Construção de Compiladores

Analisador de precedência fraca

Professor: Luciano Ferreira Silva, Dr.

Reconhecimento de sentença

- O procedimento de reconhecimento de uma sentença com o analisador sintático de precedência fraca também é uma aplicação do autômato de pilha;
- A sentença a ser analisada é uma seqüência de tokens fornecida pelo analisador léxico, acrescida ao final do símbolo delimitador de sentença;
- O delimitador de sentença também é inserido como o primeiro elemento da pilha;

Reconhecimento de sentenças

- **A cada passo do procedimento a tabela DR é consultada com:**
  - ✓ o símbolo da pilha sendo indica a linha;
  - ✓ o símbolo analisado na sentença indica a coluna;
- **Se a entrada nessa célula estiver vazia tem-se uma situação de erro;**
- **Se indicar deslocamento (D) então:**
  - ✓ o símbolo no início da sentença é removido e inserido no topo da pilha.

Reconhecimento de sentenças

- **Se indicar redução (R), então:**
  - ✓ Todos os símbolos no topo da pilha que combinam com o lado direito da produção são removidos da pilha;
  - ✓ E substituídos pelo símbolo do lado esquerdo, que é empilhado.
  - ✓ Essa ação corresponde a criar uma subárvore cuja raiz é o símbolo que foi inserido na pilha e cujos filhos são os elementos retirados da pilha.

Reconhecimento de sentenças

- **O procedimento é concluído com sucesso quando:**
  - ✓ Estiverem armazenados na pilha dois símbolos:
    - O sentencial no topo;
    - O delimitador de sentença;
  - ✓ Estiver sendo analisado na sentença o delimitado de sentença;
- **Esta situação equivale à obtenção da árvore sintática completa:**
  - ✓ Com o símbolo sentencial na raiz e nenhum símbolo da sentença sem pertencer à árvore;

Reconhecimento de sentenças

- **Considere a gramática *F*:**
  - ✓ Símbolos não-terminais $V_N = \{S, X\}$, sendo S o símbolo sentencial;
  - ✓ Símbolos terminais $V_T = \{a, b, c, d, e\}$
  - ✓ Produções $P = \{S \rightarrow aSb, S \rightarrow Xc, X \rightarrow d, X \rightarrow e\}$
- **Considere a sentença *aadcbb* para teste do reconhecimento;**

Reconhecimento de sentença

Lista de tokens

a	***a***	***d***	**c**	***b***	***b***	***$***

$

Pilha

	a	*b*	*c*	*d*	*e*	*$*
S		*D*				
X			*D*			
a	*D*			*D*	*D*	
b		*R*				*R*
c		*R*				*R*
d			*R*			
e			*R*			
$	*D*			*D*	*D*	

Reconhecimento de sentença

Lista de tokens

a	***d***	**c**	***b***	***b***	***$***

a
$

Pilha

	a	*b*	*c*	*d*	*e*	*$*
S		*D*				
X			*D*			
a	*D*			*D*	*D*	
b		*R*				*R*
c		*R*				*R*
d			*R*			
e			*R*			
$	*D*			*D*	*D*	

a

Reconhecimento de sentença

Lista de tokens

d	**c**	***b***	***b***	***$***

a
a
$

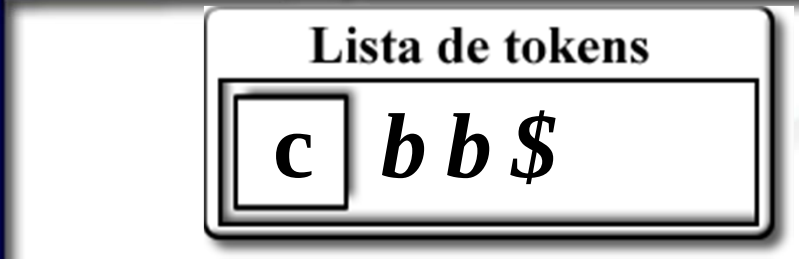
Pilha

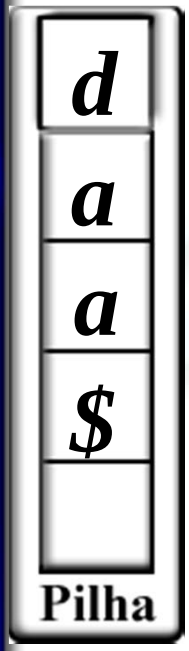
	a	*b*	*c*	*d*	*e*	*$*
S		*D*				
X			*D*			
a	*D*			*D*	*D*	
b		*R*				*R*
c		*R*				*R*
d			*R*			
e			*R*			
$	*D*			*D*	*D*	

a

a

Reconhecimento de sentença





	a	b	c	d	e	$
S		D				
X			D			
a	D			D	D	
b		R				R
c		R				R
d			R			
e			R			
$	D			D	D	

a

a

d

Produção: X → d

Reconhecimento de sentença

Lista de tokens

c	*b b $*

Pilha
X
a
a
$

	a	*b*	*c*	*d*	*e*	*$*
S		*D*				
X			*D*			
a	*D*			*D*	*D*	
b		*R*				*R*
c		*R*				*R*
d			*R*			
e			*R*			
$	*D*			*D*	*D*	

