Лекция 7. Контекстно-свободные грамматики и языки. Часть 1

Главные вопросы, которые мы обсуждаем, представлены на СЛАЙДЕ 1. Обсудим более широкий класс языков по сравнению РЯ, а именно контекстно-свободные языки. Они имеют естественное рекурсивное описание в виде контекстно-свободных грамматик. Данный класс играет важнейшую роль в теории и технологиях компиляции с конца 1950 – начала 1960 гг. Контекстно-свободные языки также описываются с помощью МПА, которые эквивалентны контекстно-свободным грамматикам. Кроме того, они полезны при изучении свойств контекстно-свободных языков.

7.1 Контекстно-свободные грамматики

По сложившейся традиции начнем с неформального представления контекстно-свободных грамматик (*CFG – Context Free Grammar*, далее – КСГ) и рассмотрим их основные свойства.

Пусть L – это язык палиндромов над алфавитом $\{0, 1\}$, $L = \{ww^R\}$. Применяя лемму о разрастании, можно доказать, что L – не РЯ.

Существует естественное рекурсивное определение того, что строка из символов 0 и 1 принадлежит L. Оно начинается с базиса, утверждающего, что несколько очевидных строк принадлежат L, а затем использует идею о том, что если строка является палиндромом, то она должна начинаться и заканчиваться одним и тем же символом. С другой стороны, при удалении крайних символов остаток строк должен быть палиндромом.

Базис. ε, 0 и 1 являются палиндромами.

Индукция. Если w – палиндром, то $0w0$ и $1w1$ – тоже палиндромы. Не являются палиндромами строки, если они не определяются этими базисом и индукцией.

КСГ представляет собой формальную запись подобных рекурсивных определений языков. КСГ, как и любая грамматика, состоит из переменных (нетерминальных символов), которые представляют собой наборы строк, или языки. Нам нужен всего один нетерминальный символ для представления класса строк языка L. Имеются правила построения строк каждого класса. При построении используются символы алфавита и уже построенные строки из разных классов. СЛАЙД 2.

Пример 42.

Правила построения палиндромов с помощью КСГ приведены на СЛАЙДЕ 3. Первые три правила образуют базис, т.к. ни в одной из RHS этих правил нет нетерминалов. Четвертое и пятое правила образуют индуктивную часть определения. Здесь символ из LHS правила соответствует классу языков, которому принадлежит та или иная строка.

Определение формальной грамматики

Как и любая другая грамматика, КСГ является четверкой элементов $G = (V_T, V_N, P, S)$ с известными нам обозначениями. Специфичным для КСГ является вид правил вывода,

которые еще называются **продукциями**. Каждая продукция в КСГ имеет вид $\alpha \to \beta$, где $|\alpha| = 1$. Иначе говоря, в LHS любой продукции всегда ровно один нетерминал. СЛАЙД 4.

Грамматика для палиндромов имеет вид $G_{pal} = (\{0,1\}, \{P\}, A, P)$. Множество A состоит из 5 продукций, приводившихся ранее на СЛАЙДЕ 3.

Пример 43.

Рассмотрим грамматику сильно упрощенного языка арифметических выражений. Мы ограничимся операторами сложения и умножения и допустим, что операнды могут быть идентификаторами, которые представляют собой строки, начинающиеся с *a* или *b*, за которыми следует произвольная последовательность из $\{a, b, 0, 1\}^*$.

Далее определимся с нетерминалами. Во-первых, нужен символ для обозначения произвольного выражения, обозначим его символом *E*. Он же – аксиома грамматики. Во-вторых, требуется нетерминал для идентификатора. Язык идентификаторов является РЯ, т.к. мы можем описать его РВ $(a+b)(a+b+0+1)^*$, и, следовательно, описать с помощью РГ, а любая такая грамматика, как известно, является КСГ.

