Análise Sintática Preditiva Não Recursiva

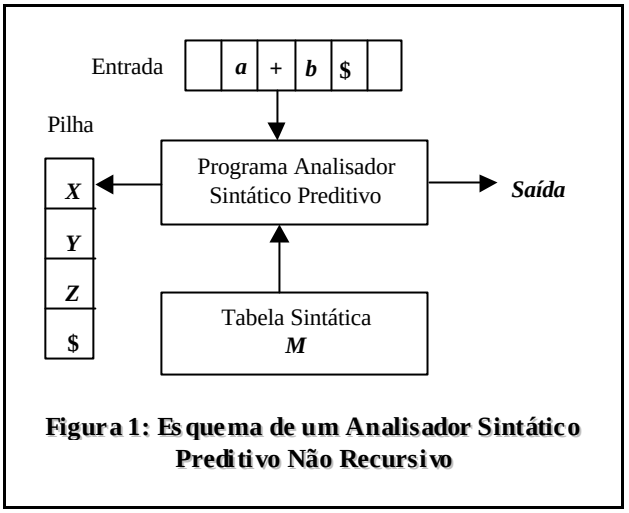

Figura 1: Esquema de um Analisador Sintático Preditivo Não Recursivo

É possível construir um *analisador sintático preditivo* não recursivo mantendo uma *pilha*, ao invés de chamadas recursivas, e consultando uma *tabela sintática* para a aplicação de uma produção a um dado *não-terminal*. A Figura 1 ilustra um *analisador sintático* deste tipo.

Pela figura podemos observar 5 componentes com tarefas distintas. A *entrada* é composta pela cadeia a ser analisada, finalizada pelo símbolo *$*, o qual representa o fim da cadeia. A *pilha* contém uma seqüência de símbolos gramaticais, com o símbolo *$* indicando o seu fundo. Inicialmente, a pilha contém o símbolo de partida da gramática sobre o topo da pilha, isto é, acima de *$*. A *tabela sintática*, aqui representada por *M*, é um array bidimensional *M*[*A*, *a*], onde *A* é um *não terminal* e *a* é um *terminal* ou o símbolo *$*. O conteúdo de *M*[*A*, *a*] é uma produção a ser aplicada quando *A* está sobre o topo da pilha e *a* é o símbolo de entrada, ou uma condição de erro. Já o *Programa* funciona conforme indicado a seguir.

Seja *X* o símbolo no topo da *pilha* e *a* o símbolo presente na entrada. O *Programa* deve seguir os seguintes passos:

1. Se $X = a = \$$, o programa para e anuncia o término, com sucesso, da análise sintática;
2. Se $X = a \neq \$$, o programa remove *X* da *pilha* e avança o ponteiro da entrada para o próximo símbolo;
3. Se *X* é um não terminal, o programa consulta a entrada *M*[*X*, *a*] da *tabela sintática*. Essa entrada será uma produção para *X*, da gramática, ou uma entrada de *erro*. Se por exemplo, $M[X, a] = \{X \rightarrow UVW\}$, o programa substitui *X*, no topo da *pilha*, por *WVU*, com *U* ao topo. Como saída o programa executa a ação definida pela produção aplicada. No nosso caso, iremos assumir que será impresso a produção utilizada. Se *M*[*X*, *a*] = ***erro***, o programa chama uma rotina de recuperação de erro.

Em termos gerais, podemos definir um algoritmo para um *analisador sintático não recursivo preditivo*, conforme se segue, tendo como entrada uma cadeia *w* a ser analisada e uma tabela sintática *M*, e como saída uma derivação mais a esquerda de *w*, se *w* estiver em *L*(*G*), ou uma condição de erro, caso contrário.

```
Colocar $S na pilha, com S, o símbolo de partida, ao topo
Fazer w$ a entrada
Faça ip apontar para o primeiro símbolo de w$
Repetir
   Seja X o símbolo ao topo da pilha
   Seja a o símbolo apontado por ip
   Se X for um terminal ou $, então
      Se X = a, então
         Remover X da pilha e avançar ip
      Senão
         Erro()
      Fim Se
   Senão
      Se M[X, a] = X → Y1Y2...Yn, então
         Remover X da pilha
         Empilhar YnYn-1... Y1, com Y1 ao topo da pilha
         Escrever a produção X → Y1Y2 ... Yn
      Senão
         Erro()
      Fim Se
   Fim Se
Até que X = $
```

