1.2. Порядок выполнения семантических правил

Наиболее универсальной и мощной группой методов для определения порядка выполнения семантических правил являются так называемые *методы дерева разбора* [2]. В этих методах порядок выполнения определяется во время компиляции с помощью топологической сортировки [7] графа зависимостей, построенного по дереву разбора для входной строки. Методы не работают только в том случае, если построенный граф зависимостей будет иметь цикл (невозможно выполнить топологическую сортировку).

Граф зависимостей для дерева разбора представляет собой ориентированный граф, показывающий информационные связи между экземплярами атрибутов. Вершинам графа сопоставляются все экземпляры атрибутов во всех узлах дерева разбора. Если атрибут a зависит от атрибута b, то от вершины b ведет дуга к вершине a. Если в СУО имеются правила с побочными действиями, их необходимо привести к виду $b := f(c_1, c_2, \ldots, c_k)$ введением фиктивного синтезируемого атрибута b (при этом ни один атрибут не должен зависеть от b).

Граф зависимостей для дерева разбора (см. рис. 2) приведен на рис. 3 (ребра дерева разбора показаны пунктирными линиями). Вершины графа помечены числами, показывающими результаты одной из его топологических сортировок. Вершины 6 и 7 графа соответствуют фиктивным синтезируемым атрибутам, введенным для правила с побочными действиями *AddType* (**id.***pnt*, *L***.***inh*). Результатом топологической сортировки является следующий порядок выполнения семантических правил (a_i означает атрибут для вершины с номером i):

a_3 := char;
a_4 := a_3;
a_5 := a_4;
AddType (**id$_1$.***pnt*, a_5);
AddType (**id$_2$.***pnt*, a_4).

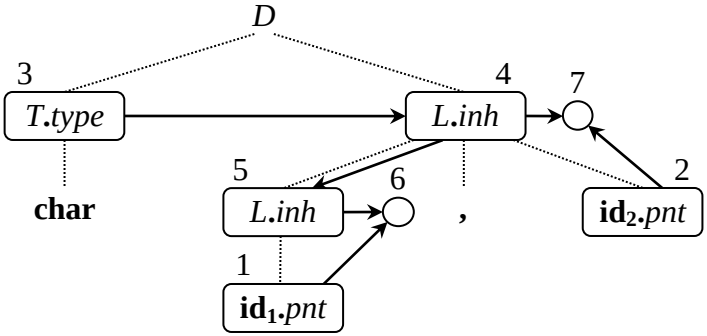

Рис. 3. Граф зависимостей для дерева разбора на рис. 2

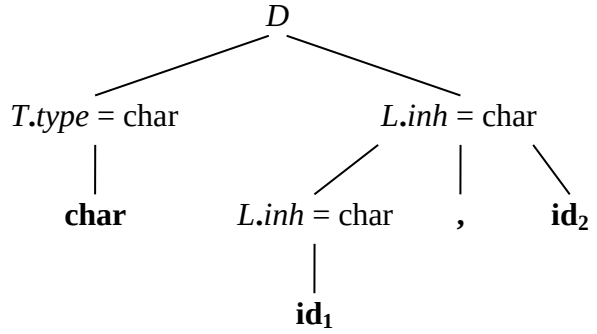

Рис. 2. Аннотированное дерево разбора строки **char id$_1$, id$_2$**

Вторая группа методов для определения порядка выполнения семантических правил основана на анализе семантических правил, связанных с продукциями. По результатам такого анализа предопределяется порядок вычисления атрибутов в процессе разработки компилятора.

Третью группу методов для определения порядка выполнения семантических правил составляют так называемые *игнорирующие методы*. В этих методах выбор порядка производится без рассмотрения семантических правил. В частности, если трансляция реализуется в процессе синтаксического анализа, то порядок выполнения правил определяется методом синтаксического анализа независимо от семантических правил. Очевидно, что эти методы применимы для ограниченных классов СУО. К таким классам относятся *S*-атрибутные и более общие *L*-атрибутные СУО.
В отличие от методов дерева разбора, где в процессе компиляции требуется явное построение дерева разбора и на его основе – графа зависимостей, методы, основанные на правилах, и игнорирующие методы этого не требуют. Поэтому они могут быть более эффективными как с точки зрения времени выполнения, так и объема используемой памяти.

1.3. L-атрибутные СУО

СУО называется *L-атрибутным*, если каждый атрибут является либо синтезируемым, либо наследуемым. При этом наследуемый атрибут символа X_j, $1 \le j \le n$ из правой части продукции $A \to X_1X_2...X_n$, вычисляемый с помощью правила, связанного с данной продукцией, зависит только от наследуемых атрибутов нетерминала A, наследуемых или синтезируемых атрибутов символов $X_1, X_2, ..., X_{j-1}$, расположенных слева от X_j, наследуемых атрибутов самого X_j (при условии, что зависимости не являются циклическими). Поскольку ограничения касаются только наследуемых атрибутов, каждое *S*-атрибутное СУО является истинным подмножеством *L*-атрибутных СУО. *L*-атрибутное СУО легко согласуется с нисходящими (*LL*-разбор) и восходящими (*LR*-разбор) методами синтаксического анализа.

Все рассмотренные выше в качестве примеров СУО (см. табл. 1 и 2) являются *L*-атрибутными. Пример СУО, не являющегося *L*-атрибутным, представлен в табл. 3.

Таблица 3

Не *L*-атрибутное СУО

Продукция	Семантические правила
1) $S \to A\ B$	$B.inh := f(A.inh, A.syn)$ $S.syn := g(A.syn, B.syn)$
2) $S \to C\ D$	$C.inh := h(S.inh, D.inh)$ $S.syn := g(C.syn, D.syn)$

Для продукции 1 правила являются корректными, поскольку наследуемый атрибут $B.inh$ зависит от атрибутов символа A, расположенного слева от B. Первое правило продукции 2 не позволяет СУО быть *L*-атрибутным, поскольку атрибут $C.inh$ зависит от атрибута $D.inh$ символа D, находящегося справа от C. Второе правило этой продукции вполне корректно.

Таким образом, в *L*-атрибутных СУО все атрибуты могут быть вычислены в соответствии со следующим прохождением дерева разбора (само дерево явно не строится). При прохождении вниз по дереву при первом посещении узла вычисляются наследуемые атрибуты этого узла. Для продукции $A \to X_1X_2...X_n$ поддеревья с корнями X_1, X_2, …, X_n обходят слева направо. После завершения обхода всех этих поддеревьев при возврате на уровень вверх, непосредственно перед тем как будет покинут узел *A*, вычисляются его синтезируемые атрибуты. Другими словами, наследуемые атрибуты вычисляются в соответствии с прямым прохождением дерева разбора, а синтезируемые атрибуты – в соответствии с обратным прохождением. Иногда эти прохождения объединяют в одно так называемое *прохождение в глубину* (тогда прямое и обратное прохождения рассматриваются как частные случаи этого общего прохождения). Тогда можно сказать, что все атрибуты *L*-атрибутного СУО могут быть вычислены в соответствии с прохождением в глубину дерева разбора, которое определяет порядок вычисления атрибутов согласно рекурсивной процедуре *Depth-First*, где *v* и *w* – узлы дерева разбора, *Sons* (*v*) – множество сыновей узла *v*.

```
procedure DepthFirst (v)
begin
    for w ∈ Sons (v) слева направо do
    begin
        Вычислить наследуемые атрибуты w
        DepthFirst (w)
    end
    Вычислить синтезируемые атрибуты v
end
```