Teoria Compilării

Drăgulici Dumitru Daniel

Facultatea de matematică și informatică,
Universitatea București

2011

Cuprins

Analiza sintactică

Analiza sintactică (**parsarea**) are ca scop identificarea modului în care tokenii (furnizați de analizorul lexical) se agregă pentru a forma structuri sintactice (instrucțiuni și expresii), încuibate unele în altele, până ce formează structura sintactică de la nivelul de vârf (programul).

Sintaxa limbajelor de programare se definește și se investighează folosind instrumente de limbaje independente de context - de exemplu gramatici independente de context.

Gramatici independente de context

Definiție: **Gramatici independente de context (GIC)**:

O GIC este un sistem $G = \langle V_N, V_T, S, P \rangle$*, unde:*

V_N *este o mulțime finită nevidă (mulțimea* neterminalelor*);*

V_T *este o mulțime finită nevidă (mulțimea* terminalelor*);*

$S \in V_N$ *este* simbolul de start*;*

$P \subseteq V_N \times (V_N \cup V_T)^\star$ *este o mulțime finită (mulțimea* producțiilor*).*

Notăm $V_G = V_N \cup V_T$*.*

Uneori mulțimea $\{A \to \gamma_1, \ldots, A \to \gamma_n\}$ a tuturor producțiilor ce au în stânga neterminalul $A \in V_N$ (i.e. mulțimea **alternativelor**) lui A va fi notată compact $A \to \gamma_1 | \ldots | \gamma_n$.

Gramatici independente de context

Definiție: *Pentru o GIC G definim:*

- **relația de derivare directă**:

  $$\alpha, \beta \in V^\star,\ \alpha \Rightarrow \beta \text{ d.d. } \exists \left\{ \begin{array}{l} u, v, \gamma \in V_G^\star \\ N \in V_N \end{array} \right. \text{ a.î. } \left\{ \begin{array}{l} \alpha = uNv \\ \beta = u\gamma v \\ N \rightarrow \gamma \in P \end{array} \right.$$

  (deci β se obține din α prin înlocuirea unei apariții a lui N cu γ, în baza producției $N \rightarrow \gamma$);

- **relația de derivare stângă directă**:
  $\alpha, \beta \in V^\star$, $\alpha \Rightarrow_s \beta$ *d.d. la fel ca la* $\Rightarrow$, *dar* $u \in V_T^\star$ *(i.e. s-a derivat cel mai din stânga neterminal);*

- **relația de derivare dreaptă directă**:
  $\alpha, \beta \in V^\star$, $\alpha \Rightarrow_d \beta$ *d.d. la fel ca la* $\Rightarrow$, *dar* $v \in V_T^\star$ *(i.e. s-a derivat cel mai din dreapta neterminal).*

Gramatici independente de context

Definiție: *Pentru o GIC G definim:*

- **relația de derivare** $\stackrel{\star}{\Rightarrow}$ / **derivare stângă** $\stackrel{\star}{\Rightarrow}_s$ / **derivare dreaptă** $\stackrel{\star}{\Rightarrow}_d$: *închiderea reflexivă și tranzitivă a celei directe*
  (ex: $\alpha \stackrel{\star}{\Rightarrow} \beta$ d.d. $\alpha = \beta$ sau $\exists\ \alpha = \alpha_0, \alpha_1, \ldots, \alpha_k = \beta \in V_G^\star$, $k \geq 1$, a.î. $\alpha_0 \Rightarrow \alpha_1 \Rightarrow \ldots \Rightarrow \alpha_k$);
- *cu $+$ în loc de $\star$ (ex: $\stackrel{+}{\Rightarrow}$) notăm închiderea tranzitivă (i.e. în cel puțin un pas) a relației directe.*

Observație:

Toate relațiile definite mai sus sunt cazuri particulare ale relației de derivare.

Definiție: *Pentru o GIC G definim:*

- *o secvență de derivări directe $\alpha_0 \Rightarrow \cdots \Rightarrow \alpha_n$ s.n.* **derivare** *(a lui α_n din α_0); daca "$\Rightarrow$" este tot timpul "$\Rightarrow_s$", resp. "$\Rightarrow_d$", ea s.n.* **derivare stângă**, *resp.* **dreaptă**.
- **formă sentențială**: *șir din $V_G^\star$ care apare în cel puțin o derivare din S.*

Gramatici independente de context

Definiție:

Fiind dată o GIC G, un **arbore de derivare** *este un arbore a.î. :*

• *nodurile sunt etichetate cu elemente din* $V_G \cup \{\lambda\}$*;*

• *rădăcina este etichetată cu* S*;*

• *descendenții direcți (fii) ai oricărui nod, dacă există, formează o mulțime total ordonată;*

• *dacă fii unui nod etichetat cu* N *au în ordine etichetele* x_1*, ...,* x_n*, atunci* $N \to x_1 \dots x_n \in P$.

Gramatici independente de context

Observație:

• *Din definiție rezultă că nodurile neterminale sunt etichetate cu elemente din* V_N *(doar ele pot apărea în membrii stângi ai producțiilor prin care se obțin fii acestora) iar nodurile terminale (frunze) pot fi etichetate cu elemente atât din* V_N *cât și din* V_T *sau cu* λ.

• *Dacă* w *este cuvântul de pe frontiera arborelui (i.e. cuvântul obținut prin concatenarea etichetelor frunzelor în ordinea dată de parcurgerea depth first) atunci* $S \stackrel{\star}{\Rightarrow} w$.

• *Reciproc, dacă* $S \stackrel{\star}{\Rightarrow} w \in V_G^{\star}$, *atunci există cel puțin un arbore de derivare care are pe frontieră* w *- el arată un mod de obținere a lui* w *din* S.
La rândul său, un arbore de derivare poate fi parcurs în general în mai multe moduri (aplicăm producțiile în altă ordine) și fiecare parcurgere definește câte o derivare a lui w *din* S. *Printre acestea însă există o singură derivare stângă și o singură derivare dreaptă.*

• *Deci, pentru o GIC dată, există o bijecție între arborii de derivare, derivările stângi, derivările drepte.*

Gramatici independente de context

Definiție: *Fie G o GIC.*

- **Limbajul generat** *de G este $L(G) = \{w \in V_T^\star : S \stackrel{\star}{\Rightarrow} w\}$.*

- *G este* **ambiguă** *d.d. există $w \in L(G)$ care are ≥ 2 arbori de derivare distincți, i.e. are ≥ 2 derivări stângi distincte, i.e. are ≥ 2 derivări drepte distincte.*

- *Două gramatici sunt* **echivalente** *dacă generează același limbaj.*

Definiție:

Un limbaj L este **independent de context** *dacă există cel puțin o GIC G a.î. $L = L(G)$.*

Observație:

- *La fel ca în cazul limbajelor regulate, există și alte caracterizări, echivalente, ale limbajelor independente de context, de ex. să fie recunoscute de automate stivă (push down).*

- *Clasa limbajelor regulate este strict inclusă în clasa limbajelor independente de context.*

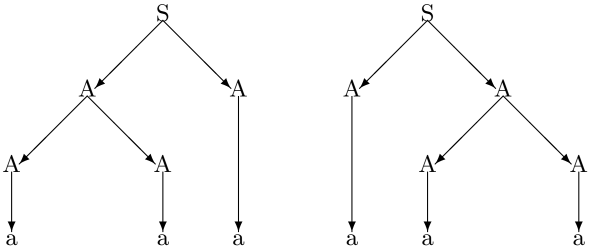
Gramatici independente de context

Exemplu:

Fie GIC G dată de:

$V_N = \{S, A\}$
$V_T = \{a\}$
simbolul de start S
$P = \{1: S \to AA, 2: A \to AA, 3: A \to a\}$ (am numerotat producţiile).

Fie $w = aaa \in V_T^\star$. Atunci $w \in L(G)$, iar doi arbori de derivare pentru w sunt:

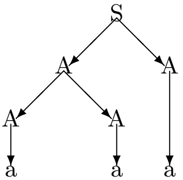


Deci G este ambiguă.

Gramatici independente de context

Exemplu:

Primul arbore defineşte următoarele derivări ale lui w din S (la fiecare pas am subliniat neterminalul înlocuit şi am notat producţia aplicată):



Producţii:
$1:\ S \to AA$
$2:\ A \to AA$
$3:\ A \to a$

$\underline{S} \xrightarrow{1} \underline{A}A \xrightarrow{2} \underline{A}AA \xrightarrow{3} a\underline{A}A \xrightarrow{3} aa\underline{A} \xrightarrow{3} aaa$ (derivare stângă)

$\underline{S} \xrightarrow{1} \underline{A}A \xrightarrow{2} \underline{A}AA \xrightarrow{3} aA\underline{A} \xrightarrow{3} a\underline{A}a \xrightarrow{3} aaa$

$\underline{S} \xrightarrow{1} \underline{A}A \xrightarrow{2} A\underline{A}A \xrightarrow{3} \underline{A}aA \xrightarrow{3} aa\underline{A} \xrightarrow{3} aaa$

$\underline{S} \xrightarrow{1} \underline{A}A \xrightarrow{2} A\underline{A}A \xrightarrow{3} Aa\underline{A} \xrightarrow{3} \underline{A}aa \xrightarrow{3} aaa$

$\underline{S} \xrightarrow{1} \underline{A}A \xrightarrow{2} AA\underline{A} \xrightarrow{3} \underline{A}Aa \xrightarrow{3} a\underline{A}a \xrightarrow{3} aaa$

$\underline{S} \xrightarrow{1} \underline{A}A \xrightarrow{2} AA\underline{A} \xrightarrow{3} A\underline{A}a \xrightarrow{3} \underline{A}aa \xrightarrow{3} aaa$

$\underline{S} \xrightarrow{1} A\underline{A} \xrightarrow{3} \underline{A}a \xrightarrow{2} \underline{A}Aa \xrightarrow{3} a\underline{A}a \xrightarrow{3} aaa$

$\underline{S} \xrightarrow{1} A\underline{A} \xrightarrow{3} \underline{A}a \xrightarrow{2} A\underline{A}a \xrightarrow{3} \underline{A}aa \xrightarrow{3} aaa$ (derivare dreaptă).

Analiza sintactică

Sintaxa limbajelor de programare se definește cu GIC neambigue, în care:
- neterminale sunt tipuri de structuri sintactice (program, declarație, instrucțiune, instrucțiune while, expresie, termen, factor, etc.);
-terminalele sunt tipuri de tokeni $\neq$ spațiu (identificator, întreg, operator aditiv, operator multiplicativ, etc.);
- simbolul de start este neterminalul corespunzător structurii sintactice de la nivelul cel mai înalt (program);
- producțiile sunt regulile/variantele de definire ale diverselor tipuri de structuri sintactice (definiția instrucțiunii while, definiția expresiei, etc.).

Cuvintele-candidat la a fi programe corecte sintactic sunt șiruri de tipuri de token (livrate de scanner).

Analizorul sintactic (parserul) verifică dacă acest cuvânt aparține limbajului generat de gramatică (i.e. este un program corect sintactic) și dacă da, construiește arborele său de derivare (acesta descrie modul de construire a programului). Arborele poate fi livrat de ex. sub forma șirului numerelor producțiior aplicate în derivarea stângă (sau dreaptă) asociată.

Analiza sintactică

Exemplu:

TODO

Analiza sintactică

Verificarea faptului că un cuvânt format din terminale aparține limbajului generat de o gramatică se face folosind un **algoritm de analiză sintactică** (**algoritm de parsare**). În caz afirmativ, algoritmul produce și o informație din care să se poată reconstitui arborele de derivare al cuvântului, de ex. șirul numerelor producțiior aplicate într-o derivare stângă (sau dreaptă) a acestuia din simbolul de start.

Există algoritmi de parsare:
- **top - down**: încearcă construirea arborelui de derivare de sus în jos (de la rădăcină spre frontieră);
- **bottom - up**: încearcă construirea arborelui de derivare de jos în sus (de la frontieră spre rădăcină).

Pentru o GIC oarecare (fără proprietăți suplimentare) în ambele cazuri algoritmii sunt exponențiali. Adăugând gramaticii proprietăți suplimentare, putem obține algoritmi mai buni, chiar liniari. Astfel sunt gramaticile de tip LL, LR, de precedență, etc. (toate sunt GIC cu proprietăți suplimentare, care permit elaborarea unor algoritmi de parsare liniari).

Cuprins

Algoritmi de parsare top - down

Algoritmii de parsare top - down încearcă construirea arborelui de derivare de sus în jos (de la rădăcină spre frontieră).

Cuprins

Algoritmul de parsare general top - down

Algoritmul de parsare general top - down nu necesită proprietăți suplimentare ale GIC.

Pentru a lucra organizat, se numerotează producțiile și la fiecare pas se încearcă aplicarea în ordine a acestora, pentru a găsi descendenții corecți ai celui mai din stânga nod neprocesat (în sensul parcurgerii depth first). Odată găsită producția corectă, se reține numărul acesteia.
Astfel, dacă cuvântul parsat aparține limbajului generat de gramatică, se obține în final o derivare stângă a sa.
Constatăm totodată că partea finalizată (formată numai din terminale) din stânga frontierei crește progresiv, iar în final ea trebuie să coincidă cu cuvântul parsat.

De aceea, pentru a eficientiza căutarea, la încercarea fiecărei producții se verifică dacă terminalele generate de ea în stânga porțiunii nefinalizate din frontieră coincid cu terminalele aflate în stânga porțiunii neconsultate încă din cuvânt, iar dacă nu, se respinge producția respectivă (nu are sens să construim arborele în continuare după această alegere).
Constatăm totodată că cuvântul parsat este consultat de la stânga la dreapta.

