

Construção de compiladores

Prof. Daniel Lucrédio

Departamento de Computação - UFSCar

1º semestre / 2015

Aula 5

Análise sintática ascendente

Ou análise sintática bottom-up

Introdução

- Vimos que existem duas formas de se reconhecer uma linguagem através de uma gramática
 - Inferência recursiva
 - Derivação
- Ex: Gramática para expressões aritméticas
 - $V = \{E, I\}$
 - $T = \{+, *, (,), a, b, 0, 1\}$
 - $P =$ conjunto de regras ao lado
 - $S = E$

$$\begin{array}{lcl} E & \rightarrow & I \\ & | & E + E \\ & | & E * E \\ & | & (E) \\ I & \rightarrow & a \\ & | & b \\ & | & Ia \\ & | & Ib \\ & | & I0 \\ & | & I1 \end{array}$$

Introdução

- Inferência recursiva

- Dada uma cadeia (conjunto de símbolos terminais)
- Do corpo para a cabeça
- Ex: $a^*(a+b00)$
 - $a^*(a+b00) \Leftarrow a^*(a+l00) \Leftarrow a^*(a+l0) \Leftarrow a^*(a+l) \Leftarrow a^*(a+E) \Leftarrow a^*(l+E) \Leftarrow a^*(E+E) \Leftarrow a^*(E) \Leftarrow a^*E \Leftarrow l^*E \Leftarrow E^*E \Leftarrow E$

- Derivação

- Dada uma cadeia (conjunto de símbolos terminais)
- Da cabeça para o corpo
- Ex: $a^*(a+b00)$
 - $E \Rightarrow E^*E \Rightarrow l^*E \Rightarrow a^*E \Rightarrow a^*(E) \Rightarrow a^*(E+E) \Rightarrow a^*(l+E) \Rightarrow a^*(a+E) \Rightarrow a^*(a+l) \Rightarrow a^*(a+l0) \Rightarrow a^*(a+l00) \Rightarrow a^*(a+b00)$

Introdução

- Análise sintática descendente
 - Fazer o processo de derivação
 - Ou
 - Criar a árvore de análise sintática “de cima para baixo”
 - Análise top-down
- Análise sintática ascendente
 - Fazer o processo de inferência
 - Ou
 - Criar a árvore de análise sintática “de baixo para cima”
 - Análise bottom-up

