Compiladores
Análise Sintática
Análise Sintática SLR(1)

Profa. Dra. Ana Carolina Lorena
UNIFESP

Análise Sintática SLR(1)

- LR simples com um símbolo de verificação à frente
  - Construção baseia-se em Conjunto canônico de itens LR(0)
    - Usa autômato produzido por LR(0)
    - SLR(1) é suficiente para tratar quase todas as estruturas sintáticas práticas de linguagens de programação

Análise Sintática SLR(1)

- Item LR(0): indica um passo intermediário no reconhecimento do lado direito de uma escolha específica de regra gramatical
  - É uma escolha de produção com uma posição identificada em seu lado direito
  - Ponto é indicação de até onde uma produção já foi analisada
  - Essa posição identificada será indicada por um ponto

Exemplo:

$E \rightarrow .E + n$

$E \rightarrow E. + n$

$E \rightarrow E +. n$

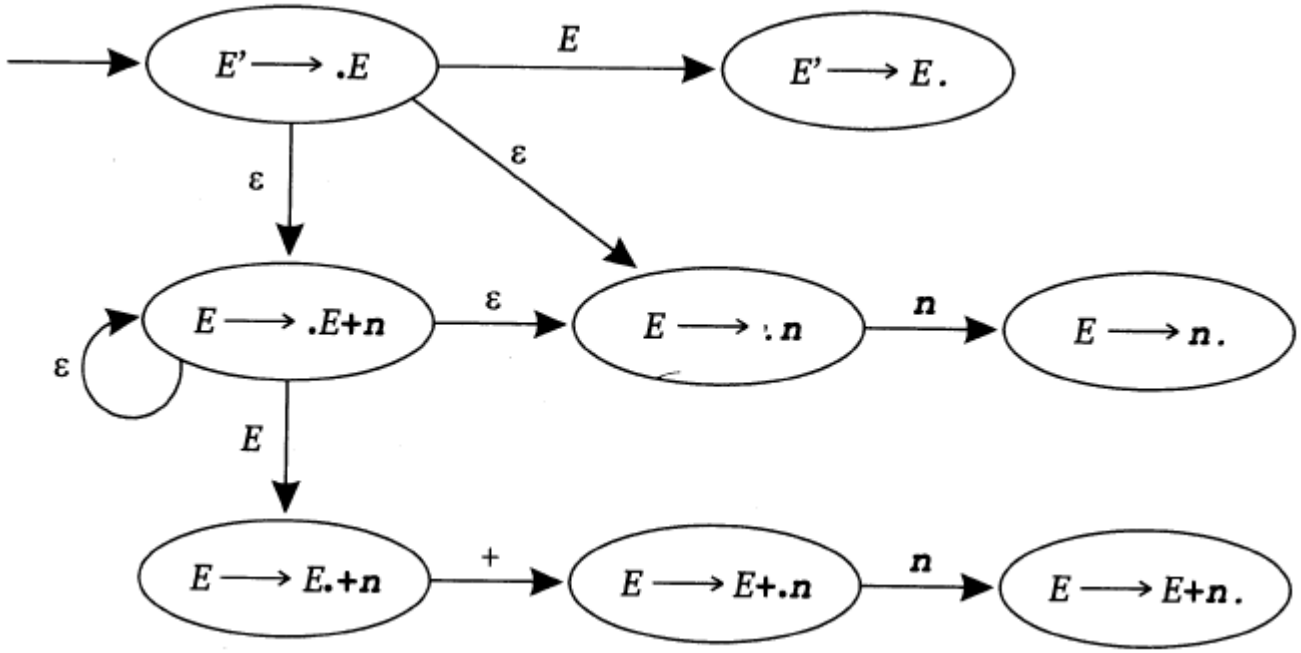
$E \rightarrow E + n.$

$E \rightarrow .n$

$E \rightarrow n.$

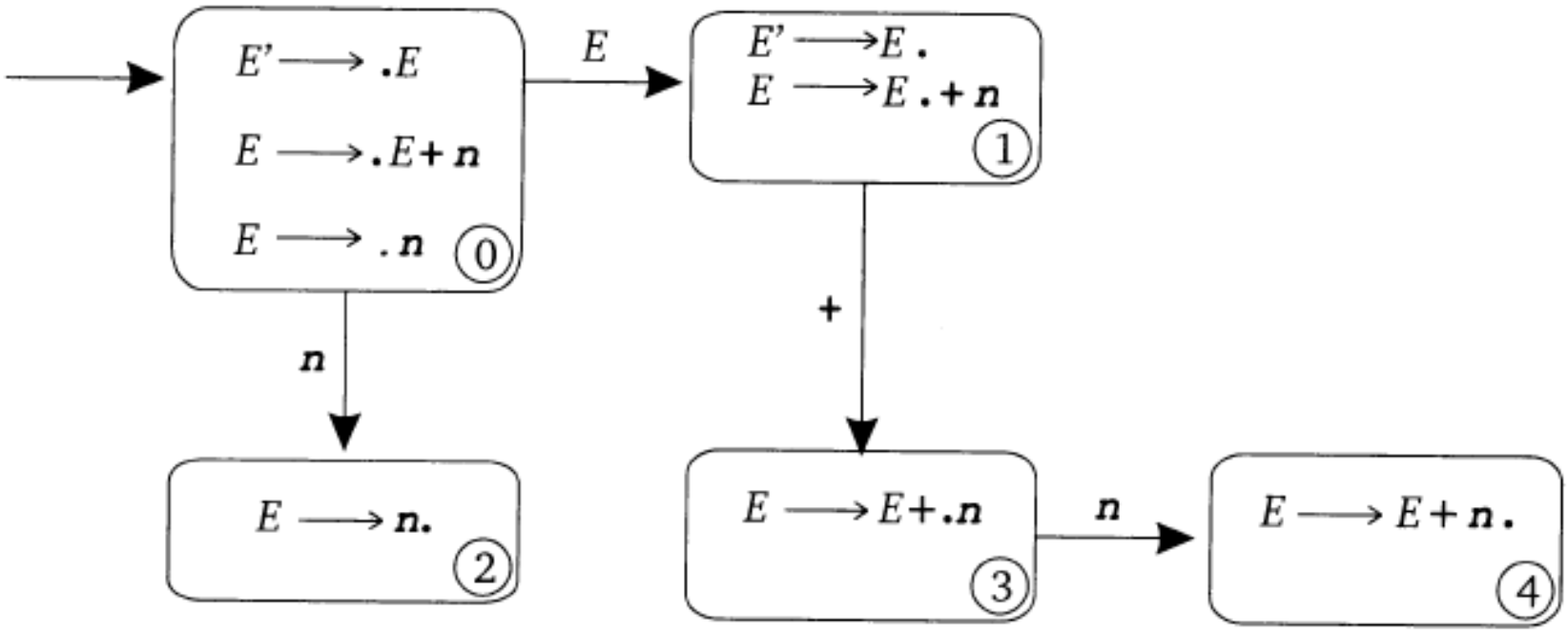
Análise Sintática SLR(1)

- AFND de itens LR(0) para a gramática:
- E' → E
- E → E+n | n



Análise Sintática SLR(1)

- AFD de itens LR(0) para a gramática:
- E' → E
- E → E+n | n



Análise Sintática SLR(1)

- Exemplos itens:

  - A → XYZ origina quatro itens:
    - A → .XYZ
    - A → X.YZ
    - A → AY.Z
    - A → XYZ.

  - A → ε origina um item:
    - A → .

