Analiza sintactică bazată pe precedența operatorilor (ASPO)

ASPO se poate aplica doar la o clasă redusă dar importantă de gramatici iar analizorul se poate construi uşor manual, pe principiul de lucru al ASDR. Gramatica trebuie să îndeplinească următoarele două cerințe:

1) Să nu aibă producții vide (cu **ε** în partea dreaptă).
2) În nici o parte dreaptă să nu fie două neterminale adiacente.

Gramaticile care satisfac ultima proprietate se numesc **gramatici de operatori**.

Exemplu: Următoarea gramatică pentru expresii nu este gramatică de operatori:

$E \rightarrow EAE \mid (E) \mid -E \mid \mathbf{id}$

$A \rightarrow + \mid - \mid * \mid / \mid \uparrow$

Ea poate fi transformată într-o gramatică de operatori prin substituirea lui A cu fiecare din alternativele sale:

$E \rightarrow E+E \mid E-E \mid E*E \mid E/E \mid E\uparrow E \mid (E) \mid -E \mid \mathbf{id}$

Ca tehnică generală de analiză sintactică, ASPO are o serie de dezavantaje:

- Sunt dificil de prelucrat acei atomi care au două precedențe diferite (exemplu: semnul minus, care este atât operator unar cât şi binar).
- Relația între analizor şi gramatică nu este întotdeauna biunivocă ceea ce face ca analizorul să accepte şiruri care nu fac parte din limbajul definit de gramatică.
- Clasa de gramatici analizabile prin tehnica ASPO este redusă.

Totuşi, datorită simplității, în numeroase compilatoare s-a utilizat această tehnică exclusiv pentru expresii, instrucțiunile şi construcțiile de nivel mai înalt fiind analizate prin alte metode (de exemplu, cu descendenți recursivi). S-au construit însă şi ASPO pentru limbaje întregi.

Definirea relațiilor de precedență

În ASPO se definesc trei relații de precedență disjuncte, notate $\lessdot$, $\doteq$ şi $\gtrdot$, care se stabilesc între anumite perechi de terminale. Pe baza acestor relații se selectează capătul formei propoziționale, în stiva analizorului. Semnificația lor este următoarea:

Relația	Semnificația
$a \lessdot b$	a “cedează precedența” lui b
$a \doteq b$	a “are aceeaşi precedență” ca b
$a \gtrdot b$	a “are precedență” față de b

Aceste relații se deosebesc în esență de relațiile algebrice <, = şi > din următoarele motive:

- au cu totul altă semantică;
- este posibil ca între două terminale să nu existe nici o relație de precedență;
- este posibil ca între două terminale să existe simultan două relații de precedență (exemplu: $a \lessdot b$ şi $a \gtrdot b$).

Relațiile de precedență între perechile de terminale se pot stabili pe două căi:

1) **Intuitiv**: pornind de la semnificația algebrică a operatorilor şi ținând cont de prioritatea şi de asociativitatea lor. De exemplu, deoarece “*” trebuie să aibă prioritate mai mare decât “+” se face $* \gtrdot +$ şi $+ \lessdot *$. Astfel se rezolvă şi ambiguitățile gramaticii expresiilor şi se poate scrie şi programul de ASPO.

2) **În mod automat**: aplicând un algoritm adecvat. În prealabil, trebuie eliminată ambiguitatea din gramatica inițială a limbajului, pentru ca ea să reflecte corect asociativitatea şi prioritatea operatorilor. Pentru gramatici mai complicate decât gramatica expresiilor este posibil ca relațiile generate automat să nu fie disjuncte şi limbajul să fie mai cuprinzător decât cel inițial.

Utilizarea relațiilor de precedență a operatorilor

Scopul introducerii relațiilor de precedență este delimitarea capetelor într-o formă propozițională dreapta. Astfel, $\lessdot$ marchează extremitatea stângă a unui capăt, $\doteq$ apare în interiorul capătului și $\gtrdot$ marchează extremitatea sa dreaptă.