Introdução

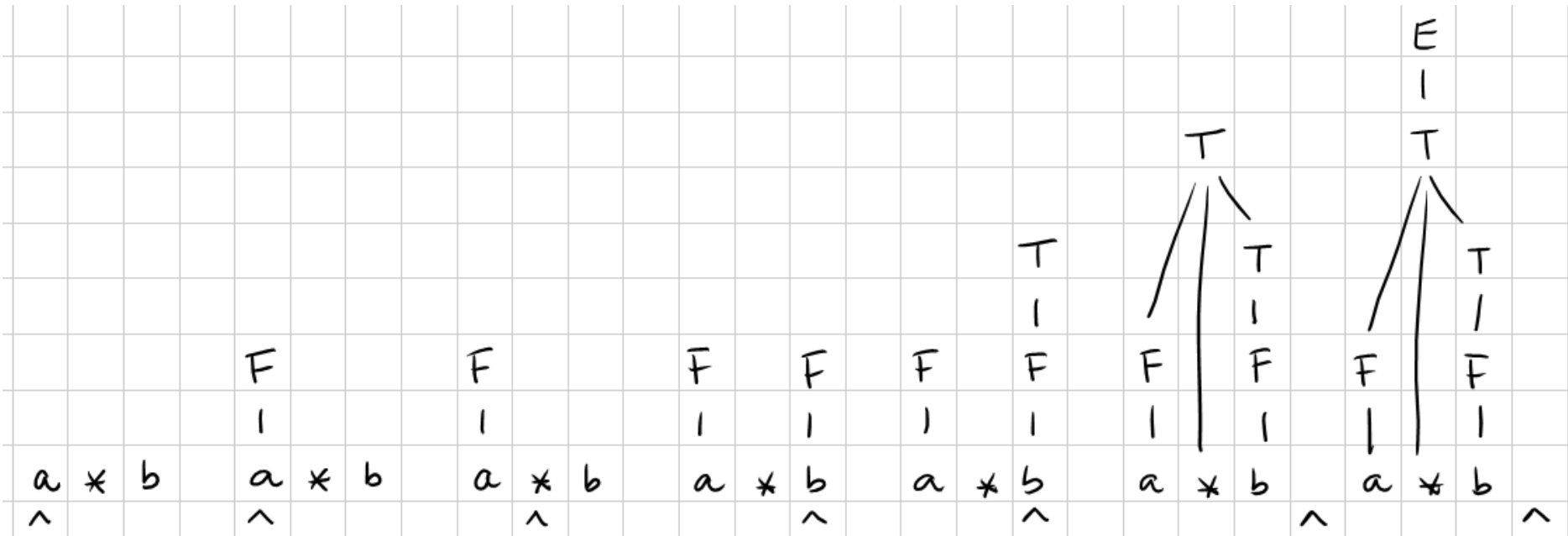
- Exemplo

$E \rightarrow T + E \mid T$

$T \rightarrow F * T \mid F$

$F \rightarrow a \mid b \mid (E)$

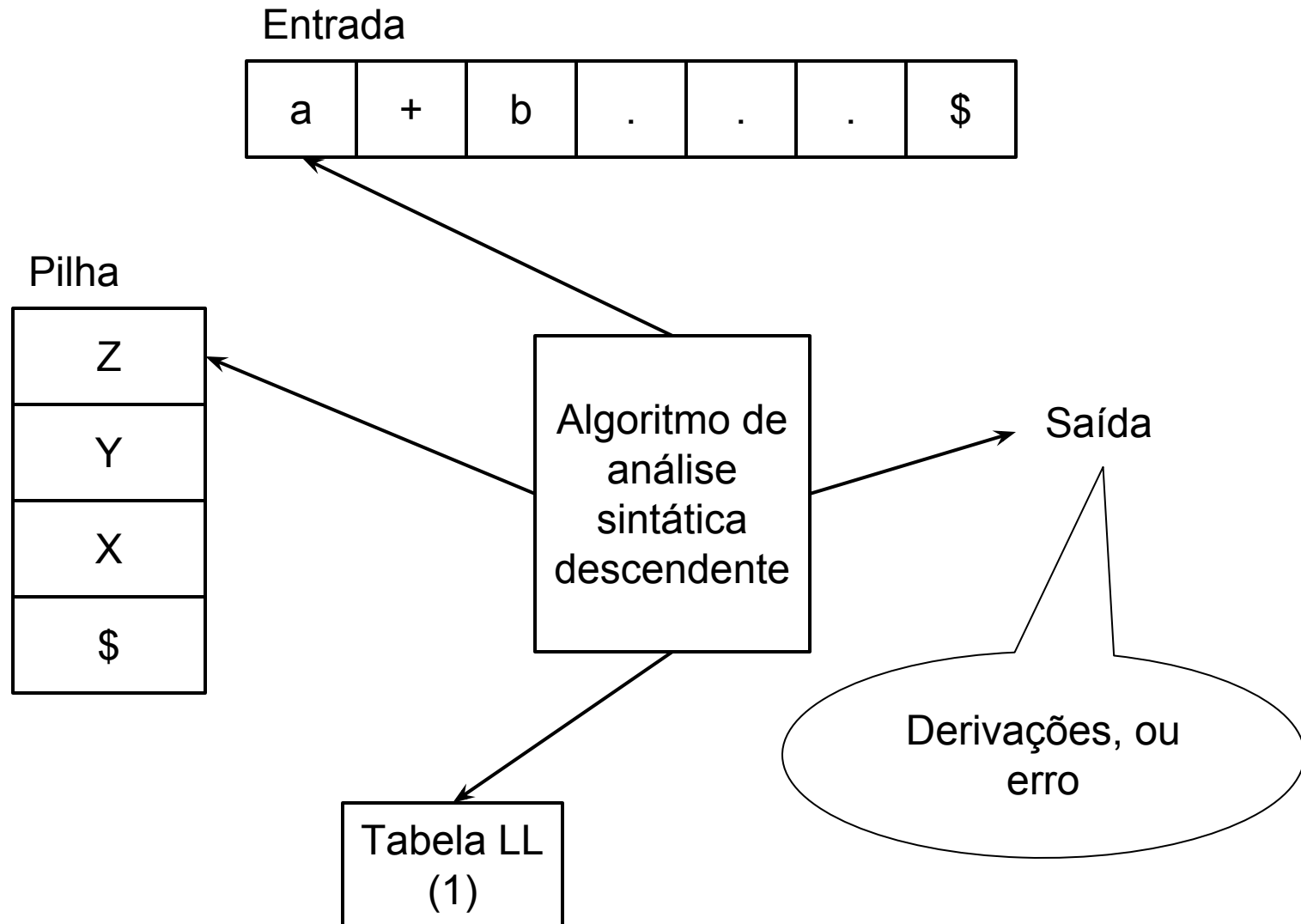
Cadeia = $a * b$



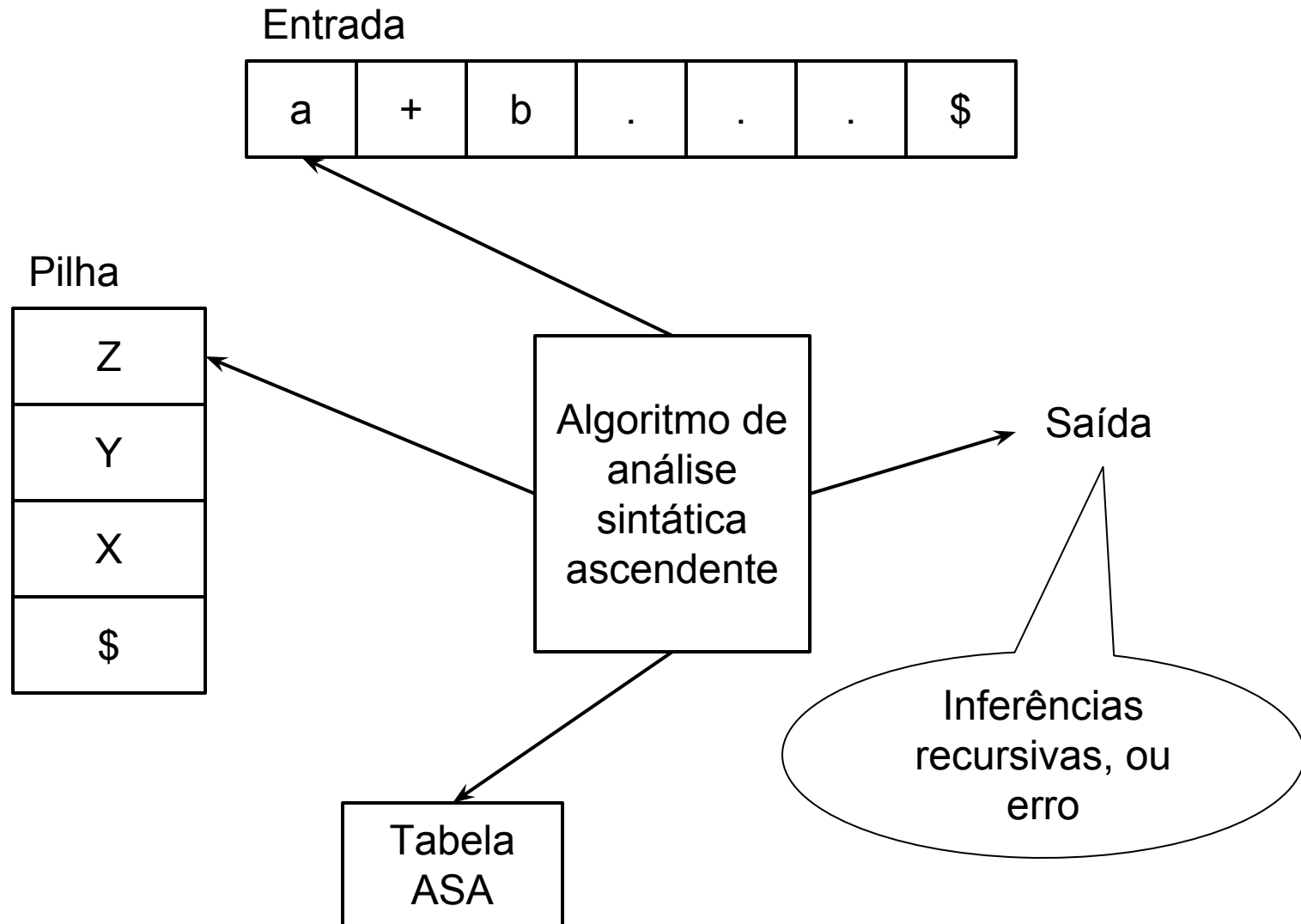
Introdução

- Em LFA
 - Analisador sintático = autômato de pilha
 - Portanto, para ambos os casos, um autômato de pilha deve ser suficiente
- Na análise sintática descendente, já vimos como fazer
 - A pilha armazena os símbolos a serem substituídos
 - Quando a pilha esvaziar, acabou
- Na análise sintática ascendente
 - A pilha vai armazenar os símbolos aguardando “redução”
 - Quando sobrar só o símbolo inicial na pilha, acabou

Introdução



Introdução



Introdução

- Análise sintática descendente = método (algoritmo) que produz uma derivação mais à esquerda para uma cadeia da entrada
- O problema principal em cada passo é determinar qual produção aplicar
- Sendo que os tokens são lidos da esquerda para a direita

- Ex:

- Entrada: $a + b * c$
- Token atual = a
- Símbolo inicial: E
- Possíveis produções de E :
 - $E + E$
 - $E * E$
 - (E)

Qual
escolher?

$$\begin{array}{l} E \rightarrow I \\ | \quad E + E \\ | \quad E * E \\ | \quad (E) \\ I \rightarrow a \mid b \mid c \end{array}$$

Exemplo: análise sintática descendente

- Gramática: $S \rightarrow n + S \mid n$
- Cadeia: $n + n$

Escolha é guiada pela tabela LL

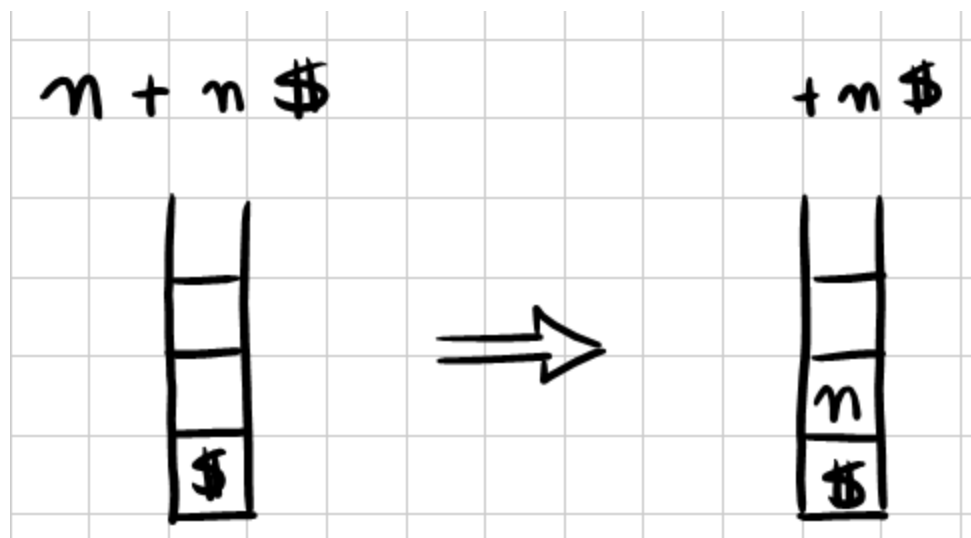
Casamento	Pilha	Entrada	Ação
	<u>S</u> \$	<u>n</u> +n\$	$S \rightarrow n + S$
<u>n</u>	<u>n</u> +S\$	<u>n</u> +n\$	match
n <u>+</u>	<u>+</u> S\$	<u>+</u> n\$	match
	<u>S</u> \$	<u>n</u> \$	$S \rightarrow n$
n+ <u>n</u>	<u>n</u> \$	<u>n</u> \$	match
	<u>\$</u>	<u>\$</u>	aceita

Introdução

- Na análise sintática ascendente, temos um processo diferente
- Para reconhecer uma cadeia de entrada:
 - **Empilha**
 - Os símbolos da cadeia de entrada
 - **Reduz**
 - O lado direito de uma produção no topo da pilha, substituindo-o pelo lado esquerdo da produção
- Os passos 1 e 2 são repetidos até que
 - ACEITA – os símbolos da cadeia de entrada foram consumidos e a pilha possui apenas o símbolo inicial da gramática
 - OU
 - ERRO – o processo foi interrompido antes de chegar ao final

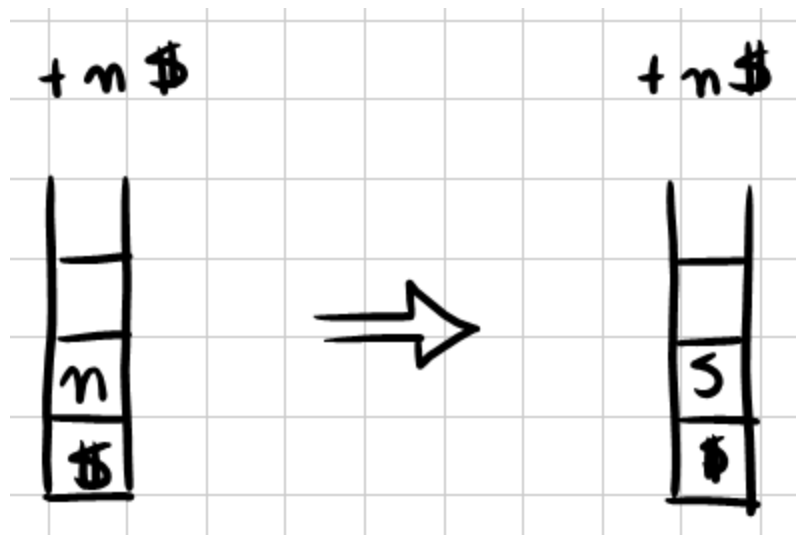
Análise sintática ascendente

- Empilhamento
 - Consiste em remover um símbolo da entrada e adicioná-lo ao topo da pilha
- Ex:
 - Gramática = $S \rightarrow S + n \mid n$
 - Entrada = $n + n$



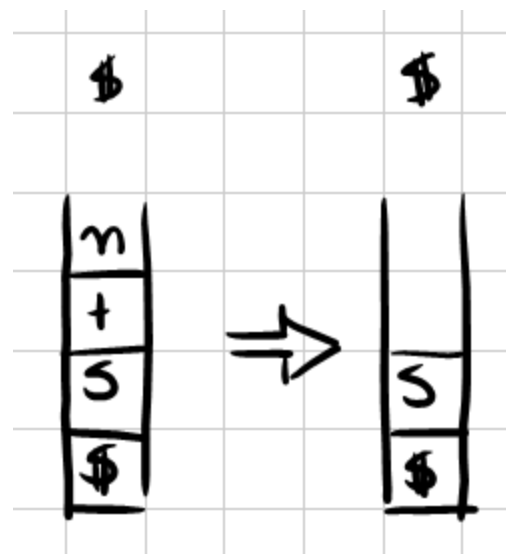
Análise sintática ascendente

- Redução
 - Consiste em substituir símbolos no topo da pilha por um único símbolo
 - Não consome a entrada
- Ex:
 - Gramática = $S \rightarrow S + n \mid n$
 - Entrada = $n + n$



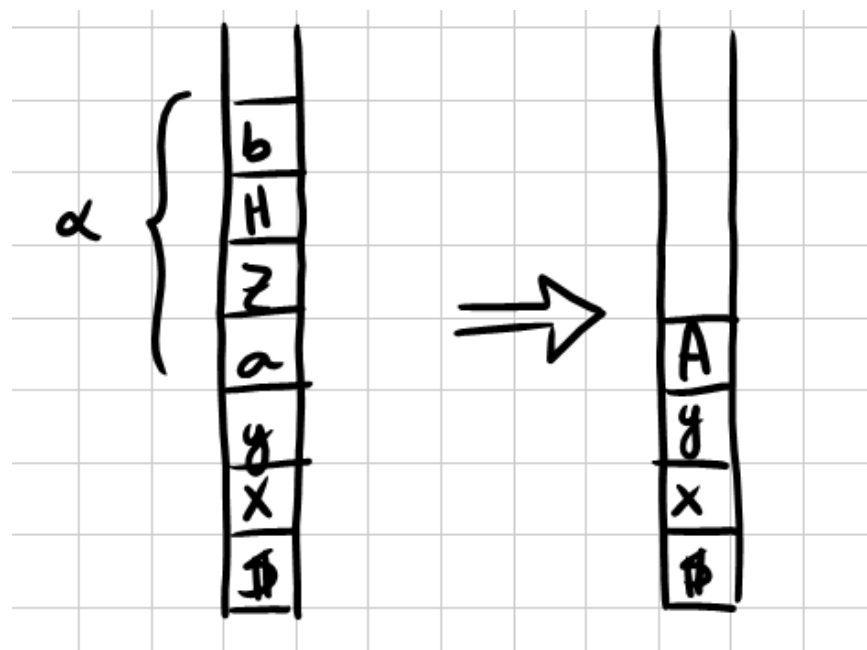
Análise sintática ascendente

- Redução
 - Consiste em substituir símbolos no topo da pilha por um único símbolo
 - Não consome a entrada
- Ex:
 - Gramática = $S \rightarrow S + n \mid n$
 - Entrada = $n + n$



