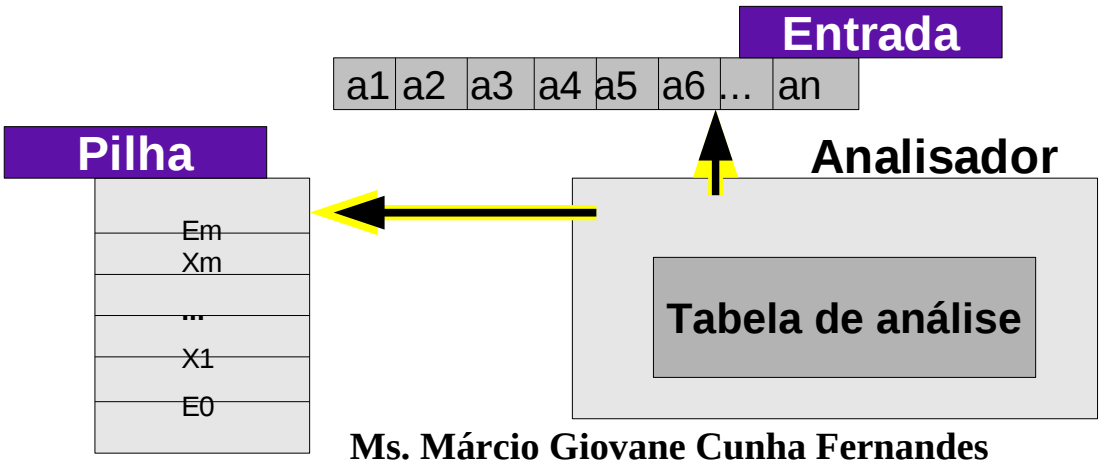
Manual para realizar a implementação de Autômato de pilha para Análise Sintática Bottom-up – SLR



Ms. Márcio Giovane Cunha Fernandes

Ms. Deborah Silva Alves Fernandes

UFG - 2011

Sumário

Autômato de pilha para Análise Sintática Bottom-up - LR

1 . Introdução

Autômato de pilha, também conhecido como autômato com memória, é uma categoria de autômato preparada a implementação da máquina de estados construída a partir de uma gramática livre de contexto. A estrutura do autômato é a mesma utilizada a implementação de linguagens regulares, mas a característica recursiva presente na gramática livre de contexto[1] requer a utilização da estrutura de dados pilha[2] para auxílio no processamento da máquina de estados.

Assim, neste texto apresentamos alguns conceitos necessários a compreensão do uso da pilha[2], bem como as teorias, técnicas e algoritmos utilizados na implementação do autômato[3].

2 . Recursividade

A teoria de recursividade é resumida em dois conceitos de solução de problemas[4]. O primeiro e conhecida como solução trivial. Esta não requer nenhuma forma de processo ou pensamento elaborado. O resultado e facilmente identificado. O exemplo mais simples e o valor do fatorial de zero ou um. E sabido que 0!=1 e que 1!=1 . Porem, o valor do fatorial de cem (100!) não e um antigo conhecido de todos. Exige-se esforço para conhecer o valor da solução. Eis, então, o segundo conceito de solução, cujo nome e solução geral ou solução não-trivial. Esta solução aplica a definição do problema para resolver o próprio problema.
Seja a definição do cálculo do fatorial de um numero inteiro:

$$n!=n*(n-1)! \quad (1)$$
$$0!=1 \text{ e } 1!=1 \quad (2)$$

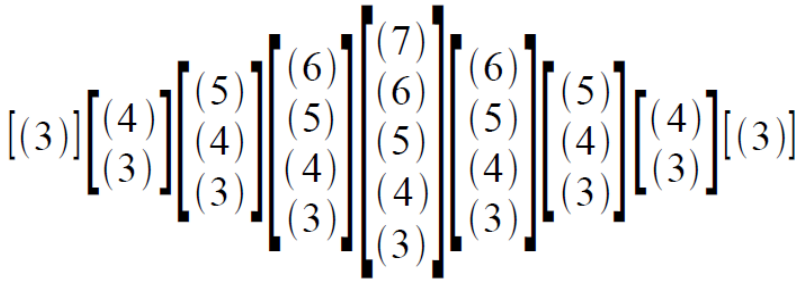
O cálculo de 5! e:

$$5!=5*(4)! \quad (3)$$
$$4!=4*(3)! \quad (4)$$
$$3!=3*(2)! \quad (5)$$
$$2!=2*(1)! \quad (6)$$
$$1!=1 \quad (7)$$

Nas equações (3), (4), (5) e (6) foi aplicado o conceito de solução geral definido em (1). Na equação (7) foi aplicado o conceito de solução trivial definido em (2). Mas ainda não sabemos o valor de 5! . Para conhece-lo, basta substituir o valor da solução (7) em (6), depois o valor de (6) em (5) e assim sucessivamente.

E interessante observar que aplicamos a solução geral recursivamente, e que esta persegue a convergência de valores a solução trivial, o ponto de parada na aplicação da recursividade. Vale observar que a convergência não significa comportamento decrescente no valor das soluções gerais. Em seguida iniciamos o caminho de volta, substituindo passo a passo as soluções ate alcançar a primeira solução geral. Computacionalmente, e utilizada a pilha[2] como artefato de memorização do caminho percorrido, exemplificado em (8).

Para implementações recursivas, o sistema operacional utiliza sua pilha para o controle do processo. Em implementações iterativas (utilizando estrutura de repetição) e necessária a aplicação explicita de tal estrutura de dados.



(8)

A recursividade, dentre outras aplicações, e também utilizada para caminhamento (pré ordem, pós-ordem, etc) na estrutura de dados arvore. Isto ocorre porque ela também possui definição recursiva[2].