a

a

X
|
d

Reconhecimento de sentença

Lista de tokens

b	***b $***

Pilha
c
X
a
a
$

	a	*b*	*c*	*d*	*e*	*$*
S		*D*				
X			*D*			
a	*D*			*D*	*D*	
b		*R*				*R*
c		*R*				*R*
d			*R*			
e			*R*			
$	*D*			*D*	*D*	

a

a

X ***c***

d

Produção: S → Xc

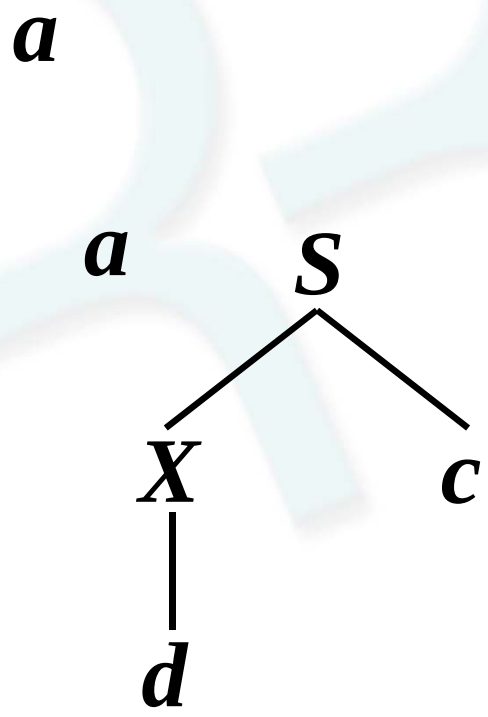
Reconhecimento de sentença

Lista de tokens

b	***b***	***$***

Pilha
S
a
a
$

	a	*b*	*c*	*d*	*e*	*$*
S		*D*				
X			*D*			
a	*D*			*D*	*D*	
b		*R*				*R*
c		*R*				*R*
d			*R*			
e			*R*			
$	*D*			*D*	*D*	



Reconhecimento de sentença

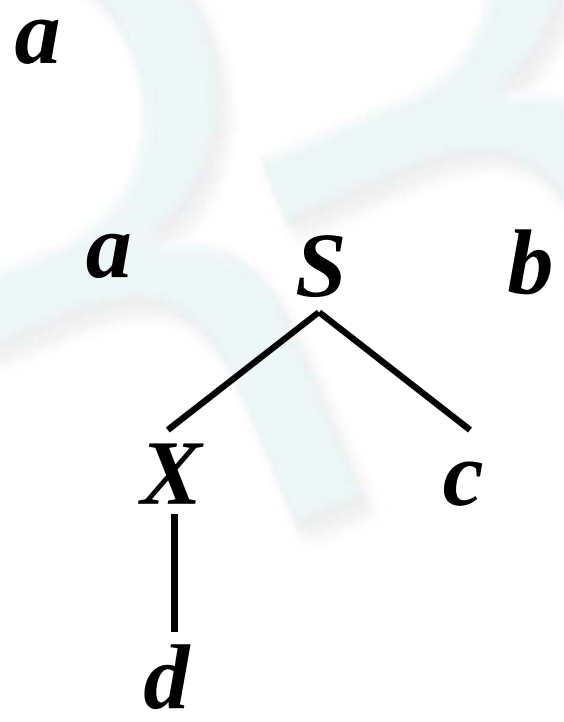
Lista de tokens

b	$

Pilha

b
S
a
a
$

	a	b	c	d	e	$
S		D				
X			D			
a	D			D	D	
b		R				R
c		R				R
d			R			
e			R			
$	D			D	D	



Produção: S → aSb

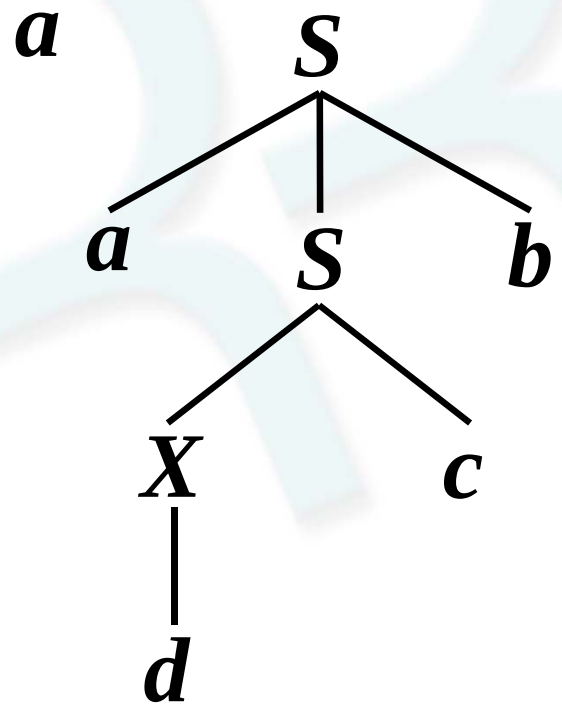
Reconhecimento de sentença

Lista de tokens

b	$

Pilha
S
a
$

	a	b	c	d	e	$
S		D				
X			D			
a	D			D	D	
b		R				R
c		R				R
d			R			
e			R			
$	D			D	D	



