Gramáticas

Definições
Hierarquia de Chomsky
Exemplos de gramáticas

Formalmente, as gramáticas são caracterizadas como quádruplas ordenadas

$$G = (V_n, V_t, P, S)$$

onde:

Vn representa o vocabulário não terminal da gramática. Este vocabulário corresponde ao conjunto de todos os símbolos dos quais a gramática se vale para definir as leis de formação das sentenças da linguagem.

Vt é o vocabulário terminal, contendo os símbolos que constituem as sentenças da linguagem. Dá-se o nome de terminais aos elementos de Vt.

P representa o conjunto de todas as leis de formação utilizadas pela gramática para definir a linguagem.

Para tanto, cada construção parcial, representada por um não-terminal, é definida como um conjunto de regras de formação relativas à definicão do não-terminal a ela referente. A cada uma destas regras de formação que compõem o conjunto P dá-se o nome de produção da gramática.

Assumimos $Vn \cap Vt = \varnothing$. Convencionamos que $Vn \cup Vt = V$
Cada produção P tem a forma:

$$\alpha \rightarrow \beta \qquad \alpha \in V+; \qquad \beta \in V^*$$

S ε Vn denota a principal categoria gramatica de G; é dito o símbolo inicial ou o axioma da gramática. Indica onde se inicia o processo de geração de sentenças.

Ex.1: G = ({S, A, B}, {a, b}, P, S)

P: {S -> AB
A -> a
B -> b}

Notação/Convenções

- Letras do alfabeto latino maiúsculas {A,B,..Z}: variáveis
- Letras do começo do alfabeto latino minúsculas {a,b,c,...}: terminais
- Letras do fim do alfabeto latino minúsculas {t,u,v,x,z}: cadeias de terminais
- Letras gregas minúsculas {α,β,γ,δ,ε, ..., ω}: cadeias de terminais e não terminais

alfa, beta, gama, delta, épsilon, zeta, eta, teta, iota, kapa, lâmbda, mi, ni, xi, ômicron, pi, rô, sigma, tau, úpsilon, fi, qui, psi, ômega

Exercício

- Tentem definir uma gramática simples, por exemplo, de algum construtor ou comando de uma linguagem de programa.

Definida uma gramática G, qual é a linguagem gerada por ela?

Precisaremos das relações $=>_G$ (deriva diretamente) e $=>_G^*$ (deriva) definidas entre as cadeias de V*

Def1. Se $\alpha \rightarrow \beta$ é uma produção de P e γ (gama) e δ (delta) são cadeias quaisquer de V*, então $\gamma\ \alpha\ \delta =>_G \gamma\ \beta\ \delta$ (deriva diretamente na gramática G).

Dizemos que a produção $\alpha \rightarrow \beta$ é aplicada à cadeia $\gamma\ \alpha\ \delta$ para obter $\gamma\ \beta\ \delta$. A relação $=>_G$ relaciona cadeias exatamente quando a segunda é obtida a partir da primeira pela aplicação de uma única produção.

No Ex.1.: S $=>_G$ AB ; aB $=>_G$ ab ou
S $=>_G$ AB $=>_G$ aB $=>_G$ ab

Def2. Suponha que $\alpha_1\ \alpha_2\ \alpha_3 \ldots \alpha_m$ são cadeias de V* e $\alpha_1 \Rightarrow_G \alpha_2$, $\alpha_2 \Rightarrow_G \alpha_3$, ..., $\alpha_{m-1} \Rightarrow_G \alpha_m$. Então dizemos que $\alpha_1 \Rightarrow_G^* \alpha_m$ (**deriva**). Aplicamos algum número de produções de P. Por convenção $\alpha \Rightarrow_G^* \alpha$ para a cadeia α.

No Ex.1.: $S \Rightarrow_G^* ab$;

$S \Rightarrow_G^* aB$;

$AB \Rightarrow_G^* ab$;

$ab \Rightarrow_G^* ab$

Def3. Forma sentencial: uma cadeia α composta de terminais e não terminais se $S =>^* \alpha$

No Ex.1: aB, AB, S, ab são formas sentenciais.

Uma forma sentencial, α, é uma sentença de G se $S \Rightarrow^* \alpha$ e $\alpha \in Vt^*$ (são composta de terminais). Ou seja, as cadeias geradas pela gramática são as sentenças de G.

