

Construção de Compiladores

Período Especial

Aula 18: Análise Sintática Ascendente

Bruno Müller Junior

Departamento de Informática
UFPR

2020

Introdução

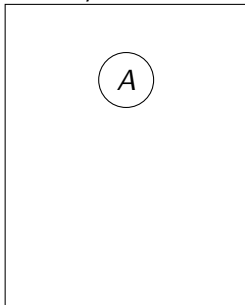
- Esta aula baseia-se no artigo “LR Parsing”, A. V. AHO, S. C. JOHNSON, ACM Computing Surveys, Vol 6, No 2, June 1974.
- Sugerimos fortemente a leitura deste artigo. É muito bem escrito, muito didático.
- Outros exemplos usados aqui estão no livro do Tomasz.

Conceituação

- Analisadores Sintáticos Ascendentes constroem a árvore de derivação da direita para a esquerda.
- Dentre os analisadores desta categoria, destacam-se os LR (*left to right parsing producing rightmost derivation*), que serão alvo do estudo.
- Uma das diferenças mais importantes para os descendentes (já estudados) é a forma de usar a produção para construir a árvore. Por exemplo, seja a produção $A \rightarrow b$:
 - um analisador descendente usa derivações.
 - um analisador ascendente usa reduções.

Derivações e Reduções

Derivação $A \rightarrow b$

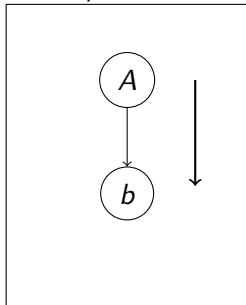


Redução $A \rightarrow b$



Conceituação

Derivação $A \rightarrow b$

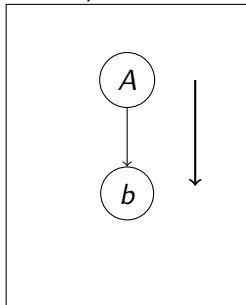


Redução $A \rightarrow b$

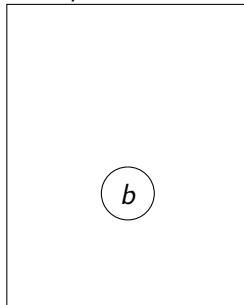


Conceituação

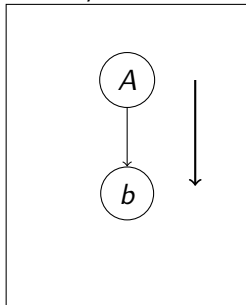
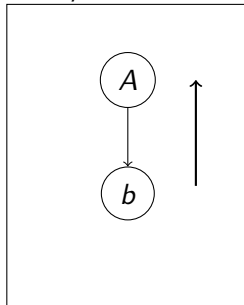
Derivação $A \rightarrow b$



Redução $A \rightarrow b$



Conceituação

Derivação $A \rightarrow b$ Redução $A \rightarrow b$ 

Análises Ascendentes

- Existem vários analisadores sintáticos nesta categoria:
 - Análise de precedência simples
 - Análise de precedência de operadores
 - Análise LR, que também tem várias abordagens:
 - SLR(0)
 - SLR(1)
 - LALR(1)
 - LR(1)
- Listados do mais restrito para o mais geral:

$$L(SLR(0)) \subset L(SLR(1)) \subset L(LR(1)) = L(LALR(1))$$

- Iremos estudar somente o SLR(0) e o SLR(1), sugerindo como funcionam os demais.
- O bison usa o método LALR(1).

Parser LR

- Componentes

Entrada contém α ;

Parser código executável;

Floresta estrutura de dados auxiliar;

Tabela ação matriz obtida a partir de G ;

Tabela desvios matriz obtida a partir de G ;

- Graficamente

$\alpha =$

		...	
--	--	-----	--



Parser LR

Código Estático

Floresta LR

Tabelas de Ação/Desvio

Gramática

Tabela de Desvios

- Representação de um grafo com uma tabela.
- linhas: vértices;
- colunas: Variáveis e terminais da gramática.
- Exemplo: considere a gramática G abaixo e tabela de desvios obtida a partir dela:

$$\begin{aligned}
 G = \{ L' &\rightarrow L\# \\
 L &\rightarrow L, E \text{ (1)} \\
 L &\rightarrow E \text{ (2)} \\
 E &\rightarrow a \text{ (3)} | b \text{ (4)}
 \end{aligned}$$