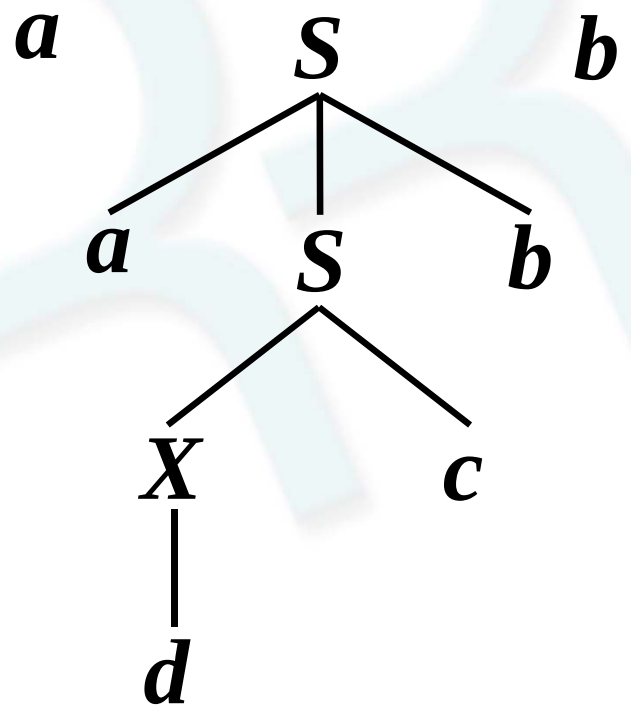
Reconhecimento de sentença

Lista de tokens

$

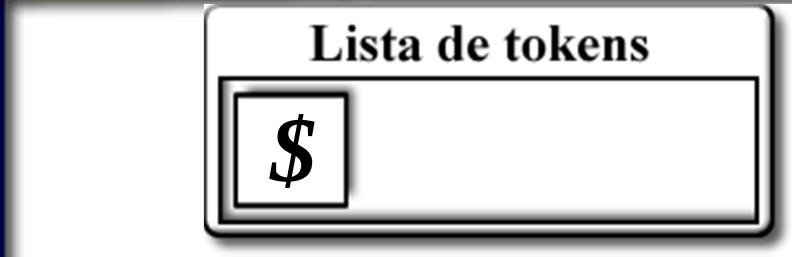
Pilha
b
S
a
$

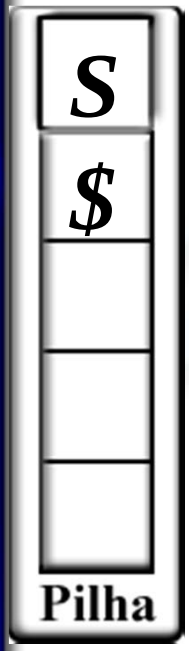
	a	b	c	d	e	$
S		D				
X			D			
a	D			D	D	
b		R				R
c		R				R
d			R			
e			R			
$	D			D	D	



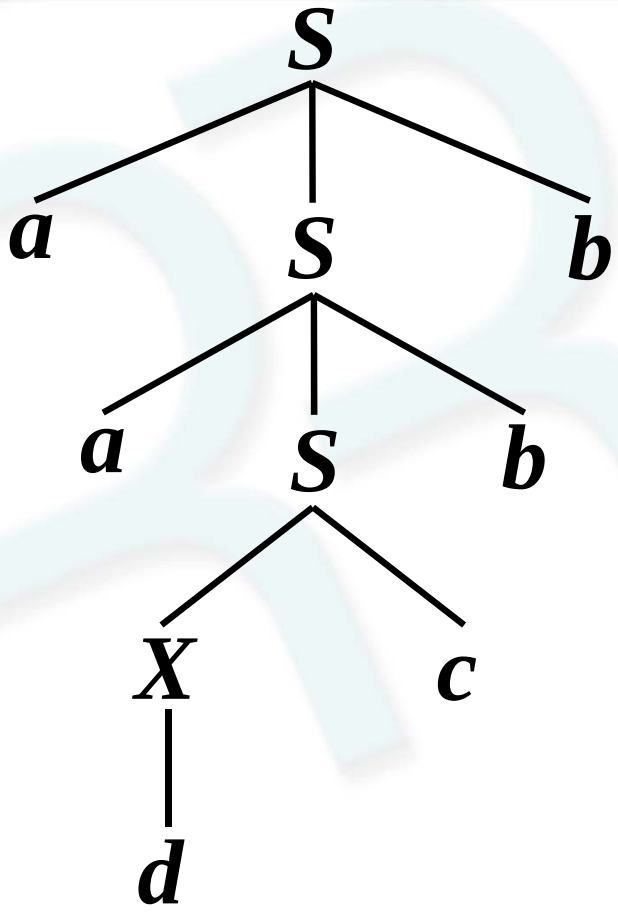
Produção: S → aSb

Reconhecimento de sentença





	a	b	c	d	e	$
S		D				
X			D			
a	D			D	D	
b		R				R
c		R				R
d			R			
e			R			
$	D			D	D	



Fim do reconhecimento

Reconhecimento de sentença

- Considere agora a gramática G`, (com E sentencial).