Análise Sintática SLR(1)

- Construção do conjunto canônico de itens LR(0) requer:
  - Acrescentar à gramática a produção S' → S (S = símbolo inicial da gramática)
  - Computar funções fecho e ir-para para a nova gramática

Análise Sintática SLR(1)

- Função fecho(I):
  - Se I é um conjunto de itens LR(0) para G, então o conjunto de itens de fecho(I) é dado pelas regras:

    - Todo item I pertence a fecho(I)

    - Se $A \rightarrow \alpha.X\beta$ está em fecho(I) e $X \rightarrow \gamma$ é uma produção de G, então adicione $X \rightarrow .\gamma$ ao fecho(I)
      - Ou seja, aumenta conjunto fecho(I) com as produções dos não-terminais que aparecem com um ponto do lado esquerdo

Análise Sintática SLR(1)

- Exemplo: S → a | [L]
  L → L ; S | S
- Para I = {S → [.L]}, o fecho é:
  - fecho(I) = {S → [.L], L → .L;S, L → .S, S → .a, S → .[L]}

Exercício: fazer para todos os outros itens que podem ser definidos

Análise Sintática SLR(1)

- Função ir-para(I,X): avanço do ponto sobre X em I
  - Consiste em coletar as produções com ponto no lado esquerdo de X, passar o ponto para a direita de X, e obter a função fecho desse conjunto
  - Formalmente: para terminal ou não-terminal X, ir-para(I,X) = fecho dos itens $A \rightarrow \alpha X.\beta$, tais que $A \rightarrow \alpha.X\beta$ pertence a I

Análise Sintática SLR(1)

- Exemplo: considere I ={S → [L.], L → L.;S}
  - ir-para(I,;) = {L → L;.S, S → .a, S → .[L]}

Exercício: fazer para todos os outros conjuntos de itens que podem ser definidos

Análise Sintática SLR(1)

- Algoritmo para obter o conjunto canônico de itens LR(0):

  - C = {I_0 = fecho({S' → .S})}
  - **Repita**
    - **Para cada** conjunto I em C e X símbolo de G, tal que ir-para(I,X) ≠∅ adicione ir-para(I,X) em C
  - **até que** todos os conjuntos tenham sido adicionados a C

Exercício: obter o conjunto canônico de itens LR(0) para a Gramática: S → a | [L]

