информатика

УДК 519.682.1 +681.142.2

Мартыненко Борис Константинович

РЕГУЛЯРНЫЕ ЯЗЫКИ И КС-ГРАММАТИКИ

Аннотация

Существует множество технологических средств построения анализаторов формальных языков, используемых при создании разного вида трансляторов языков программирования. Все они, в конечном счёте, основываются на КС-грамматиках с теми или иными ограничениями, частным случаем которых являются регулярные (автоматные) грамматики. Как правило, эти технологические средства обеспечивают лишь проверку соответствующих требований, предъявляемых к грамматике, и выдачу диагностических сообщений об их нарушениях, тогда как существует множество способов эквивалентных преобразований КС-грамматик, которые могут быть выполнены автоматически и дать грамматики, удовлетворяющие требованиям метода анализа.

Цель этой статьи – описать способ исключения несамовставленных нетерминалов из КС-грамматик за счёт введения регулярных выражений в правые части правил грамматики. В предельном случае такая преобразованная грамматика включает единственное правило для начального нетерминала.

Ключевые слова: КС-грамматика, регулярное выражение, эквивалентное преобразование грамматики.

1. ВВЕДЕНИЕ

Впервые КС-грамматики были использованы для описания синтаксиса алгоритмических языков в [1]. Формализм металингвистических формул, названный позднее BNF-формой, в [2] был дополнен возможностью использовать регулярные выражения над терминалами, нетерминалами в правых частях правил и получил ещё одну букву в обозначении класса грамматик RBNF.

В 1970-х годах RBNF-грамматики были использованы в проекте реализации языка программирования Алгол 68 [3]. Несколько позже [4] в эту модель описания языков было введено понятие операционной среды, а в правила грамматики – контекстные символы (семантики и резольверы) с их интерпретацией в виде семантических процедур и предикатных функций. Это нововведение дало возможность использовать RBNF-грамматики в качестве средства описания трансляций, учитывающего контекст конструкций входного языка трансляции. Эта модель описания и реализации трансляций использовалась при реализации ряда нетривиальных языков программирования, таких как Алгол 68 и Ада. Несколько позже она была воплощена в технологическом комплексе SYNTAX, который в течение ряда лет использовался в семинарских занятиях по технологии трансляции на математико-механическом факультете Санкт-Петербургского университета.

ТК SYNTAX включает средства эквивалентных преобразований на уровне граф-схем за счёт перевода несамовставленных нетерминалов в разряд вспомогательных понятий, определения которых подставляются вместо

применённых вхождений таких символов в правые части правил (метаподстановки). В остальном, ТК SYNTAX требует грамматик, уже приведённых к нужному виду, так же как большинство известных технологических систем генерации анализаторов.

К настоящему времени в теории формальных языков накоплено достаточно способов эквивалентных преобразований грамматик. Применить их для автоматического предварительного приведения КС-грамматик в соответствие с требованиями инструментальных систем построения анализаторов представляется актуальной и реальной задачей.

2. НЕКОТОРЫЕ ПРЕДВАРИТЕЛЬНЫЕ ЭКВИВАЛЕНТНЫЕ ПРЕОБРАЗОВАНИЯ КС-ГРАММАТИК

Считая, что основные понятия и факты теории формальных языков известны читателю, необходимые определения, алгоритмы и утверждения, на которых они основаны, а также соответствующие обозначения, будем напоминать по ходу изложения.

Пусть $G = (V_N, V_T, P, S)$ – КС-грамматика, где V_N – конечное множество нетерминалов, V_T – конечное множество терминалов ($V_N \cap V_T = \varnothing$), P – конечное множество правил вида $A \to \alpha$, $A \in V_N$, $\alpha \in V^*$. Здесь V^* – бесконечное множество всех цепочек над алфавитом $V = V_N \cup V_T$, включая и пустую цепочку, обозначаемую символом ε.

Пусть $x = \gamma_1 A \gamma_2$, где $\gamma_1, \gamma_2 \in V^*$, и $A \to \alpha \in P$. Считается, что из цепочки x непосредственно выводится цепочка y посредством указанного правила $x \Rightarrow y$, если $y = \gamma_1 \alpha \gamma_2$.

Будем использовать обозначение $x \overset{*}{\Rightarrow} y$ для вывода цепочки y из цепочки x за счёт использования нескольких правил грамматики, при этом считается, что любая цепочка x выводима сама из себя, и для этого не требуются никакие правила.

