

# Analiza zstępująca. Gramatyki typu $LL(k)$

Języki formalne i techniki translacji - Wykład 9

Maciek Gębala

4 grudnia 2018

Maciek Gębala Analiza zstępująca. Gramatyki typu  $LL(k)$

## Analiza metodą zstępującą

- Mamy ciąg tokenów (terminali)  $w \in T^*$  i gramatykę  $G = (N, T, P, S)$ .
- Chcemy sprawdzić czy  $w \in L(G)$ ?
- Szukamy lewostronnego wyprowadzenia dla  $w$ .
- Budujemy drzewo wyprowadzenia dla  $w$  zaczynając od korzenia i tworząc wierzchołki w porządku preorder.

Maciek Gębala Analiza zstępująca. Gramatyki typu  $LL(k)$

## Przykład

- Weźmy napis  $w = cad$  i gramatykę

$$\begin{aligned} S &\rightarrow cAd \\ A &\rightarrow ab|a \end{aligned}$$

- Zaczynamy od symbolu początkowego, pierwszą literą  $w$  jest  $c$  więc bierzemy produkcję  $S \rightarrow cAd$  i wyprowadzamy  $S \Rightarrow cAd$
- Pierwsza litera  $cAd$  jest zgodna z pierwszą literą  $w$  więc przechodzimy do symboli  $A$  i  $a$ . Możemy teraz użyć produkcji  $A \rightarrow ab$  i otrzymać ciąg  $cabd$ .
- Druga litera się zgadza więc sprawdzamy trzecią. Niestety mamy  $d$  i  $b$  czyli źle. Musimy wrócić do  $A$  i poszukać alternatywnego wyprowadzenia.
- Używamy produkcji  $A \rightarrow a$  i tym razem jest dobrze.
- Uzyskaliśmy wyprowadzenie:  $S \Rightarrow cAd \Rightarrow cad$ .

Maciek Gębala Analiza zstępująca. Gramatyki typu  $LL(k)$

## Przykład

- Weźmy gramatykę

$$\begin{aligned} S &\rightarrow aAd|aB \\ A &\rightarrow b|c \\ B &\rightarrow ccd|ddc \end{aligned}$$

- Weźmy słowo  $w = accd$
- $S \Rightarrow aAd$
- $S \Rightarrow aAd \Rightarrow abd$  źle,  $abd \neq accd$ , nawracamy
- $S \Rightarrow aAd \Rightarrow acd$  źle,  $acd \neq accd$ , nawracamy
- $S \Rightarrow aB$
- $S \Rightarrow aB \Rightarrow accd$  OK

Maciek Gębala Analiza zstępująca. Gramatyki typu  $LL(k)$

Kiedy nie jest jasne na podstawie pierwszego symbolu, którą produkcję wybrać do rozwinięcia nieterminala, przekształcamy te produkcje tak aby decyzję podjąć później.

## Przykład

- $instr \rightarrow if\ wyr\ then\ instr\ else\ instr | if\ wyr\ then\ instr$
- $instr \rightarrow if\ wyr\ then\ instr\ koniec$   
 $koniec \rightarrow else\ instr | \varepsilon$

# Algorytm lewostronnej faktoryzacji gramatyki

- Dla każdego nieterminala  $A$  znajdź najdłuższy prefiks  $\alpha$  wspólny dla co najmniej dwóch prawych stron  $A$ -produkcji.
- Jeśli  $\alpha \neq \varepsilon$  to zamień produkcje

$$A \rightarrow \alpha\beta_1 | \dots | \alpha\beta_n | \gamma_1 | \dots | \gamma_m$$

na produkcje

$$\begin{aligned} A &\rightarrow \alpha B | \gamma_1 | \dots | \gamma_m \\ B &\rightarrow \beta_1 | \dots | \beta_n \end{aligned}$$

gdzie  $B$  jest nowym dodatkowym nieterminalem.

- Transformację powtarzamy dopóki zmienia ona gramatykę.

# Analizatory przewidujące

Często uważnie tworząc gramatykę, usuwając w niej lewostronną rekurencję i wykonując lewostronną faktoryzację możemy uzyskać gramatykę, która przy wyprowadzeniu nie potrzebuje nawrotów.