Algoritmul de parsare general top - down

Dacă pentru un nod neterminal s-au încercat fără succes toate producțiile posibile, se revine la nodul anterior (în parcurgere depth first) pentru a se încerca acolo producția următoare.
Astfel, algortimul general top - down efectuează backtracking, fiind deci exponențial.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1 : E \to T + E \quad 2 : E \to T \quad 3 : T \to F \star T \quad 4 : T \to F \quad 5 : F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

E

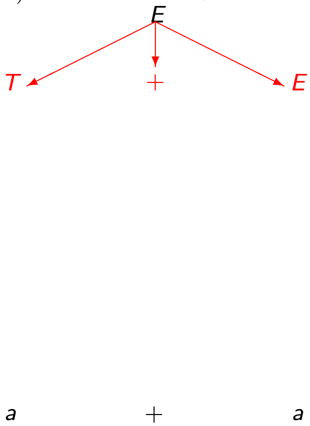
$a \quad + \quad a$

Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E, P: $1: E \to T + E \quad 2: E \to T \quad 3: T \to F \star T \quad 4: T \to F \quad 5: F \to a$ (am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

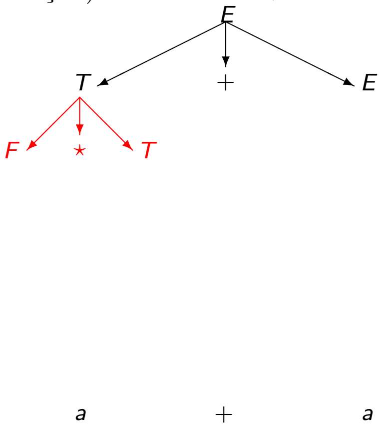


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1 : E \to T + E \quad 2 : E \to T \quad 3 : T \to F \star T \quad 4 : T \to F \quad 5 : F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

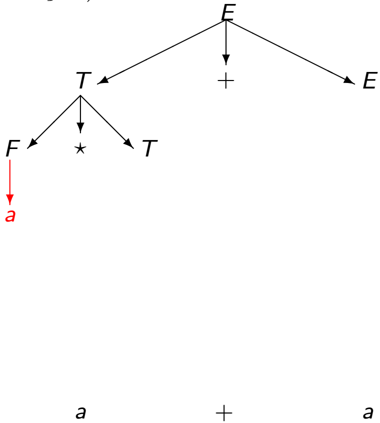


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1 : E \to T + E \quad 2 : E \to T \quad 3 : T \to F \star T \quad 4 : T \to F \quad 5 : F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

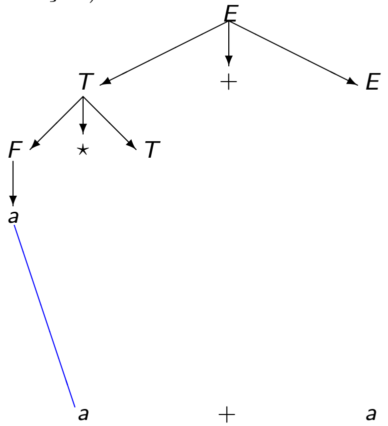


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1 : E \to T + E \quad 2 : E \to T \quad 3 : T \to F \star T \quad 4 : T \to F \quad 5 : F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

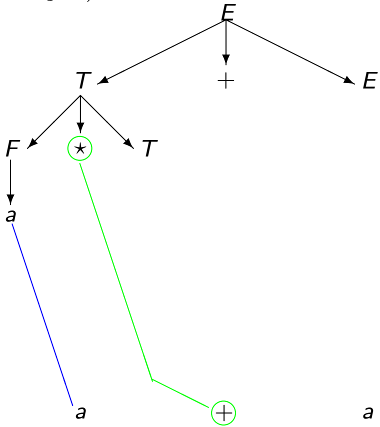


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E, P: $1: E \to T + E \quad 2: E \to T \quad 3: T \to F \star T \quad 4: T \to F \quad 5: F \to a$ (am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

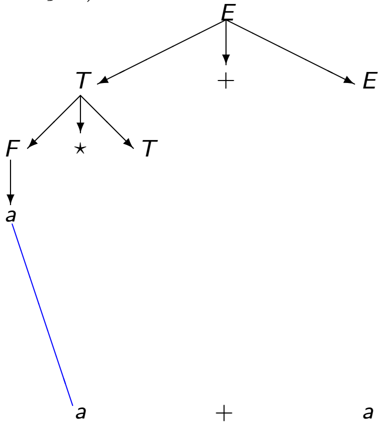


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1: E \to T + E \quad 2: E \to T \quad 3: T \to F \star T \quad 4: T \to F \quad 5: F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

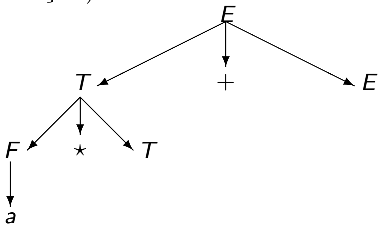


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1 : E \to T + E \quad 2 : E \to T \quad 3 : T \to F \star T \quad 4 : T \to F \quad 5 : F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:



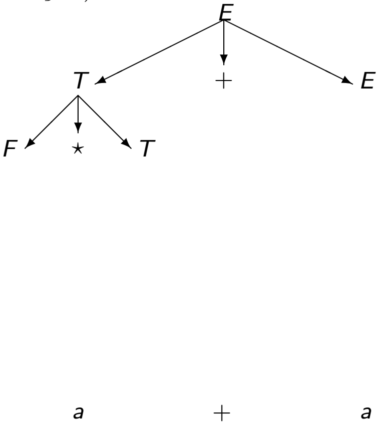
$a \qquad + \qquad a$

Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1 : E \to T + E \quad 2 : E \to T \quad 3 : T \to F \star T \quad 4 : T \to F \quad 5 : F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

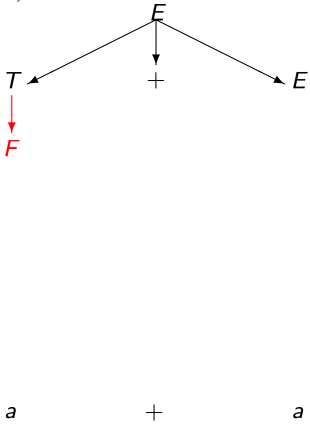


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1 : E \to T + E \quad 2 : E \to T \quad 3 : T \to F \star T \quad 4 : T \to F \quad 5 : F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

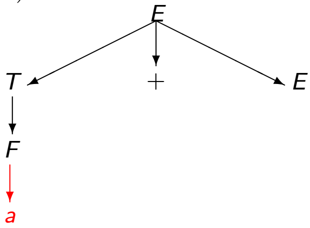


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1 : E \to T + E$ $\quad 2 : E \to T$ $\quad 3 : T \to F \star T$ $\quad 4 : T \to F$ $\quad 5 : F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:



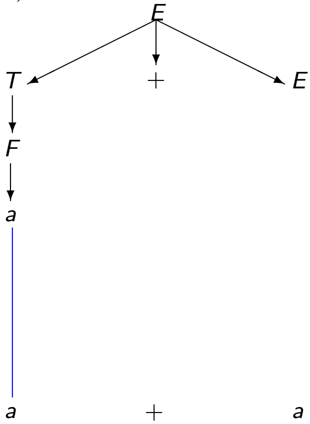
$a \qquad + \qquad a$

Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E, P: $1: E \to T + E \quad 2: E \to T \quad 3: T \to F \star T \quad 4: T \to F \quad 5: F \to a$ (am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

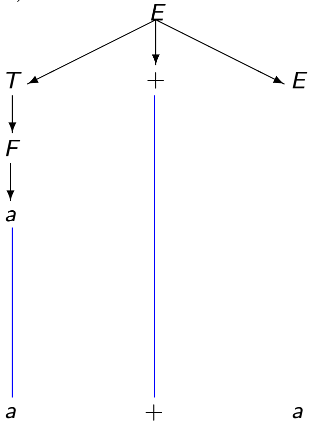


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1: E \to T + E \quad 2: E \to T \quad 3: T \to F \star T \quad 4: T \to F \quad 5: F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

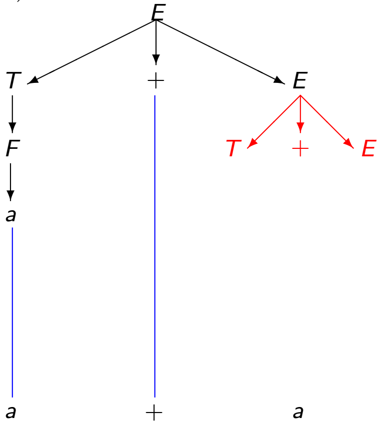


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E, P: $1: E \to T + E \quad 2: E \to T \quad 3: T \to F \star T \quad 4: T \to F \quad 5: F \to a$ (am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

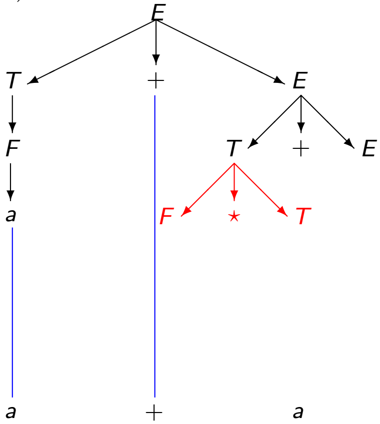


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1: E \to T + E \quad 2: E \to T \quad 3: T \to F \star T \quad 4: T \to F \quad 5: F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

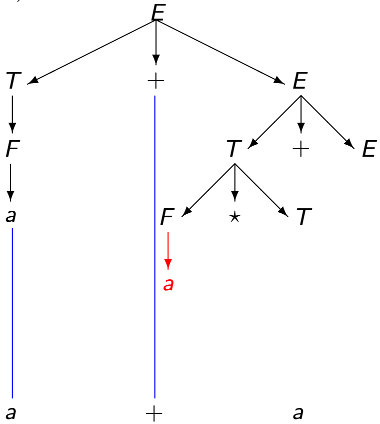


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1: E \to T + E \quad 2: E \to T \quad 3: T \to F \star T \quad 4: T \to F \quad 5: F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

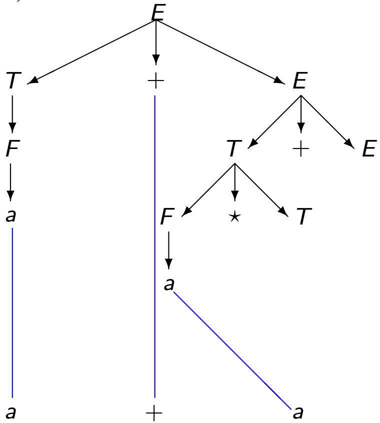


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E, P: $1: E \to T + E \quad 2: E \to T \quad 3: T \to F \star T \quad 4: T \to F \quad 5: F \to a$ (am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

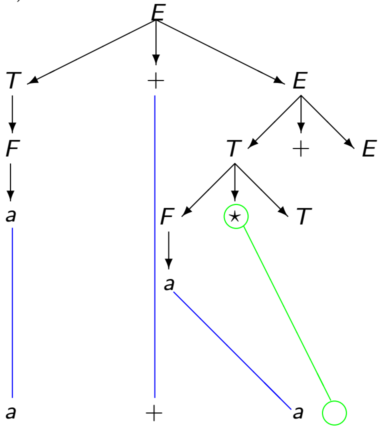


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1: E \to T + E$ $\quad 2: E \to T$ $\quad 3: T \to F \star T$ $\quad 4: T \to F$ $\quad 5: F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

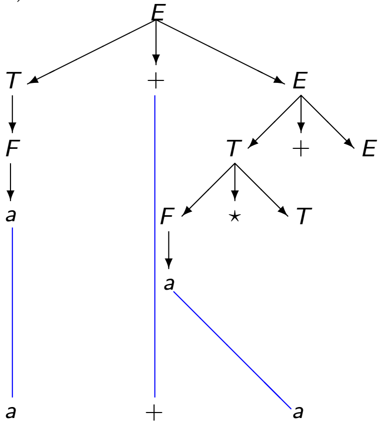


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1 : E \to T + E \quad 2 : E \to T \quad 3 : T \to F \star T \quad 4 : T \to F \quad 5 : F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

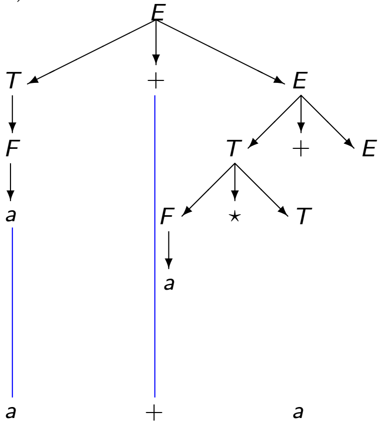


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1 : E \to T + E \quad 2 : E \to T \quad 3 : T \to F \star T \quad 4 : T \to F \quad 5 : F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

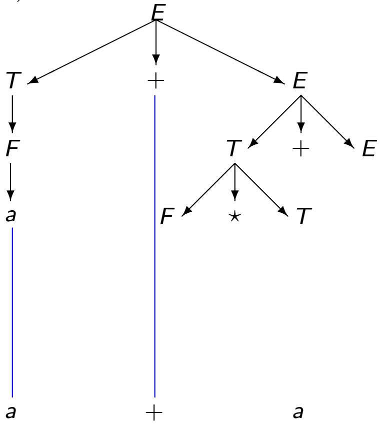


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1 : E \to T + E \quad 2 : E \to T \quad 3 : T \to F \star T \quad 4 : T \to F \quad 5 : F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