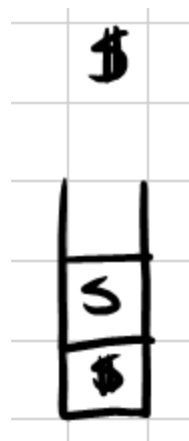
Análise sintática ascendente

- Quando empilhar/reduzir?
- Conceito de “gancho” (handle)
 - Para cada produção $A \rightarrow \alpha$
 - α é um “gancho”
 - Quando α aparecer no topo da pilha, posso substituir por A
 - Ex: $A \rightarrow aZHb$
 - $\alpha = aZHb$



Análise sintática ascendente

- Aceita
 - Quando consumir toda a entrada
 - Pilha contém somente o símbolo inicial
- Ex:
 - Gramática = $S \rightarrow S + n \mid n$
 - Entrada = $n + n$



Análise sintática ascendente

- Exemplo
 - Gramática: $S \rightarrow S + n \mid n$
 - Entrada: $n + n$

Agora escrevemos a pilha da esquerda para a direita, para facilitar

Apareceu um “gancho” aqui

E aqui também

Pilha	Entrada	Ação
\$	<u>n</u> +n\$	empilha n
\$ <u>n</u>	+ <u>n</u> \$	reduz $S \rightarrow n$
\$S	+ <u>n</u> \$	empilha +
\$S+	<u>n</u> \$	empilha n
\$ <u>S+n</u>	<u>\$</u>	reduz $S \rightarrow S+n$
\$S	<u>\$</u>	aceita

Análise sintática ascendente

- Desafio
 - Detectar o aparecimento do “gancho” na pilha
 - Exige olhar um ou mais símbolos da pilha
 - E também olhar símbolos à frente na entrada
 - Normalmente, busca-se olhar somente um símbolo à frente, por uma questão de eficiência

Análise sintática ascendente

- Analisadores sintáticos ascendentes (ASA) – 2 tipos
 - Analisador de precedência de operadores
 - Opera sobre a classe das gramáticas de operadores
 - Guiado por uma tabela de precedência
- Analisador LR (k)
 - Left to right with Rightmost derivation
 - Lê a sentença em análise da esquerda para a direita
 - Produz uma derivação mais à direita ao reverso
 - Inferência recursiva
 - Considerando-se k símbolos na cadeia de entrada
- Diferença está na forma com que detectam o aparecimento do “gancho”

Análise sintática ascendente

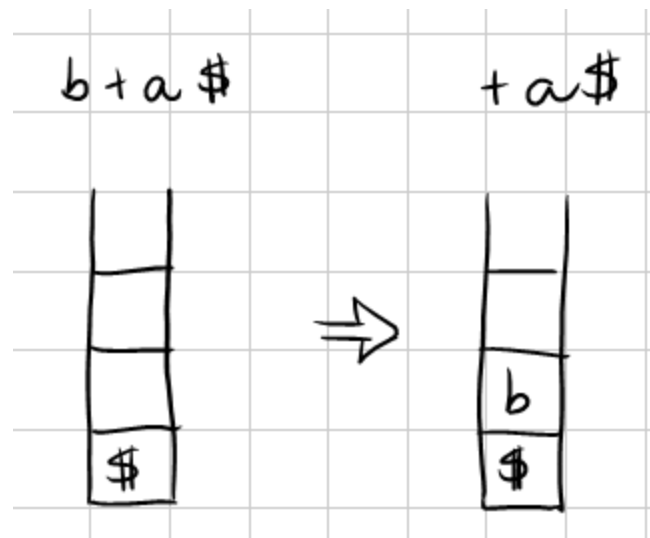
- Primeiro veremos a versão mais “simples”
 - Não é poderosa o suficiente, mas serve como uma boa introdução à técnica que veremos a seguir
- Analisador de precedência de operadores
 - Opera sobre a classe das gramáticas de operadores
 - Gramática de operadores
 - Não há símbolos não-terminais adjacentes nas regras (ou seja, não-terminais são sempre separados por terminais)
 - Não há produções que derivam a cadeia vazia
- Exemplo: $E \rightarrow E + E \mid E * E \mid E ** E \mid (E) \mid id$
- Guiado por uma tabela de precedência
 - Gancho – identificado com base nas relações de precedência entre terminais
 - $<$ - Um terminal tem menor precedência que outro
 - $=$ - Dois terminais tem a mesma precedência
 - $>$ - Um terminal tem maior precedência que outro

Análise sintática ascendente

- Importante:
 - Precedência indica relação entre operadores diferentes
 - Exemplo:
 - $* > +$ ($*$ tem maior precedência do que $+$)
- Mas a precedência também engloba o conceito de associatividade
 - Que podemos enxergar como a precedência entre duas ocorrências de um mesmo operador
 - Exemplo:
 - $+ > +$ ($+$ tem maior precedência do que $+$)
 - É equivalente a dizer que $+$ é associativo à esquerda
 - Exemplo 2:
 - $** < **$ ($**$ tem menor precedência do que $**$)
 - É equivalente a dizer que $**$ é associativo à direita

ASA de precedência de operadores

- Como é feita?
- Seja **p** o terminal mais ao topo da pilha (os não-terminals são ignorados) e **c** o primeiro terminal da cadeia sendo analisada
 - Se $p < c$ ou $p = c$, então empilha c

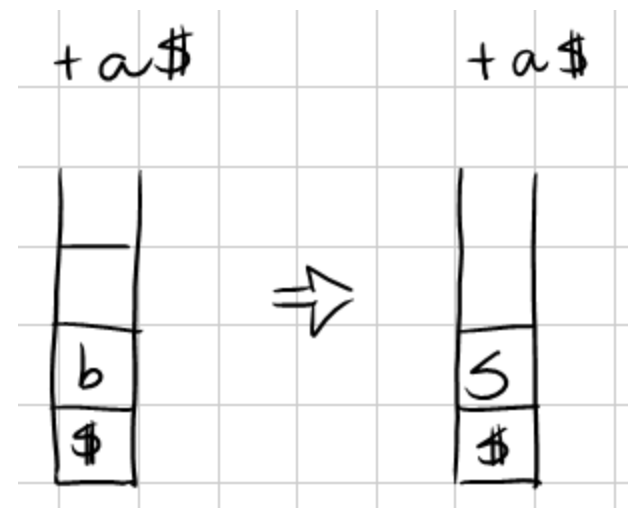


Supondo que $\$ < b$ ou $\$ = c$

ASA de precedência de operadores

- Como é feita?
- Seja **p** o terminal mais ao topo da pilha (os não-terminals são ignorados) e **c** o primeiro terminal da cadeia sendo analisada
 - Se $p > c$, então precisamos:
 1. Encontrar o “gancho” na pilha
 2. Fazer uma redução

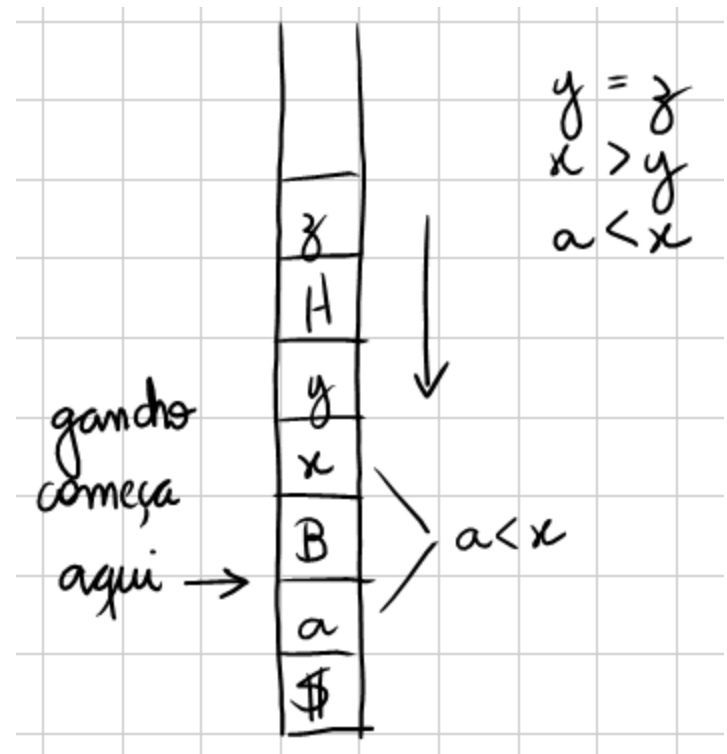
Supondo que b
 $> +, \$ < b$ e S
 $\rightarrow b$



ASA de precedência de operadores

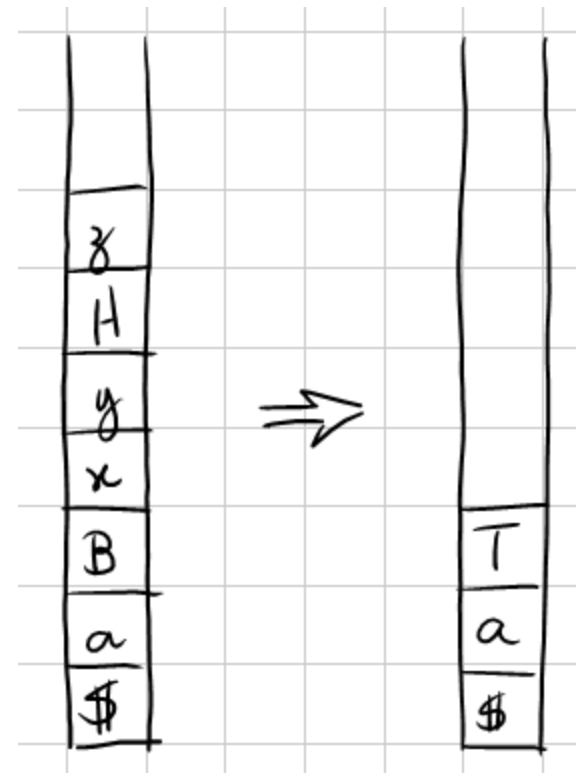
- Como encontrar o “gancho”?
 - Vou olhando para baixo na pilha, desempilhando
 - até encontrar dois terminais em sequência (ignorando não-terminais) tal que:
 - o segundo (de baixo) tem menor precedência que o primeiro (de cima)

Neste exemplo,
o “gancho” é
BxyHz



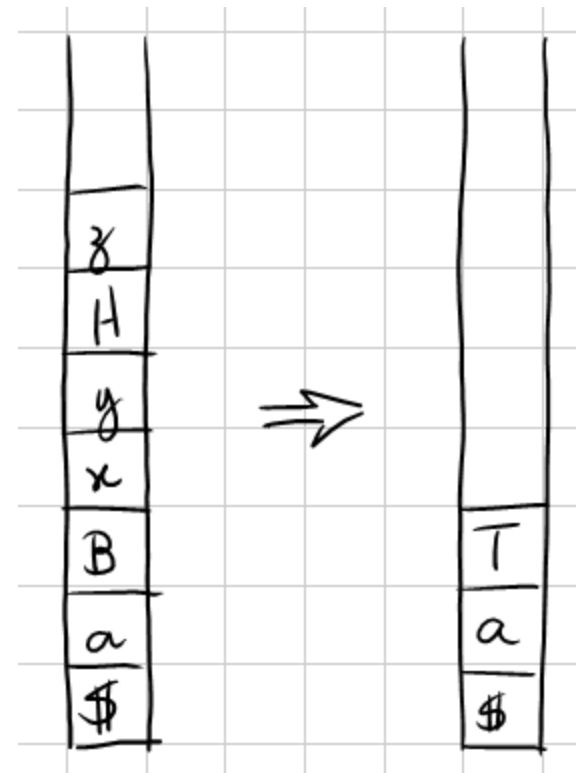
ASA de precedência de operadores

- Uma vez encontrado o “gancho”, é necessário fazer uma redução
 - Basta procurar uma regra cujo lado direito produz o “gancho” encontrado
- Ex: se o “gancho” = $BxyHz$
- E existe uma produção
 - $T \rightarrow BxyHz$
- O resultado da redução ficaria:



