# Analiza sintactică ascendentă

● Un tip general de analiză numit **cu deplasare şi reducere (ASDR)**. Aceasta încearcă să construiască arborele sintactic pentru şirul de intrare începând de la frunze spre rădăcină (de jos în sus–ascendent). Procesul se poate privi ca reducerea unui şir de terminale, "w", la simbolul de start al gramaticii.

**Exemplu**:

$S \to aABe$ $\quad A \to Abc \mid b$ $\quad B \to d$

Propoziția "abbcde" poate fi redusă la S în următorii paşi:

abbcde $\quad$ aAbcde $\quad$ aAde $\quad$ aABe $\quad$ S

Şirul de reduceri urmăreşte, în sens invers, următoarea derivare dreapta:

$S \underset{d}{\Rightarrow} aABe \underset{d}{\Rightarrow} aAde \underset{d}{\Rightarrow} aAbcde \underset{d}{\Rightarrow} abbcde$

# Noțiuni de bază

● Un **capăt al unui şir** este un subşir care corespunde părții drepte a unei producții şi a cărui reducere la neterminalul din partea stângă a producției reprezintă un pas în parcurgerea în sens invers a unei derivări dreapta. În multe cazuri, subşirul stâng **β** care corespunde părții drepte a unei producții $A \to \beta$ nu este un **capăt** deoarece o reducere prin producția $A \to \beta$ produce un şir care nu poate fi redus în continuare, până la simbolul de start al gramaticii.

**Exemplu**: înlocuirea lui *b* cu A în pasul 2 al exemplului de mai sus

$aAbcde \to aAAcde$ (care nu poate fi redus la S)

⇒ Trebuie dată o definiție mai precisă pentru noțiunea de "capăt".

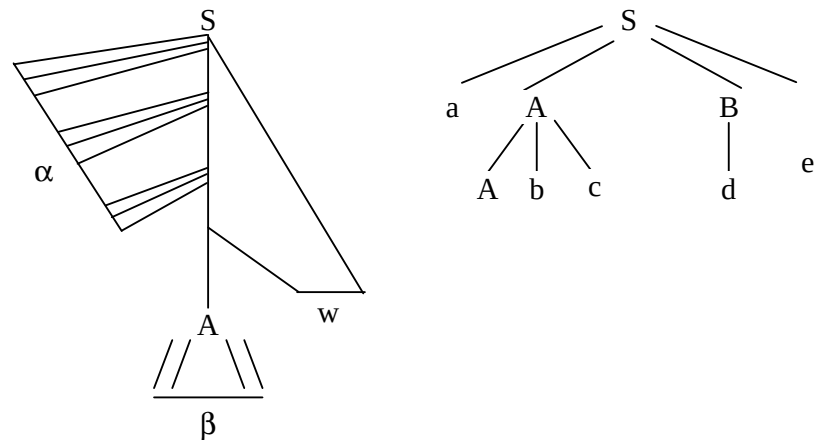
● Un **capăt** al unei forme propoziționale drepte $\gamma$ constă, formal dintr-o **producție** $A \to \beta$ şi o **poziție** în $\gamma$ în care şirul $\beta$ poate fi găsit şi înlocuit prin A pentru a produce forma propozițională anterioară a unei derivări dreapta a lui $\gamma$.

● Dacă $S \overset{*}{\underset{d}{\Rightarrow}} \alpha Aw \underset{d}{\Rightarrow} \alpha\beta w$, atunci $A \to \beta$ şi poziția care urmează lui $\alpha$, defineşte un capăt al lui $\alpha\beta w$. Subşirul *w*, situat în dreapta unui capăt, conține numai terminale.

● În cazul în care gramatica este ambiguă, este posibil să existe mai multe derivări dreapta ale şirului $\alpha\beta w$ ca atare pot exista mai multe capete. Dacă gramatica este neambiguă, capătul corespunzător unei forma propoziționale este unic.



Grafic, un capăt $\beta$, al unei forme propoziționale $\alpha\beta w$, se poate reprezenta astfel:




Capătul $A \to \beta$ în Arborele Sintactic pentru $\alpha\beta w$

● Capătul reprezintă cel mai din stânga subarbore complet format dintr-un nod şi toți fii săi. A este nodul interior cel mai de jos şi cel mai din stânga având toți fiii prezenți în arbore. Reducerea lui $\beta$ la *A* în $\alpha\beta w$, numită **fasonarea** capătului $\beta$, constă în îndepărtarea fiilor lui A din arborele sintactic.

