AGH

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

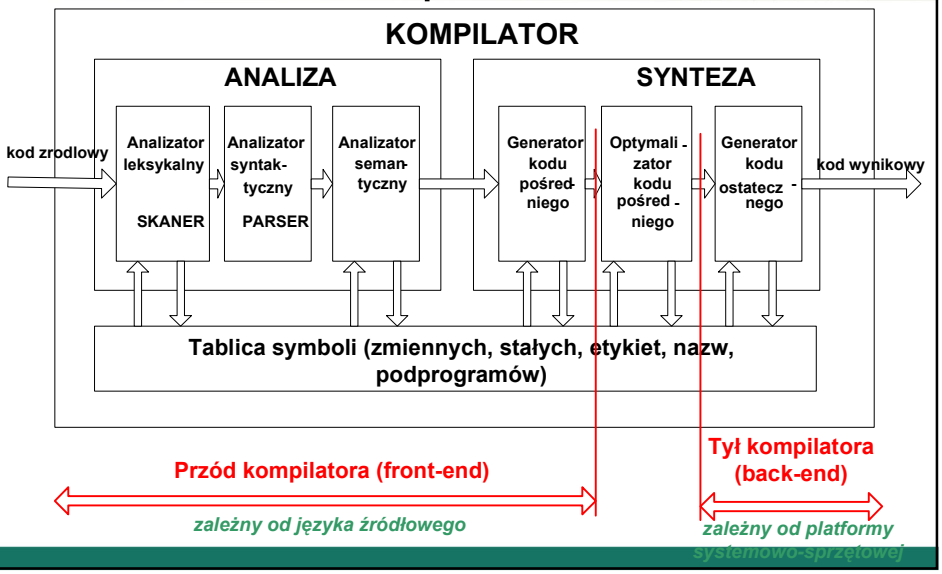
Automat ze stosem

Analiza syntaktyczna

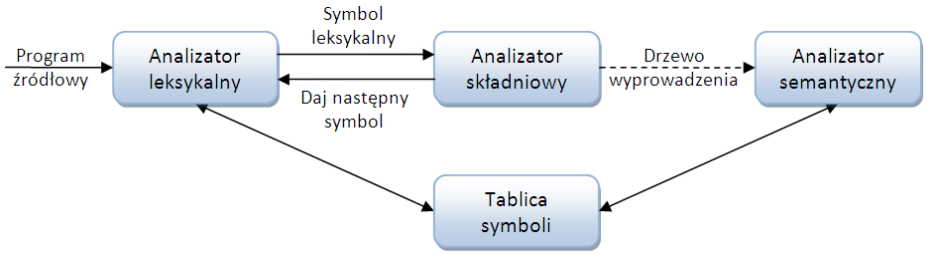
Dr inż. Janusz Majewski
Katedra Informatyki

AGH

Przypomnienie: struktura kompilatora



Współpraca parsera z innymi analizatorami



Analizator składniowy (parser) jest modułem wykonującym analizę syntaktyczną oraz zarządzającym całym przodem kompilatora, a w szczególności całym blokiem analizy. W takt pracy parsera działa zarówno analizator leksykalny (dostarczając parserowi kolejne tokeny), jak i analizator semantyczny (analizując np. poprawność typów) oraz generator kodu pośredniego (tłumacząc zanalizowane fragmenty programu do kodu pośredniego).

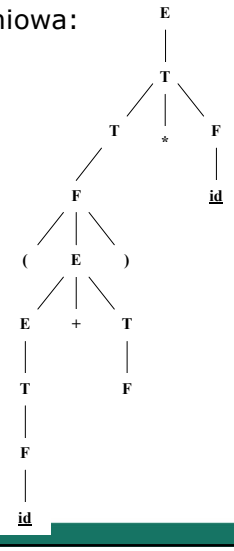
Zadanie analizy syntaktycznej

Analizator syntaktyczny (parser) ma za zadanie dokonać rozbioru gramatycznego tekstu źródłowego analizowanego programu. Analizator leksykalny (skaner) dokonuje zamiany tekstu wejściowego na ciąg tokenów. Parser pracuje na podstawie gramatyki syntaktycznej. Buduje on drzewo rozbioru analizowanego ciągu tokenów, czyli przeprowadza wyprowadzenie łańcucha tokenów w gramatyce składniowej. Od parsera oczekujemy jakiegoś zanotowania tego wywodu. Najczęściej stosowanym sposobem notowania wyprowadzenia jest podanie ciągu numerów produkcji, na podstawie którego można odtworzyć drzewo rozbioru.

