Московский авиационный институт
(национальный исследовательский университет)

Институт информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Курсовая работа по курсу «Системы программирования»

# Тема: «Построение графа зависимостей и вычисление выражений для атрибутных грамматик»

Студент: Бирюков В. В.
Преподаватель: Семёнов А. С.
Группа: М8О-207Б-19
Дата:
Оценка:
Подпись:

# Определения

Атрибутивная транслирующая грамматика (далее АТ грамматика) – контекстно свободная грамматика, расширенная множеством операционных символов, которые могут встречаться в правых частях правил.

То есть грамматика G = (T, V, Op, S, P), такая, что в любом правиле

$A \to X_1X_2...X_n \in P$, $X_i \in T \cup V \cup Op$.

Каждый терминальный или нетерминальный символ грамматики может иметь конечное множество атрибутов, и каждый атрибут имеет (возможно, бесконечное) множество допустимых значений.

При задании символа X с множеством атрибутов $\{a_1, a_2, \ldots, a_m\}$ во множестве V или T будем обозначать его, как $X_{\{a1, a2, \ldots, am\}}$ или $X_A$, $A = \{a_1, a_2, \ldots, a_m\}$, $m \geq 0$.

$A \to X_{1\{a1, a2, \ldots, am\}}X_{2\{a1, a2, \ldots, am\}}...X_{n\{a1, a2, \ldots, am\}} \in P$, $X_i \in T \cup V \cup Op$.

Операционный символ – функция $Op_i : \{X_1, X_2, \ldots, X_n\} \to$ *действие*, которая сопоставляет множеству символов какое-либо действие, совершаемое с их атрибутами. Для описания операционных символов в правилах грамматики будем записывать их *действия* в фигурных скобках: $\{X_{1.a} = X_{2.a} + X_{3.a}\}$.

# Алгоритмы

**Алгоритм**: Выполнение АТ грамматики в процессе LL-анализа

**Вход**: Входная цепочка, управляющая таблица M LL-анализа

**Выход**: Результат выполнения АТ грамматики

LL-транслятор состоит из входной ленты, управляющего устройства (таблица M) и трех стеков: для магазинных символов, для хранения копий магазинных символов, для хранения длин примененных правил. Конфигурацию транслятора будем представлять в виде ($\alpha\$$, $X\beta\perp$, $X\gamma$, $x_1...x_k$), где $\alpha\$$ – входная цепочка, $X\beta\perp$ – цепочка в магазине, $X\gamma$ – сохраненные символы, $x_1...x_k$ – длины правил, символы которых сейчас находятся в стеке копий.

Также можно добавить выходную цепочку, с номерами примененных правил. Не будем рассматривать ее здесь для уменьшения размера конфигураций.

Управляющая таблица M LL-анализа строится по грамматике, соответствующей АТ грамматике без операционных символов, стандартным алгоритмом. (Или ее можно построить сразу по АТ грамматике, изменив функции FIRST и FOLLOW таким образом, чтобы они при вычислении пропускали операционные символы)

Множество магазинных символов дополним символом •, который будет обозначать конец правила.

Стек копий нужен для того, чтобы иметь доступ с атрибутам символов, которые уже отсутствуют в магазине. Стек длин правил используется для получения со стека копий нужного числа символов.

Рассмотрим, как выглядит такт работы транслятора с учетом дополнительных стеков.

1. Перенос

   Пусть на вершине магазина находится нетерминальный символ X, и согласно таблице M необходимо применить правило $X \to X_1, \ldots, X_n$. Тогда символ X удаляется из

магазина, символ конца правила • помещается в магазин, символы $X_1$, …, $X_n$ помещаются в магазин, не операционные символы $X_1$, …, $X_k$ помещаются в стек копий, копия символа X ниже из стека снова помещается на стек копий, длина правила без операционных символов, с учетом левой части, k+1 помещается в стек длин. Такт срабатывания: $(\alpha\$, X\beta\perp, \gamma, \delta) \vdash (\alpha\$, X_1 \ldots X_n\beta\perp, XX_1 \ldots X_k\gamma, k+1\ \delta)$