Формально грамматика выражений определяется $G = (\{+,*, (,), a, b, 0, 1\}, \{E, I\}, P, E)$. Множество P продукций грамматики приведено на СЛАЙДЕ 5. Круглые скобки заменены на фигурные из-за того, что первые являются встроенным символом в JFLAP.

Первая продукция является базисом для выражений, указывающее на то, что выражение может быть идентификатором. Продукции с 2-й по 4-ю описывают индуктивное образование выражений, причем 2 и 3 продукции говорят, что выражение может состоять из двух выражений. Четвертая продукция – указывает на то, что если выражение заключить в круглые скобки, то результатом тоже будет выражение.

Продукции с 5-й по 10-ю описывают идентификатор, из них 5-ая и 6-ая продукции являются базисом, остальные индуктивно определяют произвольный идентификатор.

Порождения с использованием грамматики

Для того чтобы убедиться, что данные строки принадлежат языку некоторого нетерминала, мы применяем продукции КСГ. Есть два способа действий. В первом случае применяют продукции от RHS к LHS. Далее нужно убедиться, что полученная строка принадлежит к языку нетерминала из LHS. Такая процедура носит название **обратного порождения**, или **рекурсивного вывода**.

При другом способе продукции применяются от LHS к RHS. Аксиома разворачивается с использованием одной из его продукций. Затем разворачивается полученная строка путем замены одного из нетерминалов на RHS одной из его продукций, и этот процесс повторяется до получения строки, состоящей из одних терминалов. Язык грамматики – это все строки, полученные таким образом. Использование грамматики называется **выводом**, или **порождением**. СЛАЙД 6.

Пример 44.

Вывод о том, что строка $a*(a+b00)$ принадлежит языку нетерминала *E*, можно отразить с помощью порождения, начиная от этого символа. Мы можем получить несколько вариантов. Первый:

$E => E * E => I * E => a * E => a * (E) => a * (E + E) => a * (I + E) => a * (a + E) =>$
$=> a * (a + I) => a * (a + I0) => a * (a + I00) => a * (a + b00)$. СЛАЙД 7.

Второй способ в табличном виде представлен на СЛАЙДЕ 8.

Обратное порождение для первого способа (очевидно, что мы просто развернули стрелки, читая справа налево):

$a * (a + b00) <= a * (a + I00) <= a * (a + I0) <= a * (a + I) <= a * (a + E) <=$
$<= a * (I + E) <= a * (E + E) <= a * (E) <= a * E <= I * E <= E * E <= E.$

Версия 0.9pre-release от 20.03.2014. Возможны незначительные изменения.

Иногда используется еще один способ записи обратного порождения:
$E \Rightarrow_R E * E \Rightarrow_R I * E \Rightarrow_R a * E \Rightarrow_R a * (E) \Rightarrow_R a * (E + E) \Rightarrow_R a * (I + E) \Rightarrow_R$
$\Rightarrow_R a * (a + E) \Rightarrow_R a * (a + I) \Rightarrow_R a * (a + I0) \Rightarrow_R a * (a + I00) \Rightarrow_R a * (a + b00)$.
Мы можем использовать сокращенную запись с использованием известного РВ-оператора:
$E \Rightarrow^* a * (a + b00)$ и $E \Rightarrow_R^* a * (a + b00)$. СЛАЙД 9.

На практике часто необходимо найти для строки левое и правое порождение.
Если на каждом шаге порождения мы заменяем крайний левый нетерминал, то такое порождение называется **левым**. Наш первый вариант иллюстрирует левое порождение строки $a * (a + b00)$. Второй вариант порождения этой строки левым не являлся.
Если на каждом шаге порождения мы заменяем крайний правый нетерминал, то такое порождение называется **правым**. СЛАЙД 10.