Para exemplificar o funcionamento do algoritmo, considere abaixo, já eliminada a recursão a esquerda e fatorada a esquerda.

$$E \rightarrow TE'$$
$$E' \rightarrow +TE' \mid \lambda$$
$$T \rightarrow FT'$$
$$T' \rightarrow *FT' \mid \lambda$$
$$F \rightarrow (E) \mid \mathbf{id}$$

A *tabela sintática* para esta gramática é dada pela Figura 2. A entradas em branco são entradas de *erro*, enquanto que as demais indicam uma produção com a qual se deve expandir o *não terminal* ao topo da *pilha*.

Não Terminal	Símbolos de Entrada					
	id	+	*	(	)	$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow l$	$E' \rightarrow l$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow l$	$T' \rightarrow *FT'$		$T' \rightarrow l$	$T' \rightarrow l$
F	$F \rightarrow \mathbf{id}$			$F \rightarrow (E)$		

Figura 2: Tabela Sintática *M* para Expressões Matemáticas Contendo Apenas Adição e Multiplicação.

Suponha que a entrada contenha a sentença ***id + id * id***. Para esta entrada, o analisador preditivo realizará a seqüência de movimentos descritos pela figura.

Pilha	Entrada	Saída
$E	***id + id * id $***	
$E'T	***id + id * id $***	*E → TE'*
$E'T'F	***id + id * id $***	*T → FT'*
*$E'T'**id***	***id + id * id $***	*F → **id***
$E'T'	***+ id * id $***	
$E'	***+ id * id $***	*T' → l*
$E'T+	***+ id * id $***	*E' → +TE'*
$E'T	***id * id $***	
$E'T'F	***id * id $***	*T → FT'*
*$E'T'**id***	***id * id $***	*F → **id***
$E'T'	**** id $***	
*$E'T'F**	**** id $***	*T' → *FT'*
$E'T'F	***id $***	
*$E'T'**id***	***id $***	*F → **id***
$E'T'	***$***	
$E'	***$***	*T' → l*
$	***$***	*E' → l*

Figura 3: Movimentos feitos pelo Analisador Preditivo Não Recursivo para a entrada *id + id * id*.

Primeiro e Seguinte

A construção de analisadores sintáticos preditivos não recursivos é auxiliada por duas funções associadas à gramática *G*. Estas funções, denominadas de ***Primeiro*** (*First*) e ***Seguinte*** (*Follow*), nos permitem preencher as entradas de uma tabela sintática preditiva para *G*, sempre que possível. Os conjuntos de *tokens* produzidos pela função ***Seguinte*** podem também serem usados como *tokens* de sincronização durante a recuperação de erros.

Seja α qualquer cadeia de símbolos gramaticais e *A* um não terminal que ocorre em alguma forma sentencial de *G*. Podemos definir intuitivamente *Primeiro*(α) como sendo o conjunto de todos os terminais que começam as cadeias derivadas a partir de α, e *Seguinte*(*A*) como sendo o conjunto de terminais *a* que podem figurar imediatamente à direita de *A* em alguma forma sentencial, isto é, $S \Rightarrow^* \alpha Aab$, para algum α e *b*.

De uma maneira mais precisa, podemos calcular *Primeiro*(*X*) para todos os símbolos gramaticais *X*, aplicando repetidamente as regras abaixo até que nenhum terminal ou λ possa ser adicionado a qualquer conjunto *Primeiro*.