Wyprowadzenie w takiej gramatyce może być łatwo sprawdzane prostym, deterministycznym automatem ze stosem.

## Przykład

- Weźmy gramatykę

$$\begin{aligned} E &\rightarrow E+T | T \\ T &\rightarrow T*F | F \\ F &\rightarrow (E) | id \end{aligned}$$

- Eliminujemy lewostronną rekurencję

$$\begin{aligned} E &\rightarrow TG \\ G &\rightarrow +TG | \varepsilon \\ T &\rightarrow FV \\ V &\rightarrow *FV | \varepsilon \\ F &\rightarrow (E) | id \end{aligned}$$

- Gramatyka nie potrzebuje lewostronnej faktoryzacji.

| Stos  | Wejście          | Wyjście                     |
|-------|------------------|-----------------------------|
| \$E   | $id + id * id\$$ | $E \rightarrow TG$          |
| \$GT  | $id + id * id\$$ | $T \rightarrow FV$          |
| \$GVF | $id + id * id\$$ | $F \rightarrow id$          |
| \$GV  | $+id * id\$$     | $V \rightarrow \varepsilon$ |
| \$G   | $+id * id\$$     | $G \rightarrow +TG$         |
| \$GT  | $id * id\$$      | $T \rightarrow FV$          |
| \$GVF | $id * id\$$      | $F \rightarrow id$          |
| \$GV  | $*id\$$          | $V \rightarrow *FV$         |
| \$GVF | $id\$$           | $F \rightarrow id$          |
| \$GV  | $\$$             | $V \rightarrow \varepsilon$ |
| \$G   | $\$$             | $G \rightarrow \varepsilon$ |
| \$    | $\$$             |                             |

Zbiory *FIRST* i *FOLLOW*

- FIRST* pomagają wybrać produkcję którą możemy użyć do wyprowadzania napisu.

$$FIRST(\alpha) = \{x \in T : \exists \beta \alpha \Rightarrow^* x\beta\}$$

- FOLLOW* pomagają synchronizować symbole podczas odzyskiwania kontroli w trybie paniki.

$$FOLLOW(A) = \{x \in T : \exists \alpha \exists \beta S \Rightarrow^* \alpha Ax\beta\}$$

- Na podstawie tych funkcji generujemy tablicę analizatora przewidującego która dla nieterminala  $A$  i terminala  $a$  wyznacza którą produkcję możemy użyć.

Wyznaczanie *FIRST*( $X$ )

Dla wszystkich symboli  $X$  z gramatyki  $G$  tworzymy zbiory *FIRST* według następujących reguł

- Jeśli  $X$  jest terminalem to  $FIRST(X) = \{X\}$ .
- Jeśli  $X \rightarrow \varepsilon$  jest produkcją to do  $FIRST(X)$  dodajemy  $\varepsilon$ .
- Jeśli  $X$  jest nieterminalem i  $X \rightarrow Y_1 Y_2 \dots Y_k$  to  $a$  dodajemy do  $FIRST(X)$  jeżeli istnieje  $i$  takie, że  $a \in FIRST(Y_i)$  oraz  $\varepsilon \in FIRST(Y_j)$  dla każdego  $j < i$ .  $\varepsilon \in FIRST(X)$  jeśli należy do wszystkich  $FIRST(Y_i)$

Ponadto

- $FIRST(X\alpha) = FIRST(X)$  gdy  $\varepsilon \notin FIRST(X)$
- $FIRST(X\alpha) = FIRST(X) \cup FIRST(\alpha)$  gdy  $\varepsilon \in FIRST(X)$

Wyznaczanie *FOLLOW*( $A$ )

Dla wszystkich nieterminali  $A$  *FOLLOW*( $A$ ) tworzymy według następujących reguł

- Dla symbolu początkowego  $S$  do *FOLLOW*( $S$ ) dodajemy  $\$$ .
- Jeśli mamy produkcję  $A \rightarrow \alpha B \beta$  to do *FOLLOW*( $B$ ) dodajemy wszystkie symbole z  $FIRST(\beta)$  poza  $\varepsilon$ .
- Jeśli mamy produkcję  $A \rightarrow \alpha B$  albo produkcję  $A \rightarrow \alpha B \beta$ , gdzie  $\varepsilon \in FIRST(\beta)$  to do *FOLLOW*( $B$ ) dodajemy wszystkie symbole z *FOLLOW*( $A$ ).