Gramática G`, equivalente à gramática G mas sem ambiguidade

E → E + M	Produção 1: adição com recursividade
E → M	Produção 2: marca o fim da recursividade da adição
M → M x P	Produção 3: multiplicação com recursividade, contempla a associatividade
M → P	Produção 4: marca o fim recursividade da multiplicação
P → (E)	Produção 5: possibilita o uso de parênteses
P → v	Produção 6: terminal

- E considere também a sentença v + v × v.

Reconhecimento de sentença

- **Obtendo os conjuntos *ESQ* e *DIR*:**

Gramática G`

$E \rightarrow E + M$

$E \rightarrow M$

$M \rightarrow M \times P$

$M \rightarrow P$

$P \rightarrow (E)$

$P \rightarrow v$

$ESQ(E) = \{M, P, (, v\}$ $\quad DIR(E) = \{M, P,), v\}$

$ESQ(M) = \{P, (, v\}$ $\quad DIR(M) = \{P,), v\}$

$ESQ(P) = \{(, v\}$ $\quad DIR(P) = \{), v\}$

Reconhecimento de sentença

- **Obtendo as relações ≈:**

Gramática G`

$E \rightarrow E + M$

$E \rightarrow M$

$M \rightarrow M \times P$

$M \rightarrow P$

$P \rightarrow (E)$

$P \rightarrow v$

$E \approx +$ $\quad + \approx M$

$M \approx \times$ $\quad \times \approx P$

$(\approx E$ $\quad E \approx)$

Reconhecimento de sentença

Obtendo as relações «:

- É preciso analisar as relações ≈ que têm símbolos não-terminais do lado direito e os elementos do conjunto *ESQ* do não- terminal.

$+ \approx M$

$ESQ(M) = \{P, (, v\}$

$+ \mathbin{«} P$

$+ \mathbin{«} ($

$+ \mathbin{«} v$

$\times \approx P$

$ESQ(P) = \{(, v\}$

$\times \mathbin{«} ($

$\times \mathbin{«} v$

$(\approx E$

$ESQ(E) = \{M, P, (, v\}$

$(\mathbin{«} M$

$(\mathbin{«} P$

$(\mathbin{«} ($

$(\mathbin{«} v$

Reconhecimento de sentença

Obtendo as relações »:

- É preciso analisar as relações ≈ entre um símbolo não-terminal no lado esquerdo e um símbolo terminal do lado direito e os elementos do conjunto *DIR* do não- terminal.
- Não há relações ≈ entre dois não-terminais.

$E \approx +$ $\quad DIR(E) = \{M, P,), v\}$

$M \approx \times$ $\quad DIR(M) = \{P,), v\}$

$E \approx)$ $\quad DIR(E) = \{M, P,), v\}$

$M \gg +$

$P \gg +$

$) \gg +$

$v \gg +$

$P \gg \times$

$) \gg \times$

$v \gg \times$

$M \gg)$

$P \gg)$

$) \gg)$

$v \gg)$

Reconhecimento de sentença

- Tabela com as relações Wirth-Weber:

	E	*M*	*P*	+	×	(	)	*v*
E				≈			≈	
M				»	≈		»	
P				»	»		»	
+		≈	«			«		«
×			≈			«		«
(	≈	«	«			«		«
)				»	»		»	
v				»	»		»	

Reconhecimento de sentença

1. As condições 1, 2 e 3 são satisfeitas;
2. Não há nenhum par de símbolo que esteja relacionando simultaneamente pela relação » e por a alguma outra relação;
3. Analisando produções que terminam com o mesmo símbolo:
   1. $E \rightarrow E \underline{+} M$ e $\underline{E} \rightarrow M$; o cruzamento entre + e E é vazio;
   2. $M \rightarrow M \underline{\times} P$ e $\underline{M} \rightarrow P$; o cruzamento entre $\times$ e M é vazio;
4. Portanto a gramática é de precedência fraca

Reconhecimento de sentença

- ## Construção da tabela DR:

