Cursos:	Bacharelado em Ciência da Computação e Bacharelado em Sistemas de Informação
Disciplinas:	(1493A) Teoria da Computação e Linguagens Formais, (4623A) Teoria da Computação e Linguagens Formais e (1601A) Teoria da Computação
Professora:	Simone das Graças Domingues Prado
e-mail:	simonedp@fc.unesp.br
home-page:	wwwp.fc.unesp.br/~simonedp/discipl.htm

Apostila 05
Assunto: Linguagens dos tipos 0 e 1

Objetivos:
⇒ Estudar as Gramáticas Irrestritas
⇒ Estudar as Gramáticas Sensíveis ao Contexto
⇒ Estudar as Linguagens Enumeráveis Recursivamente (tipo 0)
⇒ Estudar as Linguagens Sensíveis ao Contexto (tipo 01)

Conteúdo:
1. Gramática Irrestrita
2. Máquina de Turing e as Linguagens do tipo 0 e 1
3. Linguagens Recursivamente Enumeráveis
4. Linguagens Recursivas
5. Linguagens Sensíveis ao Contexto
6. Exercícios

1. Gramática Irrestrita

<u>Definição 03:</u>

Uma Gramática G = (V,T,S,P) é chamada irrestrita se todas as produções são da forma:

$u \to v$

onde $u \in (V \cup T)^+$ e $v \in (V \cup T)^*$

Podem existir variáveis e terminais no lado direito e esquerdo das produções. Aliás, a única restrição é não permitir o λ no lado esquerdo.

Exemplo 01:

Seja $L = \{a^n b^n c^n \mid n \geq 0\}$, então a Gramática Irrestrita G = ({S,C},{a,b,c},S,P)

P = { S → abc | λ
ab → aabbC
Cb → bC
Cc → cc}

Seja $w = a^3b^3c^3$ = aaabbbccc

$S \to \underline{ab}c \to a\underline{ab}bCc \to aaabb\underline{Cb}Cc \to aaabb\underline{bCCc} \to aaabbb\underline{Cc}c \to aaabbbccc$

Exemplo 02:

Seja $L = \{a^n b^{2n} a^n \mid n \geq 1\}$, então a Gramática Irrestrita G = ({S,C},{a,b},S,P)

P = { S → aAbba
aAb → aabbbA | ab
bAb → bbA
bAa → Bbaa
bB → Bb
aB → aA}

Seja $w = a^3b^6a^3$ = aaabbbbbbaaa

$S \to \underline{aAb}ba \to aabb\underline{bAb}a \to aabbb\underline{bAa} \to aabb\underline{bB}baa \to aabbBbbaa \to aab\underline{bB}bbaa \to aa\underline{bB}bbbaa$
$\to aaBbbbbaa \to a\underline{aB}bbbbaa \to a\underline{aAb}bbbaa \to aaabb\underline{bAb}bbaa \to aaabbb\underline{bAb}baa$
$\to aaabbbb\underline{bAb}aa \to aaabbbbb\underline{bAa}a \to aaabbbb\underline{bB}baaa \to aaabbb\underline{bB}bbaaa \to aaabb\underline{bB}bbbaaa$
$\to aaab\underline{bB}bbbbaaa \to aaa\underline{bB}bbbbbaaa \to aa\underline{aB}bbbbbbaaa \to aa\underline{aAb}bbbbbaaa \to aaabbbbbbaaa$

Exemplo 03:

Seja $L = \{a^n b^n a^n b^n \mid n \geq 1\}$, então a Gramática Irrestrita G = ({S,C},{a,b},S,P)

P = { S → aAbab
aAb → aabbA | ab
bAb → bbA
bAa → baB
aBa → aaB

aBb → Caabb
aCa → Caa
bC → Cb
aCb → aAb}

Seja w = $a^3b^3a^3b^3$ = aaabbbaaabbb
S → <u>aAb</u>ab → aabb<u>Aa</u>b → aabba<u>Bb</u> → aab<u>bC</u>aabb→ aa<u>bC</u>baabb → a<u>aCb</u>baabb
→ a<u>aAb</u>baabb → aaab<u>bAb</u>aabb → aaabb<u>bAa</u>abb → aaabbb<u>aBa</u>bb
→ aaabbba<u>aBb</u>b → aaabbb<u>aCa</u>abbb → aaabb<u>bC</u>aaabbb → aaab<u>bC</u>baaabbb
→ aaa<u>bC</u>bbaaabbb → aa<u>aCb</u>bbaaabbb → aa<u>aAb</u>bbaaabbb → aaabbbaaabbb

<u>Teorema 05:</u>
L é uma Linguagem Recursivamente Enumerável se, e somente se, L é gerada por uma Gramática Irrestrita.

