Zadanie:

Rozważmy następującą gramatykę niejednoznaczną dla n dwuargumentowych operatorów infiksowych:

$E \rightarrow E\ \theta_1\ E \mid E\ \theta_2\ E \mid ... \mid E\ \theta_n\ E \mid (E) \mid id$

Przyjmijmy, że wszystkie operatory są lewostronnie łączne i że θ_i ma priorytet wyższy niż θ_j, jeżeli $i > j$.

a) Zbuduj zbiory sytuacji SLR dla tej gramatyki. Jak wiele zbiorów powstaje w zależności od n?
b) Zbuduj tablicę analizatora SLR.
c) Ilu kroków wymaga wyprowadzenie "$id\ \theta_i\ id\ \theta_j\ id$" ?

Rozwiązanie:

Ponumerujmy najpierw kolejne produkcje gramatyki. Ponieważ zarówno ilość produkcji, jak i operatorów jest nie określona z góry i zależy od n, w trakcie rozwiązywania zadania będą się pojawiać grupy produkcji, zbiorów czy stanów o długości n, które będą do siebie podobne, a różnić się będą tylko indeksem operatora θ_i – takim elementy będziemy przydzielać specjalne numery postaci i.j, tak jak poniżej – 1.1 .. 1.n, aby wyróżnić je jako grupę.

Oto lista produkcji w gramatyce:

(0) $E' \rightarrow E$
(1.1) $E \rightarrow E\ \theta_1\ E$
(1.2) $E \rightarrow E\ \theta_2\ E$
...
(1.n) $E \rightarrow E\ \theta_n\ E$
(2) $E \rightarrow (E)$
(3) $E \rightarrow id$

Zbudujmy następnie kanoniczny system zbiorów SLR(1) sytuacji dopuszczalnych.

$T_0 = \{ [E' \rightarrow .\ E], [E \rightarrow .\ E\ \theta_1\ E], [E \rightarrow .\ E\ \theta_2\ E], ..., [E \rightarrow .\ E\ \theta_n\ E], [E \rightarrow .\ (\ E\ )], [E \rightarrow .\ id] \}$
$T_1 = GOTO(T_0, E) = \{ [E' \rightarrow E\ .], [E \rightarrow E\ .\ \theta_1\ E], [E \rightarrow E\ .\ \theta_2\ E], ..., [E \rightarrow E\ .\ \theta_n\ E] \}$
$T_2 = GOTO(T_0, \text{“(“}) = \{ [E \rightarrow (\ .\ E\ )], [E \rightarrow .\ E\ \theta_1\ E], [E \rightarrow .\ E\ \theta_2\ E], ..., [E \rightarrow .\ E\ \theta_n\ E], [E \rightarrow .\ (\ E\ )],$
$[E \rightarrow .\ id] \}$
$T_3 = GOTO(T_0, id) = \{ [E \rightarrow id\ .] \}$

$T_{4.1} = GOTO(T_1, \theta_1) = \{ [E \rightarrow E\ \theta_1\ .\ E], [E \rightarrow .\ E\ \theta_1\ E], [E \rightarrow .\ E\ \theta_2\ E], ..., [E \rightarrow .\ E\ \theta_n\ E], [E \rightarrow .\ (\ E\ )],$
$[E \rightarrow .\ id] \}$
$T_{4.2} = GOTO(T_1, \theta_2) = \{ [E \rightarrow E\ \theta_2\ .\ E], [E \rightarrow .\ E\ \theta_1\ E], [E \rightarrow .\ E\ \theta_2\ E], ..., [E \rightarrow .\ E\ \theta_n\ E], [E \rightarrow .\ (\ E\ )],$
$[E \rightarrow .\ id] \}$
...
$T_{4.n} = GOTO(T_1, \theta_n) = \{ [E \rightarrow E\ \theta_n\ .\ E], [E \rightarrow .\ E\ \theta_1\ E], [E \rightarrow .\ E\ \theta_2\ E], ..., [E \rightarrow .\ E\ \theta_n\ E], [E \rightarrow .\ (\ E\ )],$
$[E \rightarrow .\ id] \}$

$T_5 = GOTO(T_2, E) = \{ [E \rightarrow (\ E\ .\ )], [E \rightarrow E\ .\ \theta_1\ E], [E \rightarrow E\ .\ \theta_2\ E], ..., [E \rightarrow E\ .\ \theta_n\ E] \}$
$GOTO(T_2, \text{“(“}) = T_2$, $GOTO(T_2, id) = T_3$

$T_{6.1} = GOTO(T_{4.1}, E) = \{ [E \rightarrow E\ \theta_1\ E\ .], [E \rightarrow E\ .\ \theta_1\ E], [E \rightarrow E\ .\ \theta_2\ E], ..., [E \rightarrow E\ .\ \theta_n\ E] \}$
$T_{6.2} = GOTO(T_{4.2}, E) = \{ [E \rightarrow E\ \theta_2\ E\ .], [E \rightarrow E\ .\ \theta_1\ E], [E \rightarrow E\ .\ \theta_2\ E], ..., [E \rightarrow E\ .\ \theta_n\ E] \}$
...

$T_{6.n}$ = GOTO($T_{4.n}$, E) = { [E → E θ_n E **.**], [E → E **.** θ_1 E], [E → E **.** θ_2 E], ..., [E → E **.** θ_n E] }
∀i∈1..n: GOTO($T_{4.i}$, “(“) = T_2, GOTO($T_{4.i}$, *id*) = T_3

T_7 = GOTO(T_5, “)”) = { [E → (E) **.**] }
∀i∈1..n: GOTO(T_5, θ_i) = $T_{4.i}$
∀i∈1..n, ∀j∈1..n: GOTO($T_{6.j}$, θ_i) = $T_{4.i}$

To wszystkie stany jakie możemy utworzyć. Zatem odpowiedź na pytanie a) brzmi: zbiorów jest 2n+6 (T_0, T_1, T_2, T_3, T_5, T_7, oraz dwie serie $T_{4.1}$ – $T_{4.n}$ i $T_{6.1}$ – $T_{6.n}$). Teraz możemy zbudować tablicę parsera; obydwa wymiary tablicy zależą od n, więc nie można narysować jej w całości – serie elementów 1..n będziemy reprezentować przez elementy o indeksach 1, i oraz n. Pola “-” oznaczają błąd (err).

Do zbudowania tablicy będą nam jeszcze potrzebne zbiory FOLLOW nieterminali E i E':

$FIRST_1$(E') = { “(“, *id* }
$FIRST_1$(E) = { “(“, *id* }

$FOLLOW_1$(E') = { \$ }
$FOLLOW_1$(E) = { \$, θ_1, θ_2, ..., θ_n, “)” }

	θ_1	**θ_k**	**θ_n**	**(**	**)**	***id***	**\$**	**E**
T_0	-	-	-	sh 2	-	sh 3	-	T_1
T_1	sh 4.1	sh 4.k	sh 4.n	-	-	-	acc	-
T_2	-	-	-	sh 2	-	sh 3	-	T_5
T_3	red 3	red 3	red 3	-	red 3	-	red 3	-
$T_{4.1}$	-	-	-	sh 2	-	sh 3	-	$T_{6.1}$
$T_{4.i}$	-	-	-	sh 2	-	sh 3	-	$T_{6.i}$
$T_{4.n}$	-	-	-	sh 2	-	sh 3	-	$T_{6.n}$
T_5	sh 4.1	sh 4.k	sh 4.n	-	sh 7	-	-	-
$T_{6.1}$	sh 4.1 red 1.1	sh 4.k red 1.1	sh 4.n red 1.1	-	red 1.1	-	red 1.1	-
$T_{6.i}$	sh 4.1 red 1.i	sh 4.k red 1.i	sh 4.n red 1.i	-	red 1.i	-	red 1.i	-
$T_{6.n}$	sh 4.1 red 1.n	sh 4.k red 1.n	sh 4.n red 1.n	-	red 1.n	-	red 1.n	-
T_7	red 2	red 2	red 2	-	red 2	-	red 2	-

Jak widać, w tabeli pojawiły się konflikty – co było nieuchronne, ponieważ gramatyka od której zaczęliśmy była niejednoznaczna. Należy się teraz zastanowić, jak usunąć te konflikty, biorąc pod uwagę dwie dodatkowe informacje podane na początku:

- wszystkie operatory są lewostronnie łączne
- najwyższy priorytet ma θ_n, potem θ_{n-1}, itd., a najniższy ma θ_1

Konflikty występują tylko we fragmencie tablicy zaznaczonym na pomarańczowo, o rzeczywistym rozmiarze n×n: dla stanów od $T_{6.1}$ do $T_{6.n}$ i dla terminali od θ_1 do θ_n. Przeanalizujmy dokładniej sytuacje których

te pola dotyczą. Co oznacza że automat jest w stanie $T_{6.i}$? Wiemy, że:
$T_{6.i}$ = GOTO($T_{4.i}$, E)
$T_{4.i}$ = GOTO(T_1, θ_i)
T_1 = GOTO(T_0, E)

Zatem jeżeli bierzemy pod uwagę pole ($T_{6.i}$, θ_k), to przedstawiona w nim sytuacja jest następująca: parser wczytał wcześniej tokeny E, θ_i, E, a kolejnym tokenem czekającym na wejściu jest θ_k. Fragment wyrażenia który jest dla nas istotny, wygląda więc prawdopodobnie tak: "E θ_i E **.** θ_k E" (gdzie kropka oznacza miejsce do którego doszedł parser). Co w takiej sytuacji powinien on zrobić? To zależy od indeksów i oraz k – są możliwe trzy opcje:

- i < k: wtedy prawy operator (θ_k) ma większy priorytet, więc fragment "E θ_k E" powinien zostać zredukowany w pierwszej kolejności; dlatego należy wybrać operację <u>shift</u>, aby najpierw wczytać resztę wyrażenia, potem zredukować "E θ_k E", a na końcu zredukować "E θ_i E"
- i > k: wtedy lewy operator (θ_i) ma większy priorytet, więc fragment "E θ_i E" powinien zostać zredukowany w pierwszej kolejności; dlatego należy wybrać operację <u>redukcji</u>, aby najpierw zredukować "E θ_i E", potem wczytać resztę wyrażenia, a na końcu zredukować "E θ_k E"
- i = k: wtedy operatory mają równe priorytety, zatem kierujemy się drugą informacją – że wszystkie operatory mają lewostronną łączność. Oznacza to, że "E θ_i E θ_i E" powinno zostać zinterpretowane jako "(E θ_i E) θ_i E", więc tak jak w poprzednim przypadku, wybieramy <u>redukcję</u>.

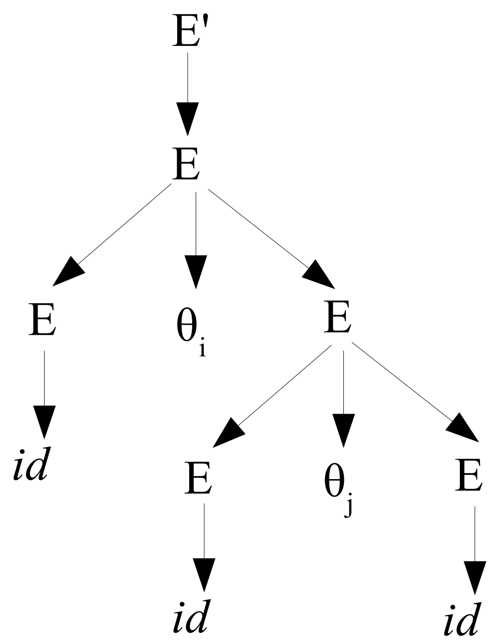
A więc konflikty powinny zostać rozwiązane w następujący sposób: dla i < k wybieramy shift, a dla i ≥ k wybieramy redukcję. Oto omawiany fragment tabeli w wersji ostatecznej, dla ustalenia uwagi zakładamy że n=5:

	θ_1	θ_2	θ_3	θ_4	θ_5
$T_{6.1}$	~~sh 4.1~~ red 1.1	sh 4.2 ~~red 1.1~~	sh 4.3 ~~red 1.1~~	sh 4.4 ~~red 1.1~~	sh 4.5 ~~red 1.1~~
$T_{6.2}$	~~sh 4.1~~ red 1.2	~~sh 4.2~~ red 1.2	sh 4.3 ~~red 1.2~~	sh 4.4 ~~red 1.2~~	sh 4.5 ~~red 1.2~~
$T_{6.3}$	~~sh 4.1~~ red 1.3	~~sh 4.2~~ red 1.3	~~sh 4.3~~ red 1.3	sh 4.4 ~~red 1.3~~	sh 4.5 ~~red 1.3~~
$T_{6.4}$	~~sh 4.1~~ red 1.4	~~sh 4.2~~ red 1.4	~~sh 4.3~~ red 1.4	~~sh 4.4~~ red 1.4	sh 4.5 ~~red 1.4~~
$T_{6.5}$	~~sh 4.1~~ red 1.5	~~sh 4.2~~ red 1.5	~~sh 4.3~~ red 1.5	~~sh 4.4~~ red 1.5	~~sh 4.5~~ red 1.5

Pola na przekątnej oraz poniżej niej zawierają redukcję, a pola powyżej przekątnej – shift.

c) Ilu kroków wymaga wyprowadzenie: “*id* θ_i *id* θ_j *id*” ?

Oto drzewo rozbioru syntaktycznego danego wyrażenia:



Widać, że do wyprowadzenia danego wyrażenia z E' potrzeba 6 kroków: $E' \Rightarrow E \Rightarrow E\ \theta_i\ E \Rightarrow E\ \theta_i\ E\ \theta_j\ E \Rightarrow id\ \theta_i\ E\ \theta_j\ E \Rightarrow id\ \theta_i\ id\ \theta_j\ E \Rightarrow id\ \theta_i\ id\ \theta_j\ id$ (lub 11 kroków, licząc razem z shiftami potrzebnymi do wczytania wyrażenia).