A gramática livre de contexto e definida recursivamente[1]. E utilizada para reconhecimento de padrões que podem ser reconhecidos somente apos a leitura de toda a entrada. Um exemplo e o padrão

$$P=\{w \mid w = 0_n 1_n, \text{ onde } n > 0 \}\ . \qquad (9)$$

Uma gramática para este padrão é dada por:

$$S \rightarrow 01$$
$$S \rightarrow 0S1 \qquad (10)$$

Não há outra maneira de identificar a palavra 000111, senão pela exploração recursiva, aplicando as produções da gramática (10), explorando todos os símbolos dela.

$$S \Rightarrow 0\ S\ 1 \Rightarrow 00\ S\ 11 \Rightarrow 000111 \qquad (11)$$

ou

$$00\ 01\ 11 \Rightarrow 00S11 \Rightarrow 0S1 \Rightarrow S \qquad (12).$$

Ao reconhecimento de uma palavra deste padrão aplicamos a técnica de derivação (11), substituição (12) ou graficamente representada por uma estrutura conhecida como arvore sintática ou arvore de derivação, o que reforça o uso da estrutura de dados pilha na implementação de uma maquina de estados para o reconhecimento de padrões descritos através da gramática livre de contexto.

3 . Implementação do autômato de pilha

A implementação do autômato de pilha e composta por cinco passos:

1. Criar a gramática aumentada;
2. Enumerar a gramática;
3. Construir o autômato;
4. Construir o conjunto primeiro e seguinte(sequencial);
5. Construir a tabela sintática;
6. Algoritmo de interpretação da tabela sintática.

3.1 Gramática aumentada

Aumentar a gramática significa acrescentar produções a ela, com o objetivo de retirar o símbolo de partida do lado direito das produções.

Seja a gramática:

$E \rightarrow E+T$

$E \rightarrow T$

T →T∗F
T →F
F→i. (13)

O símbolo de partida é E. Ele aparece no lado direito em duas produções de (13). Em (14), o acréscimo de outra produção possibilitou eleger outro símbolo de partida, o E'. Criamos uma produção com um novo não-terminal a esquerda e o lado direito fazendo referencia ao antigo símbolo de partida.

Exemplo:

E' →E
E→ E+T
E→T
T →T∗F
T →F
F→i. (14)

3.2 Enumeração da gramática

É o ato de enumerar cada produção da gramática.

Exemplo:

1. E' →E
2. E→ E+T
3. E→T
4. T →T∗F
5. T →F
6. F→i (15)

3.3 Construir o autômato

Há duas técnicas para a construção do autômato. Utilizamos a técnica dos itens pontilhados e fechamento para tal tarefa.

Um item pontilhado e uma produção com um ponto(.) em alguma posição do lado direto da produção. A mesma produção pode gerar mais de um item pontilhado (a mesma produção possui instancias com o ponto aparecendo em posições diferentes). Os itens (16) e (17) são diferentes itens pontilhados criados a partir da mesma produção.

T →.T∗F (16)
T →T∗. F (17)

O ponto indica o que já foi utilizado de uma produção no reconhecimento de um padrão e aquilo que ela ainda e capaz de reconhecer. Em (16) a produção não reconheceu nada e e capaz de reconhecer o padrão T*F. Em (17) a produção já reconheceu parte do padrão – T* – e e capaz de reconhecer F.

Itens pontilhados com ponto ao final da produção são chamados de itens de redução. Outras posições do ponto na produção são conhecidos como itens de deslocamento. Todo estado do autômato que possui pelo menos um item de redução e considerado estado final ou de aceitação.

A técnica de fechamento trabalha a capacidade de derivação dos símbolos não-terminais do lado direito das produções. O fechamento no símbolo não-terminal E – f(E) – e o conjunto das produções

E→ E+T
E→T . (18)

Estas produções ainda possuem o não-terminal T que deve sofrer o fechamento – f(T). O resultado e o conjunto das produções:

E→ E+T
E→T
T →T∗F
T →F (19)

que por sua vez ainda possuem o não-terminal F, que também deve sofre f(F), resultando em

E→ E+T
E→T
T →T∗F
T →F
F→i (20)

Há um porém em todo esse processo. Ele ocorrera sobre os itens pontilhados. Assim, somente os não-terminais do lado direito dos itens, com o ponto imediatamente a sua esquerda, sofrerão o fechamento.
As regras de construção são apresentadas a seguir:

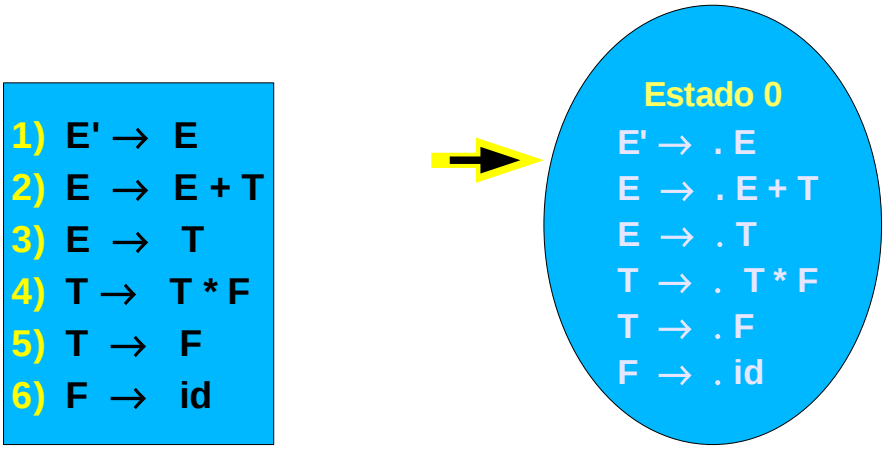
1. O estado inicial (zero) e construído com o fechamento sobre o item que possui símbolo de partida. No caso f($E' \rightarrow . E$). O resultado e o conjunto