Тогда $L(G) = \{w \mid w \in V_T^*,\ S \overset{*}{\underset{G}{\Rightarrow}} w\}$ – язык, порождаемый грамматикой G.

Далее рассматриваются все эквивалентные преобразования КС-грамматик регулярного языка, предшествующие получению регулярного выражения, его представляющего.

1) ПРОДУКТИВНОСТЬ НЕТЕРМИНАЛОВ

Пусть $G = (V_N, V_T, P, S)$ – произвольная КС-грамматика. Нетерминал $A \in V_N$ называется *продуктивным*, если существует вывод вида $A \overset{*}{\underset{G}{\Rightarrow}} w$, где $w \in V_T^*$.

Известно, что *любой КС-язык может быть порождён КС-грамматикой, каждый нетерминал которой продуктивен*.

Для получения такой грамматики используется следующий вспомогательный алгоритм.

Алгоритм 1. Проверка пустоты языка $L(G)$

Вход: $G = (V_N, V_T, P, S)$ – КС-грамматика.

Выход: $L(G) = \varnothing$.

Метод:

Шаг 1. Начать построение коллекции деревьев вывода с единственного дерева, представленного только корнем – узлом с меткой S.

Шаг 2. Добавлять к этой коллекции любое дерево, которое может быть получено из дерева, уже имеющегося в коллекции, посредством применения одного правила, при условии, что образующееся дерево не имеет ни одной ветви длиннее, чем число нетерминалов данной грамматики.

Шаг 3. Если в построенной коллекции есть хотя бы одно дерево, представляющее вывод терминальной цепочки, то язык $L(G)$ не пуст. Иначе – язык $L(G)$ пуст.

Алгоритм 2. Исключение непродуктивных нетерминалов

Вход: $G = (V_N, V_T, P, S)$ – КС-грамматика.

Выход: $G_1 = (V_{N_1}, V_T, P_1, S)$, причём $L(G_1) = L(G)$.

Метод: Для каждого нетерминала $A \in V_N$ рассмотрим грамматику $G_A = (V_N, V_T, P, A)$.

Если язык $L(G_A)$ пуст, то мы удалим A из множества V_N, а также все правила, использующие A в правой или левой части.

После удаления из G всех таких нетерминалов и правил мы получим новую грамматику $G_1 = (V_{N_1}, V_T, P_1, S)$, где V_{N_1} и P_1 – оставшиеся нетерминалы и правила.

II) ДОСТИЖИМОСТЬ НЕТЕРМИНАЛОВ

Говорят, что нетерминал $A \in V_N$ *достижим*, если существует вывод вида $S \overset{*}{\underset{G}{\Rightarrow}} \alpha_1 A \alpha_2$ для некоторых цепочек $\alpha_1, \alpha_2 \in V^*$.

Известно, что *любой КС-язык может быть порождён КС-грамматикой, каждый нетерминал которой достижим*. Для получения такой грамматики можно использовать следующий алгоритм.

Алгоритм 3. Исключение недостижимых нетерминалов

Вход: $G_1 = (V_{N_1}, V_T, P_1, S)$ – КС-грамматика, все нетерминалы которой продуктивны.

Выход: $G_2 = (V_{N_2}, V_T, P_2, S)$ – эквивалентная КС-грамматика, все нетерминалы которой используются в выводах цепочек языка.

Метод: Мы можем эффективно построить множество V_{N_2} всех нетерминалов A, таких что существует вывод вида $S \overset{*}{\underset{G}{\Rightarrow}} \alpha_1 A \alpha_2$, где $\alpha_1, \alpha_2 \in V^*$.

Для начала поместим S в искомое множество. Затем последовательно будем добавлять к этому множеству любой нетерминал, который появляется в правой части любого правила из P_1, определяющего нетерминал, уже имеющийся в этом множестве.

Процесс завершается, когда никакие новые элементы не могут быть добавлены к упомянутому множеству.

Положим $G_2 = (V_{N_2}, V_T, P_2, S)$, где P_2 – множество правил, оставшихся после исключения всех правил из P_1, которые используют символы из $V_{N_1} \setminus V_{N_2}$ в левых или правых частях правил. Известно, что $L(G_1) = L(G_2)$.

Грамматики, в которых все нетерминалы продуктивные и достижимые, называются *приведёнными*. Они хороши тем, что не содержат нетерминалов и правил, не участвующих в порождении данного языка.

