LUCRAREA 2

Simplificarea gramaticilor independente de context

Cuprinsul lucrării: Sunt prezentate şi exemplificate următoarele 4 modalităţi de simplificare a gramaticilor independente de context:

- Eliminarea λ-producţiilor
- Eliminarea producţiilor ciclice
- Eliminarea recursivităţii stânga
- Factorizarea stânga

Unele exemple din acest capitol au fost preluate şi adaptate din [3], [7] şi [1].

2.1 Eliminarea λ-producţiilor

O gramatică λ-free (lambda-free) este o gramatică independentă de context cu următoarele proprietăţi:

1. În nicio producţie a gramaticii şirul vid nu apare în partea dreaptă a acesteia
2. Dacă există o astfel de producţie, atunci ea este $S \to \lambda$, unde S este simbolul de start, iar S nu apare în partea dreaptă a niciunei alte producţii

Teoremă: Pentru orice gramatică independentă de context G, cu λ neaparţinând lui $L(G)$, există o gramatică echivalentă λ-free G'.
Demonstraţie:
Primul pas este crearea unei mulţimi V_N care cuprinde toţi neterminalii din G care pot fi transformaţi, direct sau indirect, în şirul vid (λ). Acest lucru se realizează în felul următor:

1. Pentru toate producţiile $A \to \lambda$, se adaugă A în V_N
2. Se repetă pasul următor până când nu se mai adaugă noi simboluri la mulţimea V_N: Pentru toate producţiile de forma $B \to A_1A_2...A_n$, unde $A_1, A_2, ..., A_n$ sunt deja în mulţimea V_N , se adaugă B la V_N.

Al doilea pas, după formarea mulţimii V_N, este formarea de noi producţii, adică crearea noii muţimi P' a gramaticii echivalente G'. Pentru a realiza acest lucru, se analizează toate producţiile din P de forma:

$$A \to x_1x_2...x_m, \quad m \geq 1,$$

unde fiecare $x_i \in V \cup T$. Pentru fiecare astfel de producţie din P, se adaugă în P' atât producţia respectivă, cât şi toate producţiile generate pe baza ei în care sunt înlocuite toate simbolurile neterminale din V_N cu λ, în toate combinaţiile posibile. Spre exemplu, dacă x_i şi x_j sunt ambele în mulţimea V_N, atunci în P' se va regăsi o producţie în care x_i va fi înlocuit cu λ, una în care x_j va fi înlocuit cu λ, şi respectiv încă o producţie în care atât x_i cât şi x_j vor fi înlocuite cu λ. Există o excepţie de la această regulă, şi anume cazul în care toate simbolurile x_i ale producţiei se regăsesc în mulţimea V_N, şi atunci producţia rezultată $A \to \lambda$ nu va fi adăugată lui P'. Noua gramatică G' astfel obţinută este o gramatică λ-free, echivalentă cu G.

Exemplu:
Se dă gramatica G definită de producţiile:

$$\begin{array}{l} S \to ABC \\ A \to BB|\lambda \\ B \to CC|a \\ C \to AA|b \end{array}$$

Se cere să se găsească gramatica λ-free G' echivalentă.
Rezolvare:
Primul pas constă din formarea mulţimii V_N.

1. Se adaugă initial simbolurile neterminale care pot fi transformate direct în λ. $V_N = \{A\}$

2. Pentru producţiile care au în partea dreaptă doar simboluri neterminale deja aflate în V_N, se adaugă la V_N simbolurile neterminale din partea stângă. Se repetă acest pas până când nu se mai adaugă noi simboluri la mulţimea V_N.

(a) $V_N = \{A, C\}$ (pentru producţia $C \to AA$)

(b) $V_N = \{A, C, B\}$ (pentru producţia $B \to CC$)

(c) $V_N = \{A, C, B, S\}$ (pentru producţia $S \to ABC$)

S-a obţinut deci mulţimea $V_N = \{S, A, B, C\}$. Pe baza acestei mulţimi, al doilea pas constă din generarea de noi producţii după regula expusă anterior. De exemplu, din prima producţie a lui P:

$$S \to ABC$$

Se obţin noile producţii:

$$S \to ABC,\ S \to AB,\ S \to AC,\ S \to BC,\ S \to A,\ S \to B,\ S \to C$$

în final, gramatica G' echivalentă obţinută este:

$$\begin{array}{c} S \to ABC|AB|AC|BC|A|B|C \\ A \to BB|B \\ B \to CC|C|a \\ C \to AA|A|b \end{array}$$

2.2 Eliminarea producţiilor ciclice

O gramatică are un ciclu dacă există un simbol neterminal A astfel încât $A \Rightarrow +A$ (A să fie derivabil din el însuşi printr-o succesiune de producţii). Un astfel de simbol se numeşte simbol ciclic.
Dacă o gramatică G este o gramatică λ-free, atunci toate ciclurile (dacă există) din G pot fi eliminate fără a afecta limbajul generat $L(G)$ folosind un algoritm specific [13].
Fie o gramatică independentă de context $G = (N, T, S, P)$. Pentru eliminarea ciclurilor din această gramatică, mulţimea producţiilor P va fi înlocuită cu noua mulţime P_c obţinută din P prin înlocuirea fiecărei producţii $A \to B$, unde B e un simbol ciclic, cu noi producţii de tipul $A \to \alpha$, astfel încât α nu este o variabilă ciclică şi există o producţie $C \to \alpha$ astfel încât $B \Rightarrow^* C$.
Gramatica echivalentă rezultată este $G_c = (N, T, S, P_c)$.
Exemplu:
Fie gramatica definită prin producţiile:

$$S \to X|Xb|SS$$
$$X \to S|a$$

Atât S cât şi X sunt simboluri ciclice pentru că: $S \Rightarrow +S$ şi $X \Rightarrow +X$. Aplicând algoritmul de mai sus, se elimină apariţiile simbolurilor ciclice în partea dreaptă a respectivelor producţii, obţinându-se gramatica echivalentă definită de noile producţii:

$$S \to a|Xb|SS$$
$$X \to Xb|SS|a$$

2.3 Eliminarea recursivităţii stânga

O gramatică recursivă stânga este o gramatică în care se regăseşte cel puţin un simbol neterminal A şi producţii specifice prin care să se ajungă la relaţia:

$$A \Rightarrow +A\alpha \text{ unde } \alpha \in (N \cup T)^*$$

Cu alte cuvinte, printr-o succesiune de producţii, simbolul neterminal A este derivat într-o formă în care acelaşi simbol apare primul în partea stângă a şirului rezultat.
Recursivitatea stânga este de două feluri:

1. Directă (imediată sau evidentă), în cazul în care există o producţie a gramaticii de forma $A \to A\alpha$,

2. Indirectă (sau ,,recursivitate în k paşi“) atunci când printr-un şir de producţii succesive se poate ajunge pornind de la simbolul neterminal A la forma propoziţională $A\alpha$.

În general, recursivitatea stânga directă poate fi îndepărtată relativ simplu, prin introducerea unui nou simbol neterminal şi a unor noi producţii în gramatica iniţială.
Presupunând că avem următoarele producţii ale neterminalului A care manifestă recursivitate stângă:

$$A \to A\alpha_1|A\alpha_2|...|A\alpha_n|\beta_1|\beta_2|...|\beta_m$$

cu proprietatea că niciun simbol β_i nu începe cu A.
Atunci putem înlocui producţiile lui A cu următoarele producţii în care introducem noul simbol neterminal A':

$$A \to \beta_1 A'|\beta_2 A'|...|\beta_m A'$$
$$A' \to \alpha_1 A'|\alpha_2 A'|...|\alpha_n A'|\lambda$$

Sintetizând, putem afirma că recursivitatea imediată se elimină prin înlocuirea unei producţii de forma:

$$A \to A\alpha|\beta$$

Cu două producţii de forma:

$$A \to \beta A'$$
$$A' \to \alpha A'|\lambda$$

Exemplu:
Eliminaţi recursivitatea stângă din gramatica definită prin producţiile:

$$E \to E + T|T$$
$$T \to T * F|F$$
$$F \to (E)|a$$

Unde $\{E, T, F\}$ sunt simboluri neterminale iar $\{a, +, *, (,)\}$ sunt simboluri terminale.

Rezolvare:
Parcurgem neterminalii pentru a identifica unde ne confruntăm cu recursivitate stângă. Pentru gramatica dată, neterminalii E şi T au producţii care prezintă recursivitate stângă imediată. în acest caz, rescriem producţiile neterminalilor respectivi conform raţionamentului descris mai sus, introducând două noi simboluri neterminale, E' şi, respectiv T'. Astfel, noile producţii sunt următoarele:

$$E \to TE'$$
$$E' \to +TE'|\lambda$$
$$T \to FT'$$
$$T' \to *FT'|\lambda$$
$$F \to (E)|a$$

Pentru eliminarea recursivităţii stânga atât directă cât şi indirectă există următorul algoritm (denumit Algoritmul lui Paull [10]):

```
Se aranjează neterminalii gramaticii într-o ordine oarecare A1, A2,
...  , An.
for i := 1 to n do begin
     for j := 1 to i - 1 do begin
     Se înlocuieşte fiecare producţie de forma A_i → A_j β cu producţiile:
```

```
        A_i → α_1β|α_2β|...|α_kβ
            unde
        A_j → α_1|α_2|...|α_k
        reprezintă toate producţiile actuale ale lui A_j
    end {for j}
Se elimină recursivitatea stângă imediată din producţiile lui Ai
dacă este cazul
end {for i}
```

Exemplu:
Eliminaţi recursivitatea stângă din gramatica definită prin producţiile:

$$S \to Aa|b$$
$$A \to Ac|Sd|\epsilon$$

Rezolvare:
Vom ordona simbolurile neterminale astfel: S, A ($S = A_1, A = A_2$).

- Când i = 1, nu se execută bucla după j şi se trece direct la eliminarea recursivităţii imediate pentru producţiile lui S (nu e cazul),
- Când i = 2 şi j = 1, se înlocuieşte producţia $A \to Sd$ cu producţiile $A \to Ac|Aad|bd|\epsilon$,
- În final, se elimină recursivitatea stângă imediată din producţiile lui A.

Se obţine astfel noua gramatică având producţiile:

$$S \to Aa|b$$
$$A \to bdA'|A'$$
$$A' \to cA'|adA'|\epsilon$$

2.4 Factorizarea stânga

Factorizarea stânga este o tehnică de transformare a unei gramatici independente de context utilă pentru a obţine o gramatică echivalentă ce poate fi utilizată în parsarea de tip top-down.
Ideea de bază în spatele factorizării stânga este ca atunci când nu este evident care din două producţii alternative ale aceluiaşi neterminal A este mai potrivită pentru a continua derivarea, aceste producţii pot fi rescrise astfel încât decizia se amână până când se cunosc suficiente caractere de intrare pentru a face alegerea corectă.
Pentru a exemplifica, se consideră următoarele producţii:

$$\begin{aligned} S &\rightarrow if\ E\ then\ S \\ &\mid if\ E\ then\ S\ else\ S \end{aligned}$$

La întâlnirea unui atom de intrare ,,if", nu se poate lua o decizie imediată cu privire la derivarea următoare a lui S.
În general, dacă $A \rightarrow \alpha\beta_1|\alpha\beta_2$ sunt două producţii ale lui A iar şirul de intrare începe cu un şir de caractere nevid derivate din α, nu putem şti dacă să derivăm în continuare în $\alpha\beta_1$ sau $\alpha\beta_2$.
Cu toate acestea, se poate modifica gramatica pentru a îndepărta acestă problemă. Regula este să se modifice producţiile de forma

$$A \rightarrow \alpha\beta_1|\alpha\beta_2$$

în următoarele producţii noi, prin introducerea unui nou simbol neterminal A':

$$\begin{aligned} A &\rightarrow \alpha A' \\ A' &\rightarrow \beta_1|\beta_2 \end{aligned}$$

Aplicând acest raţionament, putem rescrie gramatica pentru instrucţiunea if enunţată mai sus astfel:

$$\begin{aligned} S &\rightarrow if\ E\ then\ S\ ElsePart \\ ElsePart &\rightarrow else\ S\ |\epsilon \end{aligned}$$

Exemplu:
Fie gramatica definită de producţiile:

$$\begin{aligned} S &\rightarrow Aa|Ab \\ A &\rightarrow aabA|aabaA|aa \end{aligned}$$

Se cere să se obţină gramatica echivalentă cu aceasta, factorizată stânga.

Rezolvare:
Din analiza gramaticii se observă faptul că trebuie factorizate stânga atât producţiile simbolului neterminal S cât şi A. Primul pas îl reprezintă factorizarea producţiilor lui S, pentru care introducem un nou simbol neterminal S'. Astfel, primele două producţii devin:

$$\begin{aligned} S &\rightarrow AS' \\ S' &\rightarrow a|b \end{aligned}$$

În cazul producţiilor lui A observăm prefixul aa comun, şi aplicând raţionamentul prin introducerea unui nou simbol neterminal B, acestea devin:

$$A \to aaB$$
$$B \to bA|baA|\lambda$$

Se observă însă faptul că primele două producţii ale lui B prezintă şi ele un prefix comun, şi anume b. După factorizarea acestora prin introducerea neterminalului C, obţinem:

$$B \to bC|\lambda$$
$$C \to A|aA$$

Concluzionând, gramatica echivalentă factorizată stânga este:

$$S \to AS'$$
$$S' \to a|b$$
$$A \to aaB$$
$$B \to bC|\lambda$$
$$C \to A|aA$$

2.5 Probleme propuse

1. Pentru gramatica definită de producţiile de mai jos, eliminaţi toate λ-producţiile. Determinaţi gramatica echivalentă.

$$S \to aCbb$$
$$B \to CD$$
$$C \to D|a|\lambda$$
$$D \to B|b|\lambda$$

2. Pentru gramatica definită de producţiile de mai jos, eliminaţi toate λ-producţiile. Determinaţi gramatica echivalentă.

$$S \to AaB|aaB$$
$$A \to \lambda$$
$$B \to bbA|\lambda$$

3. Eliminaţi recursivitatea stângă din gramatică definită de producţiile de mai jos. Determinaţi gramatica echivalentă.

$$S \to Aa|Bb$$
$$A \to Sa|a$$
$$B \to Bb|b$$

4. Eliminaţi recursivitatea stângă din gramatică definită de producţiile de mai jos. Determinaţi gramatica echivalentă.

$$S \to S_1\$$$
$$S_1 \to S_1 + T|T$$
$$T \to T * F|F$$
$$F \to [S_1]|a$$

Observaţie: {$, +, *, [,], a} sunt simboluri terminale.

5. Factorizaţi stânga gramatica definită de producţiile de mai jos. Determinaţi gramatica echivalentă.

$$S \to abA|aa$$
$$A \to bb|bS$$