








Judson Santos Santiago

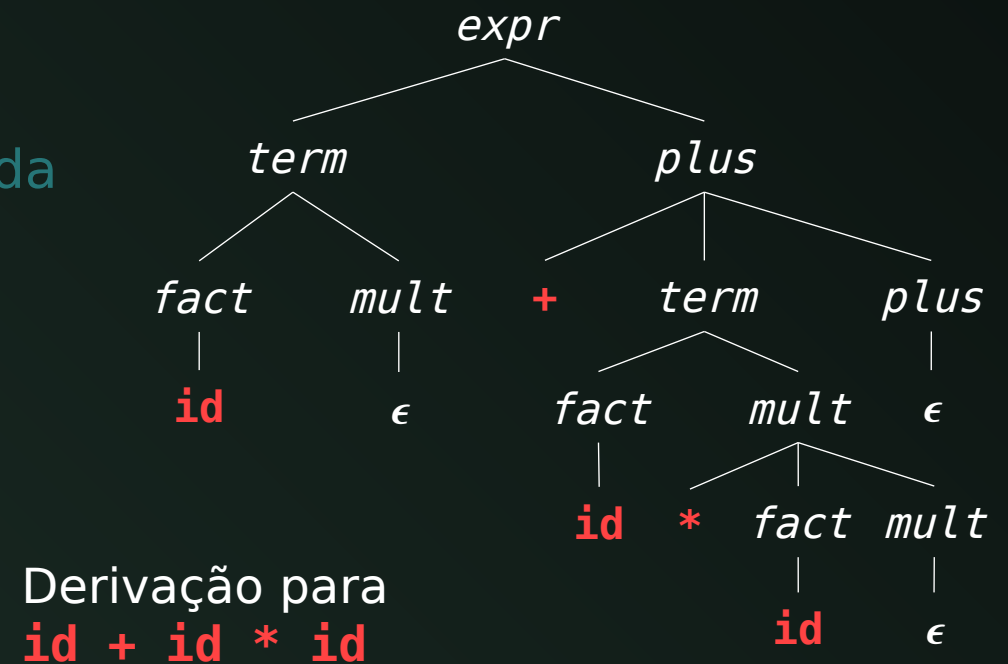
Analizador Preditivo

Compiladores

Introdução

- A **análise descendente** constrói uma **árvore de derivação**
 - De cima para baixo
 - Da esquerda para direita
 - Produz uma **derivação mais à esquerda**

<i>expr</i>		<i>term plus</i>
<i>plus</i>		<i>+ term</i>
		<i>plus</i>
		ϵ
<i>term</i>		<i>fact mult</i>
<i>mult</i>		<i>* fact</i>
		<i>mult</i>
		ϵ
<i>fact</i>		<i>(expr)</i>
		<i>id</i>



Introdução

- Os **analisadores sintáticos preditivos** podem ser construídos para uma classe de gramática chamada de **LL(1)**
 - L = a entrada é lida da esquerda para a direita
 - L = é realizada uma derivação mais à esquerda
 - 1 = apenas um símbolo precisa ser lido para tomar decisões
- A classe de gramáticas LL(1)
 - É **rica o suficiente** para especificar a maioria das linguagens de programação
 - Mas **não é simples** de escrever
 - Tem que ser **fatorada, sem recursão à esquerda e sem ambiguidade**

Gramáticas LL(1)


- Uma gramática é LL(1) se obedecer às seguintes condições:
 - Sempre que $A \rightarrow \alpha \mid \beta$ forem duas produções distintas:
 1. α e β não derivam cadeias começando com o mesmo terminal
 2. No máximo um dos dois, α ou β , pode derivar a cadeia vazia
 3. Se $\beta \xRightarrow{*} \epsilon$, então α não deriva nenhuma cadeia começando com um terminal em FOLLOW(A). De modo semelhante, se $\alpha \xRightarrow{*} \epsilon$, então β não deriva nenhuma cadeia começando com um terminal em FOLLOW(A)

Gramáticas LL(1)

- As duas primeiras regras são equivalentes a dizer que $\text{FIRST}(\alpha)$ e $\text{FIRST}(\beta)$ são conjuntos disjuntos

