TRANSFORMAÇÃO DE GRAMÁTICAS LIVRES DO CONTEXTO PARA EXPRESSÕES REGULARES ESTENDIDAS

Acadêmico: Cleison Vander Ambrosi
Orientador: José Roque Voltolini da Silva

Roteiro da Apresentação

- Introdução
- Motivação
- Objetivos
- Hierarquia de Chomsky
- Tipos de Recursividade
- Recursividade Não-Auto-Embutida à Esquerda
- Recursividade Não-Auto-Embutida à Direita
- Recursividade Auto-Embutida
- Montagem da Expressão Regular
- Teorema de Kleene

Introdução

- Mostrar uma forma de conversão de uma linguagem livre do contexto na íntegra para uma expressão regular estendida;
- Existem métodos para transformação de uma linguagem livre do contexto, mas não na sua totalidade para expressões regulares;
- Problema quando existe definição auto-embutida (recursividade).

Motivação

- O fato que motivou o desenvolvimento deste trabalho foi a relação entre linguagens livres do contexto e linguagens regulares e a possibilidade de validar as linguagens livres do contexto utilizando os teoremas usados para validar as linguagens regulares;

- O assunto sobre transformações de uma linguagem livre do contexto na sua íntegra para uma expressão regular não foi encontrado na literatura tradicional;

Objetivos

- transformar linguagens livres do contexto em uma expressão regular estendida;
- eliminar a recursividade que possa surgir no processo de transformação;
- aplicar o Teorema de Kleene para criação do autômato finito determinístico.

Hierarquia de Chomsky

Composição das produções de uma Gramática Livre do Contexto

Símbolo ‘Pode Ser’

uma Opção

Operador ‘OU’

$$N \rightarrow A \;|\; \text{‘y’} \;|\; \#id \;|\; \wedge$$

Símbolo vazio

Identificador léxico

Literal

Não-Terminal

Não-Terminal

Tipos de Recursividade que podem ser encontrados

$$N \rightarrow \alpha_1 \mid N\,\beta_1 \mid \alpha_n$$

- Não-Auto-Embutidas
  - À direita
  - À esquerda
- Auto-Embutidas

Recursividade Não-Auto-Embutida à Esquerda

$$N \rightarrow N\,\alpha_1 \mid \beta_1 \mid N\,\alpha_n \mid \beta_n$$

Opções com Recursividade

$$N \rightarrow (\beta_1 \mid \beta_n)\;(\alpha_1 \mid \alpha_n)\,*$$

Operador Estrela

Recursividade Não-Auto-Embutida à Direita

$$N \rightarrow \underbrace{\alpha_1 N} \mid \beta_1 \mid \underbrace{\alpha_n N} \mid \beta_n$$

Opções com
Recursividade

$$N \rightarrow (\alpha_1 \mid \alpha_n) * (\beta_1 \mid \beta_n)$$

Operador Estrela

Recursividade Auto-Embutida

$$N \rightarrow \underbrace{\alpha_1 \, N \, \alpha_n}_{\text{Opção com Recursividade}} \mid \beta_1 \mid \beta_n$$

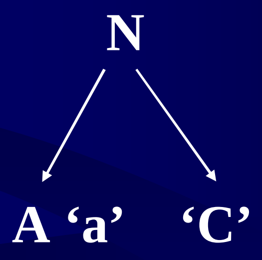
$$N \rightarrow \alpha_1 \sim (\beta_1 \mid \beta_n) \; \alpha_n \text{”}$$

Operadores de Iteração

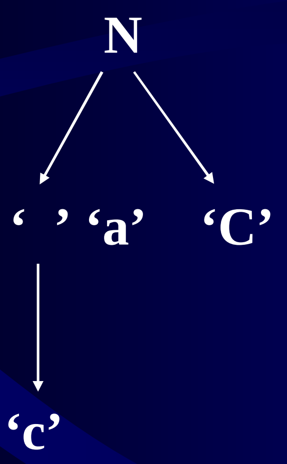
Montagem da Expressão Regular

N→ A 'a' | C
A→ 'c'
C→ 'd' C | 'x'

('d')* ('x')

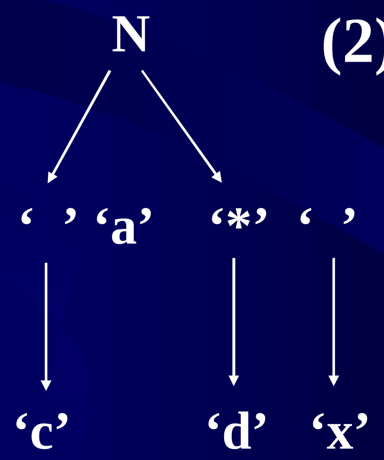


(1)



(‘c’) ‘a’ | C

(2)



