Тема 3. Восходящий синтаксический анализ

3.1. Реализация S-атрибутных СУО

Наиболее простым является использование СУТ для реализации *S*-атрибутного СУО в процессе восходящего синтаксического анализа. В этом случае лежащая в основе СУО грамматика должна принадлежать классу *LR*. Преобразование *S*-атрибутного СУО в СУТ заключается в размещении действий в конце продукции. Эти действия выполняются в процессе свертки правой части продукции в нетерминал из левой части. СУТ со всеми действиями, расположенными в конце правой части продукции, называются *постфиксными*.

Пусть дано *S*-атрибутное СУО вычисления простого арифметического выражения, представленное в табл. 4.

Таблица 4

S-атрибутное СУО вычисления простого арифметического выражения

Продукция	Семантические правила
1) $S \to E\perp$	$Print\,(E.val)$
2) $E \to E_1 + T$	$E.val := E_1.val + T.val$
3) $E \to T$	$E.val := T.val$
4) $T \to T_1 * F$	$T.val := T_1.val * F.val$
5) $T \to F$	$T.val := F.val$
6) $F \to (E)$	$F.val := E.val$
7) $F \to$ **num**	$F.val := GetVal\,(\textbf{num}.ns)$

Каждый нетерминал имеет по одному синтезируемому атрибуту *val*, терминал **num** имеет синтезируемый атрибут *ns* (номер строки в таблице символов, предоставляемый лексическим анализатором в качестве атрибута токена числовой константы **num**). Функция *GetVal* (**num**.*ns*) возвращает значение числовой константы из соответствующей строки таблицы символов. Семантическое правило *Print*(*E.val*) первой продукции выводит результат вычисления выражения.

Постфиксная СУТ, реализующая данное СУО, имеет следующий вид:

$S \to E\perp$ $\{Print\,(E.val)\}$

$E \to E_1 + T$ $\{E.val := E_1.val + T.val\}$

$E \to T$ $\{E.val := T.val\}$

$T \to T_1 * F$ $\{T.val := T_1.val * F.val\}$

$T \to F$ $\{T.val := F.val\}$

$F \to (E)$ $\{F.val := E.val\}$

$F \to$ **num** $\{F.val := GetVal\,(\textbf{num}.ns)\}$.

Атрибут символа грамматики можно разместить в стеке состояний *LR*-анализатора вместе с состоянием, представляющим этот символ (модифицировав соответствующим образом структуру элемента стека), или использовать специальный стек для хранения значений атрибутов.

После детализации действий соответствующими операциями со стеком (используется отдельный стек атрибутов) постфиксная СУТ примет следующий вид:

$S \to E\perp$ $\{Print\,(Pop\,(St))\}$

$E \to E_1 + T$ $\{t_2 := Pop\,(St);\ t_1 := Pop\,(St);\ Push\,(t_1 + t_2,\ St)\}$

$E \to T$

$T \to T_1 * F$ $\{t_2 := Pop\,(St);\ t_1 := Pop\,(St);\ Push\,(t_1 * t_2,\ St)\}$

$T \to F$

$F \to (E)$

$F \to$ **num** $\{Push\,(GetVal\,(\textbf{num}.ns),\ St)\}$.

Здесь операция $Push\,(x,\ St)$ размещает значение x в стеке St, функция $Pop\,(St)$ исключает элемент из вершины стека и возвращает его значение; t_1 и t_2 – промежуточные переменные.

Примеры реализации постфиксных СУТ с использованием стека синтаксического анализатора можно посмотреть в [1; 2].