

Analizorul descendent cu reveniri

- Automat: configuratii, tranzitii
- Se poate folosi pentru
gramatici nerecursive la stanga

Analizorul descendent cu reveniri

Configuratie:

(s, i, α, β)

- s- starea automatului
 - q – stare normala
 - r – stare de revenire (sau b – back)
 - t – stare de terminare (terminare cu succes)
 - e – stare de eroare
- i – pozitia (urmatoare) in secventa de intrare
- α – stiva de lucru (de istorie): istoria r.p. aplicate
- β – stiva de intrare: partea inca neprelucrata

Analizorul descendent cu reveniri

- configuratie initiala: $(q, 1, \varepsilon, S)$

- Tranzitii:

- expandare:

$$(q, i, \alpha, A\beta) \vdash (q, i, \alpha A_1, \gamma_1 \beta)$$

- avans:

$$(q, i, \alpha, a_i \beta) \vdash (q, i+1, \alpha a_i, \beta)$$

- insucces de moment:

$$(q, i, \alpha, a\beta) \vdash (r, i, \alpha, a\beta) \quad , a \neq a_i$$

$$(q, i, \alpha, \varepsilon) \vdash (r, i, \alpha, \varepsilon) \quad , i \neq n+1$$

- succes:

$$(q, n+1, \alpha, \varepsilon) \vdash (t, n+1, \alpha, \varepsilon)$$

- revenire:

$$(r, i, \alpha a, \beta) \vdash (r, i-1, \alpha, a\beta)$$

- alta incercare:

$$(r, i, \alpha A_j, \gamma_j \beta) \vdash \dots$$

daca $\exists A_{j+1} \rightarrow \gamma_{j+1}$

$$\vdash (q, i, \alpha A_{j+1}, \gamma_{j+1} \beta)$$

altfel daca $i = 1, A = S$

$$\vdash (e, i, \alpha, A\beta)$$

$\alpha = \varepsilon, \beta = \varepsilon$

altfel

$$\vdash (r, i, \alpha, A\beta)$$

Orice altceva: blocare (eroare)

Analizorul descendent cu reveniri

Observatii:

- Se numeroteaza regulile de productie cu acelasi membru stang
- Stiva de lucru contine informatiile referitoare la regulile de productie aplicate

Schita subalgoritm ASDR

Subalgoritmul ASDR($G, x, \text{sir_prod}$)

```

s:=q; i:=1;  $\alpha$ := $\epsilon$ ;  $\beta$ :=S; //config initiala
Cattimp ((s $\neq$ t) si (s  $\neq$ e)) executa
  daca (s=q) atunci
    daca ( $\beta$ = $\epsilon$ ) atunci
      daca (i=n+1)) atunci
        s:=t
      altfel
        s:=r
      sf_daca
    altfel
      daca (varf( $\beta$ ) = un neterminal A) atunci
        pop( $\beta$ , A);
        push( $\alpha$ , A1);
        push( $\beta$ , Y1);
      altfel
        daca (varf( $\beta$ ) = terminalul xi) atunci
          i:=i+1;
          pop( $\beta$ , xi);
          push( $\alpha$ , xi);
        altfel s:=r;
      sf_daca
    sf_daca
  sf_daca
altfel
  // ...

```

```

daca(s=r) atunci
  daca (varf( $\alpha$ ) = un terminal  $a$ ) atunci
    i:=i-1;
    pop( $\alpha$ ,  $a$ );
    push( $\beta$ ,  $a$ );
  altfel // varf( $\alpha$ ) - un  $A_j$  oarecare (indicatie pt.  $A \rightarrow Y_j$ )
    daca (exista r.p.  $A \rightarrow Y_{j+1}$  ) atunci
      s:=q;
      pop( $\alpha$ ,  $A_j$ );
      push( $\alpha$ ,  $A_{j+1}$ );
      pop( $\beta$ ,  $Y_j$ );
      push( $\beta$ ,  $Y_{j+1}$ );
    altfel
      daca (i=1) si (A=S) atunci s:=e;
      altfel
        pop( $\alpha$ ,  $A_j$ );
        push( $\beta$ ,  $A_j$ );
      sf_daca
    sf_daca
  Sf_daca
Sf_daca
Sf_cattimp
Daca s=e atunci
  tipareste "Eroare"
altfel
  tipareste "Secventa este acceptata"
  constructie_sir_prod(G,  $\alpha$ );

Sf_daca
Sf_subalgoritm

```

•

Subalgoritmul constructie_sir_prod(G, α) este:

sir_prod:="";

Cattimp not *vida(α)* executa

daca *varf(α)* este de forma A_j atunci

sir_prod:= sir_prod + *indicativ_rp(varf(α))*;

sf_daca

pop(α);

Sf_Cattimp

Sf_subalgoritm

Gramatici recursive → curs 5