

Analizorul descendente cu reveniri

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

Analizorul descendente 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 descendente cu reveniri

- configurație initială: $(q, 1, \varepsilon, S)$

- Tranzitii:

- expandare:
- avans:
- insucces de moment:

- succes:
- revenire: $(r, i, \alpha a, \beta)$
- alta incercare: $(r, i, \alpha A_j, \gamma_j \beta)$

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

altfel daca $i = 1, A = S$

altfel

$$\begin{array}{ll} (q, i, \alpha, A\beta) & \vdash (q, i, \alpha A_1, \gamma_1 \beta) \\ (q, i, \alpha, a_i \beta) & \vdash (q, i+1, \alpha a_i, \beta) \\ (q, i, \alpha, a\beta) & \vdash (r, i, \alpha, a\beta) , a <> a_i \\ (q, i, \alpha, \varepsilon) & \vdash (r, i, \alpha, \varepsilon) , i \neq n+1 \\ (q, n+1, \alpha, \varepsilon) & \vdash (t, n+1, \alpha, \varepsilon) \end{array}$$

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

$$\vdash \dots$$

$$\begin{array}{l} \vdash (q, i, \alpha A_{j+1}, \gamma_{j+1} \beta) \\ \vdash (e, i, \alpha, A\beta) \end{array}$$

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

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

Analizorul descendant 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, sir_prod)

```
s:=q; i:=1; α:=ε; β:=S; //config initiala
Cattimp ((s≠t) si (s ≠e)) execută
daca (s=q) atunci
    daca (β=ε) atunci
        daca (i=n+1) atunci
            s:=t
        altfel
            s:=r
        sf_daca
    altfel
        daca (varf(β) = un neterminal A) atunci
            pop(β, A);
            push(α, A1);
            push(β, Y1);
        altfel
            daca (varf(β) = terminalul xi) atunci
                i:=i+1;
                pop(β, xi);
                push(α, xi);
            altfel s:=r;
            sf_daca
        sf_daca
    altfel
        // ...
```

```
daca(s=r) atunci
    daca (varf(α) = un terminal a) atunci
        i:=i-1;
        pop(α, a);
        push(β, a);
    altfel // varf(α) - un Aj oarecare (indicatie pt. A→Yj)
        daca (exista r.p. A→Yj+1 ) atunci
            s:=q;
            pop(α, Aj);
            push(α, Aj+1);
            pop(β, Yj );
            push(β, Yj+1 );
    altfel
        daca (i=1) si (A=S) atunci s:=e;
        altfel
            pop(α, Aj );
            push(β, Aj );
        sf_daca
        sf_daca
    Sf_daca
    Sf_daca
    Sf_daca
    Sf_cattimp
    Daca s=e atunci
    altfel
        tipareste "Eroare"
        tipareste "Seventa este acceptata"
        constructie_sir_prod(G, α);
    Sf_daca
    Sf_subalgoritm
```

•

Subalgoritmul constructie_sir_prod(G, α) este:

sir_prod:=“”;

Cattimp not $vida(\alpha)$ executa

 daca $varf(\alpha)$ este de forma A_j atunci

 sir_prod:= sir_prod + $indicativ_rp(varf(\alpha))$;

 sf_daca

 pop(α);

Sf_Cattimp

Sf_subalgoritm