$ESQ(E) = \{M, P, (, v\}$

$DIR(E) = \{M, P,), v\}$

$\$ \lessdot M$

$\$ \lessdot P$

$\$ \lessdot ($

$\$ \lessdot v$

$M \gtrdot \$$

$P \gtrdot \$$

$) \gtrdot \$$

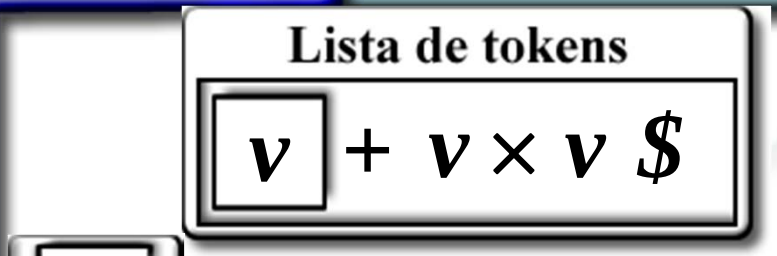
$v \gtrdot \$$

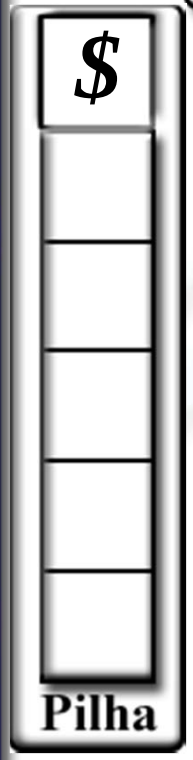
Reconhecimento de sentença

- **Tabela DR:**

	+	×	(	)	v	$
E	D			D		
M	R	D		R		R
P	R	R		R		R
+			D		D	
×			D		D	
(			D		D	
)	R	R		R		R
v	R	R		R		R
$			D		D	

Reconhecimento de sentença





	+	×	(	)	v	$
E	D			D		
M	R	D		R		R
P	R	R		R		R
+			D		D	
×			D		D	
(			D		D	
)	R	R		R		R
v	R	R		R		R
$			D		D	

Reconhecimento de sentença

Lista de tokens

+	*v* × *v* $

Pilha

v
$

	+	×	(	)	*v*	$
E	D			D		
M	R	D		R		R
P	R	R		R		R
+			D		D	
×			D		D	
(			D		D	
)	R	R		R		R
v	R	R		R		R
$			D		D	

v

Produção: P → v

Reconhecimento de sentença

Lista de tokens

+	*v* × *v* $

Pilha

P
$

	+	×	(	)	*v*	$
E	D			D		
M	R	D		R		R
P	R	R		R		R
+			D		D	
×			D		D	
(			D		D	
)	R	R		R		R
v	R	R		R		R
$			D		D	

P
|
v

Produção: M → P

Reconhecimento de sentença

Lista de tokens