Пример 45.
Следующее порождение – правое для той же строки, что и в предыдущем примере.
$E \Rightarrow E * E \Rightarrow E * (E) \Rightarrow E * (E + E) \Rightarrow E * (E + I) \Rightarrow E * (E + I0) \Rightarrow E * (E + I00) \Rightarrow$
$\Rightarrow E * (E + b00) \Rightarrow E * (I + b00) \Rightarrow E * (a + b00) \Rightarrow I * (a + b00) \Rightarrow a * (a + b00)$.
Интересно, что количество шагов порождения оказалось одинаковым. Второй вариант порождения из примера 44 не является правым.

Язык, задаваемый грамматикой

Если G – это КСГ, то язык, задаваемый грамматикой G, обозначается $L(G)$ и представляет собой множество терминальных строк, порождаемых из аксиомы. Формально, $L(G) = \left(w \in V_T^* \middle| S \Rightarrow_G^* w\right)$. Если язык задается некоторой КСГ, то он называется **контекстно-свободным языком** (далее – КСЯ).
Например, грамматика из примера 42 была КСГ, которая определяла множество палиндромов из 0 и 1. Следовательно, это множество является КСЯ. Оставим в качестве самостоятельного упражнения доказательство необходимости и достаточности этого утверждения.
Любую строку, которая участвует на каждом шаге порождения, мы называем **сентенциальной формой** (далее – СФ). Очевидно, что КСЯ представляет собой множеств СФ, состоящих исключительно из терминальных символов. СЛАЙД 11.

7.2 Деревья разбора

Для порождений существует полезное представление в древовидной форме, которая наглядно показывает, как символы строки группируются в подстроки, каждая из которых принадлежит языку одного из нетерминалов грамматики. Это дерево известно под названиями **дерева вывода**, или **дерева разбора** (*parse tree*), или **синтаксического дерева**, и является в теории компиляции основной структурой данных для представления исходной программы. За счет этого облегчается трансляция программы при использовании рекурсивных функций.
Пусть G – это КСГ. Деревья разбора для G – это деревья со следующими свойствами (СЛАЙД 12).
0. Корень помечается аксиомой грамматики.
1. Каждый внутренний узел помечен нетерминалом из V_N.
2. Каждый лист отмечен либо нетерминалом, либо терминалом, либо ε. В последнем случае лист должен быть единственным потомком родительского узла.
3. Если внутренний узел помечен A, и его прямые потомки отмечены слева направо $X_1, \ldots, X_k$, то $A \to X_1, \ldots, X_k$ является продукцией из P. Если X отмечает единственного потомка, и $A \to \varepsilon$, то X только в этом случае может быть ε.

Пример 46.

На СЛАЙДЕ 13 показано дерево разбора строки из примера 44. Также там показано дерево разбора строки 0110 из примера 42.

Если мы прочитаем отметки всех листьев дерева разбора, то получим строку, которая называется **кроной дерева**, и всегда является строкой, выводимой из корня (аксиомы).

Если крона является терминальной строкой, т.е. все листья дерева отмечены терминалами или ε, то все такие кроны представляют собой строки языка рассматриваемой грамматики.

В специальной литературе представлены доказательства равносильности обратных и прямых порождений и деревьев разбора.

7.3 Неоднозначность в грамматиках и языках

Наша грамматика из примера 43 дает возможность порождать выражения с любой последовательностью операторов сложения и умножения, в продукции *E->E+E* и *E->E*E* позволяют порождать эти выражения в произвольном порядке.

Пример 47.

Для СФ *E + E * E* можно получить два порождения:

1. *E => E+E => E+E*E*
2. *E => E*E => E+E*E*

Нетрудно построить два различных дерева разбора этой строки (СЛАЙД 14) , хотя существование нескольких порождений для некоторой строки неравносильно нескольким деревьям разбора. Смысл этого различия кроется в порядке выполнения арифметических операций. Первое порождение задает, что *a*1 + *a*2 * *a*3 группируется как *a*1 + (*a*2 * *a*3), что соответствует привычному для нас пониманию группирования арифметических выражений. Второе же группирует заданное выражение (*a*1 + *a*2) * *a*3.

Однако само по себе существование различных порождений строк еще не означает, что грамматика «плохая».

Пример 48.

Используя грамматику выражений, можно убедиться, что для строки *a + b* есть несколько порождений, например, два следующих.

1. *E => E+E => I+E => a+E => a+I => a+b*
2. *E => E+E => E+I => I+I => I+b => a+b*

Существенной разницы между этими порождениями нет. Очевидно, что выражение состоит из двух идентификаторов, и что их нужно сложить. На самом деле оба порождения приводят к одному и тому же дереву разбора, но в детали такого «приведения» мы оставляем на самостоятельное изучение.

Иными словами, неоднозначность происходит не от количества порождений, а от существования двух или более деревьев разбора. Мы говорим, что КСГ является **неоднозначной**, если найдется хотя бы одна строка *w* из V_T^*, для которой существуют два дерева разбора с корнем *S* и кроной *w*. Если каждая строка имеет не более одного дерева разбора в грамматике, то она **однозначна**. СЛАЙД 15.

На СЛАЙДЕ 16 показаны два дерева разбора для строки *a+a*a*.

Исключение неоднозначности

Не существует алгоритма проверки КСГ на однозначность. Кроме того, существуют КСЯ, имеющие только неоднозначные КСГ, из которых вообще невозможно исключение неоднозначности.

Для многих конструкций в практически используемых языках программирования существуют приемы устранения неоднозначностей. Мы проиллюстрируем их на примере грамматики выражений. В ней существуют две причины неоднозначностей.

1. Не учитываются приоритеты операций. Нам необходимо обеспечить такую грамматику, которая группирует умножение перед сложением.

2. Последовательность одинаковых операций может группироваться и слева, и справа. По текущей грамматике мы можем получить два разных дерева разбора СФ $E + E + E$. Понятно, что обе эти операции ассоциативны, и не имеет значения, в каком направлении они группируются. Однако нам надо выбрать что-то одно. Более привычной выглядит группировка слева.

Решение первой проблемы заключается в том, что вводятся дополнительные нетерминалы, каждый из которых представляет выражения одинакового уровня. В нашем случае это решение разбивается на несколько пунктов. СЛАЙД 17.

1. **Сомножитель**, или **фактор** (*factor*) – это выражение, которое не может быть разделено ни одной из операций. Сомножителями в нашем языке являются только идентификаторы и выражения в скобках. Обозначим сомножитель символом F.

2. **Слагаемое**, или **терм** (*term*) – это выражение, которое не может быть разделено операцией сложения. Термом в нашем языке является только произведение одного или нескольких факторов. Обозначим терм символом T.

3. **Выражение** (*expression*) – это любые выражения, которые могут быть разделены знаками операций. Выражением в нашем языке является сумма одного или нескольких термов. Оставим для выражения обозначение символом E.

На СЛАЙДЕ 18 приведена полученная КСГ для выражений, при этом она является однозначной. Можно посмотреть дерево разбора на строке $a+a*a$.

Левые порождения как выражение неоднозначностей

Хотя порождения, как мы отметили, не обязательно уникальны, даже если грамматика однозначна, оказывается, что в однозначной грамматике и левые, и правые порождения уникальны. Мы рассмотрим только левые.

Пример 49.

Оба дерева разбора на СЛАЙДЕ 16 имеют одинаковую крону. По ним получаются два левых порождения. СЛАЙД 19.

1. $E => E+E => I+E => a+E => a+E*E => a+I*E => a+a*E => a+a*I => a+a*a$

2. $E => E*E => E+E*E => I+E*E => a+E*E => a+I*E => a+a*E => a+a*I => a+a*a$

Видно, что эти левые порождения различаются. Тем самым мы продемонстрировали, как различия в деревьях влияют на выбор шагов в левом порождении.