Tabela de Desvios						
	L	E	a	b	,	#
0	1	2	3	4		
1					5	
2						
3						
4						
5		6	3	4		
6						

Tabela de Desvios

	Tabela de Desvios					
	L	E	a	b	,	#
0	1	2	3	4		
1						5
2						
3						
4						
5		6	3	4		
6						

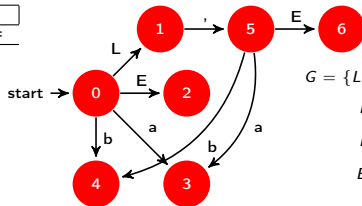

 $G = \{L'\}$
 $\rightarrow L\#$
 $L \rightarrow L, E \textcircled{1}$
 $L \rightarrow E \textcircled{2}$
 $E \rightarrow a \textcircled{3} | b \textcircled{4}$

Tabela de Ações

- A tabela de ações indica o que deve ser feito em um vértice do grafo em função do token da entrada.

e (empilha)

R (Reduz)

A (Aceita)

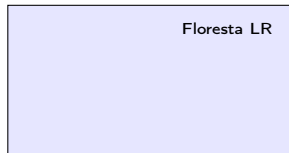
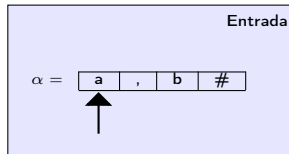
(erro)

Tabela de Ações				
	a	b	,	#
0	e	e		
1			e	A
2			R ₂	R ₂
3			R ₃	R ₃
4			R ₄	R ₄
5	e	e		
6			R ₁	R ₁

Empilha

- cria um nova árvore com o token atual;
- consome o token;
- Exemplo: Ação(5,a)

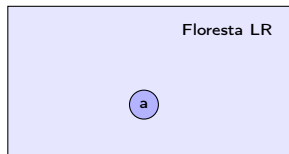
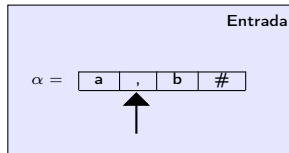
Tabela de Ações				
	a	b	,	#
0	e	e		
1			e	A
2			R_2	R_2
3			R_3	R_3
4			R_4	R_4
5	\odot	e		
6			R_1	R_1



Empilha

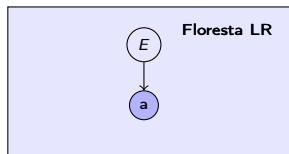
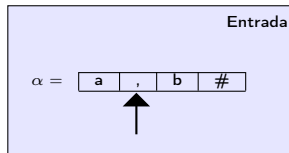
- cria uma nova árvore com o token atual;
- consome o token;
- Exemplo: Ação(5,a)

Tabela de Ações				
	a	b	,	#
0	e	e		
1			e	A
2			R_2	R_2
3			R_3	R_3
4			R_4	R_4
5	\odot	e		
6			R_1	R_1

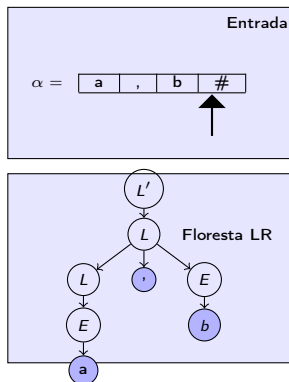


-
- Floresta LR
- a

- | | a | b | , | # |
|---|---|---|-------|-------|
| 0 | e | e | | |
| 1 | | | e | A |
| 2 | | | R_2 | R_2 |
| 3 | | | R_3 | R_3 |
| 4 | | | R_4 | R_4 |
| 5 | e | e | | |
| 6 | | | R_1 | R_1 |



- | Tabela de Ações | | | | |
|-----------------|---|---|-------|-------|
| | a | b | , | # |
| 0 | e | e | | |
| 1 | | | e | (A) |
| 2 | | | R_2 | R_2 |
| 3 | | | R_3 | R_3 |
| 4 | | | R_4 | R_4 |
| 5 | e | e | | |
| 6 | | | R_1 | R_1 |



- As tabelas são obtidas a partir da gramática.
- Se mudar a gramática, mudam as tabelas.
- Estudaremos:
 - Como funciona o algoritmo com tabelas dadas (G).
 - Como construir as tabelas $SLR(0)$ - a mais restrita;
 - Como criar tabelas para classes mais amplas de gramáticas com $SLR(1)$;
 - Sugerir como estender ainda mais ($LR(1)$, $LALR(1)$).