Sempre que $A \rightarrow \alpha \mid \beta$ forem duas produções distintas:

1. α e β não derivam cadeias começando com o mesmo terminal
2. No máximo um dos dois, α ou β , pode derivar a cadeia vazia

```
stm   while (expr) stmt1  
t  
    | do stmt1 while (expr);  
    | for (expr; expr; expr) stmt1
```

Gramáticas LL(1)

- A terceira regra diz que $\text{FIRST}(\alpha)$ e $\text{FOLLOW}(A)$ são disjuntos quando $\text{FIRST}(\beta)$ contém ϵ , e que $\text{FIRST}(\beta)$ e $\text{FOLLOW}(A)$ são disjuntos quando $\text{FIRST}(\alpha)$ contém ϵ .

Sempre que $A \rightarrow \alpha \mid \beta$ forem duas produções distintas:

3. Se $\beta \Rightarrow \epsilon$, então α não deriva nenhuma cadeia começando com um terminal em $\text{FOLLOW}(A)$. De modo semelhante, se $\alpha \Rightarrow \epsilon$, então β não deriva nenhuma cadeia começando com um terminal em $\text{FOLLOW}(A)$



Gramáticas LL(1)

- Analisadores preditivos podem ser construídos para gramáticas LL(1)
 - As regras fazem com que, para um dado símbolo da entrada, a escolha da produção seja única e previsível
 - Construções de fluxo de controle, com suas palavras-chave distintas, geralmente satisfazem as restrições:

```
inst if (expr) inst else inst
| while (expr) inst
| { block }
```
 - **if**, **while** e **{** definem a produção a ser utilizada

Tabela de Reconhecimento

- Analisadores sintáticos preditivos podem ser implementados:
 - Por funções recursivas
 - Por uma **tabela de reconhecimento preditivo**
 - Seja $M[A, \mathbf{a}]$ um **arranjo bidimensional**
 - A é um não-terminal
 - \mathbf{a} é um terminal ou \$, o marcador de fim de entrada
 - O **método**_{*} **se baseia** na seguinte ideia:
 - A produção $A \rightarrow \alpha$ é escolhida se o próximo símbolo de entrada estiver em $FIRST(\alpha)$
 - Com $\alpha \Rightarrow \epsilon$, a produção $A \rightarrow \alpha$ é escolhida se o símbolo corrente da entrada estiver em $FOLLOW(A)$ ou se o \$ foi alcançado e ele está em $FOLLOW(A)$

Tabela de Reconhecimento

- Construção da tabela de reconhecimento sintático preditivo
 - Entrada: Gramática G
 - Saída: Tabela de análise M
- Método: Para cada produção $A \rightarrow \alpha$ da gramática, faça:
 1. Para cada terminal a em $FIRST(\alpha)$, inclua $A \rightarrow \alpha$ em $M[A, a]$
 2. Se ϵ pertence a $FIRST(\alpha)$, inclua $A \rightarrow \alpha$ em $M[A, a]$ para cada terminal a em $FOLLOW(A)$. Se ϵ pertence a $FIRST(\alpha)$ e $\$$ pertence a $FOLLOW(A)$, acrescente também $A \rightarrow \alpha$ em $M[A, \$]$
 3. Se ao final não houver nenhuma produção em $M[A, a]$, defina $M[A, a]$ como erro

Construção da Tabela

- Considerando as produções:

<i>expr</i>		<i>term plus</i>
<i>plus</i>		<i>+ term plus</i>
		ϵ
<i>term</i>		<i>fact mult</i>
<i>mult</i>		<i>* fact mult</i>
		ϵ
<i>fact</i>		<i>(expr)</i>
		<i>id</i>

$\text{FIRST}(fact) = \{ (, id \}$	$\text{FOLLOW}(expr) = \{ \$,) \}$
$\text{FIRST}(term) = \{ (, id \}$	$\text{FOLLOW}(plus) = \{ \$,) \}$
$\text{FIRST}(expr) = \{ (, id \}$	$\text{FOLLOW}(term) = \{ +, \$,) \}$
$\text{FIRST}(plus) = \{ +, \epsilon \}$	$\text{FOLLOW}(mult) = \{ +, \$,) \}$
$\text{FIRST}(mult) = \{ *, \epsilon \}$	$\text{FOLLOW}(fact) = \{ *, +, \$,) \}$

expr \rightarrow *term plus*

$\text{FIRST}(term\ plus) = \text{FIRST}(term) = \{ (, id \}$

$M[expr, (] = expr \rightarrow term\ plus$

$M[expr, id] = expr \rightarrow term\ plus$

Construção da Tabela

- Considerando as produções:

<i>expr</i>		<i>term plus</i>
<i>plus</i>		<i>+ term plus</i>
		ϵ
<i>term</i>		<i>fact mult</i>
<i>mult</i>		<i>* fact mult</i>
		ϵ
<i>fact</i>		<i>(expr)</i>
		<i>id</i>

$\text{FIRST}(fact) = \{ (, id \}$	$\text{FOLLOW}(expr) = \{ \$,) \}$
$\text{FIRST}(term) = \{ (, id \}$	$\text{FOLLOW}(plus) = \{ \$,) \}$
$\text{FIRST}(expr) = \{ (, id \}$	$\text{FOLLOW}(term) = \{ +, \$,) \}$
$\text{FIRST}(plus) = \{ +, \epsilon \}$	$\text{FOLLOW}(mult) = \{ +, \$,) \}$
$\text{FIRST}(mult) = \{ *, \epsilon \}$	$\text{FOLLOW}(fact) = \{ *, +, \$,) \}$

plus \rightarrow *+* *term plus*

$\text{FIRST}(+ \text{ term plus}) = \text{FIRST}(+) = \{ + \}$
 $M[plus, +] = plus \rightarrow + \text{ term plus}$

Construção da Tabela

- Considerando as produções:

<i>expr</i>	☰	<i>term plus</i>
<i>plus</i>	☰	<i>+ term plus</i>
		ϵ
<i>term</i>	☰	<i>fact mult</i>
<i>mult</i>	☰	<i>* fact mult</i>
		ϵ
<i>fact</i>	☰	<i>(expr)</i>
		<i>id</i>

$\text{FIRST}(fact) = \{ (, id \}$
 $\text{FIRST}(term) = \{ (, id \}$
 $\text{FIRST}(expr) = \{ (, id \}$
 $\text{FIRST}(plus) = \{ +, \epsilon \}$
 $\text{FIRST}(mult) = \{ *, \epsilon \}$

$\text{FOLLOW}(expr) = \{ \$,) \}$
 $\text{FOLLOW}(plus) = \{ \$,) \}$
 $\text{FOLLOW}(term) = \{ +, \$,) \}$
 $\text{FOLLOW}(mult) = \{ +, \$,) \}$
 $\text{FOLLOW}(fact) = \{ *, +, \$,) \}$

plus $\rightarrow \epsilon$

$\text{FOLLOW}(plus) = \{ \$,) \}$
 $M[plus, \$] = plus \rightarrow \epsilon$
 $M[plus,)] = plus \rightarrow \epsilon$

Construção da Tabela

- Considerando as produções:

<i>expr</i>	☰	<i>term plus</i>
<i>plus</i>	☰	<i>+ term plus</i>
		ϵ
<i>term</i>	☰	<i>fact mult</i>
<i>mult</i>	☰	<i>* fact mult</i>
		ϵ
<i>fact</i>	☰	<i>(expr)</i>
		<i>id</i>

$\text{FIRST}(fact) = \{ (, id \}$	$\text{FOLLOW}(expr) = \{ \$,) \}$
$\text{FIRST}(term) = \{ (, id \}$	$\text{FOLLOW}(plus) = \{ \$,) \}$
$\text{FIRST}(expr) = \{ (, id \}$	$\text{FOLLOW}(term) = \{ +, \$,) \}$
$\text{FIRST}(plus) = \{ +, \epsilon \}$	$\text{FOLLOW}(mult) = \{ +, \$,) \}$
$\text{FIRST}(mult) = \{ *, \epsilon \}$	$\text{FOLLOW}(fact) = \{ *, +, \$,) \}$

term \rightarrow *fact mult*

$\text{FIRST}(fact\ mult) = \text{FIRST}(fact) = \{ (, id \}$
 $M[term, (] = term \rightarrow fact\ mult$
 $M[term, id] = term \rightarrow fact\ mult$