ASA de precedência de operadores

- Importante, a redução pode envolver mais de um passo
 - Pode ser que o “gancho” não apareça explicitamente em uma regra
- Ex: se o “gancho” = $BxyHz$
- E existe uma produção
 - $T \rightarrow AxyJz$
 - $A \rightarrow B$
 - $J \rightarrow H$
- O resultado da redução também ficaria:

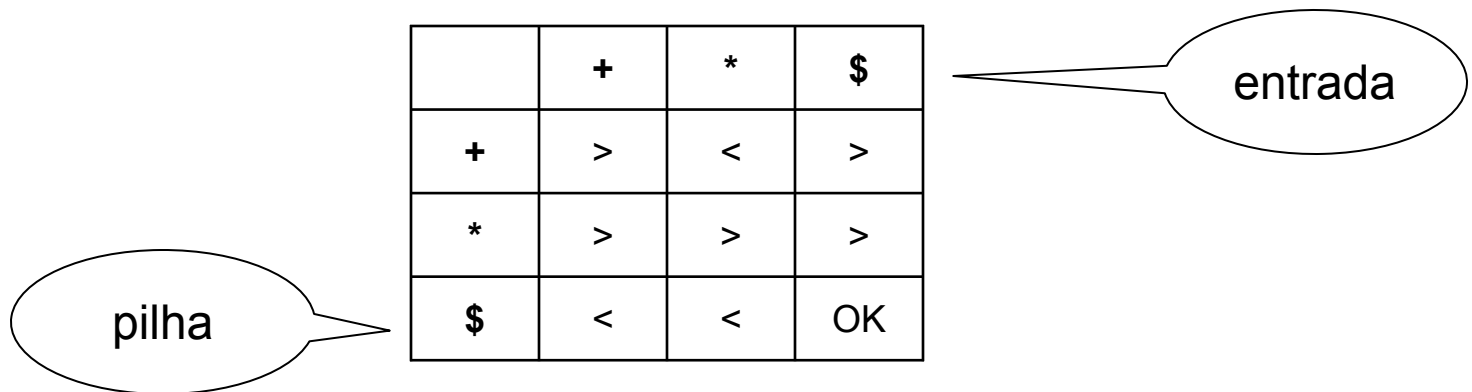


ASA de precedência de operadores

- Erros sintáticos são detectados:
 - Na comparação da precedência entre operadores
 - Ex:) e (não podem aparecer em sequência, ou com um não-terminal entre eles, portanto não existe precedência definida
 - Na tentativa de redução
 - Ex: $E + - E$ não aparece em nenhum corpo de regra (direta ou indiretamente)

Tabela de precedência

- É uma tabela que mapeia todas as precedências possíveis entre os terminais e o delimitador de fim de cadeia “\$”
 - A primeira coluna marca símbolos na pilha
 - A primeira linha marca símbolos na entrada
 - As células marcam as precedências entre um símbolo na pilha e um símbolo na entrada (nessa ordem)



	+	*	\$
+	>	<	>
*	>	>	>
\$	<	<	OK

Tabela de precedência

- Construção da tabela de precedência – 2 métodos
 - Intuitivo
 - Baseado no conhecimento da precedência e associatividade dos operadores
 - Mecânico
 - Obtém-se a tabela diretamente da gramática
 - As gramáticas NÃO podem ser ambíguas
 - As produções devem refletir a associatividade e a precedência dos operadores

Tabela de precedência – método intuitivo

- $E \rightarrow E + E \mid E * E \mid E ** E \mid (E) \mid id$

****** tem maior precedência e é associativo à direita

***** tem precedência intermediária e é associativo à esquerda

+ tem menor precedência e é associativo à esquerda

	+	*	**	()	id	\$
+							
*							
**							
(
)							
id							
\$							

Tabela de precedência – método intuitivo

- $E \rightarrow E + E \mid E * E \mid E ** E \mid (E) \mid id$

Se x tem maior precedência do que y, então tem-se que
- x (na pilha) > y (na entrada)

Ex: * > +

 ** > +

 ** > *

	+	*	**	()	id	\$
+							
*	>						
**	>	>					
(
)							
id							
\$							

Tabela de precedência – método intuitivo

- $E \rightarrow E + E \mid E * E \mid E ** E \mid (E) \mid id$

Se x tem maior precedência do que y, então tem-se que

- x (na pilha) > y (na entrada) e y (na pilha) < x (na entrada)

Ex: * > +

+ < *

** > +

+ < **

** > *

* < **

	+	*	**	()	id	\$
+		<	<				
*	>		<				
**	>	>					
(
)							
id							
\$							

Tabela de precedência – método intuitivo

- $E \rightarrow E + E \mid E * E \mid E ** E \mid (E) \mid id$

Se x e y têm precedência igual (ou são iguais) e

- Se são associativos à esquerda, então $x > y$ e $y > x$

Ex: $* > *$ e $+ > +$

- Se são associativos à direita, então $x < y$ e $y < x$

Ex: $** < **$

	+	*	**	()	id	\$
+	>	<	<				
*	>	>	<				
**	>	>	<				
(
)							
id							
\$							

Tabela de precedência – método intuitivo

- $E \rightarrow E + E \mid E * E \mid E ** E \mid (E) \mid id$

As relações entre operadores e demais símbolos terminais (operandos e delimitadores) são fixas. Para qq operador z

$z < (\quad \quad z >) \quad \quad z < id \quad \quad z > \$$
 $(< z \quad \quad) > z \quad \quad id > z \quad \quad \$ < z$

	+	*	**	()	id	\$
+	>	<	<	<	>	<	>
*	>	>	<	<	>	<	>
**	>	>	<	<	>	<	>
(<	<	<				
)	>	>	>				
id	>	>	>				
\$	<	<	<				

Tabela de precedência – método intuitivo

- $E \rightarrow E + E \mid E * E \mid E ** E \mid (E) \mid id$

As relações entre os operandos também são fixas

$(< (\quad) >) \quad id >) \quad \$ < (\quad (=)$

$id > \$ \quad \$ < id \quad (< id \quad) > \$$

	+	*	**	()	id	\$
+	>	<	<	<	>	<	>
*	>	>	<	<	>	<	>
**	>	>	<	<	>	<	>
(<	<	<	<	=	<	
)	>	>	>		>		>
id	>	>	>		>		>
\$	<	<	<	<		<	

Tabela de precedência – método intuitivo

- $E \rightarrow E + E \mid E * E \mid E ** E \mid (E) \mid id$

A relação entre \$ e \$ indica aceitação da cadeia de entrada

	+	*	**	()	id	\$
+	>	<	<	<	>	<	>
*	>	>	<	<	>	<	>
**	>	>	<	<	>	<	>
(<	<	<	<	=	<	
)	>	>	>		>		>
id	>	>	>		>		>
\$	<	<	<	<		<	OK

Tabela de precedência – método mecânico

- Exemplo: $E \rightarrow E + E \mid E * E \mid E ** E \mid (E) \mid id$
- Primeiro passo: remover a ambiguidade

- $E \rightarrow E + T \mid T$
- $T \rightarrow T * F \mid F$
- $F \rightarrow P ** F \mid P$
- $P \rightarrow id \mid (E)$

Tabela de precedência – método mecânico

- Agora basta seguir às seguintes regras, para dois terminais a e b , e o símbolo de fim de cadeia $\$$:

1. $a = b$, se:
 - $\alpha a \beta b \bar{\delta}$ é lado direito de uma produção
 - β é ϵ ou não-terminal
2. $a < b$, se:
 - $\alpha a X \beta$ é lado direito de uma produção
 - $X \Rightarrow^* \gamma b \bar{\delta}$
 - γ é ϵ ou não-terminal
3. $\$ < b$, se:
 - $S \Rightarrow^* \gamma b \bar{\delta}$ (S é o símbolo inicial)
 - γ é ϵ ou não-terminal
4. $a > b$, se:
 - $\alpha X b \beta$ é lado direito de uma produção
 - $X \Rightarrow^* \gamma a \bar{\delta}$
 - $\bar{\delta}$ é ϵ ou não-terminal
5. $a > \$$, se:
 - $S \Rightarrow^* \gamma a \bar{\delta}$ (S é o símbolo inicial)
 - $\bar{\delta}$ é ϵ ou não-terminal

Tabela de precedência – método mecânico

- Destrinchando as regras:

1. $a = b$, se:

- $\alpha a \beta b \delta$ é lado direito de uma produção
- β é ϵ ou não-terminal
- Basta procurar dois terminais que aparecem em algum lado direito
 - Com um não-terminal ou nada entre eles
 - Exs: (E), + T -, ^ #
- Neste caso, a precedência destes terminais é igual
 - Exs: (=), +=-, ^=#

Tabela de precedência – método mecânico

- Destrinchando as regras:

2. $a < b$, se:

- $\alpha a X \beta$ é lado direito de uma produção
- $X \Rightarrow^* \gamma b \delta$
- γ é ϵ ou não-terminal
- Basta procurar dois terminais que aparecem em regras diferentes
 - E a primeira regra contém uma chamada para a segunda regra, DEPOIS do primeiro terminal
- Ex:
 - $E \rightarrow E + T$
 - $T \rightarrow T * F$
- Ou seja, depois do +, tem uma chamada pra regra T, onde aparece *
 - Portanto $+ < *$

Tabela de precedência – método mecânico

- Destrinchando as regras:

2. $a < b$, se:

- $\alpha a X \beta$ é lado direito de uma produção
- $X \Rightarrow^* \gamma b \delta$
- γ é ϵ ou não-terminal
- É importante observar que a chamada não é necessariamente direta!
- Ex:
 - $E \rightarrow E + T$
 - $T \rightarrow F$
 - $F \rightarrow F ** P$
- Ou seja, depois do +, tem uma chamada (indireta) pra regra F, onde aparece **
 - Portanto $+ < **$

Tabela de precedência – método mecânico

- Destrinchando as regras:

2. $a < b$, se:

- $\alpha a X \beta$ é lado direito de uma produção
- $X \Rightarrow^* \gamma b \delta$
- γ é ϵ ou não-terminal
- O segundo terminal (b), deve aparecer na regra sendo chamada de um jeito especial:
 - Do seu lado esquerdo, só pode ter um não-terminal ou nada!!
- Ex:
 - $E \rightarrow E + T$
 - $T \rightarrow (T * F)$
- Nesse caso, $+ < ($, pois não tem nada à sua esquerda na regra sendo chamada!
- Mas não posso dizer que $+ < *$, pois aparece pelo menos um terminal à sua esquerda!

