FUNDAMENTELE PROIECTĂRII COMPILATOARELOR

CURSUL 6

PARSER DESCENDENT (TOP-DOWN) GENERAL

Fie $G = (N, \Sigma, S, P)$ gramatică independentă de context care descrie sintaxa unui limbaj de programare. Producțiile gramaticii sunt numerotate cu $1, \dots, |P|$.

Considerăm mulțimea:

$$\mathcal{C} = \Sigma^*\# \times (N \cup \Sigma)^*\# \times \{1, \dots, |P|\}^*, \text{ unde:}$$

- $\mathcal{C}$ – mulțimea configurațiilor parserului
- $\#$ simbol nou, $\# \notin N \cup \Sigma$

Numim configurație a parser-ului un triplet $c = (x\#, \alpha\#, \pi) \in \mathcal{C}$, unde $x \in \Sigma^*$,

$\alpha \in (N \cup \Sigma)^*, \pi \in \{1, \dots, |P|\}^*$, iar:

- $x\#$ este cuvântul ce a rămas de analizat pe banda de intrare
- $\alpha\#$ reprezintă conținutul stivei, cu $\#$ la bază
- $\pi = p_1 p_2 \dots p_k$, $p_1, p_2, \dots, p_k \in \{1, \dots, |P|\}$ reprezintă numerele de ordine ale producțiilor aplicate până la momentul actual.

Definiție. Parserul descendent atașat g.i.c. $G = (N, \Sigma, S, P)$ constă din perechea $(\mathcal{C}_0, \vdash)$, unde:

- $\mathcal{C}_0 = \{(w\#, S\#, \lambda) | w \in \Sigma^*\} \subseteq \mathcal{C}$ este mulțimea configurațiilor inițiale;
- $\vdash \subseteq \mathcal{C} \times \mathcal{C}$ reprezintă relația de tranziție între configurații pentru care avem regulile:

1) $(u\#, A\gamma\#, \pi) \vdash (u\#, \beta\gamma\#, \pi r)$, unde $r: A \to \beta \in P$
2) $(av\#, a\gamma\#, \pi) \vdash (v\#, \gamma\#, \pi), \forall\, a \in \Sigma$
3) $(\#, \#, \pi)$ configurația de acceptare
4) O configurație c pentru care nu există c' cu $c \vdash c'$ în sensul 1)-2), o numim configurație de eroare

Notam cu $\vdash^*$ închiderea reflexivă și tranzitivă a relației $\vdash$ și cu $\vdash^+$ inchiderea tranzitivă a acestei relații.

Parserul descris mai sus corespunde translatorului stivă nedeterminist T_G (vezi cursul 4), în care nu mai este pusă în evidență starea, deoarece nu este relevantă.

Dacă pentru neterminalul A avem două producții $r_1: A \to \beta_1, r_2: A \to \beta_2, \beta_1 \neq \beta_2$, atunci la pasul 1) putem alege (nedeterminist) oricare dintre cele 2 producții, adică din configurația $(u\#, A\gamma\#, \pi)$ putem să trecem fie în $(u\#, \beta_1\gamma\#, \pi r_1)$, fie în $(u\#, \beta_2\gamma\#, \pi r_2)$.

Lema 1. Dacă în parserul descendent $(\mathcal{C}_0, \vdash)$ atașat g.i.c. $G = (N, \Sigma, S, P)$ avem:

$$(uv\#, \gamma\#, \lambda) \vdash^* (v\#, \delta\#, \pi), u, v \in \Sigma^*, \gamma, \delta \in (N \cup \Sigma)^*, \pi \in \{1, \dots, |P|\}^*$$

atunci în G avem derivarea stângă:

$$\gamma \overset{\pi}{\Rightarrow}_s u\delta$$

Demonstrație. Facem inducție după $n = |\pi|$.

Baza. $n = 0$. Rezultă că în secvența $(uv\#, \gamma\#, \lambda) \vdash^* (v\#, \delta\#, \lambda)$ s-au aplicat doar tranziții de tipul 2), deci $\gamma = u\delta$ și atunci evident $\gamma \overset{\pi}{\Rightarrow}_s u\delta, \pi = \lambda$.

Ipoteza de inducție. Presupunem afirmația din enunț adevărată pentru orice π cu $|\pi| = n, n$ dat.

Saltul inductiv. Fie π cu $|\pi| = n + 1$.

Atunci $\pi = \pi' r, |\pi'| = n, r: A \to \beta \in P$ este ultima productie aplicată. Putem scrie:

$(uv\#, \gamma\#, \lambda) = (u'u''v\#, \gamma\#, \lambda) \vdash^* (u''v\#, A\gamma'\#, \pi') \vdash (u''v\#, \beta\gamma'\#, \pi' r) = (u''v\#, u''\delta\#, \pi) \vdash^* (v\#, \delta\#, \pi)$, unde $u = u'u'', \beta\gamma' = u''\delta$.

În conformitate cu ipoteza de inducție, deoarece $(u'u''v\#, \gamma\#, \lambda) \vdash^* (u''v\#, A\gamma'\#, \pi')$, rezultă că $\gamma \overset{\pi'}{\Rightarrow}_s u'A\gamma'$. Deoarece $u' \in \Sigma^*$, rezultă că A este cel mai din stânga neterminal, deci aplicând $r: A \to \beta$ obținem:

$$\gamma \overset{\pi'}{\Rightarrow}_s u'A\gamma' \overset{r}{\Rightarrow} u'\beta\,\gamma' = u'u''\delta = u\delta,$$

adică $\gamma \overset{\pi}{\Rightarrow}_s u\delta$, qed.

Corolar 1. Dacă în parserul descendent $(\mathcal{C}_0, \vdash)$ asociat gramaticii $G = (N, \Sigma, S, P)$ are loc $(w\#, S\#, \lambda) \vdash^* (\#, \#, \pi)$, atunci $S \overset{\pi}{\Rightarrow}_s w$.

Lema 2. Fie $G = (N, \Sigma, S, P)$ g.i.c. și $(\mathcal{C}_0, \vdash)$ parserul descendent asociat. Dacă în G are loc derivarea stângă $\gamma \overset{\pi}{\Rightarrow}_s uA\delta, u \in \Sigma^*$, atunci $\forall v \in \Sigma^*$,

$$(uv\#, \gamma\#, \lambda) \vdash^* (v\#, A\delta\#, \pi)$$

Demonstrație. Facem inducție după $n = |\pi|$.

Baza. $n = 0$. Rezultă că $\gamma = uA\delta$, iar $(uv\#, uA\delta\#, \lambda) \vdash^* (v\#, A\delta\#, \lambda), \pi = \lambda$.

Ipoteza de inducție. Presupunem afirmația din enunț adevărată pentru orice π cu $|\pi| = n, n$ dat.

Saltul inductiv. Fie π cu $|\pi| = n + 1$. Atunci $\pi = \pi' r, |\pi'| = n$, iar în G avem:

$$\gamma \overset{\pi'}{\Rightarrow}_s u'A'\delta' \overset{r}{\Rightarrow}_s u'u''A\delta''\delta' = uA\delta$$

unde $r: A' \to u''A\delta'' \in P$, $u = u'u'', \delta = \delta''\delta'$. În conformitate cu ipoteza de inducție, deoarece $\gamma \overset{\pi'}{\Rightarrow}_s u'A'\delta'$, rezultă că:

$$(u'v'\#, \gamma\#, \lambda) \vdash^* (v'\#, A'\delta'\#, \pi'), \forall v' \in \Sigma^*.$$

Consideram $v' = u''v, v \in \Sigma^*$ oarecare. Atunci obținem:

$$(uv\#, \gamma\#, \lambda) = (u'u''v\#, \gamma\#, \lambda) \vdash^* (u''v\#, A'\delta'\#, \pi')$$
$$\vdash (u''v\#, u''A\delta''\delta'\#, \pi' r) \vdash^* (v\#, A\delta\#, \pi), \text{qed.}$$

Corolar 2. Dacă în $G = (N, \Sigma, S, P)$ are loc derivarea $S \overset{\pi}{\Rightarrow}_s w$, atunci în parserul descendent $(\mathcal{C}_0, \vdash)$ asociat gramaticii are loc $(w\#, S\#, \lambda) \vdash^* (\#, \#, \pi)$.

Demonstrație. Punem în evidență ultimul pas al derivării $S \overset{\pi}{\Rightarrow}_s w$:

$$S \overset{\pi'}{\Rightarrow} uA\delta \overset{r}{\Rightarrow} u\beta\delta, r: A \to \beta \in P, \pi = \pi' r, u\beta\delta = w \in \Sigma^*.$$

În conformitate cu Lema 2, pentru $\gamma = S, v = \beta\delta$, obținem:

$$(w\#, S\#, \lambda) = \left(u\beta\delta\#, S\#, \lambda\right) \vdash^* \left(\beta\delta\#, A\delta\#\,, \pi'\right) \vdash (\beta\delta\#, \beta\delta\#, \pi' r)$$

$$\vdash^* (\#, \#, \pi), \text{qed.}$$

Teorema 1. Pentru o g.i.c. $G = (N, \Sigma, S, P)$ și parserul descendent $(\mathcal{C}_0, \vdash)$ asociat avem $S \overset{\pi}{\Rightarrow}_s w$ dacă și numai dacă $(w\#, S\#, \lambda) \vdash^* (\#, \#, \pi)$ (altfel spus $w \in L(G)$ dacă și numai dacă $(w\#, S\#, \lambda) \vdash^*$ acceptare.

Demonstrație. Rezultă din Corolarele 1 și 2.

Această teoremă asigură corectitudinea parserului descendent (top-down).

PARSER PREDICTIV

Dacă în parserul descendent general, asociat gramaticii G, la regulile de de tipul

1) $(u\#, A\gamma\#, \pi) \vdash (u\#, \beta\gamma\#, \pi r)$, unde $r: A \to \beta \in P$

alegerea producției este unică, bazându-ne pe un anumit criteriu, obținem un parser determinist, numit 1-predictiv, iar derivările stângi rezultate sunt unice.

Vom arăta că pentru orice gramatică $LL(1)$ (tare) există un astfel de parser.

TABELA DE ANALIZĂ SINTACTICĂ 1-PREDICTIVĂ

Input: $G = (N, \Sigma, S, P)$ gramatică independentă de context fără simboluri inutilizabile

Output: Tabela de analiză sintactică 1-predictivă

$$M: N \times (\Sigma \cup \{\#\}) \to \left\{(\beta, r)\middle| r: A \to \beta \in P\right\} \cup \{eroare\}$$

Algoritm:

1. Se calculează mulțimile $First\,(X)$ pentru fiecare $X \in N \cup \Sigma \cup \{\#\}$, folosind algoritmul din cursul anterior (evident, $First\,(\#) = \{\#\}$).

2. Se calculează mulțimile $Follow\,(A)$ pentru fiecare $A \in N$, folosind algoritmul din cursul anterior, cu excepția faptului că **se inițializează $\boldsymbol{Follow\,(S)}$ cu $\{\#\}$ in loc de $\{\lambda\}$**. Configurația inițială este $(w\#, S\#, \lambda)$, deci după S urmează #, iar w este șirul analizat.
3. Pentru fiecare producție $r: A \to \alpha \in P$, unde $r \in \{1, \dots, |P|\}$ este numărul de ordine al producției, repetă
   3.1. Pentru fiecare $x \in \Sigma \cup \{\#\}$,$x \in First\,(\alpha \cdot Follow\,(A))$ setează $M[A, x] \leftarrow (\alpha, r)$
   3.2. Pentru fiecare y$\in \Sigma \cup \{\#\}, y \notin First\,(\alpha \cdot Follow\,(A))$ setează $M[A, y] \leftarrow eroare$

Observații.

- În tabela M se introduc doar intrări pentru $M[A, x]$ ca în 3.1, în rest se consideră automat că $M[A, y] = eroare$ (3.2.)
- Cu algoritmul de mai sus este posibil să obținem pentru un același neterminal A și același șir x două intrări diferite în tabela M. Aceasta înseamnă de fapt că există două producții distincte $r: A \to \alpha \in P, r': A \to \beta \in P, \alpha \neq \beta$ pentru care $x \in First\,\big(\alpha \cdot Follow\,(A)\big) \cap First\,(\beta \cdot Follow\,(A))$, deci $M[A, x] \leftarrow (\alpha, r)$ și $M[A, x] \leftarrow (\beta, r')$
- Rezultă că pentru o gramatică independentă de context, G, fără simboluri inutilizabile, tabela M nu are intrări multiple dacă și numai dacă G este $LL(1)$ (tare).
- Algoritmul de mai sus furnizează o metodă de a verifica dacă o gramatică independentă de context dată este $LL(1)$ (tare).

ANALIZORUL SINTACTIC PENTRU GRAMATICI $LL(1)$ (TARI)

Input: $G = (N, \Sigma, S, P)$ gramatică independentă de context de tip $LL(1)$ tare; tabela de analiză sintactică M construită ca mai sus; $w \in \Sigma^*$ șirul analizat.

Output: Derivarea stângă (unică), π, a lui w, dacă $w \in L(G)$.

Algoritm: Mișcările (tranzițiile) parserului sunt date de:

1. $(u\#, A\gamma\#, \pi) \vdash (u\#, \beta\gamma\#, \pi r)$, dacă $M[A, First\,(u\#)] = (\beta, r)$
2. $\big(av\#, a\gamma\#, \pi\big) \vdash \big(v\#, \gamma\#, \pi\big), \forall a \in\ \Sigma$
3. $(\#, \#, \pi) \vdash$ acceptare, $\pi \neq \lambda$
4. $(au\#, bv\#, \pi) \vdash$ eroare, pentru $a, b \in \Sigma, a \neq b$
5. $(u\#, A\gamma\#, \pi) \vdash$ eroare, dacă $M[A, First\,(u\#)] =$ eroare.

Observații. Parserul de mai sus este determinist, deoarece G fiind $LL(1)$ tare, rezultă că M nu are intrări multiple.

Exemplu. Fie gramatica G cu producțiile

$E \to TR$

$R \to +TR \mid *\,TR \mid \lambda$

$T \to (E) \mid a$

X	$First(X)$	$Follow(X)$
E	$(, a$	$\#,)$
T	$(, a$	$+, *, \#,)$
R	$+, *, \lambda$	$\#,)$

1. $E \to TR, \quad First\big(TR \cdot Follow(E)\big) = \{(, a\}$
2. $R \to +TR, \ \ First\big(+TR \cdot Follow(R)\big) = \{+\}$
3. $R \to *\,TR, \ \ First\big(*\,TR \cdot Follow(R)\big) = \{*\}$
4. $R \to \lambda, \ \ First\big(\lambda \cdot Follow(R)\big) = Follow(R) = \{\#,)\}$
5. $T \to (E), \ \ First\Big((E) \cdot Follow(T)\Big) = \{(\}$
6. $T \to a, \ \ First\big(a \cdot Follow(T)\big) = \{a\}$

Tabela de analiză sintactică 1-predictivă obținută:

M	a	$+$	$*$	$($	$)$	$\#$
E	$(TR, 1)$	Eroare	Eroare	$(TR, 1)$	eroare	eroare
T	$(a, 6)$	Eroare	Eroare	$((E), 5)$	eroare	eroare
R	Eroare	$(+TR, 2)$	$(* TR, 3)$	eroare	$(\lambda, 4)$	$(\lambda, 4)$

Deoarece M nu are intrări multiple, rezultă că G este de tip $LL(1)$ tare, deci este chiar de tip $LL(1)$.

Analizăm șirul $w = a * a$

$(a * a\#, E\#, \lambda) \vdash (a * a\#, TR\#,1) \vdash (a * a\#, aR\#,16) \vdash (* a\#, R\#,16) \vdash$

$(* a\#,* TR\#,163) \vdash (a\#, TR\#,163) \vdash (a\#, aR\#,1636) \vdash (\#, R\#,1636) \vdash$

$(\#, \#,16364) \vdash$ acceptare

16364 reprezintă derivarea stângă (unică) a lui $a * a$ din S.

ALGORITMUL EARLEY

- A fost dezvoltat de J. Earley în 1968. Este un algoritm de tip top-down polinomial, $O(n^3)$, unde n lungimea șirului de intrare. Algoritmul original funcționează doar pentru gramatici fără λ-producții.
- J. Aycock & R.N. Horspool au îmbunătățit algoritmul în anii 2001-2002, extinzându-l pentru orice tip de gramatică independentă de context. Complexitatea algoritmului este tot $O(n^3)$.
- Algoritmul Earley este folosit îndeosebi în parsarea limbajelor naturale.

Input: $G = (N, \Sigma, S, P)$ gramatică independentă de context; $w = a_1 a_2 \dots a_n \in \Sigma^*$ șirul analizat, $n = |w| \geq 0$.

Metoda: Se crează mulțimile $S_0, S_1, \dots, S_n$ de configurații Earley de forma $[A \to \alpha.\beta, j]$, unde $A \to \alpha\beta \in P, j$ pointer catre $S_j, 0 \leq j \leq n$. O configurație se adaugă la S_j dacă și numai dacă nu exista deja în S_j.

Semnificația configurației $[A \to \alpha.\beta, j]$:

- α este subșirul care a fost analizat
- β urmează să fie analizat

Gramatica G se extinde la $G' = (N \cup \{S'\}, \Sigma, S', P \cup \{S' \to S\})$. Evident, $L(G) = L(G')$.

Se inițializează S_0 cu $[S' \to .S, 0]$

Operațiile de creare a mulțimilor $S_0, S_1, \dots, S_n$ (pentru intrarea $w = a_1 a_2 \dots a_n$):

Scanarea. Dacă $[A \to \alpha.a\beta, j] \in S_i, 0 \leq i \leq n-1$ și $a = a_{i+1}$, atunci $S_{i+1} \leftarrow S_{i+1} \cup [A \to \alpha a.\beta, j]$ (inițial $S_{i+1} = \emptyset$).

Predicția. a) Dacă $[A \to \alpha.B\beta, j] \in S_i, 0 \leq i \leq n$, atunci $\forall B \to \gamma \in P$ $S_i \leftarrow S_i \cup [B \to .\gamma, i]$.

b) În plus, dacă $B \overset{*}{\Rightarrow} \lambda$, atunci $S_i \leftarrow S_i \cup [A \to \alpha B.\beta, j]$.

Completarea. Dacă $[A \to \alpha., j] \in S_i$ și $[B \to \beta.A\gamma, k] \in S_j, j \leq i$, atunci

$S_i \leftarrow S_i \cup [B \to \beta A.\gamma, k]$.

Observații.

1. Pentru fiecare mulțime S_i se aplică mai întâi operațiile de predicție și completare, făcând treceri repetate peste configarațiile lui S_i până când nu se mai adaugă noi configurații.
2. Scanarea presupune trecerea de la mulțimea S_i la $S_{i+1}, i < n$, dacă și numai dacă următorul simbol din șirul de intrare, a_{i+1}, apare într-o configurație din S_i având punctul în față, cu alte cuvinte apare într-o configurație $[A \to \alpha.a_{i+1}\beta, j]$ din S_i, fiind următorul simbol analizat plecând de la acea configurație. Dacă $i < n$ și S_{i+1} nu poate fi construită (în S_i nu există nicio

configurație în care $'.'$ apare în fața lui a_{i+1}, următorul simbol din intrare), atunci $w \notin L(G)$. STOP

3. În cazul predicției: atunci când $'.'$ se află în fața unui neterminal B, aceasta înseamnă că analiza va continua plecând de la acel neterminal, de aceea se introduc în mulțimea curentă S_i toate configurațiile de forma $[B \to .\gamma, i], B \to \gamma \in P$. Indicele asociat noii configurații va fi i, deoarece începând din acest punct se va continua derivarea stângă plecând de la B.
4. Ceea ce au adăugat Aycock & Horspool la algoritmul Earley este b) din predicție, care permite ca atunci când $[A \to \alpha.B\beta, j] \in S_i$ și $B \overset{*}{\Rightarrow} \lambda$, analiza să continue și cu β, care urmează după B.
5. În cazul operației de completare: dacă o configurație finală $[A \to \alpha., j]$ este în mulțimea curentă S_i, atunci ne „întoarcem" în S_j, de unde provine $A \to \alpha.$, și verifică dacă $'.'$ apare în fața neterminalului A în vreuna din configurațiile lui S_j $([B \to \beta.A\gamma, k])$. Dacă da, deoarece în S_i am parsat complet pe α (avem $[A \to \alpha., j] \in S_i$), este ca și cum am parsat A din configurația $[B \to \beta.A\gamma, k] \in S_j$, de aceea vom introduce în S_i configurația $[B \to \beta A.\gamma, k]$.
6. Dacă S_n este construită și în S_n apare $[S' \to S.,0]$ atunci $w \in L(G)$; altfel, $w \notin L(G)$.
7. Pentru a reconstitui derivările stângi ce permit generarea lui w, atunci când $w \in L(G)$, se păstrează o structură arborescentă între configurații, în funcție de modul în care decurg una din cealaltă.

Exemplu. Fie gramatica G: $S \to S + S \mid n;\ w = n + n, |w| = 3$

Construim mulțimile S_0, S_1, S_2, S_3.

$$S_0 \begin{bmatrix} S' \to & .S, & 0 \\ S \to & .S + S, & 0 \\ S \to & .n, & 0 \end{bmatrix} \qquad \underset{w_1 = 'n'}{S_1} \begin{bmatrix} S \to & n., & 0 \\ S' \to & S., & 0 \\ S \to & S. + S, & 0 \end{bmatrix}$$

$$\underset{w_2 = '+'}{S_2} \begin{bmatrix} S \to & S + .S, & 0 \\ S \to & .S + S, & 2 \\ S \to & .n, & 2 \end{bmatrix} \qquad \underset{w_3 = 'n'}{S_3} \begin{bmatrix} S \to & n., & 2 \\ S \to & S + S., & 0 \\ S \to & S. + S, & 2 \\ S' \to & S., & 0 \\ S \to & S. + S, & 0 \end{bmatrix} \leftarrow$$

Rezultă că $n + n \in L(G)$.