+	*v* × *v* $

Pilha

M
$

	+	×	(	)	*v*	$
E	D			D		
M	R	D		R		R
P	R	R		R		R
+			D		D	
×			D		D	
(			D		D	
)	R	R		R		R
v	R	R		R		R
$			D		D	

M
|
P
|
v

Produção: E → M

Reconhecimento de sentença

Lista de tokens

+	*v* × *v* $

Pilha

E
$

	+	×	(	)	*v*	$
E	D			D		
M	R	D		R		R
P	R	R		R		R
+			D		D	
×			D		D	
(			D		D	
)	R	R		R		R
v	R	R		R		R
$			D		D	

E

M

P

v

Reconhecimento de sentença

Lista de tokens

v	×	v	$

Pilha

+
E
$

	+	×	(	)	v	$
E	D			D		
M	R	D		R		R
P	R	R		R		R
+			D		D	
×			D		D	
(			D		D	
)	R	R		R		R
v	R	R		R		R
$			D		D	

E +

|

M

|

P

|

v

Reconhecimento de sentença

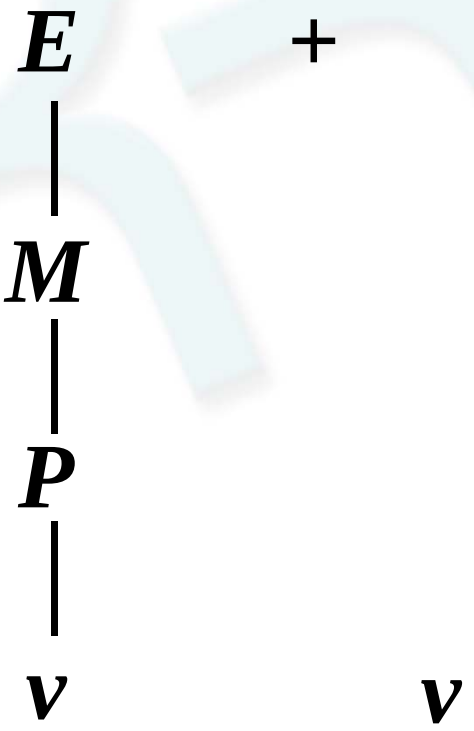
Lista de tokens

×	*v*	*$*

Pilha

v
+
E
$

	+	×	(	)	*v*	*$*
E	D			D		
M	R	D		R		R
P	R	R		R		R
+			D		D	
×			D		D	
(			D		D	
)	R	R		R		R
v	R	R		R		R
$			D		D	



Produção: P → v

Reconhecimento de sentença

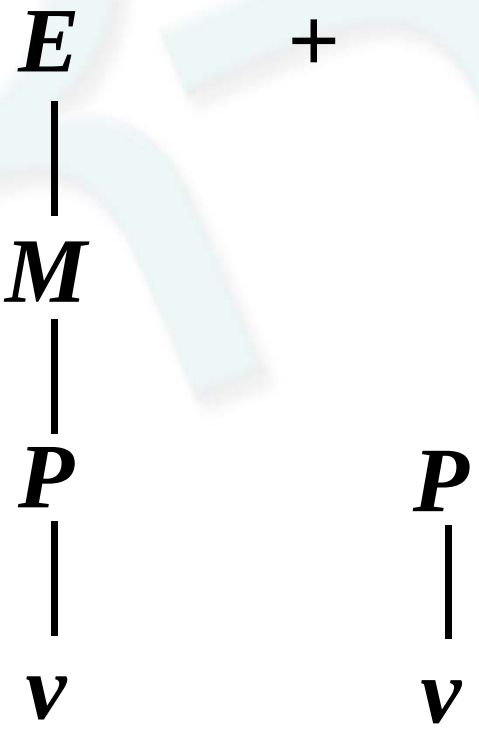
Lista de tokens

×	*v*	*$*

Pilha

P
+
E
$

	+	×	(	)	*v*	*$*
E	D			D		
M	R	D		R		R
P	R	R		R		R
+			D		D	
×			D		D	
(			D		D	
)	R	R		R		R
v	R	R		R		R