Tabela de precedência – método mecânico

- Destrinchando as regras:

2. $a < b$, se:

- $\alpha a X \beta$ é lado direito de uma produção
- $X \Rightarrow^* \gamma b \delta$
- γ é ϵ ou não-terminal
- Finalmente, devemos considerar chamadas recursivas à direita:
 - Isso irá definir a associatividade à direita de um terminal
- Ex:
 - $E \rightarrow T + E \mid E$
- Nesse caso, E faz o papel do X na regra acima
 - Portanto $+ < +$ ($+$ é associativo à direita)

Tabela de precedência – método mecânico

- Destrinchando as regras:

3. $\$ < b$, se:

- $S \Rightarrow^* \gamma b \delta$ (S é o símbolo inicial)
- γ é ϵ ou não-terminal
- Basta procurar um terminal que aparece num corpo de regra que é derivável a partir do símbolo inicial (direta ou indiretamente)
 - E à esquerda desse terminal, tem um não-terminal ou nada
- Exs (assumindo E como símbolo inicial):
 - $E \rightarrow E + T \mid T$
 - $T \rightarrow T * F$
- Concluimos que $\$ < +$ e $\$ < *$

Tabela de precedência – método mecânico

- Destrinchando as regras:

4. $a > b$, se:

- $\alpha X b \beta$ é lado direito de uma produção
- $X \Rightarrow^* \gamma a \delta$
- δ é ϵ ou não-terminal
- Similar à regra 2, porém “invertendo” os terminais
 - “b” aparece na primeira regra, e “a” na segunda regra
- Basta procurar dois terminais que aparecem em regras diferentes
 - E a primeira regra contém uma chamada para a segunda regra, ANTES do segundo terminal (b)
- Ex:
 - $E \rightarrow T + E$
 - $T \rightarrow T * F$
- Ou seja, antes do + (terminal b), tem uma chamada pra regra T, onde aparece * (terminal a)
 - Portanto $* > +$

Tabela de precedência – método mecânico

- Destrinchando as regras:

4. $a > b$, se:

- $\alpha X b \beta$ é lado direito de uma produção
- $X \Rightarrow^* \gamma a \delta$
- δ é ϵ ou não-terminal
- É importante observar que a chamada não é necessariamente direta!
- Ex:
 - $E \rightarrow E + T \mid T$
 - $T \rightarrow T * F \mid F$
 - $F \rightarrow P ** F$
- Ou seja, antes do +, tem uma chamada (indireta) pra regra T, onde aparece *, portanto $* > +$
- Antes do +, também tem uma chamada (indireta) para a regra F, onde aparece **, portanto $** > +$
- Da mesma forma, identificamos que $** > *$

Tabela de precedência – método mecânico

- Destrinchando as regras:

4. $a > b$, se:

- $\alpha X b \beta$ é lado direito de uma produção
- $X \Rightarrow^* \gamma a \delta$
- δ é ϵ ou não-terminal
- O primeiro terminal (a), deve aparecer na regra sendo chamada de um jeito especial:
 - Do seu lado direito, só pode ter um não-terminal ou nada!!
 - Obs: na regra 2, era do lado esquerdo!!!
- Ex:
 - $E \rightarrow T + E$
 - $T \rightarrow (T * F)$
- Nesse caso, $) > +$, pois não tem nada à sua direita na regra sendo chamada!
- Mas não posso dizer que $* > +$, pois aparece pelo menos um terminal à sua direita!

Tabela de precedência – método mecânico

- Destrinchando as regras:

4. $a > b$, se:

- $\alpha X b \beta$ é lado direito de uma produção
- $X \Rightarrow^* \gamma a \delta$
- δ é ϵ ou não-terminal
- Finalmente, devemos considerar chamadas recursivas à esquerda:
 - Isso irá definir a associatividade à esquerda de um terminal
- Ex:
 - $E \rightarrow E + T \mid T$
- Nesse caso, E faz o papel do X na regra acima
 - Portanto $+ > +$ ($+$ é associativo à esquerda)

Tabela de precedência – método mecânico

- Destrinchando as regras:

5. $a > \$$, se:

- $S \Rightarrow^* \gamma a \delta$ (S é o símbolo inicial)
- δ é ϵ ou não-terminal
- Similar à regra 3, porém “invertida”
- Basta procurar um terminal que aparece num corpo de regra que é derivável a partir do símbolo inicial (direta ou indiretamente)
 - E à direita desse terminal, tem um não-terminal ou nada
 - Na regra 3 era à esquerda que precisávamos analisar
- Exs (assumindo E como símbolo inicial):
 - $E \rightarrow E + T \mid T$
 - $T \rightarrow T * F \mid (E)$
- Concluimos que $+ > \$$, $* > \$$ e $) > \$$

Tabela de precedência – método mecânico

- Exemplo:

$E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow P**F \mid P$

$P \rightarrow id \mid (E)$

	+	*	**	()	id	\$
+							
*							
**							
(
)							
id							
\$							

Tabela de precedência – método mecânico

- Exemplo:

$E \rightarrow E+T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow P ** F \mid P$

$P \rightarrow id \mid (E)$



Pela regra 1

	+	*	**	()	id	\$
+							
*							
**							
(=		
)							
id							
\$							

Tabela de precedência – método mecânico

- Exemplo:

$E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow P**F \mid P$

$P \rightarrow id \mid (E)$



Pela regra 2

	+	*	**	()	id	\$
+		<	<	<		<	
*			<	<		<	
**			<	<		<	
(<	<	<	<	=	<	
)							
id							
\$							

Tabela de precedência – método mecânico

- Exemplo:

$E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow P**F \mid P$

$P \rightarrow id \mid (E)$



Pela regra 3

	+	*	**	()	id	\$
+		<	<	<		<	
*			<	<		<	
**			<	<		<	
(<	<	<	<	=	<	
)							
id							
\$	<	<	<	<		<	

Tabela de precedência – método mecânico

- Exemplo:

$E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow P**F \mid P$

$P \rightarrow id \mid (E)$



Pela regra 4

	+	*	**	()	id	\$
+	>	<	<	<	>	<	
*	>	>	<	<	>	<	
**	>	>	<	<	>	<	
(<	<	<	<	=	<	
)	>	>	>		>		
id	>	>	>		>		
\$	<	<	<	<		<	

Tabela de precedência – método mecânico

- Exemplo:

$E \rightarrow E+T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow P ** F \mid P$

$P \rightarrow id \mid (E)$



Pela regra 5

	+	*	**	()	id	\$
+	>	<	<	<	>	<	>
*	>	>	<	<	>	<	>
**	>	>	<	<	>	<	>
(<	<	<	<	=	<	
)	>	>	>		>		>
id	>	>	>		>		>
\$	<	<	<	<		<	

Tabela de precedência – método mecânico

- Exemplo:

$E \rightarrow E+T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow P ** F \mid P$

$P \rightarrow id \mid (E)$



Finalmente

	+	*	**	()	id	\$
+	>	<	<	<	>	<	>
*	>	>	<	<	>	<	>
**	>	>	<	<	>	<	>
(<	<	<	<	=	<	
)	>	>	>		>		>
id	>	>	>		>		>
\$	<	<	<	<		<	OK

Tabela de precedência

- Exercício – Construa a tabela de precedência de operadores para a seguinte gramática

- $E \rightarrow E \text{ 'or' } T \mid T$
- $T \rightarrow T \text{ 'and' } F \mid F$
- $F \rightarrow \text{'(' } E \text{ ')'} \mid \text{var}$

	var	'or'	'and'	'('	')'	\$
var						
'or'						
'and'						
'('						
')'						
\$						

Tabela de precedência

- Resposta

	var	'or'	'and'	' ('	')	\$
var		>	>		>	>
'or'	<	>	<	<	>	>
'and'	<	>	>	<	>	>
' ('	<	<	<	<	=	
')		>	>		>	>
\$	<	<	<	<		ok

Tabela de precedência

- Exercício – Construa a tabela de precedência de operadores para a seguinte gramática (elimine a ambiguidade primeiro)

$\text{expr} \rightarrow \text{expr op expr} \mid \text{NUM} \mid (\text{expr})$

$\text{op} \rightarrow + \mid - \mid * \mid / \mid \% \mid ^$

- Precedência:

$+ , - < * , / , \% < ^$

- Associatividade

- Todos são associativos à esquerda, exceto o operador de potência (^)

Tabela de precedência

- **Versão não-ambígua da gramática**

$\text{expr} \rightarrow \text{expr op1 termo} \mid \text{termo}$

$\text{termo} \rightarrow \text{termo op2 fator} \mid \text{fator}$

$\text{fator} \rightarrow \text{elem op3 fator} \mid \text{elem}$

$\text{elem} \rightarrow \text{NUM} \mid (\text{expr})$

$\text{op1} \rightarrow + \mid -$

$\text{op2} \rightarrow * \mid / \mid \%$

$\text{op3} \rightarrow ^$

Tabela de precedência

[illegible]

- Resposta

[illegible]

Tabela de precedência

- Exercício – Construa a tabela de precedência de operadores para a seguinte gramática
 - $E \rightarrow E + E \mid E * E \mid E ? E : E \mid \text{var} \mid \text{NUM}$
- Precedência:
 $? : < + < *$
- Associatividade
 - Todos são associativos à esquerda

Tabela de precedência

- Versão não-ambígua da gramática

- $E \rightarrow E \mid E : T \mid T$
- $T \rightarrow T + F \mid F$
- $F \rightarrow F * P \mid P$
- $P \rightarrow \text{var} \mid \text{NUM}$

	?	:	+	*	var	NUM	\$
?							
:							
+							
*							
var							
NUM							
\$							

Tabela de precedência

- Resposta

	?	:	+	*	var	NUM	\$
?	<	=	<	<	<	<	
:	>	>	<	<	<	<	>
+	>	>	>	<	<	<	>
*	>	>	>	>	<	<	>
var	>	>	>	>			>
NUM	>	>	>	>			>
\$	<		<	<	<	<	ok

Algoritmo da ASA de precedência de operadores

```
1.  (* Seja S o símbolo inicial da gramática, p o símbolo terminal mais
2.  ao topo da pilha e c o primeiro símbolo da cadeia de entrada *)
3.  do
4.    if ($S é o topo da pilha and $ é o primeiro símbolo da cadeia)
5.      then ACEITA
6.    else if (p < c or p = c) then
7.      empilha c;                (* enquanto a precedência for < ou =, empilha *)
8.      avance na leitura da entrada;
9.    else if (p > c) then        (* precedência > significa fim do handle *)
10.     desempilha até encontrar a relação < entre o terminal do topo da pilha
11.     e o último terminal desempilhado; (* não-terminais são ignorados
12.                                     *)
13.     empilha o não-terminal correspondente;                (* reduz *)
14.  until ACEITA or ERRO;
```