Capătul poate fi găsit prin următorul procedeu:

1) Se baleiază șirul, cu relațiile de precedență introduse, de la extremitatea stângă și până la întâlnirea primului marcaj $\gtrdot$. Acesta reprezintă extremitatea dreaptă a capătului.

2) Se baleiază șirul înapoi, omițând eventualele simboluri $\doteq$, până la întâlnirea primului marcaj $\lessdot$. Acesta reprezintă extremitatea stângă a capătului.

3) Se consideră ca fiind capăt întreg șirul de simboluri gramaticale situat la stânga marcajului $\gtrdot$ și la dreapta marcajului $\lessdot$, inclusiv toate neterminalele care apar ca incluse sau înconjurând terminalele dintre cele două marcaje. Considerarea neterminalelor înconjurătoare este necesară pentru a asigura că nu vor apărea două neterminale adiacente în forma propozițională următoare.

Exemplu: Se consideră șirul de intrare: $**id**+**id**∗ **id** $ și matricea relațiilor de precedență de mai jos:

	id	+	∗	$
id		$\cdot\rangle$	$\cdot\rangle$	$\cdot\rangle$
+	$\langle\cdot$	$\cdot\rangle$	$\langle\cdot$	$\cdot\rangle$
∗	$\langle\cdot$	$\cdot\rangle$	$\cdot\rangle$	$\cdot\rangle$
$	$\langle\cdot$	$\langle\cdot$	$\langle\cdot$	

Șirul cu relațiile de precedență introduse, este:

$\$\langle\cdot\underline{\mathbf{id}}\cdot\rangle + \langle\cdot id\cdot\rangle ∗ \langle\cdot id\cdot\rangle \$$	capătul : **id**
$E+id∗id$	red : E→**id**
$\$\langle\cdot+\langle\cdot\underline{\mathbf{id}}\cdot\rangle ∗ \langle\cdot id\cdot\rangle \$$	capătul : **id**
$E+E∗id$	red : E→**id**
$\$\langle\cdot id+\langle\cdot ∗ \langle\cdot\underline{\mathbf{id}}\cdot\rangle \$$	capătul : **id**
$E+E∗E$	red : E→**id**
$\$\langle\cdot+\langle\cdot ∗\cdot\rangle \$$	capătul : E∗E
$E+E$	red : E→E∗E
$\$\langle\cdot+\cdot\rangle \$$	capătul : E+E
E	red : E→E+E

Deoarece neterminalele nu influențează analiza sintactică, nu trebuie făcută distincție între ele. În stiva analizorului este suficient să se țină un singur fel de marcaj, "neterminal", pentru a indica locurile respective (utile doar pentru înregistrarea atributelor semantice).

Acțiunile analizorului sunt cele cunoscute:

1) **Deplasare**: cât timp nu s-a găsit extremitatea dreaptă a capătului adică, între simbolul terminal cel mai apropiat de vârful stivei și simbolul curent de intrare, este valabilă una din relațiile $\langle\cdot$ sau $\doteq$.

2) **Reducere**: S-a găsit extremitatea dreaptă a capătului adică, între simbolul terminal cel mai apropiat de vârful stivei și simbolul curent de intrare, este relația $\cdot\rangle$. Prin baleierea în sens invers a conținutului stivei și verificarea relațiilor de precedență se caută extremitatea stângă după care se efectuează reducerea.

3) **Acceptare**: în situația în care ambele simboluri care se compară (vârful stivei și simbolul curent de intrare) sunt $.

4) **Eroare**: dacă se ajunge în situația să se compare o pereche de terminale între care nu există nici o relație de precedență. Se va apela la o rutină specială de tratare și revenire.

Ideile de mai jos pot fi sintetizate în următorul algoritm:

Algoritmul de ASPO

Intrare: șirul *w* și tabela relațiilor de precedență.

Ieșire: dacă șirul *w* este corect se va obține un schelet al arborelui său sintactic (datorită substituirii neterminalelor pe parcursul analizei, nodurile interioare, corespunzătoare acestora, vor fi uniformizate); în caz contrar, se va da un mesaj de eroare.