(‘c’) ‘a’ | ('d') * ('x')

Árvore com os operadores de iteração

N → a N b | x

N → a~ (x) b”

Teorema de Kleene

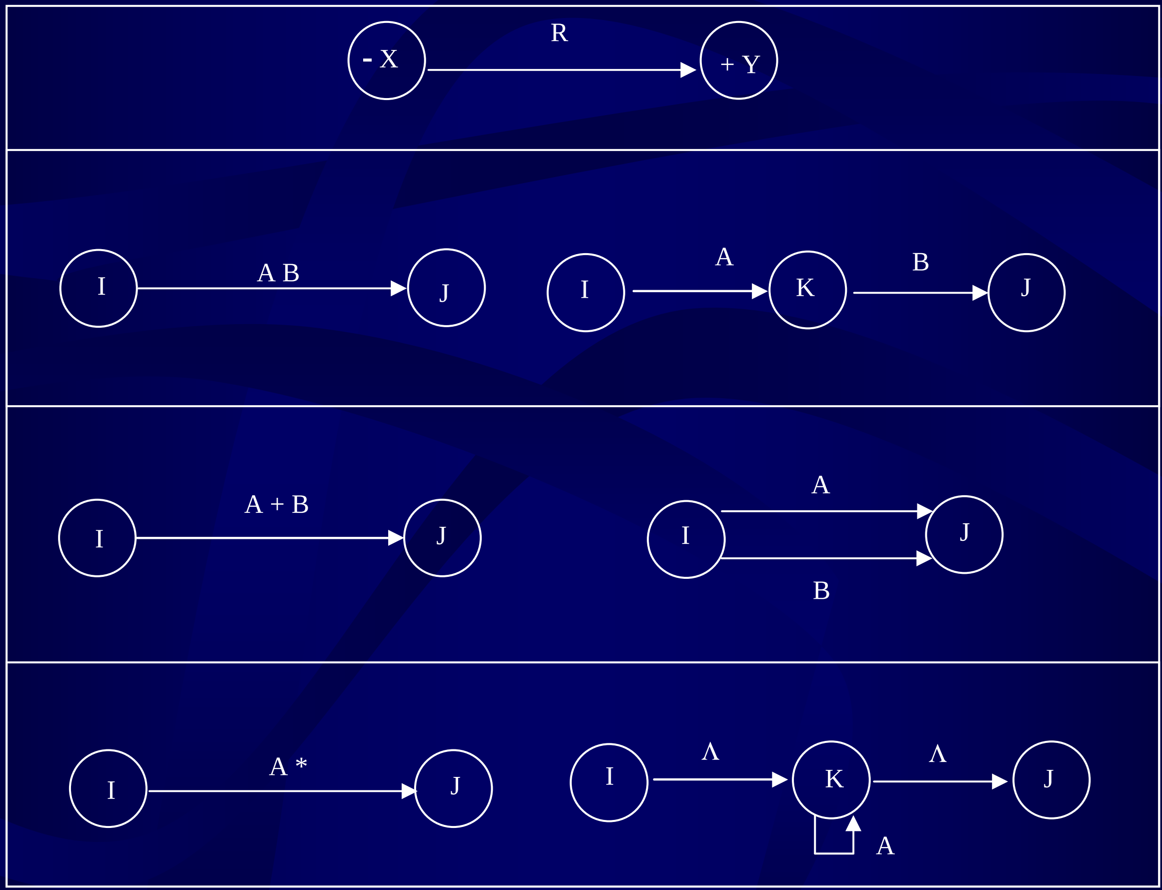
"Qualquer conjunto reconhecido por uma máquina de estado finito é regular, e qualquer conjunto regular pode ser reconhecido por uma máquina de estado finito determinístico".

Transformação de uma expressão regular em um autômato finito determinístico pelo método tradicional de Kleene

Para tanto são necessários os seguintes passos:

- transformar a expressão regular em um grafo de transições;
- transformar o grafo de transições em uma tabela de transições;
- a partir da tabela gerar o autômato finito determinístico.

Algoritmo de Kleene para transformação da expressão regular em um grafo de transições



Transformação da Expressão Regular: a (b | c)* em um Grafo de Transições pelo método tradicional de Kleene