**Exemplu**. Se consideră gramatica:

(1) $E \to E+E$ $\quad$ (2) $E \to E*E$ $\quad$ (3) $E \to (E)$ $\quad$ (4) $E \to \mathbf{id}$

şi derivarea dreaptă:

$E \underset{d}{\Rightarrow} \underline{E+E} \underset{d}{\Rightarrow} E+\underline{E*E} \underset{d}{\Rightarrow} E+E*\underline{\mathbf{id}_3} \underset{d}{\Rightarrow} E+\underline{\mathbf{id}_2}*\mathbf{id}_3 \underset{d}{\Rightarrow} \underline{\mathbf{id}_1}+\mathbf{id}_2*\mathbf{id}_3$

Sublinierile marchează capetele din fiecare formă propozițională.

Gramatica dată fiind ambiguă, se poate obține acelaşi şir şi prin următoarea derivare dreapta:

$E \underset{d}{\Rightarrow} \underline{E*E} \underset{d}{\Rightarrow} E*\underline{\mathbf{id}_3} \underset{d}{\Rightarrow} \underline{E+E}*\mathbf{id}_3 \underset{d}{\Rightarrow} E+\underline{\mathbf{id}_2}*\ \mathbf{id}_3 \underset{d}{\Rightarrow} \underline{\mathbf{id}_1}+\mathbf{id}_2*\mathbf{id}_3$

În forma propozițională $E+E*\mathbf{id}_3$ sunt două capete, $\underline{E+E}$ şi $\underline{\mathbf{id}_3}$, corespunzând primei şi respectiv celei de-a doua derivări prezentate. Ca şi în exemplul prezentat la gramatici ambigue, existența celor două derivări semnifică priorități relative diferite ale operațiilor de $*$ şi + (în primul caz, $*$ mai prioritar ca +).

## Fasonarea capetelor

Procesul de parcurgere în sens invers a unei derivări dreapta se numeşte **fasonarea capetelor**. Se porneşte de la un şir de terminale, notat cu **w**, care urmează să fie analizat. Dacă w aparţine limbajului descris prin gramatica dată atunci el se poate obţine după un pas oarecare **n** într-o derivare dreapta încă necunoscută:

$$S=\gamma_0 \underset{d}{\Rightarrow} \gamma_1 \underset{d}{\Rightarrow} \gamma_2 \underset{d}{\Rightarrow} \dots \underset{d}{\Rightarrow} \gamma_{n-1} \underset{d}{\Rightarrow} \gamma_n = w$$

Pentru a reconstitui această derivare în ordine inversă, se localizează capătul $\beta_n$ în forma propoziţiei $\gamma_n$, după care se înlocuieşte $\beta_n$ cu partea stângă a unei producţii $A_n \rightarrow \beta_n$, obţinându-se forma propoziţională dreapta $\gamma_{n-1}$. Apoi se continuă acest proces pentru $\gamma_{n-1}$, ş.a.m.d. Procesul se termină cu succes dacă, în final, se obţine ca formă propoziţională simbolul de start **S**.

**Exemplu**: se consideră gramatica din exemplul anterior şi şirul de intrare $\mathbf{id_1+id_2*id_3}$. Secvenţa de reduceri prezentată în figură este inversul secvenţei din prima derivare a exemplului anterior (cu $*$ prioritar faţă de +).

| Forma propoziţională dreapta | Capăt | Producţia pentru reducere |
|---|---|---|
| $\mathbf{id_1+id_2*id_3}$ | $\mathbf{id_1}$ | $E \rightarrow \mathbf{id}$ |
| $E+\mathbf{id_2*id_3}$ | $\mathbf{id_2}$ | $E \rightarrow \mathbf{id}$ |
| $E+E*\mathbf{id_3}$ | $\mathbf{id_3}$ | $E \rightarrow \mathbf{id}$ |
| $E+E*E$ | $E*E$ | $E \rightarrow E*E$ |
| $E+E$ | $E+E$ | $E \rightarrow E+E$ |
| $E$ | | |

Reduceri efectuate de un analizor sintactic cu deplasare şi reducere



## Implementarea cu stivă a analizei sintactice de tip deplasare – reducere (ASDR)

Pentru implementarea AS prin fasonarea capetelor, trebuie rezolvate două probleme:

- localizarea subşirului care urmează să fie redus, într-o formă propoziţională dreaptă;
- alegerea producţiei care se aplică, dacă gramatica are mai multe producţii cu aceeaşi parte dreaptă.