Situația inițială a structurii de date:

STIVA	**INTRARE**
$	w$

Algoritmul propriu-zis:

```
(1)  poziționează pointerul de intrare pe primul simbol din w;
(2)  repeat forever
(3)     if ∗ atât simbolul din vârful stivei cât și simbolul curent de intrare
(4)        sunt $ then return {acceptare}
        else begin
(5)           ∗ fie a simbolul terminal cel mai apropiat de vârful stivei
              și b simbolul curent de intrare;
(6)           if (a⋖ b) or (a≐ b) then begin {deplasare}
(7)                ∗ introdu b în stivă;
(8)                ∗ avansează cu o poziție pointerul de intrare
(9)           end else if a ⋗ b then {reducere}
(10)                  repeat
(11)                       ∗ extrage din stivă
(12)                  until ∗ terminalul din vârful stivei este în relația ⋖
                            cu terminalul cel mai recent extras
(13)                 else eroare ()
           end
```

Deducerea intuitivă a relațiilor de precedență din asociativitatea şi prioritatea algebrică a operatorilor

Singura cerință care trebuie avută în vedere la stabilirea relațiilor de precedență este aceea ca ele să conducă la analiza corectă a limbajului definit de gramatică. Ținând cont de faptul că ASPO se aplică în primul rând la gramatici pentru expresii sau similare cu acestea, iar între operatorii din expresii există reguli de asociativitate şi prioritate riguroase care rezolvă eventualele ambiguități, aceste reguli pot reprezenta baza stabilirii relațiilor de precedență.

Pentru cazul operatorilor binari, notați cu θ, θ_1 şi θ_2, relațiile de precedență pot fi deduse astfel:

1) Dacă θ_1 are prioritate algebrică mai mare ca θ_2, se crează relațiile de precedență: $\theta_1 \cdot\!> \theta_2$ şi $\theta_2 <\!\cdot\, \theta_1$

Exemplu: $* \cdot\!> +$, $+ <\!\cdot *$ şi din E+<u>E∗E</u>+E, E∗E se va reduce primul.

2) Dacă θ1 şi θ2 sunt de prioritate egală (inclusiv cazul când ambele reprezintă acelaşi operator) apar următoarele situații:

a) θ_1 şi θ_2 sunt asociativi la stânga $\Rightarrow \theta_1 \cdot\!> \theta_2$ şi $\theta_2 \cdot\!> \theta_1$

b) θ_1 şi θ_2 sunt asociativi la dreapta $\Rightarrow \theta_1 <\!\cdot\, \theta_2$ şi $\theta_2 <\!\cdot\, \theta_1$

Exemplu: $+ \cdot\!> +$, $+ \cdot\!> -$, $- \cdot\!> -$, $- \cdot\!> +$ şi din <u>E-E</u>+E se va selecta la început capătul E-E

$\uparrow <\!\cdot \uparrow$ (asociativ la dreapta) şi din E↑<u>E↑E</u> se va selecta la început ultimul E↑E

3) Oricare ar fi θ , există următoarele relații de precedență cu celelalte terminale: id, (,), $

$\theta <\!\cdot$ **id**	$\theta <\!\cdot$ (	$\theta \cdot\!>$)	$\theta \cdot\!>$ $
id $\cdot\!> \theta$	($<\!\cdot\, \theta$	) $\cdot\!> \theta$	$ $<\!\cdot\, \theta$

Pentru a asigura reducerea la E a lui **id** şi a (E), între terminalele care nu sunt operatori se introduc următoarele relații:

($\doteq$)	$ $<\!\cdot$ (	$ $<\!\cdot$ **id**
($<\!\cdot$ (	**id** $\cdot\!>$ $	) $\cdot\!>$ $
($<\!\cdot$ **id**	**id** $\cdot\!>$)	) $\cdot\!>$)

Exemplu: gramatica E→E+E | E-E | E∗E | E/E | E↑E | (E) | **id**

Operatorii au următoarele priorități:

1) ↑ este cel mai prioritar şi este asociativ la dreapta;
2) ∗ şi / au prioritatea următoare şi sunt asociativi la stânga;
3) + şi – au cea mai mică prioritate şi sunt asociativi la stânga