Analiza składniowa

Gramatyka składniowa:

- E→E+T
- E→T
- T→T*F
- T→F
- F→(E)
- F→id



Analizowane słowo: (id+id)*id
Do zbudowania drzewa rozbioru syntakt. (podstawa konstrukcji kodu wynikowego) niezbędna jest wiadomość, że ciąg numerów produkcji związany jest z wywodem (tutaj) lewostronnym tworzonym metodą top-down

E
(2) ⇒ T
(3) ⇒ T*F
(4) ⇒ F*F
(5) ⇒ (E)*F
(1) ⇒ (E+T)*F
(2) ⇒ (T+T)*F
(4) ⇒ (F+T)*F
(6) ⇒ (id+T)*F
(4) ⇒ (id+F)*F
(6) ⇒ (id+id)*F
(6) ⇒ (id+id)* id ⇒ acc

ciąg numerów produkcji jest wyjściem z parsera

Gramatyki LL i LR

LL(k)

- przeglądanie wejścia od lewej do prawej
- odtwarzanie wywodu lewostronnego metodą top-down
- liczba symboli, które trzeba podglądać na wejściu, aby jednoznacznie określić numer produkcji, którą należy zastosować

LR(k)

- przeglądanie wejścia od lewej do prawej
- odtwarzanie wywodu prawostronnego metodą bottom-up
- liczba symboli, które trzeba podglądać na wejściu, aby jednoznacznie określić numer produkcji, którą należy zastosować

Na ogół stosowane są gramatyki:

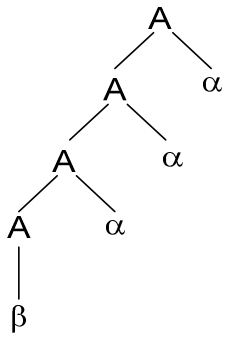
- LL(1)
- LR(1) – oraz jej podklasy: SLR(1) lub LALR(1), dające możliwość znacznego uproszczenia algorytmu parsingu przy niewielkim ograniczeniu „możliwości” gramatyki.

Usuwanie lewostronnej rekurencji

Mówimy, że gramatyka jest lewostronnie rekurencyjna, jeśli ma taki nieterminal A, że istnieje wyprowadzenie $A \Rightarrow^{+} A\alpha$ dla pewnego niepustego łańcucha α.

Jeśli mamy produkcje:

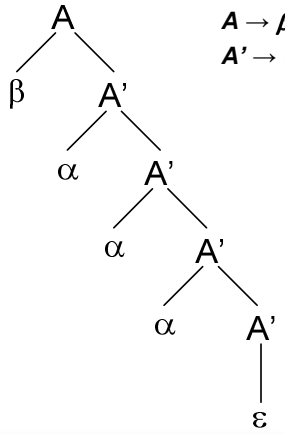
$A \rightarrow A\alpha \mid \beta$



To możemy je zastąpić przez:

$A \rightarrow \beta A'$

$A' \rightarrow \alpha A' \mid \varepsilon$



Przykład

$A \rightarrow A\alpha \mid \beta$

$A \rightarrow \beta A'$

$A' \rightarrow \alpha A' \mid \varepsilon$

1) E → E+T	1) E → TE’
2) E → T	2) E’ → +TE’
	3) E’ → ε
3) T→T*F	4) T → FT’
4) T→F	5) T’ → *FT’
	6) T’ → ε
5) F→(E)	7) F → (E)
6) F→id	8) F → id

Zbiory $FIRST_k$ i $FOLLOW_k$

- Zbiór $FIRST_k(\alpha)$ jest zbiorem wszystkich terminalnych przedrostków długości k (lub mniejszej, jeżeli z α wyprowadza się łańcuch terminalny krótszy niż k) łańcuchów, które mogą być wyprowadzalne z α
- Zbiór $FOLLOW_k(\beta)$ zawiera terminalne łańcuchy o długości k (lub mniejszej—patrz powyżej), które mogą pojawić się w wyprowadzeniach jako następniki β. Uwaga: jeśli $FOLLOW_k(\beta)$ zawiera x, |x|<k, to wówczas zastępujemy x przez x$, gdzie $—prawy ogranicznik słowa (dla wygody).

Przykład (1)

1) E → TE’
2) E’ → +TE’
3) E’ → ε
4) T → FT’
5) T’ → *FT’
6) T’ → ε
7) F → (E)
8) F → id

} $FIRST_1(F)=\{(, id\}$

Przykład (2)

1) $E \rightarrow TE'$
2) $E' \rightarrow +TE'$
3) $E' \rightarrow \varepsilon$
4) $T \rightarrow FT'$ → $FIRST_1(T)=FIRST_1(F)=\{(, \underline{id}\}$
5) $T' \rightarrow *FT'$
6) $T' \rightarrow \varepsilon$
7) $F \rightarrow (E)$
8) $F \rightarrow \underline{id}$

} $FIRST_1(F)=\{(, \underline{id}\}$

Przykład (3)

1) $E \rightarrow TE'$ → $FIRST_1(E)=FIRST_1(T)=\{(, \underline{id}\}$
2) $E' \rightarrow +TE'$
3) $E' \rightarrow \varepsilon$
4) $T \rightarrow FT'$ → $FIRST_1(T)=FIRST_1(F)=\{(, \underline{id}\}$
5) $T' \rightarrow *FT'$
6) $T' \rightarrow \varepsilon$
7) $F \rightarrow (E)$
8) $F \rightarrow \underline{id}$