III) ЦЕПНЫЕ ПРАВИЛА

Известно, что *любой контекстно-свободный язык может быть порожден контекстно-свободной грамматикой, не содержащей цепных правил, то есть правил вида* $A \to B$, *где* A *и* B *– нетерминалы.*

Алгоритм 4. Исключение цепных правил

Вход: $G_2 = (V_{N_2}, V_T, P_2, S)$ – приведённая КС-грамматика.

Выход: $G_3 = (V_{N_3}, V_T, P_3, S)$ – КС-грамматика без цепных правил, эквивалентная исходной.

Метод: Мы построим новое множество правил P_3, прежде всего включив в него все нецепные правила из P_2. Затем из каждого цепного правила вида $A \to B \in P_2$ образуем множество новых нецепных правил вида $A \to \alpha$, если существует нецепное правило $B \to \alpha \in P_2$, и пополним множество P_3 новыми нецепными правилами, полученными таким путём.

Вывод одной и той же цепочки в грамматике P_3 короче вывода в грамматике P_2 на число применённых цепных правил из P_2.

IV) СВОЙСТВО САМОВСТАВЛЕННОСТИ

Пусть $G = (V_N, V_T, P, S)$ – КС-грамматика. Символ $A \in V_N$ называется *самовставленным*, если существует вывод вида $A \overset{*}{\underset{G}{\Rightarrow}} \alpha_1 A \alpha_2$, где $\alpha_1 \neq \varepsilon$ и $\alpha_2 \neq \varepsilon$ в предположении, что в грамматике не существует ε-правил, и, кроме того, если $\varepsilon \in L(G)$, то S не должен встречаться в правой части никакого правила, и пустое предложение порождается единственным правилом вида $S \to \varepsilon$.

Последнее требование можно легко удовлетворить эквивалентным преобразованием грамматики с помощью следующего алгоритма.

Алгоритм 5. Исключение начального нетерминала S из правых частей правил

Вход: $G = (V_N, V_T, P, S)$ – КС-грамматика.

Выход: $G' = (V'_N, V_T, P', S')$ – КС-грамматика, эквивалентная исходной.

Метод: Пусть S' – символ, не принадлежащий ни алфавиту нетерминалов, ни алфавиту терминалов.

Положим $G' = (V_N \cup \{S'\}, V_T, P', S')$, где $P' = P / \{S \to \varepsilon\} \cup \{S' \to \alpha \mid S \to \alpha \in P\}$.

3. ПОЛУЧЕНИЕ РЕГУЛЯРНОГО ВЫРАЖЕНИЯ ПО КС-ГРАММАТИКЕ

Известно, что *любой контекстно-свободный язык, порождаемый КС-грамматикой без самовставлений, является регулярным множеством.*

По следствию из теоремы С.К. Клини [5], если язык регулярный, то он может быть представлен регулярным выражением, то есть конечной формулой с терминальными цепочками в качестве операндов и операциями *конкатенации*, *объединения* и *замыкания*, называемых *регулярными*. Наша задача – описать способ его получения, исходя из КС-грамматики, полученной с помощью предварительных преобразований (см. раздел 2).

Множество правил КС-грамматики без самовставлений можно рассматривать как систему уравнений с коэффициентами в виде терминальных цепочек, в которой нетерминалы играют роль неизвестных. Решение такой системы заключается в нахождении регулярных выражений, значения которых представляют множества терминальных цепочек, выводимых из соответствующих нетерминалов грамматики. Метод решения такой системы правил грамматики аналогичен методу Гаусса для линейных алгебраических уравнений. Фактически нас интересует значение регулярного выражения для начального нетерминала грамматики.

Алгоритм 6. Исключение несамовставленных нетерминалов

Вход: $G = (V_N, V_T, P, S)$ – приведённая КС-грамматика без самовставлений.

Выход: регулярное выражение, представляющее язык $L(G)$.

Метод: Введём отношение *зависимости* нетерминалов, определяемое правилами КС-грамматики следующим образом.

Пусть $A \to \alpha_1 B \alpha_2$, где A, $B \in V_N$, α_1, $\alpha_2 \in V^*$, есть правило грамматики. Будем считать, что нетерминал A *зависит* от нетерминала B.

Шаг 1. Построим граф зависимостей нетерминалов. В нём будет столько вершин, сколько нетерминалов в грамматике, по одной на каждый нетерминал. Вершины связываются ориентированными дугами от вершины A к вершине B в соответствии с отношением зависимости, определённым выше.

Шаг 2. С помощью топологической сортировки [6] распределим вершины графа зависимостей по уровням. В результате вершина S окажется на самом высоком уровне, а вершины, не зависящие от других, – на самом низком уровне (*см.* рис. 1).

Шаг 3. Выполним подстановки значений нетерминалов в виде регулярных выражений в правые части правил исходной КС-грамматики в порядке от минимального уровня (0) к максимальному (8) вдоль дуг в противоположном направлении.

При этом прежде чем выполняется упомянутая выше подстановка, леворекурсивное правило, то есть правило вида

$$A: A\alpha; \beta \qquad (1)$$

где α, β – регулярные выражения относительно терминалов, преобразуется к виду

$$A: \beta, (\alpha)^* \qquad (2)$$

а праворекурсивное правило, то есть правило вида

$$A: \alpha A; \beta \qquad (3)$$

преобразуется к виду

$$A: (\alpha)^* \beta \qquad (4)$$

Шаг 4. Алгоритм завершается, когда образуется регулярное выражение для начального нетерминала грамматики S. Именно оно и является выходом данного алгоритма.

Замечание 1. Ясно, что шаг 2 алгоритма 6 выполним, только если в исходном графе зависимостей нетерминалов не существует циклов (*см.* [6]). Однако, петли допустимы.

Замечание 2. На графе зависимостей нетерминалов КС-грамматики (см. рис. 1) лево-сторонняя рекурсия отражается в виде петли.

Замечание 3. На рис. 1 против нетерминалов показаны правые части соответствующих правил исходной грамматики и/или результаты вышеупомянутых подстановок с учётом исключения левосторонней рекурсии, если таковая встречается, и других очевидных регулярных тождеств [4: гл. 5], однако без объяснения, как выбирать и применять регулярные тождества.

Покажем на следующем примере (рис. 1) как преобразовать КС-грамматику чисел языка программирования Алгол 68 [3] к регулярному выражению. Ради краткости, понятия языка Алгол 68 (они указаны в фигурных скобках) обозначены нетерминальными символами A_1–A_{15}. В результате топологической сортировки нетерминалы распо-

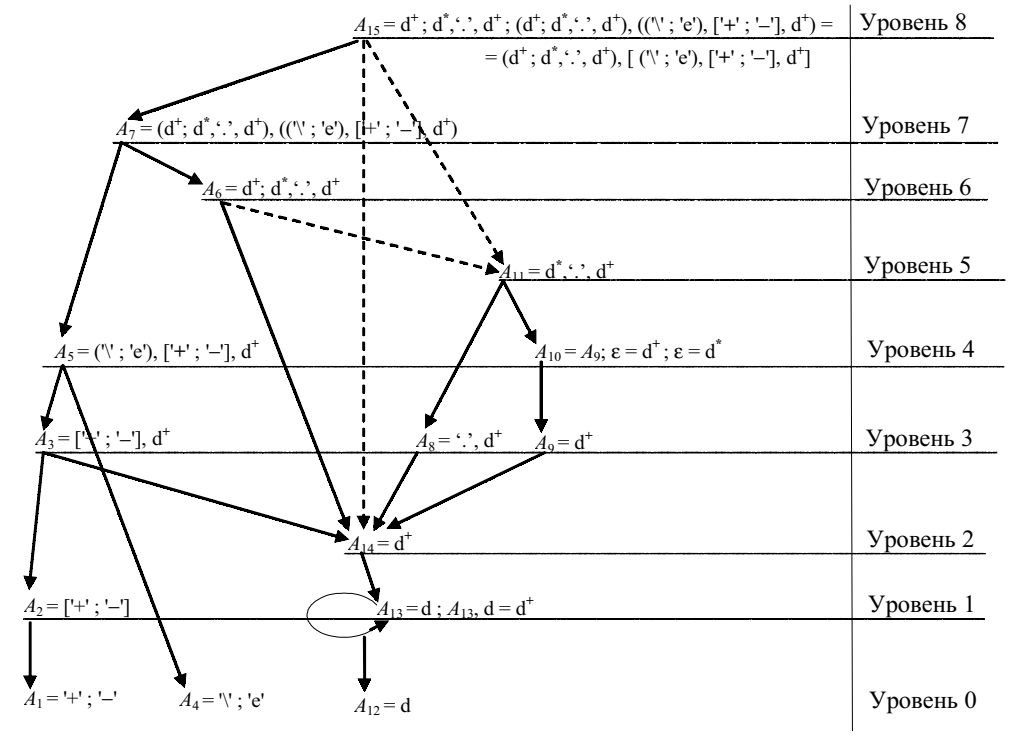
Пример 1. Грамматика чисел в языке Алгол 68.