Rezultă următoarea tabelă a relațiilor de precedență (absența relației înseamnă caz de eroare):

	+	-	∗	/	↑	**id**	(	)	$
+	$\cdot\!>$	$\cdot\!>$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$\cdot\!>$	$\cdot\!>$
-	$\cdot\!>$	$\cdot\!>$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$\cdot\!>$	$\cdot\!>$
∗	$\cdot\!>$	$\cdot\!>$	$\cdot\!>$	$\cdot\!>$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$\cdot\!>$	$\cdot\!>$
/	$\cdot\!>$	$\cdot\!>$	$\cdot\!>$	$\cdot\!>$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$\cdot\!>$	$\cdot\!>$
↑	$\cdot\!>$	$\cdot\!>$	$\cdot\!>$	$\cdot\!>$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$\cdot\!>$	$\cdot\!>$
id	$\cdot\!>$	$\cdot\!>$	$\cdot\!>$	$\cdot\!>$	$\cdot\!>$			$\cdot\!>$	$\cdot\!>$
(	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$\doteq$	
)	$\cdot\!>$	$\cdot\!>$	$\cdot\!>$	$\cdot\!>$	$\cdot\!>$			$\cdot\!>$	$\cdot\!>$
$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$		

Exemplu: **id∗(id↑id)-id/id**

$ $<\!\cdot$ **id** $\cdot\!>$ ∗ $<\!\cdot$ ($<\!\cdot$ id $\cdot\!>$ ↑ $<\!\cdot$ id $\cdot\!>$) $\cdot\!>$ - $<\!\cdot$ id $\cdot\!>$ / $<\!\cdot$ id $\cdot\!>$ $

E→id

$ $<\!\cdot$ ∗ $<\!\cdot$ ($<\!\cdot$ <u>**id**</u> $\cdot\!>$ ↑ ...

E→id

$ $<\!\cdot$ ∗ $<\!\cdot$ ($<\!\cdot$ ↑ $<\!\cdot$ <u>**id**</u> $\cdot\!>$) ...

E→id

$ $<\!\cdot$ ∗ $<\!\cdot$ ($<\!\cdot$ ↑ $\cdot\!>$) ...

E→E↑E

$ $<\!\cdot$ ∗ $<\!\cdot$ ($\doteq$) $\cdot\!>$ - ...

E→(E)

$ $<\!\cdot$ <u>∗</u> $\cdot\!>$ - ...

E→E∗E

$ $<\!\cdot$ - $<\!\cdot$ <u>**id**</u> $\cdot\!>$ / ...

E→id

$ $<\!\cdot$ - $<\!\cdot$ / $<\!\cdot$ <u>**id**</u> $\cdot\!>$ $

E→id

$ $<\!\cdot$ - $<\!\cdot$ <u>/</u> $\cdot\!>$ $

E→E/E

$ $<\!\cdot$ <u>-</u> $\cdot\!>$ $

E→E-E

<u>**E**</u>

Acceptare

Tratarea operatorilor unari

Pentru acei operatori unari (prefix sau postfix) care nu sunt în acelaşi timp şi binari, se pot uşor extinde regulile din paragraful precedent. Pentru exemplificare să considerăm θ ca fiind un operator oarecare, binar sau unar, şi să definim relațiile sale de precedență cu operatorul unar ¬ (negare logică):

$\theta \lessdot \neg$	şi
$\neg \gtrdot \theta$	dacă θ are o prioritate mai mică decât ¬
$\neg \lessdot \theta$	dacă θ are o prioritate mai mare decât ¬ .

Exemplu: ¬ (operator unar) mai prioritar decât & (operator binar). Conform acestor reguli, expresia E& ¬ E&E va fi evaluată astfel: (E&(¬ E)) &E.