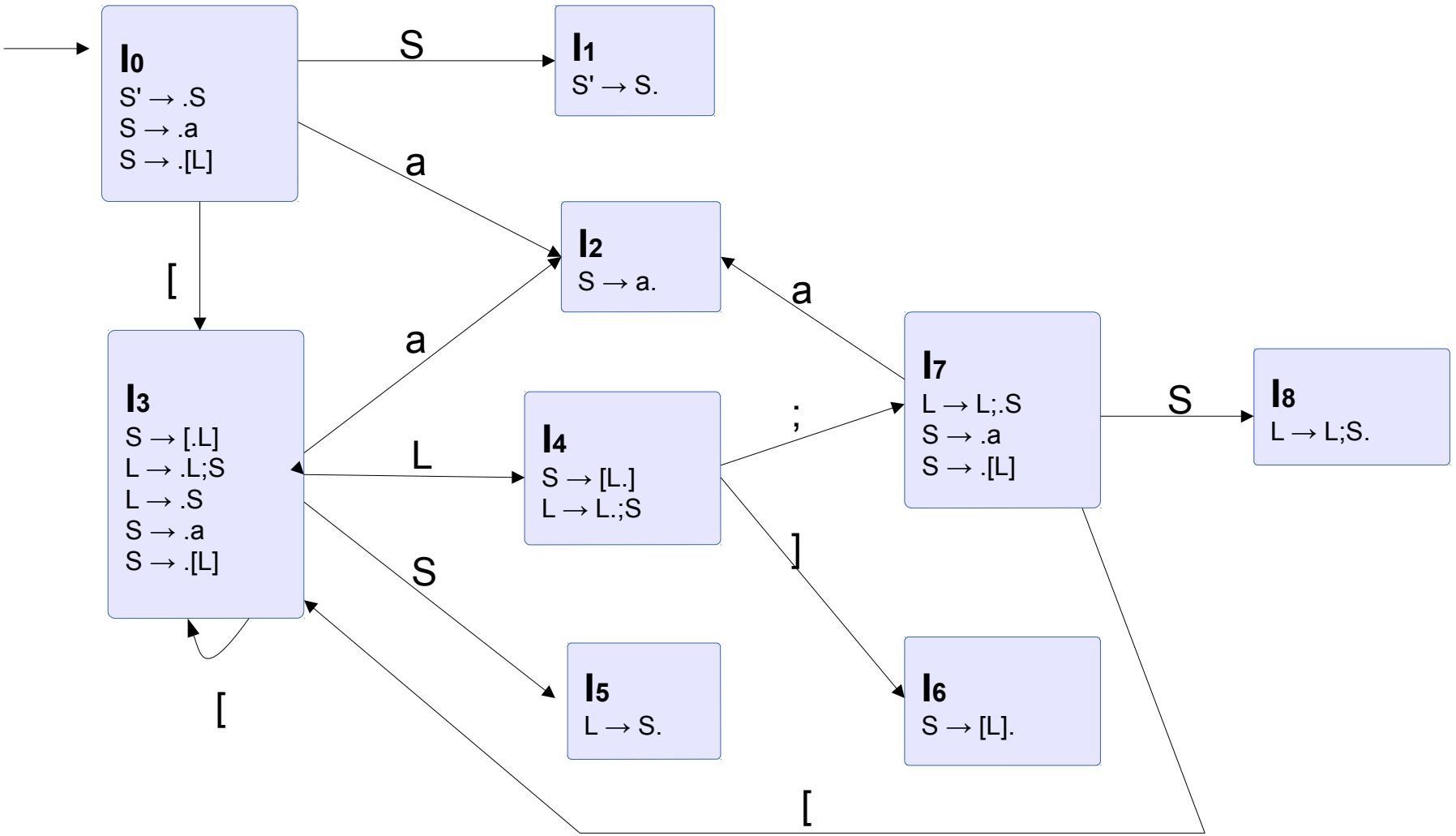
L → L ; S | S

Análise Sintática SLR(1)

- Exercício:
- S' → S
- S → a
- S → [L]
- L → L ; S
- L → S

I_0 = {S' → .S, S → .a, S → .[L]}
ir-para(I_0,S) = {S' → S.} = I_1
ir-para(I_0,a) = {S → a.} = I_2
ir-para(I_0,[) = {S → [.L], L → .L;S, L → .S, S → .a,
S → .[L]} = I_3
ir-para(I_3,L) = {S → [L.], L → L.;S} = I_4
ir-para(I_3,S) = {L → S.} = I_5
ir-para(I_3,a) = I_2
ir-para(I_3,[) = I_3
ir-para(I_4,]) = {S → [L].} = I_6
ir-para(I_4,;) = {L → L;.S, S → .a, S → .[L]} = I_7
ir-para(I_7,S) = {L → L;S.} = I_8
ir-para(I_7,a) = I_2
ir-para(I_7,[) = I_3

Análise Sintática SLR(1)



Análise Sintática SLR(1)

- Construção da Tabela de Análise SLR:
  - Dada G, obtém-se G', aumentando G com a produção S' → S, onde S é o símbolo inicial de G
  - A partir de G', determina-se o conjunto canônico C
  - Constroem-se então tabelas Ação-Transição

Análise Sintática SLR(1)