2. Выброс •

   Пусть на вершине магазина находится символ конца правила •. Это означает, что ранее было применено некое правило $X \rightarrow X_1, \ldots, X_n$, следовательно, k верхних символов стека копий это символы $X, X_1, \ldots, X_k$, верхнее число в стеке длин – k+1. Символы $X, X_1, \ldots, X_k$ необходимо удалить из стека копий, так как их атрибуты уже обработаны и для дальнейшей трансляции не нужны. Такт срабатывания: $(\alpha\$, \bullet\ \beta\perp, XX_1 \ldots X_k\gamma, k+1\ \delta) \vdash (\alpha\$, \beta\perp, \gamma, \delta)$

3. Выброс

   Пусть на вершине магазина и во входной цепочке находится терминальный символ $X_{\{a\}}$, возможно с некоторыми атрибутами. Это означает, что в стеке копий на некоторой глубине также находится символ $X_{\{a\}}$. В ходе выброса необходимо перенести значения атрибутов символа из входной цепочки в атрибуты символа в стеке копий. Такт срабатывания: $(X_{\{a=z\}}\alpha\$, X_{\{a\}}\beta\perp, \gamma_1 X_{\{a\}}\gamma_2, \delta) \vdash (\alpha\$, \beta\perp, \gamma_1 X_{\{a=z\}}\gamma_2, \delta)$

4. Выполнение

   Пусть на вершине магазина находится операционный символ Y. Операционный символ может производить операции только с символами в пределах его правила, следовательно, ему может понадобиться n верхних символов из стека, причем n – это верхний элемент стека длин. Такт срабатывания: $(\alpha\$, Y\beta\perp, X_{\{a\}}X_{1\{a\}}\ldots X_{n\{a\}}\gamma, n\ \delta) \vdash (\alpha\$, \beta\perp, X_{\{a=z0\}}X_{1\{a=z1\}}\ldots X_{n\{a=zn\}}\gamma, n\ \delta)$

Начальная конфигурация: $(\alpha\$, S\perp, \varepsilon, \varepsilon)$, где S – стартовый нетерминал.

Присвоить a первый символ входной строки

Присвоить X символ на вершине стека

```
do {
    Удалить символ из магазина
    if (X ∈ Op) {
        n = верхний элемент стека длин
        Выполнить функцию X, передав ей как аргументы n символов из стека копий
    } else if (X == •) {
        n = верхний элемент стека длин
        Удалить n символов из стека копий
        Удалить верхний элемент из стека длин
    } else if (M[X, a] = ВЫБРОС) {
        Скопировать атрибуты символа a в атрибуты символа X из стека копий
        Удалить X из магазина
    } else if (M[X, a] = A → α) {
```

```
            Поместить • в магазин
            k = 0 // Длина правила без операционных символов
            foreach (b ∈ α) {
                if (b ∉ Op) {
                    Поместить b на стек копий
                    k = k + 1
                }
                Поместить b в магазин
            }
            Поместить A из стека копий ниже на вершину стека копий
            Поместить k+1 в стек длин
        } else {
            Ошибка
            break
        }
        Присвоить a следующий символ входной строки
        Присвоить X символ на вершине стека
} while (X ≠ $)
if (a ≠ $) {
        Ошибка
}
```

# Пример

В качестве исходной грамматики возьмем КС-грамматику описывающую цепочку произведений чисел:

$G_0 = (\{n, +\}, \{T, T', F\}, T, P_0)$

$P_0$:

1. $T \rightarrow FT'$
2. $T' \rightarrow *FT'$
3. $T' \rightarrow \varepsilon$
4. $F \rightarrow n$

После добавления атрибутов, операционных символов и одного нетерминала, полученная АТ грамматика сможет вычислять значения таких выражений и печатать их.

$G = (\{n_{\{val\}}, *\}, \{S, T_{\{val\}}, T'_{\{inh,\ syn\}}, F_{\{val\}}\}, Op, S, P)$

P:

1. $S \rightarrow T_{\{val\}}$ $\{print(T_{val})\}$
2. $T_{\{val\}} \rightarrow F_{\{val\}}$ $\{T'_{inh} = F_{val}\}$ $T'_{\{inh,\ syn\}}$ $\{T_{val} = T'_{syn}\}$

3. $T'_{\{inh, syn\}} \rightarrow *F_{\{val\}} \{T'_{1.inh} = T'_{inh} * F_{val}\} T'_{1.\{inh, syn\}} \{T'_{syn} = T'_{1.syn}\}$
4. $T'_{\{inh, syn\}} \rightarrow \varepsilon \{T'_{syn} = T'_{inh}\}$
5. $F_{\{val\}} \rightarrow n_{\{val\}} \{F_{val} = n_{val}\}$

Предполагаем, что $n_{val}$ содержит числовое значение.