## Przykład

- $FIRST(E) = FIRST(T) = FIRST(F) = \{ (, id \}$
- $FIRST(G) = \{ +, \varepsilon \}$
- $FIRST(V) = \{ *, \varepsilon \}$
- $FOLLOW(E) = FOLLOW(G) = \{ \}, \$ \}$
- $FOLLOW(T) = FOLLOW(V) = \{ +, \cdot, \$ \}$
- $FOLLOW(F) = \{ +, *, \cdot, \$ \}$

Maciek Gębala Analiza zstępująca, Gramatyki typu  $LL(k)$

## Budowa tablic analizatora przewidującego

Dla każdej produkcji  $A \rightarrow \alpha$

- 1 dla każdego  $a \in T$  jeśli  $a \in FIRST(\alpha)$  to wpisz  $A \rightarrow \alpha$  do  $M[A, a]$ .
- 2 jeśli  $\varepsilon \in FIRST(\alpha)$  to dla każdego  $b \in FOLLOW(A)$  wpisz  $A \rightarrow \alpha$  do  $M[A, b]$ .

Maciek Gębala Analiza zstępująca, Gramatyki typu  $LL(k)$

## Przykład

|     | $id$ | $+$           | $*$   | $($  | $)$           | $\$$          |
|-----|------|---------------|-------|------|---------------|---------------|
| $E$ | $TG$ |               |       | $TG$ |               |               |
| $G$ |      | $+TG$         |       |      | $\varepsilon$ | $\varepsilon$ |
| $T$ | $FV$ |               |       | $FV$ |               |               |
| $V$ |      | $\varepsilon$ | $*FV$ |      | $\varepsilon$ | $\varepsilon$ |
| $F$ | $id$ |               |       | $(E$ |               |               |

Maciek Gębala Analiza zstępująca, Gramatyki typu  $LL(k)$

## Gramatyki $LL(1)$

- Pierwsze L – przeglądanie od lewej do prawej.
- Drugie L – tworzenie lewostronnego wyprowadzenia.
- 1 – używanie do podejmowania decyzji jednego symbolu w każdym kroku.
- Dla każdego nieterminala z dwóch różnych prawych stron produkcji nie da się wyprowadzić ciągów zaczynających się od tego samego terminala.

Niestety nie wszystkie gramatyki bezkontekstowe dają się sprowadzić do postaci  $LL(1)$ .

Maciek Gębala Analiza zstępująca, Gramatyki typu  $LL(k)$

Notatki

Notatki

Notatki

Notatki

Gramatyka jest typu  $LL(1)$  gdy z (rozpatrujemy wyprowadzenia lewostronne)

$$S \Rightarrow^* wA\alpha \Rightarrow w\beta\alpha \Rightarrow^* wx, \quad S \Rightarrow^* wA\alpha \Rightarrow w\gamma\alpha \Rightarrow^* wy$$

oraz

$$FIRST(x) = FIRST(y)$$

wynika, że  $\beta = \gamma$ .

Inaczej: jeśli  $S \Rightarrow^* wA\alpha \Rightarrow w\beta\alpha \Rightarrow^* wx$ , to aby odgadnąć następny krok w wyprowadzeniu należy poznać pierwszy znak  $x$ .

## Testowanie na własność $LL(1)$

$G$  jest  $LL(1)$ , gdy dla  $S \Rightarrow^* wA\alpha$  i dowolnych produkcji  $A \rightarrow \beta$ ,  $A \rightarrow \gamma$  mamy

