Algoritm derivator LL(1). Tabel de parsare

Algoritm derivator LL(1)

Convertirea automatului LL(1) in proceduri recursive: Descendenta recursiva (Recursive descent)

- derivator descendent recursiv: starea automatului este o pozitie din derivator
- stiva - locatii de unde derivatorul poate relua executia
- daca starea e $[X \to \mu.B\nu; \omega]$, $B \in N$: se pune pe stiva informatia despre $[X \to \mu B.\nu; \omega]$ inainte de a lua in considerare $B \to \beta$.
- daca folosim limbaje de programare cu suport pt recursivitate: procedura pt fiecare nonterminal B + mecanismul standard de recursivitate pentru a implementa stiva automatului

Schema de program

$q \to \varepsilon$	q: end
$qt \to q'$	q: if symbol = t then next_symbol else error; q'
	q : X; q' :...
	....
	proc X:
$qt_1 \to q'q_1t_1$	*begin*
....	*case symbol of*
$qt_m \to q'q_mt_m$	t_1 : *begin* q_1 : ... *end*;
	....
	t_m: *begin* q_m : ... *end*;
unde	otherwise error
$q = [Y \to \mu.X\nu;]$	*end*
	end

Reguli de transformare

1. nonterminal X - procedura X; simbolul de start - programul principal
2. corpul functiei X:
   - ramificare case pt productiile cu X in partea stanga
   - fiecare nonterminal din partea dreapta a productiei - apel al procedurii corespunzatoare
   - fiecare terminal din partea dreapta a productiei - verificare a presentei terminalului, urmat de apel al *next_symbol*
3. daca niciunul dintre terminalele asteptate nu e prezent - apel functia de tratare a erorilor

Exemplu cu automatul obtinut pe strong $LL(1)$

$Z \to E,\ E \to FE_1,\ E_1 \to \varepsilon | + FE_1,\ F \to i | (E)$

$$
\begin{array}{ll}
q_0: & [Z \to \bullet E] \\
q_1: & [Z \to E \bullet] \\
q_2: & [E \to \bullet F E_1] \\
q_3: & [E \to F \bullet E_1] \\
q_4: & [F \to \bullet i] \\
q_5: & [F \to \bullet (E)] \\
q_6: & [E \to F E_1 \bullet] \\
q_7: & [E_1 \to \bullet \epsilon]
\end{array}
\qquad
\begin{array}{ll}
q_8: & [E_1 \to \bullet + F E_1] \\
q_9: & [F \to i \bullet] \\
q_{10}: & [F \to (\bullet E)] \\
q_{11}: & [E_1 \to + \bullet F E_1] \\
q_{12}: & [F \to (E \bullet)] \\
q_{13}: & [E_1 \to + F \bullet E_1] \\
q_{14}: & [F \to (E) \bullet] \\
q_{15}: & [E_1 \to + F E_1 \bullet]
\end{array}
$$

q_0 : $[Z \to \bullet E]$
q_1 : $[Z \to E\bullet]$
q_2 : $[E \to \bullet F E_1]$
q_3 : $[E \to F \bullet E_1]$
q_4 : $[F \to \bullet i]$
q_5 : $[F \to \bullet(E)]$
q_6 : $[E \to F E_1 \bullet]$
q_7 : $[E_1 \to \bullet\epsilon]$
q_8 : $[E_1 \to \bullet + F E_1]$
q_9 : $[F \to i\bullet]$
q_{10} : $[F \to (\bullet E)]$
q_{11} : $[E_1 \to + \bullet F E_1]$
q_{12} : $[F \to (E\bullet)]$
q_{13} : $[E_1 \to +F \bullet E_1]$
q_{14} : $[F \to (E)\bullet]$
q_{15} : $[E_1 \to +F E_1 \bullet]$

$q_0 i \to q_1 q_2 i$, $q_0(\to q_1 q_2($,
$q_1 \to \epsilon$,
$q_2 i \to q_3 q_4 i$, $q_2(\to q_3 q_5($,
$q_3 \# \to q_6 q_7 \#$, $q_3) \to q_6 q_7)$, $q_3 + \to q_6 q_8 +$,
$q_4 i \to q_9$,
$q_5(\to q_{10}$,
$q_6 \to \epsilon$,
$q_7 \to \epsilon$,
$q_8 + \to q_{11}$,
$q_9 \to \epsilon$,
$q_{10} i \to q_{12} q_2 i$, $q_{10}(\to q_{12} q_2($,
$q_{11} i \to q_{13} q_4 i$, $q_{11}(\to q_{13} q_5($,
$q_{12}) \to q_{14}$,
$q_{13} \# \to q_{15} q_7 \#$, $q_{13}) \to q_{15} q_7)$, $q_{13} + \to q_{15} q_8 +$,
$q_{14} \to \epsilon$,
$q_{15} \to \epsilon$