<u>Prova:</u>
(1) Dado que existe uma gramática irrestrita, L = L(G) é uma Linguagem Recursivamente Enumerável.
Como a gramática define um procedimento de enumeração para toda cadeia na linguagem ($S \Rightarrow w$ e $S \Rightarrow x \Rightarrow w$) , as derivações podem ser dadas por uma máquina de Turing.

(2) Dado que L é uma Linguagem Recursivamente Enumerável então L é gerada por uma gramática Irrestrita.
Como L é uma LRE então, por definição, existe uma Máquina de Turing que a reconhece. Como se quer construir uma gramática irrestrita, tal que L = L(G), então significa que se quer L = L(M) = L(G), e assim temos de construir a Gramática a partir da Máquina de Turing.

Algoritmo para, a partir de uma Máquina de Turing, gerar uma Gramática Irrestrita.

Seja uma Máquina de Turing, $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$, tal que para cada $w \in L$,

$q_0w \vdash^* x\, q_f\, y$ para algum $q_f \in F$ e $x, y \in \Gamma^*$.

Então é necessário encontrar uma Gramática Irrestrita G = (V,T,S,P) que:

$q_0w \Rightarrow^* x\, q_f\, y$

Pré-requisitos:
1. S pode derivar q_0w para todo w
2. a segunda equação é possível se e somente se a primeira existir
3. quando a cadeia $x\, q_f\, y$ ($q_f \in F$ e $x, y \in \Gamma^*$) é gerada, a gramática transforma essa cadeia na cadeia original $w \in L$

A completa seqüência de derivações é, portanto:

$$S \Rightarrow^* q_0 w \Rightarrow^* x\, q_f\, y \Rightarrow^* w$$

Problema: a gramática tem de relembrar w

$S \Rightarrow^* q_0 w$

$$S \rightarrow V_{\square\square}\, S \mid S\, V_{\square\square} \mid T$$
$$T \rightarrow T\, V_{aa} \mid V_{a0a} \qquad \text{para todo } a \in \Sigma^*$$

$q_0 w \Rightarrow^* x\, q_f\, y$

$\delta(q_i, c) = (q_j, d, R)$ então $V_{aic}\, V_{pk} \rightarrow V_{ad}\, V_{pjk}$ para todo $a,p \in \Sigma \cup \{\square\}$ e $k \in \Gamma$

$\delta(q_i, c) = (q_j, d, L)$ então $V_{pk}\, V_{aic} \rightarrow V_{pjk}\, V_{ad}$ para todo $a,p \in \Sigma \cup \{\square\}$ e $k \in \Gamma$

$x\, q_f\, y \Rightarrow^* w$

$V_{ajb} \rightarrow a$ para todo $q_j \in F$, $a \in \Sigma \cup \{\square\}$ e $b \in \Gamma$

$c\, V_{ab} \rightarrow ca$ para todo $a,c \in \Sigma \cup \{\square\}$ e $b \in \Gamma$

$V_{ab}\, c \rightarrow ac$ para todo $a,c \in \Sigma \cup \{\square\}$ e $b \in \Gamma$

$\square \rightarrow \lambda$

Exemplo 04:

Seja $L = L(aa^*)$ e a Máquina de Turing $M = (\{q_0,q_1\},\{a\},\{a,\square\}, \delta, q_0, \square, \{q_1\})$ com

$\delta(q_0,a) = (q_0,a,R)$

$\delta(q_0,\square) = (q_1,\square,L)$

para w = aa tem-se: $q_0aa \vdash aq_0a \vdash aaq_0\square \vdash aq_1a$ ok. M reconhece w = aa

Então a Gramática Irrestrita a ser gerada é

As variáveis iniciais:

