Limbaje formale, automate şi compilatoare

Curs 10

- Analiza sintactică descendentă
  - Parser descendent general
- Gramatici LL(1)
  - Definiţie
  - Caracterizare
  - FIRST, FOLLOW
  - Tabela de parsare
  - Algoritmul de analiză sintactică LL(1)

Analizor sintactic descendent. Configurații

- O configuraţie (u#, γ#,π) este interpretată în felul următor:
  - –γ# este conţinutul stivei cu simbolul # la bază.
  - –u# este conţinutul intrării.
  - –π este conţinutul ieşirii.

Analizor sintactic descendent. Tranziții

- Dacă C este mulţimea configuraţiilor atunci $\vdash \subseteq C \times C$ este relaţia de tranziţie definită astfel:
  - $(u\#, A\gamma\#, \pi) \vdash (u\#, \beta\gamma\#, \pi r)$, unde $r = A \rightarrow \beta \in P$. (aplicare regulă, expandare)
  - $(uv\#, u\gamma\#, \pi) \vdash (v\#, \gamma\#, \pi)$. (potrivire)
  - –$(\#, \#, \pi)$ este configuraţie de acceptare dacă $\pi \neq \varepsilon$.
  - –O configuraţie c pentru care nu există c' astfel ca $c \vdash c'$ produce eroare.
  - Configuraţii iniţiale: $(w\#, S\#, \varepsilon)$ unde $w \in T^*$

Exemplu

- 1.E→E+T, 2.E→T, 3.T→T*F, 4.T→F, 5.F→(E), 6.F→a

Intrare	Stivă	Ieşire
a+a*a#	E#	
a+a*a#	E+T#	1
a+a*a#	T+T#	12
a+a*a#	F+T#	124
a+a*a#	a+T#	1246
a*a#	T#	1246
a*a#	T*F#	12463
a*a#	F*F#	124634
a*a#	a*F#	1246346
a#	F#	1246346
a#	a#	12463466
#	#	12463466

Corectitudinea analizorului sintactic

- **Teorema** (de corectitudine a analizorului)
  - Fie gramatica redusă G=(N,T,S,P) şi w∈T*. Atunci, are loc $(w\#,S\#,\varepsilon)\vdash^{+}(\#,\#,\pi)$ (acceptare) dacă şi numai dacă w∈L(G) şi π este o derivare exrem stângă a cuvântului w.
- **Lema 1**
  - Dacă în analizorul sintactic descendent ataşat gramaticii G=(N,T,S,P) are loc calculul $(uv\#,u\gamma\#,\varepsilon)\vdash^{+}(v\#,\psi\#,\pi)$, atunci în gramatica G are loc derivarea $\gamma^{\pi}\Rightarrow_{st}u\psi$, oricare ar fi u,v∈T*, γ, ψ∈Σ*, π∈P*.
- **Lema 2**
  - Dacă în gramatica G are loc derivarea $\gamma^{\pi}\Rightarrow_{st}u\psi$ şi 1:ψ∈N∪{ε} atunci în parserul descendent are loc calculul: $(uv\#,u\gamma\#,\varepsilon)\vdash^{+}(v\#,\psi\#,\pi)$, ∀v∈T*.
- Demonstraţii: inducţie după lungimea lui π

Observații

- Parserul este nedeterminist
- Există tipuri de gramatici pentru care este determinist?
- Construirea unei tabele de parsare

Gramatici LL(k)

- LL(k): Parsing from **L**eft to right using **L**eftmost derivation and **k** symbols lookahead.
- Informal, o gramatică este LL(k) dacă tranziţia de tip "aplicare producţie" din parser se face cu o unică regulă $A \rightarrow \beta$, determinată prin următoarele k simboluri de la intrare.
- Definiţie
  - $k{:}\alpha$ reprezintă primele k simboluri din α (sau α dacă $|\alpha|<k$)
  - $\alpha{:}k$ reprezintă ultimele k simboluri din α (sau α dacă $|\alpha|<k$)

Gramatici LL(k)

- **Defininiţie**
  - O gramatică independentă de context redusă este gramatică LL(k), k≥1, dacă pentru orice două derivări de forma:
    - $S \Rightarrow^*_{st} uA\gamma \Rightarrow_{st} u\beta_1\gamma \Rightarrow^*_{st} ux$
    - $S \Rightarrow^*_{st} uA\gamma \Rightarrow_{st} u\beta_2\gamma \Rightarrow^*_{st} uy$
  - unde u, x, y∈T*, pentru care k:x = k:y, are loc $\beta_1 = \beta_2$.

