Задание 11

LL- и LR-анализ

Ключевые слова [1]:*язык, контекстно-свободный язык, магазинный автомат, грамматика,* LL(k)*-грамматика,* LL(1)*-анализатор, функции* FIRST, FOLLOW.

1 Нисходящий и восходящий разбор

Напомним определение вывода КС-грамматики.

Выводом цепочки α называется такая последовательность применений правил с указанием раскрываемого нетерминала, что применяя правила из неё начиная с аксиомы получается цепочка α. Если цепочка α не содержит нетерминалов, то α принадлежит языку, порождаемому КС-грамматикой. Нам будет удобно пользоваться такими понятиями как левый вывод (правый вывод). *Левым выводом* называют такой вывод, что на каждом его шаге раскрывается самый левый нетерминал в промежуточной цепочке. Правый вывод определяется аналогично.

Также напомним что мы называем деревом вывода или деревом разбора. С формальным определением дерева разбора вы можете познакомиться, например, в книге Хопкрофта, Мотвани и Ульмана, а мы воспользуемся неформальным описанием этого понятия. *Деревом разбора* для грамматики G называется упорядоченное дерево, в корне которого находится аксиома S, каждая вершина помечена нетерминалом, терминалом или пустым словом, если вершина помечена терминалом или ε, то эта вершина является листом, если же вершина помечена нетерминалом A, то существует такое правило $A \to X_1 X_2 \dots X_n \in P$ ($X_i \in N \cup T$), что вершины-потомки A помечены символами $X_1, X_2 \dots X_n$ слева направо.

Будем говорить, что для КС-грамматики G слово w разобрано, если известно хотя бы одно из её деревьев вывод.

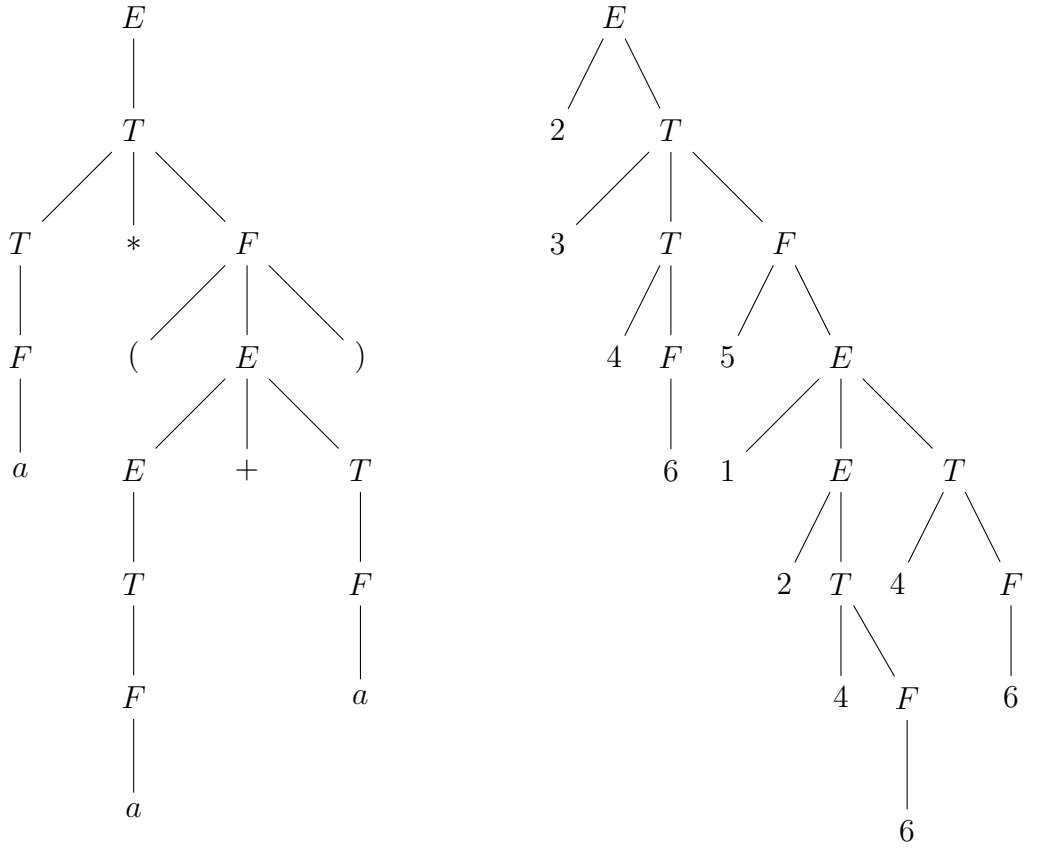
Левым разбором цепочки $\alpha \in (N \cup T)^*$ будем называть последовательность правил, применённых при левом выводе цепочки α. *Правым разбором* цепочки α назовём обратную последовательность правил, применённых при правом выводе цепочки α.