Теорема 7.1. Для каждой КСГ G и w из V_T^* строка w имеет два дерева разбора тогда и только тогда, когда w имеет два разных левых порождения.

Доказательство. Для доказательства необходимости нужно обратить внимание на то, что если у двух деревьев разбора впервые появляется узел, в котором применяются различные продукции, то левые порождения, которые строятся, также используют разные продукции и, следовательно, являются различными.

Идея построения дерева разбора по левому порождению проста. Оно начинается с корня, обозначенного аксиомой КСГ. На каждом шаге заменяется нетерминал, который будет соответствовать построенному крайнему слева узлу дерева без потомков, но отмеченному этим символом. По используемой на этом шаге продукции мы определяем, какие потомки должны быть у этого узла. Если существуют два разных порождения, то на первом шаге, где они различаются, построенные узлы получат разные списки потомков, что гарантирует различие деревьев разбора. Это от них и требовалось.

Существенная неоднозначность

КСЯ L называется **существенно неоднозначным**, если все его грамматики неоднозначны. Если хотя бы одна грамматика однозначна, то и КСЯ тоже однозначен. В частности, язык выражений из предыдущих примеров однозначен, т.к. нам удалось построить для него однозначную грамматику.

Рассмотрим один язык, неоднозначность которого можно доказать, хотя мы это сделаем неформальным способом. СЛАЙД 20.

$$L = \left\{a^n b^n c^m d^m \middle| n \geq 1, m \geq 1\right\} \cup \left\{a^n b^m c^m d^n \middle\| n \geq 1, m \geq 1\right\}.$$

Этот язык представляет собой множество строк вида $aa^*bb^*cc^*dd^*$, в которых символов a и b поровну, за ними идет равное количество c и d, либо символов a и d поровну, а также b и c. Язык L является КСЯ, его грамматика приведена на СЛАЙДЕ 21. Для порождения разных видов строк в ней используются два разных подмножества продукций.

Эта КСГ неоднозначна. Например, у строки *aabbccdd* есть два левых порождения. Одно из них представлено в табличном виде на СЛАЙДЕ 22.

Второе выглядит следующим образом:

S=>C=>aCd=>aaDdd=>aabDcdd=>aabbccdd.

Соответствующие деревья разбора показаны на СЛАЙДЕ 23.

Доказательство того, что все КСГ для языка L неоднозначны, чрезвычайно сложно. Нужно обосновать, что все строки, за исключением конечного их числа, у которых поровну всех символов, порождаются двумя разными путями. Первый путь – порождение их как строка с равным количеством a и b, а также c и d. Второй – как строка, где поровну a и d, а также b и c.

Единственный способ породить строки с равным количеством a и b, заключается в использовании нетерминала A. Так или иначе, но нам не избежать некоторого способа порождения символов a, при котором наблюдается соответствие с символами b.

Аналогично, для нетерминала B, который порождает соответствующие друг другу символы c и d. Кроме того, в КСГ должны присутствовать нетерминалы C и D, порождающие парные a и d, а также b и c. Если эти аргументы формализовать, то окажется, что независимо от изменений, которые можно внести в исходную грамматику, она будет порождать, по меньшей мере, одну строку вида $a^n b^n c^n d^n$ двумя способами, как мы показали выше.

7.4 Эквивалентность автоматов с магазинной памятью и контекстно-свободных грамматик

Как известно, языки, допускаемые МПА по заключительному состоянию и по пустому стеку, эквивалентны. Покажем, что языки, определяемые КСГ, эквивалентны языкам МПА по пустому стеку.

Пусть по грамматике G строится МПА, имитирующий ее левые порождения. Любую СФ, порождаемую таким образом, которая не является терминальной, можно записать в виде $xA\alpha$, где A – крайний левый нетерминал, x – строка терминалов слева от A, α – строка терминалов и нетерминалов справа. $A\alpha$ называется **остатком**, или **хвостом** (*tail*) этой СФ. У терминальной СФ остатком, очевидно, является ε.