$S \rightarrow V_{\square\square}\, S \mid S\, V_{\square\square} \mid T$

$T \rightarrow T\, V_{aa} \mid V_{a0a}$

De $\delta(q_0,a) = (q_0,a,R)$ gera-se as produções

$V_{a0a}\, V_{aa} \rightarrow V_{aa}\, V_{a0a}$

$V_{a0a}\, V_{\square a} \rightarrow V_{aa}\, V_{\square 0a}$

$V_{a0a}\, V_{a\square} \rightarrow V_{aa}\, V_{a0\square}$

$V_{a0a}\, V_{\square\square} \rightarrow V_{aa}\, V_{\square 0\square}$

$V_{\square 0a}\, V_{aa} \rightarrow V_{\square a}\, V_{a0a}$

$V_{\square 0a}\, V_{\square a} \rightarrow V_{\square a}\, V_{\square 0a}$

$V_{\square 0a}\, V_{a\square} \rightarrow V_{\square a}\, V_{a0\square}$

$V_{\square 0a}\, V_{\square\square} \rightarrow V_{\square a}\, V_{\square 0\square}$

De $\delta(q_0, \square) = (q_1, \square, L)$ gera-se as produções

$V_{aa}\ V_{a0\square} \rightarrow V_{a1a}\ V_{a\square}$
$V_{aa}\ V_{\square 0\square} \rightarrow V_{a1a}\ V_{\square\square}$
$V_{a\square}\ V_{a0\square} \rightarrow V_{a1\square}\ V_{a\square}$
$V_{a\square}\ V_{\square 0\square} \rightarrow V_{a1\square}\ V_{\square\square}$
$V_{\square a}\ V_{a0\square} \rightarrow V_{\square 1a}\ V_{a\square}$
$V_{\square a}\ V_{\square 0\square} \rightarrow V_{\square 1a}\ V_{\square\square}$
$V_{\square\square}\ V_{a0\square} \rightarrow V_{\square 1\square}\ V_{a\square}$
$V_{\square\square}\ V_{\square 0\square} \rightarrow V_{\square 1\square}\ V_{\square\square}$

Produções finais:

$V_{a1a} \rightarrow a$
$V_{\square 1a} \rightarrow a$
$V_{a1\square} \rightarrow a$
$V_{\square 1\square} \rightarrow a$

$a\ V_{aa} \rightarrow aa$
$\square\ V_{aa} \rightarrow \square a$
$a\ V_{\square a} \rightarrow a\square$
$\square\ V_{\square a} \rightarrow \square\square$
$a\ V_{a\square} \rightarrow aa$
$\square\ V_{a\square} \rightarrow \square a$
$a\ V_{\square\square} \rightarrow a\square$
$\square\ V_{\square\square} \rightarrow \square\square$

$V_{aa}\ a \rightarrow aa$
$V_{aa}\ \square \rightarrow a\square$
$V_{\square a}\ a \rightarrow \square a$
$V_{\square a}\ \square \rightarrow \square\square$
$V_{a\square}\ a \rightarrow aa$
$V_{a\square}\ \square \rightarrow a\square$
$V_{\square\square}\ a \rightarrow \square a$
$V_{\square\square}\ \square \rightarrow \square\square$

$\square \rightarrow \lambda$

Então:

$S \Rightarrow S\ V_{\square\square} \Rightarrow T\ V_{\square\square} \Rightarrow T\ V_{aa}\ V_{\square\square} \Rightarrow V_{a0a}\ V_{aa}\ V_{\square\square} \Rightarrow V_{aa}\ V_{a0a}\ V_{\square\square} \Rightarrow V_{aa}\ V_{aa}\ V_{\square 0\square} \Rightarrow V_{aa}\ V_{a1a}\ V_{\square\square}$
$\Rightarrow V_{aa}\ a\ V_{\square\square} \Rightarrow V_{aa}\ a\square \Rightarrow a\ a\square \Rightarrow a\ a$

2. Máquina de Turing e as Linguagens do tipo 0 e 1

Nas apostilas anteriores foram trabalhados as Linguagens Regulares e as Linguagens Livres de Contexto, suas Gramáticas e seus Autômatos, bem como a Máquina de Turing. Agora é o momento de estudar as linguagens Sensíveis ao Contexto (tipo 1) e as Linguagens Recursivamente Enumeráveis (tipo 0) que usam como reconhecedores as Máquinas de Turing (veja tabela 01) fechando a Hierarquia de Chomsky (Figura 1).

