



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)

Proste
(Simple)

Przeoglądanie
wejścia od
lewej strony
do prawej

Odtwarzanie
wyvodu
prawostronnego

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)

γ - żywotny (aktywny) prefiks gramatyki G

$\Leftrightarrow \gamma$ - prefiks łańcucha $\alpha\beta$

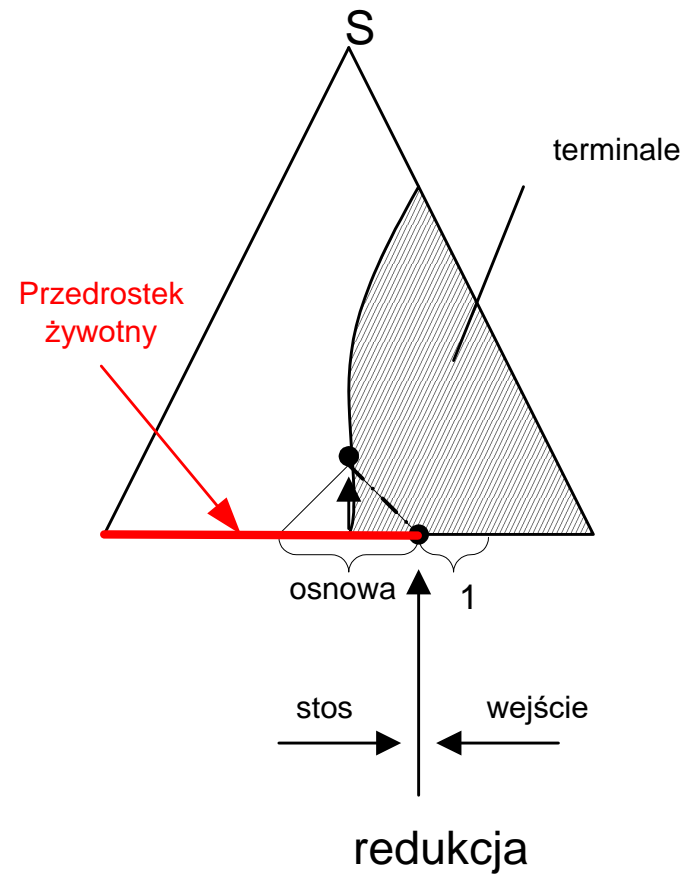
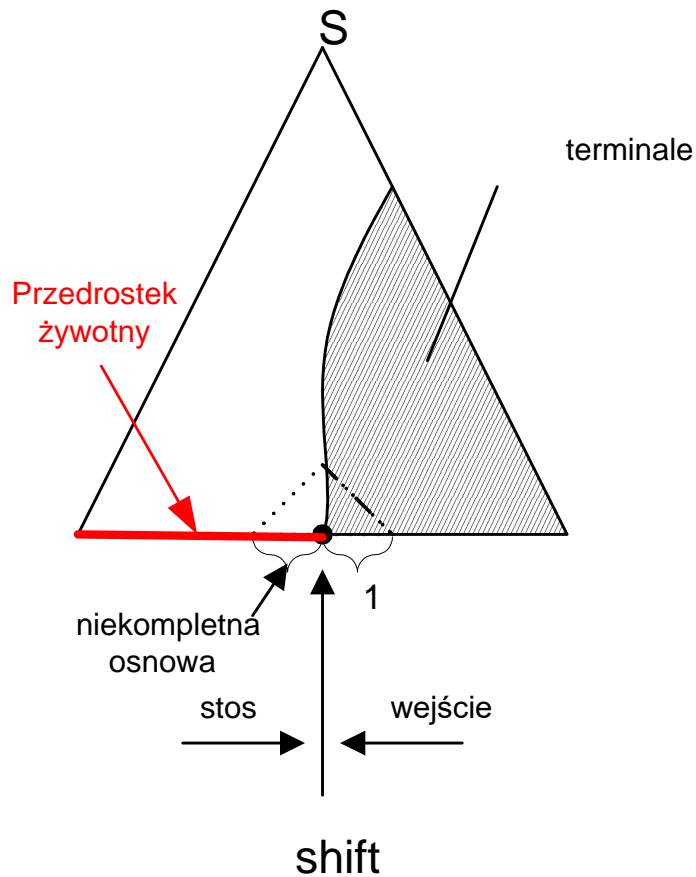
$$S \underset{R}{\overset{*}{\Rightarrow}} \alpha A w \underset{R}{\Rightarrow} \alpha \beta w$$