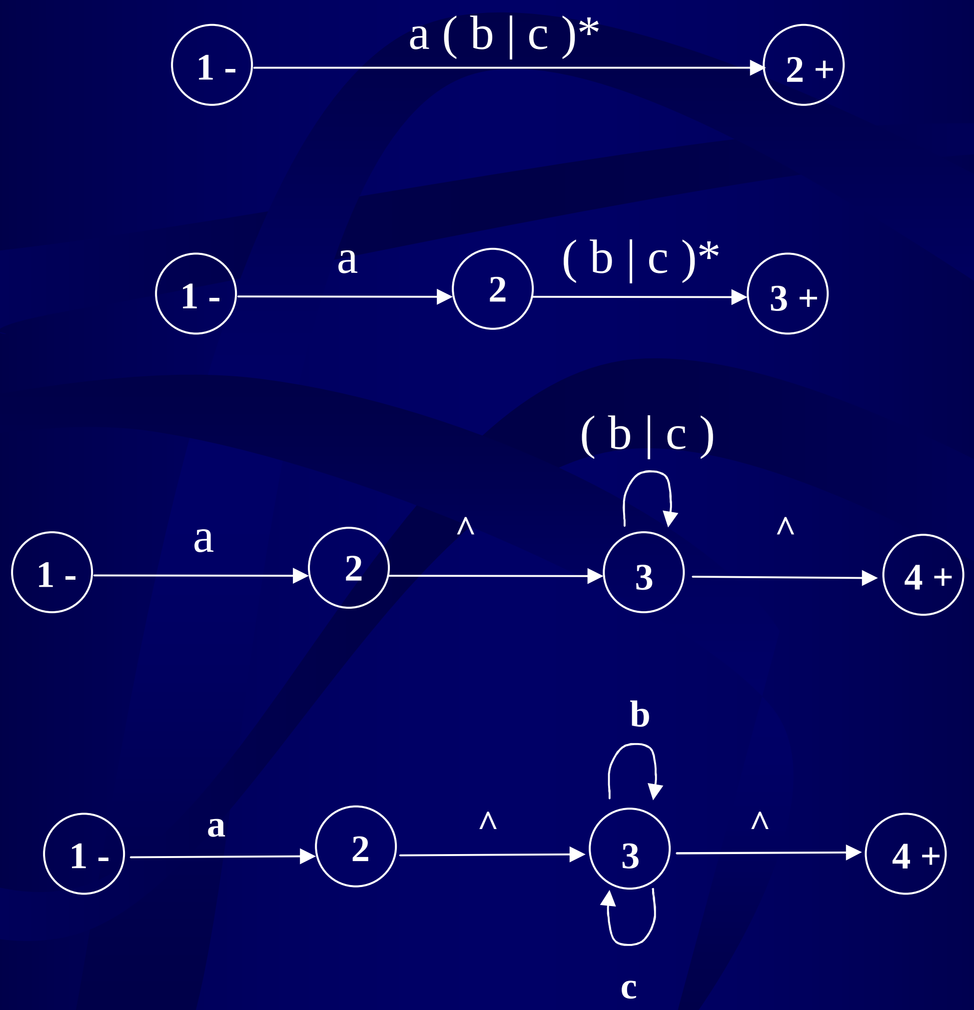
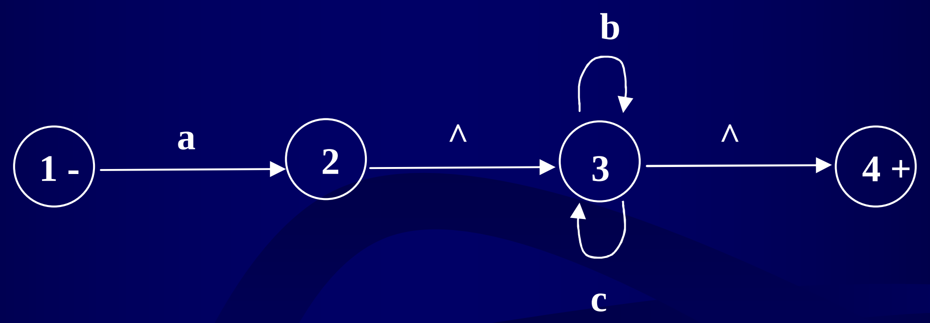
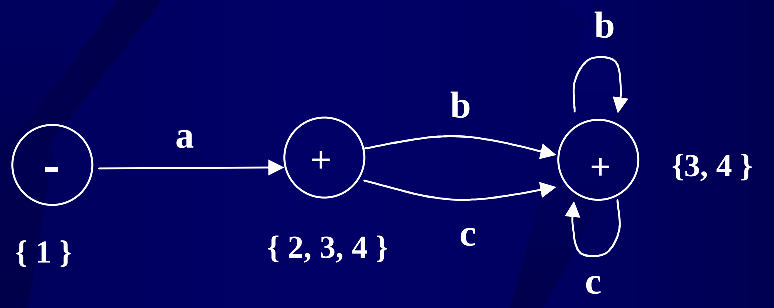


Tabela de transições



mi	a	b	c	final
{ 1 }	{ 2, 3, 4 }	{ ^ }	{ ^ }	
{ 2, 3, 4 }	{ ^ }	{ 3, 4 }	{ 3, 4 }	+
{ 3, 4 }	{ ^ }	{ 3, 4 }	{ 3, 4 }	+