Construção da Tabela

- Considerando as produções:

<i>expr</i>		<i>term plus</i>
<i>plus</i>		<i>+ term plus</i>
		ϵ
<i>term</i>		<i>fact mult</i>
<i>mult</i>		<i>* fact mult</i>
		ϵ
<i>fact</i>		<i>(expr)</i>
		<i>id</i>

$\text{FIRST}(fact) = \{ (, id \}$	$\text{FOLLOW}(expr) = \{ \$,) \}$
$\text{FIRST}(term) = \{ (, id \}$	$\text{FOLLOW}(plus) = \{ \$,) \}$
$\text{FIRST}(expr) = \{ (, id \}$	$\text{FOLLOW}(term) = \{ +, \$,) \}$
$\text{FIRST}(plus) = \{ +, \epsilon \}$	$\text{FOLLOW}(mult) = \{ +, \$,) \}$
$\text{FIRST}(mult) = \{ *, \epsilon \}$	$\text{FOLLOW}(fact) = \{ *, +, \$,) \}$

mult \rightarrow * *fat mult*

$\text{FIRST}(* \text{ fat mult}) = \text{FIRST}(*) = \{ * \}$
 $M[mult, *] = mult \rightarrow * \text{ fat mult}$

Construção da Tabela

- Considerando as produções:

<i>expr</i>		<i>term plus</i>
<i>plus</i>		<i>+ term plus</i>
		ϵ
<i>term</i>		<i>fact mult</i>
<i>mult</i>		<i>* fact mult</i>
		ϵ
<i>fact</i>		<i>(expr)</i>
		<i>id</i>

$\text{FIRST}(fact) = \{ (, id \}$	$\text{FOLLOW}(expr) = \{ \$,) \}$
$\text{FIRST}(term) = \{ (, id \}$	$\text{FOLLOW}(plus) = \{ \$,) \}$
$\text{FIRST}(expr) = \{ (, id \}$	$\text{FOLLOW}(term) = \{ +, \$,) \}$
$\text{FIRST}(plus) = \{ +, \epsilon \}$	$\text{FOLLOW}(mult) = \{ +, \$,) \}$
$\text{FIRST}(mult) = \{ *, \epsilon \}$	$\text{FOLLOW}(fact) = \{ *, +, \$,) \}$
	$\}$

mult $\rightarrow \epsilon$

$\text{FOLLOW}(mult) = \{ +, \$,) \}$
 $M[mult, +] = mult \rightarrow \epsilon$
 $M[mult, \$] = mult \rightarrow \epsilon$
 $M[mult,)] = mult \rightarrow \epsilon$

Construção da Tabela

- Considerando as produções:

<i>expr</i>		<i>term plus</i>
<i>plus</i>		<i>+ term plus</i>
		ϵ
<i>term</i>		<i>fact mult</i>
<i>mult</i>		<i>* fact mult</i>
		ϵ
<i>fact</i>		<i>(expr)</i>
		<i>id</i>

$\text{FIRST}(fact) = \{ (, id \}$	$\text{FOLLOW}(expr) = \{ \$,) \}$
$\text{FIRST}(term) = \{ (, id \}$	$\text{FOLLOW}(plus) = \{ \$,) \}$
$\text{FIRST}(expr) = \{ (, id \}$	$\text{FOLLOW}(term) = \{ +, \$,) \}$
$\text{FIRST}(plus) = \{ +, \epsilon \}$	$\text{FOLLOW}(mult) = \{ +, \$,) \}$
$\text{FIRST}(mult) = \{ *, \epsilon \}$	$\text{FOLLOW}(fact) = \{ *, +, \$,) \}$

fact \rightarrow *(expr)*

$\text{FIRST}((expr)) = \{ (\}$
 $M[fact, (] = fact \rightarrow (expr)$

Construção da Tabela

- Considerando as produções:

<i>expr</i>		<i>term plus</i>
<i>plus</i>		<i>+ term plus</i>
		ϵ
<i>term</i>		<i>fact mult</i>
<i>mult</i>		<i>* fact mult</i>
		ϵ
<i>fact</i>		<i>(expr)</i>
		<i>id</i>

$\text{FIRST}(fact) = \{ (, id \}$
 $\text{FIRST}(term) = \{ (, id \}$
 $\text{FIRST}(expr) = \{ (, id \}$
 $\text{FIRST}(plus) = \{ +, \epsilon \}$
 $\text{FIRST}(mult) = \{ *, \epsilon \}$

$\text{FOLLOW}(expr) = \{ \$,) \}$
 $\text{FOLLOW}(plus) = \{ \$,) \}$
 $\text{FOLLOW}(term) = \{ +, \$,) \}$
 $\text{FOLLOW}(mult) = \{ +, \$,) \}$
 $\text{FOLLOW}(fact) = \{ *, +, \$,) \}$

fact \rightarrow *id*

$\text{FIRST}(id) = \{ id \}$
 $M[fact, id] = fact \rightarrow id$

Construção da Tabela

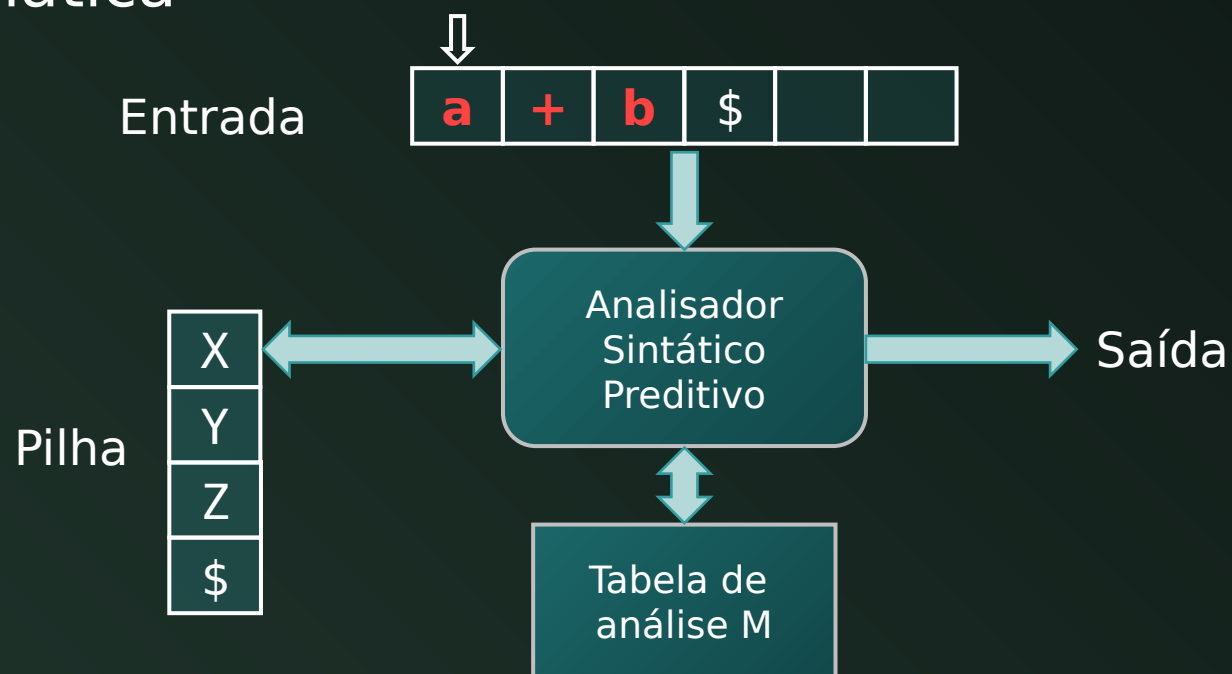
- Tabela de análise preditiva:

Não termina 	Símbolo de entrada					
	id	+	*	()	\$
<i>expr</i> →	<i>term plus</i>			<i>term plus</i>		
<i>plus</i> →		+ <i>term plus</i>			€	€
<i>term</i> →	<i>fact mult</i>			<i>fact mult</i>		
<i>mult</i> →		€	* <i>fact mult</i>		€	€
<i>fact</i> →	id			(<i>expr</i>)		

Entradas em branco na tabela representam estados de erro da análise

Analizador Preditivo

- Um **analizador sintático preditivo sem recursão** pode ser construído usando a tabela de análise e uma pilha de símbolos da gramática



X é o símbolo no topo da pilha e **a** é o símbolo corrente da entrada

Analizador Preditivo

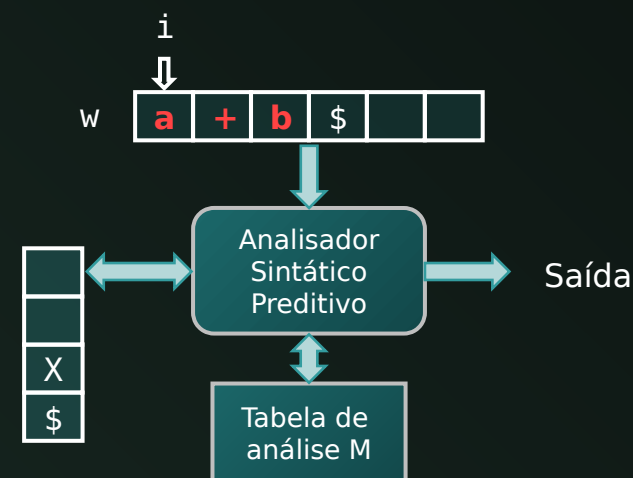
Algoritmo: Reconhecimento preditivo dirigido por tabela

Entrada: Uma cadeia w de entrada e uma tabela M

Saída: A derivação mais à esquerda de w , ou um erro

Método: Inicialmente o buffer contém $w\$$ e a pilha o símbolo inicial de G , acima de $\$$

```
define i para que aponte para o primeiro símbolo de w;  
define X como o símbolo no topo da pilha;  
while (X ≠ $) /* pilha não está vazia */  
{  
    if (X é a) desempilha e avança i;  
    else if (X é um terminal) erro();  
    else if (M[X,a] é uma entrada vazia) erro();  
    else if (M[X,a] =  $X \rightarrow Y_1 Y_2 \dots Y_k$ ) {  
        desempilha X e empilha  $Y_1 Y_2 \dots Y_k$ , com  $Y_1$  no topo;  
    }  
    define X como o símbolo no topo da pilha;  
}
```



Analizador Preditivo

- Derivando a cadeia

id + id * id

Não termin al	Símbolo de entrada					
	id	+	*	()	\$
<i>expr</i> →	<i>term plus</i>			<i>term plus</i>		
<i>plus</i> →		<i>+ term plus</i>			€	€
<i>term</i> →	<i>fact mult</i>			<i>fact mult</i>		
<i>mult</i> →		€	<i>* fact mult</i>		€	€
<i>fact</i> →	<i>id</i>			<i>(expr)</i>		

Pilha	Entrada	Ação
<i>expr</i> \$ id + id * id \$		
<i>term plus</i> \$ id + id * id \$		<i>expr</i> → <i>term plus</i>
<i>fact mult plus</i> \$ id + id * id \$		<i>term</i> → <i>fact mult</i>
id <i>mult plus</i> \$ id + id * id \$		<i>fact</i> → id
<i>mult plus</i> \$ + id * id \$		casou id
<i>plus</i> \$ + id * id \$		<i>mult</i> → €
+ <i>term plus</i> \$ + id * id \$		<i>plus</i> → + <i>term plus</i>
<i>term plus</i> \$ id * id \$		

Analizador Preditivo

- Derivando a cadeia

id + id * id

Não termin al	Símbolo de entrada					
	id	+	*	()	\$
<i>expr</i> →	<i>term plus</i>			<i>term plus</i>		
<i>plus</i> →		<i>+ term plus</i>			€	€
<i>term</i> →	<i>fact mult</i>			<i>fact mult</i>		
<i>mult</i> →		€	<i>* fact mult</i>		€	€
<i>fact</i> →	<i>id</i>			<i>(expr)</i>		

Pilha	Entrada	Ação
<i>fact mult plus</i> \$	<i>id * id</i> \$	<i>term</i> → <i>fact mult</i>
<i>id mult plus</i> \$	<i>id * id</i> \$	<i>fact</i> → <i>id</i>
<i>mult plus</i> \$	<i>* id</i> \$	casou <i>id</i>
<i>* fact mult plus</i> \$	<i>* id</i> \$	<i>mult</i> → <i>* fact mult</i>
<i>fact mult plus</i> \$	<i>id</i> \$	casou <i>*</i>
<i>id mult plus</i> \$	<i>id</i> \$	<i>fact</i> → <i>id</i>
<i>mult plus</i> \$	\$	casou <i>id</i>
<i>plus</i> \$	\$	<i>mult</i> → €
\$	\$	<i>plus</i> → €

Detecção de Erros

- O erro é detectado no reconhecimento preditivo quando:
 - O terminal no topo da pilha não casa com o próximo símbolo da entrada
 - Tem-se um não-terminal A no topo da pilha e um símbolo **a** na entrada, mas a posição M[A,**a**] na tabela está vazia

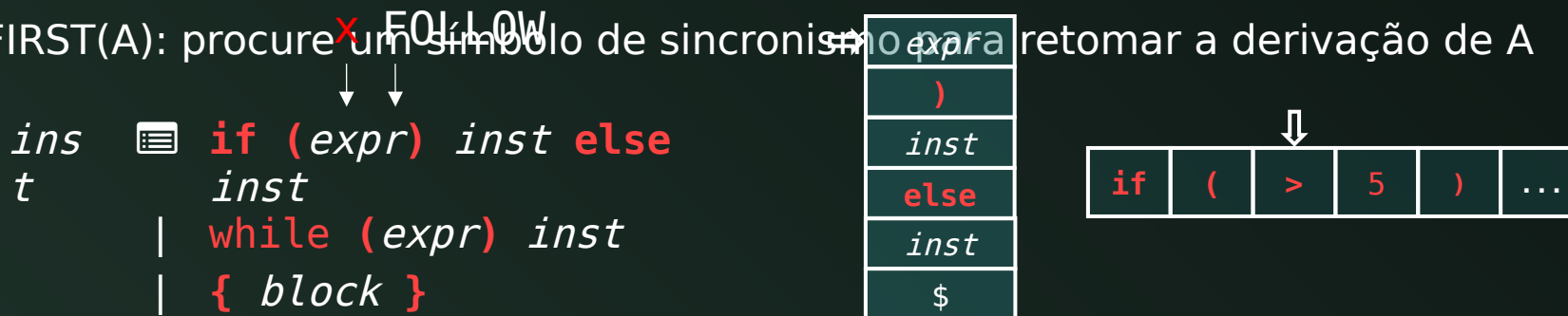
Pilha	Entrada	Ação
<i>expr</i> \$	+ id * id \$	

Não termin al	Símbolo de entrada					
	id	+	*	()	\$
<i>expr</i> →	<i>term plus</i>			<i>term plus</i>		

Recuperação de Erros

- A tabela e a pilha permitem aplicar facilmente duas **estratégias**:
 - **Modo pânico**: consiste em **ignorar símbolos** até encontrar um token do conjunto de **tokens de sincronismo**
 - Para um não-terminal A, os seguintes símbolos podem ser tokens de sincronismo:

- FOLLOW(A): em caso de erro, desempilhe A e ignore tokens até achar um de sincronismo
- FIRST(A): procure um símbolo de sincronismo **no expr** para retomar a derivação de A



Recuperação de Erros

- A tabela e a pilha permitem aplicar facilmente duas **estratégias**:
 - **Recuperação em nível de frase**: consiste em preencher as posições vazias da tabela com apontadores para **rotinas de erro**
 - As rotinas podem emitir mensagens de erro apropriadas
 - Elas podem também substituir, inserir ou excluir símbolos da entrada

Não termin al	Símbolo de entrada					
	id	+	*	()	\$
<i>expr</i> →	<i>term plus</i>	ERRO	ERRO	<i>term plus</i>	ERRO	ERRO
<i>plus</i> →	ERRO	+ <i>term plus</i>	ERRO	ERRO	ε	ε

Resumo

- A construção de um analisador sintático preditivo pode ser feita:
 - Com gramáticas LL(1)
 - Não ambíguas
 - Fatoradas
 - Sem recursão à esquerda
 - Usando funções recursivas ou uma tabela
- As funções FIRST e FOLLOW auxiliam
 - Na construção da tabela
 - Na detecção e recuperação de erros