Fusão das tabelas

- Para ocupar menos espaço, é comum combinar as tabelas em uma só.

Tabela de Ações				
	a	b	,	#
0	e	e		
1			e	A
2			R_2	R_2
3			R_3	R_3
4			R_4	R_4
5	e	e		
6			R_1	R_1

$G_1 = \{L' \rightarrow L\#$
 $L \rightarrow L, E \textcircled{1}$
 $L \rightarrow E \textcircled{2}$
 $L \rightarrow a \textcircled{3}$
 $L \rightarrow b \textcircled{4}$

Tabela de Desvios						
	L	E	a	b	,	#
0	1	2	3	4		
1					5	
2						
3						
4						
5		6	3	4		
6						

Fusão das tabelas

- vermelho: tabela de ações;
- azul: tabela de desvios;

$$\begin{aligned}
 G_1 = \{L' &\rightarrow L\# \\
 &L \rightarrow L, E \textcircled{1} \\
 &L \rightarrow E \textcircled{2} \\
 &L \rightarrow a \textcircled{3} \\
 &L \rightarrow b \textcircled{4}
 \end{aligned}$$

Tabela de Ação e Desvios						
	L	E	a	b	,	#
0	1	2	e3	e4		
1					e5	A
2					R ₂	R ₂
3					R ₃	R ₃
4					R ₄	R ₄
5		6	e3	e4		
6					R ₁	R ₁

Execução

- Cada passo do parser usa como parâmetros:
 - o token de entrada;
 - a floresta de sub-árvores;
 - os estados, que podem ser visto como:
 - uma linha das tabelas;
 - o estado indicado no topo de cada árvore (cada árvore tem que ter um estado no topo);
- Aqui será visto uma descrição informal do algoritmo.
- A descrição formal pode ser encontrada no artigo usado como referência.

$\alpha =$

a , b #

Parser LR

Código Estático

Floresta LR

Tabelas de Ação/Desvio

	L	E	a	b	,	#
0	1	2	e3	e4		
1					e5	A
2					R ₂	R ₂
3					R ₃	R ₃
4					R ₄	R ₄
5		6	e3	e4		
6					R ₁	R ₁

Gramática

$G =$
 $L' \rightarrow L\#$ (1)
 $L \rightarrow L, E$ (2)
 $L \rightarrow E$ (3)
 $E \rightarrow a$ (4)
 $E \rightarrow b$ (5)

Código Estático

1. lê token
2. coloca estado inicial na floresta

0
1
2
3
4
5
6

L	E	a	b	,	#
1	2	e3	e4		
				e5	A
				R ₂	R ₂
				R ₃	R ₃
				R ₄	R ₄
	6	e3	e4		
				R ₁	R ₁

$$\begin{array}{llll} G = & L' & \rightarrow & L\# \\ & L & \rightarrow & L, E \\ & L & \rightarrow & E \\ & E & \rightarrow & a \\ & E & \rightarrow & b \end{array} \quad \begin{array}{l} (1) \\ (2) \\ (3) \\ (4) \end{array}$$

```

graph TD
    L_prime((L')) --> L1((L))
    L_prime --> L2((L))
    L_prime --> E1((E))
    L1 --> L3((L))
    L1 --> E2((E))
    L1 --> a1((a))
    L3 --> L4((L))
    L3 --> E3((E))
    L3 --> a2((a))
    L4 --> b1((b))
    E2 --> b2((b))
    L2 --> L5((L))
    L2 --> E4((E))
    L2 --> a3((a))
    L5 --> L6((L))
    L5 --> E5((E))
    L5 --> a4((a))
    L6 --> b3((b))
    E4 --> b4((b))
  