Dacă operatorul unar este în acelaşi timp şi binar (cazul semnului -) situația se complică foarte mult, chiar dacă cei doi operatori, identici ca formă, ar avea aceeaşi precedență.. Pentru exemplificare se poate studia tratarea şirului **id**∗-**id**, conform tabelei anterioare. O soluție posibilă este modificarea ANLEX astfel încât să returneze coduri lexicale diferite pentru cele două situații de utilizare a semnului "-". Dacă se realizează acest lucru, operatorul unar minus devine un operator distinct şi poate fi tratat cu regulile de mai sus. Trebuie totuşi menționat că nu este simplu pentru ANLEX să distingă între cele două situații, deoarece este nevoie să privească operatorul în contextul în care apare.

Funcții de precedență

Un dezavantaj al ASPO poate să fie dimensiunea relativ mare a tabelei relațiilor de precedență. În majoritatea cazurilor însă, nu este obligatoriu ca analizorul să păstreze în memorie această tabelă. Ea poate fi substituită prin două **funcții de precedență** notate f şi g, care pun în corespondență simbolurile gramaticale cu numere întregi. Funcțiile **f** şi **g** vor fi astfel alese încât să satisfacă următoarele condiții:

1)	$f(a) < g(b)$	dacă	$a \lessdot b$
2)	$f(a) = g(b)$	dacă	$a \doteq b$
3)	$f(a) > g(b)$	dacă	$a \gtrdot b$

În acest fel, determinarea relației de precedență între *a* şi *b* se reduce la simpla comparare numerică între f(a) şi g(b). Un dezavantaj al utilizării funcțiilor de precedență este acela că nu mai pot fi sesizate acele cazuri de eroare marcate în tabelă prin intrări vide (lipsa relației de precedență). Cauza este aceea că între numerele f(a) şi g(b) există întotdeauna una dintre relațiile 1), 2) sau 3). Dezavantajul nu este însă foarte mare pentru că erorile nu se pierd ci doar se amână detectarea lor până în momentul încercării de reducere, când nu va putea fi găsit un capăt corespunzător.

Nu orice tabelă a relațiilor de precedență poate fi codificată prin funcții de precedență pentru că nu pot fi construite funcțiile astfel încât să respecte cerințele de mai sus pentru toate relațiile. Totuşi, pentru multe cazuri practice, funcțiile există şi se pot construi.

Exemplu: Tabela de precedență anterioară are următoarea pereche de funcții de precedență:

	+	–	∗	/	↑	(	)	**id**	$
f	2	2	4	4	4	0	6	6	0
g	1	1	3	3	5	5	0	5	0

Se observă că f(**id**) > g(**id**) ⇒ **id** $\gtrdot$ **id**, deşi, în realitate, între **id** şi **id** nu există relație de precedență – caz de eroare.

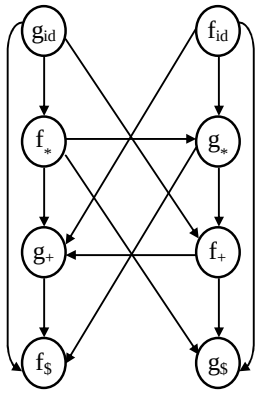
Algoritmi de construire a funcțiilor de precedență:

Metoda

1) Crează simbolurile f_a şi g_a pentru fiecare terminal a, inclusiv $.
2) Se partiționează simbolurile create în grupuri, astfel încât dacă $\doteq$ b atunci f_a şi g_b sunt în același grup.
3) Se crează un graf orientat ale cărui noduri sunt constituite din grupurile găsite anterior iar arcele se stabilesc astfel:
   a) pentru toate perechile a şi b pentru care a $\lessdot$ b se duce un arc de la grupul lui g_b la grupul lui f_a;
   b) dacă a $\gtrdot$ b, se duce un arc de la grupul lui f_a la grupul lui g_b.

   Existența unui arc sau a unui drum de la f_a la g_b înseamnă că valoarea lui f(a) trebuie să depăşească pe cea a lui g(b). Invers, un drum de la g_b la f_a înseamnă că trebuie ca g(b) > f(a).
4) Dacă graful construit la punctul 3) are un ciclu, atunci nu există funcții de precedență. Dacă nu există cicluri atunci, pentru orice terminal *a*, inclusiv $, f(a) va primi ca valoare lungimea celui mai lung drum care începe în grupul lui f_a iar g(a) va fi egală cu lungimea celui mai lung drum care începe în grupul lui g_a.

