máquina de Turing que aceite a linguagem (a + b)* onde existe mais a's que b's.

15. Escreva uma Máquina de Turing que aceite a linguagem (a + b)*, na qual há pelo menos um par de a's.

16. Sabe-se que um autômato finito (AFD/AFN) processa linguagem linear a direita (regular) e que um autômato a pilha (APN), que é equivalente a um AFN + pilha, processa linguagem livre de contexto.

Afirmação: "Qualquer máquina de Turing pode ser simulada por algum APN com duas pilhas."

Comente esta afirmação.

17. Construa a máquina de Turing que aceite o conjunto de todas as sentenças que contenham dois 0s consecutivos ou dois 1s consecutivos. Teste para 010110.

18. Considere a seguinte máquina de Turing T que reconhece a LLC L = $\{0^n1^n \mid n \geq 1\}$. Seja T = (Q, Σ, q0, qa, δ), onde

$$Q = \{q_0, q_1, \ldots, q_5\}$$
$$\Sigma = \{0, 1, Y, Z\}$$
$$q_a = q_5$$

sendo que *Y* e *Z* são símbolos da fita, mas não símbolos de entrada. δ é dado por:

1) $\delta(q_0, 0) = (q_1, Y, R)$

(*T* irá alternativamente substituir um 0 por *Y*, então um 1 por *Z*. No estado q_0, um 0 é substituído por um *Y*, e *T* move para a direita no estado q_1 procurando um 1.)

2) a) $\delta(q_1, 0) = (q_1, 0, R)$

b) $\delta(q_1, Z) = (q_1, Z, R)$

c) $\delta(q_1, 1) = (q_2, Z, L)$

(*T* se move para a direita no estado q_1 (regras 2a e 2b). Quando um 1 é encontrado, ele é mudado para um *Z*, e o estado se torna q_2 (regra 2c). Em q_2, vemos que *T* se move para a esquerda, procurando por um 0 para converter para um *Y*. Movendo para a esquerda, *T* encontrará um bloco de *Z*'s, então talvez um bloco de 0's, então um *Y*.)

3) a) $\delta(q_2, Z) = (q_2, Z, L)$

b) $\delta(q_2, Y) = (q_3, Y, R)$

c) $\delta(q_2, 0) = (q_4, 0, L)$

(*T* se move para a esquerda através de *Z*'s (3a). Se *T* encontra um *Y* enquanto no estado q_2, não há mais 0's para converter. *T* vai para o estado

q_3 para checar que não há mais 1's (3b). Se um 0 é encontrado, *T* vai para o estado q_4 e se move para a esquerda para converter o 0 mais a esquerda (3c).)

4) a) $\delta(q_4, 0) = (q_4, 0, L)$

b) $\delta(q_4, Y) = (q_0, Y, R)$

(*T* se move através de 0's (4a). Se um *Y* é encontrado, *T* passou o 0 mais a esquerda e então deve mover para a direita, para converter o 0 em um *Y*. Entra no estado q_0 e o processo descrito nas regras 1 a 4 se repete (regra 4b).)

5) a) $\delta(q_3, Z) = (q_3, Z, R)$

b) $\delta(q_3, B) = (q_5, Z, R)$

(*T* entra no estado q_3 quando não houver mais 0's (veja 3a). *T* deve mover à direita (5a). Se um branco for encontrado antes de um 1, então não há mais 1's (5b). A entrada está em *L* e *T* entra no estado q_5, o estado de aceitação.)

6) δ é indefinida, para outros casos diferentes de 1 a 5 acima.

Verifique como *T* age com a entrada 000111 através de transições entre descrições instantâneas.

Compiladores (Análises Sintática e Semântica)

19. Considere a gramática a seguir:

G=({S, E, C, N}, {if, then, not, +, teste, comando}, P , S), em que P é

<S> ::= if <E> then <C> | if <N> <E> then <C>
<E> ::= <E> + teste | λ
<C> ::= comando
<N> ::= not | λ

Responda/faça:

(a) Calcule os conjuntos Primeiro e Seguidor para os símbolos não terminais.
(b) A gramática é LL(1)? Justifique sua resposta. Se não for LL(1), transforme-a.

20. Considere a gramática a seguir:

G=({S, E, C}, {do, while, comando, true}, P, S), em que P é

<S> ::= do <E> while <C>
<E> ::= comando | λ
<C> ::= true

Faça:

(a) Considere a análise sintática descendente não recursiva. Construa a tabela sintática para a gramática anterior.
(b) Utilizando a tabela anterior, reconheça a cadeia **do while true**

21. Sobre a tabela de símbolos:

a) As duas estruturas de dados mais usadas para implementar Tabelas de Símbolos em compiladores comerciais são **árvores de busca binária** e **tabelas hash**. Descreva cada uma delas e diga quais as suas vantagens e desvantagens.
b) Mostre os descritores dos seguintes tipos de dados, com todas as informações necessárias para a realização da checagem de tipos:

```
typedef enum A {segunda, terca, quarta, quinta, sexta, sabado, domingo};

typedef int B[10][4];

typedef struct
{
      int X;
      float Y;
} C;
```

22. Sobre Análise Semântica:
Dado o programa em C abaixo, mostre na Tabela de Símbolos os identificadores e atributos que estão disponíveis a cada início de comando composto.

```
#include <stdio.h>

int i;

void p2 ()
{
   i = i + 2;
}

void p1 (int k)
{
   int i;
   i = k;
   p2();
   printf("%d\n", i);
}

void p4()
{
   int i;
   i = 4;
}

void p3()
{
   i = i + 3;
   printf("%d\n", i);
   p4();
   printf("%d\n", i);
}

void main()
{
   i = 10;
   p1(i);
   printf("%d\n", i);
   p3();
   printf("%d\n", i);
}
```

23. Sobre Checagem de Tipos.
A gramática abaixo gera programas representados pelo não terminal <P>, consistindo de uma sequência de declarações <D> seguida de uma única expressão <E>.

<P> ::= <D> ; <E>
<D> ::= <D> ; <D> | <id> : <T>
<T> ::= array [<inteiro>] of <TB> | <TB>
<TB> ::= real | integer
<E> ::= <inteiro> | <real> | <id> | <id> [<E>] |
<E> div <E> | <E> mod <E> | <E> <op> <E>
<op> ::= + | - | * | /

Um exemplo de programa gerado pela gramática acima é:

x : integer;
y : real;
x mod 1000

OBS: **1)** A gramática tem 2 tipos básicos (inteiro e real) e um construtor de conjuntos dos tipos básicos com índices inteiros começando em 1.

2) Os operandos div e mod fornecem respectivamente a divisão e o resto da divisão inteira.

3) Os operadores +, -, * fornecem, respectivamente, a adição, subtração, e multiplicação de inteiros ou reais dependendo dos operandos. / fornece a divisão real e o resultado será sempre real independente dos operandos. Para os outros 3 operadores, no mínimo um real presente faz o resultado ser real; inteiro caso contrário.

Pede-se:

a) Faça a checagem de tipos para esta gramática. Para isto, utilize primeiro as rotinas semânticas relacionadas com a inserção de informações necessárias na TS que serão utilizadas posteriormente para a checagem; insiram pontos de checagem de tipos com seus algoritmos respectivos.
b) Adicionem o tipo **`boolean`** à gramática e os operadores relacionais. O que muda na checagem de tipos? Implemente como pedido no item (a).