
AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

# Parsery SLR(1)

## Teoria kompilacji

**Dr inż. Janusz Majewski**
**Katedra Informatyki**

# Podklasy gramatyk LR(1)

$G_{LR(0)} \subset G_{LR(1)}$

Wiele gramatyk nie spełnia wymagań LR(0), ale spełnia wymagania LR(1) z nadmiarem. Z drugiej strony, konstrukcja parsera LR(1) jest procesem dość złożonym, a sama tablica LR(1) zajmuje stosunkowo duży obszar pamięci. Stąd pojawiły się gramatyki pośrednie SLR(1) oraz LALR(1). Obejmują one dostatecznie szeroką podklasę języków LR(1), zaś rozmiary tablic parserów SLR(1) i LALR(1) są znacznie mniejsze niż w przypadku tablic kanonicznego LR(1).

$G_{LR(0)} \subset G_{SLR(1)} \subset G_{LALR(1)} \subset G_{LR(1)}$

$G_{LR(0)} \neq G_{SLR(1)} \neq G_{LALR(1)} \neq G_{LR(1)}$

# Nazwa gramatyki: SLR(k)

**S L R (k)**

- **S** — Proste (Simple)
- **L** — Przeglądanie wejścia od lewej strony do prawej
- **R** — Odtwarzanie wywodu prawostronnego
- **(k)** — Wystarcza znajomość "k" następnych symboli łańcucha wejściowego i historii dotychczasowych redukcji, aby wyznaczyć jednoznacznie osnowę i dokonać jej redukcji

# Przedrostki żywotne
# Sytuacje dopuszczalne

Żywotny przedrostek (viable prefix)

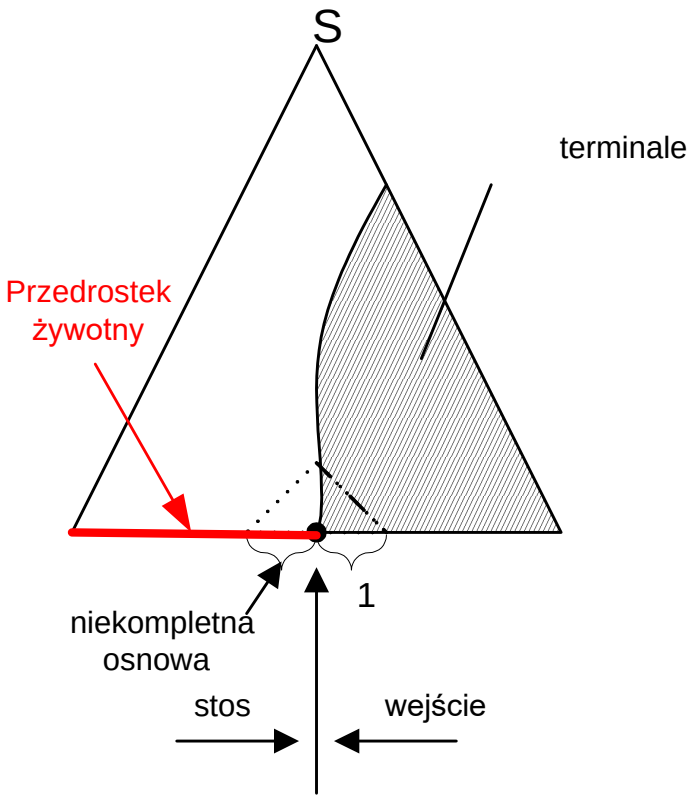
$\gamma$ - żywotny (aktywny) prefiks gramatyki G
$\Leftrightarrow \gamma$ - prefiks łańcucha $\alpha\beta$

$$S \overset{*}{\underset{R}{\Rightarrow}} \alpha A w \underset{R}{\Rightarrow} \alpha\beta w$$

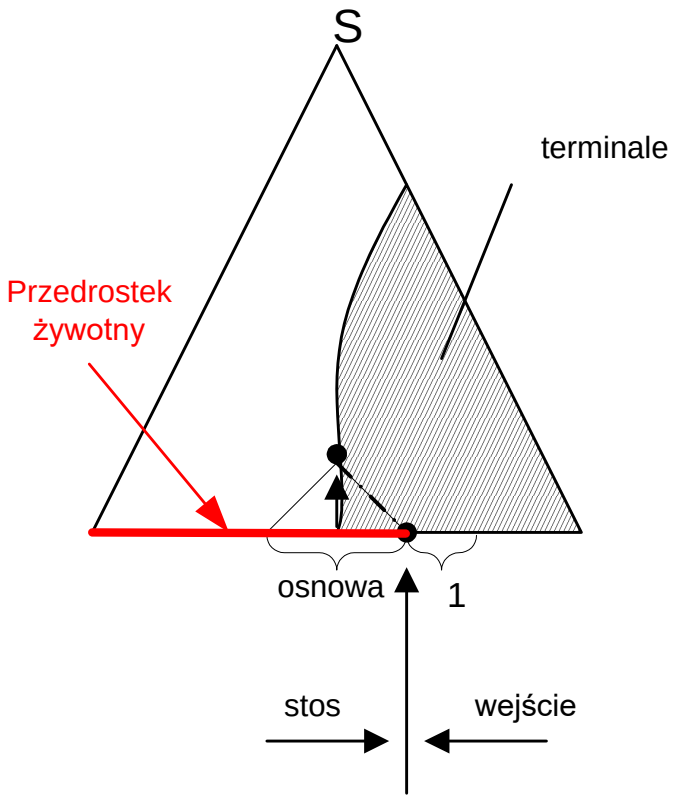
gdzie: $\alpha, \beta, \gamma \in (\Sigma \cup N)^*$ $w \in \Sigma^*$ $A \in V$

Żywotny przedrostek jest to łańcuch będący przedrostkiem pewnej prawostronnie wyprowadzalnej formy zdaniowej, nie wychodzący poza prawy koniec jej osnowy.