Exemplu: Se consideră matricea de precedență redusă (pentru simbolurile **id**, +, * şi $). Nu există relația $\doteq$ aşa că fiecare simbol constituie el însuşi un grup conform algoritmului, rezultă următorul graf:



Nu există cicluri $\Rightarrow$ există funcții de precedență

	+	*	**id**	$
f	2	4	4	0
g	1	3	5	0

f($)=g($)=0 deoarece $f_\$$ şi $g_\$$ nu au arce de ieşire. g(+)=1 pentru că cel mai lung drum este format dintr-un singur arc. ş. a. m. d.

Revenirea din erori în analiza sintactică bazată pe precedența operațiilor.

În procesul de ASPO există două situații în care se pot detecta erori sintactice:

1) Dacă s-a găsit un capăt (conform relațiilor de precedență) dat nu există nici o producție având acel capăt ca parte dreaptă.
2) Dacă între terminalul din vârful stivei şi simbolul curent de intrare nu există relație de precedență.

Într-un capăt detectat în stivă apar numai simboluri terminale însă trebuie ținut cont şi de pozițiile în care există neterminale care, deşi nu sunt nominalizate, au totuşi locuri rezervate în stiva de analiză. Ca atare, erorile de tipul 1) constau în negăsirea unei producții care să aibă în partea dreaptă aceleaşi terminale şi în aceeaşi ordine ca şi cele din capăt şi aceleaşi poziții pentru neterminale. Pentru aceasta, algoritmul de ASPO prezentat anterior trebuie completat, în porțiunea de tratare a reducerilor şi cu verificarea corespondenței între capătul extras din stivă şi partea dreaptă a unei producții. Această operație are ca efect şi localizarea producției pe baza căreia se face reducerea, operație esențială la analiza semantică, deoarece regulile semantice sunt asociate producțiilor şi ele se execută în timpul reducerilor.

Alte situații de eroare, în afara celor semnalate la punctele 1) şi 2) de mai sus nu pot să apară. De exemplu, la extragerea din stivă, există certitudinea că se va localiza extremitatea stângă a capătului (apariția unei relații $\lessdot$), cel târziu între $-ul de la baza stivei şi primul simbol situat deasupra lui, deoarece $ $\lessdot$ $\forall$ simbol. Ca atare, extragerea din stivă, operație care reprezintă efectiv reducerea, se va putea finaliza cu certitudine.

Tratarea erorilor în timpul reducerilor (tipul 1)

În momentul detectării unei erori de tipul 1), pentru a emite un mesaj potrivit, trebuie căutată o parte dreaptă a unei producții, asemănătoare cu capătul extras.

Exemple:

a) Dacă s-a extras un capăt ce conține terminalele **abc** şi există o parte dreaptă de forma **aEcE** în care apar terminalele **ac** dar nu şi **b**, se poate emite mesajul: "**b** ilegal în linia ..."

b) În altă situație se poate cere să inserăm un terminal pentru a găsi corespondența cu partea dreaptă:

abE**d**c "lipseşte **d** în linia ..."

c) În situația aEbc iar capătul extras nu solicită prezența unui neterminal între a şi b, se poate emite mesajul "lipseşte E în linia ..."

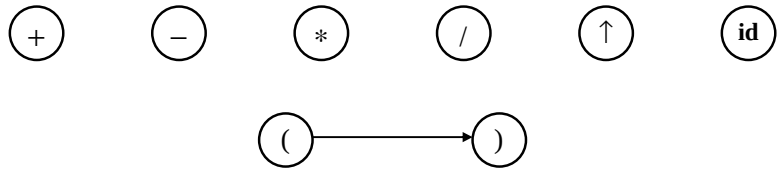
Posibilitatea de a emite un mesaj de eroare exact în situația în care capătul nu corespunde cu partea dreaptă a nici unei producții din gramatică este direct legată de existența unui număr finit sau infinit de şiruri care pot fi extrase ca şi capete. Considerând un astfel de şir, notat $b_1\ b_2\ ...\ b_k$, dacă el este capăt, atunci între simbolurile adiacente care îl compun este valabilă doar relația de precedență $\doteq$, adică $b_1 \doteq b_2 \doteq ... \doteq b_k$. Dacă din analiza tabelei de precedență a operatorilor rezultă existența unui număr finit de secvențe de terminale legate prin $\doteq$, atunci fiecare astfel de caz se poate trata distinct şi, având ca element de referință partea dreaptă cea mai apropiată, se poate elabora un mesaj de eroare potrivit.