Def4. A Linguagem L gerada por uma gramática G é definida como o conjunto de cadeias geradas por G. Ou seja,

$L(G) = \{x \mid x \in Vt^* \text{ e } S \Rightarrow_G^* x\}$ ou $\{x \mid x$ é sentença de $G\}$

1. A cadeia consiste somente de terminais
2. A cadeia pode ser derivada a partir do símbolo inicial da gramática

Def5. Duas gramáticas G1 e G2 são equivalentes sse $L(G1) = L(G2)$

Exemplos de Gramáticas

G1 = ({S}, {0,1}, P1, S)
P1: { 1. S -> 0S1
2. S -> 01 }

Qual é a linguagem gerada por G1? Aplicamos o **processo de derivação** para saber L(G1), que é o processo de obtenção de cadeias a partir de uma gramática.

G2 = ({S,B,C}, {a,b,c}, P2, S)
P2: { 1. S -> aSBC 2. S -> aBC
3. CB -> BC 4. aB -> ab
5. bB -> bb 6. bC -> bc
7. cC -> cc}
L(G2) = ?

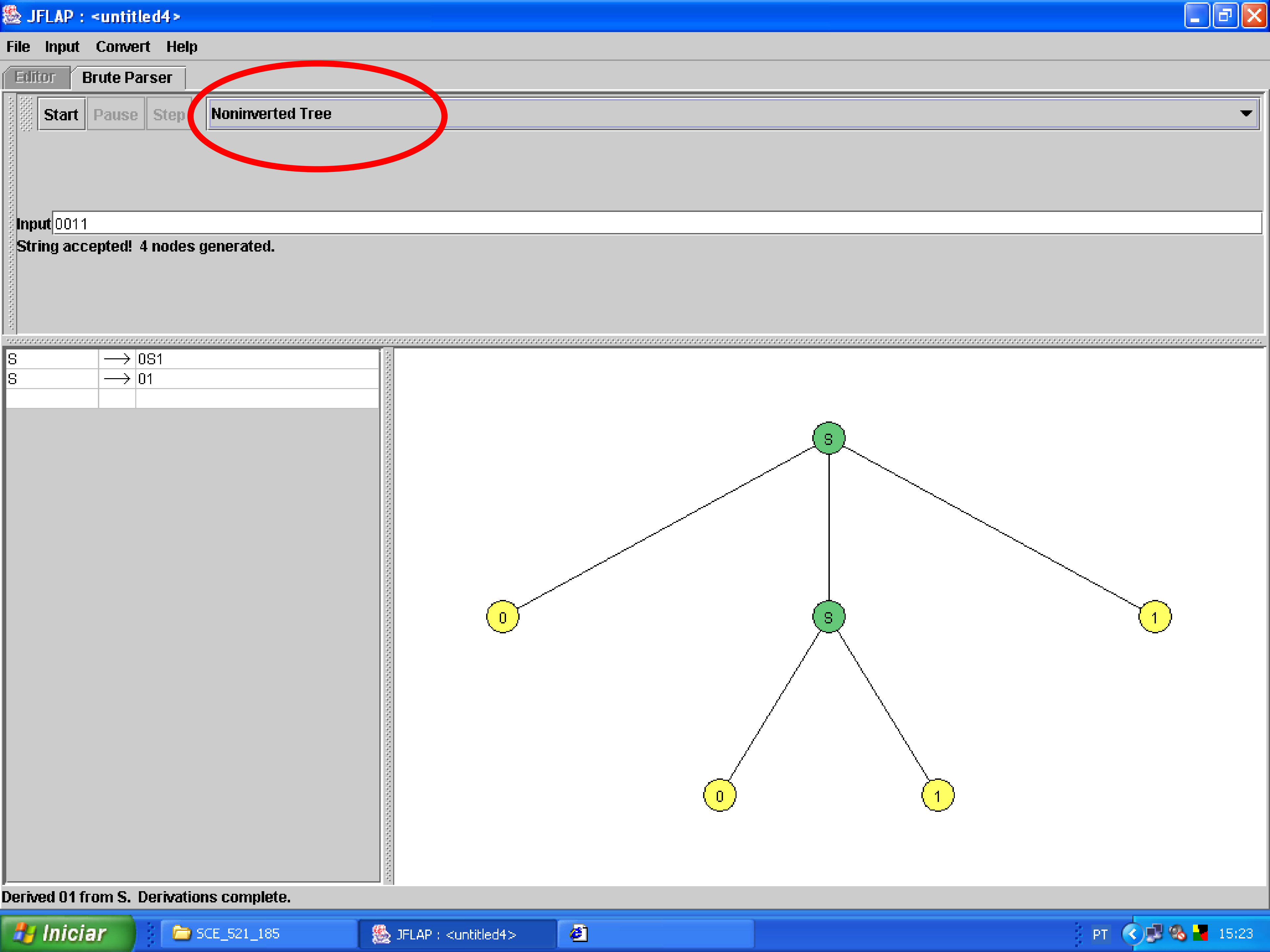
G1

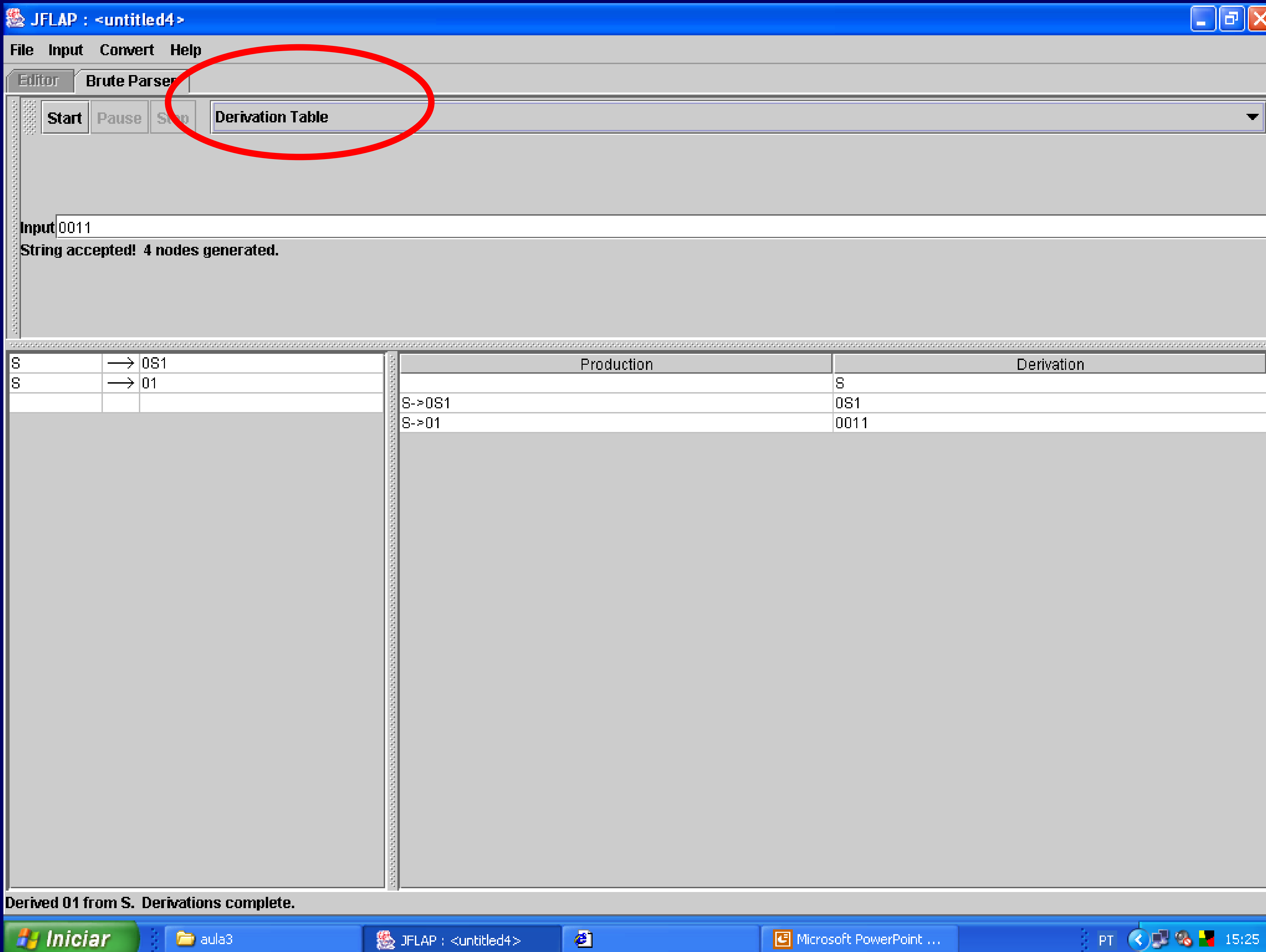
- A menor cadeia gerada é 01: $S \Rightarrow^2 01$

- Se aplicarmos n-1 vezes a produção 1, seguida da produção 2 teremos:

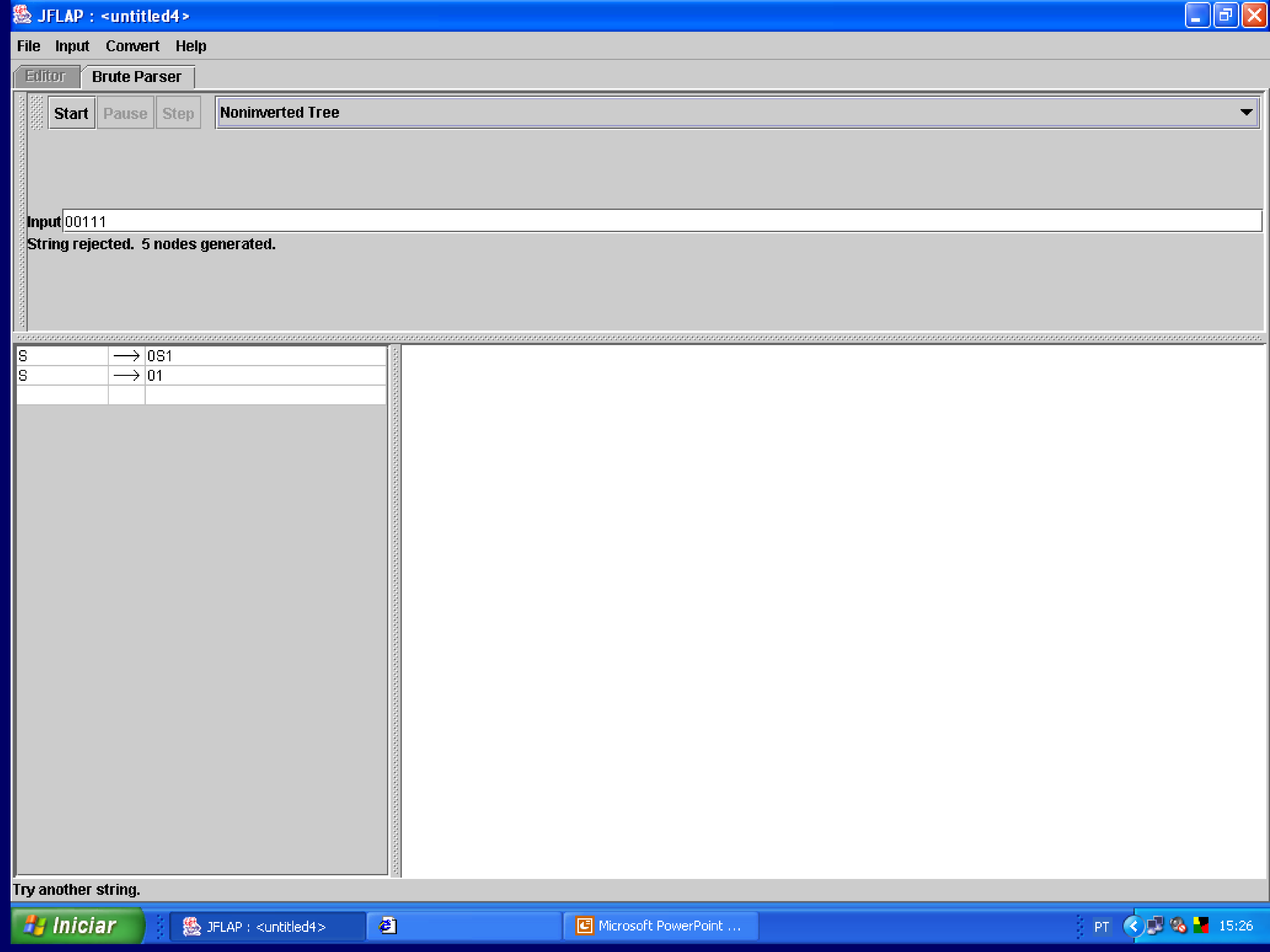
- $S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 0^3S1^3 \Rightarrow^*$
- $0^{n-1}S1^{n-1} \Rightarrow 0^n1^n$
- Portanto, $L(G1) = \{0^n1^n \mid n >= 1\}$

  ou $S \Rightarrow^* 0^n1^n$


JFLAP : <untitled4>
File Input Convert Help
Editor
Brute Parser
Start
Pause
Step
Noninverted Tree
Input 0011
String accepted! 4 nodes generated.
S → 0S1
S → 01
S
0
S
1
0
1
Derived 01 from S. Derivations complete.
Iniciar
SCE_521_185
JFLAP : <untitled4>
PT
15:23



