Limbaje Formale și Compilatoare (LFC)

- Curs -

Ş.l.dr.ing. Octavian MACHIDON

octavian.machidon@unitbv.ro

Universitatea Transilvania din Brașov

Astăzi

- Simplificarea gramaticilor independente de context
    1. Eliminarea λ-producțiilor
    2. Eliminarea producțiilor ciclice
    3. Eliminarea recursivității stânga
    4. Factorizarea stânga

1. Eliminarea λ-producțiilor

- O gramatica λ-free (lambda-free) este o gramatica independentă de context cu următoarele proprietăți:
    1. În nicio producție a gramaticii șirul vid nu apare în partea dreaptă a acesteia
    2. Dacă există o astfel de producție, atunci ea este $S \rightarrow \lambda$, unde S este simbolul de start, iar S nu apare în partea dreapta a niciunei alte producții

Teoremă

- Pentru orice gramatică independentă de context G, cu λ neaparținând lui L(G), există o gramatică echivalentă λ-free G'.

Demonstrație:

- Primul pas este crearea unei mulțimi V_N care cuprinde toți neterminalii din G care pot transformați, direct sau indirect, în șirul vid (λ). Acest lucru se realizează în felul următor:

  1. Pentru toate productiile $A \rightarrow \lambda$, se adauga A în V_N
  2. Se repetă pasul următor până când nu se mai adaugă noi simboluri la mulțimea V_N:

     Pentru toate producțiile de forma $B \rightarrow A_1A_2...A_n$, unde $A_1, A_2, ..., A_n$ sunt deja în mulțimea V_N, se adauga B la V_N.

Teoremă

- Al doilea pas, dupa formarea multimii V_N, este formarea de noi producții, adică crearea noii mulțimi P' a gramaticii echivalente G'.
- Pentru a realiza acest lucru, se analizează toate producțiile din P de forma: $A \rightarrow x_1x_2...x_m$, $m \geq 1$, unde fiecare $x_i \in V \cup T$.
- Pentru fiecare astfel de producție din P, se adaugă în P' atât producția respectivă, cât și toate producțiile generate pe baza ei în care sunt înlocuite toate simbolurile neterminale din V_N cu λ, în toate combinațiile posibile.

Teoremă

- Spre exemplu, daca x_i si x_j sunt ambele în mulțimea V_N, atunci în P’ se va regăsi o producție în care x_i va fi înlocuit cu λ, una în care x_j va fi înlocuit cu λ, și respectiv încă o producție în care atât x_i cât și x_j vor fi înlocuite cu λ.
- Există o excepție de la această regulă, și anume cazul în care toate simbolurile x_i ale producției se regăsesc în mulțimea V_N, și atunci producția rezultata $A \to \lambda$ nu va fi adaugata lui P’.
- Noua gramatica G’ astfel obținută este o gramatică λ-free, echivalentă cu G.

Exemplu

Se dă gramatica G definită de producţiile:

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

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 $S \to ABC$)

Rezolvare (continuare)

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:

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

2. Eliminarea producțiilor 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 nouă 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:

$$
\begin{gathered}
S \to X|Xb|SS \\
X \to S|a
\end{gathered}
$$

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:

$$
\begin{gathered}
S \to a|Xb|SS \\
X \to Xb|SS|a
\end{gathered}
$$

3. Eliminarea recursivității 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 $A\alpha$.

Eliminarea recursivității stânga (cont.)

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':

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

Eliminarea recursivității stânga (cont.)

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:

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

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:

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

Algoritmul lui Paull

- Folosit pentru eliminarea recursivității stânga atât directă cât și indirectă

```
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 Ai → Ajβ cu producţiile:
          Ai → α1β|α2β|...|αkβ
              unde
          Aj → α1|α2|...|αk
          reprezintă toate producţiile actuale ale lui Aj
     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:

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

4. Factorizarea stânga

- Factorizarea stânga este o tehnică de transformare a unei gramatici independente de context utilă pentru a obține o gramatică echivalentă ce poate fi utilizată în parsarea de tip top-down (analiza sintactică).
- Ideea de baza în spatele factorizării stânga este că atunci când nu e evident care din două producții alternative ale aceluiași neterminal A este mai potrivită pentru a continua derivarea, aceste producții pot fi rescrise astfel încât decizia se amână pâna când se cunosc suficiente caractere de intrare pentru a face alegerea corectă.

Exemplul 1

- Fie gramatica:

$$S \to if\ E\ then\ S$$
$$\mid if\ E\ then\ S\ else\ S$$

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 $A \to \alpha\beta_1|\alpha\beta_2$ 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

$$A \to \alpha\beta_1|\alpha\beta_2$$

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

$$A \to \alpha A'$$
$$A' \to \beta_1|\beta_2$$

Exemplul 1 (cont.)

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

$$S \rightarrow if\ E\ then\ S\ ElsePart$$
$$ElsePart \rightarrow else\ S\ |\epsilon$$

Exemplul 2

Fie gramatica definită de producţiile:

$$S \to Aa|Ab$$
$$A \to aabA|aabaA|aa$$

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:

$$S \to AS'$$
$$S' \to a|b$$

Exemplul 2 (cont.)

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:

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

Cursul viitor:

- Automate finite de stari
  - Definiție și reprezentări
  - Automate fiinite deterministe (AFD)
  - Automate finite nedeterministe (AFN)
  - Transformarea AFN în AFD

Întrebări?