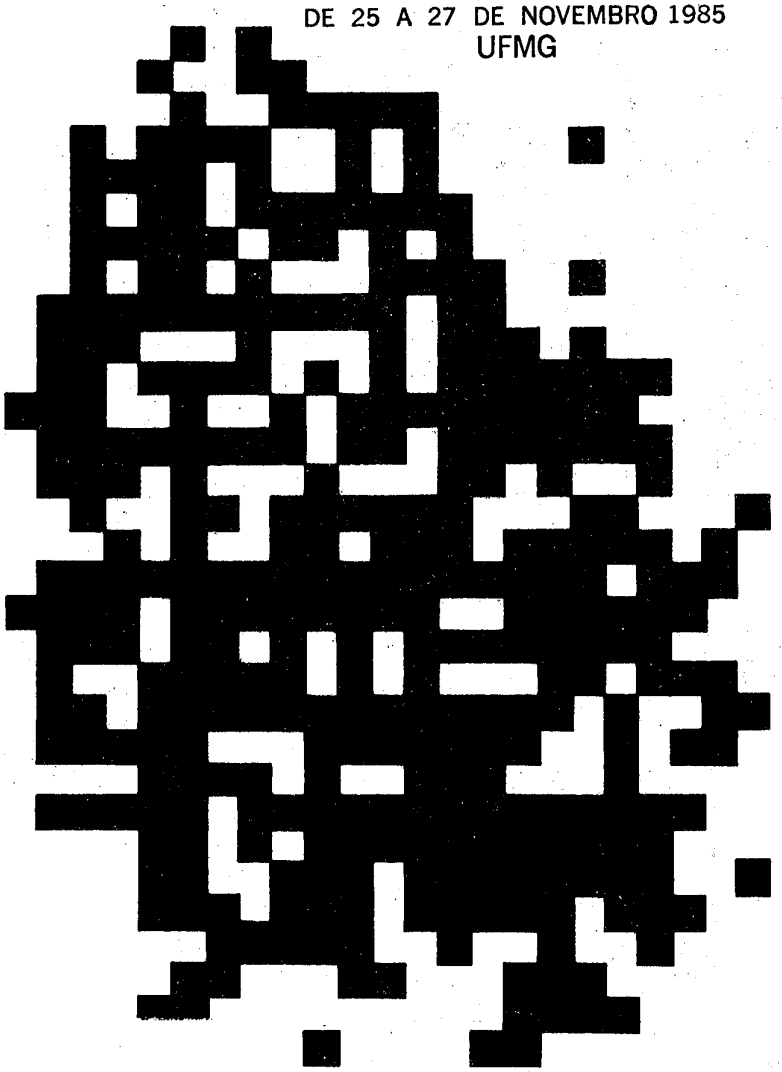
V SIMPÓSIO

SOBRE DESENVOLVIMENTO DE SOFTWARE BÁSICO

DE 25 A 27 DE NOVEMBRO 1985
UFMG



005.306
S612

ANAIS

5º SIMPÓSIO SOBRE DESENVOLVIMENTO DE SOFTWARE BÁSICO

BELO HORIZONTE 25 A 27 DE NOVEMBRO DE 1985

A N A I S

PROMOÇÃO: SOCIEDADE BRASILEIRA DE COMPUTAÇÃO - SBC
, COMISSÃO ESPECIAL PARA LINGUAGENS E SISTEMAS DE PROGRAMAÇÃO
UNIVERSIDADE FEDERAL DE MINAS GERAIS - UFMG

PATROCÍNIO: CONSELHO NACIONAL DE DESENVOLVIMENTO CIENTÍFICO E TECNOLÓGICO - CNPq

UM GERADOR DE ANALISADORES R*S

(Versão Preliminar)

José Lucas Rangel (*)
Deptº de Informática PUC-RJ
Instituto de Matemática UFRJ

(**) Sérgio de Mello Schneider
Deptº de Eng. Elétrica
Universidade Fed. Uberlândia

Este artigo descreve um novo tipo de analisador sintático empilha-reduz, ascendente, para gramáticas livres de contexto, em que se busca simultaneamente acelerar o processo de análise e diminuir o tamanho das tabelas do analisador através dos seguintes mecanismos:

1 - as decisões sobre as ações de redução levam em consideração o estado descoberto pelo desempilhamento do lado direito da produção envolvida;

2 - reduções por produções simples (produções do tipo A->B onde A e B são não-terminais) são pré-calculadas e não gastam efetivamente nenhum tempo.

O nome R*S deriva da ação característica do método, que é produzir um empilhamento (S ou "shift") após uma sequência de zero ou mais reduções (R*).