Ca structură de date de bază se poate utiliza pentru implementare o stivă în care se păstrează simbolurile gramaticale şi un tampon de intrare care conţine şirul de analizat, **w**. Baza stivei şi extremitatea dreaptă a şirului de intrare va fi marcată prin simbolul $. Iniţial, stiva este goală (conţine doar $) iar la intrare este şirul **w** întreg:

| STIVA | INTRARE |
|---|---|
| $ | w$ |

ANSIN deplasează în stivă zero sau mai multe simboluri de intrare până când în vârful stive apare un capăt, $\beta$. Apoi se reduce $\beta$ la partea stângă a producţiei corespunzătoare şi se continuă în mod ciclic aceste operaţii până când se detectează o eroare sau până când stiva conţine doar simbolul de start iar intrarea este vidă:

| STIVA | INTRARE |
|---|---|
| $S | $ |

În acest caz AS se opreşte şi semnalează terminarea cu succes a analizei.

**Exemplu**. Se urmăresc pas cu pas acţiunile ANSIN deplasare – reducere la analiza şirului de intrare $id_1+id_2*id_3$ (gramatica expresiilor din exemplul anterior).

| STIVA | INTRARE | ACŢIUNE |
|---|---|---|
| (1) $ | $id_1+id_2*id_3$ | deplasare |
| (2) $id_1 | $+id_2*id_3$ | reducere $E \rightarrow \mathbf{id}$ |
| (3) $E | $+id_2*id_3$ | deplasare |
| (4) $E+ | $id_2*id_3$ | deplasare |
| (5) $E+id_2 | $*id_3$ | reducere $E \rightarrow \mathbf{id}$ |
| (6) $E+E | $*id_3$ | deplasare |
| (7) $E+E* | $id_3$ | deplasare |
| (8) $E+E*id_3 | $ | reducere $E \rightarrow \mathbf{id}$ |
| (9) $E+E*E | $ | reducere $E \rightarrow E*E$ |
| (10) $E+E | $ | reducere $E \rightarrow E+E$ |
| (11) $E | $ | acceptă |

Configuraţiile ANSIN deplasare – reducere pentru intrarea $id_1+id_2*id_3$

ANSIN cu deplasare şi reducere execută următoarele patru acțiuni:

1) **Deplasare**: Simbolul de intrare următor este introdus în vârful stivei.

2) **Reducere**: Se realizează în situația în care extremitatea dreaptă a unui capăt se află în vârful stivei. Analizorul trebuie să localizeze extremitatea sa stângă, să stabilească neterminalul cu care se înlocuieşte capătul şi să realizeze efectiv reducerea lui.

3) **Acceptare**: Analizorul semnalează terminarea cu succes a analizei.

4) **Eroare**: Analizorul descoperă apariția unei erori şi se apelează o rutină de revenire.

Există o motivație importantă care justifică utilizarea stivei la acest tip de analiză sintactică: capătul va fi întotdeauna în vârful stivei şi nu în interior. Modalitatea de alegere a acțiunii, astfel încât analizorul să lucreze corect, depinde de tipul concret de analiză.

## Prefixe viabile

Mulțimea prefixelor formelor propoziționale dreapta care pot apărea în stiva unui ASDR se numesc **prefixe viabile**. Altfel spus, un prefix viabil este un prefix al unei forme propoziționale dreapta care nu continuă dincolo de extremitatea dreaptă a celui mai din dreapta capăt al acelei forme propoziționale. Conform acestei definiții este întotdeauna posibil să se adauge simboluri terminale la extremitatea unui prefix viabil, pentru a obține o formă propozițională dreapta. De aceea, pe parcursul analizei, nu va apărea nici o eroare atâta timp cât porțiunea din intrare văzută până la un anumit punct poate fi redusă la un prefix viabil.



# Analizoare sintactice LR

Tehnica LR reprezintă o metodă eficientă de AS ascendentă care poate fi aplicată pentru o clasă largă de GIC. Tehnica este numită ASLR(k). Pentru k=1, el poate fi omis.