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

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

Przedrostki żywotne

Sytuacje dopuszczalne



Przedrostki żywotne

Sytuacje dopuszczalne

LR(0) – sytuacja

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

LR(0) - sytuacja dopuszczalna

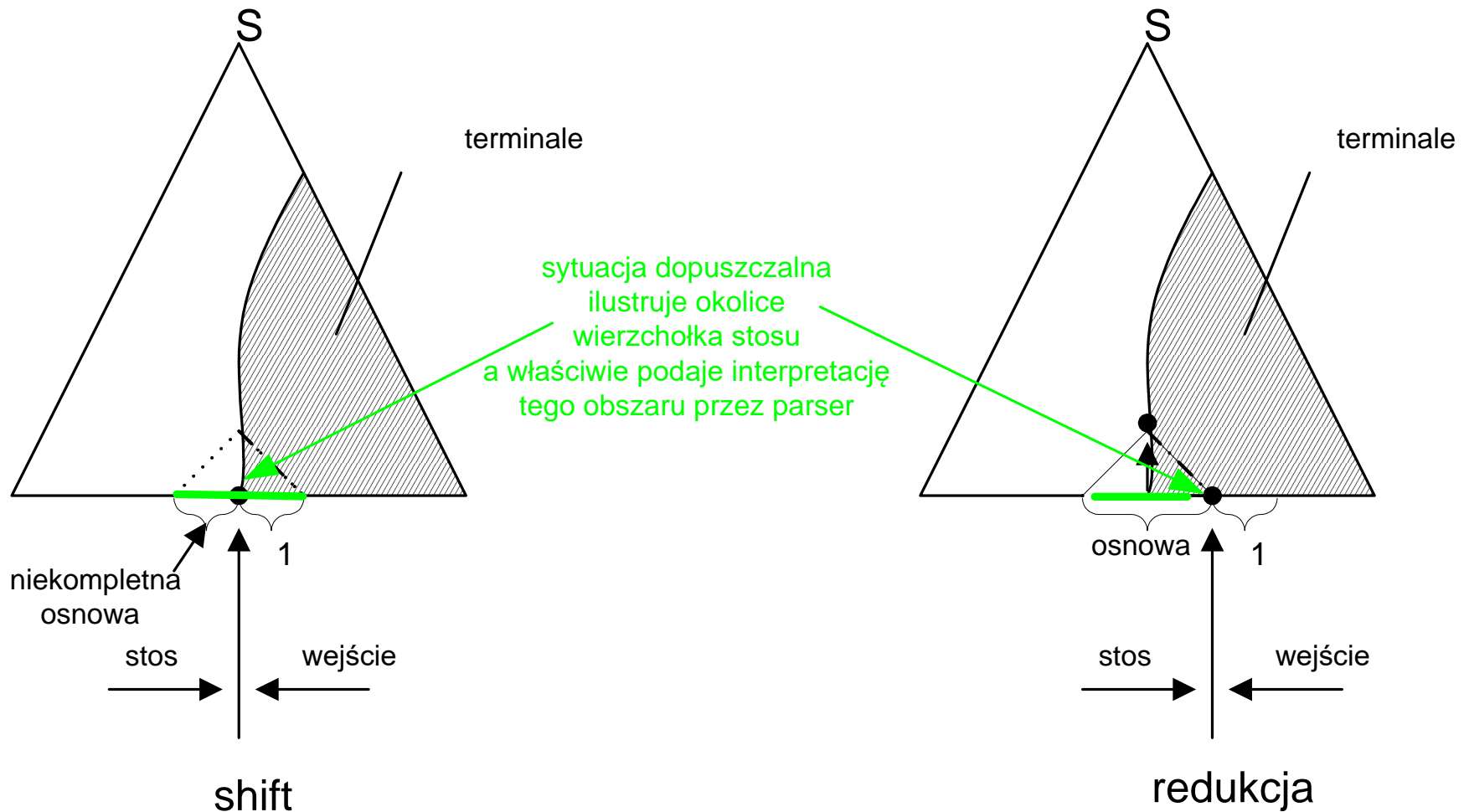
$[A \rightarrow \beta_1 \bullet \beta_2]$ - LR(0)-sytuacja jest sytuacją dopuszczalną dla żywotnego prefiksu $\alpha\beta_1$ wtedy i tylko wtedy gdy