JFLAP : <untitled4>
File Input Convert Help
Editor
Brute Parser
Start
Pause
Step
Derivation Table
Input 0011
String accepted! 4 nodes generated.
S → 0S1
S → 01
Production
Derivation
S
S->0S1
0S1
S->01
0011
Derived 01 from S. Derivations complete.
Iniciar
aula3
JFLAP : <untitled4>
Microsoft PowerPoint ...
PT
15:25


JFLAP : <untitled4>
File Input Convert Help
Editor
Brute Parser
Start Pause Step
Noninverted Tree
Input 00111
String rejected. 5 nodes generated.
S → 0S1
S → 01
Try another string.
Iniciar
JFLAP : <untitled4>
Microsoft PowerPoint ...
PT
15:26

G2

- A menor cadeia gerada é abc: $S \Rightarrow^2 aBC \Rightarrow^4 abC \Rightarrow^6 abc$
- Usamos 1 n-1 vezes: $S \Rightarrow^* a^{n-1}S(BC)^{n-1}$
- Usamos a 2 uma vez: $S \Rightarrow^* a^n(BC)^n$
- A 3 permite trocar B com C para que B´s precedam os C´s
  - Para n = 2 aaBCBC => aaBBCC (usamos a regras 3 1 vez)
  - Para n = 3 aaaBCBCBC => aaaBBCCBC => aaaBBCBCC => aaaBBBCCC (usamos a regra 3 3 vezes)
  - Para n = 4 aaaaBCBCBCBC => aaaBBCCBCBC => aaaaBBCBCCBC => aaaaBBBCCCBC =>aaaaBBBCCBCC => aaaaBBBCBCCC => aaaaBBBBCCCC (usamos a regra 3 5 vezes);
  - Para n = 5 usamos a 3 10 vezes.
- Assim $S \Rightarrow^* a^nB^nC^n$
- Usamos a 4 uma vez: $S \Rightarrow^* a^nbB^{n-1}C^n$
- Aplicamos a 5 n-1 vezes: $S \Rightarrow^* a^nb^nC^n$
- Aplicamos a 6 uma vez: $S \Rightarrow^* a^nb^ncC^{n-1}$
- Aplicamos a 7 n-1 vezes: $S \Rightarrow^* a^nb^nc^n$

$$L(G2) = \{a^nb^nc^n \mid n >= 1\}$$

Tipos de Gramáticas

- Chamamos o tipo de gramática que definimos de **tipo 0 ou Recursivamente enumerável/ Estrutura de Frase/ Irrestritas.**

$$\alpha \rightarrow \beta \qquad \alpha \in V^+; \quad \beta \in V^*$$

- Não há restrições nas regras de produção.

G= ({S,A,B,C,D,E}, {a}, P,S)

P = { 1. S-> ACaB
2. Ca -> aaC
3. CB -> DB
4. CB -> E
5. aD -> Da
6. AD -> AC
7. aE -> Ea
8. AE -> λ }

L(G) =?

- Menor cadeia: aa

S =>^{1} ACaB =>^{2} AaaCB =>^{4} AaaE =>^{7} AaEa =>^{7} AEaa =>^{8} aa

- A e B servem como marcadores da esq e dir para as formas sentenciais.
- C é o marcador que se move através da cadeia de a´s entre A e B, dobrando seu número pela produção 2.
- Quando C alcança o marcador à direita B, ele se torna um D ou E pela produção 3 ou 4.
- Se um D é escolhido, então ele migra à esquerda pela produção 5 até que o marcador à esq, A, seja alcançado.
- Nesse ponto, D se torna C de novo pela produção 6 e o processo recomeça.
- Se um E é escolhido, o marcador à direita (B) é consumido.
- O E migra à esquerda pela produção 7 e consome o marcador à esq pela produção 8.

L(G) = { a^{2^n} | n é um inteiro positivo}

Até este ponto não foi imposta qualquer restrição sobre a gramática ou sobre as produções que denotam as leis de formação da linguagem que está sendo definida.

As gramáticas gerais têm limitações em relação à sua aplicabilidade no contexto do estudo dos compiladores, devido às dificuldades que acarretam em seu tratamento, sendo que as linguagens de programação de interesse não exigem toda a generalidade que as gramáticas gerais definidas acima são capazes de oferecer.

Torna-se atraente o estudo de casos particulares, de aplicação mais restrita, porém suficiente para resolver os problemas levantados ao se projetar compiladores para linguagens de interesse. Sendo assim, dividimos as gramáticas em quatro classes, que serão vistas a seguir.