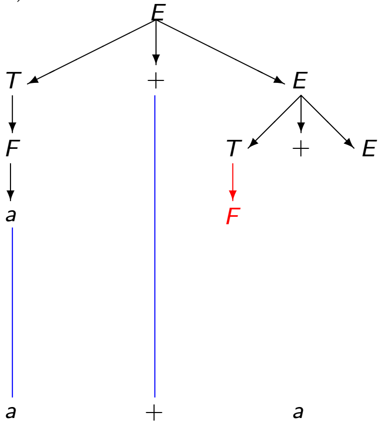


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1: E \to T + E \quad 2: E \to T \quad 3: T \to F \star T \quad 4: T \to F \quad 5: F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

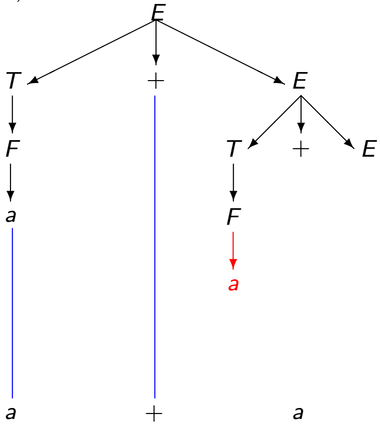


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E, P: $1: E \to T + E \quad 2: E \to T \quad 3: T \to F \star T \quad 4: T \to F \quad 5: F \to a$ (am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

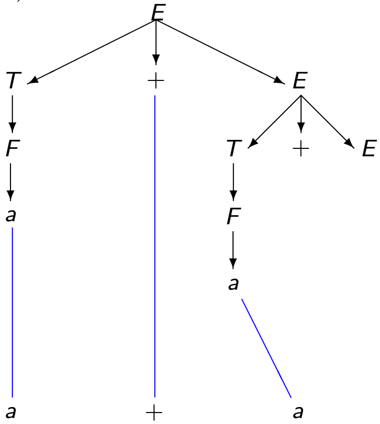


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E, P: $1: E \rightarrow T + E \quad 2: E \rightarrow T \quad 3: T \rightarrow F \star T \quad 4: T \rightarrow F \quad 5: F \rightarrow a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

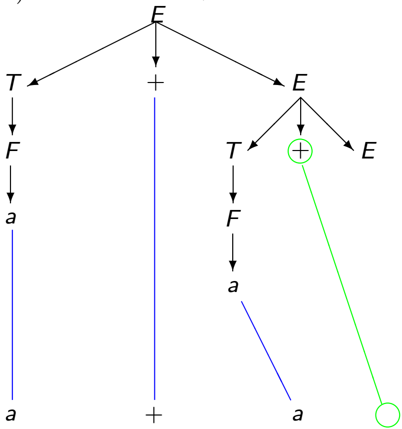


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1: E \to T + E \quad 2: E \to T \quad 3: T \to F \star T \quad 4: T \to F \quad 5: F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

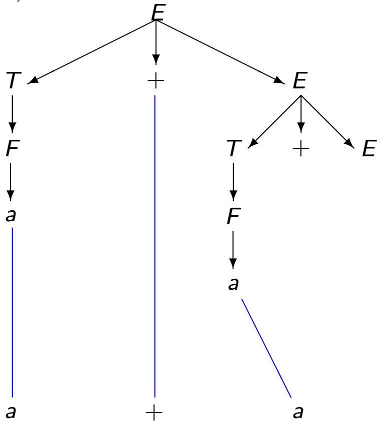


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1: E \to T + E \quad 2: E \to T \quad 3: T \to F \star T \quad 4: T \to F \quad 5: F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

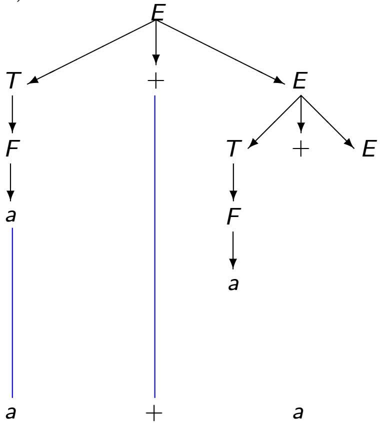


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1: E \to T + E \quad 2: E \to T \quad 3: T \to F \star T \quad 4: T \to F \quad 5: F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

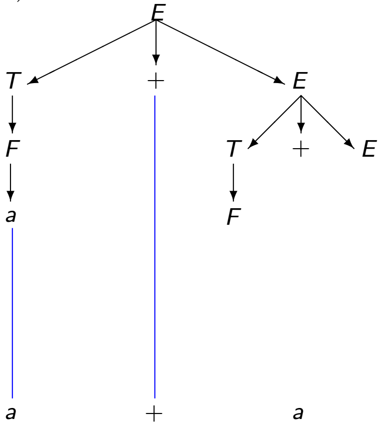


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E, P: $1: E \to T + E$ $\quad 2: E \to T$ $\quad 3: T \to F \star T$ $\quad 4: T \to F$ $\quad 5: F \to a$ (am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

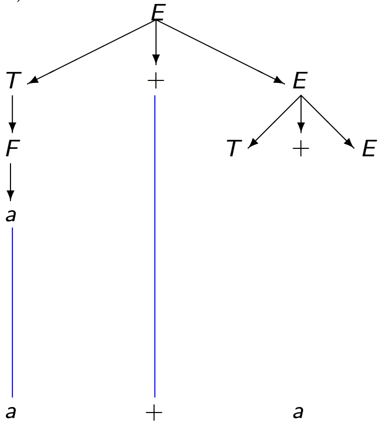


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1 : E \to T + E \quad 2 : E \to T \quad 3 : T \to F \star T \quad 4 : T \to F \quad 5 : F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

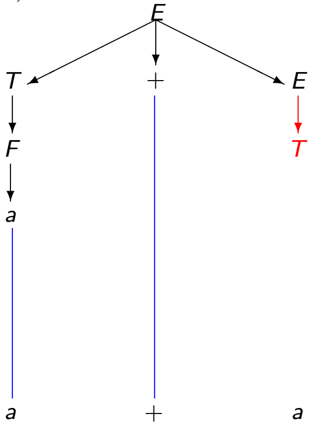


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1: E \to T + E$ $\quad 2: E \to T$ $\quad 3: T \to F \star T$ $\quad 4: T \to F$ $\quad 5: F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

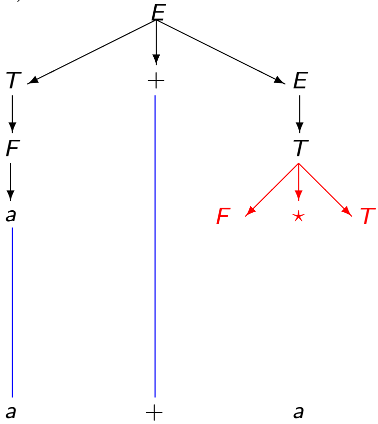


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1: E \to T + E \quad 2: E \to T \quad 3: T \to F \star T \quad 4: T \to F \quad 5: F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

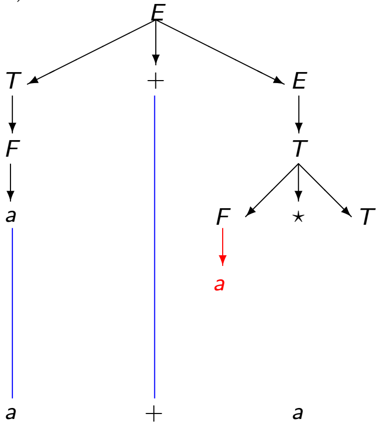


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1 : E \to T + E \quad 2 : E \to T \quad 3 : T \to F \star T \quad 4 : T \to F \quad 5 : F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

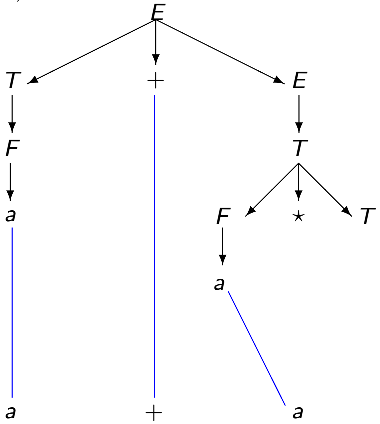


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1 : E \to T + E \quad 2 : E \to T \quad 3 : T \to F \star T \quad 4 : T \to F \quad 5 : F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

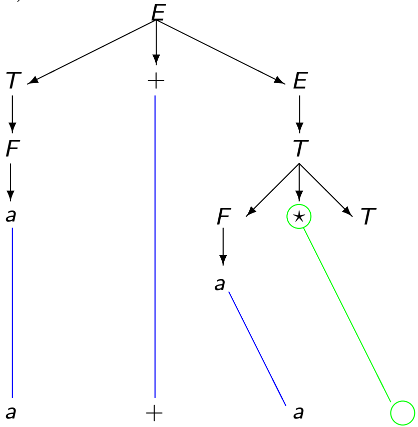


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E, P: $1 : E \to T + E$ $\quad 2 : E \to T$ $\quad 3 : T \to F \star T$ $\quad 4 : T \to F$ $\quad 5 : F \to a$ (am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

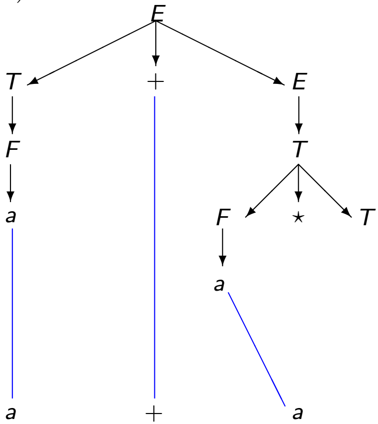


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1: E \to T + E \quad 2: E \to T \quad 3: T \to F \star T \quad 4: T \to F \quad 5: F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

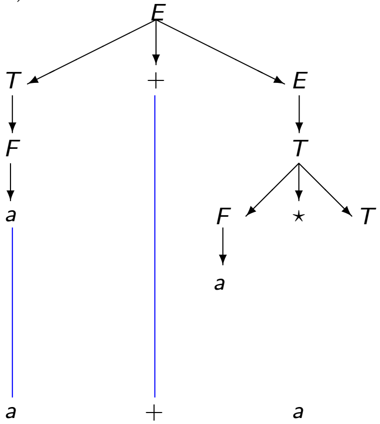


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E, P: $1: E \to T + E \quad 2: E \to T \quad 3: T \to F \star T \quad 4: T \to F \quad 5: F \to a$ (am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

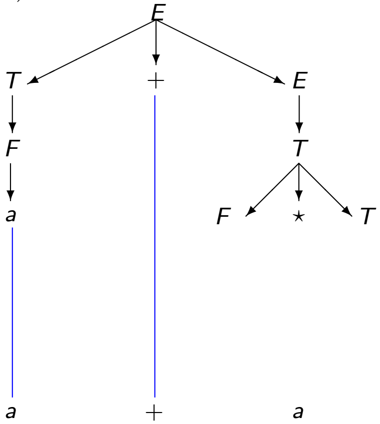


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E,
P: $1: E \to T + E \quad 2: E \to T \quad 3: T \to F \star T \quad 4: T \to F \quad 5: F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

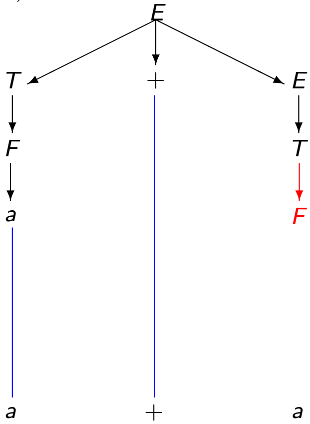


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E,\ T,\ F\}$, $V_T = \{a,\ +,\ \star\}$, simbolul de start: E,
P: $1: E \to T + E \quad 2: E \to T \quad 3: T \to F \star T \quad 4: T \to F \quad 5: F \to a$
(am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

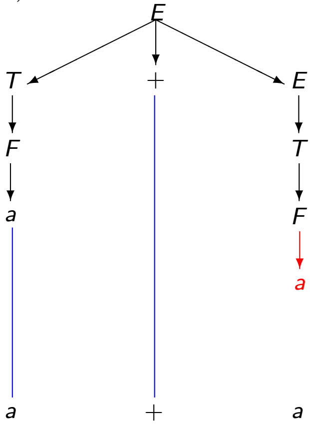


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E, P: $1: E \to T + E$ $2: E \to T$ $3: T \to F \star T$ $4: T \to F$ $5: F \to a$ (am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

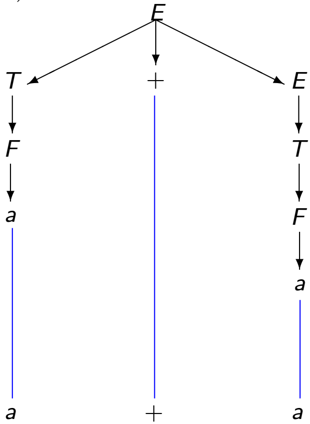


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Exemplu:

Fie G GIC dată de: $V_N = \{E, T, F\}$, $V_T = \{a, +, \star\}$, simbolul de start: E, P: $1: E \to T + E \quad 2: E \to T \quad 3: T \to F \star T \quad 4: T \to F \quad 5: F \to a$ (am numerotat producţiile). Pentru $w = a + a$ avem succesiv:

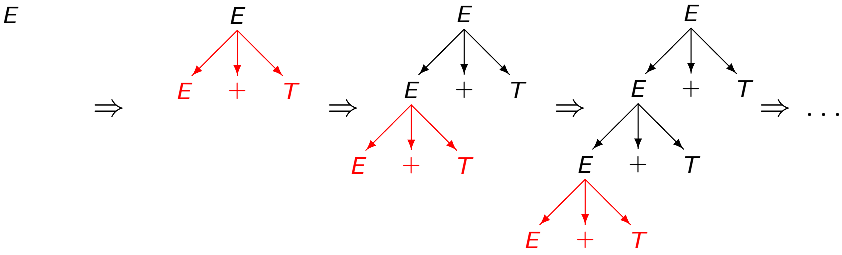


Am scris cu roşu elementele nou adăugate, cu albastru potrivirile, cu verde+încercuire nepotrivirile.