# Przedrostki żywotne
# Sytuacje dopuszczalne




shift




redukcja

# Przedrostki żywotne
# Sytuacje dopuszczalne

LR(0) – sytuacja

$[A \to \beta_1 \bullet \beta_2]$ - jest LR(0)-sytuacją, gdy $(A \to \beta_1\beta_2) \in P$

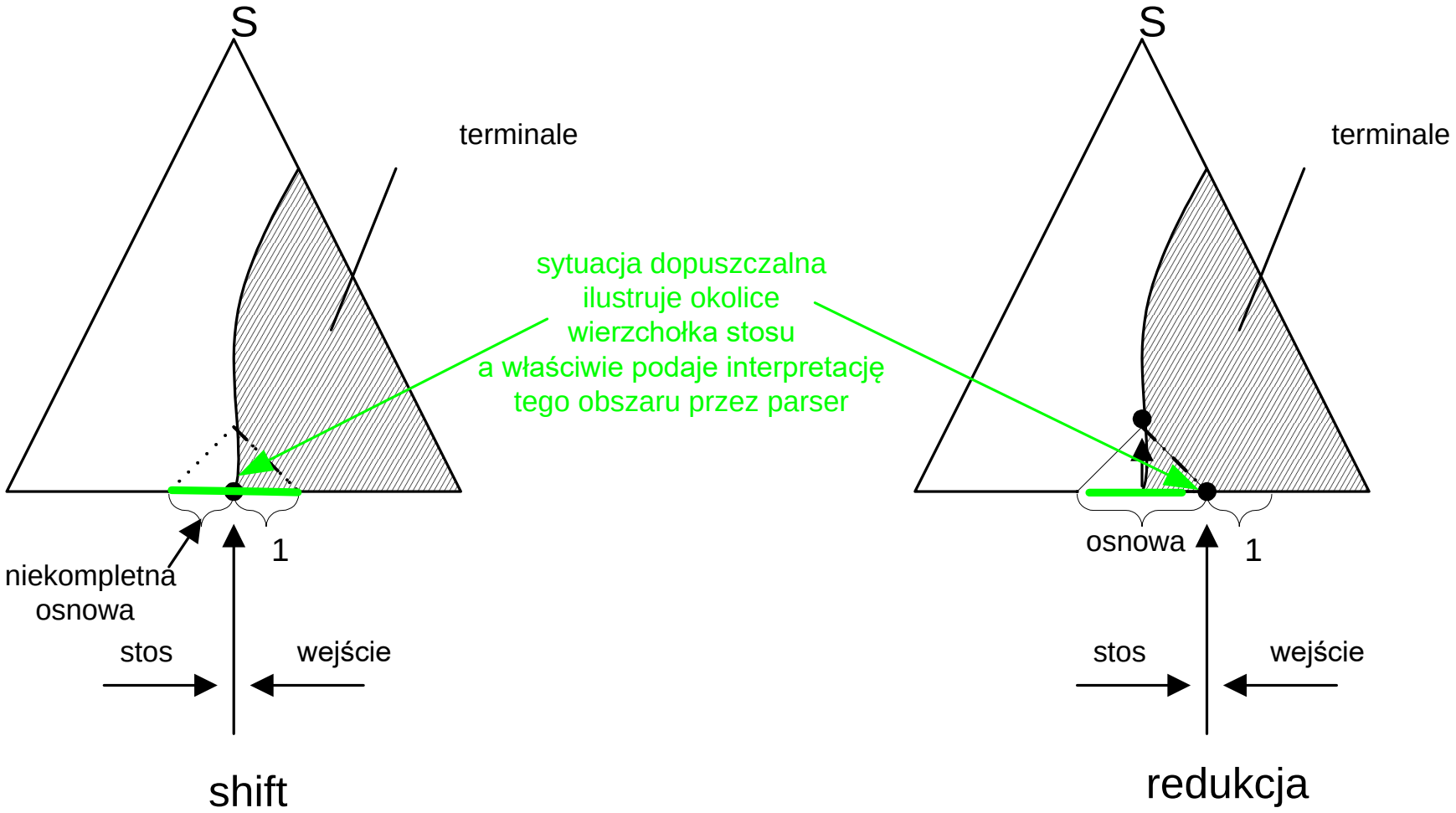
LR(0) - sytuacja dopuszczalna

$[A \to \beta_1 \bullet \beta_2]$ - LR(0)-sytuacja jest sytuacją dopuszczalną dla żywotnego prefiksu $\alpha\beta_1$ wtedy i tylko wtedy gdy