} $FIRST_1(F)=\{(, \underline{id}\}$

Przykład (4)

1) $E \rightarrow TE'$ → $FIRST_1(E)=FIRST_1(T)=\{(, \underline{id}\}$
2) $E' \rightarrow +TE'$
3) $E' \rightarrow \varepsilon$ } $FIRST_1(E')=\{+, \varepsilon\}$
4) $T \rightarrow FT'$ → $FIRST_1(T)=FIRST_1(F)=\{(, \underline{id}\}$
5) $T' \rightarrow *FT'$
6) $T' \rightarrow \varepsilon$ } $FIRST_1(T')=\{*, \varepsilon\}$
7) $F \rightarrow (E)$
8) $F \rightarrow \underline{id}$ } $FIRST_1(F)=\{(, \underline{id}\}$

Wyznaczanie zbiorów $FOLLOW_1$

Wyznaczenie $FOLLOW_1$ dla wszystkich $A \in V$

Stosować poniższe reguły (1), (2), (3) dopóty nic nowego nie może już zostać dodane do żadnego zbioru $FOLLOW_1$.

1) $FOLLOW_1(S) := FOLLOW_1(S) \cup \{\$\}$
2) if $(A \rightarrow \alpha B \beta) \in P$ then
   $FOLLOW_1(B) := FOLLOW_1(B) \cup (FIRST_1(\beta) - \{\varepsilon\})$
3) if $(A \rightarrow \alpha B) \in P$ or $((A \rightarrow \alpha B \beta) \in P$ and $(\varepsilon \in FIRST_1(\beta)))$ then
   $FOLLOW_1(B) := FOLLOW_1(B) \cup FOLLOW_1(A)$;

Przykład (5)

1) $E \rightarrow TE'$ ⟶ $FOLLOW_1(E) \supseteq \{\$\}$
2) $E' \rightarrow +TE'$
3) $E' \rightarrow \varepsilon$
4) $T \rightarrow FT'$
5) $T' \rightarrow *FT'$
6) $T' \rightarrow \varepsilon$
7) $F \rightarrow (E)$
8) $F \rightarrow \underline{id}$

Przykład (6)

1) $E \rightarrow TE'$ ⟶ $FOLLOW_1(E) \supseteq \{\$\}$
2) $E' \rightarrow +TE'$ ⟶ $FOLLOW_1(T) \supseteq \{+\}$
3) $E' \rightarrow \varepsilon$
4) $T \rightarrow FT'$ ⟶ $FOLLOW_1(F) \supseteq \{*\}$
5) $T' \rightarrow *FT'$ ⟶ $FOLLOW_1(F) \supseteq \{*\}$
6) $T' \rightarrow \varepsilon$
7) $F \rightarrow (E)$ ⟶ $FOLLOW_1(E) \supseteq \{)\}$
8) $F \rightarrow \underline{id}$

Czyli w tej chwili:
$FOLLOW_1(E)=\{\$,)\}$
$FOLLOW_1(T)=\{+\}$
$FOLLOW_1(F)=\{*\}$

Przykład (7)

1) $E \rightarrow TE'$ → $FOLLOW_1(E') \leftarrow FOLLOW_1(E)$
2) $E' \rightarrow +TE'$ → $FOLLOW_1(T) \leftarrow FOLLOW_1(E)$
3) $E' \rightarrow \varepsilon$
4) $T \rightarrow FT'$ → $FOLLOW_1(T') \leftarrow FOLLOW_1(T)$
5) $T' \rightarrow *FT'$ → $FOLLOW_1(F) \leftarrow FOLLOW_1(T)$
6) $T' \rightarrow \varepsilon$
7) $F \rightarrow (E)$
8) $F \rightarrow id$

Poprzednio:
$FOLLOW_1(E)=\{\$,)\}$
$FOLLOW_1(T)=\{+\}$
$FOLLOW_1(F)=\{*\}$

Ostatecznie:
$FOLLOW_1(E)=\{\$,)\}$
$FOLLOW1(E')=\{\$,)\}$
$FOLLOW_1(T)=\{+, \$,)\}$
$FOLLOW_1(T')=\{+, \$,)\}$
$FOLLOW_1(F)=\{*, +, \$,)\}$

Lewostronna faktoryzacja

Jeżeli produkcje gramatyki mające ten sam symbol po lewej stronie posiadają prawe strony rozpoczynające się od tego samego przedrostka, to można dla nich wykonać przekształcenie zwane lewostronną faktoryzacją. Polega ona na zamianie produkcji postaci:

$$\mathbf{A \rightarrow \alpha\beta_1 \mid \ldots \mid \alpha\beta_k}$$

na produkcje:

$$\mathbf{A \rightarrow \alpha A'}$$

$$\mathbf{A' \rightarrow \beta_1 \mid \ldots \mid \beta_k}$$