\exists wywód:

$$S \xRightarrow[R]{*} \alpha A w \xRightarrow[R]{} \alpha \beta_1 \beta_2 w$$

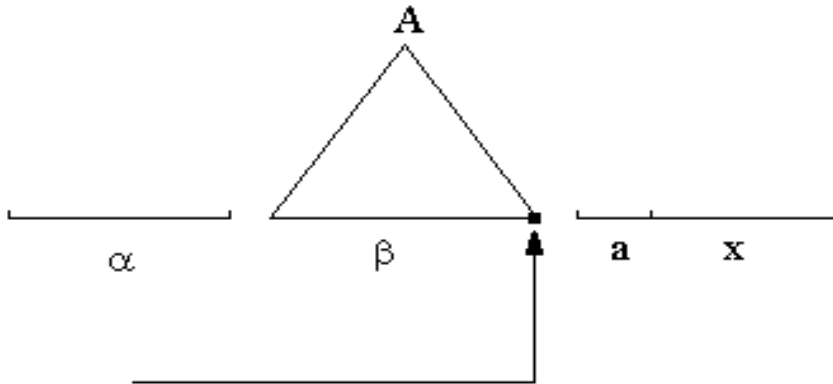
Przedrostki żywotne

Sytuacje dopuszczalne



Przedrostki żywotne

Sytuacje dopuszczalne

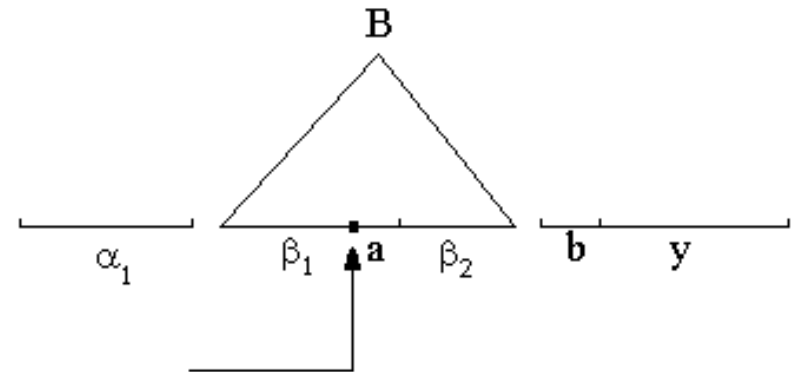


Sytuacja $[A \rightarrow \beta \bullet]$
dopuszczalna dla żywotnego przedrostka $\alpha\beta$

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

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

Efekt: nowa konfiguracja z żywotnym przedrostkiem αA



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) $E' \rightarrow E$

(1) $E \rightarrow E+T$

(2) $E \rightarrow T$

(3) $T \rightarrow T^*F$

(4) $T \rightarrow F$

(5) $F \rightarrow (E)$

(6) $F \rightarrow id$

$E \bullet$

$E+T \bullet$

$E+T^*F \bullet$

$E+T^*id \bullet$

$E+T^* \bullet id$

$E+T \bullet$ *id

$E+F \bullet ^*id$

$E+id \bullet ^*id$

$E+ \bullet id^*id$

$E \bullet +id^*id$

$T \bullet +id^*id$

$F \bullet +id^*id$

$id \bullet +id^*id$

$\bullet id+id^*id$

$[E \rightarrow E+T \bullet]$ lub $[T \rightarrow T \bullet ^*F]$ dla prefiksu $E+T$