Gramatici LL(k)

- **Teorema**
  - Orice gramatică LL(k) este neambiguă.
- **Teorema**
  - Dacă G este o gramatică stâng recursivă, atunci nu există nici un număr k astfel încât G să fie LL(k).
- **Teorema**
  - Clasele de limbaje LL(k)formează o ierarhie infinită:
    - $\mathscr{LL}(0) \subset \mathscr{LL}(1) \subset \mathscr{LL}(2) \subset \ldots \subset \mathscr{LL}(k) \subset \mathscr{LL}(k+1) \subset \ldots$
- **Lema**
  - Există limbaje care nu sunt LL(k) pentru nici o valoare $k \in N$.

Gramatici LL(1). Caracterizare

- **Teoremă**
  - O gramatică G = (N, T, S, P) este gramatică LL(1) dacă şi numai dacă pentru orice A∈N şi pentru orice două producţii $A \rightarrow \beta_1 | \beta_2$ are loc:

  - FIRST (β_1 FOLLOW (A))∩FIRST(β_2 FOLLOW (A)) = ∅

Tabela de parsare LL(1)

- 1.for(A∈N)
  - 2.for(a∈T∪{#})
    - 3.M(A,a)=∅;
- 4.for(p=A→β∈P){
  - 5.for(a∈FIRST(β)-{ε})
    - 6.M(A,a)=M(A,a)∪{(β,p)};
  - 7.if(ε∈FIRST(β)){
    - 8.for(b∈FOLLOW(A)){
      - 9.if(b== ε) M(A,#)=M(A,#)∪{(β,p)};
      - 10.else M(A,b)=M(A,b)∪{(β,p)};
    - }//endfor
  - }//endif
- }//endfor
- 11.for(A∈N)
  - 12.for(a∈T∪{#})
    - 13.if(M(A,a)=∅) M(A,a)={eroare};

Exemplu

- $S \rightarrow aSa \mid bSb \mid c$

M	a	b	c	#
S	(aSa, 1)	(bSb, 2)	(c, 3)	eroare

- $S \rightarrow aSa \mid bSb \mid a \mid b \mid c \mid \varepsilon$

M	a	b	c	#
S	(aSa, 1) (a, 3) (ε, 6)	(bSb, 2) (b, 4) (ε, 6)	(c, 5)	(ε, 6)

Parser LL(1)

- Configuraţia iniţială: (w#, S#, ε)
- Tranziţii
  - (u#, Aγ#,π) ⊢ (u#, βγ#,πr), dacă M(A, 1:u#)=(β, r), (expandare)
  - (uv#, uγ#,π) ⊢ (v#, γ#,π) (potrivire)
  - (#, #,π) ⊢ acceptare, dacă π ≠ ε (acceptare)
  - (au#, bγ#,π) ⊢ eroare dacă a ≠ b
  - (u#, Aγ#,π) ⊢ eroare dacă M(A, 1:u#)=eroare

Exemplu

- 1. S → E, 2. S → B, 3. E → ε, 4. B → a, 5. B → begin SC end, 6. C → ε , 7. C → ;SC

	FIRST	FOLLOW
S	**a, begin, ε**	**end, ;, ε**
E	**ε**	**end, ;, ε**
B	**a, begin**	**end, ;, ε**
C	**;, ε**	**end**

	a	begin	end	;	#
S	(B,2)	(B, 2)	(E, 1)	(E, 1)	(E, 1)
E	error	error	(ε, 3)	(ε, 3)	(ε, 3)
B	(a, 4)	(begin SC end, 5)	error	error	error
C	error	error	(ε, 6)	(;SC, 7)	error

Intrare	Stivă	Acţiune	Ieşire
begin a;;a end#	S#	expandare	2
begin a;;a end#	B#	expandare	5
begin a;;a end#	begin SC end#	potrivire	
a;;a end#	SC end#	expandare	2
a;;a end#	BC end#	expandare	4
a;;a end#	aC end#	potrivire	
;;a end#	C end#	expandare	7
;;a end#	;SC end#	potrivire	
;a end#	SC end#	expandare	1
;a end#	EC end#	expandare	3
;a end#	C end#	expandare	7
;a end#	;SC end#	potrivire	
a end#	SC end#	expandare	2
a end#	BC end#	expandare	4
a end#	aC end#	potrivire	
end#	C end#	expandare	6
end#	end#	potrivire	
#	#	acceptare	

Eliminarea recursiei stângi

- Fie G = (N, T, S, P) o gramatică în formă redusă
- Fie A∈N imediat recursiv
- Fie $A \rightarrow A\alpha_1|A\alpha_2|\ldots A\alpha_k|\beta_1|\beta_1|\ldots$ toate regulile care încep cu A. Fie P_A mulţimea acestor reguli.
- Gramatica G' unde A nu este recursiv imediat
  - G'=(N∪{A'}, T, S, P')
  - $P'=P-P_A \cup \{A' \rightarrow \alpha_1 A'|\alpha_2 A'|\ldots\alpha_k A'|\ \varepsilon,\ A \rightarrow \beta_1 A'|\beta_2 A'|\ldots\}$

Eliminarea recursiei stângi

- $E \rightarrow E+T \mid E\ -T \mid -T \mid T$
- $T \rightarrow T*F \mid T/\ F \mid F$
- $F \rightarrow (E) \mid a$

- $E \rightarrow TE' \mid -TE'$
- $E' \rightarrow +T\ E' \mid -TE' \mid \varepsilon$
- $T \rightarrow FT'$
- $T' \rightarrow *FT' \mid /FT' \mid \varepsilon$
- $F \rightarrow (E) \mid a$

Eliminarea recursiei stângi

	FIRST	FOLLOW
E	(a -	ε)
E'	+ - ε	ε)
T	(a	+ - ε
T'	* / ε	+ - ε)
F	(a	* / + - ε)

	a	+	-	*	/	(	)	#
E	(TE',1)	error	(-TE',2)	error	error	(TE',1)	error	error
E'	error	(+TE',3)	(-TE',4)	error	error	error	(ε, 5)	(ε, 5)
T	(FT', 6)	error	error	error	error	(FT', 6)	error	error
T'	error	(ε, 9)	(ε, 9)	(*FT',7)	(/FT',8)	error	(ε, 9)	(ε, 9)
F	(a, 11)	error	error	error	error	((E),10)	Error	error

Factorizare la stânga

- Factorizarea este o transformare aplicată asupra unei gramatici pentru a obţine o gramatică echivalentă, eventual LL(1)

- Dacă există producţiile

  - $A \rightarrow \alpha\beta_1$, $A \rightarrow \alpha\beta_2$ cu $|\alpha| > 1$

- (deci gramatica nu este LL(1)), acestea se înlocuiesc cu $A \rightarrow \alpha A'$, A' un nou neterminal, şi producţiile $A' \rightarrow \beta_1$ şi $A' \rightarrow \beta_2$. Metoda de transformare se numeşte **factorizare la stânga**

Factorizare la stânga: Exemplu

- Fie gramatica
  - S → if E then S | if E then S else S | a
  - E → b
- factorizarea la stânga: α = if E then S
- Gramatica echivalentă va fi
  - S → if E then S S’ | a
  - S’ → else S | ε
  - E → b

Factorizare la stânga: Exemplu

	a	**b**	**if**	**then**	**else**	**#**
S	(a,2)	error	(If E then S S',1)	error	error	error
S'	error	error	error	error	(else S,3) (ε,4)	(ε,4)
E	error	(b,5)	error	error	error	error

- Prin factorizare nu este sigur că obţinem o gramatică LL(1).

- Putem rezolva ambiguitatea alegând regula **S'→else S** în **M(S', else)**. Această alegere ar corespunde asocierii lui else pentru acel *if* precedent cel mai apropiat de el, soluţie adoptată de majoritatea limbajelor de programare.

Corectitudinea parserului LL(1)

- Este dovedită pe baza
  - Teoremei de corectitudine a parserului descendent general,
  - Teoremei de caracterizare a gramaticilor LL(1)
  - Modului în care a fost construită tabela de parsare

Bibliografie

- A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques, and Tools*, Second Edition. Addison-Wesley, 2007
- G. Grigoraş, *Construcția compilatoarelor. Algoritmi fundamentali*, Editura Universității "Alexandru Ioan Cuza", Iaşi, 2005