- Algoritmo para construir tabela SLR para G:
  - Seja C = {I0, I1, …, In}
    - Os estados do analisador são 0, 1, …, n (0 é o estado inicial)
    - Linha i da tabela é construída a partir do conjunto I_i :
      - Regras para ações para o estado i:
        - Se ir-para(I_i,a) = I_j, então Ação[i,a] = empilha a j
        - Se A → α. está em I_i, então para todo a em FOLLOW(A), Ação[i,a] = reduz n, sendo n número da regra A → α.
        - Se S' → S. está em I_i, então Ação[i,$]= aceita
      - Regra para transições para o estado i:
        - Se ir-para(I_i,A) = I_j, então Transição[i,A] = j

Entradas não definidas correspondem a situações de erro

Se ações conflitantes são geradas pelas regras de ação, a gramática não é SLR(1)

Análise Sintática SLR(1)

- Exemplo:
- 1) S → a 2) S → [L] 3) L → L;S 4) L → S

	AÇÃO					TRANSIÇÃO	
	a	[	]	;	$	S	L
0	e2	e3				1	
1					AC		
2			r1	r1	r1		
3	e2	e3				5	4
4			e6	e7			
5			r4	r4			
6			r2	r2	r2		
7	e2	e3				8	
8			r3	r3			

Análise Sintática Ascendente

- Funcionamento: seja Em o estado no topo da pilha e ai a entrada corrente
  - Analisador consulta Ação[Em, ai], que pode ser:
    - Empilha Ex: empilha ai Ex
    - Reduz n (n = número de produção $A \rightarrow \beta$): desempilha 2r símbolos, em que r = $|\beta|$, e o empilhamento de A Ey, onde Ey resulta em consulta a Transição[Em-r, A]
    - Aceita: analisador reconhece entrada como válida
    - Erro: analisador identifica erro sintático

Análise Sintática Ascendente

- Exemplo:

S → a | [L]

L → L ; S | S

Para a entrada [a ; a], as ações do analisador são:

Pilha	Entrada	Ação
$ 0	[a ; a] $	Empilha e vai para estado 3
$ 0 [3	a ; a] $	Empilha e vai para estado 2
$ 0 [3 a 2	; a] $	Reduz pela regra 1 (S → a)
$ 0 [3 S 5	; a] $	Reduz por regra 4 L → S
$ 0 [3 L 4	; a] $	Empilha e vai para estado 7
$ 0 [3 L 4 ; 7	a] $	Empilha e vai para estado 2
$ 0 [3 L 4 ; 7 a 2	] $	Reduz por regra 1 S → a
$ 0 [3 L 4 ; 7 S 8	] $	Reduz por regra 3 L → L ; S
$ 0 [3 L 4	] $	Empilha e vai para estado 6
$ 0 [3 L 4] 6	$	Reduz por regra 2 S → [L]
$ 0 S 1	$	aceita

Análise Sintática SLR(1)

- Exercícios:
- 1) E → E + n 2) E → n

- Exercício 1: construa o conjunto canônico de itens dessa gramática e desenhe o AFD equivalente

- Exercício 2: construa a tabela de análise sintática SLR dessa gramática

- Exercício 3: use essa tabela para fazer a análise sintática de n+n+n

Análise Sintática SLR(1)

- Ambiguidades:
  - Quando há ambiguidade entre empilhar e reduzir, normalmente prefere empilhar
  - Ambiguidades entre reduções indicam normalmente erro no projeto da gramática

Análise Sintática SLR(1)

- Exemplo:
- 1) S → I
- 2) S → outra
- 3) I → if S
- 4) I → if S else S

	if	else	outra	$	S	I
0	e4		e3		1	2
1				AC		
2		r1		r1		
3		r2		r2		
4	e4		e3		5	2
5		**e6**		r3		
6	e4		e3		7	2
7		r4		r4		

Análise Sintática Ascendente

- Bibliografia consultada

LOUDEN, K. C. **Compiladores: princípios e práticas.** São Paulo: Pioneira Thompson Learning, 2004. (Cap. 5)

PRICE, A. M. A. e TOSCANI, S. S. **Implementação de Linguagens de Programação: Compiladores**. Bookman, 2008 (Cap. 3)