$E' \rightarrow . E$
$E \rightarrow . E+T$
$E \rightarrow .T$
$T \rightarrow .T*F$
$T \rightarrow .F$
$F \rightarrow . i$

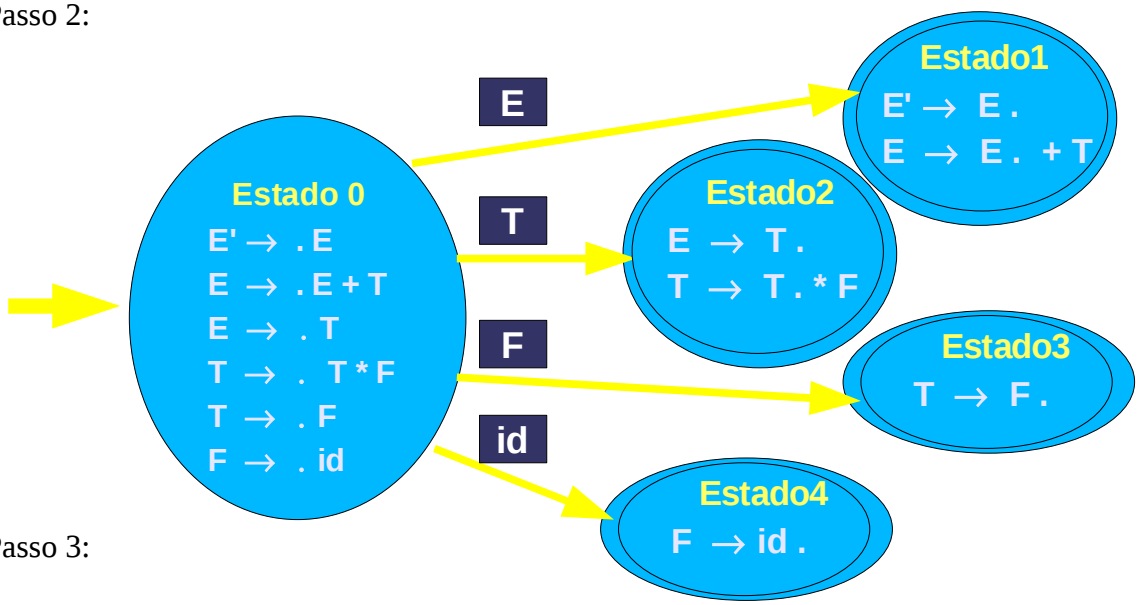
2. Para cada símbolo com o ponto imediatamente a esquerda, realizar-se-a uma transição, criando um novo estado. O ponto deve ser deslocado sobre o símbolo da transição; o procedimento de fechamento deve ser realizado neste estado. Este procedimento sera executado para todos os estados, nesta ordem, ate que não haja a criação de nenhum estado novo.

Exemplo:

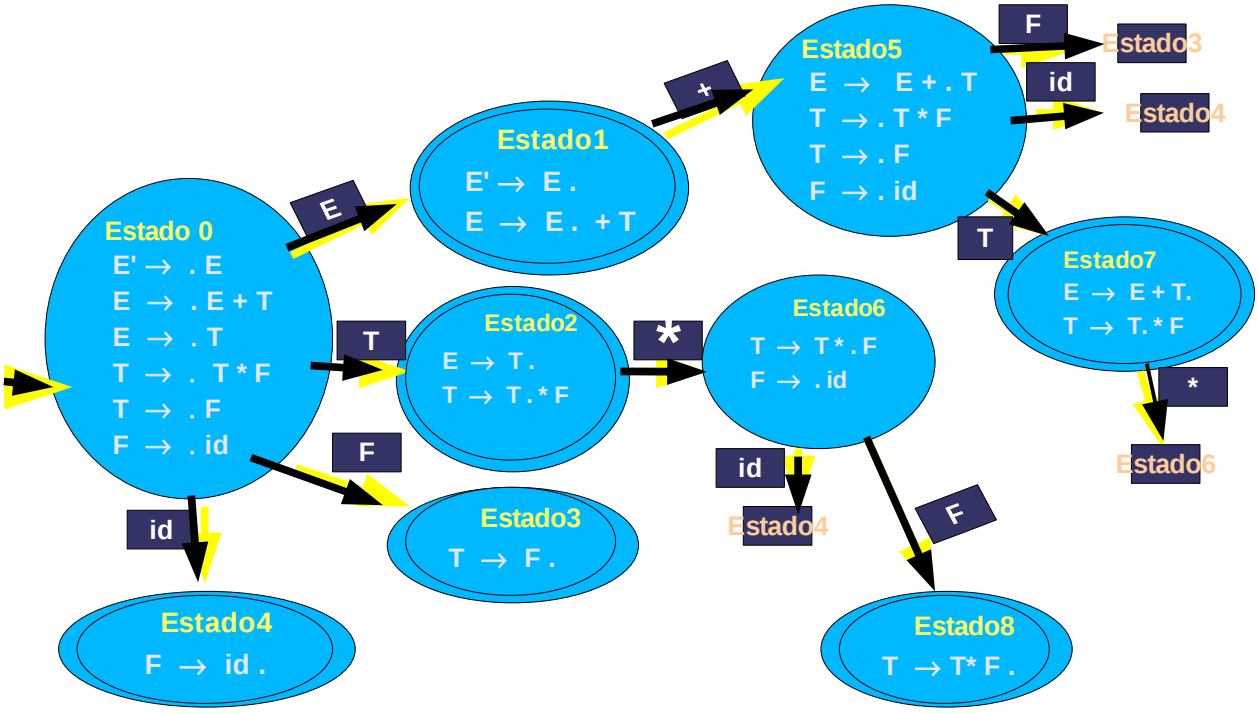
Passo 1:



Passo 2:



Passo 3:



3.4 Conjuntos primeiro e seguinte(seqüencial)

A construção do conjunto dos primeiros e importante para a construção do conjunto dos seguintes. O conjunto seguinte constitui uma informação importante para a navegação no autômato. Esta atividade possui característica recursiva. Há transições entre os estados (solução geral) em busca de uma solução trivial. Quando esta e alcançada, a pilha e utilizada para lembrar o caminho de volta, resolvendo as soluções gerais. Caso necessário, outro caminho de estados e percorrido em busca de outra solução trivial, ate que toda a entrada seja processada. As informações do conjunto seguinte ajudarão na identificação da solução trivial durante o processo descrito.
O conjunto dos primeiros e o conjunto de todos os terminais que começam as produções da gramática, inclusive todas as derivações. A seguir, exploramos as situações possíveis no cálculo deste conjunto.