**Principalele avantaje** ale analizei LR sunt:
- se pot elabora analizoare LR practic pentru toate construcțiile de limbaje exprimabile prin GIC;
- deşi este cea mai generală metodă fără reveniri, dintre cele cunoscute, ASLR poate fi implementată la fel de eficient ca şi alte metode de tipul deplasare – reducere;
- un ASLR poate detecta o eroare imediat ce este posibil, într-o baleiere de la stânga la dreapta a şirului de intrare (mai repede şi mai exact decât alte metode).

**Principalele dezavantaje** ale metodei sunt:
- cantitatea relativ mare de memorie necesară;
- volumul mare de muncă pentru construcția manuală a unui analizor, pentru gramatica unui limbaj de programare tipic.

Există însă în prezent generatoare de ASLR care primesc la intrare o GIC şi produc la ieşire tabelele de analiză necesare analizorului. Dacă gramatica conține ambiguități sau alte construcții dificil de analizat într-o baleiere stânga – dreapta a intrării, generatorul poate localiza aceste construcții şi poate informa proiectantul compilatorului.

În prezent sunt utilizate **trei variante de construire a tabelei** de ASLR pentru o gramatică:

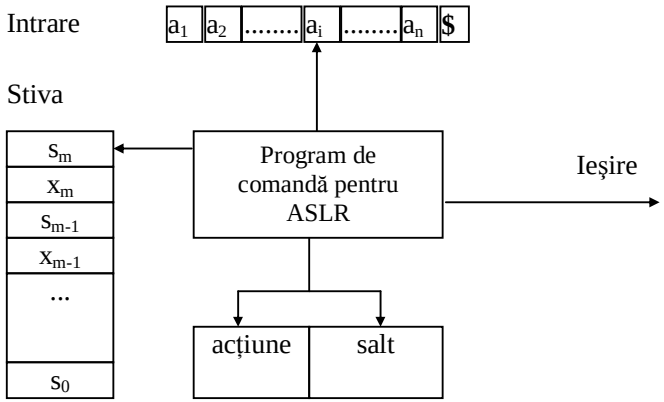
1) LR simplu (SLR): cea mai uşor de implementat dar şi cea mai slabă (pot fi analizate doar o clasă mai redusă de gramatici);

2) LR canonic (LR): cea mai puternică dar şi cea mai costisitoare (efort + memorie);

3) LR cu anticipare (LALR): varianta de compromis, intermediară; corespunde cel mai bine LP uzuale.

# Algoritmul de analiză sintactică LR

Schema de principiu a unui ASLR este următoarea:




Modelul unui ASLR

Un ASLR constă din:
- tampon de intrare
- stivă
- un program de comandă
- o tabelă de analiză cu două părți (acțiune, salt)
- ieşire

Programul de comandă este acelaşi pentru toate ASLR; ceea ce diferă de la o variantă de analizor la alta sunt doar tabelele de analiză. Programul de analiză citeşte, unul câte unul, caracterul din tamponul de intrare. În stivă se memorează şiruri de forma:

$s_0\ x_1\ s_1\ x_2\ s_2 \ldots x_m\ s_m$ cu sau $s_m$ în vârf

Fiecare $x_i$ este un simbol gramatical şi fiecare $s_i$ este un simbol numit **stare**. Se consideră că fiecare stare conține, în rezumat, informația prezentă în stivă sub acea stare. Combinația dintre simbolul de stare din vârful stivei ($s_m$) şi simbolul de intrare curent ($a_i$), utilizată ca indici în tabela de analiză, determină comportarea analizorului (decizia deplasare – reducere). Într-o implementare concretă, simbolurile gramaticale nu trebuie să apară în stivă (sunt suficiente stările). Prezența lor contribuie doar la înțelegerea comportării analizorului.



Programul de comandă lucrează astfel: La început se consultă *acțiune* $[s_m,a_i]$, care poate să conțină una din următoarele valori:

1) **deplasează** şi acoperă cu starea *s*;
2) **reduce** prin producția $A \rightarrow \beta$;
3) **acceptă** şirul;
4) **eroare**.

Tabela "salt" se comportă ca o funcție care primeşte ca argument o stare şi un simbol gramatical şi produce o stare nouă (valorile din această tabelă sunt stări). Această tabelă este funcția de tranziție a unu AFD care recunoaşte prefixele viabile ale gramaticii date, G. Starea inițială a acestui AFD este starea pusă inițial în vârful stivei ($s_0$).