МПА имитирует последовательность «левопорождаемых» СФ, используемых в грамматике для порождения терминальной СФ w. Хвост $A\alpha$ появляется в магазине с нетерминалом A на вершине. При этом x представляется прочитанными на входе символами, а суффикс СФ w после x еще не прочитан.

Пусть МПА находится в конфигурации $(q, y, A\alpha)$, представляющей нашу строку $xA\alpha$. Он «угадывает» продукцию, используемую для расширения A. Допустим, это продукция $A \to \beta$. Переход МПА состоит в том, что A на вершине магазина заменяется на β, тем самым достигается МО $(q, y, \beta\alpha)$. У этого МПА всего одно состояние q.

Теперь (q, y, $\beta\alpha$) может не оказаться представленной следующей левопорождаемой СФ, поскольку у β может быть терминальный префикс. Более того, β может вообще не иметь нетерминалов, а у α может оказаться терминальный префикс. Все терминалы требуется удалить из начала СФ $\beta\alpha$ до появления следующего нетерминала в магазине. Эти терминалы сравниваются со следующими входными символами для того, чтобы убедиться в правильности наших предположений о левом порождении входной строки w. Если предположение неправильно, соответствующая ветвь вычислений МПА отбрасывается.

Если таким образом нам удается угадать левое порождение w, то рано или поздно мы дойдем до левопорождаемой строки w. В этот момент все нетерминалы в магазине расширены, и все терминалы совпали с входными символами. Магазин опустошен, и МПА допускает по пустому магазину. СЛАЙД 24.

Более формально, пусть $G = (V_T, V_N, Q, S)$ – КСГ. Построим МПА $P = (\{q\}, V_T, V_T \cup V_N, \delta, q, Z_0, S)$, который допускает $L(G)$ по пустому магазину. Функция переходов определена так (СЛАЙД 25):

1. $\delta(q, \varepsilon, A) = \{(q, \beta) \mid A \to \beta$ – продукция $G\}$ для всех нетерминалов A.
2. $\delta(q, a, a) = \{(q, \varepsilon)\}$ для всех терминалов a.

Пример 50.

Преобразуем КСГ арифметических выражений из примера 43 в МПА. Множеством входных символов для МПА является {a, b, 0, 1, (,), +, *}. Объединив эти символы с нетерминалами E и I, мы получим магазинный алфавит. Функция переходов определена следующим образом (СЛАЙД 26).

а) $\delta(q, \varepsilon, I) = \{(q, a), (q, b), (q, Ia), (q, Ib), (q, I0), (q, I1)\}$;

б) $\delta(q, \varepsilon, E) = \{(q, I), (q, E+E), (q, E*E), (q, (E))\}$;

в) $\delta(q, a, a) = \delta(q, b, b) = \delta(q, 0, 0) = \delta(q, 1, 1) = \delta(q, (, () = \delta(q,),)) =$
$= \delta(q, +, +) = \delta(q, *, *) = \{(q, \varepsilon)\}$.

Пункты а) и б) появились у нас по правилу 1, а переходы в) – по правилу 2. Других переходов у нашего автомата нет.

Примем без доказательства следующую теорему (СЛАЙД 27).

Теорема 7.2. Если МПА P построен по КСГ G в соответствии с описанной выше конструкцией, то $N(P) = L(G)$.

Покажем, что для любого МПА P найдется КСГ, язык которой совпадает с языком, допускаемым P по пустому магазину. Идея базируется на простой мысли, что выталкивание одного символа из магазина вместе с прочтением некоторого входа является основным событием во время работы МПА. При выталкивании МПА может изменять свое состояние, поэтому нам придется учитывать состояние, которое достигает автомат, когда стек опустошается.

Наша конструкция эквивалентной КСГ использует переменные, каждая из которых представляет событие, состоящее в следующем:

1. Окончательное удаление элемента X из магазина.
2. Изменение состояния, скажем, с p на q, когда X заменяется на ε в магазине.