Algoritmul de parsare general top - down

Observație:

Dacă în exemplul de mai sus am fi avut producţia $1 : E \to E + T$ în loc de $1 : E \to T + E$, am fi intrat în recursie infinită:



Deci gramatica trebuie să aibă proprietăţi suplimentare, pentru a evita recursivitatea la stânga. Vom vedea însă că cerând aceste proprietăţi nu restrângem generalitatea, deoarece orice GIC poate fi transformată a.î. să le aibă.

Algoritmul de parsare general top - down

Fie $G = \langle V_N,\ V_T,\ S,\ P\rangle$ o GIC cu $L(G) \neq \emptyset$.

Definiție:

- $A \in V_G$ *este* **inaccesibil** *dacă nu există nici o derivare de forma* $S \stackrel{\star}{\Rightarrow} \alpha A \gamma$ *cu* $\alpha, \gamma \in V_G^\star$.
- $A \in V_G$ *este* **neutilizat** *dacă nu există nici o derivare de forma* $S \stackrel{\star}{\Rightarrow} \alpha A \gamma \stackrel{\star}{\Rightarrow} \alpha\beta\gamma$ *cu* $\alpha, \beta, \gamma \in V_T^\star$.

Propoziție:

G este echivalentă cu o GIC $G' = \langle V'_N,\ V'_T,\ S',\ P'\rangle$ *fără simboluri inaccesibile.*

Algoritm (A): Eliminarea simbolurilor inaccesibile

1) $V_0 \leftarrow \{S\}$; $i \leftarrow 1$;
2) $V_i \leftarrow V_{i-1} \cup \{X\ :\ \exists\, A \rightarrow \alpha X \beta \in P,\ A \in V_{i-1}\}$;
3) if $V_i \neq V_{i-1}$ then begin $i \leftarrow i+1$; goto 2 end;
4) $V'_N \leftarrow V_N \cap V_i$; $V'_T \leftarrow V_T \cap V_i$; $S' \leftarrow S$;
   $P' \leftarrow$ mulțimea producțiilor din P cu ambii membri din $V_i^\star$.

Algoritmul de parsare general top - down

Propoziție:

G este echivalentă cu o GIC $G'' = \langle V_N'', V_T, S, P'' \rangle$ *a.î.*
$\forall A \in V_N'' \, \exists \alpha \in V_T^\star$ *a.î.* $A \overset{\star}{\underset{G''}{\Rightarrow}} \alpha$.

Algoritm (B):
1) $V_0 \leftarrow \emptyset$; $i \leftarrow 1$;
2) $V_i \leftarrow V_{i-1} \cup \{A : \exists A \to \alpha \in P, \alpha \in (V_{i-1} \cup V_T)^\star\}$;
3) if $V_i \neq V_{i-1}$ then begin $i \leftarrow i + 1$; goto 2 end;
4) $V_N'' \leftarrow V_i$; $P'' \leftarrow$ mulțimea producțiilor din P cu membrul stâng din V_i, și membrul drept din $(V_i \cup V_T)^\star$.

Observații:
• Ca un corolar, rezultă că e decidabil dacă $L(G) \neq \emptyset$. Într-adevăr, aplicăm algoritmul (B) și în final verificăm dacă $S \in V_i$.
• Dacă nu ceream de la început $L(G) \neq \emptyset$, la aplicarea algoritmului (B) putea rezulta $V_N'' = \emptyset$, deci să nu avem nici simbol de start, în contradicție cu definiția GIC.
Această cerință nu este un impediment, deoarece gramaticile ce definesc sintaxa limbajelor de programare generează limbaje nevide.

Algoritmul de parsare general top - down

Algoritm (C): Eliminarea simbolurilor neutilizate
Aplicăm (B), apoi (A).

Exemplu:

Să eliminăm simbolurile neutilizate pentru GIC dată de:
$V_N = \{S, A, B, C\}$, $V_T = \{a, b\}$, S,
P: $S \to A|B$, $A \to aB|bS|b$, $B \to AB|Ba$, $C \to AS|b$

algoritmul (B): V: $\underbrace{\quad\quad}_{\text{pas } 0} \underbrace{A, \ C,}_{\text{pas } 1} \underbrace{S}_{\text{pas } 2}$

Reţinem: $V_N = \{S, A, C\}$, $V_T = \{a, b\}$, S, P: $S \to A$, $A \to bS|b$, $C \to AS|b$

algoritmul (A): V: $\underbrace{S}_{\text{pas } 0}, \underbrace{A}_{\text{pas } 1}, \underbrace{b}_{\text{pas } 2}$

Reţinem: $V_N = \{S, A\}$, $V_T = \{b\}$, S, P: $S \to A$, $A \to bS|b$.

Algoritmul de parsare general top - down

Fie $G = \langle V_N,\ V_T,\ S,\ P\rangle$ o GIC.

Definiție:

- **λ-producție**: *producție de forma* $A \to \lambda$.
- **redenumire**: *producție de forma* $A \to B$ *(*$A, B \in V_N$*).*

Propoziție:

G este echivalentă cu o GIC $G' = \langle V_N',\ V_T',\ S',\ P'\rangle$ a.î.:
• *dacă $\lambda \notin L(G)$ atunci G' nu are λ-producții.*
• *dacă $\lambda \in L(G)$ atunci singura λ-producție a lui G' este $S' \to \lambda$, iar S' nu apare în membrul drept al nici unei producții.*

Algoritmul de parsare general top - down

Algoritm (C): Eliminarea λ-producțiilor

1) cu un algoritm asemănător lui (A) sau (B) construim $N_\lambda = \{A : A \stackrel{+}{\Rightarrow} \lambda\}$;
mai exact, folosim regulile:
$V_0 \leftarrow \{A : A \to \lambda \in P\}$
$V_i \leftarrow V_{i-1} \cup \{A : A \to \alpha \in P,\ \alpha \in V_{i-1}^\star\}$
totodată, inițializăm $P' \leftarrow \emptyset$;
2) Fie $A \to \alpha_0 B_1 \ldots \alpha_{k-1} B_k \alpha_k \in P$ cu $k \geq 0$ și pentru orice i avem $B_i \in N_\lambda$
iar α_i nu conține simboluri din N_λ;
if $k = 0$ then begin $P' \leftarrow P' \cup \{A \to \alpha_0\} \setminus \{A \to \lambda\}$; goto 4 end;
3) $P' \leftarrow P' \cup \{A \to \alpha_0 X_1 \ldots \alpha_{k-1} X_k \alpha_k : X_i \in \{B_i, \lambda\}$ pt. orice $i\} \setminus \{A \to \lambda\}$;
4) $P \leftarrow P \setminus \{A \to \alpha_0 B_1 \ldots \alpha_{k-1} B_k \alpha_k\}$;
if $P \neq \emptyset$ then goto 2;
5) if $S \in N_\lambda$ then $G' \leftarrow \langle V_N \cup \{S'\},\ V_T,\ S',\ P' \cup \{S' \to S|\lambda\}\rangle$
else $G' \leftarrow \langle V_N,\ V_T,\ S,\ P'\rangle$.

Algoritmul de parsare general top - down

Exemplu:

Să eliminăm λ-producţiile pentru G dată de: $S \to aSbS|bSaS|\lambda$ (neterminalele sunt litere mari, terminalele litere mici, simbolul de start este S, restul se deduce din context).

N_λ: $\underbrace{S}_{\text{pas } 0}$

producţia $S \to aSbS$ introduce în G': $S \to aSbS|abS|aSb|ab$
producţia $S \to bSaS$ introduce în G': $S \to bSaS|baS|bSa|ba$
producţia $S \to \lambda$ nu introduce nimic în G'
cum $S \in N_\lambda$, în G' apare şi $S' \to S|\lambda$, cu S' simbol de start nou

Aşadar, gramatica G' va fi dată de:
Neterminalele: S, S'
Terminalele: a, b (aceleaşi ca în G)
Simbolul de start S' (simbol nou)
Producţiile: $S \to aSbS|abS|aSb|ab|bSaS|baS|bSa|ba$, $S' \to S|\lambda$

Algoritmul de parsare general top - down

Propoziție:

G este echivalentă cu o GIC G' fără redenumiri.

Algoritm (D): Eliminarea redenumirilor

1) Pentru orice $A \in V_N$ construim $R_A = \{B \in V_N \,:\, A \overset{\star}{\Rightarrow} B\}$ astfel:
   a) $R_0 \leftarrow \{A\}$; $i \leftarrow 1$;
   b) $R_i \leftarrow R_{i-1} \cup \{C \in V_N \,:\, B \rightarrow C \in P,\, B \in R_{i-1}\}$;
   c) if $R_i \neq R_{i-1}$ then begin $i \leftarrow i + 1$; goto b end;
   d) $R_A \leftarrow R_i$;

   Inițializăm $P' \leftarrow \emptyset$;
2) Fie $B \rightarrow \alpha \in P$;
   if $\alpha \in V_N$ then goto 4;
3) $P' \leftarrow P' \cup \{A \rightarrow \alpha \,:\, B \in R_A\}$;
4) $P \leftarrow P \setminus \{B \rightarrow \alpha\}$;
   if $P \neq \emptyset$ then goto 2;
5) G' se obține din G înlocuind P cu P'.

Algoritmul de parsare general top - down

Exemplu:

Să eliminăm redenumirile pentru G dată de:

$E \to E + T|T$, $T \to T \star F|F$, $F \to (E)|a$

(cu aceleaşi convenţii de notare, simbolul de start fiind E).

R_E: $\underbrace{E}_{\text{pas } 0}, \underbrace{T}_{\text{pas } 1}, \underbrace{F}_{\text{pas } 2}$

R_T: $\underbrace{T}_{\text{pas } 0}, \underbrace{F}_{\text{pas } 1}$

R_F: $\underbrace{F}_{\text{pas } 0}$

producţia $E \to E + T$ generează $E \to E + T$
producţia $T \to T \star F$ generează $E \to T \star F$, $T \to T \star F$
producţia $F \to (E)$ generează $E \to (E)$, $T \to (E)$, $F \to (E)$
producţia $F \to a$ generează $E \to a$, $T \to a$, $F \to a$
(restul producţiilor din G sunt redenumiri şi nu generează nimic)

P' este format doar din producţiile generate, deci avem:
P': $E \to E + T|T \star F|(E)|a$, $T \to T \star F|(E)|a$, $F \to (E)|a$.

Algoritmul de parsare general top - down

Definiție:

O GIC G este **proprie** *dacă este fără simboluri neutilizate, fără λ-producții și fără redenumiri.*

Propoziție:

Orice GIC G cu $L(G) \neq \emptyset$ este echivalentă cu o GIC proprie.

Algoritm: Aplicăm în ordine (B), (A), (C), (D).

Algoritmul de parsare general top - down

Fie $G = \langle V_N, V_T, S, P \rangle$ o GIC.

Definiție:

- $A \in V_N$ *este* **recursiv la stânga** *(resp.* **imediat recursiv la stânga***) dacă* $A \stackrel{+}{\Rightarrow} A\alpha$ *(resp.* $A \Rightarrow A\alpha$*) cu* $\alpha \neq \lambda$.
- G *este* **recursivă la stânga** *dacă are cel puțin un neterminal recursiv la stânga.*

Algoritmul de parsare general top - down

Algoritm: Eliminarea recursivității imediate la stânga

for [fiecare $A \in V_N$] do begin

Notăm alternativele lui A: $A \to A\alpha_1 | \dots | A\alpha_m | \beta_1 | \dots | \beta_n$,

unde $\alpha_i \neq \lambda$ și β_i nu începe cu A, pt. orice i;

Introducem A' neterminal nou;

Eliminăm producțiile $A \to A\alpha_1 | \dots | A\alpha_m | \beta_1 | \dots | \beta_n$;

Introducem producțiile $A \to \beta_1 | \dots | \beta_n | \beta_1 A' | \dots | \beta_n A'$

și $A' \to \alpha_1 | \dots | \alpha_m | \alpha_1 A' | \dots | \alpha_m A'$

end

Observații:

- Producțiile de forma $A \to A$ pot fi eliminate înainte, cu algoritmul de eliminare a redenumirilor.
- Prin aplicarea algoritmului de mai sus pot apărea neterminale recursive la dreapta.