decyzja: shift, bo: $*$ $\notin FOLLOW_1(E)$

Przykład – gramatyka jednoznaczna

stan	f						g		
	\$	+	*	()	<u>id</u>	E	T	F
T ₀				<u>shift-4</u>		<u>shift-5</u>	T ₁	T ₂	T ₃
T ₁	<u>acc</u>	<u>shift-6</u>							
T ₂	<u>red-2</u>	<u>red-2</u>	<u>shift-7</u>		<u>red-2</u>				
T ₃	<u>red-4</u>	<u>red-4</u>	<u>red-4</u>		<u>red-4</u>				
T ₄				<u>shift-4</u>		<u>shift-5</u>	T ₈	T ₂	T ₃
T ₅	<u>red-6</u>	<u>red-6</u>	<u>red-6</u>		<u>red-6</u>				
T ₆				<u>shift-4</u>		<u>shift-5</u>		T ₉	T ₃
T ₇				<u>shift-4</u>		<u>shift-5</u>			T ₁₀
T ₈		<u>shift-6</u>			<u>shift-11</u>				
T ₉	<u>red-1</u>	<u>red-1</u>	<u>shift-7</u>		<u>red-1</u>				
T ₁₀	<u>red-3</u>	<u>red-3</u>	<u>red-3</u>		<u>red-3</u>				
T ₁₁	<u>red-5</u>	<u>red-5</u>	<u>red-5</u>		<u>red-5</u>				



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: γ - 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 $[A \rightarrow \alpha \bullet B \beta] \in I$ do

for każda produkcja $(B \rightarrow \eta) \in P'$ do

$I := I \cup \{[B \rightarrow \bullet \eta]\}$;

until nic nowego nie dodano do I ;

return (I) ;

end;

Wyznaczanie funkcji GOTO

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

Wy: J – zbiór wszystkich sytuacji dopuszczalnych dla prefiksu aktywnego γX

Metodę ilustruje funkcja $GOTO(I, X)$

function $GOTO(I, X)$;

begin

$J := \emptyset$;

for każda _sytuacja $[A \rightarrow \alpha \bullet X \beta] \in I$ do

$J := J \cup \{ [A \rightarrow \alpha X \bullet \beta] \}$;

return $CLOSURE(J)$;

end;

Konstrukcja kanonicznego systemu zbiorów LR(0)-sytuacji dopuszczalnych

We: G' – gramatyka uzupełniona $\langle V', \Sigma, P', S' \rangle$
dla gramatyki $G = \langle V, \Sigma, P, S \rangle \in \mathbf{G}_{BK}$

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

Metodę ilustruje funkcja ITEMS (G');

function ITEMS (G');

begin

$J_0 := \{\text{CLOSURE}(\{[S' \rightarrow \bullet S]\})\};$

repeat

for każdy zbiór $I \in J_0$ do

for każdy $X \in (V \cup \Sigma)$ do

if GOTO(I, X) $\neq \emptyset$ then

$J_0 := J_0 \cup \{\text{GOTO}(I, X)\};$

until nic nowego nie dodano do J_0 ;

return J_0 ;

end;

Gramatyki SLR(1)

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

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

G jest gramatyką SLR(1) \Leftrightarrow

$$(\forall I \in J_0) \left(\begin{array}{l} \forall [A \rightarrow \alpha g \beta] \in I \\ [B \rightarrow \gamma g \delta] \in I \end{array} \right) \left. \vphantom{(\forall I \in J_0)} \right\} \underline{\text{różne}} \text{ LR(0)-sytuacje!}$$

Zachodzi dokładnie jeden z poniższych warunków:

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

$$(2) \beta \neq \varepsilon \wedge \delta = \varepsilon \wedge \beta = a \beta_1 \wedge a \in \Sigma \wedge \text{FOLLOW}_1(B) \cap \{a\} = \emptyset$$

$$(3) \beta = \varepsilon \wedge \delta \neq \varepsilon \wedge \delta = a \delta_1 \wedge a \in \Sigma \wedge \text{FOLLOW}_1(A) \cap \{a\} = \emptyset$$

$$(4) \beta = \varepsilon \wedge \delta = \varepsilon \wedge \text{FOLLOW}_1(A) \cap \text{FOLLOW}_1(B) = \emptyset$$



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 $T_j \in \mathcal{T}$ 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 $f(T_j, a) := \underline{shift-k}$

(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 $f(T_j, a) := \underline{red-i}$;

(c) if $[S' \rightarrow S \bullet] \in I_j$ then $f(T_j, \$) := \underline{acc}$;

end;

for każdy $A \in V$ do

if $GOTO(I_j, A) \neq \emptyset$ and $GOTO(I_j, A) = I_k$ then $g(T_j, A) := T_k$;

end;

Tworzenie tablicy parsera SLR(1)

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

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

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

Przykład

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

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

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

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

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

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

- | | |
|-----|----------------------|
| (0) | $E' \rightarrow E$ |
| (1) | $E \rightarrow E+T$ |
| (2) | $E \rightarrow T$ |
| (3) | $T \rightarrow T^*F$ |
| (4) | $T \rightarrow F$ |
| (5) | $F \rightarrow (E)$ |
| (6) | $F \rightarrow id$ |

Przykład c.d.

$$A_6 = GOTO(A_1, +) =$$

$$\{[E \rightarrow E+\bullet T],$$

$$[T \rightarrow \bullet T * F],$$

$$[T \rightarrow \bullet F],$$

$$[F \rightarrow \bullet (E)],$$

$$[F \rightarrow \bullet \underline{id}]\}$$

$$A_7 = GOTO(A_2, *) =$$

$$\{[T \rightarrow T * \bullet F],$$

$$[F \rightarrow \bullet (E)],$$

$$[F \rightarrow \bullet \underline{id}]\}$$

$$A_8 = GOTO(A_4, E) =$$

$$\{[F \rightarrow (E) \bullet],$$

$$[E \rightarrow E \bullet + T]\}$$

$$A_2 = GOTO(A_4, T)$$

$$A_3 = GOTO(A_4, F)$$

$$A_4 = GOTO(A_4, ()$$

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

$$A_9 = GOTO(A_6, T) =$$

$$\{[E \rightarrow E + T \bullet],$$

$$[T \rightarrow T \bullet * F]\}$$

- | |
|---------------------------|
| (0) $E' \rightarrow E$ |
| (1) $E \rightarrow E + T$ |
| (2) $E \rightarrow T$ |
| (3) $T \rightarrow T * F$ |
| (4) $T \rightarrow F$ |
| (5) $F \rightarrow (E)$ |
| (6) $F \rightarrow id$ |

Przykład c.d.

$$A_3 = GOTO(A_6, F)$$

$$A_4 = GOTO(A_6, ()$$

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

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

$$A_4 = GOTO(A_7, ()$$

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

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

$$A_6 = GOTO(A_8, +)$$

$$A_7 = GOTO(A_9, *)$$

- (0) $E' \rightarrow E$
- (1) $E \rightarrow E+T$
- (2) $E \rightarrow T$
- (3) $T \rightarrow T^*F$
- (4) $T \rightarrow F$
- (5) $F \rightarrow (E)$
- (6) $F \rightarrow id$

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) $E' \rightarrow E$
- (1) $E \rightarrow E+T$
- (2) $E \rightarrow T$
- (3) $T \rightarrow T*F$
- (4) $T \rightarrow F$
- (5) $F \rightarrow (E)$
- (6) $F \rightarrow id$

stan	f						g		
	\$	+	*	()	id	E	T	F
T ₀				<u>shift-4</u>		<u>shift-5</u>	T ₁	T ₂	T ₃
T ₁	<u>acc</u>	<u>shift-6</u>							
T ₂	<u>red-2</u>	<u>red-2</u>	<u>shift-7</u>		<u>red-2</u>				

.....

Przykład – gramatyka jednoznaczna

stan	f						g		
	\$	+	*	()	id	E	T	F
T ₀				<u>shift-4</u>		<u>shift-5</u>	T ₁	T ₂	T ₃
T ₁	<u>acc</u>	<u>shift-6</u>							
T ₂	<u>red-2</u>	<u>red-2</u>	<u>shift-7</u>		<u>red-2</u>				
T ₃	<u>red-4</u>	<u>red-4</u>	<u>red-4</u>		<u>red-4</u>				
T ₄				<u>shift-4</u>		<u>shift-5</u>	T ₈	T ₂	T ₃
T ₅	<u>red-6</u>	<u>red-6</u>	<u>red-6</u>		<u>red-6</u>				
T ₆				<u>shift-4</u>		<u>shift-5</u>		T ₉	T ₃
T ₇				<u>shift-4</u>		<u>shift-5</u>			T ₁₀
T ₈		<u>shift-6</u>			<u>shift-11</u>				
T ₉	<u>red-1</u>	<u>red-1</u>	<u>shift-7</u>		<u>red-1</u>				
T ₁₀	<u>red-3</u>	<u>red-3</u>	<u>red-3</u>		<u>red-3</u>				
T ₁₁	<u>red-5</u>	<u>red-5</u>	<u>red-5</u>		<u>red-5</u>				

- (0) $E' \rightarrow E$
- (1) $E \rightarrow E+T$
- (2) $E \rightarrow T$
- (3) $T \rightarrow T * F$
- (4) $T \rightarrow F$
- (5) $F \rightarrow (E)$
- (6) $F \rightarrow id$

Symulacja działania parsera SLR dla gramatyki jednoznacznej

Stos	Wejście	Wyjście
T_0	<u>id</u> + <u>id</u> \$	ϵ
T_0 <u>id</u> T_5	+ <u>id</u> \$	ϵ
T_0 FT ₃	+ <u>id</u> \$	6
T_0 TT ₂	+ <u>id</u> \$	64
T_0 ET ₁	+ <u>id</u> \$	642
T_0 ET ₁ + T_6	<u>id</u> \$	642
T_0 ET ₁ + T_6 <u>id</u> T_5	\$	642
T_0 ET ₁ + T_6 FT ₃	\$	6426
T_0 ET ₁ + T_6 TT ₉	\$	64264
T_0 ET ₁	\$	642641
akceptacja		

Przykład gramatyki niejednoznacznej

Rozważana gramatyka:

- (0) $E' \rightarrow E$
- (1) $E \rightarrow E + E$
- (2) $E \rightarrow E * E$
- (3) $E \rightarrow (E)$
- (4) $E \rightarrow \underline{id}$

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

$$(0) E' \rightarrow E$$

$$(1) E \rightarrow E + T$$

$$(3) T \rightarrow T * F$$

$$(5) F \rightarrow (E)$$

$$(6) F \rightarrow \underline{id}$$

$$(*) (2) E \rightarrow T$$

$$(4) T \rightarrow F$$

(*) 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

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

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

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

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

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

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

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

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

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

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

$\text{FOLLOW}_1(E) = \{\$,), +, *\}$

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:

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

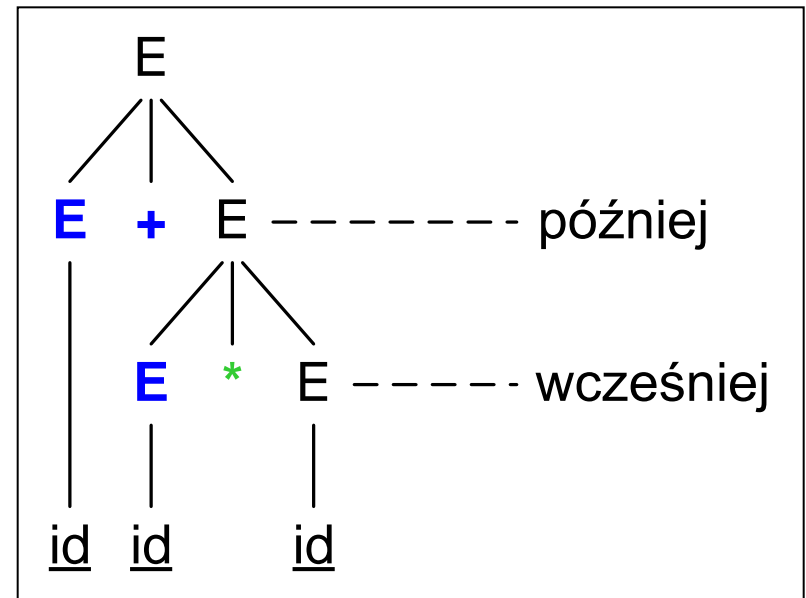
$$f(T_7, *) = \underline{red\ 2} \quad f(T_7, *) = \underline{shift}$$

Przykład – gramatyka niejednoznaczna, usuwanie konfliktów

- (0) $E' \rightarrow E$
- (1) $E \rightarrow E + E$
- (2) $E \rightarrow E * E$
- (3) $E \rightarrow (E)$
- (4) $E \rightarrow id$

$E \bullet$
 $E + E \bullet$
 $E + E * E \bullet$
 $E + E * id \bullet$
 $E + E * \bullet id$
 $E + E \bullet * id$
 $E + id \bullet * id$
 $E + \bullet id * id$
 $E \bullet + id * id$
 $id \bullet + id * id$
 $\bullet id + id * id$

$[E \rightarrow E + E \bullet]$; $[E \rightarrow E \bullet + E]$ lub $[E \rightarrow E \bullet * E]$



Przykład gramatyki niejednoznacznej

Rozważamy ciąg wejściowy: $\underline{id} + \underline{id} * \underline{id}$

Ponieważ “*” ma wyższy priorytet niż “+”, więc “*” powinna być wcześniej zredukowana niż “+”

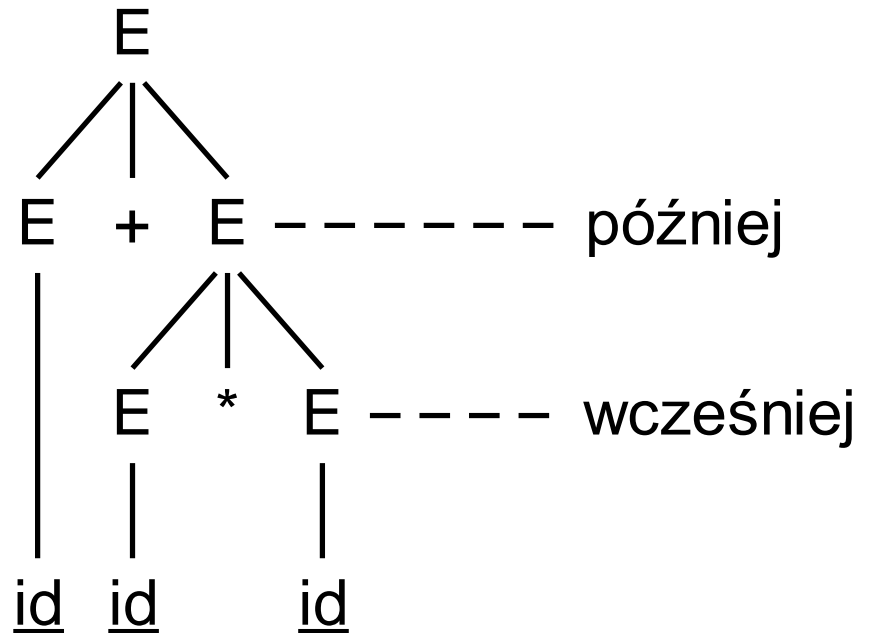
Stos:

$T_0 E T_1 + T_4 E T_7$

we:

$* \underline{id} \$$

DECYZJA: $f(T_7, *) = \underline{shift} \ 5$





Przykład gramatyki niejednoznacznej

Rozważamy ciąg wejściowy: id+id+id

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

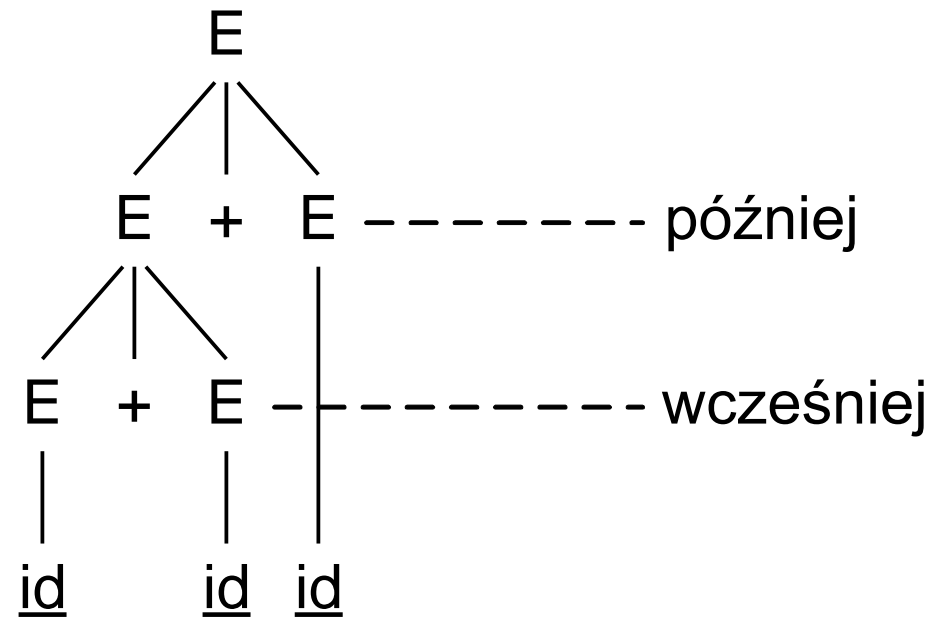
Stos:

$T_0ET_1+T_4ET_7$

we:

+ id \$

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



Przykład – gramatyka niejednoznaczna, usuwanie konfliktów

stan	<u>id</u>	+	*	()	\$	E
T ₀	<u>shift 3</u>			<u>shift 2</u>			T ₁
T ₁		<u>shift 4</u>	<u>shift 5</u>			<u>acc</u>	
T ₂	<u>shift 3</u>			<u>shift 2</u>			T ₆
T ₃		<u>red 4</u>	<u>red 4</u>		<u>red 4</u>	<u>red 4</u>	
T ₄	<u>shift 3</u>			<u>shift 2</u>			T ₇
T ₅	<u>shift 3</u>			<u>shift 2</u>			T ₈
T ₆		<u>shift 4</u>	<u>shift 5</u>		shift 9		
T ₇		<u>shift 4</u> <u>red 1</u>	<u>shift 5</u> red 1		<u>red 1</u>	<u>red 1</u>	
T ₈		<u>shift 4</u> <u>red 2</u>	<u>shift 5</u> <u>red 2</u>		<u>red 2</u>	<u>red 2</u>	
T ₉		<u>red 3</u>	<u>red 3</u>		<u>red 3</u>	<u>red 3</u>	

Symulacja działania parsera SLR dla gramatyki niejednoznacznej

Stos	Wejście	Wyjście
T_0	<u>id</u> + <u>id</u> \$	ϵ
T_0 <u>id</u> T_3	+ <u>id</u> \$	ϵ
T_0 E T_1	+ <u>id</u> \$	4
T_0 E T_1 + T_4	<u>id</u> \$	4
T_0 E T_1 + T_4 <u>id</u> 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	<u>id</u> + <u>id</u> \$	ϵ
E'T	<u>id</u> + <u>id</u> \$	1
E'T'F	<u>id</u> + <u>id</u> \$	14
E'T' <u>id</u>	<u>id</u> + <u>id</u> \$	148
E'T'	+ <u>id</u> \$	148
E'	+ <u>id</u> \$	1486
E'T+	+ <u>id</u> \$	14862
E'T	<u>id</u> \$	14862
E'T'F	<u>id</u> \$	148624
E'T' <u>id</u>	<u>id</u> \$	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