Este artigo descreve também um gerador de analisadores R*S que gera tabelas correspondentes a partir da gramática dada na forma BNF. O gerador realiza alguma compactação das tabelas que pode ser completada de acordo com as características específicas de cada aplicação.

O analisador R*S pode emitir a sequência de análise ("parse") completa, permitindo que as ações semânticas sejam associadas mesmo às produções simples. Mantém ainda a propriedade dos prefixos corretos e a implementação do analisador pode ser feita de forma que se tenha as características da detecção imediata de erros. Os detalhes destes resultados são encontrados em [Schneider 85]. A notação usada é a habitual em trabalhos sobre o assunto [Aho/Ullman 72, 73].

2. Base teórica do método R*S de análise sintática

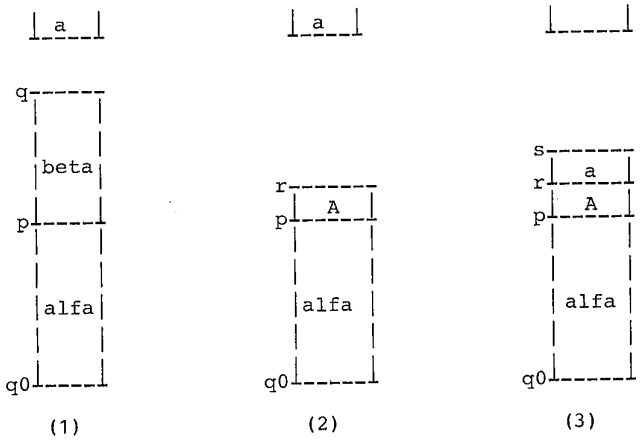
A teoria das gramáticas e analisadores R*S pode ser encontrada em [Schneider 85]. A apresentação que se segue visa apenas demonstrar em linhas gerais o funcionamento do algoritmo de análise e mostrar em que condições ele pode ser aplicado.

O princípio fundamental da análise R*S é o de suprimir os itens simples completos dos estados LR(0) (itens da forma A->B., onde A e $\overline{B}$ são não terminais), e utilizar o estado que se encontra sob o

(*) Rua Marquês de S. Vicente, 225
Gávea
20.000 Rio de Janeiro RJ

(**) Campus S. Mônica
38.400 Uberlândia MG

"handle", no caso de uma redução, para indicar com quais reduções por produções simples uma redução por produção não simples deve ser completada, de forma a se obter o não-terminal adequado. Assim, se tivermos a situação (1) abaixo



onde supomos que o estado q no topo da pilha indica uma redução pela produção B-> beta, o símbolo de "lookahead" a e o estado p descoberto após a retirada do lado direito beta são examinados para decidir:

(a) para qual não terminal A, B deve ser reduzido através de zero ou mais passos envolvendo redução por regras simples;

(b) se após a redução o símbolo a (usado como "lookahead") deve ser empilhado ou deve ainda ser usado como "lookahead" por outra(s) redução(ões) não simples;

(c) quais serão os estados r (e s) resultantes do empilhamento de A (e eventualmente de a).

Quando a é empilhado temos a situação (3); caso contrário a situação (2).

O algoritmo de análise sintática repete ações de redução (como as descritas anteriormente) e de empilhamento (como na análise LR) até que o conteúdo da pilha de análise seja a forma setencial S, que caracteriza o reconhecimento da sentença analisada, ou até que ocorra uma situação de erro.

Para melhor compeensão dos mecanismos envolvidos, apresentamos um exemplo no Apêndice, que pode ser acompanhado em paralelo.

3. Gramáticas R*S

O método R*S admite três variantes intituladas: canônica (R*Sc), "lookahead" (R*Sl) e simples (R*Ss). No presente trabalho nos concentramos na apresentação do caso R*Ss. As classes das gramáticas aceitas pelas variantes "lookahead" e simples são mais abrangentes que as correspondentes LR; no caso canônico a classe é a das gramáticas LR(1). Esses resultados, generalizados para o caso k>=0 estão em [Schneider 85].

Nesta seção mostraremos as condições que devem ser satisfeitas por uma gramática para ser analisável pelo método R*S simples.

Definição: O conjunto K de estados R*Ss de uma gramática G e definido da forma abaixo:

(a) um estado é um conjunto de itens LR(0) da gramática G aumentada pela produção S'->S;

(b) os estados são obtidos da mesma forma que no caso LR(0), exceto que:

  (1) o estado inicial é obtido pelo fechamento de {[S'->$.S$]};

  (2) nas transições entre estados, não se consideram os itens simples completos (da forma [A->B.] com A e B não terminais).