Se numeşte **configurație** a unui ASLR o pereche în care prima componentă este conținutul momentan al stivei iar a doua componentă este partea rămasă neparcursă din intrare:

$$(s_0\ x_1\ s_1 \ldots x_m\ s_m, a_i\ a_{i+1} \ldots a_n\ \$)$$

Următoarea operație efectuată de analizor este determinată de citirea următorului simbol de intrare $a_i$, şi de starea din vârful stivei, $s_m$, care conduce la consultarea tabelei "acțiune". În funcție de conținutul elementului *acțiune* $[s_m,a_i]$, rezultă în final una din următoarele configurații:

1) Dacă *acțiune* $[s_m,a_i]$="deplasare *s*", se realizează, deplasarea lui $a_i$ din intrare în stivă, urmată de acoperirea lui $a_i$ în stivă, cu starea s=*salt*$[s_m,a_i]$; $a_{i+1}$ devine noul simbol de intrare. Prin aceste mişcări se obține configurația: $(s_0\ x_1\ s_1 \ldots x_m\ s_m a_i\ s, a_{i+1}\ a_{i+2} \ldots a_n\ \$)$
2) Dacă *acțiune* $[s_m,a_i]$="reducere $A \rightarrow \beta$" atunci se obține configurația: $(s_0\ x_1\ s_1 \ldots x_{m-r}\ s_{m-r}\ As, a_i\ a_{i+1} \ldots a_n\ \$)$ unde s= *salt*$[s_{m-r},A]$ iar r este $|\beta|$ (lungimea lui $\beta$). S-au extras 2r simboluri din stivă (**r** simboluri de stare +**r** simboluri gramaticale), expunând starea $s_{m-r}$, completând stiva cu A şi s. Simbolul de intrare curent nu este modificat. Ieşirea constă în executarea acțiunii semantice asociate cu producția de reducere (în exemplele de mai jos – o simplă tipărire a producției utilizate).
3) Dacă *acțiune* $[s_m,a_i]$="**acceptare**", analiza sintactică se termină.
4) Dacă *acțiune* $[s_m,a_i]$="**eroare**", se va apela o rutină de revenire din erori, iar configurația rezultată depinde de activitatea acestei rutine.

**Algoritmul de analiză** este prezentat schematic mai jos:

**Intrare**: un şir de intrare **w** şi o tabelă de ASLR cu funcțiile **acțiune** şi **salt** corespunzătoare gramaticii G.

**Ieşire**: Dacă **w**$\in$ L(G) $\Rightarrow$ o analiză ascendentă pentru w altfel $\Rightarrow$ o indicație de eroare.

**Metoda**: Se porneşte de la configurația:

$$(s_0, w\$)$$

$\uparrow$ în stivă $\uparrow$ în tamponul de intrare

după care se execută programul de mai jos, până se întâlneşte o acțiune de **acceptare** sau **eroare**.

```
∗ poziționează ip la primul simbol de intrare;
repeat forever begin
    ∗ fie s starea din vârful stivei şi a simbolul indicat de ip;
    if acțiune [s,a]=deplasează s' then begin
            ∗ introdu a apoi s' în vârful stivei; {deplasează şi acoperă cu s'}
            ∗avansează ip la următorul simbol de intrare
    end
    else if acțiune [s,a]=reduce A→β then begin
                ∗ extrage 2∗|β| simboluri din stivă;
                ∗ fie s' starea rămasă, după extragere, în vârful stivei;
                ∗ introdu A, apoi salt [s', A] în stivă;
                ∗ tipăreşte producția A→β
            end
                    else if acțiune [s,a]=acceptă then return
                            else eroare()
end
```

**Exemplu**: Se consideră gramatica pentru expresii:

(1) E→E+T
(2) E→T
(3) T→T∗F
(4) T→F
(5) F→(E)
(6) F→**id**

Tabelele (funcțiile) acțiune şi salt corespunzătoare sunt:

| stare | acțiune | | | | | | salt | | |
|---|---|---|---|---|---|---|---|---|---|
| | id | + | ∗ | ( | ) | $ | E | T | F |
| 0 | $d_5$ | | | $d_4$ | | | 1 | 2 | 3 |
| 1 | | $d_6$ | | | | acc | | | |
| 2 | | $r_2$ | $d_7$ | | $r_2$ | $r_2$ | | | |
| 3 | | $r_4$ | $r_4$ | | $r_4$ | $r_4$ | | | |
| 4 | $d_5$ | | | $d_4$ | | | 8 | 2 | 3 |
| 5 | | $r_6$ | $r_6$ | | $r_6$ | $r_6$ | | | |
| 6 | $d_5$ | | | $d_4$ | | | | 9 | 3 |
| 7 | $d_5$ | | | $d_4$ | | | | | 10 |
| 8 | | $d_6$ | | | $d_{11}$ | | | | |
| 9 | | $r_1$ | $d_7$ | | $r_1$ | $r_1$ | | | |
| 10 | | $r_3$ | $r_3$ | | $r_3$ | $r_3$ | | | |
| 11 | | $r_5$ | $r_5$ | | $r_5$ | $r_5$ | | | |