$q_0 q_0$	$(i+i)\#$	$q_0(\to q_1 q_2($
$q_0 q_1 q_2$	$(i+i)\#$	$q_2(\to q_3 q_5($
$q_0 q_1 q_3 q_5$	$(i+i)\#$	$q_5(\to q_{10}$
$q_0 q_1 q_3 q_{10}$	$i+i)\#$	$q_{10} i \to q_{12} q_2 i$
$q_0 q_1 q_3 q_{12} q_2$	$i+i)\#$	$q_2 i \to q_3 q_4 i$
$q_0 q_1 q_3 q_{12} q_3 q_4$	$i+i)\#$	$q_4 i \to q_9$
$q_0 q_1 q_3 q_{12} q_3 q_9$	$+i)\#$	$q_9 \to \varepsilon$
$q_0 q_1 q_3 q_{12} q_3$	$+i)\#$	$q_3 + \to q_6 q_8 +$
$q_0 q_1 q_3 q_{12} q_6 q_8$	$+i)\#$	$q_8 + \to q_{11}$
$q_0 q_1 q_3 q_{12} q_6 q_{11}$	$i)\#$	$q_{11} i + \to q_{13} q_4 i$
$q_0 q_1 q_3 q_{12} q_6 q_{13} q_4$	$i)\#$	$q_4 i \to q_9$
$q_0 q_1 q_3 q_{12} q_6 q_{13} q_9$	$)\#$	$q_9 \to \varepsilon$
$q_0 q_1 q_3 q_{12} q_6 q_{13}$	$)\#$	$q_{13}) \to q_{15} q_7)$
$q_0 q_1 q_3 q_{12} q_6 q_{15} q_7$	$)\#$	$q_7 \to \varepsilon$
$q_0 q_1 q_3 q_{12} q_6 q_{15}$	$)\#$	$q_{15} \to \varepsilon$
$q_0 q_1 q_3 q_{12} q_6$	$)\#$	$q_6 \to \varepsilon$
$q_0 q_1 q_3 q_{12}$	$)\#$	$q_{12}) \to q_{14}$
$q_0 q_1 q_3 q_{14}$	$\#$	$q_{14} \to \varepsilon$
$q_0 q_1 q_3$	$\#$	$q_3 \# \to q_6 q_7 \#$
$q_0 q_1 q_6 q_7$	$\#$	$q_7 \to \varepsilon$
$q_0 q_1 q_6$	$\#$	$q_6 \to \varepsilon$
$q_0 q_1$	$\#$	$q_1 \to \varepsilon$
q_0	$\#$	$q_1 \to \varepsilon$

$$q_0 = [Z \to .E]$$
$$q_1 = [Z \to E.],\ q_2 = [E \to .FE_1]$$
$$q_3 = [E \to F.E_1],\ q_5 = [F \to .(E)]$$
$$q_{10} = [F \to (.E)]$$
$$q_{12} = [F \to (E.)], q_2 = [E \to .FE_1]$$
$$q_3 = [E \to F.E_1], \ldots$$

Schema de program

$q \to \varepsilon$	q: end
$qt \to q'$	q: if symbol = t then next_symbol else error; q'
	q : X; q' :...
	....
	proc X:
$qt_1 \to q'q_1t_1$	*begin*
....	*case symbol of*
$qt_m \to q'q_mt_m$	t_1 : *begin* q_1 : ... *end*;
	....
	t_m: *begin* q_m : ... *end*;
unde	otherwise error
$q = [Y \to \mu.X\nu;]$	*end*
	end