Переменную обозначим составным символом $[pXq$.

Примем без доказательства следующую теорему.

Теорема 7.3. Пусть P – МПА. Тогда существует КСГ G, для которой $L(G) = N(P)$.

Пример 51.

Преобразуем в грамматику МПА для языка *if-else* из примера 39 (см. предыдущий раздел лекционного курса). Его формальное определение имеет вид (СЛАЙД 28):

Версия 0.9pre-release от 20.03.2014. Возможны незначительные изменения.

$$P_N = \left(\{q_0\}, \{i, e\}, \{X\}, \delta_N, q_0, Z_0, \varnothing\right)$$
$$\delta_N\left(q_0, i, X\right) = \left\{(q_0, XX)\right\}$$
$$\delta_N\left(q_0, e, X\right) = \left\{(q_0, \varepsilon)\right\}$$

Напомним, что этот МПА допускает строки, нарушающие правило, что каждое *e* должно соответствовать некоторому предшествующему *i*. У МПА одно состояние и один магазинный символ, то КСГ строится достаточно просто (СЛАЙД 29). В ней только два нетерминала.

а) *S* – аксиома.

б) [*qXq*] – единственная тройка, которую можно собрать из состояний и магазинных символов нашего МПА.

Грамматика имеет следующие продукции.

1. Единственная продукция для аксиомы *S* -> [*qXq*]. Если бы у МПА было *n* состояний, то было бы *n* продукций такого вида, т.к. последним может быть любое из них. Первое состояние должно быть начальным, а магазинный символ – аксиомой, как в нашей первой продукции.

2. Вследствие того, что $\delta_N(q, i, X)$ содержит (*q*, *XX*), получаем продукцию [*qXq*] -> *i*[*qXq*][*qXq*]. Если бы у МПА было *n* состояний, то одно такое правило порождало бы n^2 продукций, т.к. промежуточным состоянием в RHS, а также последним состоянием в LHS и RHS продукций могли быть любые два состояния. Иначе говоря, если бы *r* и *p* были двумя произвольными состояниями, то создавалась бы продукция вида [*qXq*] -> *i*[*qXr*] [*rXp*].

3. Из того, что $\delta_N(q, e, X)$ содержит (*q*, *ε*), получаем продукцию продукцию [*qXq*] -> *e*. В этом случае список магазинных символов, которыми заменяется *X*, пуст, поэтому единственный символ в RHS – это входной символ, приводящий к этому переходу.

Удобно заменить сложносоставной символ [*qXq*] на один «простой», который еще не использовался ни в одной из LHS, например, *A*. Тогда КСГ будет состоять из продукций

S->*A*, *A*->*iAA*, *A*->*e*.

И поскольку *S* и *A* порождают одни и те же строки, то можно их объявить одинаково и записать следующие продукции КСГ:

S->*iSS*, *S*->*e*.

7.5 Детерминированные автоматы с магазинной памятью и контекстно-свободные грамматики

Мы видели, что ДМПА могут допускать языки вроде L_{wbwr}, которые не являются РЯ. Для того чтобы убедиться в нерегулярности, мы можем использовать лемму о разрастании. Пусть *n* – константа из леммы. Рассмотрим строку $w = 0^n b 0^n$. Если ее наращивать, то изменяется длина первой группы символов 0, и получаются строки, у которых «центральный маркер» расположен не в центре. Эти строки не принадлежат L_{wbwr}, и мы пришли к противоречию. Значит, этот язык нерегулярен.

Существуют КСЯ вроде L_{wwr}, которые не могут допускаться по заключительному состоянию никаким ДМПА. Интуитивное доказательство сравнительно просто. Если *P* – ДМПА, допускающий язык L_{wwr}, то при чтении последовательности символов 0 он должен вталкивать их в магазин или нечто другое, что может подсчитать их количество. Можно записывать один *Y* для каждых 00 на входе и использовать состояние для запоминания четности/нечетности количества символов 0.

