**Вариант 2 (1.1.2)**

**Грамматика:**

Показать, что грамматика G=({E}, {'.', '?', ':', '(', ')', id}, P, E) с правилами P={E -> E.id(E) | (E) | E?E:E | id} не является LL(1)-грамматикой. Найти эквивалентную ей LL(1)-грамматику G', построить LL(1)-анализатор для G' и выполнить разбор цепочки w = "(id)?(id.id(id)):(id?id:id)".

Нетерминалы: E

Терминалы: . ? : ( ) id

Выводы:

E -> E . id ( E )

E -> ( E )

E -> E ? E : E

E -> id

Не является LL(1) грамматикой так как содержит левую рекурсию. Нетерминал один, достаточно процедуры устранения непосредственной левой рекурсии:

G’:

Нетерминалы: E E'

Терминалы без изменений:

Выводы:

E' -> . id ( E ) E'

E -> ( E ) E'

E' -> ? E : E E'

E -> id E'

E' -> \_\_eps

FIRST(E) = ( id

FIRST(E’) = \_\_eps . ?

FOLLOW(E) = $ ) : . ?

FOLLOW(E’) = $ ) : . ?

Не является LL1, и левая факторизация не помогает.

Построим таблицу анализа этой грамматики:

Для продукции E' -> . id ( E ) E':

FIRST( . id ( E ) E' ) = . - добавляем E' -> . id ( E ) E' в ячейку M[E', . ]

Для продукции E -> ( E ) E':

FIRST( ( E ) E' ) = ( - добавляем E -> ( E ) E' в ячейку M[E, ( ]

Для продукции E' -> ? E : E E' - добавляем E' -> ? E : E E' в ячейку M[E', ? ]

Для продукции E -> id E' - добавляем E -> id E' в ячейку M[E, id ]

Для продукции E' -> \_\_eps:

FOLLOW(E’) = $ ) : . ?

добавляем E' -> \_\_eps в ячейки

таблица:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | . | ? | : | ( | ) | id | $ |
| E |  |  |  | ( E ) E' |  | id E' |  |
| E’ | . id ( E ) E' \_\_eps | ? E : E E'  \_\_eps | \_\_eps |  | \_\_eps |  | \_\_eps |

Разбор цепочки w = "(id)?(id.id(id)):(id?id:id)".

По факту это цепочка ( id ) ? ( id . id ( id ) ) : ( id ? id : id ) $

Разбор:

|  |  |  |
| --- | --- | --- |
| Стек | Вход | Выход |
| $E | (id)?(id.id(id)):(id?id:id)$ |  |
| $E’)E( | (id)?(id.id(id)):(id?id:id)$ | E -> ( E ) E’ |
| $E’)E | id)?(id.id(id)):(id?id:id)$ |  |
| $E’)E’id | id)?(id.id(id)):(id?id:id)$ | E -> id E’ |
| $E’)E’ | )?(id.id(id)):(id?id:id)$ |  |
| $E’) | )?(id.id(id)):(id?id:id)$ | E’ -> \_\_eps |
| $E’ | ?(id.id(id)):(id?id:id)$ |  |
| $E’E:E? | ?(id.id(id)):(id?id:id)$ | E’ -> ? E : E E’ |
| $E’E:E | (id.id(id)):(id?id:id)$ |  |
| $E’E:E’)E( | (id.id(id)):(id?id:id)$ | E -> ( E ) E’ |
| $E’E:E’)E | id.id(id)):(id?id:id)$ |  |
| $E’E:E’)E’id | id.id(id)):(id?id:id)$ | E -> id E’ |
| $E’E:E’)E’ | .id(id)):(id?id:id)$ |  |
| $E’E:E’)E’)E(id. | .id(id)):(id?id:id)$ | E’ -> . id ( E ) E’ |
| $E’E:E’)E’)E(id | id(id)):(id?id:id)$ |  |
| $E’E:E’)E’)E( | (id)):(id?id:id)$ |  |
| $E’E:E’)E’)E | id)):(id?id:id)$ |  |
| $E’E:E’)E’)E’id | id)):(id?id:id)$ | E -> id E’ |
| $E’E:E’)E’)E’ | )):(id?id:id)$ |  |
| $E’E:E’)E’) | )):(id?id:id)$ | E’ -> \_\_eps |
| $E’E:E’) | ):(id?id:id)$ | E’ -> \_\_eps |
| $E’E: | :(id?id:id)$ | E’ -> \_\_eps |
| $E’E | (id?id:id)$ |  |
| $E’E’)E( | (id?id:id)$ | E -> ( E ) E’ |
| $E’E’)E | id?id:id)$ |  |
| $E’E’)E’id | id?id:id)$ | E -> id E’ |
| $E’E’)E’ | ?id:id)$ |  |
| $E’E’)E’E:E? | ?id:id)$ | E’ -> ? E : E E’ |
| $E’E’)E’E:E | id:id)$ |  |
| $E’E’)E’E:E’id | id:id)$ | E -> id E’ |
| $E’E’)E’E:E’ | :id)$ |  |
| $E’E’)E’E: | :id)$ | E’ -> \_\_eps |
| $E’E’)E’E | id)$ |  |
| $E’E’)E’E’id | id)$ | E -> id E’ |
| $E’E’)E’E’ | )$ |  |
| $E’E’)E’ | )$ | E’ -> \_\_eps |
| $E’E’) | )$ | E’ -> \_\_eps |
| $E’E’ | $ |  |
| $E’ | $ | E’ -> \_\_eps |
| $ | $ | E’ -> \_\_eps |

Если попытаться применить левую факторизацию...

Устраним \_\_eps-вывод:

G’:

Нетерминалы: E E'

Терминалы без изменений:

Выводы:

E' -> . id ( E ) E'

E' -> . id ( E )

E -> ( E ) E'

E -> ( E )

E' -> ? E : E E'

E' -> ? E : E

E -> id E'

E -> id

И применим левую факторизацию:

E’ -> . id ( E ) D

D -> E’

D -> \_\_eps

E -> ( E ) F

F -> E’

F -> \_\_eps

E’ -> ? E : E G

G -> E’

G -> \_\_eps

E -> id H

H -> E’

H -> \_\_eps

Та же схема, что и раньше, но с ещё большим числом \_\_eps – продукций. Привести к чистой LL1 не удастся.

**Вариант 2 (1.2.1)**

**Грамматика:**

Для грамматики G = ({S, A, B}, {**a**, **b**, '(', ')', '>'}, P, S} с правилами P = {S  A, S B, A  (A), A  **a,**

B  (B>, B **b**} построить LR(1)-анализатор и выполнить разбор цепочки w = "(((((**b**>>>>>".