[1] минимальный необходимый объем понятий и навыков по этому разделу)

Пример 1. Грамматики $G = (N, T, P, E)$, и $G_\pi = (N, T', P', E)$, $T = \{a, +, *\}$ заданы правилами:

$E \to E + T$	(1)	$E \to 1ET$
$E \to T$	(2)	$E \to 2T$
$T \to T * F$	(3)	$T \to 3TF$
$T \to F$	(4)	$T \to 4F$
$F \to (E)$	(5)	$F \to 5E$
$F \to a$	(6)	$F \to 6$

Построим дерево разбора для слова $w = a + (a * a)$ и дерево вывода в грамматики G', соответствующее выводу w :



Если грамматика G выводит слово w, то применяя соответствующие правила в G' выводим из неё слово $\pi_l(w)$, соответствующее левому выводу слова w.

Назовём *переводом* бинарное отношение T, действующее из языка L_1 в язык L_2. Если пара слов (u, v) удовлетворяет отношению T, то будем говорить, что слово u транслируется переводом T в слово v, а слово v будем называть *выходом* для u. Мы будем рассматривать синтаксически управляемые переводы. Неформально, перевод является синтаксически управляемым, если существует пара грамматик, правила которых занумерованы и если на шаге вывода из одной грамматики получена цеапочка α, а для соответствующего шага вывода из другой грамматики получена цепочка β, то любой нетерминал входящий в цепочку α, входит и в цепочку β с одинаковой кратностью. Например, в лингвистике нетерминалы могут соответствовать частям речи. Тогда при переводе с одного языка

на другой подлежащее перейдёт в подлежащее, а сказуемое в сказуемое, таким образом, синтаксис определяет некоторые особенности семантики языка. Эта особенность также весьма полезна и при построении компиляторов. Формально, перевод называется синтаксически управляемым, если есть синтаксически управляемая схема (СУ-схема), его реализующая. Определим формально СУ-схему.

Определение 1. Синтаксически управляемой схемой назовём пятёрку $T = (N, \Sigma, \Delta, R, S)$, где N множество нетерминалов, Σ и Δ алфавиты входа и выхода схемы, R – множество правил вида $A \to \alpha, \beta$, причём нетерминалы входящие в цепочку α входят также и в цепочку β, причём с той же кратностью.

Как легко видеть, из языка $L(G)$ существует СУ-перевод в язык $L(G')$, схема которого строится по грамматикам. А именно множество R строится по соответствующим парам правил, описанных выше.

Упражнение 1. Предъявить алгоритм построения по грамматики G синтаксический перевод T_l, переводящий слово w из $L(G)$ в левый вывод данного слова $\pi_l(w)$.

Восходящий разбор строится аналогично по правому выводу.

Упражнение 2. Построить правый вывод $w = a + (a * a)$ по дереву разбора. По правому выводу построить разбор $\pi_r(w)$.

Упражнение 3. Построить по описанной выше грамматике G схему СУ-перевода, реализующую перевод $w \to \pi_r(w)$. Предъявить алгоритм построения схемы данного СУ-перевода по грамматике.