Algoritmul de parsare general top - down

Algoritm: Eliminarea recursivității la stânga
Presupunem că G este fără λ-producții și fără redenumiri.
Notăm $V_N = \{A_1, \ldots, A_n\}$, $S = A_1$ (simbolul de start).

```
for i := 1 to n do begin
    for j := 1 to i − 1 do begin
        for [fiecare A_i → A_j γ ∈ P] do begin
            if [A_j → δ_1 | ... | δ_k sunt alternativele lui A_j] then
                    Introducem A_i → δ_1 γ | ... | δ_k γ;
            Eliminăm A_i → A_j γ
        end
    end;
    Eliminăm recursivitatea imediată la stânga pentru A_i
end
```

Obs: Algoritmul încearcă să transforme gramatica a.î. din orice $A_i \in V_N$ să nu se poată deriva decât cuvinte α care încep cu terminale sau cu neterminale A_j cu $j > i$. Faptul că gramatica nu are λ-producții și nici redenumiri elimină riscul ca un prefix al lui α să se deriveze în λ, expunând în stânga un neterminal A_j cu $j \leq i$, și elimină existența unei ciclicități $A \Rightarrow \ldots \Rightarrow A$, $A \in V_N$, ceea ce ar împiedica eliminarea recursivității imediate la stânga pentru A.

Algoritmul de parsare general top - down

Exemplu:

Să eliminăm recursivitatea la stânga pentru G dată de:

$E \to E + T|T$, $T \to T \star F|F$, $F \to (E)|a$

(simbolul de start este E, ordinea neterminalelor este: F, T, E).

pentru F: păstrăm $F \to (E)|a$;
pentru T: eliminăm $T \to F$ şi introducem $T \to (E)|a$;
avem deci $T \to T \star F|(E)|a$;
eliminăm recursivitatea imediată la stânga pentru T:
introducem neterminalul nou T';
înlocuim $T \to T \star F|(E)|a$ cu
$T \to (E)|a|(E)T'|aT'$,
$T' \to \star F| \star FT'$;
pentru E: eliminăm $E \to T$ şi introducem $E \to (E)|a|(E)T'|aT'$;
avem deci $E \to E + T|(E)|a|(E)T'|aT'$;
eliminăm recursivitatea imediată la stânga pentru E:
introducem neterminalul nou E';
înlocuim $E \to E + T|(E)|a|(E)T'|aT'$ cu
$E \to (E)|a|(E)T'|aT'|(E)E'|aE'|(E)T'E'|aT'E'$,
$E' \to +T| + TE'$;
pentru T' şi E' nu avem nimic de transformat.

Algoritmul de parsare general top - down

Exemplu:

Rezultă gramatica dată de:

$E \to (E)|a|(E)T'|aT'|(E)E'|aE'|(E)T'E'|aT'E',$

$E' \to +T| + TE',$

$T \to (E)|a|(E)T'|aT',$

$T' \to \star F| \star FT',$

$F \to (E)|a$

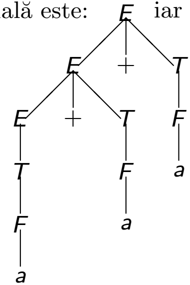
(simbolul de start este tot E).

Algoritmul de parsare general top - down

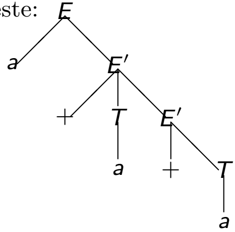
Observație:

Algoritmul de mai sus elimină recursivitatea la stânga dar introduce recursivitate la dreapta, ceea ce schimbă asociativitatea regulilor.

În contextul exemplului anterior, cuvântul $a + a + a$ este generat de ambele gramatici (iniţială şi transformată), dar arborele său de derivare în gram. iniţială este:



iar în gram. transformată este:



Într-adevăr, faptul că două gramatici sunt echivalente înseamnă că generează aceleaşi cuvinte, dar nu este obligatoriu ca şi arborii lor de derivare să arate la fel.

Algoritmul de parsare general top - down

Observație:

Primul arbore de mai sus corespunde ordinii de calcul $(a + a) + a$, în timp ce al doilea corespunde ordinii de calcul $a + (a + a)$. În cazul lui $+$ nu este afectată semantica (operaţia matematică de adunare fiind asociativă), dar în cazul lui $-$ sau $/$ ar fi.

O soluţie este introducerea de neterminale suplimentare pentru a forţa modul de asociere şi precedenţa dorită.

Algoritmul de parsare general top - down

Algoritmul de parsare general top - down se poate formaliza astfel:

Intrare: • $G = \langle V_N, V_T, S, P \rangle$ o GIC fără simboluri neutilizate și nerecursivă la stânga, cu producțiile numerotate 1, ..., $|P|$;
• $w = a_1 \ldots a_n \in V_T^+$ (deci $n \geq 1$);

Ieșire: • dacă $w \in L(G)$, atunci o analiză stângă a lui w (i.e. șirul numerelor producțiilor folosite într-o derivare stângă a sa din S);
• dacă $w \notin L(G)$, atunci eroare.

Algoritmul de parsare general top - down

Notații:
- pentru fiecare $A \in V_N$ indexăm alternativele lui A, adică considerăm o ordine a producțiilor care au pe A în stânga: $A \to \gamma_1$, ..., $A \to \gamma_{k_A}$; vom nota cu A_i indexul producției $A \to \gamma_i$;
- lucrăm cu configurații (descrieri instantanee) de forma $(s;\ i;\ \alpha;\ \beta)$, unde:
  - $s \in \{q, r, t, e\}$ reprezintă starea algoritmului (stare curentă, revenire, terminare, resp. eroare);
  - i este poziția curentă în w; se consideră și un delimitator \$ aflat pe poziția $n+1$;
  - $\alpha_{\_}$ este o stivă cu vârful la dreapta, ce reține alternativele și drumurile parcurse pe ele;
  - ${}_{\_}\beta$ este o stivă cu vârful la stânga, ce reține forma sentențială curentă.

Algoritmul de parsare general top - down

Configurația inițială este: $(q, 1, \lambda, S\$)$.

Trecerea de la o configurație la alta se face în baza următoarelor reguli (în fiecare moment se poate aplica cel mult una dintre ele):

1. $(q; i; \alpha; A\beta) \vdash (q; i; \alpha A_1; \gamma_1\beta)$, dacă A_1 este indexul producției $A \to \gamma_1$ ca primă alternativă a lui A;
2. $(q; i; \alpha; a\beta) \vdash (q; i+1; \alpha a; \beta)$, dacă $a = a_i$, $1 \leq i \leq n$;
3. $(q; i; \alpha; a\beta) \vdash (r; i; \alpha; a\beta)$, dacă $a \in V_T$, $a \neq a_i$, $1 \leq i \leq n+1$;
4. $(q; n+1; \alpha; \$) \vdash (t; n+1; \alpha; \$)$ și STOP; avem $w \in L(G)$ iar o analiză stângă pentru w se obține aplicând lui α morfismul de monoizi (i.e. extinderea la șiruri) generat de funcția $g(a) = \lambda$, dacă $a \in V_T$, și $g(A_i) = m$, dacă A_i este indexul ca alternativă a lui A a producției cu numărul m (în numerotarea inițială de la 1 la $|P|$);
5. $(r; i; \alpha a; \beta) \vdash (r; i-1; \alpha; a\beta)$, dacă $a \in V_T$;

6a. $(r; i; \alpha A_j; \gamma_j\beta) \vdash (q; i; \alpha A_{j+1}; \gamma_{j+1}\beta)$, dacă $j+1 \leq k_A$;

6b. $(r; i; \alpha A_j; \gamma_j\beta) \vdash (e; n+1; \alpha A_j; \lambda)$, dacă $i = 1$, $A = S$, $j = k_S$;

6c. $(r; i; \alpha A_j; \gamma_j\beta) \vdash (r; i; \alpha; A\beta)$, altfel.

Algoritmul de parsare general top - down

Propoziție:

$(q;\, 1;\, \lambda;\, S\$) \overset{\star}{\vdash} (t;\, n+1;\, \alpha;\, \$)$ *d.d.* $S \overset{\pi}{\Rightarrow}_s w$, *unde* $\pi = g(\alpha)$.

Obs: cu $\overset{\pi}{\Rightarrow}_s$ am notat relația de derivare stângă prin aplicarea succesivă a producțiilor cu numerele din π.

Algoritmul de parsare general top - down

Exemplu:

Să aplicăm algoritmul pentru aceeaşi gramatică şi cuvânt ca în exemplul ilustrat grafic mai devreme:

Fie G GIC dată de: $V_N = \{E, T, F\}$
$V_T = \{a, +, \star\}$
simbolul de start: E

$$
\begin{array}{lll}
P: & 1 : E \to T + E & (E_1) \\
 & 2 : E \to T & (E_2) \\
 & 3 : T \to F \star T & (T_1) \\
 & 4 : T \to F & (T_2) \\
 & 5 : F \to a & (F_1)
\end{array}
$$

(am scris în stânga numărul ca producţie şi în dreapta numărul ca alternativă).

Fie $w = \underset{1}{a}\underset{2}{+}\underset{3}{a}$ (am notat dedesubt poziţiile simbolurilor în cuvânt).

Atunci vom avea (am notat deasupra fiecărei "$\vdash$" numărul regulii aplicate):

Algoritmul de parsare general top - down

Exemplu:

$1: E \to T + E \quad (E_1), \quad 3: T \to F \star T \quad (T_1), \quad 5: F \to a \quad (F_1), \quad w = \underset{1}{a}\underset{2}{+}\underset{3}{a}$

$2: E \to T \quad (E_2), \quad 4: T \to F \quad (T_2)$

$(q;\ 1;\ \lambda;\ E\$) \overset{1}{\vdash}$

$(q;\ 1;\ E_1;\ T + E\$) \overset{1}{\vdash}$

$(q;\ 1;\ E_1 T_1;\ F \star T + E\$) \overset{1}{\vdash}$

$(q;\ 1;\ E_1 T_1 F_1;\ a \star T + E\$) \overset{2}{\vdash}$

$(q;\ 2;\ E_1 T_1 F_1 a;\ \star T + E\$) \overset{3}{\vdash}$

$(r;\ 2;\ E_1 T_1 F_1 a;\ \star T + E\$) \overset{5}{\vdash}$

$(r;\ 1;\ E_1 T_1 F_1;\ a \star T + E\$) \overset{6c}{\vdash}$

$(r;\ 1;\ E_1 T_1;\ F \star T + E\$) \overset{6a}{\vdash}$

$(q;\ 1;\ E_1 T_2;\ F + E\$) \overset{1}{\vdash}$

$(q;\ 1;\ E_1 T_2 F_1;\ a + E\$) \overset{2}{\vdash}$

$(q;\ 2;\ E_1 T_2 F_1 a;\ +E\$) \overset{2}{\vdash}$

$(q;\ 3;\ E_1 T_2 F_1 a+;\ E\$) \overset{1}{\vdash}$

Algoritmul de parsare general top - down

Exemplu:

$1: E \to T + E \quad (E_1),\quad 3: T \to F \star T \quad (T_1),\quad 5: F \to a \quad (F_1),\quad \mathrm{w}=\underset{1}{a}\underset{2}{+}\underset{3}{a}$
$2: E \to T \quad (E_2),\quad 4: T \to F \quad (T_2)$

$$(q;\, 3;\, E_1 T_2 F_1 a + E_1;\; T + E\$) \overset{1}{\vdash}$$

$$(q;\, 3;\, E_1 T_2 F_1 a + E_1 T_1;\; F \star T + E\$) \overset{1}{\vdash}$$

$$(q;\, 3;\, E_1 T_2 F_1 a + E_1 T_1 F_1;\; a \star T + E\$) \overset{2}{\vdash}$$

$$(q;\, 4;\, E_1 T_2 F_1 a + E_1 T_1 F_1 a;\; \star T + E\$) \overset{3}{\vdash}$$

$$(r;\, 4;\, E_1 T_2 F_1 a + E_1 T_1 F_1 a;\; \star T + E\$) \overset{5}{\vdash}$$

$$(r;\, 3;\, E_1 T_2 F_1 a + E_1 T_1 F_1;\; a \star T + E\$) \overset{6c}{\vdash}$$

$$(r;\, 3;\, E_1 T_2 F_1 a + E_1 T_1;\; F \star T + E\$) \overset{6a}{\vdash}$$

$$(q;\, 3;\, E_1 T_2 F_1 a + E_1 T_2;\; F + E\$) \overset{1}{\vdash}$$

$$(q;\, 3;\, E_1 T_2 F_1 a + E_1 T_2 F_1;\; a + E\$) \overset{2}{\vdash}$$

$$(q;\, 4;\, E_1 T_2 F_1 a + E_1 T_2 F_1 a;\; + E\$) \overset{3}{\vdash}$$

$$(r;\, 4;\, E_1 T_2 F_1 a + E_1 T_2 F_1 a;\; + E\$) \overset{5}{\vdash}$$

$$(r;\, 3;\, E_1 T_2 F_1 a + E_1 T_2 F_1;\; a + E\$) \overset{6c}{\vdash}$$

Algoritmul de parsare general top - down

Exemplu:

$1: E \to T + E \quad (E_1), \quad 3: T \to F \star T \quad (T_1), \quad 5: F \to a \quad (F_1), \quad w = \underset{1}{a}\underset{2}{+}\underset{3}{a}$