Classes Gramaticais

Conforme as restrições impostas ao formato das produções de uma gramática, a classe de linguagens que tal gramática gera varia correspondentemente. A teoria mostra que há quatro classes de gramáticas capazes de gerar quatro classes correspondentes de linguagens, de acordo com a denominada **Hierarquia de Chomsky:**

Gramáticas Irrestritas ou Tipo 0

Gramáticas Sensíveis ao Contexto ou Tipo 1

Gramáticas Livres de Contexto ou Tipo 2

Gramáticas Regulares ou Tipo 3

Linguagens LEF

As linguagens geradas pelas Gramáticas com Estrutura de Frase ou do Tipo 0 são chamadas de Linguagens com Estrutura de Frase (LEF) ou Linguagens do Tipo 0.

Gramáticas Sensíveis ao/Dependentes de Contexto ou Tipo 1

Se às regras de substituição for imposta a restrição de que nenhuma substituição possa reduzir o comprimento da forma sentencial à qual a substituição é aplicada, cria-se uma classe de gramáticas ditas sensíveis ao contexto. As gramáticas que obedecem a estas restrições pertencem, na hierarquia de Chomsky, ao conjunto das Gramáticas Sensíveis ao Contexto (GSC) ou do Tipo 1.

Para as GSC, as produções são todas da forma

$\alpha \rightarrow \beta$, com $|\alpha| <= |\beta|$ (produções não decrescentes)
onde $\alpha, \beta \in (Vn \cup Vt)^+$

Alguns autores colocam as produções de uma GDC como:
$\alpha 1 A \alpha 2 \rightarrow \alpha 1 \beta \alpha 2$ com $\alpha 1, \alpha 2, \beta \in V^*$, $\beta <> \lambda$ e $A \in Vn$
Para motivar o nome sensível ao contexto desde que a produção $\alpha 1 A \alpha 2 \rightarrow \alpha 1 \beta \alpha 2$ permite que A seja trocado por β no contexto de $\alpha 1$ e $\alpha 2$.

A gramática do Ex 2 é uma GSC e também as variações dela abaixo:

G1 = ({A,B,C}, {a,b,c},P1,A)
P1 = { A -> abc   A -> aBbc
Bb -> bB   Bc -> Cbcc
bC -> Cb   aC -> aaB
aC -> aa }

G2 = ({S,C}, {a,b,c},P2,S)
P2 = { S -> abc
ab -> aabbC
Cb -> bC
Cc -> cc }

Linguagens LSC

As linguagens geradas pelas Gramáticas Sensíveis ao Contexto ou do Tipo 1 são chamadas de Linguagens Sensíveis ao Contexto (LSC) ou Linguagens do Tipo 1.

Resultado 1:

Toda gramática do tipo 1 é também do tipo 0.

Corolário 1:

Toda LSC é também uma LEF (mas nem toda LEF é LSC).

Gramáticas Livres de Contexto ou Tipo 2

As **Gramáticas Livres de Contexto (GLC) ou do Tipo 2** são aquelas cujas regras de produção são da forma:

$A \rightarrow \alpha$ onde $A \in Vn$, $\alpha \in V+$

Ou seja, quando do lado esquerdo da regra há apenas um símbolo não-terminal (uma variável)

A gramática do Ex 1 é uma GLC. Outro exemplo:

G = ({S,A,B}, {a,b}, P,S)
P = {S -> aB | bA
A -> a | aS | bAA
B -> b | bS | aBB }

L(G) = ?

- Menores cadeias: ab e ba

S => aB => ab

S => bA => ba

S => aB => abS => abbA => abba

=> abaB => abab

=> aaBB => aabb

S => bA => baS => baaB => baab

=> babA => baba

=> bbAA => bbaa

L(G) = {w $\in$ {a,b}+ | nro(a) = nro(b)}

Todas as combinações de cadeias em V+ com nro(a) = nro(b)

Linguagens LLC

As linguagens geradas pelas Gramáticas Livres de Contexto ou do Tipo 2 são chamadas de Linguagens Livres de Contexto (LLC) ou Linguagens do Tipo 2.

Resultado 2:

Toda gramática do tipo 2 é também do tipo 1.

Corolário 2:

Toda LLC é também uma LSC (mas nem toda LSC é uma LLC).

BNF

Outra maneira de se representar as Gramáticas Livres de Contexto é através da Forma Normal de Backus.