ASA de precedência de operadores

- Exemplo:

$E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow P**F \mid P$

$P \rightarrow id \mid (E)$

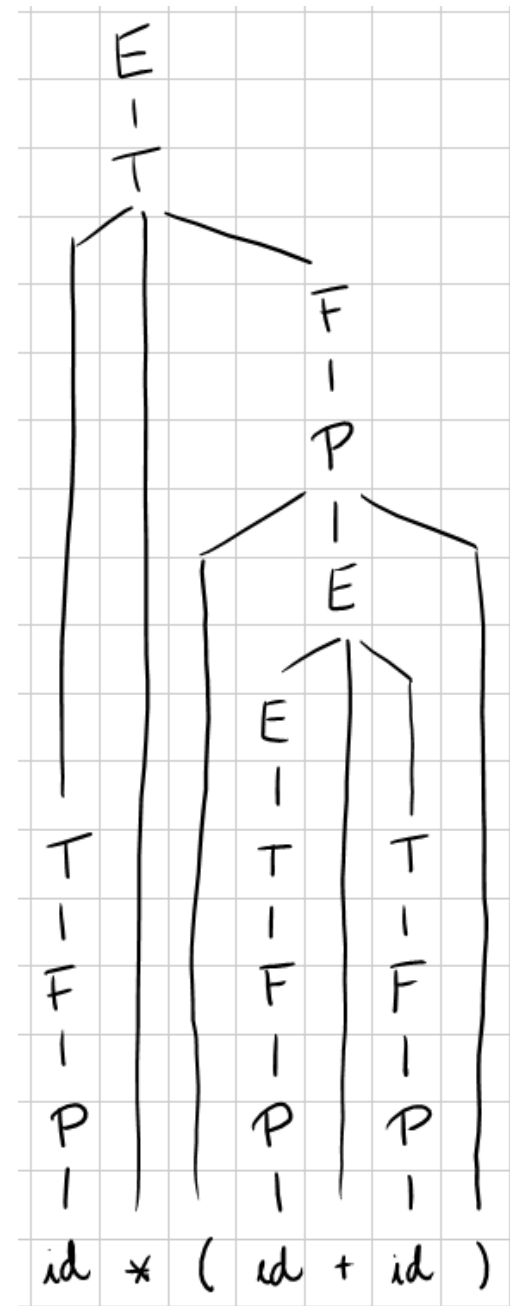
Reconhecer cadeia
`id * (id + id)`

	+	*	**	()	id	\$
+	>	<	<	<	>	<	>
*	>	>	<	<	>	<	>
**	>	>	<	<	>	<	>
(<	<	<	<	=	<	
)	>	>	>		>		>
id	>	>	>		>		>
\$	<	<	<	<		<	OK

ASA de precedência de operadores

- Resposta:

Pilha	Entrada	Ação
\$	id*(id+id)\$	empilha id
\$id	*(id+id)\$	reduz, gancho=id
\$P	*(id+id)\$	empilha *
\$P*	(id+id)\$	empilha (
\$P*(id+id)\$	empilha id
\$P*(id	+id)\$	reduz, gancho=id
\$P*(P	+id)\$	empilha +
\$P*(P+	id)\$	empilha id
\$P*(P+id)\$	reduz, gancho=id
\$P*(P+P)\$	reduz, gancho=P+P
\$P*(E)\$	empilha)
\$P*(E)	\$	reduz, gancho=(E)
\$P*P	\$	reduz, gancho=P*P
\$T	\$	reduz para E
\$E	\$	ACEITA



ASA de precedência de operadores

- Exemplo:

$$E \rightarrow E+T \mid T$$

$$T \rightarrow T*F \mid F$$

$$F \rightarrow P**F \mid P$$

$$P \rightarrow id \mid (E)$$

	+	*	**	()	id	\$
+	>	<	<	<	>	<	>
*	>	>	<	<	>	<	>
**	>	>	<	<	>	<	>
(<	<	<	<	=	<	
)	>	>	>		>		>
id	>	>	>		>		>
\$	<	<	<	<		<	OK

Reconhecer cadeia

id *)id + id)

Pilha	Entrada	Ação
\$	id*)id+id)\$	empilha id
\$id	*)id+id)\$	reduz, gancho=id
\$P	*)id+id)\$	empilha *
\$P*)id+id)\$	tenta reduzir, gancho = P* Impossível reduzir! Erro!

ASA de precedência de operadores

- Exemplo:

$$E \rightarrow E+T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow P ** F \mid P$$

$$P \rightarrow id \mid (E)$$

	+	*	**	()	id	\$
+	>	<	<	<	>	<	>
*	>	>	<	<	>	<	>
**	>	>	<	<	>	<	>
(<	<	<	<	=	<	
)	>	>	>		>		>
id	>	>	>		>		>
\$	<	<	<	<		<	OK

Reconhecer cadeia

id * id)

Pilha	Entrada	Ação
\$	id*id)\$	empilha id
\$id	*id)\$	reduz, gancho=id
\$P	*id)\$	empilha *
\$P*	id)\$	empilha id
\$P*id)\$	reduz, gancho=id
\$P*P)\$	reduz, gancho=P*P
\$T)\$	Relação de precedência indefinida na tabela. Erro!

ASA de precedência de operadores

• Exercício:

- $E \rightarrow E \text{ 'or' } T \mid T$
- $T \rightarrow T \text{ 'and' } F \mid F$
- $F \rightarrow \text{'(' } E \text{ ')'} \mid \text{var}$

Reconhecer cadeia

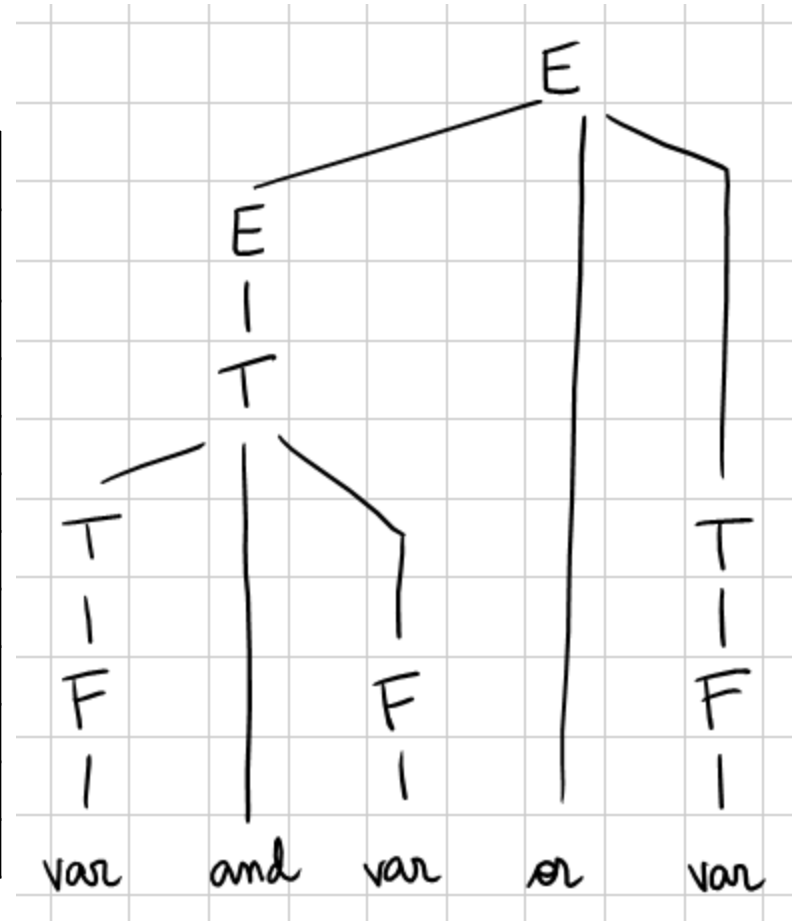
var and var or var

	var	or	and	()	\$
var		>	>		>	>
or	<	>	<	<	>	>
and	<	>	>	<	>	>
(<	<	<	<	=	
)		>	>		>	>
\$	<	<	<	<		ok

ASA de precedência de operadores

- Resposta:

Pilha	Entrada	Ação
\$	var and var or var\$	empilha var
\$var	and var or var\$	reduz, gancho=var
\$F	and var or var\$	empilha and
\$F and	var or var\$	empilha var
\$F and var	or var\$	reduz, gancho=var
\$F and F	or var\$	reduz, gancho=F and F
\$T	or var\$	empilha or
\$T or	var\$	empilha var
\$T or var	\$	reduz, gancho=var
\$T or F	\$	reduz, gancho=T or F
\$E	\$	ACEITA



ASA de precedência de operadores

- Exercício:

$$\text{expr} \rightarrow \text{expr op1 termo} \mid \text{termo}$$
$$\text{termo} \rightarrow \text{termo op2 fator} \mid \text{fator}$$

fator → NUM op3 fator | NUM | (expr)

op1 \rightarrow + | -

op2 → * | / | %

$$\text{op3} \rightarrow \wedge$$

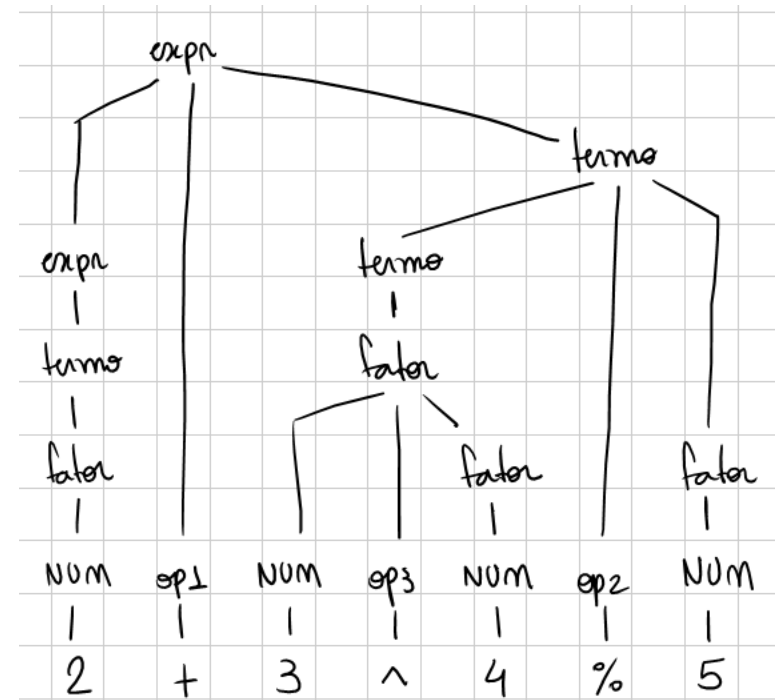
Reconhecer cadeia

$$2 + 3^4 \div 5$$
[illegible]

ASA de precedência de operadores

- Resposta:

Pilha	Entrada	Ação
\$	2+3^4%5\$	empilha 2
\$2	+3^4%5\$	reduz, gancho=2
\$NUM	+3^4%5\$	empilha +
\$NUM +	3^4%5\$	empilha 3
\$NUM + 3	^4%5\$	reduz, gancho=3
\$NUM + NUM	^4%5\$	empilha ^
\$NUM + NUM ^	4%5\$	empilha 4
\$NUM + NUM ^ 4	%5\$	reduz, gancho=4
\$NUM + NUM ^ NUM	%5\$	reduz, gancho=NUM^NUM
\$NUM + fator	%5\$	empilha %
\$NUM + fator %	5\$	empilha 5
\$NUM + fator % 5	\$	reduz, gancho=5
\$NUM + fator % NUM	\$	reduz, gancho=fator%NUM
\$NUM + termo	\$	reduz, gancho=NUM+termo
\$expr	\$	ACEITA



ASA de precedência de operadores

Exercício:

- $E \rightarrow E \ ? \ E \ : \ T \mid T$
- $T \rightarrow T \ + \ F \mid F$
- $F \rightarrow F \ * \ P \mid P$
- $P \rightarrow \text{var} \mid \text{NUM}$

Reconhecer cadeia

a ? 2 + 3 : b ? 3 + 2 : 0

	?	:	+	*	var	NUM	\$
?	<	=	<	<	<	<	
:	>	>	<	<	<	<	>
+	>	>	>	<	<	<	>
*	>	>	>	>	<	<	>
var	>	>	>	>			>
NUM	>	>	>	>			>
\$	<		<	<	<	<	ok

ASA de precedência de operadores

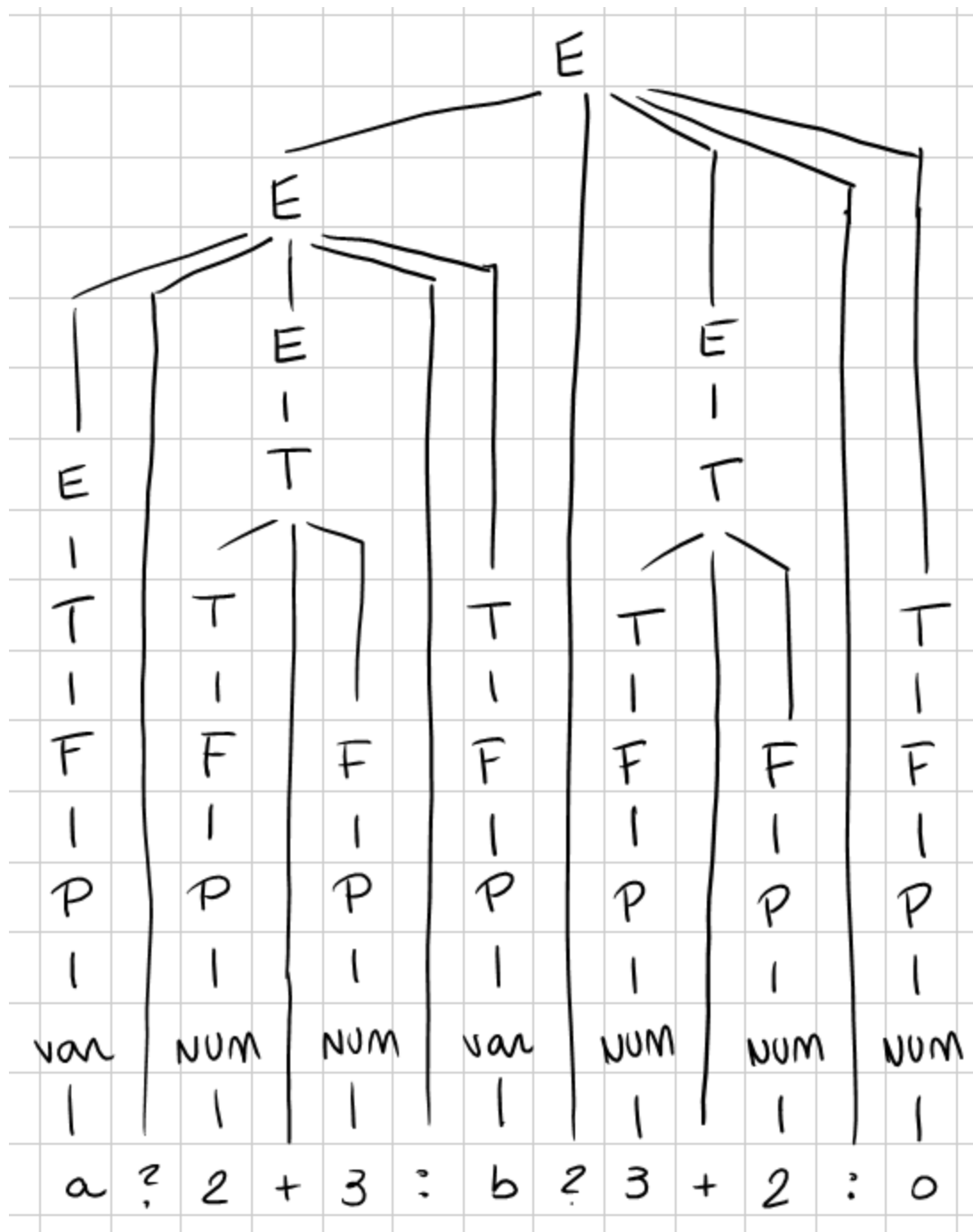
• Resposta:

Pilha	Entrada	Ação
\$	a?2+3:b?3+2:0\$	empilha a
\$a	?2+3:b?3+2:0\$	reduz, gancho=a
\$P	?2+3:b?3+2:0\$	empilha ?
\$P?	2+3:b?3+2:0\$	empilha 2
\$P?2	+3:b?3+2:0\$	reduz, gancho=2
\$P?P	+3:b?3+2:0\$	empilha +
\$P?P+	3:b?3+2:0\$	empilha 3
\$P?P+3	:b?3+2:0\$	reduz, gancho=3
\$P?P+P	:b?3+2:0\$	reduz, gancho=P+P
\$P?T	:b?3+2:0\$	empilha :
\$P?T:	b?3+2:0\$	empilha b
\$P?T:b	?3+2:0\$	reduz, gancho=b
\$P?T:P	?3+2:0\$	reduz, gancho=P?T:P
\$E	?3+2:0\$	empilha ?
\$E?	3+2:0\$	empilha 3
\$E?3	+2:0\$	reduz, gancho=3
\$E?P	+2:0\$	empilha +

Pilha	Entrada	Ação
\$E?P+	2:0\$	empilha 2
\$E?P+2	:0\$	reduz, gancho=2
\$E?P+P	:0\$	reduz, gancho=P+P
\$E?T	:0\$	empilha :
\$E?T:	0\$	empilha 0
\$E?T:0	\$	reduz, gancho=0
\$E?T:P	\$	reduz, gancho=E?T:P
\$E	\$	ACEITA

ASA de precedência de operadores

- Resposta (árvore):



ASA de precedência de operadores

- Funções de precedência
 - Diminuem a necessidade de espaço
 - $O(n^2)$ com matriz (tabela) de precedência
 - $O(2n)$ com funções de precedência
 - onde n é o número de terminais da gramática
- Mapeiam símbolos terminais para inteiros
- Funções de precedência utilizadas
 - f : para símbolo de pilha
 - g : para símbolo de entrada

Funções de precedência

- Algoritmo

1. Criar símbolos f_a e g_a para cada terminal a e para o símbolo $\$$
2. Distribuir os símbolos criados em **grupos**
 - Se $a = b$ então f_a e g_b ficam no mesmo grupo
 - Se $a = b$ e $c = b$ então f_a e f_c ficam no mesmo grupo que g_b
 - Se, no caso anterior, tem-se ainda que $c = d$ então f_a , f_c , g_b e g_d ficam no mesmo grupo mesmo que $a = d$ não ocorra
3. Gerar um grafo direcionado no qual os nós são os grupos formados em 2
 - Para quaisquer a e b
 - Se $a > b$ construa um arco do grupo f_a para o grupo g_b
 - Se $a < b$ construa um arco do grupo g_b para o grupo f_a
4. Se o grafo contém ciclo, as funções de precedência não existem. Se não houver ciclos
 - $f(a)$ é igual ao comprimento do caminho mais longo iniciando em f_a
 - $g(a)$ é igual ao comprimento do caminho mais longo iniciando em g_a

Funções de precedência

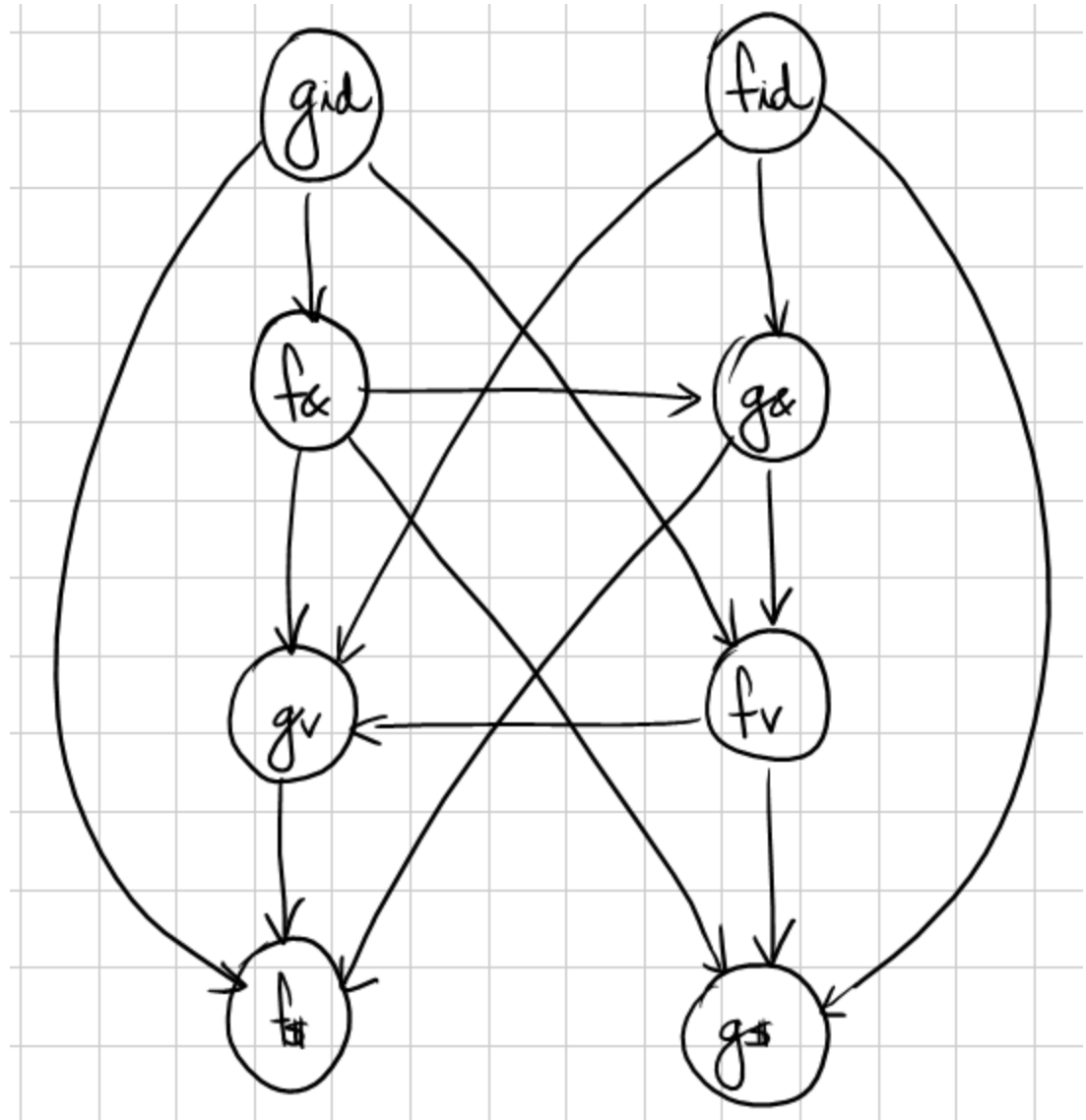
- Exemplo
- Dada a gramática de expressões lógicas (versão + simples)
 - $E \rightarrow E \vee T \mid T$
 - $T \rightarrow T \& F \mid F$
 - $F \rightarrow \text{id}$
- Encontre as funções de precedência correspondentes