2 От FIRST к FIRST_k

Алгоритмы вычисления FIRST и FOLLOW приведены в предыдущем задании. Основное, что нужно понять из этого раздела — определение функции FIRST_k. Остальное нужно для понимания как перейти от LL(1)-грамматик к LL(k)-грамматикам, а позже — от LR(1)-анализаторов к LR(k)-анализаторам.

Сначала введём вспомогательную операцию на множествах. Пусть L_1 и L_2 некоторые языки. Тогда язык $L_1 \oplus_k L_2$ состоит из всех таких слов w, что либо в языке L_1 есть слово w_1 длины не меньшей k и $w = w_1[1, k]$, либо слово x длины не большей k лежит в L_1, слово y лежит в L_2, слово

u есть их конкатенация xy и, наконец, $w = u[1,k]$, если $|u| > k$ или просто $w = u$, если $|u| < k$. Формально

$$L_1 \oplus_k L_2 = \{w \mid \exists x \in L_1, \exists y \in L_2, u = xy, |u| \leqslant k \Rightarrow w = u, |u| > k \Rightarrow w = u[1,k]\}$$

Другой вариант формального определения, чтобы окончательно запутать читателя:

$$L_1 \oplus_k L_2 = \{xy \mid x \in L_1, y \in L_2, |xy| \leqslant k\} \cup \{u[1,k] \mid \exists x \in L_1, \exists y \in L_2, u = xy, |xy| > k\}$$

Из определения операции $\oplus_k$ следует, что для $X_1, X_2, \ldots X_n \in N \cup T$ справедливо

$$\mathrm{FIRST}_k(X_1 X_2 \ldots X_n) = \mathrm{FIRST}_k(X_1) \oplus_k \mathrm{FIRST}_k(X_2) \oplus_k \ldots \oplus_k \mathrm{FIRST}_k(X_n).$$

Фактически, когда мы вычисляли функцию FIRST, мы вычисляли её используя оператор $\oplus_1$. Перепишем алгоритм вычисления функции FIRST для вычисления функции FIRST_k.

Алгоритм:

Шаг 0. Для каждого терминала σ положим $F_i(\sigma) = \sigma$ для любого i. Для каждого нетерминала Y, рассмотрим все правила вида $Y \to x\alpha$, где x – слово (быть может пустое!), а цепочка α либо начинается с нетермнинала, либо пуста. Если $|x| \geqslant k$, добавим к множеству $F_0(Y)$ слово $x[1,k]$. Иначе, если $\alpha = \varepsilon$, добавим к множеству $F_0(Y)$ слово x.

Шаг i. Добавить к множеству $F_i(X)$ множество $F_{i-1}(X)$. Для каждого правила $X \to \beta = Y_1 \ldots Y_n$

добавить к $F_i(X)$ множество $F_{i-1}(Y_1) \oplus_k \ldots \oplus_k F_{i-1}(Y_n)$,

вычислить $F_i(Y_j)$, для $j = 1..n$

Остановка. $F_i(Y) = F_{i-1}(Y)$ для любого Y из N. Положить $\mathrm{FIRST}_k(X) = F_i(X)$.

Упражнение 4. Доказать корректность работы данного алгоритма.

На практике удобно вычислять функцию FIRST_k для всех нетерминалов сразу.

3 LL(k)-грамматики

Мы не будем строить анализаторы для LL(k)-грамматик, где $k > 1$ в силу нехватки времени. Тем не менее, мы будем работать с определением LL(k)-грамматики и её свойствами.

Вспомним, что грамматика является LL(k)-грамматикой тогда и только тогда, когда она левоанализируема, т.е. существует детерминированный анализатор (ДМП-автомат с выходом), который реализует СУ перевод $w \to \pi_l(w)$.

С таким определением не очень удобно работать с точки зрения анализа грамматики, поэтому мы будем также использовать эквивалентное ему определение.