$2: E \to T \quad (E_2), \quad 4: T \to F \quad (T_2)$

$$(r;\ 3;\ E_1T_2F_1a + E_1T_2;\ F + E\$) \overset{6c}{\vdash}$$

$$(r;\ 3;\ E_1T_2F_1a + E_1;\ T + E\$) \overset{6a}{\vdash}$$

$$(q;\ 3;\ E_1T_2F_1a + E_2;\ T\$) \overset{1}{\vdash}$$

$$(q;\ 3;\ E_1T_2F_1a + E_2T_1;\ F \star T\$) \overset{1}{\vdash}$$

$$(q;\ 3;\ E_1T_2F_1a + E_2T_1F_1;\ a \star T\$) \overset{2}{\vdash}$$

$$(q;\ 4;\ E_1T_2F_1a + E_2T_1F_1a;\ \star T\$) \overset{3}{\vdash}$$

$$(r;\ 4;\ E_1T_2F_1a + E_2T_1F_1a;\ \star T\$) \overset{5}{\vdash}$$

$$(r;\ 3;\ E_1T_2F_1a + E_2T_1F_1;\ a \star T\$) \overset{6c}{\vdash}$$

$$(r;\ 3;\ E_1T_2F_1a + E_2T_1;\ F \star T\$) \overset{6a}{\vdash}$$

$$(q;\ 3;\ E_1T_2F_1a + E_2T_2;\ F\$) \overset{1}{\vdash}$$

$$(q;\ 3;\ E_1T_2F_1a + E_2T_2F_1;\ a\$) \overset{2}{\vdash}$$

$$(q;\ 4;\ E_1T_2F_1a + E_2T_2F_1a;\ \$) \overset{4}{\vdash}$$

Algoritmul de parsare general top - down

Exemplu:

$1: E \to T + E \quad (E_1), \quad 3: T \to F \star T \quad (T_1), \quad 5: F \to a \quad (F_1), \quad \mathrm{w}=\underset{1}{a}\underset{2}{+}\underset{3}{a}$
$2: E \to T \quad (E_2), \quad 4: T \to F \quad (T_2)$

$(t;\ 4;\ E_1T_2F_1a + E_2T_2F_1a;\ \$)$ accept
$\pi = g(E_1T_2F_1a + E_2T_2F_1a) = 145245.$

Algoritmul de parsare general top - down

Observație:

Dacă am fi numerotat producţiile în ordinea:

$$\begin{array}{llll} 1: E \rightarrow T+E & (E_1), & 3: T \rightarrow F & (T_1), & 5: F \rightarrow a & (F_1) \\ 2: E \rightarrow T & (E_2), & 4: T \rightarrow F \star T & (T_2), & & \end{array}$$

am fi ajuns la succes după mai puţini paşi (exerciţiu).

În forma prezentată însă algoritmul nu determina o numerotare optimă a producţiilor (numerotarea se dă de la început). Determinarea unei numerotări mai bune ar adăuga însă complexitate suplimentară algoritmului.

Cuprins

Algoritmi de parsare *k*-predictivi

În algoritmul general top-down acceptarea unei producții s-a bazat pe consultarea simbolurilor aflate în stânga porțiunii rămase din cuvânt și a formei sentențiale curente.

Backtrackingul ar putea fi eliminat dacă la fiecare pas consultarea unui număr de k terminale aflate în stânga porțiunii rămase din cuvânt și a simbolului din stânga formei sentențiale curente ar desemna direct, în mod unic, acțiunea "potrivită": producția "potrivită" care trebuie aplicată, finalizarea cu succes, finalizarea cu eșec, etc.;

Cu alte cuvinte, ar fi bine dacă am avea o tabelă (numită **tabelă k-predictivă**) care pentru orice cuvânt din terminale de lungime $< k$ sau prefix de lungime k al unui cuvânt mai lung și pentru orice simbol care ar putea apărea în stânga unei forme sentențiale ar asocia acțiunea respectivă.

Obținem astfel un algoritm de parsare de tip top-down fără revenire, numit **algoritm de parsare k-predictiv**, care este liniar.

Algoritmi de parsare k-predictivi

Fie $G = \langle V_N,\ V_T,\ S,\ P\rangle$ o GIC.

Definiție:

Pentru orice $\alpha \in V_G^\star$ *notăm:*

$First_k(\alpha) = \{w \in V_T^\star \ :\ (|w| < k \text{ și } \alpha \overset{\star}{\Rightarrow} w)$
$\quad \text{sau } (|w| = k \text{ și } \exists x \in V_G^\star \text{ a.î. } \alpha \overset{\star}{\Rightarrow} wx)\}.$

În particular, $First_0(\alpha) = \{\lambda\}$.

$Follow_k(\alpha) = \{w \in V_T^\star \ :\ \exists x, \beta \in V_G^\star \text{ a.î. } S \overset{\star}{\Rightarrow} x\alpha\beta \text{ și } w \in First_k(\beta)\}.$

Deci:

- $First_k(\alpha)$ este mulțimea șirurilor de k terminale care pot apărea la începutul unui șir derivat din α, sau a șirurilor de $< k$ terminale ce se pot deriva integral din α.
- $Follow_k(\alpha)$ este mulțimea șirurilor de k sau $< k$ terminale ca mai sus care pot apărea după ceva derivat din α, în cadrul unui șir derivat din S.

Algoritmi de parsare k-predictivi

Algoritm de calcul pentru $First_k(\alpha)$, $\alpha \in V_G^\star$:
1) for [orice $a \in V_T$] do $First_k(a) \leftarrow \{a\}$;
2) for [orice $A \in V_N$] do $First_k(A) \leftarrow \emptyset$;
repeat
for [orice $A \to \alpha \in P$] do $First_k(A) \leftarrow First_k(A) \cup First_k(\alpha)$

$$\left(\begin{array}{ll} \text{unde}: & First_k(\lambda) = \{\lambda\} \\ & First_k(X_1 \ldots X_n) = First_k(First_k(X_1) \cdot \ldots \cdot First_k(X_n)),\ X_i \in G \\ \text{iar pt. } M \subseteq V_G^\star: & First_k(M) = \bigcup\limits_{\alpha \in M} First_k(\alpha) \end{array}\right)$$

until [mulțimile $First_k(A)$, $A \in V_N$, nu se mai schimbă];
3) În general, pentru $\alpha \in V_G^\star$ vom avea:
dacă $\alpha = \lambda$, atunci $First_k(\alpha) = \{\lambda\}$;
dacă $\alpha = X_1 \ldots X_n$, $X_i \in V_G$, atunci $First_k(\alpha) = First_k(First_k(X_1) \cdot \ldots \cdot First_k(X_n))$.

Algoritmi de parsare k-predictivi

Algoritm de calcul pentru $Follow_k(A)$**,** $A \in V_N$:

for [orice $A \in V_N \setminus \{S\}$] do $Follow_k(A) \leftarrow \emptyset$;
$Follow_k(S) \leftarrow \{\lambda\}$;
repeat
  for [orice $A \to \alpha B \gamma \in P$, cu $B \in V_N$ și $\alpha, \gamma \in V_G^\star$] do
    $Follow_k(B) \leftarrow Follow_k(B) \cup First_k(First_k(\gamma) \cdot Follow_k(A))$;
until [mulțimile $Follow_k(A)$, $A \in V_N$, nu se mai schimbă];

Algoritmi de parsare k-predictivi

Algoritmul de parsare k-predictiv se formalizează astfel:

Fie $G = \langle V_N,\ V_T,\ S,\ P\rangle$ o GIC cu producțiile numerotate de la 1 la $|P|$.
Vom numi **extensia** sa GIC $G' = \langle V_N \cup \{S'\},\ V_T \cup \{\$\},\ S',\ P \cup \{S' \to S\$\}\rangle$, unde S' și \$ sunt simboluri noi.
Pt. orice $i \in \mathbf{N}$ notăm $V_T^{(i)} = \{w \in V_T^\star\ :\ |w| = i\}$; în particular, $V_T^{(0)} = \{\lambda\}$.

Presupunem defintă o aplicație (numită **tabelă de parsare *k*-predictivă**)
$M : ((\bigcup_{i=0}^{k-1} V_T^{(i)})\{\$\} \cup V_T^{(k)}) \times (V_G \cup \{\$\})$
$\to \{accept,\ delete,\ error\} \cup \{(\beta, i)\ :\ i : A \to \beta \in P\}$
astfel încât:

$M(a\alpha, a) = delete$, pentru orice $a \in V_T$;
$M(\$, \$) = accept$.

Algoritmi de parsare k-predictivi

Algoritmul de parsare k-predictiv construit pe baza lui M este:

- Se lucrează cu **configurații** de forma $(w\$;\ \alpha\$;\ \pi)$, unde $w \in V_T^\star$, $\alpha \in V_G^\star$, $\pi \in \{1, \ldots, |P|\}^\star$;
- **Configurația inițială** este $(w\$;\ S\$;\ \lambda)$, unde w este cuvântul de analizat;
- **Trecerea de la o configurație la alta** se face în baza următoarelor reguli:

1. $(\gamma;\ A\delta;\ \pi) \vdash (\gamma;\ \beta\delta;\ \pi i)$, dacă $M(First_k(\gamma), A) = (\beta, i)$, $i : A \to \beta$;
2. $(a\gamma;\ a\delta;\ \pi) \vdash (\gamma;\ \delta;\ \pi)$, $a \in V_T$ (corespunde cazului $M(a\gamma, a) = delete$, $a \in V_T$);
3. $(\$;\ \$;\ \pi) \vdash accept$ (corespunde cazului $M(\$, \$) = accept$);
4. $(\gamma;\ \delta;\ \pi) \vdash error$, altfel (corespunde cazurilor când $M(.,.) = error$).

Algoritmi de parsare k-predictivi

Definim funcția parțială $\Theta_M : V_T^\star \multimap\!\!\rightarrow \{1, \ldots, |P|\}^\star$ prin:

$$\Theta_M(w) = \begin{cases} \pi, & \text{dacă } (w\$;\ S\$;\ \lambda) \overset{\star}{\vdash} (\$;\ \$;\ \pi) \vdash \mathit{accept} \\ \text{nedefinită}, & \text{dacă } (w\$;\ S\$;\ \lambda) \overset{\star}{\vdash} \mathit{error}. \end{cases}$$

(prin $\overset{\star}{\vdash}$ am notat închiderea reflexivă și tranzitivă a relației $\vdash$).

Definiție:

Spunem că (Θ, M) este un algoritm de parsare k-predictiv **valid** *pentru G și că M este o tabelă de parsare* **validă** *pentru G, dacă*

$L(G) = \{w \in V_T^\star : \exists\pi$ a.î. $(w\$;\ S\$;\ \lambda) \overset{\star}{\vdash} (\$;\ \$;\ \pi)$ (i.e. $\Theta_M(w) = \pi)\}$.
Obs: În acest caz va rezulta și că $S \overset{\pi}{\Rightarrow}_s w$.

Algoritmi de parsare k-predictivi

Nu pentru orice GIC se poate construi o tabelă de parsare k-predictivă.

O asemena tabelă se poate construi pentru gramatici de tip $LL(k)$ sau $LL(k)$ tari.

Cuprins

Alg. de parsare pt. gramatici LL(k) tare

$G = \langle V_N, V_T, S, P\rangle$ o GIC.

Definiție:

- *G este* **de tip** *$LL(k)$, $k \in \mathbf{N}$, dacă pentru orice derivare stângă*
$$S \overset{\star}{\Rightarrow}_s wA\alpha \ (w \in V_T^\star,\ A \in V_N,\ \alpha \in V_G^\star)$$
și pentru orice producții $A \to \beta$, $A \to \gamma \in P$ a.î.
$$\begin{array}{l} S \overset{\star}{\Rightarrow}_s wA\alpha \Rightarrow_s w\beta\alpha \Rightarrow_s wx \\ S \overset{\star}{\Rightarrow}_s wA\alpha \Rightarrow_s w\gamma\alpha \Rightarrow_s wy \end{array} \quad (x, y \in V_T^\star)$$
cu $Fist_k(x) = First_k(y)$, avem $\beta = \gamma$.
- *G este* **de tip** *LL dacă există $k \in \mathbf{N}$ a.î. G este de tip $LL(k)$.*

Observăm deci că pentru o gramatică de tip LL, la orice pas al unei derivări stângi, șirul de terminale generat până la momentul curent w, neterminalul curent A și primele k terminale care urmează în cuvântul analizat (acele "$First_k$", care conțin fiecare doar câte 1 element) determină în mod unic producția care se aplică din A, adică ce doream.

Denumirea LL (**left-left**) provine de la faptul că în algoritmul de parsare cuvântul analizat este parcurs de la stânga la dreapta (primul "left"), iar în final se obține o derivare stângă a sa (al doilea "left").

Alg. de parsare pt. gramatici LL(k) tare

Definiție:

- *Un limbaj este* **de tip** $LL(k)$, $k \in \mathbf{N}$, *dacă există o gramatică de tip* $LL(k)$ *care îl generează.*
- *Un limbaj este* **de tip** *LL dacă există* $k \in \mathbf{N}$ *a.î. limbajul este de tip* $LL(k)$.

Alg. de parsare pt. gramatici LL(k) tare

$G = \langle V_N, V_T, S, P \rangle$ o GIC.

Definiție:

G este **de tip** *$LL(k)$* **tare**, *$k \in \mathbf{N}$, dacă pt. orice producții $A \to \beta$, $A \to \gamma \in P$ cu $\beta \neq \gamma$ avem $First_k(\beta Follow_k(A)) \cap First_k(\gamma Follow_k(A)) = \emptyset$.*

Propoziție:

- *Dacă G nu are simboluri neutilizate și este de tip LL, atunci G este neambiguă.*
- *Dacă G nu are simboluri neutilizate și este recursivă la stânga, atunci G nu este de tip LL.*
- *Pentru orice $k \in \mathbf{N}$, dacă G este de tip $LL(k)$, atunci G este de tip $LL(k+1)$.*
- *Pentru orice $k \in \mathbf{N}$, dacă G este de tip $LL(k)$ tare, atunci G este de tip $LL(k)$.*
- *Dacă G nu are simboluri neutilizate, atunci G este de tip $LL(1)$ tare d.d. G este de tip $LL(1)$.*
- *(Caracterizare) G fără simboluri neutilizate este $LL(k)$, $k \geq 1$, d.d. pentru orice derivare stângă $S \overset{\star}{\Rightarrow}_s wA\alpha$ cu $w \in V_T^\star$ și orice producții $A \to \beta$, $A \to \gamma \in P$ avem $First_k(\beta\alpha) \cap First_k(\gamma\alpha) = \emptyset$.*

Alg. de parsare pt. gramatici LL(k) tare

Exemplu:

Să verificăm, cu proprietatea de caracterizare, că următoarea gramatică este $LL(1)$: $S \to aAS|b,\ A \to bSA|a$.

