§ 3. Вычисление атрибутов, совмещенное с LR-анализом (восходящий анализ)

Опр. Атрибутная грамматика называется S-атрибутной, если все атрибуты – синтезируемые.

Замечание: естественный способ вычисления синтезируемых атрибутов снизу-вверх совпадает с восстановлением снизу-вверх дерева вывода при LR-анализе.
Следовательно, синтаксический анализ можно совместить с семантическим, т.е. с вычислением атрибутов.

Ключевые формулировки:
1. Каждый элемент в стеке будем считать записью (терминология из дисциплины «Базы данных»). Запись имеет два поля:
синтаксическое поле state (состояние);
семантическое поле val (значение синтезируемого атрибута).
Если атрибутов несколько, то val – вектор (atr1, atr2,…).

2. К элементам стека можно обращаться, указав верхний элемент: (state[top], val[top]), или указав смещение от верхнего элемента: (state[top – i], val[top – i]).

3. Если при LR-анализе происходит перенос, то в начале такта выполняется присваивание ntop := top + 1.
Далее: state[ntop] := b^j, val[ntop] := b.lexval,
где b – терминал, для которого выполняется перенос,
b^j – название состояния LR(k)-автомата, которое записывается в стек,
b.lexval – лексическое значение терминала b.
В конце такта выполняется присваивание top := ntop.

4. Если происходит свертка, то выполняются присваивания:

ntop := top – r + 1,

state[ntop] := A^j,

val[ntop] := f(val[top – r + 1],…, val[top – 1], val[top]),

top := ntop.

где $A \to \alpha$ – правило, по которому выполняется свертка,

r – длина цепочки α,

A^j – название состояния LR(k)-автомата, которое записывается в стек,

f – векторная функция, применяемая в семантических правилах.

Пример. Построим два фрагмента протокола анализа числового выражения 2 + 5∗3, которому соответствует цепочка w = x + x∗x в расширенной грамматике

G = {E′→E, E→E+T| T, T→T∗F| F, F→(E)| x}

Такт	Стек	Входной поток	Пояснения
1	$\begin{pmatrix}\nabla\\ \varnothing\end{pmatrix}$	◊x + x∗x⊣	top = 1
			Перенос
2	$\begin{pmatrix}\nabla\\ \varnothing\end{pmatrix}\begin{pmatrix}x^1\\ 2\end{pmatrix}$	x◊ + x∗x⊣	top = 2
			Свертка F→x
3	$\begin{pmatrix}\nabla\\ \varnothing\end{pmatrix}\begin{pmatrix}F^1\\ 2\end{pmatrix}$	x◊ + x∗x⊣	top = 2
			Свертка T→F

Такт	Стек	Входной поток	Пояснения
4	$\begin{pmatrix}\nabla\\ \varnothing\end{pmatrix}\begin{pmatrix}T^{1}\\ 2\end{pmatrix}$	x◊ + x∗x⊣	top = 2
			Свертка E→T
5	$\begin{pmatrix}\nabla\\ \varnothing\end{pmatrix}\begin{pmatrix}E^{1}\\ 2\end{pmatrix}$	x◊ + x∗x⊣	top = 2
			Перенос
6	$\begin{pmatrix}\nabla\\ \varnothing\end{pmatrix}\begin{pmatrix}E^{1}\\ 2\end{pmatrix}\begin{pmatrix}+^{1}\\ \varnothing\end{pmatrix}$	x + ◊x∗x⊣	top = 3