1. Se X for um *terminal*, então *Primeiro*(X) é $\{X\}$;
2. Se $X \rightarrow l$, for uma produção, então adicionar l a *Primeiro*(X);
3. Se X for um *não terminal* e $X \rightarrow Y_1Y_2 ...Y_n$ uma produção, então colocar a em *Primeiro*(X) se, para algum i, a estiver em *Primeiro*(Y_i) e l estiver em todos *Primeiro*(Y_1), ..., *Primeiro*(Y_{i-1}); isto é, se $Y_1 ... Y_{i-1} \Rightarrow^* \lambda$;
4. Se X for um *não terminal*, $X \rightarrow Y_1Y_2 ...Y_n$ uma produção, e l estiver em todos *Primeiro*(Y_i), $i = 1, 2, ..., n$, então adicionar l a *Primeiro*(X);
5. Se X for uma cadeia do tipo $X_1X_2 ...X_n$, adicionar a *Primeiro*($X_1X_2 ...X_n$) todos os símbolos de *Primeiro*(X_1) – $\{l\}$. Se $l \in$ *Primeiro*(X_1), então adicionar todos os símbolos de *Primeiro*(X_2) – $\{l\}$ a *Primeiro*($X_1X_2 ...X_n$). Se $l \in$ *Primeiro*(X_2), então adicionar todos os símbolos de *Primeiro*(X_3) – $\{l\}$ a *Primeiro*($X_1X_2 ...X_n$), e assim sucessivamente. Finalmente, adicionar l a *Primeiro*($X_1X_2 ...X_n$), se e somente se, para todo i, $i = 1, 2, ..., n$, $l \in$ *Primeiro*(X_i).

Do mesmo modo, podemos calcular *Seguinte*(X) para todos os não terminais X, aplicando repetidamente as regras abaixo até que nada mais possa ser adicionado a qualquer conjunto *Seguinte*.

1. Colocar $ em *Seguinte*(S), onde S é o símboloe de partida da gramática e $ o marcador de fim da entrada;
2. Se existir uma produção $X \rightarrow aAb$, colocar em *Seguinte*(A), tudo em *Primeiro*(b), exceto l;
3. Se existir uma produção $X \rightarrow aA$ ou uma produção $X \rightarrow aAb$, onde $l \in$ *Primeiro*(b), isto é, $b \Rightarrow^* \lambda$, então, adcionar tudo de *Seguinte*(X) em *Seguinte*(A).

Como exemplo, considere as produções abaixo:

$E \rightarrow TE'$
$E' \rightarrow +TE' \mid \lambda$
$T \rightarrow FT'$
$T' \rightarrow *FT' \mid \lambda$
$F \rightarrow (E) \mid id$

Os conjuntos *Primeiro* e *Seguinte* para estas produções encontram-se listados a seguir:

Primeiro(E) = *Primeiro*(T) = *Primeiro*(F) = *Primeiro*(TE') { **(** , **id** }
Primeiro(E') = *Primeiro*($+TE'$) = { **+** , l }
Primeiro(T') = *Primeiro*($*FT'$) ={ ***** , l }

Seguinte(E) = *Seguinte* (E') = { **)** , **$** }
Seguinte(T) = *Seguinte* (T') = { **+** , **)** , **$** }
Seguinte(F) = *Seguinte* (E') = { **+** , ***** , **)** , **$** }

Tabelas Sintáticas Preditivas

Uma *tabela sintática M* para uma gramática *G* pode ser construída por intermédio do algoritmo a seguir.

1. Para cada produção $X \rightarrow a$ da gramática, execute os passos 2 , 3 e 4;
2. Para cada terminal *a* em *Primeiro*(*a*), adicione $X \rightarrow a$ a *M*[*X*, *a*];
3. Se $l \in Primeiro(a)$, então adicione $X \rightarrow a$ a *M*[*X*, *b*] para cada terminal $b \in Seguinte(X)$;
4. Se $l \in Primeiro(a)$ e $\$ \in Seguinte(X)$, então adicione $X \rightarrow a$ a *M*[*X*, $];
5. Faça cada entrada indefinida de *M* ser ***erro***.

Aplicando-se o algoritmo acima à gramática da seção anterior obtemos a *tabela síntática* da Figura 2, pois:

1. Como para **E** → **TE'**, *Primeiro*(*E*) = *Primeiro*(*TE´*) = {**(**,**id**}, então *M*[*E*, **(**] = *M*[*E*, **id**] = **E** → **TE'**;
2. Como para **E´** → **+TE'** | λ, *Primeiro*(*E´*) = *Primeiro*(+*TE´*) = {**+**, l} e $l \in Primeiro(+TE´)$, então *M*[*E´*, **+**] = **E´** → **+TE'** e *M*[*E´*, **)**] = *M*[*E´*, **$**] = **E´** → *l*, pois *Seguinte*(*E´*) = {**)**, $}; e assim sucessivamente.

Gramáticas LL(1)

Suponha a gramática do exemplo anterior. Para esta gramática, podemos implementar um *analisador sintático descendente preditivo*, analisando apenas um (*1*) único símbolo da entrada (*lookahead*), para tomarmos as decisões sintáticas. Tal símbolo deve ser analisado a partir da *esquerda para a direita* (*Left to rigth*), produzindo uma *derivação linear mais a esquerda* (*left linear*). Portanto, dizemos que esta gramática é *LL*(*1*).

Gramáticas *LL*(1) possem várias propriedades distintas. Nenhuma gramática ambígua ou recursiva à esquerda pode ser *LL*(*1*). Pode ser também demonstrado que uma gramática é *LL*(*1*) se, e somente se, sempre que $A \rightarrow a \mid b$ forem duas produções distintas de *G*, e vigorarem as seguintes condições:

1. *a* e *b* não derivam, ao mesmo tempo, cadeias começando pelo mesmo terminal *a*, qualquer que seja *a*;
2. No máximo um dos dois, *a* ou *b*, derivam λ;
3. Se $b \Rightarrow^{*} \lambda$, então *a* não deriva qualquer cadeia começando por um terminal em *Seguinte*(*A*);

Por exemplo, a gramática a seguir não é *LL*(*1*), pois ela é ambígua. Isto é, ela não atende à terceira condição citada acima.

$S \rightarrow \mathbf{i}\ E\ \mathbf{t}\ S\ S' \mid \mathbf{a}$
$S \rightarrow \mathbf{e}\ S' \mid l$
$E \rightarrow \mathbf{b}$

Gramáticas *recursivas à esquerda* ou *ambíguas* não são *LL*(*1*), pois pelo menos uma de suas entradas na *tabela sintática* são *multiplamente definidas*, conforme pode ser visto pela Figura 4. Nela, podemos observar que a entrada para *M*[*S´*, *e*] contém tanto $S \rightarrow \mathbf{e}\ S'$ como $S \rightarrow l$, uma vez que *Seguinte*(*S´*) = { **e, $** }.

Não Terminal	Símbolos de Entrada					
	a	***b***	***e***	***i***	***t***	***$***
S	$S \rightarrow a$			$S \rightarrow iEtSS'$		
S'			$S' \rightarrow l$ $S' \rightarrow eS$			$S' \rightarrow l$
E		$E \rightarrow b$				

Figura 4: Tabela Sintática *M* para a Gramática *S* ® *iEtSS´* | *a*, *S'* ® **eS | *l*, *E* ® *b*.**

Poderiamos tentar transformar esta gramática para *LL*(*1*) eliminando a *recursão à esquerda* e *fatorando à esquerda*. Entretanto, existem algumas gramáticas para as quais nenhuma alteração irá produzir uma gramática *LL*(*1*), como é o caso do nosso exemplo.

Exercícios

1. Determine o conjunto *Primeiro* e *Seguinte* para a gramática do exemplo anterior.
2. Determine o conjunto *Primeiro* e *Seguinte* para a gramática a seguir, construa a sua tabela sintática e indique se ela é *LL(1)* e o porquê.

   $S \rightarrow aSbS \mid bSaS \mid \lambda$

3. Determine o conjunto *Primeiro* e *Seguinte* para a gramática a seguir, construa a sua tabela sintática e indique se ela é *LL(1)* e o porquê.

   $S \rightarrow AaAb \mid BbBa$
   $A \rightarrow \lambda$
   $B \rightarrow \lambda$

4. Determine o conjunto *Primeiro* e *Seguinte* para a gramática a seguir, construa a sua tabela sintática e indique se ela é *LL(1)* e o porquê.

   $S \rightarrow \mathbf{id}\ A\ \bullet$
   $A \rightarrow (\ B\ ) \mid \lambda$
   $B \rightarrow \mathbf{id}\ C$
   $C \rightarrow ,\ B \mid \lambda$

5. Determine o conjunto *Primeiro* e *Seguinte* para a gramática a seguir, construa a sua tabela sintática e indique se ela é *LL(1)* e o porquê.

   $S \rightarrow \mathbf{id}\ B$
   $B \rightarrow ,\ S \mid :\ C$
   $C \rightarrow \mathbf{int}\ D \mid \mathbf{real}\ D$
   $D \rightarrow ;\ S \mid l$