Dacă $S \overset{\star}{\Rightarrow}_s wS\alpha,\ w \in V_T^\star$, atunci avem $\begin{cases} First_1(aAS\alpha) = \{a\} \\ First_1(b\alpha) = \{b\} \end{cases}$,

deci $First_1(aAS\alpha) \cap First_1(b\alpha) = \emptyset$.

Dacă $S \overset{\star}{\Rightarrow}_s wA\alpha,\ w \in V_T^\star$, atunci avem $\begin{cases} First_1(bSA\alpha) = \{b\} \\ First_1(a\alpha) = \{a\} \end{cases}$,

deci $First_1(bSA\alpha) \cap First_1(a\alpha) = \emptyset$.

Exerciţiu: verificaţi aceeaşi proprietate folosind definiţia.

Alg. de parsare pt. gramatici LL(k) tare

Pentru o GIC $G = \langle V_N, V_T, S, P\rangle$ de tip $LL(k)$ tare, $k \in \mathbb{N}$ putem construi o tabelă de parsare k-predictivă astfel:

- Numerotăm producțiile sub forma $i : A \to \alpha$, $i \in \{1, \ldots, |P|\}$.
- Considerăm extensia $G' = \langle V_N \cup \{S'\}, V_T \cup \{\$\}, S', P \cup \{S' \to S\$\}\rangle$ a lui G, unde S' și \$ sunt simboluri noi.
- Definim $M : ((\bigcup_{i=0}^{k-1} V_T^{(i)})\{\$\} \cup V_T^{(k)}) \times (V_G \cup \{\$\})$
  $\to \{accept, delete, error\} \cup \{(\beta, i)\ :\ i : A \to \beta \in P\}$
  astfel:
  - pentru orice producție $i : A \to \beta \in P$ și orice $u \in First_k(\beta Follow_k(A))$, definim $M(u, A) = (\beta, i)$;
  - $M(a\alpha, a) = delete$, pentru orice $a \in V_T$;
  - $M(\$, \$) = accept$;
  - $M(., .) = error$, în rest;

  (mulțimile $First_k, Follow_k$ sunt calculate relativ la G').

Alg. de parsare pt. gramatici LL(k) tare

Observație:

- În cazul $k = 1$, prima proprietate din definiţia lui M se poate înlocui cu:
  - pentru orice $i : A \to \beta \in P$ efectuăm următoarele:
    - pentru orice $a \in First_1(\beta) \setminus \{\lambda\}$ definim $M(a, A) = (\beta, i)$;
    - dacă $\lambda \in First_1(\beta)$, atunci pentru orice $b \in Follow_1(A)$ definim $M(b, A) = (\beta, i)$.

- Pentru o GIC G oarecare (nu neapărat $LL(k)$ tare) construcţia lui M de mai sus ar putea să nu se facă corect, în sensul că regulile indicate ar putea da mai multe valori diferite pentru un acelaşi $M(.,.)$ (i.e. M să aibă **intrări multiple**). Asemenea intrări multiple se mai numesc şi **conflicte**.

Pericol este doar în cazurile $M(u, A)$, $A \in V_N$, deoarece pentru un u şi A fixate ar putea exista mai multe producţii $A \to \beta$ cu $u \in First_k(\beta Follow_k(A))$.

Vom prezenta mai tărziu câteva tehnici de transformare a unei GIC a.î. să nu mai apară conflicte.

Alg. de parsare pt. gramatici LL(k) tare

Propoziție:

• *Pentru o GIC G tabela M construită ca mai sus nu are intrări multiple d.d. G este de tip $LL(k)$ tare.*

• *Dacă G este de tip $LL(k)$ tare, atunci algoritmul de parsare k-predictiv construit pe baza tabelei M construită ca mai sus este valid pentru G.*

Alg. de parsare pt. gramatici LL(k) tare

Exemplu:

Aplicaţie pentru o GIC fără simboluri neutilizate în cazul $LL(1)$ (i.e. $LL(1)$ tare):

Fie GIC următoare (care defineşte expresii aritmetice):

$$\begin{array}{lll} 1: E \to TX & 4: T \to FY & 7: F \to (E) \\ 2: X \to +TX & 5: Y \to \star FY & 8: F \to a \\ 3: X \to \lambda & 6: Y \to \lambda & \end{array}$$

(deci $V_N = \{E, T, X, F, Y\}$, $V_T = \{+, \star, (,), a\}$, simbol de start este E şi am numerotat producţiile).

Să aplicăm algoritmul pentru a parsa cuvântul $w = (a \star a)$.

Începem prin a construi tabela de parsare 1-predictivă M.
Conform observaţiei anterioare, avem nevoie să calculăm
  $First_1(\beta)$, pentru orice producţie $A \to \beta$
  $Follow_1(A)$, pentru orice neterminal A.
În acest scop, vom aplica algoritmii de calcul pentru $First_k$, $Follow_k$ prezentaţi mai înainte, pentru GIC extinsă (notăm noul simbol de start E').
În loc de $First_1$, $Follow_1$ vom scrie prescurtat Fi, resp. Fo.

Alg. de parsare pt. gramatici LL(k) tare

Exemplu:

$1: E \to TX$	$3: X \to \lambda$	$5: Y \to \star FY$	$7: F \to (E)$	$9: E' \to E\$$
$2: X \to +TX$	$4: T \to FY$	$6: Y \to \lambda$	$8: F \to a$	

Mai întâi calculăm $Fi(\beta)$, pentru orice producţie $A \to \beta$.

În tabelul următor, liniile corespund terminalelor, neterminalelor şi membrilor drepţi ai producţiilor, iar coloanele corespund paşilor din algoritm.
În celula corespunzătoare liniei α şi coloanei i scriem ce se adaugă la pasul i în $Fi(\alpha)$.
Coloanele se construiesc succesiv spre dreapta până nu se mai adaugă nimic pe nici o linie.
În final, pe linia fiecărui α vom găsi elementele lui $Fi(\alpha)$.

Alg. de parsare pt. gramatici LL(k) tare

Exemplu:

$1: E \to TX$ $\quad$ $3: X \to \lambda$ $\quad$ $5: Y \to \star FY$ $\quad$ $7: F \to (E)$ $\quad$ $9: E' \to E\$$

$2: X \to +TX$ $\quad$ $4: T \to FY$ $\quad$ $6: Y \to \lambda$ $\quad$ $8: F \to a$

α	pas 1	pas 2	pas 3	pas 4	pas 5	pas 6
$+$	$+$					
$\star$	$\star$					
(	(					
)	)					
a	a					
$\$$	$\$$					
E'					(, a	
E				(, a		
X		$+, \lambda$				
T			(, a			
F		(, a				
Y		$\star, \lambda$				
TX			(, a			
$+TX$	$+$					
λ	λ					
FY		(, a				
$\star FY$	$\star$					
(E)	(					
$E\$$				(, a		

Alg. de parsare pt. gramatici LL(k) tare

Exemplu:

$1: E \to TX$	$3: X \to \lambda$	$5: Y \to \star FY$	$7: F \to (E)$	$9: E' \to E\$$
$2: X \to +TX$	$4: T \to FY$	$6: Y \to \lambda$	$8: F \to a$	

În final reţinem:

$Fi(TX) = \{(, a\}$
$Fi(+TX) = \{+\}$
$Fi(\lambda) = \{\lambda\}$
$Fi(FY) = \{(, a\}$
$Fi(\star FY) = \{\star\}$
$Fi((E)) = \{(\}$
$Fi(a) = \{a\}$
$Fi(E\$) = \{(, a\}$

Vom avea însă nevoie şi de:

$Fi(E) = \{(, a\}$
$Fi(X) = \{+, \lambda\}$
$Fi(Y) = \{\star, \lambda\}$

Alg. de parsare pt. gramatici LL(k) tare

Exemplu:

$1: E \to TX$	$3: X \to \lambda$	$5: Y \to \star FY$	$7: F \to (E)$	$9: E' \to E\$$
$2: X \to +TX$	$4: T \to FY$	$6: Y \to \lambda$	$8: F \to a$	

Acum calculăm $Fo(A)$, pentru orice neterminal A.

La $Fo(E)$ se adaugă la fiecare pas:

$Fi(Fi())Fo(F)) = \{)\}$ (cf. prod. $F \to (E)$)

$Fi(Fi(\$)Fo(E')) = \{\$\}$ (cf. prod. $E' \to E\$$)

La $Fo(X)$ se adaugă la fiecare pas:

$Fi(Fi(\lambda)Fo(E)) = Fo(E)$ (cf. prod. $E \to TX$)

$Fi(Fi(\lambda)Fo(X)) = Fo(X)$ (cf. prod. $X \to +TX$) - este nerelevant

La $Fo(T)$ se adaugă la fiecare pas:

$Fi(Fi(X)Fo(E)) = Fi(\{+, \lambda\}Fo(E)) = \{+\} \cup Fo(E)$ (cf. prod. $E \to TX$)

$Fi(Fi(X)Fo(X)) = Fi(\{+, \lambda\}Fo(X)) = \{+\} \cup Fo(X)$ (cf. prod. $X \to +TX$)

Alg. de parsare pt. gramatici LL(k) tare

Exemplu:

$1: E \to TX$	$3: X \to \lambda$	$5: Y \to \star FY$	$7: F \to (E)$	$9: E' \to E\$$
$2: X \to +TX$	$4: T \to FY$	$6: Y \to \lambda$	$8: F \to a$	

La $Fo(F)$ se adaugă la fiecare pas:

$Fi(Fi(Y)Fo(T)) = Fi(\{\star, \lambda\}Fo(T)) = \{\star\} \cup Fo(T)$ (cf. prod. $T \to FY$)

$Fi(Fi(Y)Fo(Y)) = Fi(\{\star, \lambda\}Fo(Y)) = \{\star\} \cup Fo(Y)$ (cf. prod. $Y \to *FY$)

La $Fo(Y)$ se adaugă la fiecare pas:

$Fi(Fi(\lambda)Fo(T)) = Fo(T)$ (cf. prod. $T \to FY$)

$Fi(Fi(\lambda)Fo(Y)) = Fo(Y)$ (cf. prod. $Y \to \star FY$) - este nerelevant

$Fo(E')$ rămâne $\{\lambda\}$ (ca la iniţializare).

Alg. de parsare pt. gramatici LL(k) tare

Exemplu:

$1: E \to TX$ $\quad 3: X \to \lambda$ $\quad 5: Y \to \star FY$ $\quad 7: F \to (E)$ $\quad 9: E' \to E\$$
$2: X \to +TX$ $\quad 4: T \to FY$ $\quad 6: Y \to \lambda$ $\quad 8: F \to a$

În tabelul următor, liniile corespund neterminalelor iar coloanele paşilor; $tabel(N, i)$ conţine ce se adaugă la $Fo(N)$ la pasul i; coloanele se completează spre dreapta, iar algoritmul se opreşte când nu se mai adaugă nimic pe nici o linie (atunci pe linia fiecărui N vom găsi elementele lui $Fo(N)$):

Neterminal	pas 1	pas 2	pas 3
E'	λ		
E		), \$	
X		), \$	
T		+,), \$	
F		$\star$, +,), \$	
Y		+,), \$	

Alg. de parsare pt. gramatici LL(k) tare

Exemplu:

$1: E \to TX$	$3: X \to \lambda$	$5: Y \to \star FY$	$7: F \to (E)$	$9: E' \to E\$$
$2: X \to +TX$	$4: T \to FY$	$6: Y \to \lambda$	$8: F \to a$	

În final reţinem:

$Fo(E') = \{\lambda\}$
$Fo(E) = \{), \$\}$
$Fo(X) = \{), \$\}$
$Fo(T) = \{+,), \$\}$
$Fo(F) = \{\star, +,), \$\}$
$Fo(Y) = \{+,), \$\}$

Alg. de parsare pt. gramatici LL(k) tare

Exemplu:

$1: E \to TX$	$3: X \to \lambda$	$5: Y \to \star FY$	$7: F \to (E)$	$9: E' \to E\$$
$2: X \to +TX$	$4: T \to FY$	$6: Y \to \lambda$	$8: F \to a$	

Acum construim tabela de parsare 1-predictivă $M : (\{\$\} \cup V_T) \times (V_G \cup \{\$\}) \to \{acc, del, err\} \cup \{(\beta, i) \;:\; i : A \to \beta$ producţie în $G\}$

Pentru $1 : E \to TX$ avem $Fi(TXFo(E)) = \{(, a\}$ (deoarece $Fi(TX) = \{(, a\}$), deci $M((, E) = (TX, 1)$, $M(a, E) = (TX, 1)$.

Asemănător:

Pentru $2 : X \to +TX$ avem $Fi(+TXFo(X)) = \{+\}$.

Pentru $3 : X \to \lambda$ avem $Fi(\lambda Fo(X)) = \{), \$\}$.