Дана КС-грамматика $G = (V_N, V_T, P, S)$, где

$V_N = \{A_{15\{\text{number}\}}, A_{1\{\text{plusminus}\}}, A_{2\{\text{plusminus option}\}}, A_{3\{\text{power of ten}\}}, A_{4\{\text{times ten to the power choice}\}}, A_{5\{\text{exponent part}\}}, A_{6\{\text{stagnant part}\}}, A_{7\{\text{floating point numeral}\}}, A_{8\{\text{fractional part}\}}, A_{9\{\text{integral part}\}}, A_{10\{\text{integral part option}\}}, A_{11\{\text{variable point numeral}\}}, A_{12\{\text{digit cypher}\}}, A_{13\{\text{digit cypher sequence}\}}, A_{14\{\text{fixed point numeral}\}}\}$, $S = A_{15}$,

$V_T = \{$ '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '.', '\', 'e', '+'; '–' $\} = d$,

$P = \{A_{15}$: $A_{14}; A_{11}; A_7$.
A_{14}: A_{13}.
A_{13}: $A_{12}; A_{13}, A_{12}$.
A_{12}: '0'; '1'; '2'; '3'; '4'; '5'; '6'; '7'; '8'; '9'.
A_{11}: A_{10}, A_8.
A_{10}: $A_9; \varepsilon$.
A_9: A_{14}.
A_8: '.', A_{14}.
A_7: A_6, A_5.
A_6: $A_{14}; A_{11}$.
A_5: A_4, A_3.
A_4: '\'; 'e'.
A_3: A_2, A_{14}.
A_2: $A_1; \varepsilon$.
A_1: '+'; '-'.$\}$



$A_{15}: d^+; [A_{14}], '.', A_{14}; A_7 =$
$= d^+; [d^+], '.', d^+; (A_{14}; A_{11}), (\' ; 'e'), ((' + ' ; '- ' ;), A_{14}) =$
$= d^+; d^*, '.', d^+; (d^+; [A_{14}], '.', A_{14}), (\' ; 'e'), ((' + ' ; '- ';), d^+) =$
$= d^+; d^*, '.', d^+; (d^+; [d^+], '.', d^+), (\' ; 'e'), ((' + ' ; '- ';), d^+) =$
$= d^+; d^*, '.', d^+; (d^+; d^*, '.', d^+), (\' ; 'e'), ([' + ' ; '- '], d^+) =$
$= (d^+; d^*, '.', d^+), [(\' ; 'e'), ([' + ' ; '- '], d^+)].$

Рис. 1. Результат топологической сортировки графа зависимости нетерминалов и итоговое регулярное выражение

лагаются по уровням в соответствии с отношениями зависимости, определяемыми правилами грамматики.

Именно, если $A \rightarrow \ldots B \ldots$ – правило, где A и B – нетерминалы, то соответствующие вершины графа зависимостей связываются дугой. Причём уровень вершины B должен быть больше уровня вершины A (на рис. 1 на уровне 0 находятся вершины A_1, A_4, A_{12}; на уровне 1: A_2, A_{13}; на уровне 2: A_{14}; на уровне 3: A_3, A_8, A_9; ...; на уровне 8: A_{15}).

Заметим, что *на одном уровне располагаются нетерминалы, не зависимые друг от друга*. В частности, на самом низком уровне (0) эта независимость *абсолютная* – все нетерминалы зависят от регулярных выражений в терминалах.

Подстановки значений нетерминалов в виде регулярных выражений в правые части правил исходной КС-грамматики производятся в порядке от минимального уровня (0) к максимальному (в примере, 8). На рис. 1 против нетерминалов показаны правые части соответствующих правил исходной грамматики и/или результаты вышеупомянутых подстановок с учётом исключения левосторонней рекурсии, если таковая встречается.