Pentru a evidenția şirurile care pot fi extrase din stivă ca şi capete se va construi un graf orientat ale cărui noduri sunt terminalele gramaticii iar arcele leagă nodurile între care există relația $\doteq$. Capetele posibile sunt toate şirurile date de etichetele nodurilor care formează drumuri în acest graf. În plus, pentru ca un drum $b_1\ b_2\ ...\ b_k$ să poată fi capăt, trebuie să existe două simboluri notate *a* şi *c* (terminale sau $) astfel ca $a \lessdot b_1$ şi respectiv $b_k \gtrdot c$. Dacă respectă şi această condiție, nodurile corespunzătoare lui b_1 şi b_k se vor numi **nod inițial** şi respectiv **nod final**. Cu aceste notații, concluzia va fi următoarea: dacă în graful construit există cel puțin un drum care să conțină un ciclu, atunci numărul posibil de capete este infinit, astfel este finit.

Exemplu: Se consideră gramatica:

E→E+E | E-E | E ∗E | E/E | E↑E | (E) | **id**

a cărei tabelă de precedență a fost dată anterior. Graful corespunzător este următorul:



Singura pereche de noduri legate este cea corespunzătoare parantezelor (singurul $\doteq$ din tabelă). Toate nodurile, mai puțin ")", pot fi noduri inițiale şi toate nodurile, mai puțin "(", pot fi noduri finale. Deci există următoarele drumuri (în număr finit) de la un nod inițial la unul final: +, –, ∗, /, ↑, **id** de lungime 1 şi ( ) de lungime 2 şi toate corespund terminalelor din partea dreaptă a unei producții. Ca atare, rutina de tratare a erorilor în timpul reducerilor trebuie să verifice doar existența marcajelor de neterminale, pe pozițiile corecte, între şirurile de terminale care se reduc. Pot exista următoarele situații:

1) Dacă se reduce un capăt care conține unul din terminalele +, –, ∗, /, ↑, atunci trebuie verificat că apare câte un neterminal în ambele părți ale terminalului. Altfel, se va emite mesajul: "Lipseşte operand în stânga/dreapta operatorului ...".

2) Dacă se reduce **id**, atunci trebuie verificat să nu apară neterminal nici la stânga şi nici la dreapta sa. Dacă apare un neterminal, se emite mesajul "Lipseşte operator la stânga/dreapta identificatorului ...".

3) Dacă se reduce capătul ( ), atunci pot apărea următoarele situații de eroare:

a) Nu există neterminal între paranteze. Se poate emite mesajul: "Lipseşte expresie între ( )"

b) Există neterminale adiacente în exterior cu parantezele, de o parte sau de alta. Se pot da mesaje similare cu cel de la punctul 2): "Lipseşte operator la stânga (sau la dreapta)"

Dacă numărul de şiruri ce pot fi extrase ca şi capete este infinit, mesajele de eroare nu se pot stabili cu exactitate. În aceste cazuri, rutina de tratare a erorilor ar putea determina dacă există o parte dreaptă apropiată de şirul extras (de exemplu, la distanța de 1 sau 2, unde distanța înseamnă numărul de atomi inserat, şters sau schimbat). Dacă există o astfel de producție, se poate emite un mesaj concret, în sensul că se aşteaptă să apară acea parte dreaptă. Dacă nici o parte dreaptă nu este suficient de aproape de capăt, nu se poate emite decât un mesaj general, de genul: "Eroare de sintaxă în linia ...".