Algoritmo do Desmonte

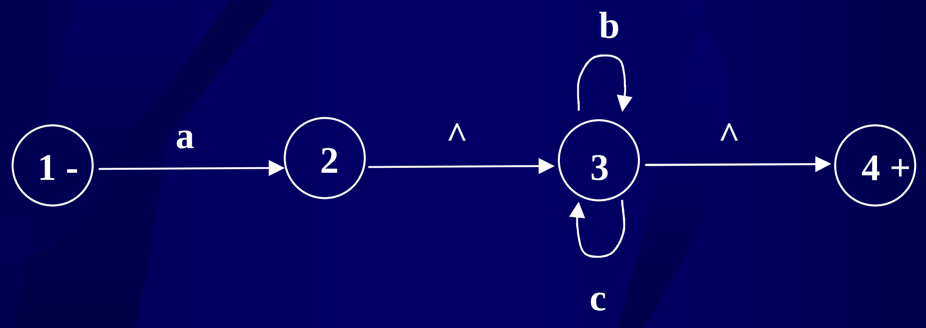
- transformar a expressão regular da forma in-fixa **a (b | c)*** para forma pós-fixa;
- montar a tabela para construção do grafo para a expressão regular na forma pós-fixa: abc|*. ;
- gerar o autômato finito determinístico a partir desta tabela.

Tabela para construção do grafo para a expressão pós-fixa: **abc|*.**

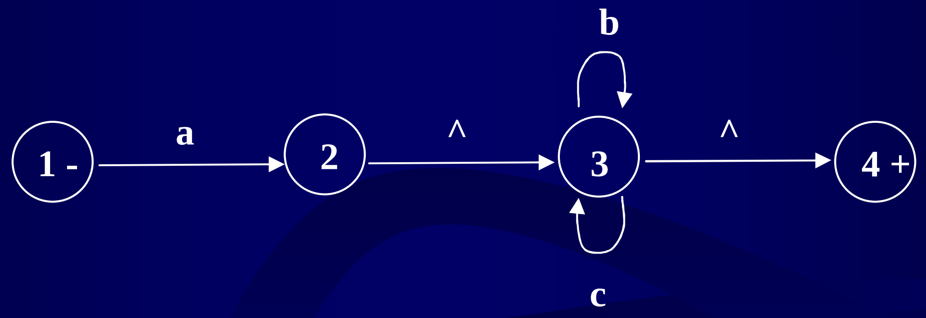
Nó de Saída	**Símbolo**	**Próximo Nó**
1	**.**	**4**
1	**a**	**2**
2	*****	**4**
2	**^**	**3**
3	**^**	**4**
3	**|**	**3**
3	**b**	**3**
3	**c**	**3**

Grafo de Transições

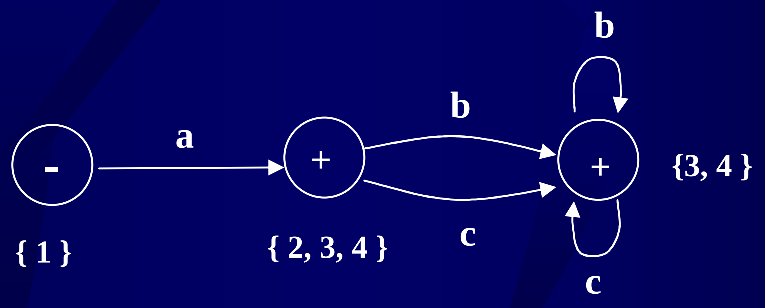
Nó de Saída	Símbolo	Próximo Nó
1	a	2
2	^	3
3	^	4
3	b	3
3	c	3



Autômato finito determinístico



mi	a	b	c	final
{ 1 }	{2,3,4}	{ ^ }	{ ^ }	
{ 2, 3, 4}	{ ^ }	{3, 4}	{3, 4}	+
{ 3, 4}	{ ^ }	{3, 4}	{3, 4}	+



Conclusão

- Para atingir o objetivo de transformar L.L.C. para E.R. foram introduzidos 2 novos operadores de iteração (‘~’, ‘”’) , modificando suas características originais tornando-a estendida;
- Para duas classificações de linguagem (regulares e livres do contexto) - duas notações diferentes para representá-las - com este trabalho é proposto uma única notação para representar os dois tipos de linguagem;
- foi necessário retirar todos os tipos de recursividade, o que inicialmente não havia sido previsto;

Extensões

- os operadores de iteração deverão ser tratados na transformação do grafo para autômato finito determinístico;
- fazer o tratamento de ações semânticas dirigidas pela sintaxe quando da transformação de uma linguagem livre do contexto para expressão regular;