	id	v	&	\$
id		>	>	>
v	<	>	<	>
&	<	>	>	>
\$	<	<	<	

Funções de precedência

- Neste caso, o passo 2 não faz nenhum agrupamento

	id	v	&	\$
id		>	>	>
v	<	>	<	>
&	<	>	>	>
\$	<	<	<	



Funções de precedência

- Passo 4

	id	v	&	\$
id		>	>	>
v	<	>	<	>
&	<	>	>	>
\$	<	<	<	

	id	v	&	\$
f	4	2	4	0
g	5	1	3	0

ASA de precedência de operadores

- Exercício - dada a gramática de expressões lógicas (versão + simples)
 - $E \rightarrow E \vee T \mid T$
 - $T \rightarrow T \& F \mid F$
 - $F \rightarrow \text{id}$
- Usando as funções de precedência

	id	v	&	\$
f	4	2	4	0
g	5	1	3	0

- Reconheça a cadeia **id & id v id**
- Usando a seguinte lógica
 - Se $f(a) < g(b)$ ou $f(a) = g(b)$ então empilha
 - Se $f(a) > g(b)$ então reduz

ASA de precedência de operadores

- Resposta

Pilha	Entrada	Ação
\$	id & id v id\$	empilha id
\$id	& id v id\$	reduz, gancho=id
\$F	& id v id\$	empilha &
\$F &	id v id\$	empilha id
\$F & id	v id\$	reduz, gancho=id
\$F & F	v id\$	reduz, gancho=F & F
\$T	v id\$	empilha v
\$T v	id\$	empilha id
\$T v id	\$	reduz, gancho=id
\$T v F	\$	reduz, gancho=T v F
\$E	\$	ACEITA

ASA de precedência de operadores

- Exercício – faça a ASA usando as funções de precedência:

- $E \rightarrow E \ ? \ E \ : \ T \mid T$
- $T \rightarrow \text{var}$

Reconhecer cadeia

a ? b ? c : d : e

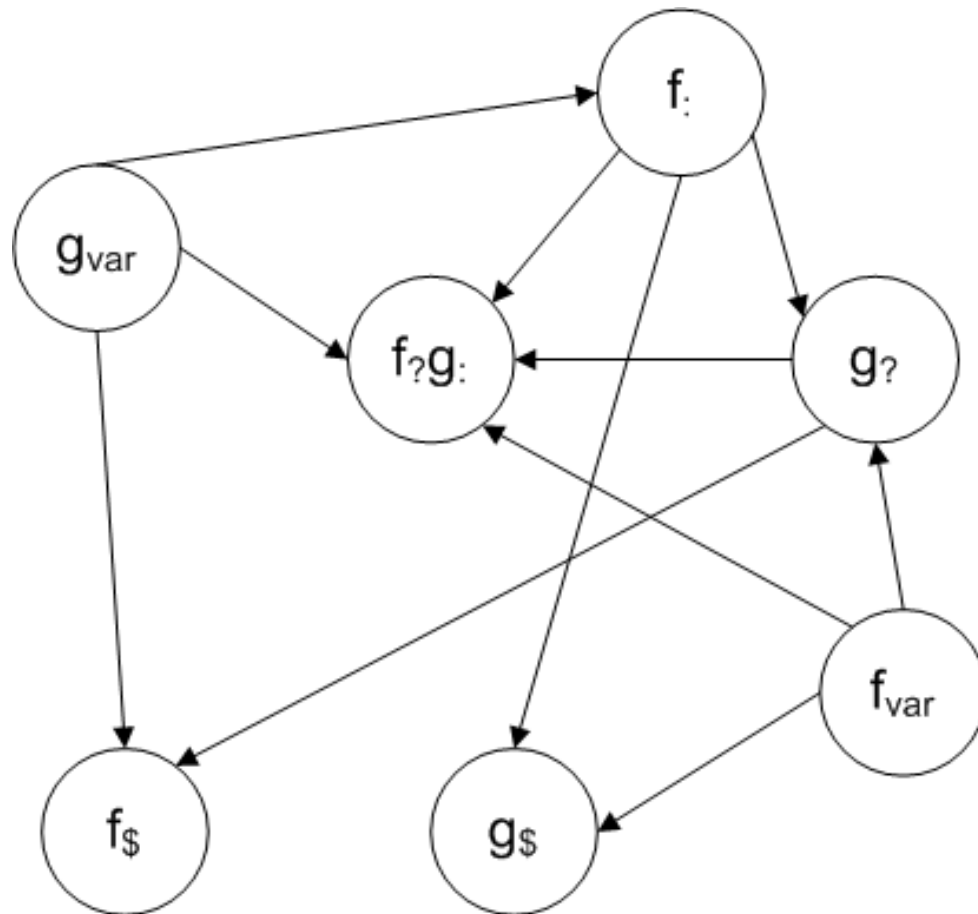
	?	:	var	\$
?	<	=	<	
:	>	>	<	>
var	>	>		>
\$	<		<	ok

ASA de precedência de operadores

- Resposta:
 - Passo 1: Criar símbolos f e g para os operadores:
 - $f_?$, $f_:$, f_{var} , $f_\$$
 - $g_?$, $g_:$, g_{var} , $g_\$$
 - Passo 2: Agrupar símbolos com mesma equivalência
 - Neste caso $? = :$
 - Portanto $f_?$ e $g_:$ ficam no mesmo grupo

ASA de precedência de operadores

- Passo 3: Desenhar o grafo de precedência



ASA de precedência de operadores

- Passo 4: Calcular os comprimentos e montar a tabela das funções de precedência

	?	:	var	\$
f	0	2	2	0
g	1	0	3	0

ASA de precedência de operadores

- Passo 5:
Realizar a
análise

Pilha	Entrada	Ação
\$	a?b?c:d:e\$	empilha a
\$a	?b?c:d:e\$	Reduz, gancho=a
\$T	?b?c:d:e\$	Empilha ?
\$T?	b?c:d:e\$	Empilha b
\$T?b	?c:d:e\$	Reduz, gancho=b
\$T?T	?c:d:e\$	Empilha ?
\$T?T?	c:d:e\$	Empilha c
\$T?T?c	:d:e\$	Reduz, gancho=c
\$T?T?T	:d:e\$	Empilha :
\$T?T?T:	d:e\$	Empilha d
\$T?T?T:d	:e\$	Reduz, gancho=d
\$T?T?T:T	:e\$	Reduz, gancho=T?T:T
\$T?E	:e\$	Empilha :
\$T?E:	e\$	Empilha e
\$T?E:e	\$	Reduz, gancho=e
\$T?E:T	\$	Reduz, gancho=T?E:T
\$E	\$	ACEITA

ASA de precedência de operadores

- Repita o exercício anterior para a cadeia $a?b:c?d:e$

Pilha	Entrada	Ação
\$	a?b:c?d:e\$	Empilha a
\$a	?b:c?d:e\$	Reduz, gancho=a
\$T	?b:c?d:e\$	Empilha ?
\$T?	b:c?d:e\$	Empilha b
\$T?b	:c?d:e\$	Reduz, gancho=b
\$T?T	:c?d:e\$	Empilha :
\$T?T:	c?d:e\$	Empilha c
\$T?T:c	?d:e\$	Reduz, gancho=c
\$T?T:T	?d:e\$	Reduz, gancho=T?T:T
\$E	?d:e\$	Empilha ?
\$E?	d:e\$	Empilha d
\$E?d	:e\$	Reduz, gancho=d
\$E?T	:e\$	Empilha :
\$E?T:	e\$	Empilha e
\$E?T:e	\$	Reduz, gancho=e
\$E?T:T	\$	Reduz, gancho=E?T:T
\$E	\$	ACEITA

ASA de precedência de operadores

- Repita o exercício anterior para a cadeia $a : b ? c$

Pilha	Entrada	Ação
\$	$a : b ? c \$$	Empilha a
$\$a$	$: b ? c \$$	Reduz, gancho= a
$\$T$	$: b ? c \$$	Empilha $:$
$\$T:$	$b ? c \$$	Empilha b
$\$T:b$	$? c \$$	Reduz, gancho= b
$\$T:T$	$? c \$$	Tenta reduzir, mas é impossível! Erro!

ASA de precedência de operadores

- Fazendo
com a tabela
de
precedência
o mesmo
exercício:

Pilha	Entrada	Ação
\$	a:b?c\$	Empilha a
\$a	:b?c\$	Reduz, gancho=a
\$T	:b?c\$	Precedência indefinida! Erro!

	?	:	var	\$
?	<	=	<	
:	>	>	<	>
var	>	>		>
\$	<		<	ok

ASA de precedência de operadores

- Com as funções de precedência, tem-se o mesmo comportamento no reconhecimento das cadeias
- Erros são detectados em ambas as abordagens
 - Porém, com a tabela, a detecção de erros pode ocorrer antes!
- Portanto: as funções de precedência poupam espaço, mas gastam (um pouquinho) mais de tempo para detectar erros

ASA de precedência de operadores

- Vantagens
 - É simples e eficiente
 - Muito eficiente no reconhecimento de expressões aritméticas e lógicas
- Desvantagens
 - Tem dificuldade em lidar com operadores iguais que tenham significados distintos
 - Por exemplo, o operador “-” que pode ser binário ou unário
 - É aplicável a apenas uma classe restrita de gramáticas (gramáticas de operadores)

ASA

- Na próxima aula
 - Veremos a técnica LR
 - Mais poderosa e abrangente do que a ASA de precedência de operadores
 - Mais poderosa e abrangente do que a ASD
 - Porém, mais difícil de implementar

Fim