Tabela 01. Linguagem, Gramática e Reconhecedor

Linguagem	**Gramática**	**Reconhecedor**
Linguagens Recursivamente enumeráveis	Gramáticas Irrestritas	Máquinas de Turing
Linguagens Sensíveis ao Contexto	Gramáticas Sensíveis ao Contexto	Máquinas de Turing com Fita Limitada ou Autômato Limitado Linearmente
Linguagens Livres de Contexto	Gramáticas Livres de Contexto	Autômatos à Pilha
Linguagens Regulares	Gramáticas Regulares	Autômatos Finitos

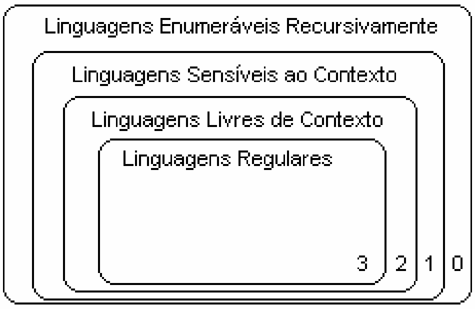

Figura 1. Hierarquia de Chomsky

3. Linguagens Recursivamente Enumeráveis

<u>Definição 01:</u>

Uma Linguagem L é dita ser uma <u>Linguagem Recursivamente Enumerável</u> se existe uma Máquina de Turing, M, que a aceita.

Ou seja, existe uma Máquina de Turing, $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, tal que para cada $w \in L$,

$q_0w \vdash^* x_1 q_f x_2$ para algum $q_f \in F$ e $x_1, x_2 \in \Gamma^*$

Onde:

- Q – conjunto de estados internos
- Σ – conjunto do alfabeto de entrada
- Γ – conjunto finito de símbolos, chamado de alfabeto da fita

δ – função de transição, definida por δ: $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
q_0 – estado inicial ($q_0 \in Q$)
F – conjunto de estados finais ($F \in Q$)

Sabe-se que na Máquina de Turing:

"A aceitação de uma cadeia pela Máquina de Turing acontece quando o estado final é atingido independente de onde a cabeça está na fita. Se a Máquina de Turing pára em algum estado não final ou simplesmente entrar em loop infinito, então a cadeia não é aceita ". (Apostila 04, pág 04)

Assim, considere os termos:

Aceita(M) = conjunto de todas as cadeias de Σ^* aceitas por M
Rejeita(M) = conjunto de todas as cadeias de Σ^* não aceitas por M
Loop(M) = conjunto de todas as cadeias de Σ^* que entram em ciclos infinitos em M

Então:

$\Sigma^* = \text{Aceita}(M) \cup \text{Rejeita}(M) \cup \text{Loop}(M)$
$\varnothing = \text{Aceita}(M) \cap \text{Rejeita}(M) \cap \text{Loop}(M)$

Nas Linguagens Recursivamente Enumeráveis tem-se, portanto:

$L(M) = \text{Aceita}(M)$
$\text{Rejeita}(M) = \Sigma^* - \{L(M) \cup \text{Loop}(M)\}$

Assim, é simples saber se uma cadeia pertence à Linguagem Recursivamente Enumerável: é só verificar se uma Máquina de Turing a aceita. O complicado é verificar se uma cadeia não pertence à Linguagem, já que, na rejeição da cadeia, a Máquina de Turing pode não parar, ou seja, entrar em looping.

Só que, podem existir linguagens que não são reconhecidas por uma Máquina de Turing, ou seja, pode não ser possível construir uma Máquina de Turing que a aceite e portanto linguagens que não são recursivamene enumeráveis. Veja o teorema abaixo.

Teorema 01:

Para qualquer alfabeto não vazio, Σ, existem linguagens que não são recursivamente enumeráveis.

Pode-se concluir então que:

- Existem problemas sem solução na forma de algoritmos, já que existem linguagens que não são reconhecidas por uma Máquina de Turing.
- O conjunto de todos os problemas solucionáveis, ou seja, reconhecidas por Máquinas de Turing, é infinitamente contável, já os problemas não solucionáveis são infinitamente não contáveis.
- Portanto, computacionalmente existem mais problemas do que algoritmos para resolvê-los.