$$FIRST(\beta\alpha) \cap FIRST(\gamma\alpha) = \emptyset.$$

**Zalety własności:** FIRST można efektywnie obliczyć!

## Dowód własności

- Niech  $c \in FIRST(\beta\alpha) \cap FIRST(\gamma\alpha)$ ,  $\beta \neq \gamma$ . Wtedy potrafimy zbudować wyprowadzenia z  $\beta\alpha$  i  $\gamma\alpha$  uzyskując  $c$  na pierwszym miejscu. Otrzymamy  $S \Rightarrow^* wA\alpha \Rightarrow w\beta\alpha \Rightarrow^* wcx$  i  $S \Rightarrow^* wA\alpha \Rightarrow w\gamma\alpha \Rightarrow^* wcy$ , ponieważ  $FIRST(cy) = c = FIRST(cx)$ , z definicji  $LL(1)$  mielibyśmy  $\beta = \gamma$ . Zatem gramatyka nie jest  $LL(1)$ .
- Niech  $G$  nie będzie  $LL(1)$ , tj.: jeśli  $S \Rightarrow^* wA\alpha \Rightarrow w\beta\alpha \Rightarrow^* wcx$  i  $S \Rightarrow^* wA\alpha \Rightarrow w\gamma\alpha \Rightarrow^* wcy$ , wtedy  $c \in FIRST(\beta\alpha) \cap FIRST(\gamma\alpha)$ . (oczywiście!)

## Test na $LL(1)$

$G$  jest  $LL(1)$  wtedy i tylko wtedy, gdy dla każdego  $A$  i każdej produkcji  $A \rightarrow \beta \mid \gamma$ :  $FIRST(\beta FOLLOW(A)) \cap FIRST(\gamma FOLLOW(A)) = \emptyset$ .

**Dowód ( $\Leftarrow$ ) - z pustości wynika  $LL(1)$**

Oczywiście:

- Jeśli  $S \Rightarrow^* wA\alpha \Rightarrow w\beta\alpha \Rightarrow^* wx$  to  $FIRST(x)$  należą do  $FIRST(\beta FOLLOW(A))$ .
- Gdyby  $S \Rightarrow^* wA\alpha \Rightarrow w\gamma\alpha \Rightarrow^* wy$  oraz  $FIRST(x) = FIRST(y) = c$ , to mielibyśmy  $c \in FIRST(\beta FOLLOW(A)) \cap FIRST(\gamma FOLLOW(A))$ .
- Ale jest to niemożliwe.

## Test na $LL(1)$

Zakładamy, że  $c \in FIRST(\beta FOLLOW(A)) \cap FIRST(\gamma FOLLOW(A))$ .

Dowód: ( $\Rightarrow$ ) z  $LL(1)$  wynika pustość

- $c \in FIRST(\beta) \cap FIRST(\gamma)$  wtedy  $S \Rightarrow^* vA\alpha \Rightarrow v\beta\alpha \Rightarrow^* vcx$  dla pewnego  $x$ ,  $S \Rightarrow^* vA\alpha \Rightarrow v\gamma\alpha \Rightarrow^* vcy$  dla pewnego  $y$ , i  $FIRST(cy) = c = FIRST(cx)$ . Równocześnie  $\beta \neq \gamma$ , więc  $G$  nie jest  $LL(1)$ !
- $c \notin FIRST(\beta)$ ,  $c \notin FIRST(\gamma)$ , ale  $c \in FOLLOW(A)$ ;  $\varepsilon \in FIRST(\beta)$ ,  $\varepsilon \in FIRST(\gamma)$ . Wtedy:  
 $S \Rightarrow^* vA\alpha \Rightarrow v\beta\alpha \Rightarrow^* v\alpha \Rightarrow^* vcx$ . Także  
 $S \Rightarrow^* vA\alpha \Rightarrow v\gamma\alpha \Rightarrow^* v\alpha \Rightarrow^* vcx$ . Wbrew definicji  $LL(1)$ .

Maciek Gębala

Analiza zstępująca. Gramatyki typu  $LL(k)$

## Test na $LL(1)$

Dowód: ( $\Rightarrow$ ) z  $LL(1)$  wynika pustość

- $c \in FIRST(\gamma)$ ,  $c \notin FIRST(\beta)$ , ale  $c \in FOLLOW(A)$  i  $\varepsilon \in FIRST(\beta)$ . Wtedy:  $S \Rightarrow^* wA\alpha \Rightarrow w\beta\alpha \Rightarrow^* w\alpha \Rightarrow^* wcx$ . Także:  $S \Rightarrow^* wA\alpha \Rightarrow w\gamma\alpha \Rightarrow^* wcy\alpha \Rightarrow^* wcy\alpha$ . Ponieważ  $FIRST(cx) = c = FIRST(cy\alpha)$  oraz  $\beta \neq \gamma$ , mamy sprzeczność z  $LL(1)$ .