- Pt tranzitii $qt_1 \rightarrow q'q_1t1...$
- schema program indica:
  $q : F(); q'$
  procedura $F()$ - case pt toate t_i
- $q_2 i \rightarrow q_3 q_4 i$, $q_2(\rightarrow q_3 q_5($
  $q_4 i \rightarrow q_9$, $q_9 \rightarrow \varepsilon$,
  $q_5(\rightarrow q_{10}$, $q_{10} i \rightarrow q_{12} q_2 i$, $q_{10}(\rightarrow q_{12} q_2($,
- $q_2 = [E \rightarrow .FE_1]$, $q_3 = [E \rightarrow F.E_1]$, $q_{10} = [F \rightarrow (.E)]$

```
q2: F(); q3

procedure F()
{ case symbol of
   'i' : { q4:  if (symbol == 'i') then next_symbol else
         error();
             q9:  ;}
   '(' : { q5:  if (symbol == '(') then next_symbol else
         error();
             q10: E();
             q12: if (symbol == ')') then next_symbol else
                  error();
             q14: ;}
    otherwise error();   }
```

```
derivator()
{ q0: E()
  q1: if (symbol != '#')
        error();
}
procedure E1()
{ case symbol of
    '#' , ')' : q7: ;
    '+' : {
          q8:  if (symbol == '+')  next_symbol(); else error
              ();
          q11: F();
          q13: E1;
          q15: ;
    otherwise   : error();
}
procedure F()
{ case symbol of
   'i' : { q4:  if (symbol == 'i') then next_symbol else
       error();
           q9:  ;}
   '(' : { q5:  if (symbol == '(') then next_symbol else
       error();
           q10: E();
           q12: if (symbol == ')') then next_symbol else
               error();
           q14: ;}
    otherwise error();   }
```

```
procedure E()
{ q2: F();
  q3: E1();
  q6: ;
 }
```

Parsing table - tabel de derivare

- Ullman 4.4 . Nonrecursive predictive parsing
- Table-driven predictive parsing: input, stiva, parsing table.
- Tabel de derivare: M[A,a] - A nonterminal, a - terminal sau #

Exemplu de tabel de derivare

	lookahead					
	i	$+$	$*$	(	)	#
E	$E \to TE'$			$E \to TE'$		
E'		$E' \to +TE'$			$E' \to \varepsilon$	$E' \to \varepsilon$
T	$T \to FT'$			$T \to FT'$		
T'		$T' \to \varepsilon$	$T' \to *FT'$		$T' \to \varepsilon$	$T' \to \varepsilon$
F	$F \to i$			$F \to (E)$		

$$
\begin{aligned}
P = \{ & E \to TE' \\
& E' \to +TE' | \varepsilon \\
& T \to FT' \\
& T' \to *FT' | \varepsilon \\
& F \to (E) | id \}
\end{aligned}
$$

Algoritm de derivare predictiva cu tabel de derivare

```
  #S (simbol de start) pe stiva, string# la intrare
set ip  to point to the first symbol of input string
repeat
  let X be the top stack symbol and a the symbol pointed to
      by ip
  if X is a terminal or # then
     if X = a then
        pop X from the stack and advance ip
     else error()
  else
     if M[X,a] = X-> Y1 Y2 ...Yk then begin
        pop X fro the stack
        push Yk, Yk-1, ...Y1 onto the stack, with Y1 on top
        output the production X-> Y1 Y2 ...Yk
     end
     else error()
until X=#
```

Exemplu de tabel de derivare

	lookahead					
	id	$+$	$*$	$($	$)$	$\#$
E	$E \to TE'$			$E \to TE'$		
E'		$E' \to +TE'$			$E' \to \varepsilon$	$E' \to \varepsilon$
T	$T \to FT'$			$T \to FT'$		
T'		$T' \to \varepsilon$	$T' \to *FT'$		$T' \to \varepsilon$	$T' \to \varepsilon$
F	$F \to id$			$F \to (E)$		

$$
\begin{aligned}
P = \{ & E \to TE' \\
& E' \to +TE' | \varepsilon \\
& T \to FT' \\
& T' \to *FT' | \varepsilon \\
& F \to (E) | id \}
\end{aligned}
$$

simbol	$FIRST_1(X)$	$FOLLOW_1(X)$
E	$\{(, id\}$	$\{), \#\}$
E'	$\{+, \varepsilon\}$	$\{), \#\}$
T	$\{(, id\}$	$\{+, \#,)\}$
T'	$\{*, \varepsilon\}$	$\{+, \#,)\}$
F	$\{(, id\}$	$\{*, +, \#,)\}$

1. for each production $A \rightarrow \alpha$ do steps 2 and 3
2. for each terminal a in $FIRST(\alpha)$, add $A \rightarrow \alpha$ to $M[A, a]$
3. if $\varepsilon \in FIRST(\alpha)$, add$A \rightarrow \alpha$ to $M[A, b$ for each terminal $b \in FOLLOW(A)$. if $\varepsilon \in FIRST(\alpha)$ and $\# \in FOLLOW(A)$, add $A \rightarrow \alpha$ to $M[A, \#]$
4. Make each undefined entry of M be error