Рассмотрим, как такая АТ грамматика будет вычисляться в процессе LL анализа
Управляющая таблица имеет вид (операционные символы сокращены до {}):

| M | n | * | $ |
|---|---|---|---|
| S | S → T{} | | |
| T | T → F{}T'{} | | |
| T’ | | T' → *F{}T'{} | T' → ε{} |
| F | F → n{} | | |
| n | Выброс | | |
| * | | Выброс | |
| ⊥ | | | Допуск |
| ∀ X ∈ Op | Выполнение | Выполнение | Выполнение |

Пусть на вход подается цепочка *3*5*. Здесь необходим этап лексического анализа, в ходе которого она преобразуется в цепочку $n_{\{val=3\}} * n_{\{val=5\}}$.

Начальная конфигурация:
$(n_{\{val=3\}} * n_{\{val=5\}} \$, S\perp, \varepsilon, \varepsilon) \vdash$
Перенос согласно правилу $S \rightarrow T_{\{val\}} \{print(T_{val})\}$, S и $T_{\{val\}}$ помещаются на стек копий, 2 – на стек длин

$(n_{\{val=3\}} * n_{\{val=5\}} \$, T \{print(T_{val})\} \bullet \perp, S\ T_{\{val\}}, 2) \vdash$
Перенос согласно правилу $T_{\{val\}} \rightarrow F_{\{val\}} \{T'_{inh} = F_{val}\} T'_{\{inh, syn\}} \{T_{val} = T'_{syn}\}$, $T_{\{val\}}$, $F_{\{val\}}$ и $T'_{\{inh, syn\}}$ помещаются на стек копий, 3 – на стек длин

$(n_{\{val=3\}} * n_{\{val=5\}} \$, F \{T'_{inh} = F_{val}\} T' \{T_{val} = T'_{syn}\} \bullet \{print(T_{val})\} \perp, T_{\{val\}}\ F_{\{val\}}\ T'_{\{inh, syn\}}\ S\ T_{\{val\}}, 3\ 2) \vdash$
Перенос $F_{\{val\}} \rightarrow n_{\{val\}} \{F_{val} = n_{val}\}$

$(n_{\{val=3\}} * n_{\{val=5\}} \$, n \{F_{val} = n_{val}\} \bullet \{T'_{inh} = F_{val}\} T' \{T_{val} = T'_{syn}\} \bullet \{print(T_{val})\} \bullet \perp, F_{\{val\}}\ n_{\{val\}}\ T_{\{val\}}\ F_{\{val\}}\ T'_{\{inh, syn\}}\ S\ T_{\{val\}}, 2\ 3\ 2) \vdash$
Выброс. Атрибуты $n_{\{val=3\}}$ копируются в $n_{\{val\}}$ из стека символов

(* $n_{val=5}$ $, $\{F_{val} = n_{val}\} \bullet \{T'_{inh} = F_{val}\}$ $T'\{T_{val} = T'_{syn}\} \bullet \{print(T_{val})\} \bullet \perp$, $F_{\{val\}}$ $n_{\{val=3\}}$ $T_{\{val\}}$ $F_{\{val\}}$ $T'_{\{inh, syn\}}$ S $T_{\{val\}}$, 2 3 2) ⊢

Выполнение операционного символа $\{F_{val} = n_{val}\}$. 2 символа из стека копий – $F_{\{val\}}$ и $n_{\{val=3\}}$ – передаются как аргументы

(* $n_{\{val=5\}}$ $, • $\{T'_{inh} = F_{val}\}$ $T'\{T_{val} = T'_{syn}\} \bullet \{print(T_{val})\} \bullet \perp$, $F_{\{val=3\}}$ $n_{\{val=3\}}$ $T_{\{val\}}$ $F_{\{val=3\}}$ $T'_{\{inh, syn\}}$ S $T_{\{val\}}$, 2 3 2) ⊢

Выброс •. Верхний элемент стека длин – 2, 2 символа удаляются из стека копий, верхний символ удаляется из стека длин.