Pentru $4 : T \to FY$ avem $Fi(FYFo(T)) = \{(, a\}$ (deoarece $Fi(FY) = \{(, a\}$).

Pentru $5 : Y \to \star FY$ avem $Fi(\star FYFo(Y)) = \{\star\}$.

Pentru $6 : Y \to \lambda$ avem $Fi(\lambda Fo(Y)) = \{+,), \$\}$.

Pentru $7 : F \to (E)$ avem $Fi((E)Fo(F)) = \{(\}$.

Pentru $8 : F \to a$ avem $Fi(aFo(F)) = \{a\}$.

Alg. de parsare pt. gramatici LL(k) tare

Exemplu:

$1: E \rightarrow TX$ $\quad 3: X \rightarrow \lambda$ $\quad 5: Y \rightarrow \star FY$ $\quad 7: F \rightarrow (E)$ $\quad 9: E' \rightarrow E\$$
$2: X \rightarrow +TX$ $\quad 4: T \rightarrow FY$ $\quad 6: Y \rightarrow \lambda$ $\quad 8: F \rightarrow a$

Pentru $1: E \rightarrow TX$ avem $Fi(TXFo(E)) = \{(, a\}$.
Pentru $2: X \rightarrow +TX$ avem $Fi(+TXFo(X)) = \{+\}$.
Pentru $3: X \rightarrow \lambda$ avem $Fi(\lambda Fo(X)) = \{), \$\}$.
Pentru $4: T \rightarrow FY$ avem $Fi(FYFo(T)) = \{(, a\}$.
Pentru $5: Y \rightarrow \star FY$ avem $Fi(\star FYFo(Y)) = \{\star\}$.
Pentru $6: Y \rightarrow \lambda$ avem $Fi(\lambda Fo(Y)) = \{+,), \$\}$.
Pentru $7: F \rightarrow (E)$ avem $Fi((E)Fo(F)) = \{(\}$.
Pentru $8: F \rightarrow a$ avem $Fi(aFo(F)) = \{a\}$.

Obţinem tabela (nu am mai trecut intrările *err*):

	E	X	T	F	Y	$+$	$\star$	(	)	a	\$
\$		$\lambda, 3$			$\lambda, 6$						*acc*
$+$		$+TX, 2$			$\lambda, 6$	*del*					
$\star$					$\star FY, 5$		*del*				
(	$TX, 1$		$FY, 4$	$(E), 7$				*del*			
)		$\lambda, 3$			$\lambda, 6$				*del*		
a	$TX, 1$		$FY, 4$	$a, 8$						*del*	

Să analizăm cuvântul $w = (a \star a)$ (la fiecare tranziţie am subliniat prefixele care devin argumente pentru funcţia M):

Alg. de parsare pt. gramatici LL(k) tare

Exemplu:

	E	X	T	F	Y	$+$	$\star$	(	)	a	\$
\$		$\lambda, 3$			$\lambda, 6$						*acc*
$+$		$+TX, 2$			$\lambda, 6$	*del*					
$\star$					$\star FY, 5$		*del*				
(	$TX, 1$		$FY, 4$	$(E), 7$				*del*			
)		$\lambda, 3$			$\lambda, 6$				*del*		
a	$TX, 1$		$FY, 4$	$a, 8$						*del*	

$((a \star a)\$;\ \underline{E}\$;\ \lambda) \vdash$

$(\underline{(}a \star a)\$;\ \underline{T}X\$;\ 1) \vdash$

$(\underline{(}a \star a)\$;\ \underline{F}YX\$;\ 14) \vdash$

$(\underline{(}a \star a)\$;\ (E)YX\$;\ 147) \vdash$

$(\underline{a} \star a)\$;\ \underline{E})YX\$;\ 147) \vdash$

$(\underline{a} \star a)\$;\ \underline{T}X)YX\$;\ 1471) \vdash$

$(\underline{a} \star a)\$;\ \underline{F}YX)YX\$;\ 14714) \vdash$

$(\underline{a} \star a)\$;\ \underline{a}YX)YX\$;\ 147148) \vdash$

$(\underline{\star} a)\$;\ \underline{Y}X)YX\$;\ 147148) \vdash$

$(\underline{\star} a)\$;\ \underline{\star}FYX)YX\$;\ 1471485) \vdash$

$(\underline{a})\$;\ \underline{F}YX)YX\$;\ 1471485) \vdash$

$(\underline{a})\$;\ \underline{a}YX)YX\$;\ 14714858) \vdash$

$(\underline{)}\$;\ \underline{Y}X)YX\$;\ 14714858) \vdash$

Alg. de parsare pt. gramatici LL(k) tare

Exemplu:

	E	X	T	F	Y	$+$	$\star$	$($	$)$	a	$\$$
$\$$		$\lambda, 3$			$\lambda, 6$						*acc*
$+$		$+TX, 2$			$\lambda, 6$	*del*					
$\star$					$\star FY, 5$		*del*				
$($	$TX, 1$		$FY, 4$	$(E), 7$				*del*			
$)$		$\lambda, 3$			$\lambda, 6$				*del*		
a	$TX, 1$		$FY, 4$	$a, 8$						*del*	

$(\,)\$;\ \underline{X})YX\$;\ 147148586) \vdash$
$(\,\underline{)}\$;\ \underline{)}YX\$;\ 1471485863) \vdash$
$(\underline{\$}\ ;\ \underline{Y}X\$;\ 1471485863) \vdash$
$(\underline{\$}\ ;\ \underline{X}\$;\ 14714858636) \vdash$
$(\underline{\$}\ ;\ \underline{\$}\ ;\ 147148586363) \vdash$
accept

Analiza rezultată este: 147148586363.

Observăm că parsarea s-a făcut fără revenire.

Exerciţiu: desenaţi arborele de derivare.

Alg. de parsare pt. gramatici LL(k) tare

Am văzut că pentru GIC oarecare, dacă încercăm să construim tabela de parsare k-predictivă M ca mai sus, pot apărea intrări multiple (**conflicte**), dacă gramatica nu este de tip $LL(k)$ tare.

Riscul să apară conflicte există doar la intrările $M(u, A)$, cu A neterminal - anume pentru un u și un A fixate ar putea exista mai multe producții $A \to \beta$ a.î. $u \in First_k(\beta Follow_k(A))$.

Uneori însă conflictele pot fi rezolvate - gramatica poate fi transformată într-una echivalentă de tip $LL(k)$ tare.

Prezentăm în continuare tipurile posibile de conflicte și câteva metode de rezolvare a lor. Vom ilustra doar pentru cazul $k = 1$.
Am văzut că în acest caz

$$u \in First_k(\beta Follow_k(A)) \Leftrightarrow \begin{cases} u \in First_1(\beta) \setminus \{\lambda\} \\ \quad \text{sau} \\ \lambda \in First_1(\beta) \text{ și } u \in Follow_1(A) \end{cases}$$

Alg. de parsare pt. gramatici LL(k) tare

- **Conflicte** *First*/*First*:

Presupunem că avem două producții $i_1 : A \to \beta_1$, $i_2 : A \to \beta_2$, $\beta_1 \neq \beta_2$, și $u \in (First_1(\beta_1) \cap First_1(\beta_2)) \setminus \{\lambda\}$.
Atunci în $M(u, A)$ ajunge și (β_1, i_1) și (β_2, i_2).

O soluție: **factorizarea la stânga**

Exemplu:

producțiile: $\begin{array}{l} A \to X \\ A \to XYZ \end{array}$ se înlocuiesc cu: $\begin{array}{l} A \to XA' \quad (A' \text{ neterminal nou}) \\ A' \to \lambda \\ A' \to YZ \end{array}$

Alg. de parsare pt. gramatici LL(k) tare

- **Conflicte** *First*/*Follow*:

Exemplu:

Dacă avem producțiile: $\begin{array}{l} 1: S \rightarrow Aab \\ 2: A \rightarrow a \\ 3: A \rightarrow \lambda \end{array}$ (S și A neterminale, a și b terminale).

atunci $M(a, A)$ poate fi: $\left\{ \begin{array}{l} (a, 2), \text{ pentru că } a \in First_1(a) \\ \qquad \text{sau} \\ (\lambda, 3), \text{ pentru că } \lambda \in First_1(\lambda) \text{ și } a \in Follow_1(A) \end{array} \right.$

O soluție: **substituția**

Exemplu:

Înlocuim producțiile 1, 2, 3 cu: $\begin{array}{l} S \rightarrow aab \\ S \rightarrow ab \end{array}$ (dispare A).

Obs. că în urma substituției pot apărea conflicte *First*/*First* - atunci facem în continuare factorizare la stânga și înlocuim aceste producții cu: $\begin{array}{l} S \rightarrow aA \\ A \rightarrow ab \\ A \rightarrow b \end{array}$

(reapare A, dar producțiile sunt altele).

Alg. de parsare pt. gramatici LL(k) tare

- **Recursivitatea la stânga**:

Am spus că o GIC $G = \langle V_N, V_T, S, P\rangle$ este recursivă la stânga dacă există $A \in V_N$ a.î. $A \stackrel{+}{\Rightarrow} A\alpha$ cu $\alpha \neq \lambda$.

Recursivitatea la stânga generează conflicte *First*/*First* cu toate alternativele neterminalului recursiv.

Exemplu:

Dacă avem producțiile:
$$\begin{array}{l} i_0 : E \to E + T \\ i_1 : E \to \alpha_1 \\ i_2 : E \to \alpha_2 \\ \dots \end{array}$$

atunci pentru orice j și orice $u \in First_1(\alpha_j) \setminus \{\lambda\}$ avem $u \in First_1(E + T)$, deci în intrarea $M(u, E)$ va ajunge și (α_j, i_j) și $(E + T, i_0)$.

Soluția: **eliminarea recursivității la stânga** (am prezentat algoritmul mai devreme).

Exerciții seminar:

La exercițiile 1 - 6 neterminalele sunt litere mari, terminalele litere mici, simbolul de start este S; mulțimile neterminalelor, terminalelor, resp. producțiilor se deduc din context.

1. Eliminați simbolurile neutilizate pentru următoarele GIC:
a) $S \to AB|CA$, $A \to a$, $B \to BC|AB$, $C \to aB|b$;
b) $S \to AB|a$, $A \to bB|cC$, $B \to SA|A$, $C \to aA|bB|S$;
c) $S \to SS|AAa$, $A \to aAB|aS|b$.

2. Determinați dacă $L(G) = \emptyset$ pentru următoarele GIC:
a) $S \to AS|A$, $A \to aB|bA$;
b) $S \to ABC$, $A \to BB|\lambda$;
c) $B \to CC|a$, $C \to AA|b$.

3. Eliminați λ-producțiile pentru următoarele GIC:
a) $S \to ABC$, $A \to BB|\lambda$, $B \to CC|a$, $C \to AA|b$
b) $S \to aSb|bSa|SS|\lambda$.

4. Eliminați redenumirile și apoi simbolurile inaccesibile pentru următoarea GIC: $S \to A$, $A \to bS|b$.

Exerciții seminar:

5. Caracterizați gramaticile de tip $LL(0)$.

6. Arătați că GIC: $S \to aAaa|bAba$, $A \to b|\lambda$
   este $LL(2)$ dar nu $LL(2)$ tare.

7. Aplicați algoritmul de parsare pentru gramatici $LL(1)$ (tare) pentru GIC:
   $1 : S \to F$, $2 : S \to (S + F)$, $3 : F \to 1$
   și cuvântul $w = (1 + 1)$.

Analiza sintactică

Teme laborator:

1. Scrieți un program care citește o GIC, apoi construiește și afișază GIC proprie și fără recursivitate la stânga echivalentă cu ea.
Algoritmii componenți (B), (A), (C), (D), eliminarea recursivității la stânga vor fi implementați ca proceduri/funcții separate, care vor primi gramaticile sursă și destinație ca parametri.

2. Implementați algoritmul de parsare general top - down.
Programul va citi o GIC (presupusă a fi fără simboluri neutilizate și nerecursivă la stânga), apoi într-un ciclu va citi diverse cuvinte și va determina dacă fac parte din limbajul generat de gramatică; în caz afirmativ, va afișa și producțiile aplicate într-o derivare stângă a sa.

3. Implementați algoritmul de parsare general top - down folosind o procedură/funcție recursivă, care la fiecare apel să facă o derivare. Celelalte detalii sunt ca la problema 2.

Teme laborator:

4. Implementați algoritmul de parsare pentru gramatici de tip $LL(1)$ (tare). Detaliile sunt asemănătoare celor din problema 2.

5. Implementați o variantă recursivă a algoritmului de parsare pentru gramatici de tip $LL(k)$ tare, pentru o gramatica fixată prin cod:

Pentru fiecare neterminal A se asociază o procedură/funcție, care consultă k simboluri din cuvântul de intrare începând de la poziția curentă (reținută într-o variabilă globală sau transmisă ca parametru), determină (unic) producția lui A care trebuie aplicată, îi afișază numărul, iar pentru fiecare simbol din membrul său drept, dacă e terminal avansează poziția curentă în cuvântul de intrare, iar dacă e neterminal apelează procedura/funcția acestuia. Tabela de parsare este implementată prin codul acestor proceduri/funcții.

Alternativ, se poate folosi o singură procedură/funcție recursivă, care primește neterminalul ca parametru. De asemenea, gramatica poate să nu fie fixată prin cod, ci programul să o citească, să construiască tabela de parsare într-o structură de date (de ex. o matrice), iar procedura/funcția unică să fie un cod general, care lucrează pe această structură de date.

Alte detalii sunt asemănătoare celor din problema 2.