Maciek Gębala

Analiza zstępująca. Gramatyki typu  $LL(k)$

## Tabele parsowania dla $LL(1)$

- $M[A, a]$  zawiera  $A \rightarrow \beta$ , jeśli  $a \in FIRST(\beta FOLLOW(A))$ .
- Zgodnie z poprzednim twierdzeniem nie ma konfliktów dla  $LL(1)$ .
- Parsing z taką tabelą określa jednoznacznie jedyne możliwe wyprowadzenie lewostronne.

Maciek Gębala

Analiza zstępująca. Gramatyki typu  $LL(k)$

## Gramatyka spoza $LL(1)$

- Gramatyka  $G$

$$\begin{aligned} S &\rightarrow \varepsilon | abA \\ A &\rightarrow Saa | b \end{aligned}$$

- $G$  nie jest  $LL(1)$ :

$$\begin{aligned} S &\Rightarrow abA \Rightarrow abSaa \Rightarrow ababAaa \Rightarrow^* ab\underline{a}... \\ S &\Rightarrow abA \Rightarrow abSaa \Rightarrow ab\underline{a}a \end{aligned}$$

- Jeden znak nie wystarczy aby rozróżnić pomiędzy  $S \rightarrow \varepsilon$  i  $S \rightarrow abA$  dla drugiego kroku wyprowadzenia.

Maciek Gębala

Analiza zstępująca. Gramatyki typu  $LL(k)$

Notatki

Notatki

Notatki

Notatki

$FIRST_k(\alpha)$  zdefiniowane tak jak  $FIRST(\alpha)$ :

- $\exists \alpha \Rightarrow^* w$ ,  $w$  składa się z terminali,  $|w| \geq k$  i  $x$  jest prefiksem  $w$  długości  $k$ , to  $x \in FIRST_k(\alpha)$ , lub
- $\exists \alpha \Rightarrow^* w$ ,  $w$  składa się z terminali,  $|w| < k$  i  $x = w$ , to  $x \in FIRST_k(\alpha)$ .

## Gramatyka $LL(k)$

Gramatyka jest  $LL(k) \Leftrightarrow$  z

$$\begin{aligned} S &\Rightarrow^* wA\alpha \Rightarrow w\beta\alpha \Rightarrow^* wx \\ S &\Rightarrow^* wA\alpha \Rightarrow w\gamma\alpha \Rightarrow^* wy \end{aligned}$$

i

$$FIRST_k(x) = FIRST_k(y)$$

wynika, że  $\beta = \gamma$ .

Inaczej: jeśli  $S \Rightarrow^* wA\alpha \Rightarrow w\beta\alpha \Rightarrow^* wx$ , to wystarczy poznać  $k$  znaków  $x$  aby określić następny krok wyprowadzenia  $S \Rightarrow^* wA\alpha$ .

## Gramatyka nie- $LL(k)$ dla dowolnego $k$

Przykład

Gramatyka

$$\begin{aligned} S &\rightarrow A|B \\ A &\rightarrow aAb|0 \\ B &\rightarrow aBbb|1 \end{aligned}$$

definiuje

$$\{a^n 0 b^n : n \geq 0\} \cup \{a^n 1 b^{2n} : n \geq 0\}.$$

Jest to język rozpoznawany deterministycznym automatem ze stosem, ale nie jest w żadnym  $LL(k)$ .  
(Blok symboli  $a$  może być dowolnie długi i nie można się zdecydować na  $S \Rightarrow A$  lub  $S \Rightarrow B$ .)

## Podstawowa własność

$G$  jest  $LL(k)$  wtedy i tylko wtedy, gdy dla dowolnych  $S \Rightarrow^* wA\alpha$ , oraz  $A \rightarrow \beta$ ,  $A \rightarrow \gamma$  mamy

$$FIRST_k(\beta\alpha) \cap FIRST_k(\gamma\alpha) = \emptyset$$

Następująca własność nie jest równoważna  $LL(k)$ !