Codurile pentru acțiuni au următoarele semnificații:

1) $d_i$ – deplasează şi introdu în stivă starea *i*;
2) $r_j$ – reduce prin producția cu numărul *j*;
3) acc – acceptă;
4) spațiu – eroare

**Observație**: Valoarea lui *salt*[s,a], pentru a–terminal, se găseşte în *acțiune*[s,a], legată cu acțiunea de deplasare. În consecință, tabela *salt* are prevăzute coloane doar pentru neterminale.

Conținutul stivei şi al intrării precum şi acțiunile efectuate de analizor la analiza şirului de intrare **id**∗**id**+**id** sunt:

| STIVA | INTRARE | ACȚIUNE |
|---|---|---|
| (1) 0 | **id**∗**id**+**id**$ | depl (5) |
| (2) 0 **id** 5 | ∗**id**+**id**$ | red (6) : F→**id** |
| (3) 0 F 3 | ∗**id**+**id**$ | red (4) . T→F |
| (4) 0 T 2 | ∗**id**+**id**$ | depl (7) |
| (5) 0 T 2 ∗ 7 | **id**+**id**$ | depl (5) |
| (6) 0 T 2 ∗ 7 **id** 5 | +**id**$ | red (6) F→**id** |
| (7) 0 T 2 ∗ 7 F 10 | +**id**$ | red (3) T→T ∗ F |
| (8) 0 T 2 | +**id**$ | red (2) E→T |
| (9) 0 E 1 | +**id**$ | depl (6) : |
| (10) 0 E 1 + 6 | **id**$ | depl (5) |
| (11) 0 E 1 + 6 **id** 5 | $ | red (6) : F→**id** |
| (12) 0 E 1 + 6 F 3 | $ | red (4) . T→F |
| (13) 0 E 1 + 6 T 9 | $ | red (1) . E→E+T |
| (14) 0 E 1 | $ | acceptă |



## Gramatici LR

O gramatică pentru care se poate construi o tabelă de analiză sintactică pentru ASDR se numeşte gramatică LR. Există GIC care nu sunt LR dar, pentru construcțiile tipice ale limbajelor de programare uzuale, acestea pot fi în general evitate sau înlocuite prin producții echivalente care se încadrează în clasa LR. Intuitiv, pentru ca o gramatică să fie LR este suficient ca un ASDR, parcurgând şirul de intrare de la stânga la dreapta, să fie capabil să recunoască toate capetele, atunci când ele apar în vârful stivei. Pentru aceasta, un ASLR nu trebuie să baleieze întreaga stivă întrucât simbolul de stare din vârful stivei conținând toată informația necesară.

Alte surse de informații utilizate de ASLR în luarea deciziilor deplasare – reducere sunt următoarele *k* simboluri din intrare. O gramatică ce poate fi analizată de un ASLR ce examinează până la *k* simboluri de intrare la fiecare mişcare se numeşte gramatică LR(k). de interes practic sunt cazurile k=0 şi k=1 (Exemplu: tabela de acțiuni din exemplul anterior utilizează un simbol de anticipare, deci k=1).

Diferența de principiu între gramaticile LL şi LR este următoarea: pentru ca o gramatică să fie LR(k) analizorul trebuie să fie capabil să recunoască apariția în stivă a părții drepte a unei producții, în stivă fiind chiar acea parte dreaptă sau tot ce derivă din ea, plus *k* simboluri de anticipare în şirul de intrare; în cazul unei gramatici LL(k), analizorul trebuie să recunoască utilizarea unei producții doar pe baza primelor *k* simboluri din subşirul derivat în partea dreaptă a producției respective. Rezultă că cerința ca o gramatică să fie LR(k) este mult mai puțin restrictivă decât în cazul LL(k) iar clasa gramaticilor LR este mai cuprinzătoare decât a gramaticilor LL.