Seja a gramática:

A→BC
B→d
C→e, (21)

o conjunto primeiro para a produção A e $P(A)=\{d\}$. De forma direta, a produção A não possui terminal, mas há a derivação $B \Rightarrow d$ que transforma $A \Rightarrow BC \Rightarrow dC$, fazendo *d* o terminal que começa a produção A.

Seja a gramática:

A→BC
A→ f
B→d
C→e, (22)

temos $P(A)=\{d , f\}$. Neste caso, de forma direta a produção A também começa com o terminal *f*.
Seja a gramática:

$A \rightarrow BC$
$A \rightarrow f$
$B \rightarrow d$
$B \rightarrow G$
$C \rightarrow e$
$G \rightarrow k$, (23)

temos $P(A)=\{d , f , k\}$. Ocorrem as derivações $A \Rightarrow BC \Rightarrow GC \Rightarrow kC$.
Seja a gramática:

$A \rightarrow BC$
$A \rightarrow f$
$B \rightarrow d$
$B \rightarrow G$
$B \rightarrow \epsilon$
$C \rightarrow e$
$G \rightarrow k$, (24)

temos $P(A)=\{d , f , k , e\}$. Ocorrem as derivações $A \Rightarrow BC \Rightarrow \epsilon C \Leftrightarrow C \Rightarrow e$.

O conjunto seguinte e o conjunto de todos os terminais que aparecem ***após um símbolo não-terminal***. Assim, haverá um conjunto de terminais para cada não-terminal da gramática.

Para realizar o cálculo deste conjunto, nessecitamos conhecer o cálculo do conjunto primeiro, bem como alguns simbolismos para representação genérica de produções. As regras para tal representação são:

1. Qualquer caracter maiúsculo do início do alfabeto português representa um, e somente um, símbolo não-terminal;
2. Qualquer caracter minúsculo do início do alfabeto português representa um, e somente um, símbolo terminal;
3. Qualquer caracter minúsculo do fim do alfabeto português representa uma sentença. Uma sentença e uma sequência qualquer de símbolos terminais;
4. Qualquer caracter do alfabeto grego representa uma forma sentencial. Uma forma sentencial e um sequência qualquer de símbolos terminais e não-terminais;
5. O símbolo $ indica que o fim dos símbolos de uma entrada.

De posse deste conhecimento, vamos definir as regras ao cálculo do conjunto seguinte:

I - O conjunto seguinte do símbolo de partida da gramática contem $. Caso o símbolo de partida não apareça no lado direito de nenhuma produção da gramática, o conjunto sera unitário {$} - sera o caso das gramáticas aumentadas –, caso contrario, o conjunto poderá conter mais elementos. Assim, para a gramática (25) temos:

$S(E') = \{\$\}$;

II - Seja a produção $A \rightarrow \alpha B\beta$, o conjunto seguinte de B – S(B) – conterá o conjunto de primeiro de β – P(β) – se, e somente se, β não deriva zero ou mais vezes em palavra vazia ε – $\neg\, \beta \Rightarrow^{*} \epsilon$. Formalizando o exemplo:

Para $A \rightarrow \alpha B\beta$, $S(B) = P(\beta) \Leftrightarrow \neg\, \beta \Rightarrow^{*} \epsilon$;

III - Sejam as produções $A \rightarrow \alpha B\beta$ ou $A \rightarrow \alpha B$, o conjunto seguinte de B – S(B) – conterá o conjunto seguinte de A – S(A) – se, e somente se, β deriva em palavra vazia ε em zero ou mais derivações – $\beta \Rightarrow^{*} \epsilon$. Formalizando:

Para $A \rightarrow \alpha B\beta$ *ou* $A \rightarrow \alpha B$, $S(B) = S(A) \Leftrightarrow \beta \Rightarrow^{*} \epsilon$.

O conjunto seguinte e calculado para cada ocorrência dos símbolos não-terminais do lado direito de cada produção. Nas regras 2 e 3, apresentadas para o cálculo do seguinte, o genérico não-terminal *B* representa o não-terminal em destaque entre as formas sentenciais α e β para a produção em destaque.

Exemplo: Seja a gramática,

1. $E' \rightarrow E$
2. $E \rightarrow E + T$
3. $E \rightarrow T$
4. $T \rightarrow T * F$
5. $T \rightarrow F$
1. $F \rightarrow i$, (25)
2.

para o cálculo do seguintes de E' temos $S(E') = \{\$\}$. Aplicamos a regra 1.O cálculo dos seguintes para o não-terminal E será realizado nas produções 1 e 2 na gramática (25). Esta escolha foi determinada por causa da ocorrência desse símbolo no lado direito daquelas produções. Para a produção 1 temos:

$$\underset{A}{E'} \rightarrow \underset{\alpha}{} \underset{B}{E} \underset{\beta}{} \qquad (26)$$

o *B* representa o *E*, símbolo em destaque. O α representa todos os símbolos antes de E, no caso e conjunto vazio e o *β* representa todos os símbolos depois de *E*, no caso em zero derivações e vazio - ε . Assim, o conjunto seguinte de *E* para esta produção será calculado aplicando a terceira regra.