$			D		D	



Produção: M → P

Reconhecimento de sentença

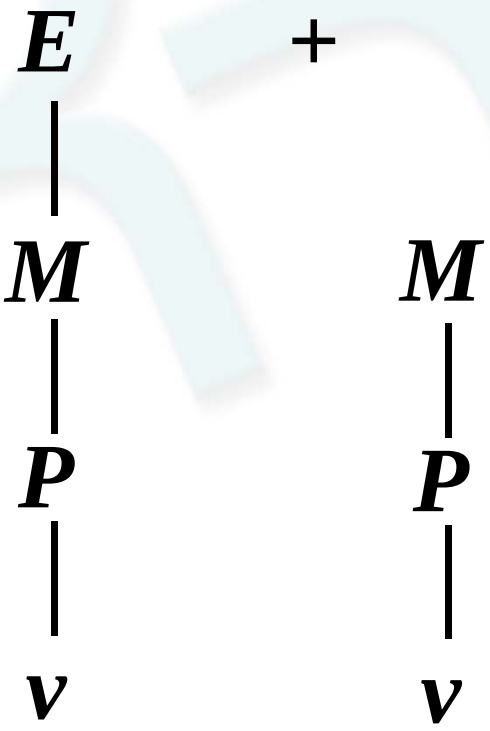
Lista de tokens

×	*v*	*$*

Pilha

M
+
E
$

	+	×	(	)	*v*	*$*
E	D			D		
M	R	D		R		R
P	R	R		R		R
+			D		D	
×			D		D	
(			D		D	
)	R	R		R		R
v	R	R		R		R
$			D		D	



Reconhecimento de sentença

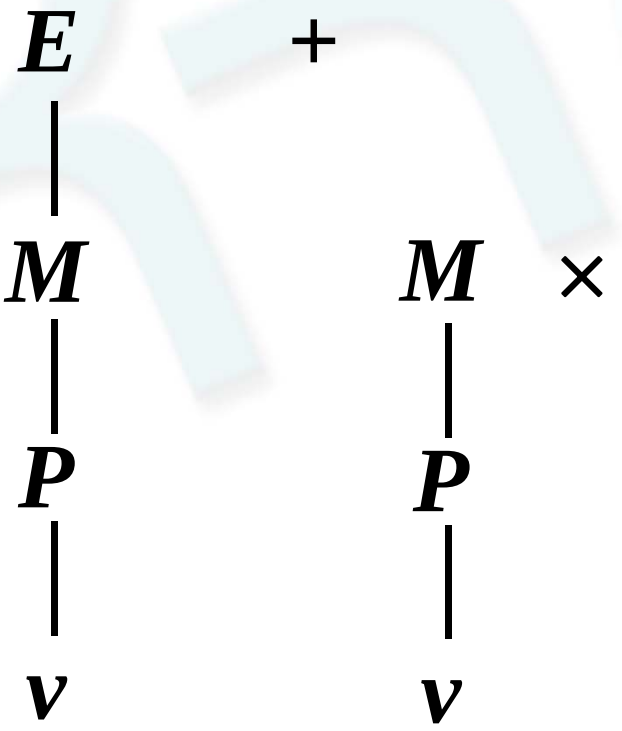
Lista de tokens

v	$

Pilha

×
M
+
E
$

	+	×	(	)	v	$
E	D			D		
M	R	D		R		R
P	R	R		R		R
+			D		D	
×			D		D	
(			D		D	
)	R	R		R		R
v	R	R		R		R
$			D		D	



Reconhecimento de sentença

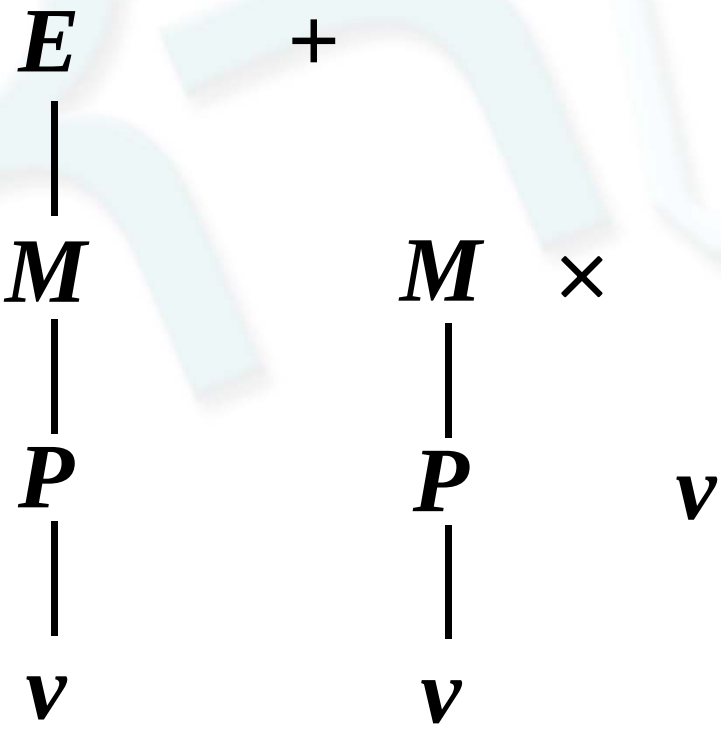
Lista de tokens

$

Pilha

v
×
M
+
E
$

	+	×	(	)	v	$
E	D			D		
M	R	D		R		R
P	R	R		R		R
+			D		D	
×			D		D	
(			D		D	
)	R	R		R		R
v	R	R		R		R
$			D		D	



Produção: P → v

Reconhecimento de sentença

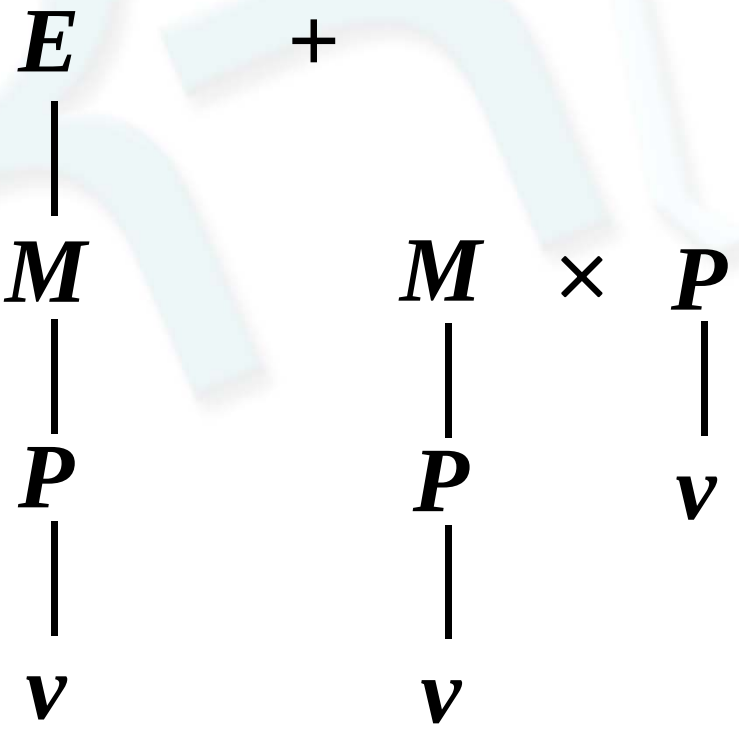
Lista de tokens

$

Pilha

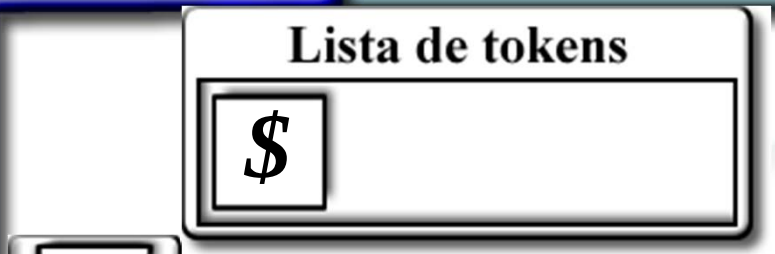
P
×
M
+
E
$

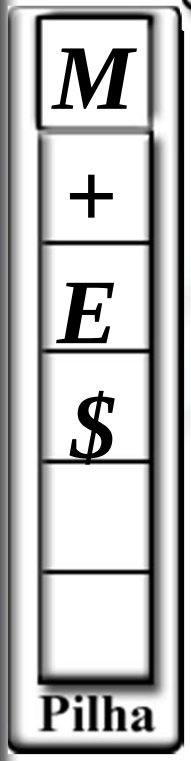
	+	×	(	)	v	$
E	D			D		
M	R	D		R		R
P	R	R		R		R
+			D		D	
×			D		D	
(			D		D	
)	R	R		R		R
v	R	R		R		R
$			D		D	