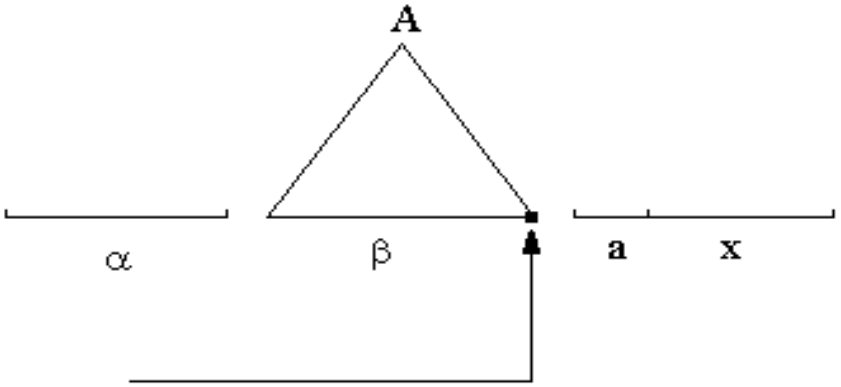
$\exists$ wywód:

$$S \underset{R}{\overset{*}{\Rightarrow}} \alpha A w \underset{R}{\Rightarrow} \alpha\beta_1\beta_2 w$$

# Przedrostki żywotne
# Sytuacje dopuszczalne



# Przedrostki żywotne
# Sytuacje dopuszczalne

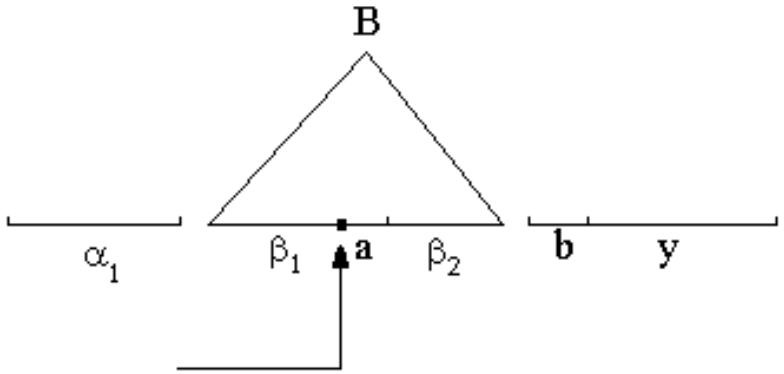



Sytuacja $[A \rightarrow \beta\bullet]$
dopuszczalna dla żywotnego przedrostka $\alpha\beta$

Decyzja: redukcja wg produkcji $A \rightarrow \beta$

**Parser SLR wykona tę redukcję, gdy $a \in FOLLOW_1(A)$**

Efekt: nowa konfiguracja z żywotnym przedrostkiem $\alpha A$




Sytuacja $[B \rightarrow \beta_1 \bullet a \beta_2]$
dopuszczalna dla żywotnego przedrostka $\alpha_1\beta_1$

Decyzja: przesunięcie (shift) terminala *a* z wejścia na stos

Efekt: nowa konfiguracja opisana sytuacją: $[B \rightarrow \beta_1 a \bullet \beta_2]$
dopuszczalną dla żywotnego przedrostka $\alpha_1\beta_1 a$

# Przykład – gramatyka jednoznaczna

(0) E'→E
(1) E→E+T
(2) E→T
(3) T→T*F
(4) T→F
(5) F→(E)
(6) F→id

E•
E+T•
E+T*F•
E+T*id•
E+T*•id
**E+T•*id**
E+F•*id
E+id•*id
E+•id*id
E•+id*id
T•+id*id
F•+id*id
id•+id*id
•id+id*id

[E→E+T•] lub [T→T•*F] dla prefiksu E+T

decyzja: *shift*, bo: * ∉ **FOLLOW$_1$(E)**

# Przykład – gramatyka jednoznaczna

| stan | f | | | | | | g | | |
|---|---|---|---|---|---|---|---|---|---|
| | $ | + | * | ( | ) | id | E | T | F |
| $T_0$ | | | | shift-4 | | shift-5 | $T_1$ | $T_2$ | $T_3$ |
| $T_1$ | acc | shift-6 | | | | | | | |
| $T_2$ | red-2 | red-2 | shift-7 | | red-2 | | | | |
| $T_3$ | red-4 | red-4 | red-4 | | red-4 | | | | |
| $T_4$ | | | | shift-4 | | shift-5 | $T_8$ | $T_2$ | $T_3$ |
| $T_5$ | red-6 | red-6 | red-6 | | red-6 | | | | |
| $T_6$ | | | | shift-4 | | shift-5 | | $T_9$ | $T_3$ |
| $T_7$ | | | | shift-4 | | shift-5 | | | $T_{10}$ |
| $T_8$ | | shift-6 | | | shift-11 | | | | |
| $T_9$ | red-1 | red-1 | **shift-7** | | red-1 | | | | |
| $T_{10}$ | red-3 | red-3 | red-3 | | red-3 | | | | |
| $T_{11}$ | red-5 | red-5 | red-5 | | red-5 | | | | |

# Kanoniczny system zbiorów LR(0)-sytuacji dopuszczalnych

$J_0$ – Kanoniczny system zbiorów LR(0)-sytuacji dopuszczalnych

$J_0$ jest zbiorem wszystkich zbiorów $I(\gamma)$ LR(0)-sytuacji dopuszczalnych;

gdzie: $\gamma$ - żywotny prefiks w gramatyce $G'$.