Например, преобразование правила для A_{15} можно представить как на рис. 1. Предполагается, что регулярные операции *объединения* (;), *конкатенации* (,), *рефлексивно-транзитивного замыкания* (звёздочка Клини) и *транзитивного замыкания* (плюс Клини) перечислены в порядке возрастания старшинства, причём две последних операции – унарные, имеют одинаковое старшинство. Скобки круглые и квадратные используются, как обычно, для явного указания порядка вычислений, причём подвыражения в квадратных скобках трактуются как необязательные члены регулярного выражения, то есть содержат пустые цепочки.

За счёт средств оптимизации управляющих таблиц анализатора, имеющихся в TK SYNTAX, анализатор, построенный по регулярному выражению в предварительным виде, получается такой же, как тот, который строится по «изящному» виду, и идентичен детерминированному конечному автомату, минимальному по числу состояний (см. табл. 1).

Заметим, что по оптимизированным управляющим таблицам можно восстановить оптимизированную граф-схему, а по ней регулярное выражение в «изящном» виде. Но это уже другая тема.

4. ЗАКЛЮЧЕНИЕ

Как описано в [4], технология SYNTAX основана на аппроксимации входных языков контекстно чувствительными регулярными сплайнами. Это понятие введено по аналогии с понятием полиномиального сплайна в вычислительной математике. Там термин *сплайн* обозначает аппроксимацию функции на разных участках её области определения различными полиномами. В данной технологии термин *регулярный сплайн* обозначает кусочно-регулярную аппроксимацию входного языка. Это находит свое отражение и в том, что в качестве средства задания языков используются RBNF-грамматики, правила которых определяют терминальные порождения нетерминалов через регулярные выражения относительно символов всех алфавитов грамматики, и в том, что соответ-

Табл. 1

Состояние	Цифра (d)	.	\ или **e**	+ или –	**ε**
1	2	3			
2	2	3	4		конечное
3	5			7	
4	6				
5	5		4		конечное
6	6				конечное
7	6				

ствующий языковой процессор подобен множеству конечных автоматов, каждый из которых распознает свой фрагмент входной цепочки. Контекстная чувствительность обеспечивается тем, что дополнительно к алфавитам терминалов и нетерминалов в грамматику вводятся алфавиты контекстных – семантических и резольверных символов.

В данной статье рассмотрены некоторые виды эквивалентных преобразований трансляционных КС-грамматик, в настоящее время ещё не реализованных в ТК SYNTAX.

Описанный выше алгоритм 6 приведения КС-грамматик к регулярному выражению можно обобщить. Если разделить множество всех символов грамматики на несамовставленные нетерминалы и все другие (самовставленные нетерминалы, терминалы, контекстные символы) и обращаться с остальными символами как с терминалами, то после исключения всех несамовставленных нетерминалов мы получим правила для самовставленных нетерминалов, правые части которых будут регулярными выражениями относительно всех других символов грамматики.

Литература

1. *Backus J.W., Bayer F.L., Green J., Katz C., Mc-Carthy., Naur P. (editor), Perlis A.J., Rutishshauser H., Samelson K., Vauquois B., Wegstein J.H., van Wijngaarden A., Woodger M.* Report on the Algorithmic Language ALGOL-60. Numerische Mathematik, 2, № 2 (1960), P. 106–136.

2. *Мартыненко Б.К.* АЛГОЛ 68. Методы реализации / Под ред. *Г.С. Цейтина.* Гл. 1. Ленинград: ЛГУ, 1976. С. 14–70.

3. Пересмотренное сообщение об Алголе 68 / Под ред. *А. ван Вейнгаарден, Б. Майу, Дж. Пек, К. Костер и др*. М., 1979.

4. *Мартыненко Б.К.* Синтаксически управляемая обработка данных. 2-издание, исправленное и дополненное. СПб.: СПбГУ, 2004.

5. *Мартыненко Б.К.* Языки и трансляции. СПб.: СПбГУ, 2004.

6. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы. Построение и анализ. М.: МЦНМО, 1999.

Abstract

There is a variety of parser generators for formal languages being used during designing compilers for programming languages of different types. All of them are based on the use of context-free grammars with various restrictions, regular grammars being a special case. As a rule, they only check the requirements imposed on grammar, and issue diagnostic messages in case of their violation.

The purpose of this article is to describe the way of eliminating all nonterminals from the non-self-embedding grammar that gives the equivalent regular expression as result.

Keywords: context-free grammar, regular expression, equivalent conversion of grammar.

Мартыненко Борис Константинович, доктор физико-математических наук, профессор кафедры информатики математико-механического факультета СПбГУ, mbk@ctinet.ru