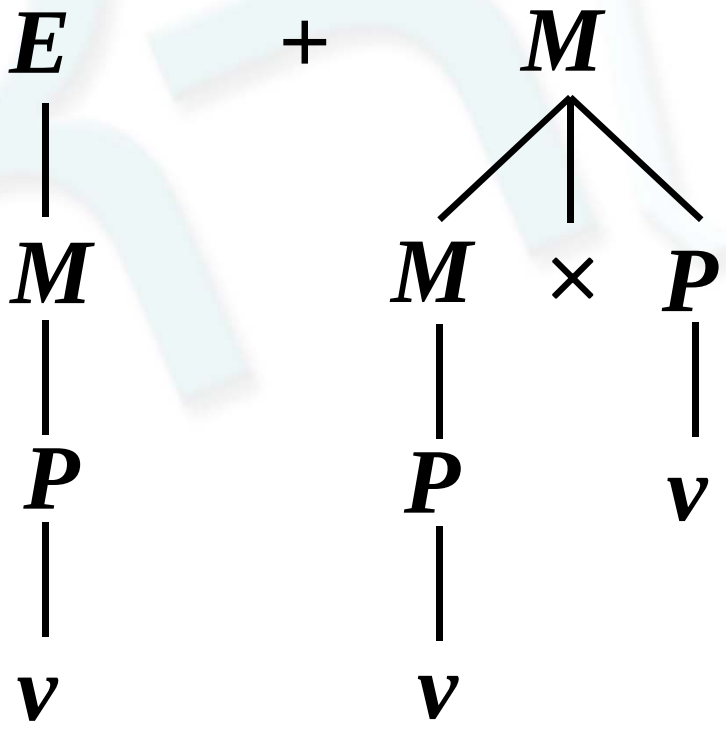
Produção: M → M × P* OU *M → P

Reconhecimento de sentença



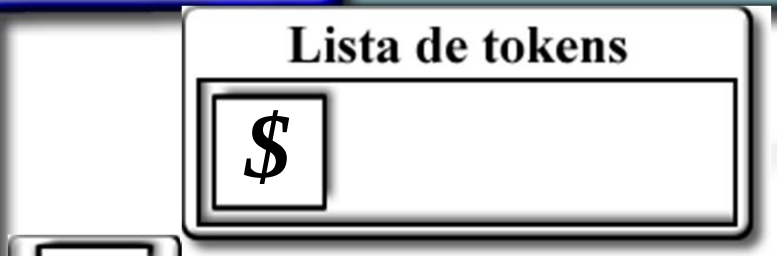


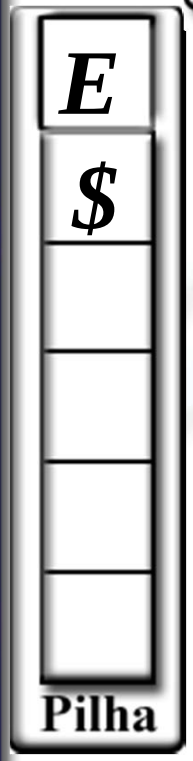
	+	×	(	)	v	$
E	D			D		
M	R	D		R		R
P	R	R		R		R
+			D		D	
×			D		D	
(			D		D	
)	R	R		R		R
v	R	R		R		R
$			D		D	



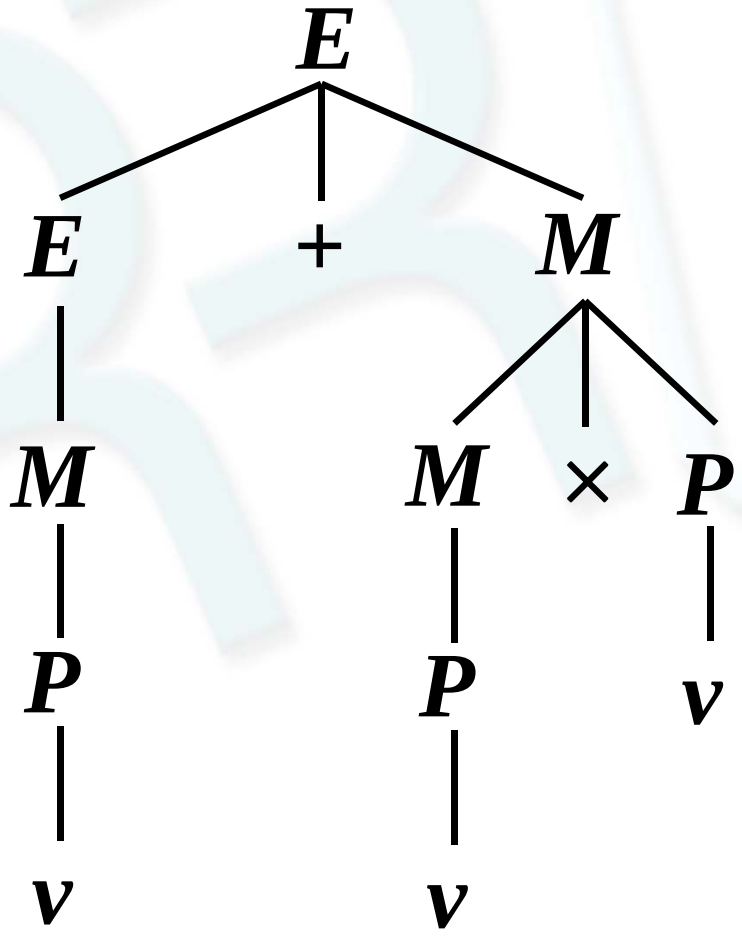
Produção: E → E + M OU E → M

Reconhecimento de sentença





	+	×	(	)	v	$
E	D			D		
M	R	D		R		R
P	R	R		R		R
+			D		D	
×			D		D	
(			D		D	
)	R	R		R		R
v	R	R		R		R
$			D		D	



Fim do reconhecimento

6° Trabalho

- Construa um analisador sintática de precedência fraca, completo, para a gramática G`.
- Sugestão: aproveite o código do analisador preditivo da gramática G``.

Gramática de precedência fraca

1. A gramática não pode conter nenhuma produção cujo lado direito seja a string vazia;
2. A gramática deve ser unicamente inversível, ou seja, não pode haver duas produções que tenham o mesmo lado direito;
3. A gramática deve ser livre de ciclos;
   - ✓ Se *A* é um símbolo não-terminal na gramática, não deve existir uma seqüência de derivações que produza como resultado o mesmo símbolo *A*;

Reconhecimento de sentença

- Tabela com as relações Wirth-Weber:

	E	*M*	*P*	+	×	(	)	*v*
E				≈			≈	
M				»	≈		»	
P				»	»		»	
+		≈	«			«		«
×			≈			«		«
(	≈	«	«			«		«
)				»	»		»	
v				»	»		»	