(* $n_{\{val=5\}}$ $, $\{T'_{inh} = F_{val}\}$ $T'\{T_{val} = T'_{syn}\} \bullet \{print(T_{val})\} \bullet \perp$, $T_{\{val\}}$ $F_{\{val=3\}}$ $T'_{\{inh, syn\}}$ S $T_{\{val\}}$, 3 2) ⊢

Выполнение операционного символа $\{T'_{inh} = F_{val}\}$. 3 символа из стека копий – $T_{\{val\}}$, $F_{\{val=3\}}$ и $T'_{\{inh, syn\}}$ – передаются как аргументы

(* $n_{\{val=5\}}$ $, $T'\{T_{val} = T'_{syn}\} \bullet \{print(T_{val})\} \bullet \perp$, $T_{\{val\}}$ $F_{\{val=3\}}$ $T'_{\{inh=3, syn\}}$ S $T_{\{val\}}$, 3 2) ⊢

Перенос $T'_{\{inh, syn\}} \rightarrow *F_{\{val\}} \{T'_{1.inh} = T'_{inh}*F_{val}\}$ $T'_{1.\{inh, syn\}} \{T'_{syn} = T'_{1.syn}\}$, длина – 4.

(* $n_{\{val=5\}}$ $, * F $\{T'_{1.inh} = T'_{inh}*F_{val}\}$ $T'_1$ $\{T'_{syn} = T'_{1.syn}\} \bullet \{T_{val} = T'_{syn}\} \bullet \{print(T_{val})\} \bullet \perp$, $T'_{\{inh=3, syn\}}$ * $F_{\{val\}}$ $T'_{1.\{inh, syn\}}$ $T_{\{val\}}$ $F_{\{val=3\}}$ $T'_{\{inh=3, syn\}}$ S $T_{\{val\}}$, 4 3 2) ⊢ Выброс

($n_{\{val=5\}}$ $, F $\{T'_{1.inh} = T'_{inh}*F_{val}\}$ $T'_1$ $\{T'_{syn} = T'_{1.syn}\} \bullet \{T_{val} = T'_{syn}\} \bullet \{print(T_{val})\} \bullet \perp$, $T'_{\{inh=3, syn\}}$ * $F_{\{val\}}$ $T'_{1.\{inh, syn\}}$ $T_{\{val\}}$ $F_{\{val=3\}}$ $T'_{\{inh=3, syn\}}$ S $T_{\{val\}}$, 4 3 2) ⊢ Перенос

($n_{\{val=5\}}$ $, n $\{F_{val} = n_{val}\} \bullet \{T'_{1.inh} = T'_{inh}*F_{val}\}$ $T'_1$ $\{T'_{syn} = T'_{1.syn}\} \bullet \{T_{val} = T'_{syn}\} \bullet \{print(T_{val})\} \bullet \perp$, $F_{\{val\}}$ $n_{\{val\}}$ $T'_{\{inh=3, syn\}}$ * $F_{\{val\}}$ $T'_{1.\{inh, syn\}}$ $T_{\{val\}}$ $F_{\{val=3\}}$ $T'_{\{inh=3, syn\}}$ S $T_{\{val\}}$, 2 4 3 2) ⊢ Выброс

($, $\{F_{val} = n_{val}\} \bullet \{T'_{1.inh} = T'_{inh}*F_{val}\}$ $T'_1$ $\{T'_{syn} = T'_{1.syn}\} \bullet \{T_{val} = T'_{syn}\} \bullet \{print(T_{val})\} \bullet \perp$, $F_{\{val\}}$ $n_{\{val=5\}}$ $T'_{\{inh=3, syn\}}$ * $F_{\{val\}}$ $T'_{1.\{inh, syn\}}$ $T_{\{val\}}$ $F_{\{val=3\}}$ $T'_{\{inh=3, syn\}}$ S $T_{\{val\}}$, 2 4 3 2) ⊢ Выполнение

($, • $\{T'_{1.inh} = T'_{inh}*F_{val}\}$ $T'_1$ $\{T'_{syn} = T'_{1.syn}\} \bullet \{T_{val} = T'_{syn}\} \bullet \{print(T_{val})\} \bullet \perp$, $F_{\{val=5\}}$ $n_{\{val=5\}}$ $T'_{\{inh=3, syn\}}$ * $F_{\{val=5\}}$ $T'_{1.\{inh, syn\}}$ $T_{\{val\}}$ $F_{\{val=3\}}$ $T'_{\{inh=3, syn\}}$ S $T_{\{val\}}$, 2 4 3 2) ⊢ Выброс •