Prova do Teorema 01:

Suponha $\Sigma = \{ a, b \}$.

a) Suponha que Xi representa o i-ésimo elemento na ordenação lexicográfica de Σ^*, ou seja, $\{ \lambda, a, b, aa, ab, ba, bb, aaa, aab, ... \}$ com $X_0 = \{ \lambda \}$, $X_1 = \{a\}$, $X_2 = \{b\}$ etc.

b) É possível codificar todas as Máquinas de Turing como uma palavra sobre Σ de tal forma que cada código represente uma única Máquina de Turing. Suponha o conjunto dos códigos ordenados lexicograficamente e suponha que Ti representa o i-ésimo código nesta ordenação.

A Linguagem L, abaixo, não é Recursivamente Enumerável:

$L = \{ Xi \mid Xi \text{ não é aceita por } Ti \}$

Prova: A prova que segue é por redução ao absurdo. Suponha que L é Recursivamente Enumerável. Então, por definição, existe uma Máquina de Turing que aceita L. Seja Tk a codificação desta Máquina de Turing, ou seja, Aceita(Tk) = L. Assim, as seguintes afirmações são válidas:

a) Por definição de L, Xk pertence a L se, e somente se, Xk não é aceita por Tk;

b) Entretanto, como Tk aceita a linguagem L, Xk pertence a L se, e somente se, Xk é aceita por Tk. Claramente b) contradiz a). Portanto, é absurdo supor que L é Enumerável Recursivamente. Logo, L não é Enumerável Recursivamente.

Com isso a figura 1 sobre a hierarquia de Chomsky pode ser modificada para (Figura 2):

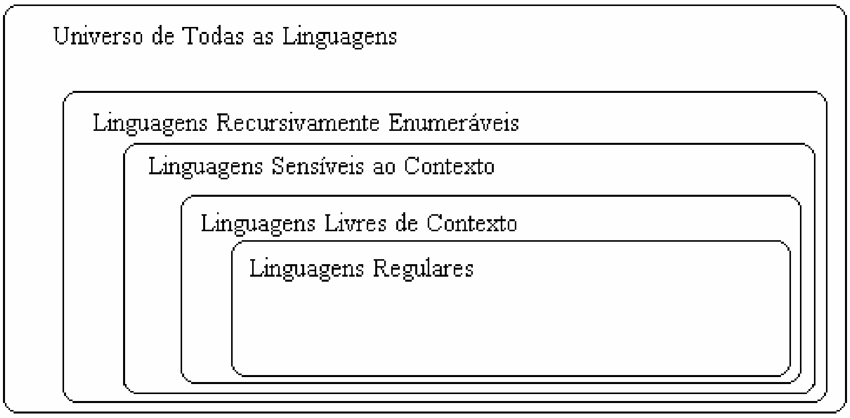

Figura 2. Hierarquia de Chomsky Modificada - 1

4. Linguagens Recursivas

Como fica difícil determinar se uma cadeia não pertence à linguagem, já que pode acontecer que ao processar essa cadeia a Máquina de Turing entre em loop infinito, pode-se restringir as Linguagens Recursivamente Enumeráveis dizendo que elas devem ter o conjunto Loop(M) = ∅, ou seja,

Aceita(M1) = L(M1) = L1

Rejeita(M1) = Σ^* - L1

Loop(M) = ∅

Ao restringir a Linguagem Recursivamente Enumerável cria-se um outro subconjunto: a Linguagem Recursiva. Veja a próxima definição.

Definição 02:

Uma Linguagem L é dita ser Recursiva se existe uma Máquina de Turing que aceita L e que pára em cada w em Σ^+.

Assim, mesmo se w não pertencer à Linguagem Recursiva, a Máquina de Turing pára ao contrário da Linguagem Recursivamente Enumerável. Veja como fica a Hierarquia de Chomsky (figura 3).