```


$\alpha =$

a	,	b	#
---	---	---	---



Parser LR

Código Estático

1. Tabela[0,a]: empilha e vai para estado 3
2. empilha token atual (cria árvore na floresta)
2. coloca novo estado no topo desta árvore

Tabelas de Ação/Desvio

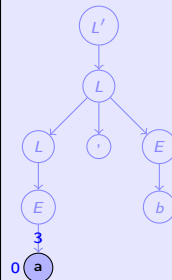
	L	E	a	b	,	#
0	1	2	e3	e4		
1					e5	A
2					R ₂	R ₂
3					R ₃	R ₃
4					R ₄	R ₄
5		6	e3	e4		
6					R ₁	R ₁

Gramática

G=

$L' \rightarrow L\#$	1
$L \rightarrow L, E$	2
$L \rightarrow E$	3
$E \rightarrow a$	4
$E \rightarrow b$	4

Floresta LR



$\alpha =$

a	,	b	#
---	---	---	---



Parser LR

Tabela[3,",#]: R_3

Código Estático

1. Aplica a redução número 3 ($E \rightarrow a$)
2. Observe que a árvore ficou sem estado.
3. **TODA ÁRVORE PRECISA DE UM ESTADO**
4. Aplica Estado da árvore anterior com topo da árvore [0,E]=2

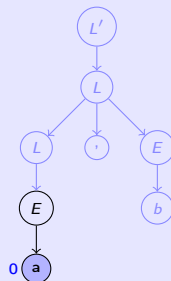
Tabelas de Ação/Desvio

	L	E	a	b	,	#
0	1	2	e3	e4		
1					e5	A
2					R_2	R_2
3					R_3	R_3
4					R_4	R_4
5		6	e3	e4		
6					R_1	R_1

Gramática

$G =$ $L' \rightarrow L\#$ ①
 $L \rightarrow L, E$ ②
 $L \rightarrow E$ ③
 $E \rightarrow a$ ④
 $E \rightarrow b$

Floresta LR



$\alpha =$

a	,	b	#
---	---	---	---



Parser LR

Código Estático

1. coloca estado no topo da árvore: Tabela[0,E]=2

Tabelas de Ação/Desvio

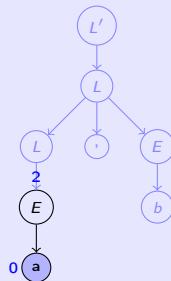
	L	E	a	b	,	#
0	1	2	e3	e4		
1					e5	A
2					R ₂	R ₂
3					R ₃	R ₃
4					R ₄	R ₄
5		6	e3	e4		
6					R ₁	R ₁

Gramática

G=

$L' \rightarrow L\#$	①
$L \rightarrow L, E$	②
$L \rightarrow E$	③
$E \rightarrow a$	④
$E \rightarrow b$	

Floresta LR



Código Estático

1. Todas árvores com estado: prossegue na tabela
2. $Tabela[2, ""] = R_2$
3. Aplica a redução número 2 ($L \rightarrow E$)

Tabelas de Ação/Desvio

0	1	2	e3	e4		
1					e5	A
2					R_2	R_2
3					R_3	R_3
4					R_4	R_4
5		6	e3	e4		
6					R_1	R_1

Gramática

$$G = \begin{array}{lll} L' & \rightarrow & L\# \\ L & \rightarrow & L, E \\ L & \rightarrow & E \\ E & \rightarrow & a \\ E & \rightarrow & b \end{array} \quad \begin{array}{l} (1) \\ (2) \\ (3) \\ (4) \end{array}$$

```

graph TD
    L_prime((L')) --> L1((L))
    L1 --> L2((L))
    L1 --> plus((+))
    L1 --> E1((E))
    L2 --> E2((E))
    E2 --> a((a))
    E1 --> b((b))
    style a fill:#0000FF,color:#FFFFFF
  
```

$\alpha =$

a	,	b	#
---	---	---	---



Parser LR

Código Estático

1. Árvore sem estado
2. TODA ÁRVORE PRECISA DE UM ESTADO
3. Aplica Estado da árvore anterior com topo da árvore $Tabela[0,L]=1$

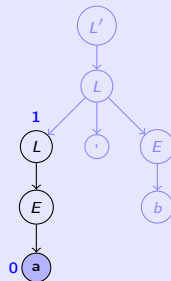
Tabelas de Ação/Desvio

	L	E	a	b	,	#
0	1	2	e3	e4		
1					e5	A
2					R ₂	R ₂
3					R ₃	R ₃
4					R ₄	R ₄
5		6	e3	e4		
6					R ₁	R ₁

Gramática

$G =$ $L' \rightarrow L\#$ ①
 $L \rightarrow L, E$ ②
 $L \rightarrow E$ ③
 $E \rightarrow a$ ④
 $E \rightarrow b$

Floresta LR





Código Estático

1. Tabela[1,""]=e5
2. Empilha ","e coloca estado 5 na árvore

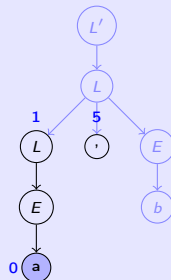
Tabelas de Ação/Desvio

	L	E	a	b	,	#
0	1	2	e3	e4		
1					e5	A
2					R ₂	R ₂
3					R ₃	R ₃
4					R ₄	R ₄
5		6	e3	e4		
6					R ₁	R ₁

Gramática


$$G = \begin{array}{lll} L' & \rightarrow & L\# \\ L & \rightarrow & L, E \\ L & \rightarrow & E \\ E & \rightarrow & a \\ E & \rightarrow & b \end{array} \quad \begin{array}{l} (1) \\ (2) \\ (3) \\ (4) \end{array}$$

Floresta LR



$\alpha =$

a	,	b	#
---	---	---	---

 Parser LR

Código Estático

1. Tabela[5,"b"]=e4
2. Empilha "b" e coloca estado 4 na árvore

Tabelas de Ação/Desvio

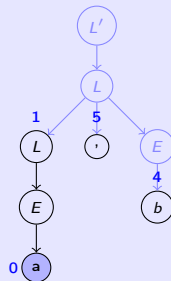
	L	E	a	b	,	#
0	1	2	e3	e4		
1					e5	A
2					R ₂	R ₂
3					R ₃	R ₃
4					R ₄	R ₄
5		6	e3	e4		
6					R ₁	R ₁

Gramática

G=

$L' \rightarrow L\#$	①
$L \rightarrow L, E$	②
$L \rightarrow E$	③
$E \rightarrow a$	④
$E \rightarrow b$	

Floresta LR



$\alpha =$

a	,	b	#
---	---	---	---



Parser LR

Código Estático

1. Tabela[4,""]=R4

Tabelas de Ação/Desvio

 0
1
2
3
4
5
6

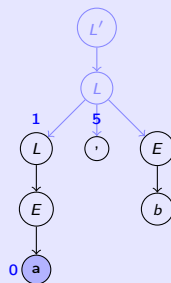
L	E	a	b	,	#
1	2	e3	e4		
				e5	A
				R ₂	R ₂
				R ₃	R ₃
				R ₄	R ₄
	6	e3	e4		
				R ₁	R ₁

Gramática

G=

L'	\rightarrow	$L\#$	①
L	\rightarrow	L, E	②
L	\rightarrow	E	③
E	\rightarrow	a	④
E	\rightarrow	b	

Floresta LR

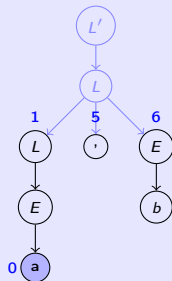


Código Estático

1. Coloca Estado Tabela[5,E]=6

0
1
2
3
4
5
6

L	E	a	b	,	#
1	2	e3	e4		
				e5	A
				R ₂	R ₂
				R ₃	R ₃
				R ₄	R ₄
	6	e3	e4		
				R ₁	R ₁

$$G = \begin{array}{lcl} L' & \rightarrow & L\# \\ L & \rightarrow & L, E \\ L & \rightarrow & E \\ E & \rightarrow & a \\ E & \rightarrow & b \end{array}$$


↑ Parser LR

1. Tabela[5,#]=R1 ($L \rightarrow L, E$)

0
1
2
3
4
5
6

L	E	a	b	,	#
1	2	e3	e4		
				e5	A
				R ₂	R ₂
				R ₃	R ₃
				R ₄	R ₄
	6	e3	e4		
				R ₁	R ₁

$$G = \begin{array}{lcl} L' & \rightarrow & L\# \\ L & \rightarrow & L, E \\ L & \rightarrow & E \\ E & \rightarrow & a \\ E & \rightarrow & b \end{array}$$

```

graph TD
    L_prime((L')) --> L1((L))
    L1 --> L1_L((L))
    L1 --> plus((+))
    L1 --> E1((E))
    L1_L --> E2((E))
    E2 --> a((a))
    E1 --> b((b))
    style L_prime fill:#d3d3d3
    style L1 fill:#d3d3d3
    style plus fill:#d3d3d3
    style a fill:#d3d3d3
    style b fill:#d3d3d3
    style L1_L fill:#d3d3d3
    style E2 fill:#d3d3d3
    style E1 fill:#d3d3d3
    style b fill:#d3d3d3

```


Parser LR

Código Estático

1. Tabela[5,#]=R1 ($L \rightarrow L, E$)

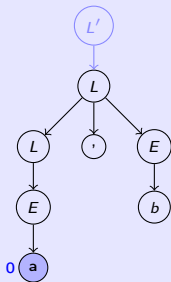
Tabelas de Ação/Desvio

0	1	2	e3	e4		
1					e5	A
2					R ₂	R ₂
3					R ₃	R ₃
4					R ₄	R ₄
5		6	e3	e4		
6					R ₁	R ₁

Gramática

$$G = \begin{array}{lll} L' & \rightarrow & L\# \\ L & \rightarrow & L, E \\ L & \rightarrow & E \\ E & \rightarrow & a \\ E & \rightarrow & b \end{array} \quad \begin{array}{l} (1) \\ (2) \\ (3) \\ (4) \end{array}$$

Floresta LR

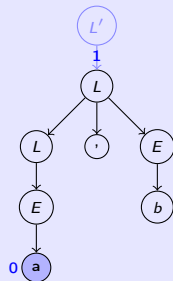


Código Estático

1. Coloca Estado Tabela[0,E] = 1

0
1
2
3
4
5
6

L	E	a	b	,	#
1	2	e3	e4		
				e5	A
				R ₂	R ₂
				R ₃	R ₃
				R ₄	R ₄
	6	e3	e4		
				R ₁	R ₁

$$G = \begin{array}{lcl} L' & \rightarrow & L\# \\ L & \rightarrow & L, E \\ L & \rightarrow & E \\ E & \rightarrow & a \\ E & \rightarrow & b \end{array}$$


↑ Parser LR

1. Tabela[1,#]=Aceita

0
1
2
3
4
5
6

L	E	a	b	,	#
1	2	e3	e4		
				e5	A
				R ₂	R ₂
				R ₃	R ₃
				R ₄	R ₄
	6	e3	e4		
				R ₁	R ₁

$$G = \begin{array}{lcl} L' & \rightarrow & L\# \\ L & \rightarrow & L, E \\ L & \rightarrow & E \\ E & \rightarrow & a \\ E & \rightarrow & b \end{array}$$

```

graph TD
    L_prime((L')) -- 1 --> L1((L))
    L1 --> L2((L))
    L1 --> comma((,))
    L1 --> E1((E))
    L2 --> E2((E))
    L2 --> a((a))
    E1 --> b((b))
    style a fill:#add8e6
    
```

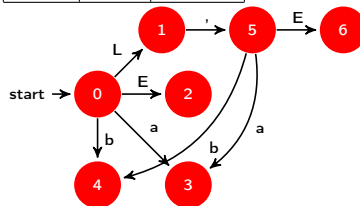
Implementação

- A demonstração de funcionamento do algoritmo usando a floresta é didática, mas não é prática.
- Uma implementação prática usa uma pilha, que representa os estados de todas as árvores da floresta.

Implementação

Floresta LR

Passo	Pilha	Entrada
Início		a, b#
1		
2		
3		
4		
5		
6		
7		
8		

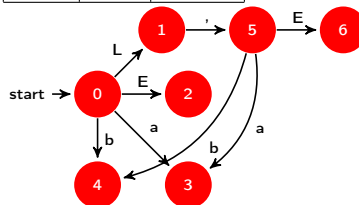


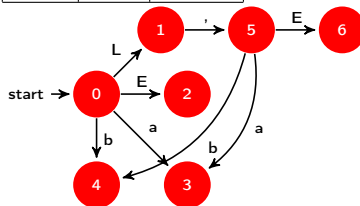
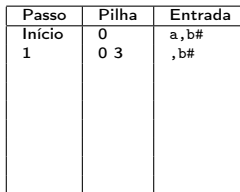
Implementação

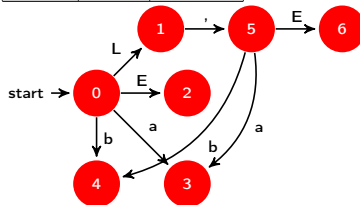
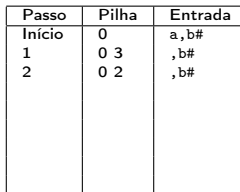
Floresta LR

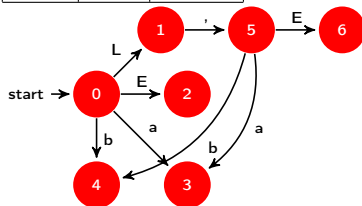
0

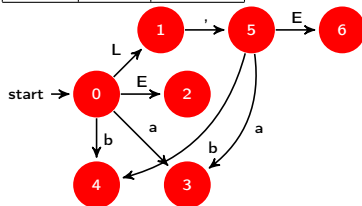
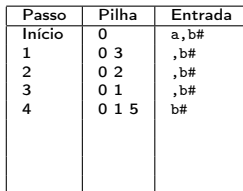
Passo	Pilha	Entrada
Início	0	a, b#

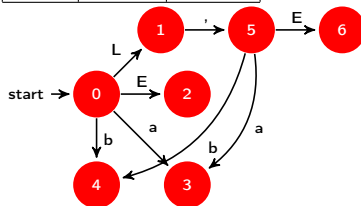
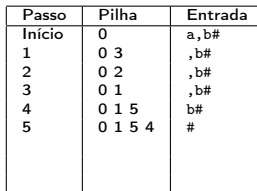






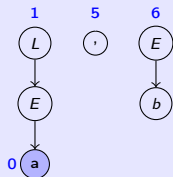




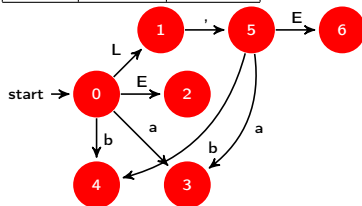


Implementação

Floresta LR

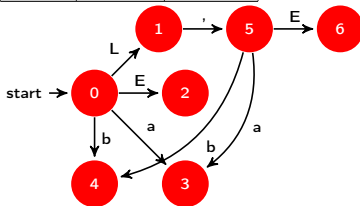


Passo	Pilha	Entrada
Início	0	a,b#
1	0 3	,b#
2	0 2	,b#
3	0 1	,b#
4	0 1 5	b#
5	0 1 5 4	#
6	0 1 5 6	#





Passo	Pilha	Entrada
Início	0	a, b#
1	0 3	, b#
2	0 2	, b#
3	0 1	, b#
4	0 1 5	b#
5	0 1 5 4	#
6	0 1 5 6	#
7	0 1	#



Exercícios

- Use a mesma gramática para verificar as seguintes entradas:
- $\alpha_1 = a\#$
- $\alpha_2 = a,\#$
- $\alpha_3 = ab\#$
- $\alpha_4 = a,b,a\#$

- O algoritmo descrito nesta aula é quase todo estático;
- O bison gera um arquivo ".c" contendo o código fonte do parser (`compilador.tab.c`);
- Para cada gramática, o bison gera “tabelas” diferentes.
- Veja o autômato a pilha em `compilador.output`
- Próximas aulas: como construir as tabelas para esta gramática;
 - ① tabelas SLR(0);
 - ② tabelas SLR(1);
 - ③ tabelas LR(1) e LALR(1);

- Página para anotações

Licença

- Slides desenvolvidos somente com software livre:
 - \LaTeX usando beamer;
 - Inkscape.
- Licença:
 - Creative Commons Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License. <http://creativecommons.org/licenses/by-nc-nd/2.5/br/>