Wyznaczanie kanonicznego systemu zbiorów LR(0)-sytuacji jest podobne jak w przypadku LR(1). Podajemy tylko "funkcje" stanowiące podstawę odpowiednich algorytmów.

# Algorytm domykania zbioru sytuacji dopuszczalnych

We: Zbiór *I* sytuacji dopuszczalnych dla pewnego żywotnego prefiksu (w gramatyce uzupełnionej *G'*)

Wy: Zbiór *I* będący domknięciem wejściowego zbioru sytuacji dopuszczalnych

Metodę ilustruje funkcja CLOSURE(*I*);

```
function CLOSURE(I);
begin
    repeat
        for każda sytuacja [A → α•Bβ] ∈ I do
            for każda produkcja (B → η) ∈ P' do
                I := I ∪ {[B → •η]};
    until nic nowego nie dodano do I;
    return (I);
end;
```

# Wyznaczanie funkcji GOTO

We: $I$ – zbiór wszystkich sytuacji dopuszczalnych dla prefiksu aktywnego $\gamma$ , $X \in (V \cup \Sigma)$

Wy: $J$ – zbiór wszystkich sytuacji dopuszczalnych dla prefiksu aktywnego $\gamma X$

Metodę ilustruje funkcja GOTO($I$ , $X$)

```
function GOTO(I , X);
begin
        J := ∅;
        for każda_sytuacja [A → α•X β] ∈ I do
                J := J ∪ { [A → αX• β] };
        return CLOSURE(J);
end;
```

# Konstrukcja kanonicznego systemu zbiorów LR(0)-sytuacji dopuszczalnych

We: $G'$ – gramatyka uzupełniona $<V', \Sigma, P', S'>$
dla gramatyki $G = < V, \Sigma, P, S > \in G_{BK}$

Wy: $J_0$ – kanoniczny system zbiorów
LR(0)-sytuacji dopuszczalnych dla $G$.

Metodę ilustruje funkcja ITEMS ($G'$);

```
function ITEMS (G');
begin
        J0 := {CLOSURE ({[S' → •S]})};
        repeat
           for każdy zbiór I ∈ J0 do
              for każdy X ∈ (V ∪ Σ) do
                 if GOTO(I , X) ≠ ∅ then
                  J0 := J0 ∪ {GOTO(I ,X)};
        until nic nowego nie dodano do Jo;
        return J0;
end;
```

# Gramatyki SLR(1)

$G = \langle V, \Sigma, P, S \rangle \in G_{BK}$

$J_0$ – kanoniczny system zbiorów LR(0)-sytuacji dopuszczalnych dla $G$

$G$ jest gramatyką SLR(1)$\Leftrightarrow$

$$(\forall I \in J_0)\left( \forall \begin{matrix} [A \to \alpha g \beta] \in I \\ [B \to \gamma g \delta] \in I \end{matrix} \right\}$$ <u>różne</u> LR(0)-sytuacje**!**

Zachodzi dokładnie jeden z poniższych warunków:

*(1)* $\beta \neq \varepsilon \wedge \delta \neq \varepsilon$

*(2)* $\beta \neq \varepsilon \wedge \delta = \varepsilon \wedge \beta = a\,\beta_1 \wedge a \in \Sigma \wedge \mathrm{FOLLOW}_1(B) \cap \{a\} = \varnothing$

*(3)* $\beta = \varepsilon \wedge \delta \neq \varepsilon \wedge \delta = a\,\delta_1 \wedge a \in \Sigma \wedge \mathrm{FOLLOW}_1(A) \cap \{a\} = \varnothing$

*(4)* $\beta = \varepsilon \wedge \delta = \varepsilon \wedge \mathrm{FOLLOW}_1(A) \cap \mathrm{FOLLOW}_1(B) = \varnothing$

# Tworzenie tablicy parsera SLR(1)

Algorytm działania parsera SLR(1) jest identyczny jak LR(1). Inny jest sposób tworzenia tablicy parsera.

Konstrukcja tablicy parsera SLR(1)

We: *G'* - gramatyka uzupełniona dla gramatyki bezkontekstowej *G*.

Wy: Tablica parsera SLR(1) - funkcje *f* i *g*

Metoda:

(1) Konstruujemy $J_0$ - kanoniczny system zbiorów LR(0)-sytuacji dopuszczalnych dla *G'*.

(2) Numerujemy produkcje gramatyki *G'*.

# Tworzenie tablicy parsera SLR(1)

(3) for każdy zbiór $I_j \in J_0$ do
begin
utwórz w tablicy parsera stan $T_j \in \mathfrak{I}$ dla analizowanego $I_j \in J_0$;
for każda sytuacja ze zbioru $I_j$ do
begin
(a) if $[A \rightarrow \alpha \bullet a\ \beta] \in I_j$ and $a \in \Sigma$ and $GOTO(I_j, a) = I_k$
then $f(T_j, a) := shift\text{-}k$
(b) if $[A \rightarrow \alpha \bullet] \in I_j$ *and* $A \neq S'$ and $i$ - numer produkcji $(A \rightarrow \alpha) \in P$
then for każdy $a \in FOLLOW_1(A)$ do $f(T_j, a) := red\text{-}i$;
(c) if $[S' \rightarrow S \bullet] \in I_j$ then $f(T_j, \$) := acc$;
end;
for każdy $A \in V$ do
if $GOTO(I_j, A) \neq \varnothing$ and $GOTO(I_j, A) = I_k$ then $g(T_j, A) := T_k$;
end;

# Tworzenie tablicy parsera SLR(1)

if w jakiejkolwiek pozycji tablicy parsera SLR(1) jest więcej niż jeden zapis then STOP; /* gramatyka nie jest SLR(1) */

(4) for każdy $T_j \in \mathfrak{T}$ do
begin
for każdy $a \in \Sigma \cup \{\$\}$ do
if $f(T_j, a)$ nieokreślone then $f(T_j, a) := err$;
for każdy $A \in V$ do
if $g(T_j, A)$ nieokreślone then $g(T_j, A) := err$;
end;

(5) Stanem początkowym parsera jest ten stan, który odpowiada zbiorowi $I \in J_0$, dla którego $[S' \rightarrow \bullet S] \in I$;

# Przykład

$A_0 = \{[E' \rightarrow \bullet E],$
$[E \rightarrow \bullet E+T],$
$[E \rightarrow \bullet T],$
$[T \rightarrow \bullet T*F],$
$[T \rightarrow \bullet F],$
$[F \rightarrow \bullet (E)],$
$[F \rightarrow \bullet \underline{id}]\}$

$A_1 = GOTO(A_0, E) =$
$\{[E' \rightarrow E\bullet],$
$[E \rightarrow E\bullet+T]\}$

$A_2 = GOTO(A_0, T) =$
$\{[E \rightarrow T\bullet],$
$[T \rightarrow T\bullet*F]\}$

$A_3 = GOTO(A_0, F) =$
$\{[T \rightarrow F\bullet ]\}$

$A_4 = GOTO(A_0, ( ) =$
$\{[F \rightarrow (\bullet E)],$
$[E \rightarrow \bullet E+T]$
$[E \rightarrow \bullet T]$
$[T \rightarrow \bullet T*F]$
$[T \rightarrow \bullet F]$
$[F \rightarrow \bullet (E)]$
$[F \rightarrow \bullet \underline{id}]\}$

$A_5 = GOTO(A_0, \underline{id}) =$
$\{[F \rightarrow id\bullet]\}$

(0) E'→E
(1) E→E+T
(2) E→T
(3) T→T*F
(4) T→F
(5) F→(E)
(6) F→id

# Przykład c.d.

$A_6 = GOTO(A_1, +) =$
$\{[E \rightarrow E+\bullet T],$
$[T \rightarrow \bullet T*F],$
$[T \rightarrow \bullet F],$
$[F \rightarrow \bullet(E)],$
$[F \rightarrow \bullet \underline{id}]\}$

$A_7 = GOTO(A_2, *) =$
$\{[T \rightarrow T*.F],$
$[F \rightarrow .(E)],$
$[F \rightarrow .\underline{id}]\}$

$A_8 = GOTO(A_4, E) =$
$\{[F \rightarrow (E.)],$
$[E \rightarrow E.+T]\}$

$A_2 = GOTO(A_4, T)$
$A_3 = GOTO(A_4, F)$
$A_4 = GOTO(A_4, ( )$
$A_5 = GOTO(A_4, \underline{id})$

$A_9 = GOTO(A_6, T) =$
$\{[E \rightarrow E+T.],$
$[T \rightarrow T.*F]\}$

(0) E'→E
(1) E→E+T
(2) E→T
(3) T→T*F
(4) T→F
(5) F→(E)
(6) F→id

# Przykład c.d.

$A_3 = GOTO(A_6, F)$
$A_4 = GOTO(A_6, ( )$
$A_5 = GOTO(A_6, \underline{id})$

$\boldsymbol{A_{10}} = GOTO(A_7,F) =$
$\boldsymbol{\{[T \rightarrow T*F\bullet]\}}$

$A_4 = GOTO(A_7, ( )$
$A_5 = GOTO(A_7, \underline{id})$

$\boldsymbol{A_{11}} = GOTO(A_8, ) ) =$
$\boldsymbol{\{[F \rightarrow (E)\bullet]\}}$

$A_6 = GOTO(A_8, +)$
$A_7 = GOTO(A_9, *)$

(0) E'→E
(1) E→E+T
(2) E→T
(3) T→T*F
(4) T→F
(5) F→(E)
(6) F→id

# Przykład – gramatyka jednoznaczna

$FIRST_1(E) = \{ (, \underline{id} \}$
$FIRST_1(T) = \{ (, \underline{id} \}$
$FIRST_1(F) = \{ (, \underline{id} \}$
$FOLLOW_1(E') = \{ \$ \}$
$FOLLOW_1(E) = \{ \$, +, ) \}$
$FOLLOW_1(T) = \{ \$, +, *, ) \}$
$FOLLOW_1(F) = \{ \$, +, *, ) \}$

$A_0 = \{[E' \rightarrow \bullet E],$
$[E \rightarrow \bullet E+T],$
$[E \rightarrow \bullet T],$
$[T \rightarrow \bullet T*F],$
$[T \rightarrow \bullet F],$
$[F \rightarrow \bullet(E)],$
$[F \rightarrow \bullet \underline{id}]\}$

$A_1 = GOTO(A_0, E) =$
$\{[E' \rightarrow E\bullet],$
$[E \rightarrow E\bullet+T]\}$

$A_2 = GOTO(A_0, T) =$
$\{[E \rightarrow T\bullet],$
$[T \rightarrow T\bullet*F]\}$

.....

(0) E'→E
(1) E→E+T
(2) E→T
(3) T→T*F
(4) T→F
(5) F→(E)
(6) F→id

| stan | f | | | | | | g | | |
|---|---|---|---|---|---|---|---|---|---|
| | $ | + | * | ( | ) | id | E | T | F |
| $T_0$ | | | | shift-4 | | shift-5 | $T_1$ | $T_2$ | $T_3$ |
| $T_1$ | acc | shift-6 | | | | | | | |
| $T_2$ | red-2 | red-2 | shift-7 | | red-2 | | | | |

.....

# Przykład – gramatyka jednoznaczna

| stan | f | | | | | | g | | |
|---|---|---|---|---|---|---|---|---|---|
| | $ | + | * | ( | ) | id | E | T | F |
| $T_0$ | | | | shift-4 | | shift-5 | $T_1$ | $T_2$ | $T_3$ |
| $T_1$ | acc | shift-6 | | | | | | | |
| $T_2$ | red-2 | red-2 | shift-7 | | red-2 | | | | |
| $T_3$ | red-4 | red-4 | red-4 | | red-4 | | | | |
| $T_4$ | | | | shift-4 | | shift-5 | $T_8$ | $T_2$ | $T_3$ |
| $T_5$ | red-6 | red-6 | red-6 | | red-6 | | | | |
| $T_6$ | | | | shift-4 | | shift-5 | | $T_9$ | $T_3$ |
| $T_7$ | | | | shift-4 | | shift-5 | | | $T_{10}$ |
| $T_8$ | | shift-6 | | | shift-11 | | | | |
| $T_9$ | red-1 | red-1 | shift-7 | | red-1 | | | | |
| $T_{10}$ | red-3 | red-3 | red-3 | | red-3 | | | | |
| $T_{11}$ | red-5 | red-5 | red-5 | | red-5 | | | | |

(0) E'→E
(1) E→E+T
(2) E→T
(3) T→T*F
(4) T→F
(5) F→(E)
(6) F→id

# Symulacja działania parsera SLR dla gramatyki jednoznacznej

| **Stos** | **Wejście** | **Wyjście** |
|---|---|---|
| $T_0$ | id+id$ | ε |
| $T_0 id T_5$ | +id$ | ε |
| $T_0 F T_3$ | +id$ | 6 |
| $T_0 T T_2$ | +id$ | 64 |
| $T_0 E T_1$ | +id$ | 642 |
| $T_0 E T_1 + T_6$ | id$ | 642 |
| $T_0 E T_1 + T_6 id T_5$ | $ | 642 |
| $T_0 E T_1 + T_6 F T_3$ | $ | 6426 |
| $T_0 E T_1 + T_6 T T_9$ | $ | 64264 |
| $T_0 E T_1$ | $ | 642641 |
| akceptacja | | |

# Przykład gramatyki niejednoznacznej

Rozważana gramatyka:

***(0) E' → E***
***(1) E → E + E***
***(2) E → E * E***
***(3) E → ( E )***
***(4) E → <u>id</u>***

jest uproszczoną, a zarazem niejednoznaczną wersją gramatyki SLR(1) z poprzedniego przykładu:

*(0) E' → E*
*(1) E → E + T*
*(3) T → T * F*
*(5) F → ( E )*
*(6) F → <u>id</u>*

(*) *(2) E → T*
*(4) T → F*

(*) W rozważanej gramatyce nie ma produkcji łańcuchowych, więc gdyby udało się skonstruować dla niej parser SLR(1) to rozbiór syntaktyczny byłby jeszcze szybszy, a także rozmiar tablicy parsera byłby mniejszy.

# Przykład gramatyki niejednoznacznej

Sposób postępowania:

(1) Konstruujemy system zbiorów LR(0)-sytuacji

(2) Próbując zbudować tablicę dla parsera SLR(1) znajdujemy konflikty

(3) Staramy się usunąć konflikty wykorzystując dodatkowe wiadomości i wymagania związane z językiem generowanym przez rozważaną gramatykę niejednoznaczną.

# Przykład gramatyki niejednoznacznej

$A_0$: $E' \to \bullet E$
$E \to \bullet E+E$
$E \to \bullet E*E$
$E \to \bullet(E)$
$E \to \bullet \underline{id}$

$A_1$: $E' \to E\bullet$
$E \to E\bullet+E$
$E \to E\bullet*E$

$A_2$: $E \to (\bullet E)$
$E \to \bullet E+E$
$E \to \bullet E*E$
$E \to \bullet(E)$
$E \to \bullet \underline{id}$

$A_3$: $E \to \underline{id}\bullet$

$A_4$: $E \to E+\bullet E$
$E \to \bullet E+E$
$E \to \bullet E*E$
$E \to \bullet(E)$
$E \to \bullet \underline{id}$

$A_5$: $E \to E*\bullet E$
$E \to \bullet E+E$
$E \to \bullet E*E$
$E \to \bullet(E)$
$E \to \bullet \underline{id}$

$A_6$: $E \to (E\bullet)$
$E \to E\bullet+E$
$E \to E\bullet*E$

$A_7$: $E \to E+E\bullet$
$E \to E\bullet+E$
$E \to E\bullet*E$

$A_8$: $E \to E*E\bullet$
$E \to E\bullet+E$
$E \to E\bullet*E$

$A_9$: $E \to (E)\bullet$

$FOLLOW_1(E) = \{\$, ), +, *\}$

# Przykład gramatyki niejednoznacznej

$A_7$: $E \rightarrow E+E\bullet$

$E \rightarrow E\bullet+E$

$E \rightarrow E\bullet*E$

Konflikty dla $A_7$:

a) ponieważ: $\{ + , * \} \subset FOLLOW_1(E)$ więc:

$f(T_7, + ) = \underline{red}\ 1$

$f(T_7, * ) = \underline{red}\ 1$

b) ponieważ: $\{ + , * \} \subset \Sigma$ więc:

$f(T_7, + ) = \underline{shift}$

$f(T_7, * ) = \underline{shift}$

Konflikty:

$f(T_7, + ) = \underline{red}\ 2$ $\quad f(T_7, + ) = \underline{shift}$

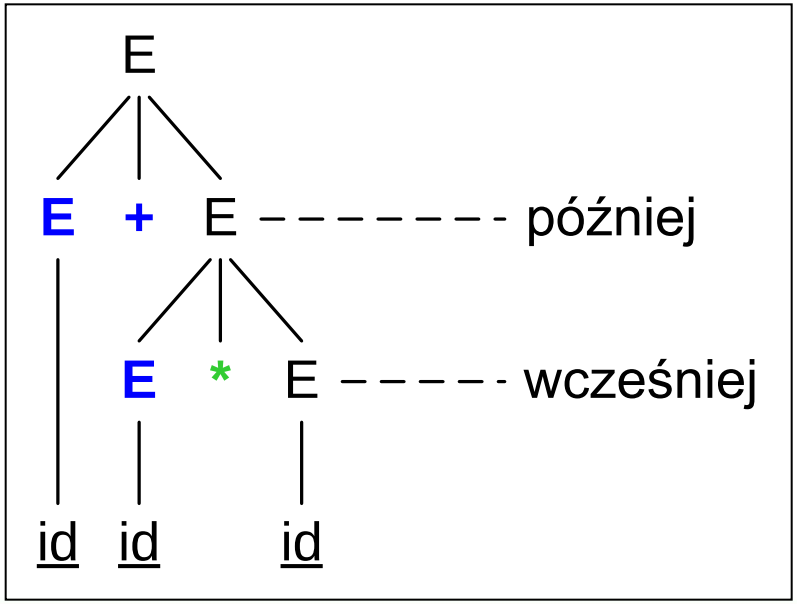
$f(T_7, * ) = \underline{red}\ 2$ $\quad f(T_7, * ) = \underline{shift}$

# Przykład – gramatyka niejednoznaczna, usuwanie konfliktów

(0) E'→E
(1) E→E+E
(2) E→E*E
(3) E→(E)
(4) E→id

E•
E+E•
E+E*E•
E+E*id•
E+E*•id
**E+E**•***id**
E+id•*id
E+•id*id
E•+id*id
id•+id*id
•id+id*id

[E→E+E•]; [E→E•+E] lub [E→E•*E]

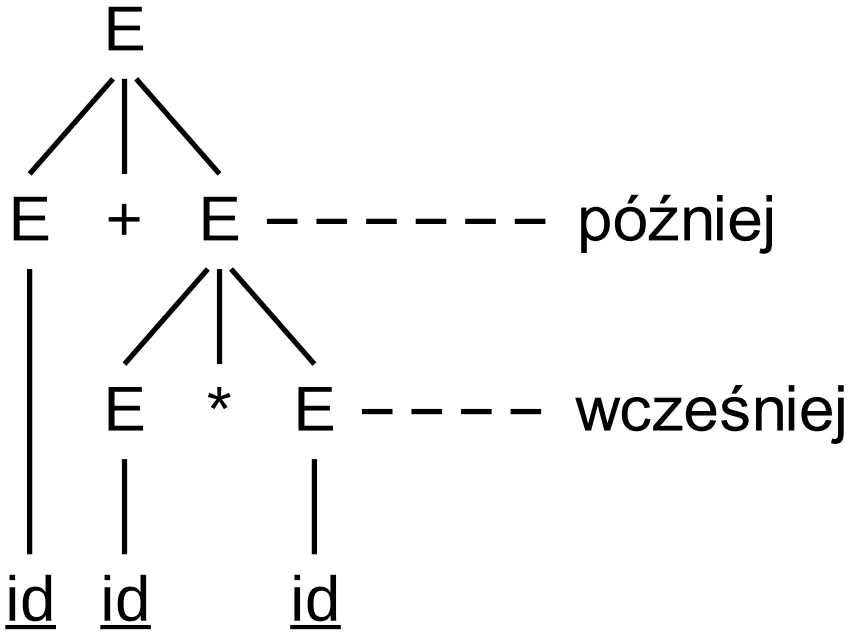


# Przykład gramatyki niejednoznacznej

Rozważamy ciąg wejściowy: *id*+*id***id*
Ponieważ "*" ma wyższy priorytet niż "+", więc "*" powinna być wcześniej redukowana niż "+"

Stos: we:
$T_0ET_1+T_4ET_7$ * *id* $

DECYZJA: $f(T_7, *) =$ *shift* 5



# Przykład gramatyki niejednoznacznej

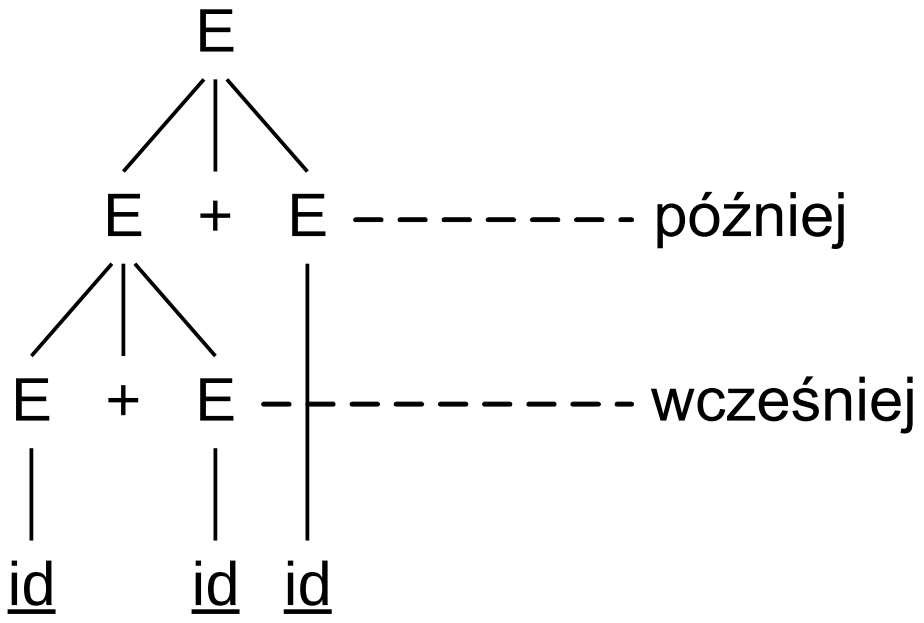
Rozważamy ciąg wejściowy: *id*+*id*+*id*

Ponieważ "+" jest lewostronnie łączny , więc najpierw powinna nastąpić redukcja lewego "+"

Stos: we:

$T_0ET_1+T_4ET_7$ + *id* $

DECYZJA: $f(T_7, +) =$ *red 1*



# Przykład – gramatyka niejednoznaczna, usuwanie konfliktów

| stan | id | + | * | ( | ) | $ | E |
|---|---|---|---|---|---|---|---|
| $T_0$ | shift 3 | | | shift 2 | | | $T_1$ |
| $T_1$ | | shift 4 | shift 5 | | | acc | |
| $T_2$ | shift 3 | | | shift 2 | | | $T_6$ |
| $T_3$ | | red 4 | red 4 | | red 4 | red 4 | |
| $T_4$ | shift 3 | | | shift 2 | | | $T_7$ |
| $T_5$ | shift 3 | | | shift 2 | | | $T_8$ |
| $T_6$ | | shift 4 | shift 5 | | shift 9 | | |
| $T_7$ | | ~~shift 4~~ **red 1** | **shift 5** ~~red 1~~ | | red 1 | red 1 | |
| $T_8$ | | ~~shift 4~~ **red 2** | ~~shift 5~~ **red 2** | | red 2 | red 2 | |
| $T_9$ | | red 3 | red 3 | | red 3 | red 3 | |

# Symulacja działania parsera SLR dla gramatyki niejednoznacznej

| **Stos** | **Wejście** | **Wyjście** |
|---|---|---|
| $T_0$ | id+id$ | ε |
| $T_0 id T_3$ | +id$ | ε |
| $T_0 E T_1$ | +id$ | 4 |
| $T_0 E T_1 + T_4$ | id$ | 4 |
| $T_0 E T_1 + T_4 id T_3$ | $ | 4 |
| $T_0 E T_1 + T_4 E T_7$ | $ | 44 |
| $T_0 E T_1$ | $ | 441 |
| akceptacja | | |

# Przypomnienie: symulacja działania parsera LL dla gramatyki jednoznacznej po usunięciu lewostronnej rekurencji

| **Stos** | **Wejście** | **Wyjście** |
|---|---|---|
| E | id+id$ | ε |
| E’T | id+id$ | 1 |
| E’T’F | id+id$ | 14 |
| E’T’id | id+id$ | 148 |
| E’T’ | +id$ | 148 |
| E’ | +id$ | 1486 |
| E’T+ | +id$ | 14862 |
| E’T | id$ | 14862 |
| E’T’F | id$ | 148624 |
| E’T’id | id$ | 1486248 |
| E’T’ | $ | 1486248 |
| E’ | $ | 14862486 |
| ε | $ | 148624863 |

akceptacja

# Porównanie działania parserów LL i LR

… dla wejścia id+id i odpowiednich gramatyk

| Rodzaj parsera | Liczba kroków | Długość wyjścia |
|---|---|---|
| LL dla gramatyki jednoznacznej po usunięciu lewostronnej rekurencji | 12 | 9 |
| SLR dla gramatyki jednoznacznej | 9 | 6 |
| SLR dla gramatyki niejednoznacznej | 6 | 3 |