Neste caso, -> é substituído por ::= e os não terminais são ladeados por < >

No caso de repetições de lado esquerdo:

<A> ::= a1
<A >::= a2
:
<A> ::= an
escreve-se: <A> ::= a1| a2| ...| an

Os símbolos <,> , ::=, | formam a metalinguagem, ou seja, são símbolos que não fazem parte da linguagem mas ajudam a descrevê-la.

Exemplo:

G = {Vn, Vt, P, S} onde:
Vn = {<sentença, <sn>, <sv>, <artigo>, <substantivo>, <verbo>}

Vt = {o, a, peixe, comeu, isca}
S = <sentença>
P = {

1. <sentença> ::= <sn> <sv>
2. <sn> ::= <artigo> <substantivo>
3. <sv> ::= <verbo> <sn>
4. <artigo> ::= o|a
5. <verbo> :: = mordeu
6. <substantivo> ::= peixe|isca }

Exercícios:

a) verifique se a cadeia "a isca mordeu o peixe" é uma sentença de L(G).
b) Dê exemplos de sentenças de L(G).

Mais GLC:

G = ({S}, {a, +, *, (,)}, P, S)
P = {
  S -> S * S
  S -> S + S
  S -> (S)
  S -> a }

L(G) = conjunto das expressões aritméticas envolvendo **, +, ( )* e *a.*

Um exemplo de cadeia formada por esta gramática é *a * (a + a).*

Processo inverso: Dada uma L(G) definir a gramática G.

$L(G) = \{a^m b^n \mid m \geq 1, n \geq 1\}$

$L(G) = \{a^m b^n \mid m \geq 1,\ n \geq 1\}$ ou $a+b+$

Resp.:
G=({S, A, B}, {a, b}, P, S)
P = {S –> AB
A –> aA | a
B -> bB | b }

***Obs.:* Caso geral:**

Se $S \Rightarrow \alpha S \mid \beta$ **então** $L(G) = \alpha^* \beta$

$L(G) = \{a^n b^n \mid n \geq 1\}$

$L(G) = \{a^n b^n \mid n \geq 1\}$

$G = (\{S\}, \{a, b\}, P, S)$

$P = \{S \text{ -> } aSb \mid ab\}$

Gramáticas Regulares ou Tipo 3

Aplicando-se mais uma restrição sobre a forma das produções, pode-se criar uma nova classe de gramáticas, as Gramáticas Regulares (GR), de grande importância no estudo dos compiladores por possuírem propriedades adequadas para a obtenção de reconhecedores simples. Nas GRs, as produções são restritas às formas seguintes:

A -> aB ou A -> a (linear à direita)

OU

A -> Ba ou A -> a (linear à esquerda)

onde A,B ∈ Vn e a ∈ Vt

Tem que escolher uma das duas formas acima.

Linguagens LR

As linguagens geradas pelas Gramáticas Regulares ou do Tipo 3 são chamadas de Linguagens Regulares (LR) ou Linguagens do Tipo 3.

Resultado 3:

Toda gramática do tipo 3 é também do tipo 2.

Corolário 3:

Toda LR é também uma LLC (mas nem toda LLC é LR).

Exemplo 1:

G = ({S}, {a, b}, P, S)
P = {
S -> aS
S -> b }

Exemplo 1:

G = ({S}, {a, b}, P, S)
P = {
S -> aS
S -> b }
Resp.: $L(G) = \{a^n b \mid n \geq 0\}$ ou a*b

Exemplo 2:

G = ({S, A}, {a, b, c}, P, S)
P = {
S -> aS | bA
A -> c }

Exemplo 2:

G = ({S, A}, {a, b, c}, P, S)
P = {
S -> aS | bA
A -> c }
Resp.: L(G) = {a^nbc | n ≥ 0}

Exemplo 3:

G = ({<Dig>, <Int>}, {+, -, 0, ..., 9}, P, <Int>)
P = {<Int> ::= +<Dig> | -<Dig>
<Dig> ::= 0<Dig> | 1<Dig>|...| 9<Dig> | 0 | 1 | 2 |...|9 }

Exemplo 3:

G = ({<Dig>, <Int>}, {+, -, 0, ..., 9}, P, <Int>)
P = {<Int> ::= +<Dig> | -<Dig>
<Dig> ::= 0<Dig> | 1<Dig>|...| 9<Dig> | 0 | 1 | 2 |...|9 }

Resp.:
L(G) = conj. números inteiros com sinal $\pm[0..9]^{+}$

Exemplo 4:

G = ({A,B,C}, {0,1}, P, A)
P = { A -> 0B | 0
B -> 1C
C -> 0B | 0 }

Exemplo 4:

$G = (\{A,B,C\}, \{0,1\}, P, A)$

$P = \{ A \rightarrow 0B \mid 0$

$B \rightarrow 1C$

$C \rightarrow 0B \mid 0 \}$

Resp.:

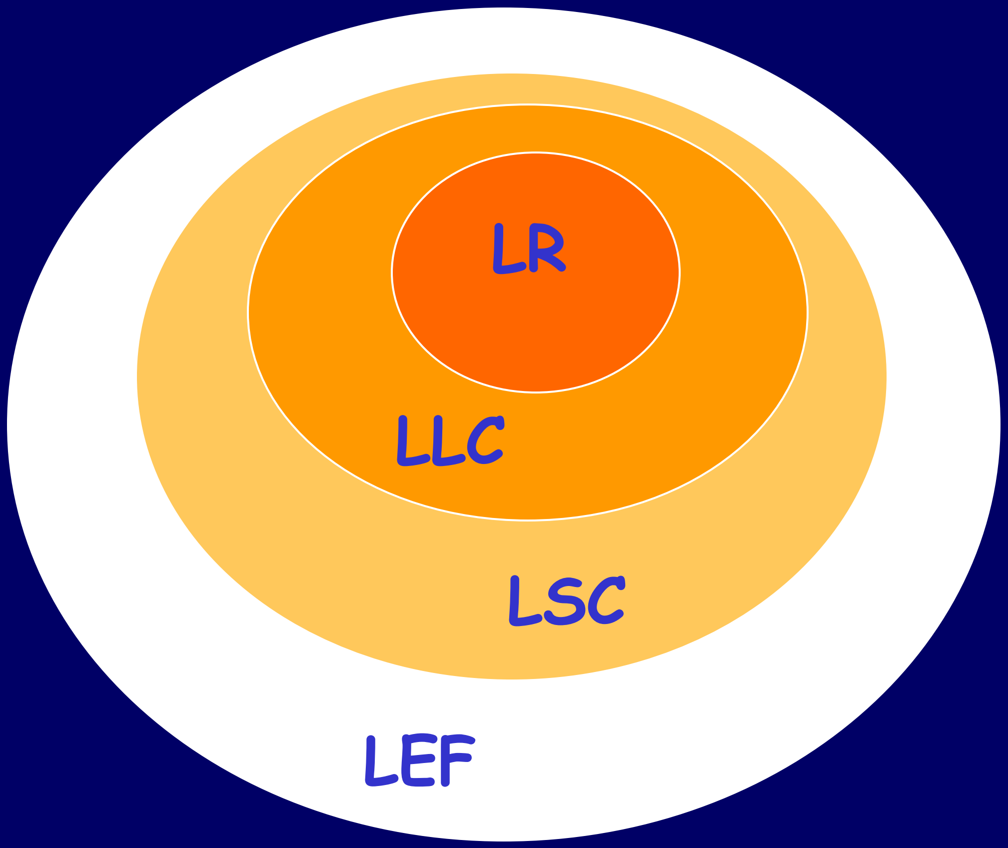
$L(G) = \{0(10)^*\}$

Hierarquia de Chomsky

Em termos gerais, para $n \in \{0, 1, 2, 3\}$ pode-se afirmar que uma *linguagem* de qualquer tipo pode ser classificada também como sendo de tipo menor, de acordo com a **Hierarquia de Chomsky**.

Uma linguagem do tipo n é caracterizada pela existência de alguma gramática do tipo n que a descreva.

Hierarquia de Chomsky



LR = Linguagens Regulares

LLC = Linguagens Livres de Contexto

LSL =Linguagens Sensíveis ao Contexto

LEF = Linguagens com Estrutura de Frase

Classifique as gramáticas, dê a quádrupla e a L(G) e diga se as ling são finitas/infinitas

1) E -> E + E | E - E | E * E | E / E | (E) | F
   F -> 0 | 1 | ... | 9

2) A -> BC
   BC -> CB
   B -> b
   C -> a

3) A -> 0A | B
   B -> 1B | λ

4) S -> 0A

A -> 1S | 1

5) S -> 0A

A -> 1B

B -> 1S | 1

6) L(G6) = $\{111(00)^n \mid n >= 0\}$

G6 = ?

7) L(G7) = $\{a^n b^n c^i \mid n >= 1$ e $i >= 0\}$

G7 = ?

8) L(G8) = $\{a^j b^n c^n \mid n >= 1$ e $j >= 0\}$

G8 = ?

9) Utilize o software JFLAP com os exemplos acima