Tratarea erorilor deplasare – reducere (tipul 2)

Dacă la consultarea matricei de precedență se constată că, între simbolul din stivă şi cel din intrare, nu există relație de precedență atunci, pentru a continua analiza, trebuie să se modifice fie stiva, fie intrarea, fie ambele structuri. Modificarea va consta în inserarea, ştergerea sau înlocuirea unui simbol. La inserări sau ştergeri există pericolul de a se provoca un ciclu infinit. De exemplu, inserarea repetată de simboluri în intrare, fără a putea reduce sau deplasa simbolurile inserate.

Pentru a preveni ciclurile infinite, rutina de tratare a erorii trebuie să garanteze că, după modificare şi revenire, simbolul curent de intrare poate fi deplasat în stivă. Pentru cazul particular când, inițial, simbolul curent de intrare este $, se evită inserarea de simboluri în intrare şi se reduce stiva. Să considerăm configurația:

STIVA	**INTRARE**
... ab	cd ...$

Configurația conduce la eroare întrucât între b şi c nu există relație de precedență. Pot exista următoarele variante de tratare a erorii:

a) Dacă $a \lessdot c$ (înseamnă $<$ sau $\doteq$), se poate extrage **b** din stivă.

b) Dacă $b \lessdot d$, se şterge *c* din intrare.

c) Se caută un simbol *e*, astfel încât $b \lessdot e \lessdot c$ care să se insereze în intrare, în fața lui *c*. Dacă nu există un astfel de simbol, se poate căuta pentru inserare un şir de simboluri $e_1, e_2, ..., e_n$ astfel ca $b \lessdot e_1 \lessdot e_2 \lessdot ... \lessdot e_n \lessdot c$.

Acțiunea exactă ce se alege reflectă, în general, intuiția proiectantului compilatorului relativ la cea mai probabilă eroare din fiecare caz. Pentru fiecare intrare necompletată din matricea de precedență trebuie să se specifice o rutină de revenire din erori (nu neapărat distinctă de cea corespunzătoare altor intrări) în care se execută modificarea adecvată a stivei şi/sau a intrării şi se emite mesajul de eroare.

Exemplu: Se consideră din nou matricea de precedență din **§ 5. 2. 3**. Se rețin din această matrice doar liniile şi coloanele pe care apar intrări vide, completate cu numele rutinelor de tratare a erorilor:

	id	(	)	$
id	e_3	e_3	$\gtrdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\doteq$	e_4
)	e_3	e_3	$\gtrdot$	$\gtrdot$
$	$\lessdot$	$\lessdot$	e_2	e_1

În esență, cele patru rutine cuprind următoarele operații:

e_1: Se apelează dacă lipseşte întreaga expresie.
Se tratează prin inserarea unui **id** în intrare.
Mesajul: “Lipseşte operand”.

e_2: Se apelează când expresia începe cu o “)”.
Se tratează prin ştergerea “)” din intrare.
Mesajul: “) fără pereche – în plus”.

e_3: Se apelează când **id** sau “(” urmează după **id** sau “)”.
Se tratează prin inserarea unui operator, de ex. “+”, în intrare.
Mesajul: “Lipseşte operator”.

e_4: Se apelează când expresia se termină cu “(“.
Se tratează prin extragerea “(“ din stivă.
Mesajul: “Lipseşte paranteză dreaptă”.

Pentru a analiza modul de funcționare al acestui mecanism, să considerăm şirul de intrare eronat **id** +). După primele acțiuni (deplasarea lui **id**, reducerea la E şi deplasarea lui “+”) se ajunge în următoarea configurație:

STIVA	**INTRARE**
$E	)$

Deoarece + $\gtrdot$) urmează o reducere a capătului “+” iar, în timpul reducerii, se observă absența lui E din dreapta operatorului. Mesaj: “Lipseşte operand în dreapta operatorului +”. După efectuarea reducerii, se ajunge la configurația:

STIVA	**INTRARE**
$E	)$

Pentru $ şi) nu există relație de precedență şi se apelează rutina **e_2**. Mesaj: “) fără pereche – în plus” Prin eliminarea “)” din intrare, se ajunge la configurația finală a analizorului:

STIVA	**INTRARE**
$E	$