(c) a função de transição delta é definida de acordo com (b2), ou seja, em delta (q,B) não são incluídos itens simples completos [A->B.], mesmo que q contenha itens da forma [A->.B].

Definição: uma gramática livre de contexto G é R*S simples se são satisfeitas as seguintes condições:

1 - ser reduzida (não possuir símbolos estéreis ou inacessíveis);

2 - não possuir ambiguidade nas derivações simples (isto é, não podem existir duas derivações diferentes usando apenas produções simples, que permitam obter um não-terminal B a partir de um não-terminal A);

3 - o analisador R*S simples correspondente a G é consistente, ou seja, as ações do analisador nunca entram em conflito:

  (a) para todo estado p em K, não pode haver dois itens [A->beta .] e [B->alfa beta .] com Follow (A) ∩ Follow (B) <> vazio e [alfa] <> 0.

  (b) não existem estados p e q em K t ais que:

    [B1->beta .] pertence a q
    [B2->beta .] pertence a q

A1 deriva simples B1
A2 deriva simples B2
delta(p,beta)=q
delta(p,A1)=r1
delta(p,A2)=r2
[C1->gama1 .] pertence a r1
[C2->gama2 .] pertence a r2
Follow(C1) ∩ Follow(C2) <> vazio

(c) não existem estados p e q em K tais que:

[B1->beta .] pertence a q
[B2->beta .] pertence a q
A1 deriva simples B1
A2 deriva simples B2
delta(p,beta)=q
delta(p,A1)=r1
delta(p,A2)=r2
[C1->gama1 .] pertence a r1
delta(r2,a) definido
a pertence a Follow(C1)

(d) não existem estados p e q em K tais que:

[B1->beta .] pertence a q
[B2->beta .] pertence a q
A1 deriva simples B1
A2 deriva simples B2
delta(p,beta)=q
delta(p,A1 a) definido
delta(p,A2 a) definido

(e) não existem estados p e q em K tais que:

[B->beta .] pertence a q
delta(q.a) definido
A deriva simples B
delta(p,A a) definido

(f) não existem estados p e q em K tais que:

[B->beta .] pertence a q
delta(q,a) definido
A deriva simples B
delta(p,A a)=r1
[C->gama .] pertence a r1
a pertence a Follow(C)

4. Algoritmos de análise e geração de tabelas

Construído o conjunto de estados K de um analisador R*S simples, podemos construir as tabelas de empilhamento (e), salto (s) e redução (r) que são usadas pelo algoritmo de análise sintática.

Esses algoritmos são apresentados a seguir de forma simplificada:

```
{geração da tabela de empilhamento e}
     para cada estado q
       para cada t erminal a
         se delta(q,a) definido
         então e[q,a] := delta(q,a);

{geração da tabela de salto s}
     para cada estado q com redução
       para cada item completo [A- beta.] em q
         para cada terminal a em Follow(A)
           se s[q,a] não-definido
           então s[q,a] := |beta|
           senão se s[q,a] <> |beta|
                 então conflito;

{geração da tabela de redução r}
     para cada estado q com redução
       para cada item completo [A->beta.] em q
         para cada estado  p tal que delta(q,beta) = p
           para cada não-terminal B tal que B deriva simples A
             se delta(p,B) definido
             então r := delta(p,B);
             para cada terminal a
               se delta(r,a) definido
               então se f[q,a,p] e e[q,a] não-definidos
                     então f[q,a,p] := r
                     senão conflito;
                     para cada item completo [C->gama.] em r
                       se a pertence a Follow(C)
                       então se f[q,a,p] e e[[q,a] não-definidos
                             então f[q,a,p] := r
                             senão conflito;
```

Observação: a implementação de f é feita da seguinte forma: são criadas duas tabelas (implementadas na forma de listas) g e h, de forma que f[q,a,p] = g[h[q,a],p], e são feitas as compactações cabíveis. O processo acima é iniciado pela marcação das posições acessíveis de erro.

```
{algoritmo de análise sintática}
     aceita := falso; erro := falso;
     empilhe o estado inicial 0;
     scan; {define um novo terminal a}
     repita
          repita
               q := topo; r := e[q,a]; scan;
               se r for definido
               então empilhe o estado r
                     se r = estado final
                     então aceita
               até que r seja não-definido;
               se não aceitou
```

```
        então se s[q,a] não-definido
              então erro
              senão desempilha s[q,a] elementos
                    p := topo; r := f[q,a,p];
                    se r não-definido
                    então erro
                    senão empilhe o estado r {foi encontrado
                    não-terminal}
  até aceitou ou erro;
```

5. Conclusão

Foram implementadas duas versões do gerador de tabelas R*S, para o Burroughs B6700 (Algol) e para compatíveis com IBM-PC (Pascal). Técnicas de compactação específicas para o método foram desenvolvidas, e são aplicadas automaticamente, sendo ainda possível, naturalmente, compactação adicional em função de características específicas de projeto de cada compilador.

As tabelas tem dimensões comparáveis com as de analisadores LALR(1) citados na literatura, sendo especialmente reduzidas em gramáticas com estrutura similar às gramáticas de operadores, onde a alternância terminal/não-terminal pode ser melhor aproveitada.

Além disso, técnicas de recuperação automática de erro específicas para o método foram desenvolvidas e serão objeto de outro trabalho. Citaremos apenas aqui que o uso do estado descoberto pela retirada do "handle" permite a inserção de símbolos em condições normalmente não possíveis nos analisadores tradicionais. Por exemplo, ao encontrar um "then" é possível introduzir corretamente um "if" antes da expressão que representa a condição testada.

No momento, a parte referente a automatização da recuperação de erros se encontra em implementação.

Apêndice

Seja a gramática G (aumentada) dada pelas produções abaixo:

```
1. E -> E + T
2.    | T
3. T -> T * F
4.    | F
5. F -> ( E )
6.    | a
```

O analisador R*Ss tem os estados da tabela abaixo. As transições (função delta) se encontram indicadas ao lado de cada estado.

0. S->$.E$	E->1	5. S->E.	r0
E->.E+T			
|.T		6. E->E+.T	T->9
T->.T*F	T->2	T->.T*F	
|.F		|.F	
F->.(E)	(->3	F->(E)	(->3
|.a	a->4	|.a	a->4
1. S->$E.$	$->5	7. T-T*.F	F->10
E->E.+T	+->6	F->.(E)	(-> 3
		|.a	a-> 4
2. T->T.*F	*->7		
		8. F->(E.)	)->11
3. F->(.E)	E->8	E->E.+T	+-> 6
E->.E+T			
|.T		9. E- E+T.	r1
T->.T*F	T->2	T->T.*F	*-> 7
|.F			
F->.(E)	(->3	10. T->T*F.	r3
|.a	a->4		
		11. F->(E).	r5
4. F- a.	r6		

As tabelas e,r e s resultantes são as seguintes (já compactadas):

empilha

e	(	a	+	)	*
0,3,6,7	3	a4			
1	$5		+6		
2,9	*7				
8	)11		+6		

g	$,+,)	*
4,9,10,11	1	2

salto

s	
4	1
9	2
10	2
11	3

h	0	3	6	7
1	1	8	9	10
2	2	2	9	10

semântica

r	
4	r6
9	r1
10	r3
11	r5

Exemplo: análise da sentença $a*(a+a)$

pilha	entrada	comentários
$0	a*(a+a)$	e(0,a)=4
$04	*(a+a)$	e(4,*)=erro s(4)=1 (descobre 0) f(4,*,0)=2 e(2,*)=7
$0T2*7	(a+a)$	e(7,()=3
$0T2*7(3	a+a)$	e(3,a)=4
$0T2*7(3a4	+a)$	e(4,+)=erro s(4)=1 (descobre 3) f(4,+,3)=8 e(8,+)=8
$0T2*7(3E8+6	a)$	e(6,a)=4
$0T2*7(3E8+6a4	)$	e(4,))=erro s(4)=1 (descobre 6) f(4,),6)=9 e(9,1))=erro s(9)=2 (descobre 3) f(9,),3)=8 e(8,))=11
$0T2*7(3E8)11	$	e11,$)=erro s(11)=3 (descobre 7) f(11,$,7)=10 e(10,$)=erro s(10)=2 (descobre 0) f(10,$,0)=1 e(1,$)=5
$0E1$5		pare!

Notas:

(1) para a compactação indicada seria conveniente a renumeração adequada dos estados, de forma a utilizar números sucessivos para estados ou símbolos que aparecem na mesma tabela.

(2) durante a compactação, a tabela s pode dispensar o segundo argumento.

Referências:

[Aho/Ullman 72, 73]
The Theory of Parsing, Translation and Compiling, Vol. I e II, Prentice Hall, 1972 e 1973.

[Schneider 85]
Gramáticas e Linguagens R*S(k), Tese de Doutoramento, Programa de Sistemas e Computação, COPPE/UFRJ, a ser apresentada.