Jeśli  $A \rightarrow \beta$ ,  $A \rightarrow \gamma$ ,  $\beta \neq \gamma$ , to  
 $FIRST_k(\beta FOLLOW_k(A)) \cap FIRST_k(\gamma FOLLOW_k(A)) = \emptyset$

Przykład

$S \rightarrow aAaa|bAba$  i  $A \rightarrow b|\epsilon$ , to widać z definicji, że gramatyka jest  $LL(2)$ .

Ale:  $FOLLOW_2(A) = \{aa, ba\}$ ,

$$FIRST_2(b FOLLOW_2(A)) \cap FIRST_2(\epsilon FOLLOW_2(A)) = \{ba\}$$

## Konstrukcja parsera $LL(k)$

- $G$  - gramatyka  $LL(k)$ ,  $wx \in L_G$
- Konstruujemy lewostronne wyprowadzenie dla  $wx$ , zakładamy, że mamy  $S \Rightarrow^* w\alpha$ , takie że  $\alpha$  zaczyna się nieterminalem,  $\alpha \Rightarrow^* X$ .
- Z definicji, z  $w$  i  $k$  następnych znaków  $x$  można określić następny krok wyprowadzenia.
- **Problem:**  $w$  nie można przechować na stosie (tam jest  $\alpha$ ).

Maciek Gębala

Analiza zstępująca, Gramatyki typu  $LL(k)$

## Tabele $LL(k)$

### Notacja

$L_1 \oplus_k L_2 = \{w : \exists x \in L_1, \exists y \in L_2 (w = xy \wedge |xy| \leq k) \vee (w = FIRST_k(xy))\}$

$T_{A,L}$  trzeba traktować jednocześnie jako funkcję. Niech  $u$  ma długość  $k$ . Wtedy

- $T_{A,L}(u) = \text{error}$ , jeśli dla żadnej produkcji  $A \rightarrow \alpha$  nie zachodzi  $u \in FIRST_k(\alpha) \oplus_k L$ ,
- $T_{A,L}(u) = (A \rightarrow \alpha, \langle Y_1, \dots, Y_m \rangle)$  jeśli dla dokładnie jednej produkcji  $A \rightarrow \alpha$  mamy  $u \in FIRST_k(\alpha) \oplus_k L$ , ( $\langle Y_1, \dots, Y_m \rangle$  zdefiniowane poniżej)
- $T_{A,L}(u) = \text{error}$  jeśli dla więcej niż jednej produkcji  $A \rightarrow \alpha$  mamy  $u \in FIRST_k(\alpha) \oplus_k L$ , (dla języka  $LL(k)$  nie powinno to zajść).

Maciek Gębala

Analiza zstępująca, Gramatyki typu  $LL(k)$

## Definicja $\langle Y_1, \dots, Y_m \rangle$

Niech  $\alpha = x_0 B_1 x_1 B_2 x_2 \dots B_m x_m$ , gdzie  $B_i$  jest nieterminalem, a  $x_i$  to ciąg terminali. Wtedy:

$$Y_i = FIRST_k(x_i B_{i+1} \dots B_m x_m \oplus_k L).$$

$Y_i$  mówi jakie ciągi wyprowadzane za  $B_i$  są dopuszczalne o ile skorzystamy z  $A \rightarrow \alpha$ .

Maciek Gębala

Analiza zstępująca, Gramatyki typu  $LL(k)$

## Konstrukcja tabel $LL(k)$

Konstruujemy tylko takie  $T_{A,L}$ , które okazują się niezbędne: dla sytuacji początkowej  $\mathcal{I} = \{T_{S, \{\epsilon\}}\}$  rozszerzamy dopóty, dopóki coś nowego się pojawia. Reguła:

jeśli  $T \in \mathcal{I}$  i  $T(u) = (A \rightarrow x_0 B_1 x_1 B_2 x_2 \dots B_m x_m, \langle Y_1, \dots, Y_m \rangle)$ ,  
dodając  $T_{B_i, Y_i}$  do  $\mathcal{I}$ .

Maciek Gębala

Analiza zstępująca, Gramatyki typu  $LL(k)$

Notatki

Notatki

Notatki

Notatki



## Parser $LL(k)$

Notatki

- Maciek Gębala Analiza zstępująca. Gramatyki typu
- $LL(k)$

## Podstawowa własność

Maciek Gębala Analiza zstępująca. Gramatyki typu  $LL(k)$ [illegible][illegible][illegible]