Такт	Стек	Входной поток	Пояснения
k	$\begin{pmatrix}\nabla\\ \varnothing\end{pmatrix}\begin{pmatrix}E^1\\ 2\end{pmatrix}\begin{pmatrix}+^1\\ \varnothing\end{pmatrix}\begin{pmatrix}T\\ 5\end{pmatrix}\begin{pmatrix}*\\ \varnothing\end{pmatrix}\begin{pmatrix}F\\ 3\end{pmatrix}$	x + x∗x◊⊣	top = 6
			Свертка T→T∗F
k+1	$\begin{pmatrix}\nabla\\ \varnothing\end{pmatrix}\begin{pmatrix}E^1\\ 2\end{pmatrix}\begin{pmatrix}+^1\\ \varnothing\end{pmatrix}\begin{pmatrix}T\\ 15\end{pmatrix}$	x + x∗x◊⊣	top = 4
			Свертка E→E+T
k +2	$\begin{pmatrix}\nabla\\ \varnothing\end{pmatrix}\begin{pmatrix}E^1\\ 17\end{pmatrix}$	x + x∗x◊⊣	top = 2
…			

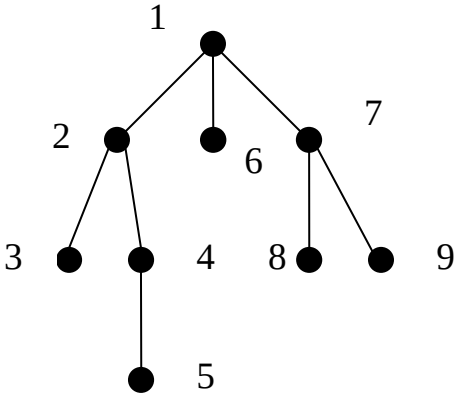
§ 4. Вычисление атрибутов, совмещенное с нисходящим анализом

Опр. Атрибутная грамматика называется L-атрибутной, если атрибуты либо синтезируемые, либо наследуемые с условием:
для правила $A \to X_1 \ldots X_n$, для любого $i = 1,\ldots n$, наследуемый атрибут $X_i.atr$ зависит от атрибутов символов $X_1, X_2,..., X_{i-1}$ (левые братья), либо от наследуемого атрибута символа A (отец).

Замечание: любая S-атрибутная грамматика является L-атрибутной.

Обход в глубину дерева вывода назовем стандартным, если сыновья любого узла просматриваются слева направо.

Пример стандартного обхода в глубину в дереве.



Теорема. Приведенная атрибутная грамматика G является L-атрибутной $\Leftrightarrow \forall \alpha \in (\Gamma \cup \Sigma)^* \setminus \{\varepsilon\} : S \Rightarrow^* \alpha$ (синтаксически корректная цепочка α) все атрибуты символов в α могут быть вычислены стандартным обходом в глубину дерева вывода.

Чтобы совместить синтаксический нисходящий анализ с семантическим используют вариант атрибутной грамматики, называемый «схемой трансляции».

В схеме трансляции семантические правила (действия) записывают среди символов правила вывода:

$A \rightarrow$ [действие 1]X_1[действие 2]$X_2 \ldots X_n$[действие (n +1)]

где $A \rightarrow X_1X_2 \ldots X_n$ – правило вывода,
для i : $1 \leq i \leq n$, [действие i] = вычисление наследуемого атрибута символа X_i,
[действие (n + 1)] = вычисление синтезируемого атрибута символа A.

Замечание: для корректной работы нисходящего анализа необходимо, чтобы в грамматике не было левой рекурсии.
В частности, LL-грамматики не содержат левой рекурсии.

Если S-атрибутная грамматика леворекурсивна, то можно построить эквивалентную ей L-атрибутную грамматику без левой рекурсии.

(В общем случае L-атрибутной грамматики нет алгоритма устранения левой рекурсии).

Напоминание: нетерминал B называется леворекурсивным, если $B \Rightarrow^{+} B\gamma$.

1. Нетерминал A называется непосредственно леворекурсивным, если существуют правила:
$A \to A\alpha_1 | \ldots | A\alpha_k | \beta_1 | \ldots | \beta_m$.

Устранение непосредственной рекурсии выполняется заменой этих правил на
$$\left\{ \begin{array}{l} A \to \beta_1 R | \ldots | \beta_m R, \\ R \to \alpha_1 R | \ldots | \alpha_k R | \varepsilon \end{array} \right\}$$

Алгоритм устранения непосредственной левой рекурсии в схеме трансляции S-атрибутной грамматики.