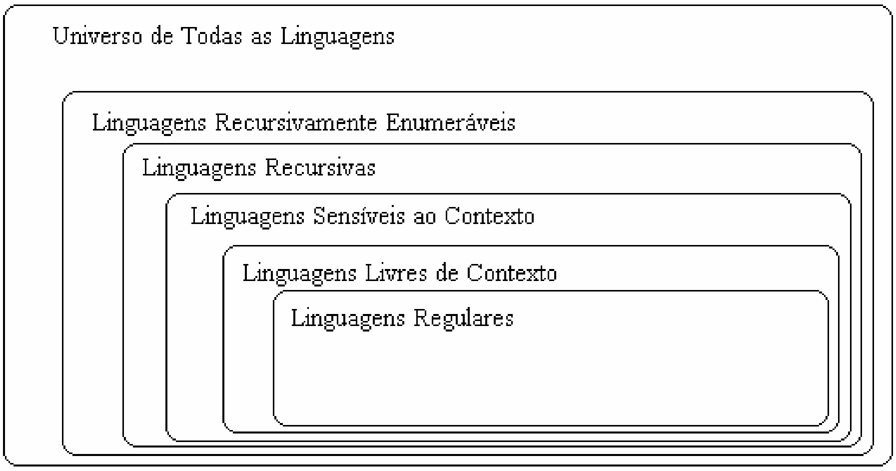

Figura 3. Hierarquia de Chomsky Modificada – 2

Veja a seguir algumas propriedades relacionadas às Linguagens Recursivas (LR):

- Se L é uma LR então seu complemento também é uma LR (Teorema 2)
- L é uma LR se, e somente se, L e seu complemento são LRE (Teorema 3)
- A classe das LRs está contida propriamente na classe das LREs (Teorema 4)

Teorema 02:
Se uma Linguagem L sobre um alfabeto Σ qualquer é recursiva, então seu complemento Σ^* - L também é uma Linguagem Recursiva.

Prova: Suponha uma Linguagem Recursiva L sobre o alfabeto Σ. Assim, existe uma Máquina de Turing M, que aceita a linguagem e sempre pára para qualquer entrada (por definição). Ou seja:

Aceita(M) = L
Rejeita(M) = Σ^* - L
Loop(M) = $\varnothing$

Seja uma outra Máquina de Turing, M_1, construída a partir de M, só que invertendo as condições de ACEITA por REJEITA e vice-versa, para obter o complemento. Portanto, M_1 aceita Σ^* - L e sempre pára para qualquer entrada. Ou seja:

Aceita(M_1) = Σ^* - L
Rejeita(M_1) = L
Loop(M_1) = $\emptyset$

Logo, Σ^* - L é uma Linguagem Recursiva.

Teorema 03:
Uma Linguagem L sobre um alfabeto Σ qualquer é Recursiva, se e somente se L e Σ^* - L são Linguagens Recursivamente Enumeráveis.

Prova:

a) Suponha L uma Linguagem Recursiva sobre Σ. Então, como foi mostrado no Teorema 02, Σ^* - L é Recursiva. Como toda Linguagem Recursiva também é Enumerável Recursivamente , então L e Σ^* - L são Enumeráveis Recursivamente;

b) Suponha L uma linguagem sobre Σ tal que L e Σ^* - L são Enumeráveis Recursivamente. Então existem M1 e M2, Máquinas de Turing tais que:

ACEITA(M1) = L
ACEITA(M2) = Σ^* - L

Seja M Máquina de Turing não-determinística conforme figura abaixo (figura 4).

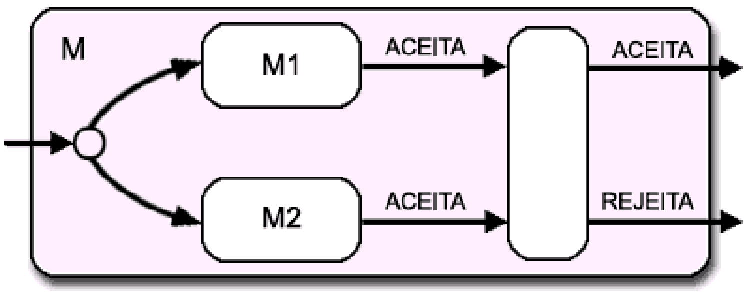

Figura 4. Uma MT não determinística
(Menezes, P.B., Cap 4, Ling Formais e Autômatos)

Para qualquer palavra de entrada, M aceita se M1 aceita e M rejeita se M2 aceita. Portanto, claramente M sempre pára. Logo, L é Recursiva.

Teorema 04:
A classe das Linguagens Recursivas **está contida propriamente** na classe das Linguagens Recursivamente Enumeráveis.

Prova:
Para mostrar que a inclusão é própria, é suficiente mostrar que existe pelos menos uma Linguagem Enumerável Recursivamente que não é Recursiva. Isso porque se a classe das Linguagens Recursivas (LR) está contida propriamente na classe das Linguagens Recursivamente Enumeráveis (LRE), quer dizer que a LR é um subconjunto da LRE e esses dois conjuntos são diferentes.

Como na Prova do Teorema 01,

a) suponha $\Sigma = \{ a, b \}$;

b) suponha que Xi representa o i-ésimo elemento na ordenação lexicográfica de Σ^*, ou seja, $\{ \lambda, a, b, aa, ab, ba, bb, aaa, aab, ... \}$ com $X_0 = \{ \lambda \}$, $X_1 = \{a\}$, $X_2 = \{b\}$ etc.

c) suponha que Ti representa o i-ésimo código nesta ordenação.

d) suponha a Linguagem Enumerável Recursivamente L, onde X_i e T_i são definidas como:

$$L = \{ X_i \mid X_i \text{ é aceita por } T_i \}$$

Então temos de provar que L não é uma Linguagem Recursiva.

L é Enumerável Recursivamente. Seja uma Máquina de Turing M que aceita uma palavra w qualquer pertencente a L:

a.1) M gera as palavras X_1, X_2, ... em ordem lexicográfica, comparando com w. Quando $X_i = w$, M sabe que w é a i-ésima palavra na enumeração;

a.2) M gera T_i, a i-ésima Máquina de Turing;

a.3) M simula T_i para a entrada $w = X_i$ e, se w pertence a ACEITA(T_i), então w pertence a ACEITA(M);

Portanto, M aceita w se, e somente se, $X_i = w$ é aceita por T_i. Logo, L é Enumerável Recursivamente;

L não é Recursiva. Conforme o Teorema 3, L é Recursiva se, e somente se, L e seu complemento são Enumeráveis Recursivamente. Como o complemento de L, ou seja: $L = \{ X_i \mid X_i \text{ não é aceita por } T_i \}$ (Teorema 01) não é Enumerável Recursivamente, então L não é Recursiva.

5. Linguagens Sensíveis ao Contexto

Definição 04:

Uma Gramática Irrestrita G = (V, T, S, P) é chamada Gramática Sensível ao Contexto se todas as produções da forma $u \to v \in P$, tem a propriedade $|u| \leq |v|$

Ou seja, a cada etapa de derivação, o tamanho da palavra derivada não pode diminuir, excetuando-se para gerar a palavra vazia, se esta pertencer à linguagem.

Teorema 07:
Uma Linguagem L é Sensível ao Contexto se existe uma Gramática Sensível ao Contexto, tal que L = L(G) ou $L = L(G) \cup \{\lambda\}$

Exemplo 05:
Seja $L = \{a^n b^n c^n \mid n \geq 1\}$, então a Gramática Sensível ao Contexto G = ({S,C},{a,b,c},S,P)
Se observarmos o exemplo 01 e retirando a produção $S \to \lambda$, a Gramática Irrestrita lá definida é uma Gramática Sensível ao Contexto, ou seja,

P = { $S \to abc$
$ab \to aabbC$
$Cb \to bC$
$Cc \to cc$ }

Exemplo 06:
Seja $L = \{a^n b^{2n} a^n \mid n \geq 1\}$, então a Gramática Irrestrita G = ({S,C},{a,b},S,P) gerada no Exemplo 02 não é Sensível ao Contexto porque se tem $aAb \to ab$, ou seja, $|aAb| > |ab|$. Uma formulação de uma Gramática Sensível ao Contexto poderia ser:

P = { $S \to aAbba \mid abba$
$aAb \to aabbbA$
$Bb \to bB$
$Ba \to Caa \mid aa$
$bCa \to Cba$
$bC \to Cb$
$aCb \to aAb$ }

Exemplo 07:
Seja $L = \{a^n b^n a^n b^n \mid n \geq 1\}$, então a Gramática Irrestrita G = ({S,C},{a,b},S,P) gerada no Exemplo 03 não é Sensível ao Contexto porque se tem, novamente, $aAb \to ab$, ou seja, $|aAb| > |ab|$. Então uma Gramática Sensível ao Contexto poderia ser:

$P = \{ \quad S \rightarrow aAbab \mid abab$
$aAb \rightarrow aabbB$
$Bb \rightarrow bB$
$Ba \rightarrow aC$
$aCb \rightarrow Daabb \mid aabb$
$Da \rightarrow aD$
$bDa \rightarrow Eba$
$bE \rightarrow Eb$
$aE \rightarrow aA \ \}$