Теорема 1. Грамматика является LL(k)-грамматикой тогда и только тогда, когда для любых двух правил $A \to \beta, A \to \gamma$, $\mathrm{FIRST}_k(\gamma\alpha) \cap \mathrm{FIRST}_k(\beta\alpha) = \emptyset$ для таких α, что $S \Rightarrow_l^* wA\alpha$.

Не все грамматики, задающие LL-языки являются LL-грамматиками. Но некоторые из них можно преобразовать к LL(k)-грамматике используя такие приёмы как левая факторизация и удаление левой рекурсии. Изучите эти приёмы по книжке Серебрякова или по Ахо и Ульману.

4 Подготовка к изучению LR-анализа

LR-анализаторы устроены гораздо сложнее, чем LL-анализаторы. Их построение можно изучить по книге Серебрякова, а за подробностями обратиться к книге Ахо и Ульмана, оригиналной статье Кнута или тексту на странице курса. В этом разделе мы приведём технические конструкции, нужные для построения LR(1)-анализатора.

Зафиксируем пополненную грамматику G. Назовём LR(k)-ситуацией набор

$$[A \to X_1 X_2 \dots X_i \cdot X_{i+1} \dots X_n, u],$$

где $A \to X_1 \dots X_n$ — правило G, u — слово длины не больше k (параметр LR(k) анализатора) над алфавитом $\Sigma \cup \{\$\}$ (возможно пустое), а «$\cdot$» — вспомогательный символ, который встречается ровно один раз.

Определим недетерминированный LR(k)-автомат Кнута для грамматики G. Множество состояний автомата — LR(k)-ситуации граммати-

ки G. Начальное состояние автомата — ситуация $[S' \to \cdot S, \$]$. Переходы определены следующим образом:

Если состояние — ситуация вида $[A \to \alpha \cdot B\gamma, u]$, то для каждого правила $B \to \beta$ и каждого слова $v \in \mathrm{FIRST}_k(\gamma u)$ определён ε переход

$$[A \to \alpha \cdot B\gamma, u] \xrightarrow{\varepsilon} [B \to \cdot\beta, v]$$

Для состояния (ситуации) $[A \to \alpha \cdot X\gamma, u]$, где $X \in \Sigma \cup N$ определён переход

$$[A \to \alpha \cdot X\gamma, u] \xrightarrow{X} [A \to \alpha X \cdot \gamma, u]$$

Множество принимающих состояний несущественно для дальнейшего использования автомата Кнута, будем считать, что оно пусто.

5 Задачи

В первых двух задачах под грамматикой G понимается грамматика, порождающая арифметические выражения.

Задача 1. Построить дерево вывода, левые и правые разборы для слова $((a))$ в грамматике G, определённой выше.

Задача 2. Постройте LL(1)-анализатор для грамматики Expr из предыдущего задания. Продемонстрируйте его работу на слове $id + id \times id$ и, в случае успеха, постройте дерево разбора по результатам работы аналтзатора.

Задача 3. Докажите, что грамматика не является LL(1)-грамматикой, но является LL(2)-грамматикой. Вычислите функции FIRST_2 и FOLLOW_2 для всех нетерминалов.

$$S \to aAaa|bAba$$
$$A \to b|\varepsilon$$

Задача 4. Написать для грамматики эквивалентную LL(1)-грамматику, построить LL(1)-анализатор и продемонстрировать его работу на слове $baab$.

$$S \to baaA|babA \qquad A \to \varepsilon|Aa|Ab$$

Задача 5. Постройте недетерминированный LR(1)-автомат Кнута для грамматики $S \to aSb \mid a$. Детерминизируйте этот автомат по алгоритму из курса — вы получите детерминированный автомат Кнута, с помощью которого строится управляющая таблица LR(1)-анализатора. При построении автомата Кнута используйте только состояния, достижимые из начального.

Задача 6*. Докажите, что язык $a^* \cup a^n b^n$ не является LL-языком.