Пусть исходная схема трансляции:

$A \to A_1\alpha_1[A.a := f(A_1.a,\text{...атрибуты символов в } \alpha_1)]$

...

$A \to A_1\alpha_k[A.a := f(A_1.a,\text{...атрибуты символов в } \alpha_k)]$

$A \to \beta_1[A.a := f(\text{атрибуты символов в } \beta_1)]$

...

$A \to \beta_m[A.a := f(\text{атрибуты символов в } \beta_m)]$

Для новой схемы трансляции добавим новые атрибуты:
R.i – наследуемый, R.s – синтезируемый.

Новая схема трансляции:

$A \to \alpha_1 [R.i := g(\text{атрибуты символов в } \alpha_1)]\ R\ [A.a := R.s]$

...

$R \to \beta_1 [R_1.i := g(R.i, \ldots \text{атрибуты символов в } \beta_1)]\ R_1 [R.s := R_1.s]$

...

$R \to \varepsilon\ [R.s := R_1.i]$

Теорема. $\forall \alpha :\ A \Rightarrow^* \alpha$ значение A.a по старой схеме трансляции равно значению A.a по новой схеме.

2. Для устранения глобальной левой рекурсии применяется алгоритм, изложенный в прошлом семестре.

Его можно модифицировать с целью устранения глобальной левой рекурсии в схеме трансляции.

Пример устранения непосредственной левой рекурсии в схеме трансляции грамматики G = {E→E+T| T, T→ x}.

Атрибут .val – значение числового выражения.

Схема трансляции:

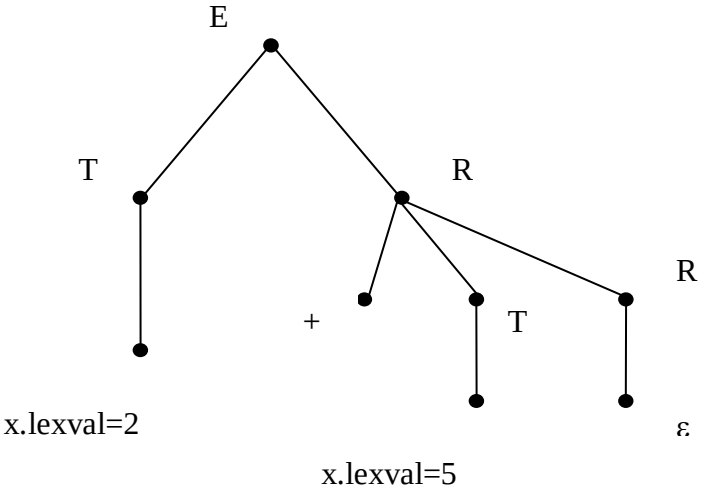
1	$E \rightarrow E_1+T$ [E.val := E_1.val + T.val]
2	E→T [E.val := T.val]
3	T→x [T.val := x.lexval]

В новой схеме правила 1 и 2 заменим на правила
E→TR,
R→+TR| ε.

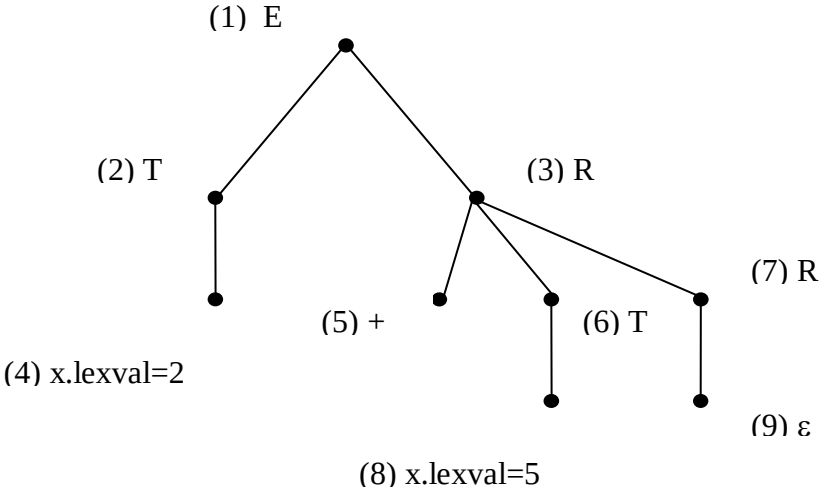
Новая схема трансляции:

	$E \rightarrow T\ [R.i := T.val]\ R\ [E.val := R.s]$
	$R \rightarrow +T\ \ [R_1.i := R.i + T.val]\ R_1\ [R.s := R_1.s]$
	$R \rightarrow \varepsilon\ [R.s := R.i]$
	$T \rightarrow x\ \ [T.val := x.lexval]$

Пример вычисления атрибутов при нисходящем анализе выражения 2 + 5:



Обход в глубину вершин дерева:



(1) R.i := T.val – не существ., E.val := R.s – не существ.
(2) T.val := 2.
(3), возврат в (2),
возврат в (1) R.i := 2, E.val := R.s – не существ.
(4) R_1.i := 2+T.val – не существ., R.s := R_1.s – не существ.
(5), возврат в (4),
(6) T.val := 5.
(7), возврат в (4) R_1.i := 2+5 = 7, R.s := R_1.s – не существ.
(8) R.s := R.i = 7.
(9), возврат в (8), в (4) R.s := R.s = 7.
возврат в (1) E.val := R.s = 7.

Рекурсивные функции для реализации схемы трансляции.

Опр. Нисходящий транслятор – программа, осуществляющая нисходящий синтаксический анализ одновременно с вычислением атрибутов, в соответствии с правилами схемы трансляции.

Нисходящий транслятор состоит из набора функций, взаимно однозначно соответствующих нетерминалам. Эти функции рекурсивно вызывают друг друга. Они используют глобальную переменную lookahead, значение которой = текущий символ входной строки в процессе разбора.

В начале разбора строки lookahead = первый символ.
Производится вызов функции S (где S – аксиома).
Так как S не может иметь наследуемых атрибутов, то функция S не имеет аргументов.

Для произвольного нетерминала A рекурсивная функция A имеет аргументы – наследуемые атрибуты A; функция возвращает синтезируемые атрибуты A.

По умолчанию можно использовать функцию, вычисляющую тип переменной или константы, и функцию, возвращающую семантическое правило для (A,a) нисходящего анализатора.

Пример рекурсивных функций для новой схемы трансляции грамматики G = {E→TR, R→+TR| ε, T→ x}.

Функция для нетерминала E (аксиома)
(E→T [R.i := T.val] R [E.val := R.s])

	E ( ) :int	Заголовок
1	i1, s1 : int	Объявление типа локальных переменных i1, s1
2	i1 := T( )	Вызов функции T
3	s1:= R(i1)	Вызов функции R
4	return s1	Возвращение результата функции E

Функция для нетерминала R

(R→+T [R_1.i := R.i + T.val] R_1 [R.s := R_1.s]

R→ε [R.s := R.i])

	R (i: int) :int	Заголовок
1	i1, s1 : int	Объявление типа локальных переменных i1, s1
2	if (lookahead = ′+′)	
3	i1:= i +T( )	Вызов функции T
4	s1:= R(i1)	Вызов функции R
5	return s1	Возвращение результата функции R
6	if (lookahead ≠′+′)	
7	s1:= i	
8	return s1	Возвращение результата функции R

Функция для нетерминала T
(T→x [T.val := x.lexval])

	T ( ) :int	Заголовок
1	s1 : int	Объявление типа локальных переменных s1
2	s1:= lookahead.lexval	
4	return s1	Возвращение результата функции T

В программы для функций можно добавить проверку, чему равен lookahead, и вывести сообщение об ошибке, если его значение не соответствует синтаксически правильным цепочкам.