($, $\{T'_{1.inh} = T'_{inh}*F_{val}\}$ $T'_1$ $\{T'_{syn} = T'_{1.syn}\} \bullet \{T_{val} = T'_{syn}\} \bullet \{print(T_{val})\} \bullet \perp$, $T'_{\{inh=3, syn\}}$ * $F_{\{val=5\}}$ $T'_{1.\{inh, syn\}}$ $T_{\{val\}}$ $F_{\{val=3\}}$ $T'_{\{inh=3, syn\}}$ S $T_{\{val\}}$, 4 3 2) ⊢ Выполнение

($, $T'_1$ $\{T'_{syn} = T'_{1.syn}\} \bullet \{T_{val} = T'_{syn}\} \bullet \{print(T_{val})\} \bullet \perp$, $T'_{\{inh=3, syn\}}$ * $F_{\{val=5\}}$ $T'_{1.\{inh=15, syn\}}$ $T_{\{val\}}$ $F_{\{val=3\}}$ $T'_{\{inh=3, syn\}}$ S $T_{\{val\}}$, 4 3 2) ⊢ Перенос

($, $\{T'_{syn} = T'_{inh}\} \bullet \{T'_{syn} = T'_{1.syn}\} \bullet \{T_{val} = T'_{syn}\} \bullet \{print(T_{val})\} \bullet \perp$, $T'_{1.\{inh=15, syn\}}$ $T'_{\{inh=3, syn\}}$ * $F_{\{val=5\}}$ $T'_{1.\{inh=15, syn\}}$ $T_{\{val\}}$ $F_{\{val=3\}}$ $T'_{\{inh=3, syn\}}$ S $T_{\{val\}}$, 1 4 3 2) ⊢ Выполнение

($, • $\{T'_{syn} = T'_{1.syn}\} \bullet \{T_{val} = T'_{syn}\} \bullet \{print(T_{val})\} \bullet \perp$, $T'_{1.\{inh=15, syn=15\}}$ $T'_{\{inh=3, syn\}}$ * $F_{\{val=5\}}$ $T'_{1.\{inh=15, syn=15\}}$ $T_{\{val\}}$ $F_{\{val=3\}}$ $T'_{\{inh=3, syn\}}$ S $T_{\{val\}}$, 1 4 3 2) ⊢ Выброс •

($, $\{T'_{syn} = T'_{1.syn}\} \bullet \{T_{val} = T'_{syn}\} \bullet \{print(T_{val})\} \bullet \perp$, $T'_{\{inh=3, syn\}}$ * $F_{\{val=5\}}$ $T'_{1.\{inh=15, syn=15\}}$ $T_{\{val\}}$ $F_{\{val=3\}}$ $T'_{\{inh=3, syn\}}$ S $T_{\{val\}}$, 4 3 2) ⊢ Выполнение

($, • $\{T_{val} = T'_{syn}\} \bullet \{print(T_{val})\} \bullet \perp$, $T'_{\{inh=3, syn=15\}}$ * $F_{\{val=5\}}$ $T'_{1.\{inh=15, syn=15\}}$ $T_{\{val\}}$ $F_{\{val=3\}}$

$T'_{\{inh=3,\ syn=15\}}$ S $T_{\{val\}}$, 4 3 2) ⊢ Выброс •

($, *{$T_{val}$ = $T'_{syn}$}* • *{print($T_{val}$)}* • ⊥, $T_{\{val\}}$ $F_{\{val=3\}}$ $T'_{\{inh=3,\ syn=15\}}$ S $T_{\{val\}}$, 3 2) ⊢ Выполнение

($, • *{print($T_{val}$)}* • ⊥, $T_{\{val=15\}}$ $F_{\{val=3\}}$ $T'_{\{inh=3,\ syn=15\}}$ S $T_{\{val=15\}}$, 3 2) ⊢ Выброс •

($, *{print($T_{val}$)}* • ⊥, S $T_{\{val=15\}}$, 2) ⊢ Выполнение *{print($T_{val}$)}*. 15 – результат вычислений.