Teorema 08:
Para toda Linguagem Sensível ao Contexto L não incluindo λ, existe algum Autômato Limitado Linearmente M tal que L = L(M)

Exemplo 08:
Seja $L = \{a^n b^n a^n b^n \mid n \geq 1\}$, então o Autômato Limitado Linearmente é dado por:
$M = (\{q_0,q_1,...,q_8,q_9\}, \{a,b,[,]\}, \{a,b,x,[,]\}, \delta, q_0, \square, \{q_9\})$ onde
$\delta(q_0, [) = \{(q_1, [, R)\}$,
$\delta(q_1, a) = \{(q_2, x, R)\}$,
$\delta(q_1, b) = \{(q_5, x, R)\}$,
$\delta(q_1, x) = \{(q_1, x, R)\}$,
$\delta(q_2, a) = \{(q_2, a, R)\}$,
$\delta(q_2, b) = \{(q_2, b, R)\}$,
$\delta(q_2,]) = \{(q_3,], L)\}$,
$\delta(q_2, x) = \{(q_3, x, L)\}$,
$\delta(q_3, b) = \{(q_4, x, L)\}$,
$\delta(q_4, a) = \{(q_4, a, L)\}$,
$\delta(q_4, b) = \{(q_4, b, L)\}$,
$\delta(q_4, x) = \{(q_1, x, R)\}$,
$\delta(q_5, a) = \{(q_5, a, R)\}$,
$\delta(q_5, b) = \{(q_5, b, R)\}$,
$\delta(q_5, x) = \{(q_6, x, L)\}$,
$\delta(q_6, a) = \{(q_7, x, L)\}$,
$\delta(q_7, a) = \{(q_7, a, L)\}$,
$\delta(q_7, b) = \{(q_7, b, L)\}$,
$\delta(q_7, x) = \{(q_8, x, R)\}$,
$\delta(q_8, b) = \{(q_5, x, R)\}$,
$\delta(q_8, x) = \{(q_9, x, R)\}$,

As Linguagens Sensíveis ao Contexto são fechadas para operação de união e intersecção. A construção dos resultados dessas operações são similares às feitas nas Linguagens Regulares e Livres de Contexto.

6. Exercícios

1. Encontrar a Gramática Irrestrita para cada uma das linguagens abaixo:
    a. $L = \{w \mid w$ tem o mesmo número de a e b$\}$
    b. $L = \{ a^i b^j c^k \mid i = j$ ou $j = k\}$
    c. $L = \{ a^i b^j c^k \mid 1 \leq i \leq j \leq k\}$
    d. $L = \{a^n \mid n$ é potencia de 2$\}$
    e. $L = \{0^x 1^y \mid x,y \geq 1\}$
    f. $L = \{wcw : w \in \{a,b\}^* \}$
    g. $L = \{ww : w \in \{a,b\}^* \}$
    h. $L = \{a^i b^i a^i : i \geq 0\}$
    i. $L = \{a^i b^i c^i d^i : i \geq 1\}$
    j. $L = \{a^i b^j c^k : 0 \leq i \leq j \leq k \}$
    k. $L = \{a^i b^j c^i d^j : i \geq 1\}$
    l. $L = \{(a^i b^i)^j (c^k d^k)^j : i,j,k \geq 1\}$
    m. $L = \{a^i b^j : j = i^2 \}$

2. Encontrar a Gramática Sensível ao Contexto para cada uma das linguagens abaixo:
    a. $L = \{a^n b^n c^{2n} \mid n \geq 1\}$
    b. $L = \{a^n b^m c^n d^m \mid n \geq 1, m \geq 1\}$
    c. $L = \{ww \mid w \in \{a,b\}^+\}$
    d. $L = \{w \mid n_a(w) = n_b(w) = n_c(w)\}$
    e. $L = \{w \mid n_a(w) = n_b(w) < n_c(w)\}$
    f. $L = \{cwc \mid c = a$ ou $c = b$ e $w \in \{a,b\}^+\}$

3. Encontrar um Autômato Limitado Linearmente que reconhece cada uma das linguagens abaixo:
    a. $L = \{a^n b^n a^n \mid n \geq 1\}$
    b. L(01(01)*)
    c. L(aa*)