$$S(B) = S(A) \simeq S(E) = S(E') = \{\$\}, \qquad (27)$$

Para a produção 2 temos:

$$\underset{A}{E} \rightarrow \underset{\alpha}{} \underset{B}{E} + T \qquad (28)$$

o *B* representa o *E*, símbolo em destaque. O α representa todos os símbolos antes de *E*, no caso e conjunto vazio e o β representa todos os símbolos depois de *E*, no caso +T. Não há $\beta \Rightarrow^{*} \epsilon$. Assim, o conjunto seguinte de *E* para a produção 2 será calculado aplicando a segunda regra.

$$S(B)= P(\beta) \simeq S\,(E)= P(+T\,)= \{+\}\ . \qquad (29)$$

Como não há outras ocorrências de *E* em outras produções, finalizamos o cálculo do conjunto seguinte para esse símbolo realizando a união dos conjuntos obtidos em (27) e (29):

$$S\,(E)= S\,(E'\,) \cup P(+T\,)= \{\$\ ,+\}\ .$$

Para o cálculo do seguinte de T escolhemos as produções 2, 3 e 4. São as que apresentam o símbolo no lado direito.

$$\overset{\mathbf{A}\ \rightarrow\ \ \alpha\ \ \mathbf{B}\ \beta}{E \rightarrow E + T} \qquad (30)$$

O símbolo genérico *B* agora aparece alinhado com o *T*. O α representa E+ e $\beta \Rightarrow {}^{*}\epsilon$.
Aplicamos neste caso a terceira regra:

$$S\,(B)= S\,(\,A) \simeq S\,(T\,)= S\,(E)= \{\$\ ,+\}\ .$$

$$\overset{\mathbf{A}\ \rightarrow\ \ \ \alpha\ \mathbf{B}\ \beta}{E\ \rightarrow\ E + T} \qquad (31)$$

Nesse caso $\alpha \Rightarrow {}^{*}\epsilon$ e $\beta \Rightarrow {}^{*}\epsilon$. Aplicamos nesse caso a terceira regra:

$$S\,(B)= S\,(\,A) \simeq S\,(T\,)= S\,(E)= \{\$\ ,+\}\ .$$

$$\overset{\mathbf{A}\rightarrow\ \ \alpha\ \ \mathbf{B}\ \beta}{T \rightarrow\ \ \ T * F} \qquad (32)$$

Nesse caso $\alpha \Rightarrow {}^{*}\epsilon$ e $\neg\, \beta \Rightarrow {}^{*}\epsilon$. Aplicamos a segunda regra:

$$S\,(B)= P(\beta) \simeq S\,(T\,)= P(* F)= \{*\}\ .$$

Como não há outras ocorrências de *T* em outras produções, finalizamos o cálculo do conjunto seguinte para esse símbolo realizando a união dos conjuntos obtidos em (30), (31) e (32):

$$S\,(T\,) = S\,(E) \cup P(* F)= \{\$\ ,\ +,\ *\}\ .$$

Para o cálculo do seguinte de *F* escolhemos as produções 4 e 5. São as que apresentam o símbolo no lado direito.

$$\overset{\mathbf{A}\ \rightarrow\ \ \alpha\ \ \mathbf{B}\ \beta}{T\ \rightarrow T\ *\ F} \qquad (33)$$

O símbolo genérico *B* agora aparece alinhado com o *F*. O α representa T* e $\beta \Rightarrow {}^{*}\epsilon$.
Aplicamos neste caso a terceira regra: $S\,(B)= S\,(\,A) \simeq S\,(F)= S\,(T\,)= \{\$\ ,\ +,\ *\}\ .$

$$\overset{A\ \ \rightarrow\ \ \alpha\ \ B\ \beta}{T\ \ \rightarrow\ \ \ F} \qquad (34)$$

Nesse caso $\alpha \Rightarrow {}^{*}\epsilon$ e $\beta \Rightarrow {}^{*}\epsilon$. Aplicamos nesse caso a terceira regra:

$$S\,(B)= S\,(\,A) \simeq S\,(F)= S\,(T\,)= \{\$\ ,\ +,\ *\}\ .$$

Como não há outras ocorrências de *F* em outras produções, finalizamos o cálculo do conjunto seguinte para esse símbolo realizando a união dos conjuntos obtidos em (33) e (34):

$$S\,(F\,)= S\,(T\,) \cup S\,(T\,)= \{\$\ ,\ +,\ *\}\ .$$

Seja a gramática:

1. $T' \rightarrow T$

2. $T \rightarrow PP$
3. $P \rightarrow eP$
4. $P \rightarrow f$ (35)

- $S(T') = \{\$\}$ primeira regra;
- $S(T) = \{\$\}$.
  $A \rightarrow \alpha B \beta$
  $T' \rightarrow T$, terceira regra. $S(T) = S(T') = \{\$\}$;
- $S(P) = \{\$, e, f\}$.
  $A \rightarrow \alpha B \beta$
  $T \rightarrow P P$, segunda regra. $S(P) = P(P) = \{e, f\}$;