($, • ⊥, S $T_{\{val=15\}}$, 2) ⊢ Выброс •

($, ⊥, ε, ε) ⊢ Конечная конфигурация. Допуск

## Другие алгоритмы

**Алгоритм**: Построение АТ грамматики создания дерева разбора

**Вход**: Грамматика G = (T, V, S, P)

**Выход**: Грамматика G' = (T', V', S', P')

// К каждому символу грамматики добавляем атрибут node

T’ = ∅

foreach ($X_A \in T$) {

    A’ = A ∪ {node}

    T’ = T’ ∪ $X_A$,

}

V’ = ∅

foreach ($X_A \in V$) {

    A’ = A ∪ {node}

    V’ = V’ ∪ $X_A$,

}

// Составляем новые правила

P' = ∅

Op’ = ∅

foreach ($A \rightarrow \alpha \in P$) {

    β = ∅

    foreach ($X_i \in \alpha$) {

        // Не добавляем имеющиеся операционные символы

        if ($X_i \notin Op$) {

            β = β$X_i$

        }

    }

    // Добавляем новый операционный символ

    // β = $X_1X_2...X_n$

    β = β {$A_{node}$ = new Node($X_{1.node}$, $X_{2.node}$, …, $X_{n.node}$)}

    Op’ = Op’ ∪ {$A_{node}$ = new Node($X_{1.node}$, $X_{2.node}$, …, $X_{n.node}$)}

    P' = P' ∪ $A \rightarrow$ β

}

// Пополняем грамматику

V' = V ∪ S'$_{\{node\}}$

$P' = P' \cup S' \rightarrow S \ \{S'_{node} = new\ Node(S_{node})\}$

$Op' = Op' \cup \{S'_{node} = new\ Node(S_{node})\}$

$G' = (T', V', Op', S', P')$

Пусть дерево разбора представлено множеством узлов Node, при этом каждый узел имеет поле Value, в котором находится символ грамматики; поле Next, которое содержит список следующих узлов; поле Id, которое содержит номер правила, соответствующее данному символу.

**Алгоритм**: Составление дерева разбора строки с использованием АТ грамматики

**Вход**: Входная строка, исходная грамматика G

**Выход**: Дерево разбора строки

Получаем АТ грамматику G', применяя предыдущий алгоритм к G.

Строим LL-транслятор по АТ грамматике G'.

Передаем входную строку транслятору.

```
Root = S'node // Корень дерева разбора находится в атрибуте S'.node
// Добавляем в дерево разбора оригинальные операционные символы
Stack ← Root // Стек для имитации рекурсивных вызовов
while (Stack не пуст) {
    Node ← Stack
    A → α = P_Node.Id // Правило, соответствующее данному узлу дерева
    // Вставляем операционные символы в нужные места в дереве
    // Удаляем атрибут node
    foreach (Child ∈ Node.Next, X_i ∈ α) {
        if (Child.Value ∈ V) {
            Stack ← Child
        }
        Child.Value_A = Child.Value_(A \ {node})
        if (X_i ∈ Op) {
            Вставляем X_i в новый узел после узла Child
        }
    }
}
```

**Алгоритм**: Выполнение АТ грамматики в процессе обхода дерева разбора

**Вход**: Корень дерева разбора

**Выход**: Результат выполнения АТ грамматики

```
foreach (Child ∈ Node.Next) {
    if (Child.Value ∈ Op) {
        Выполнить функцию Child.Value, передав ей как аргументы символ из
        родительского узла и символы из узлов-братьев
    } else {
        Обработать алгоритмом узел Child
    }
}
```

## Примеры

АТ грамматика осталась той же:

$G = (\{n_{\{val\}}, *\}, \{S, T_{\{val\}}, T'_{\{inh, syn\}}, F_{\{val\}}\}, Op, S, P)$

P:

6. $S \rightarrow T_{\{val\}}$ $\{print(T_{val})\}$
7. $T_{\{val\}} \rightarrow F_{\{val\}}$ $\{T'_{inh} = F_{val}\}$ $T'_{\{inh, syn\}}$ $\{T_{val} = T'_{syn}\}$
8. $T'_{\{inh, syn\}} \rightarrow *F_{\{val\}}$ $\{T'_{1.inh} = T'_{inh} * F_{val}\}$ $T'_{1.\{inh, syn\}}$ $\{T'_{syn} = T'_{1.syn}\}$
9. $T'_{\{inh, syn\}} \rightarrow \varepsilon$ $\{T'_{syn} = T'_{inh}\}$
10. $F_{\{val\}} \rightarrow n_{\{val\}}$ $\{F_{val} = n_{val}\}$

Рассмотрим преобразование данной АТ грамматики в АТ грамматику построения дерева.