Пусть *P* прочитал *n* символов 0, а затем увидел на входе 110^n. Он должен проверить, что после пары символов 11 находятся *n* символов 0. Для этого он должен опустошить свой магазин. Теперь он прочитал строку 0^n110^n. Если далее он видит такую же строку, то он должен ее принимать, т.к. весь вход имеет вид ww^R, где $w = 0^n110^n$. Однако если *P* видит вход 0^m110^m, где $m \neq n$, он должен ее **отвергнуть**. Т.к. его магазин пуст, он не

помнит, каким было значение n, и не способен допустить L_{wwr}.

Резюме: языки, допускаемые ДМПА по заключительному состоянию, включают РЯ как собственное подмножество, но сами образуют собственное подмножество КСЯ.

Мощность ДМПА можно уточнить, заметив, что все языки, допускаемые ими, имеют однозначные грамматики. Однако класс языков ДМПА не совпадает с существенно неоднозначными КСЯ. В частности, L_{wwr} имеет однозначную КСГ (S->0S0, S->1S1, S->ε), не являясь при этом языком ДМПА. СЛАЙД 30.

Следующей теоремой мы можем уточнить наше «резюме», сделанное выше. СЛАЙД 31.

Теорема 7.4. Если $L = N(P)$ для ДМПА P, то L имеет однозначную КСГ.

Без доказательства.

Надо отметить, что языки, допускаемые ДМПА по заключительному состоянию, имеют однозначные грамматики. Но мы знаем, как строятся КСГ из МПА, допускающих по пустому магазину. Значит, нам придется изменить язык, чтобы он обладал префиксным свойством, а затем модифицировать КСГ, чтобы она порождала исходный язык. Мы можем обеспечить это, например, с помощью «концевого маркера».

Теорема 7.5. Если $L = L(P)$ для ДМПА P, то L имеет однозначную КСГ.

Без доказательства.

Литература к лекции 7

1. Автомат с магазинной памятью - http://ru.wikipedia.org/wiki/Автомат_с_магазинной_памятью
2. Введение в схемы, автоматы и алгоритмы - http://www.intuit.ru/studies/courses/1030/205/info
3. Гилл, А. Введение в теорию конечных автоматов / А. Гилл. – М.: Наука, 1966. – 272 с.
4. Контекстно-свободная грамматика - http://ru.wikipedia.org/wiki/Контекстно-свободная_грамматика
5. Короткова, М.А. Математическая теория автоматов. Учебное пособие / М.А. Короткова. – М.: МИФИ, 2008. – 116 с.
6. Кузнецов, А.С. Теория вычислительных процессов [Текст] : учеб. пособие / А. С. Кузнецов, М. А. Русаков, Р. Ю. Царев ; Сиб. федерал. ун-т. - Красноярск: ИПК СФУ, 2008. – 184 с.
7. Молчанов, А. Ю. Системное программное обеспечение. 3-е изд. / А.Ю. Молчанов. – СПб.: Питер, 2010. – 400 с.
8. Серебряков В. А., Галочкин М. П., Гончар Д. Р., Фуругян М. Г. Теория и реализация языков программирования — М.: МЗ-Пресс, 2006 г., 2-е изд. - http://trpl7.ru/t-books/TRYAP_BOOK_Details.htm
9. Теория автоматов / Э. А. Якубайтис, В. О. Васюкевич, А. Ю. Гобземис, Н. Е. Зазнова, А. А. Курмит, А. А. Лоренц, А. Ф. Петренко, В. П. Чапенко // Теория вероятностей. Математическая статистика. Теоретическая кибернетика. — М.: ВИНИТИ, 1976. — Т. 13. — С. 109–188. — URL http://www.mathnet.ru/php/getFT.phtml?jrnid=intv&paperid=28&what=fullt&option_lang=rus