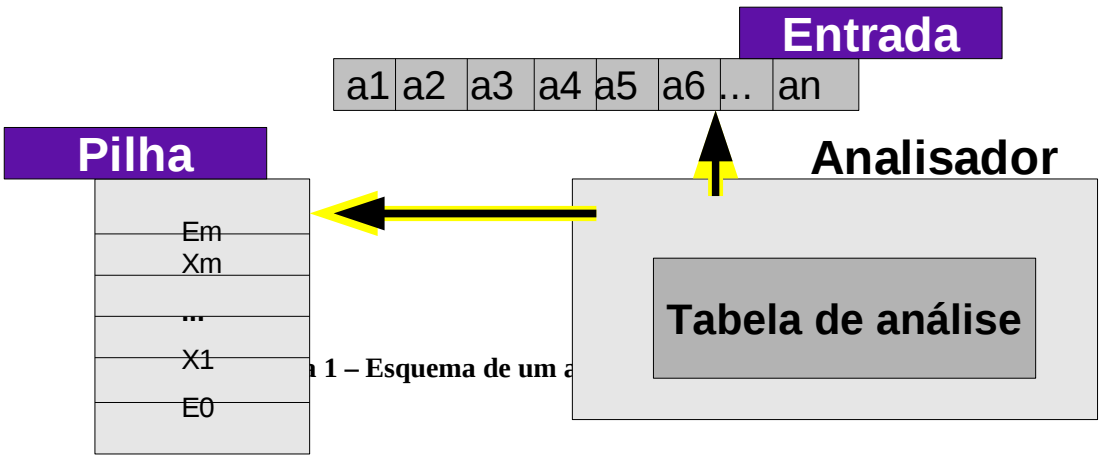
  $A \rightarrow \alpha B \beta$
  $T \rightarrow P P$, terceira regra $S(P) = S(T) = \{\$\}$;

  $A \alpha B \beta$
  $P \rightarrow e P$, terceira regra $S(P) = S(P)$;

  Assim, $P(P) \cup S(T) = \{\$, e, f\}$;

3.5 Construir a tabela sintática

A tabela sintática ou tabela de análise é uma estrutura que representa o autômato de uma gramática (Figura 1) com as informações de controle que direcionam o uso das soluções geral e trivial. Esse controle e definido a partir da combinação do autômato e do conjunto seguinte de cada item pontilhado.



1 – Esquema de um a

Dois conceitos são necessários: dada uma palavra – uma sentença –, a transição de um estado para outro no autômato por um símbolo (terminal) desta palavra e simbolizado pela letra *s* seguida de um numero, que representa o estado destino na transição. Essa letra foi obtida da palavra *stack* – pilha, indicando que devemos empilhar o símbolo e o estado –, ou *shift* – deslocar o símbolo e o estado para pilha –. O retorno no caminho realizado no autômato e representado pela letra *r* de *reduce* (redução), seguido por um numero, que representa a produção que sofre a redução. Acontece quando há um estado com item de redução $A \rightarrow \gamma$. e o terminal em análise esteja no conjunto seguinte do não-terminal A – S(A). As transições com símbolos não-terminais somente registram na tabela a combinação entre o estado origem e o estado destino, como na Figura 1.

A tabela possui os estados representados nas linhas e os símbolos as colunas (Figura 2) .

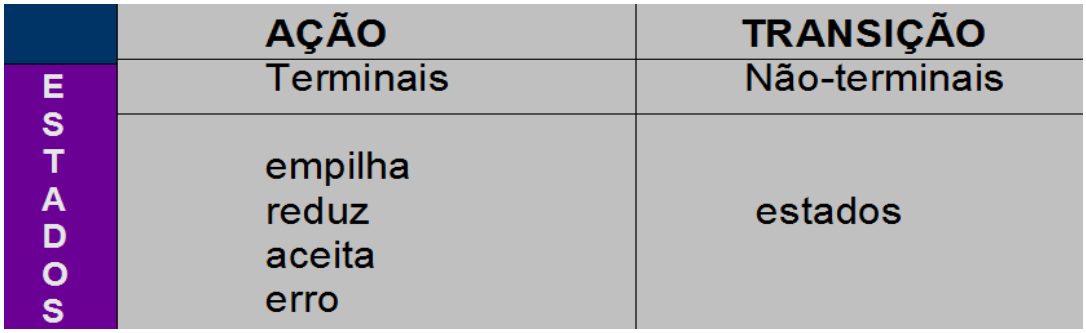


Figura 2 – Representação da tabela sintática.

Vejamos um exemplo de construção da primeira linha (utilizando a gramática exemplo). O único terminal *i* reconhecido pelo estado zero sofre transição ao estado 4. Há transições para os não-terminais E, T e F.

	i	+	*	$	E	T	F
0	S4				1	2	3

No estado 1, há um item de redução *E' →E.* . E um item especial, pois o não-terminal que gera a produção (E') e o símbolo de partida da gramática. Nesse caso, sera colocada uma mensagem de aceite (*acc*). Alcançado este estado, a palavra e aceita. Há também um item de deslocamento *E→ E.+T* , ocorrendo uma transição ao estado 5 com o terminal +.

	i	+	*	$	E	T	F
0	S4				1	2	3
1		s5		*acc*			

No estado 2, há um item de redução E→T. . Nos símbolos do conjunto seguinte *E* – S(E)={$, +} - haverá redução pela produção de número 3. Ao item de deslocamento T →T.* F , transição ao estado 6 com o terminal *.

	i	+	*	$	E	T	F
0	S4				1	2	3
1		s5		*acc*			
2		r3	s6	*r3*			

No estado 3, há somente um item de redução T →F. . Nos símbolos do conjunto seguinte T – S(T)={$, +, *} - haverá redução pela produção de número 5.

	i	+	*	$	E	T	F
0	S4				1	2	3
1		s5		*acc*			
2		r3	s6	*r3*			
3		r5	r5	*r5*			

No estado 4, há somente um item de redução *F →i.* . Nos símbolos do conjunto seguinte de F – S(F)={$, +, *} - haverá redução pela produção de número 6.

	i	+	*	$	E	T	F
0	S4				1	2	3
1		s5		*acc*			
2		r3	s6	*r3*			
3		r5	r5	*r5*			
4		r6	r6	*r6*			

No estado 5, há somente itens de deslocamento. Para os não-terminais há transições com T para o estado 7, com F para o estado 3 e empilhamento de *i* com transição para o estado 4.

	i	+	*	$	E	T	F
0	S4				1	2	3
1		s5		*acc*			
2		r3	s6	*r3*			
3		r5	r5	*r5*			
4		r6	r6	*r6*			
5	s4					7	3

No estado 6, há somente itens de deslocamento. Transição simples com F para o estado 8 e empilhamento de *i* com transição para o estado 4.

	i	+	*	$	E	T	F
0	S4				1	2	3
1		s5		*acc*			
2		r3	s6	*r3*			
3		r5	r5	*r5*			
4		r6	r6	*r6*			
5	s4					7	3
6	s4						8

No estado 7, há um item de redução e um item de deslocamento. Ao item de redução, nos terminais do conjunto seguinte de E – S(E)={$, +} - haverá redução pela produção 2. O item de deslocamento faz empilhamento de * com transição para o estado 6.

	i	+	*	$	E	T	F
0	S4				1	2	3
1		s5		*acc*			
2		r3	s6	*r3*			
3		r5	r5	*r5*			
4		r6	r6	*r6*			
5	s4					7	3
6	s4						8
7		r2	s6	*r2*			

No estado 8, há somente um item de redução. Nos elementos de S(T)={$, +, *} haverá redução pela produção de numero 4.

	i	+	*	$	E	T	F
0	S4				1	2	3
1		s5		*acc*			
2		r3	s6	*r3*			
3		r5	r5	*r5*			
4		r6	r6	*r6*			
5	s4					7	3
6	s4						8
7		r2	s6	*r2*			
8		r4	r4	*r4*			

3.6 Algoritmo para a interpretação da tabela sintática

O algoritmo que será discutido nesta seção gerencia o caminhamento no autômato de pilha, interpretando as orientações contidas na tabela.

Neste algoritmo, *ip* e um apontador que controla qual símbolo esta sendo referenciado. O símbolo *w*, como já apresentado, e a representação genérica de uma sentença. É a palavra que o algoritmo deve avaliar. O símbolo $ representa fim de entrada. *Repetir é* uma estrutura de repetição infinita.

A frase *"seja s o estado ao topo da pilha"* e interpretada da seguinte maneira: *s é* uma variável inteira. Ao passar por este comando, ela recebe o valor do estado que esta ao topo da pilha. Continuando a frase: *"e a o símbolo apontado por ip"*. *a* e uma generalização para representar um terminal. Assim, *a é* o terminal apontado por *ip* em algum momento.

A estrutura condicional e descrita da seguinte forma:

- *ação[s , a]*: ação representa o conjunto de colunas na tabela sintática que fazem referência

aos terminais. Possui esse nome, pois qualquer atividade com estes símbolos gera uma ação de empilhamento ou redução; novamente, o valor de *s* (estado ao topo da pilha) foi obtido no comando anterior e *a (*símbolo terminal) também o foi no comando anterior;

- *empilhar s'*: na tabela, o comando empilhar, como já foi explicado, e representado pela letra *s; s'* e outra variável que representa o numero que aparece logo apos ao *s* de empilhar.

```
Fazer ip apontar para o primeiro símbolo w$;
repetir para sempre início
        seja s o estado ao topo da pilha e
        a o símbolo apontado por ip;
        se ação[s,a] = empilhar s' então início
                        empilhar a e em seguida s' no topo da pilha;
                        avançar ip para o próximo símbolo da entrada;
        fim
        senão  se ação[s,a] = reduzir A → β  então início
                                desempilhar 2*|β | símbolos para fora da pilha;
                                seja s' o estado agora ao topo da pilha;
                                empilhar A e em seguida desvio[s',A];
                                escrever a produção  A → β na tela;
                fim
                senão se ação[s,a] = aceitar  então
                                        retornar;
                        senão erro();
fim
```

Figura 3 – Algoritmo para realização de análise sintática *bottom-up* utilizando autômato de pilha, adaptação de [2].

Caso a condição *ação[s,a] = empilhar* s' seja satisfeita, os comandos *"empilhar a e em seguida s' no topo da pilha"* e *"avançar ip para o próximo símbolo de entrada"* serão executados. O primeiro indica colocar os valores de *a* e *s'*, nesta ordem, no topo da pilha. O segundo, passar a referência de *ip* ao próximo símbolo em *w*.

- *reduzir A→β* : reduzir e representado na tabela pela letra *r*; *A→β* e a representação genérica de qualquer produção – a letra grega beta representa uma forma sentencial, sequência de terminais e não-terminais que estão do lado direito de uma produção. Na tabela é o número que aparece logo após a letra *r*, que e o número de uma produção;
- *desempilhar 2* | β | símbolos para fora da pilha*: é a ação de utilizar a pilha para retornar no caminho feito no autômato; Matematicamente | *β* | significa calcular numericamente a quantidade de símbolos em *β*; a multiplicação por dois acontece porque na estrutura da pilha é armazenada um símbolo e o estado da transição por este símbolo;
- *seja s' o estado agora ao topo da pilha*: *s'* e uma variável que receberá o valor numérico do estado que sobrou apos desempilhar *2* | β |*;
- *empilhar A e em seguida desvio[s', A]*: desvio representa as colunas com símbolos não-terminais. Esse nome indica que as colunas relacionadas não geram nenhuma acao na pilha, simplesmente indicam uma transição simples; assim, deve-se empilhar o *A*, o símbolo não-terminal do lado esquerdo da produção em questão e em seguida o estado calculado pelo desvio do estado *s'* e o não-terminal *A*;
- *escrever a produção A→β* : trata-se de apresentar a produção reduzida na saída padrão.

Será apresentado um exemplo de reconhecimento e outro de rejeição da sentença da entrada.

Exemplo: Aceitação da entrada.

Seja:

w: **i +i*i $**

A pilha:

Pilha: **0**

Passos de resolução:

Passo 1) w: **i +i*i $** *ip* Pilha: **0** – s=0, a=i; – ação = empilhar 4; – s'=4 – Empilha i e em seguida s' – Avançar ip	Passo 2) w: **i + i *i$** *ip* Pilha: **0 i 4** – s= 4 – ação = reduzir 6, – Produção 6: *F→ i* – Desempilhar 2*|i| Pilha: **0 F 3** – s'=0 – Empilhar F e em seguida olhar na tabela o desvio [0,F] = 3; – Empilhar o desvio 3.	Passo 3) w: **i + i *i$** *ip* Pilha: **0 F 3** – s= 3 – ação = reduzir 5, – Produção 5: *T→ F* – Desempilhar 2*|F| Pilha: **0 T 2** – s'=0 – Empilhar T e em seguida olhar na tabela o desvio [0,T] = 2 ; – Empilhar o desvio 2.
Passo 4) w: **i + i *i$** *ip* Pilha: **0 T 2** – s=2 – ação = reduzir 3, – Produção 3: *E→T* – Desempilhar 2*|T| Pilha: **0 E 1** – s'=0 – Empilhar E e em seguida o desvio [0,E] = 1 ;	Passo 5) w: **i + i *i$** *ip* Pilha: **0 E 1** – s=1, a=+; – ação = empilhar 5; – s'=5 – Empilha + e em seguida 5 – Avançar ip	Passo 6) w: **i + i *i$** *ip* Pilha: **0 E 1 + 5 i 4** – s=5 , a=i; – ação = empilhar 4; – s'=4 – Empilha i e em seguida 4 – Avançar ip
Passo 7) w: **i + i *i$** *ip* Pilha: **0 E 1 + 5 i 4** – s=4, a = * – ação = reduzir 6, – Produção 6: *F→ i* – Desempilhar 2*|i| Pilha: **0 E 1 + 5 F 3** – s'=5 – Empilhar F e em seguida o desvio [5,F] = 3 ;	Passo 8) w: **i + i *i$** *ip* Pilha: **0 E 1 + 5 F 3** – s=3, a = * – ação = reduzir 5, – Produção 5: *T→ F* – Desempilhar *2*|F|* Pilha: **0 E 1 + 5 T 7** – s'=5 – Empilhar T e em seguida o desvio [5,T] = 7 ;	Passo 9) w: **i + i *i$** *ip* Pilha: **0 E 1 + 5 T 7** – s=7 , a=*; – ação = empilhar 6; – s'=6 – Empilha * e em seguida 6 – Avançar ip
Passo 10) w: **i + i *i$** *ip* Pilha: **0 E 1 + 5 T 7 * 6** – s=6 , a=i; – ação = empilhar 4; – s'=4 – Empilha i e em seguida 4 – Avançar ip	Passo 11) w: **i + i *i$** *ip* Pilha: **0 E 1 + 5 T 7 * 6 i 4** – s=4, a = $ – ação = reduzir 6, – Produção 6: *F→ i* – Desempilhar *2*|i|* Pilha: **0 E 1 + 5 T 7 * 6 F 8** – s'=6 – Empilhar F e em seguida o desvio [6,F] = 8 ;	Passo 12) w: **i + i *i$** *ip* Pilha: **0 E 1 + 5 T 7 * 6 F 8** – s=8, a = $ – ação = reduzir 4; – Produção 4: *T →T∗F* – Desempilhar *2*|T∗F|* Pilha: **0 E 1 + 5 T 7** – s'=5 – Empilhar T e em seguida o desvio [5,T] = 7 ;

Passo 13) w: **i + i *i$** *ip* Pilha: **0 E 1 + 5 T 7** – s=7, a = $ – ação = reduzir 2; – Produção 2: $E \rightarrow E+T$ – Desempilhar $2*\lvert E+T \rvert$ Pilha: **0 E 1** – s'=0 Empilhar E e em seguida o desvio [0,E] = 1 ;	Passo 14) w: **i + i *i$** *ip* Pilha: **0 E 1** – s=1, a = $ – ação = **Aceitar;**

Construção da árvore de baixo para cima nos passos acima

Passo 1) i +	Passo 2) F | i +	Passo 3) T | F | i +
Passo 4) E | T | F | i +	Passo 5) E | T | F | i + i	Passo 6) E | T | F | i + i *
Passo 7) E | T | F F | | i + i *	Passo 8) E | T T | | F F | | i + i *	Passo 9) E | T T | | F F | | i + i * i
Passo 10) E | T T | | F F | | **i + i * i**	Passo 11) E | T T | | F F **F** | | | i + i * **i**	Passo 12) E T | / | \\ T T | | | | | | F F | F | | | | i + i * i
Passo 13) E / | \\ / | \\ E | T | | / | \\ T | T | | | | | | | F | F | F | | | | | i + i * i	Passo 14) E' | E / | \\ / | \\ E | T | | / | \\ T | T | | | | | | | F | F | F | | | | | i + i * i	

Exemplo: Não aceitação da entrada.
Seja:

w: **i i $**

A pilha:

Pilha: **0**

Passos de resolução:

Passo 1) w: **i i $** *ip* Pilha: **0** – s=0, a=i; – ação = empilhar 4; – s'=4 – Empilha i e em seguida s' – Avançar ip	Passo 2) w: **i i$** *ip* Pilha: **0 i 4** – s= 4, a = i; – ação = erro();

Construção da árvore de baixo para cima nos passos acima

Passo 1) i	Passo 2) i i

Referências Bibliográficas

[1] Hopcroft, John E.; Ullman, Jeffrey D.; Motwani, Rajeev. **Introdução à Teoria de Autômatos, Linguagens e Computação**. Ed. Campus, 2002.
[2] Aho, Alfred V.; Sethi, Ravi. **Compiladores – Princípios, Técnicas e Ferramentas**. Addison-Wesley – BR. 2ª edição.
[3] Gersting, Judith L. . **Fundamentos Matemáticos para a Ciência da Computação**. 3ª edição, Ed. LTC, 1995.
[4] Grune, Dick; Bal, Henri E.; **Projeto Moderno de Compiladores – Implementação e aplicações.** Ed. Campus, 2001.