После преобразования G, получим следующее (новый атрибут node имеет тип Node, узла дерева):

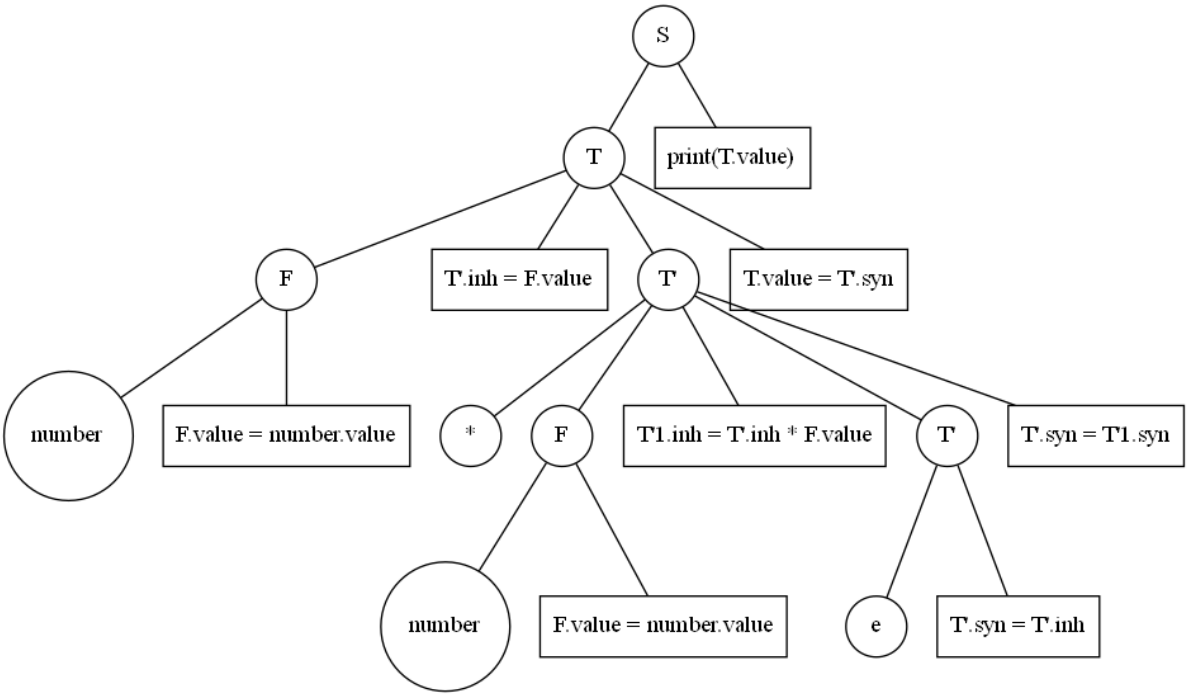
$G' = (\{n_{\{val, node\}}, *_{\{node\}}\},$
$\{S'_{\{node\}}, S_{\{node\}}, T_{\{val, node\}}, T'_{\{inh, syn, node\}}, F_{\{val, node\}}\},$
$S', Op', P')$

P':

1. $S'_{\{node\}} \rightarrow S_{\{node\}}$ $\{S'_{node} = new\ Node(S_{node})\}$
2. $S_{\{node\}} \rightarrow T_{\{val, node\}}$ $\{S_{node} = new\ Node(T_{node})\}$
3. $T_{\{val, node\}} \rightarrow F_{\{val, node\}} T'_{\{inh, syn, node\}}$ $\{T_{node} = new\ Node(F_{node}, T'_{node})\}$
4. $T'_{\{inh, syn, node\}} \rightarrow *_{\{node\}} F_{\{val, node\}} T'_{1\{inh, syn, node\}}$ $\{T'_{node} = new\ Node(*_{node}, F_{node}, T'_{1.node})\}$

5. $T'_{\{inh, syn, node\}} \rightarrow \varepsilon$ $\{T'_{node} = new\ Node(new\ Node(\varepsilon))\}$
6. $F_{\{val, node\}} \rightarrow n_{\{val, node\}}$ $\{F_{node} = new\ Node(n_{node})\}$

В LL-транслятор, построенный по данной АТ грамматике, передадим на вход строку 3*5. На выходе получится дерево:




Если выполнить такое дерево, можно получить аннотированное дерево, по которому видны промежуточные вычисления: