ИЗВЛЕЧЕНИЕ КРАЙНЕЙ РЕКУРСИИ ИЗ КСР-ГРАММАТИКИ В СИСТЕМЕ SYNGT

Л. Н. Федорченко

Санкт-Петербургский институт информатики и автоматизации РАН
199178, Санкт-Петербург, 14-я линия В.О., д.39
lnf@iias.spb.su; lnf@pop3.rcom.ru

УДК 618.3
Л. Н. Федорченко. **Извлечение крайней рекурсии из КСР-грамматики в системе SynGT** // Труды СПИИРАН. Вып. 1, т. 1. — СПб.: СПИИРАН, 2002.
Аннотация: *Рассматривается схема удаления лево- и праворекурсивных нетерминалов из правил контекстно-свободной грамматики в регулярной форме, рассмотрены примеры.* — Библ. 4 назв.

UDC 681.3
L. N. Fedorchenko. **Extracting extreme recursion from CFR-grammar in SynGT system** // SPIIRAS Proceeding. Issue 1, v. 1. — SPb.: SPIIRAS, 2002.
Abstract: *Common algorithm for extracting of left/right recursion in CFR-productions are considering in connection with equivalent syntax transformation, examples are demonstrated.* — Bibl. 4 items.

Исследования современных систем построения компиляторов и просмотр наиболее часто задаваемых вопросов в Internet-конференциях по этой тематике показали, что разработчики преодолевают значительные трудности, связанные с настройкой синтаксиса языка, то есть с подготовкой КС-грамматики для генерации языкового процессора. Контекстно-свободные грамматики не только описывают синтаксис языка программирования, но и могут задавать способ трансляции конструкций языка в промежуточную форму. Такая техника называется синтаксически-управляемой трансляцией. Она широко применяется при построении компиляторов и будет использоваться в данной статье.

В настоящее время распространение получили так называемые LR-методы анализа языка (это — SLR, LL, LALR-методы), когда на каждом шаге разбора цепочки языка принятие решения зависит от истории разбора префикса входной цепочки и, возможно, следующего символа на входе. Эти методы позволяют синтезировать детерминированные анализаторы языка магазинного типа. Каждый из LR-методов диктует определенные ограничения на входную грамматику языка (например, требуется исключить лево- или правостороннюю рекурсию). Поэтому необходимо выполнить ряд эквивалентных преобразований исходной грамматики с целью получения грамматики, принадлежащей требуемому классу грамматик, для которых возможен автоматический синтез языкового процессора.

Известно, что проблема эквивалентных преобразований произвольной КС-грамматики в желаемый класс грамматик в общем случае не имеет алгоритмического решения. Поиск частных решений этой задачи для разрабатываемых сложных компиляторов языков линии C (это — C++, Java и другие) и линии Pascal (это — Object Pascal, ADA, Simula-3 и другие) сводится к весьма трудоемким и утомительным преобразованиям синтаксиса вручную или с помощью обычных текстовых редакторов. Если принять во внимание, что, кроме вышеперечисленных и наиболее часто используемых языков программирования, существует и постоянно пополняется список специализированных языков (их уже более 3000 по данным в Internet на январь 2001 года). Эти языки используются в системах искусственного интеллекта, компьютерной алгебре, в системах 3D графики и других, для которых

Труды СПИИРАН. 2002. Вып. 1. Т. 1. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2002. Issue 1. V. 1. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

также необходимо создавать эффективно работающие языковые процессоры, то очевидна необходимость и желательность автоматизировать процесс преобразования синтаксиса языков. В период с 1998 по 1999 год в институте информатики и автоматизации РАН в рамках нескольких проектов совместных с франко-русским центром им. А.М. Ляпунова при МГУ и INRIA (Франция) и проекта по программе "Информатизация России" проводились работы по построению компилятора языка си для специальных сигнальных процессоров и набора инструментальных средств разработки трансляторов. Система Syntax Graph Transformations (SynGT) — одно из таких средств.

Нежелательность крайней рекурсии (левой или правой) в правилах КС-грамматики является наиболее частым ограничением на эти правила, описывающие синтаксис языка.

Для задания синтаксиса языка обычно используется нотация, называемая формой Бэкуса-Наура. В системе SynGT используются регулярные выражения с бинарными операциями. Такая нотация проще транслируется в обратную польскую запись для последующего анализа и обработки. С точки зрения порождаемого языка эти формы задания эквивалентны.

Напомним основные определения, определение КС-грамматики в регулярной форме (КСР-грамматики) и дадим примеры правил для такой грамматики в графической форме (примеры взяты из грамматики языка Си). Затем дадим метод извлечения крайней рекурсии из КСР-правил. Алгоритм реализован в системе SynGT (Syntax Graph Transformations), версии 4.0. в среде Delphi.

Пусть V — алфавит. $V=\{\xi_1, \xi_2,\ldots, \xi_n\}$. При этом V^* — множество всех слов в алфавите V, ε - пустое слово. Для произвольных подмножеств V^* (языков) A и B определим три бинарные операции: объединение множеств слов, которое обозначим "$;$", произведение множеств слов, которое обозначим "$,$" и обобщенная итерация множеств слов — "$\#$".

Объединением множеств слов A и B назовём множество $C=A;B$, представляющее собой теоретико-множественное объединение множеств слов A и B.

Произведением множеств слов A и B назовём множество $C=A,B,$ состоящее из всех слов вида $x=x_1x_2,$ где $x_1\in A,\ x_2\in B.$

Обобщенной итерацией множеств слов A и B будем называть множество $C=A\#B$, состоящее из всевозможных слов вида $x_1y_1x_2y_2\ldots y_{n-1}x_n$, где $x_i\in A,\ 1\le i\le n,\ y_j\in B,\ 1\le j\le n-1$, а n — произвольное положительное целое. Таким образом, операция обобщенной итерации может быть определена через традиционную (одноместную) операцию Клини (*) $A\#B = A,(B,A)^*$.

Регулярным множеством слов (регулярным языком) над V будем называть следующие множества:

- *элементарные множества слов над* V, *т.е.* $\{\xi_1\},\ldots, \{\xi_n\}$;
- *множества* e и $\varnothing$,*где* ($e=\{\varepsilon\}$, $\varnothing$ - пустое множество слов);
- *множество* $A = B\otimes C,$ где B и C - регулярные множества слов, а $\otimes$ — одна из операций $\{\#\ ;,\}$.

Множество всех регулярных множеств слов над алфавитом V обозначим через $\mathbf{R}(V)$. Оно представляет собой минимальное подмножество из 2^{V^*}, содержащее все элементарные множества слов, а также все множества, образованные из элементарных операций. Если первый или второй операнд обобщенной итерации представляет собой множество e и, если позволяет однозначность представления, то он может быть опущен.

$$A\# = A\# e = A;\ A,\ A;\ldots$$

Труды СПИИРАН. 2002. Вып. 1. Т. 1. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2002. Issue 1. V. 1. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

$$\#A = e\ \#\ A = e\ ;\ A\#\ e = e;\ A;\ A,\ A;\ldots$$

Общеупотребительная операция итерации (*) Клини идентична выражению $\#A$, а непустая итерация ($^+$) тождественна $A\#$.

Для того чтобы представить регулярное множество, определим регулярное выражение $r(A)$. Пусть A - произвольное регулярное множество слов над V.

Регулярным выражением множества A называется слово $r(A)$ над расширенным алфавитом

$W = V \cup \{\#, ;, ,, (,), \varepsilon, \varnothing\}$, причем:

1) если $A = \varnothing$, то $r(A) = \varnothing$;

2) если $A = e$, то $r(A) = \varepsilon$;

3) если $A = \{\xi\}$, где $\xi \in V$, то $r(A) = \xi$;

4) если $A = B \otimes C$, где $\otimes \in \{\#\ ;\ ,\}$, B и $C \in \mathbf{R}(V)$, то $r(A) = (r(B) \otimes r(C))$.

Данное определение подчеркивает различие между регулярным множеством и регулярным выражением. Регулярное выражение — это слово, описывающее регулярное множество. Множество, содержащее единственное пустое слово, представляется через ε. Элементарные множества слов — буквой, которую они содержат. Множество, созданное с помощью операций, описывается выражением, которое объединяет построенные аналогичным образом регулярные выражения операндов с символом операции и которое ради однозначности заключается в скобки. Если предположить, что операция обобщенной итерации имеет более высокий приоритет, чем произведение и объединение, а произведение – более высокий приоритет, чем объединение, то при записи регулярного выражения многие скобки можно опустить. Множество всех регулярных выражений над V обозначим через $R(V)$, то есть

$$R(V) = \{X \in W^* |\ \exists A\ \in \mathbf{R}(V) : r(A) = X\}.$$

Каждому регулярному выражению соответствует вполне определенное регулярное множество слов. Однако любое регулярное множество может быть представлено многими различными способами.

Два регулярных выражения $r(A)$ и $r(B)$ из $R(V)$ будем считать *эквивалентными,* если они представляют одно и тоже регулярное множество слов над V. Регулярные выражения, соответствующие одному и тому же регулярному множеству слов, могут быть получены друг из друга с помощью эквивалентных преобразований, некоторые из которых приведены ниже для A, B, C из $R(V)$:

$$A;B = B;A \quad (A;B);C = A;(B;C) \quad (A,B),C = A,(B,C) \quad (A;B),C = A,C;B,C \quad C,(A;B) = C,A;C,B$$
$$A,\varepsilon = \varepsilon,A \quad A,A^* = A^*,A = A^+ \quad (A,B)^*,A = A\#B \quad (A\#B)\#C = A\#(B;C) \quad (A^+)\#B = A\#(\varepsilon;B)$$

Пусть отображение $L: R(V) \to \mathbf{R}(V)$. Если $A \in \mathbf{R}(V)$ - регулярное множество слов над алфавитом V и $r(A) \in R(V)$ — соответствующее ему регулярное выражение, тогда множество $A = L(r(A)) \in V^*$ будет называться *языком*, порождаемым данным регулярным выражением $r(A)$.

Регулярные выражения можно использовать в качестве конечного средства определения бесконечных языков, если правила грамматики языка задавать в виде множества пар вида (A, r), где A - нетерминальный символ грамматики (нетерминал), а r — регулярное выражение, представляющее множество слов (цепочек), непосредственно выводимых из нетерминала A.

Следуя общепринятой терминологии [Ахо А., Ульман Дж., 1978], грамматики, содержащие правила вида (A, r), называются КС-грамматиками в регулярной форме. Каждое правило такой грамматики может интерпретироваться как множество

Труды СПИИРАН. 2002. Вып. 1. Т. 1. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2002. Issue 1. V. 1. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

правил вида $\{A \to x \mid x \in L(R): L(R) \in \mathbf{R}(V)\}$, где V — объединение алфавитов (словарей) терминальных и нетерминальных символов, $V = N \cup T$.

Опыт применения грамматик данного класса диктует дополнить определение грамматики [1] как четверки множеств $G=(N,T,P,S)$, множеством семантик, служащих для вставки необходимых обрабатывающих кодов в компилируемый по данной грамматике текст. Таким образом, рассматриваемая расширенная грамматика представляет собой пятерку множеств $G_R = (N,T,\Sigma,P,S)$, где N — множество нетерминалов, T — множество терминалов, Σ - множество семантик, P — множество КСР-правил, $P = \{A : R_A \;.\; \mid A \in N,\ R_A$ — регулярное выражение над символами из множества $N \cup T \cup \Sigma\}$, S — начальный нетерминал грамматики.

КСР-правила будем записывать в виде:

<нетерминал>:<регулярное выражение>.

Часть правила до двоеточия называется левой частью правила, после двоеточия — правой его частью. Не умаляя общности, будем считать, что для каждого нетерминала имеется единственное определяющее его правило. Действительно, если имеется два правила, определяющих данный нетерминал, то его можно заменить эквивалентным выражением, применив операцию объединения к регулярным выражениям в правых частях обоих правил. В дальнейшем в примерах терминальные символы берутся в кавычки, перед семантиками ставим знак доллара. Для контроля правильности записи во время набора текста грамматики в системе *SynGT* реализована подсветка различными цветами метасимволов (то есть знаков операций регулярных выражений, скобок и ограничителей), терминалов, нетерминалов, семантик и комментариев.

Пример КСР-правила из языка ADA:

$procedure\_call : name\ ,("(",((formal\_parameter,\ "\Rightarrow";\varepsilon),\ actual\_parameter)\#",",")";\ \varepsilon),";"$.

В этом правиле терминальные символы заключены в кавычки, операции скобки и разделители выделены красным цветом, семантики отсутствуют. Так представляются правила КСР-грамматики в текстовом редакторе в SynGT, который, кроме функции проверки правильности записи регулярных выражений, выполняет простые эквивалентные преобразования грамматики (удаляет пустые порождения и бесконечные циклы).

Регулярные выражения в правых частях КСР-правил определяют (в общем случае) бесконечные множества слов (регулярные множества), которые с практической точки зрения удобнее представлять в виде конечного ориентированного графа с метками в вершинах. Такой граф можно считать порождающей конечно-автоматной схемой, где каждая вершина соответствует некоторому состоянию, а ее метка специфицирует порождаемый символ. Совокупность таких графов, определяющих правые части КСР-правил, будет называться синтаксической граф-схемой КСР-грамматики *G*.

Хотя синтаксическая граф-схема (как порождающий механизм) адекватна регулярным выражениям [2], тем не менее она нагляднее выражает структуру правил КСР-грамматики, ее удобнее преобразовать из порождающей в распознающую, а затем в анализирующую схему, с дугами граф-схемы легко связать вызовы семантических процедур и, наконец, синтаксическая граф-схема позволяет использовать более простую структуру, чем вывод в КС-грамматике. Такой структурой является множество маршрутов, или путей в граф-схеме, которые позволяют проще определять и формулировать процесс порождения слов из языка $L(G)$ и проверять его грамматические свойства.

Труды СПИИРАН. 2002. Вып. 1. Т. 1. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2002. Issue 1. V. 1. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Более формально с каждой КСР — грамматикой G связываем ее графовый аналог — *синтаксическую граф-схему* $\Gamma_G=(N,T,\Sigma,K,S)$, где N - нетерминалы, T — терминалы, Σ — семантики, K = $\{K_A \mid A \in N, K_A$ —компонента графа, определяющая нетерминал $A\}$, $S \in N$ - начальный нетерминал и такая что $L_\Gamma=L(G)$. Каждая компонента синтаксической граф-схемы — это синтаксическое определение конструкции языка, дополненное информацией о его семантике. Под семантикой понимается набор выполняемых действий, которые вычисляется заранее, используя семантические правила.

Синтаксическая граф-схема строится рекурсивно из элементарных регулярных выражений (рис. 1) и операций над ними (рис. 2–4).

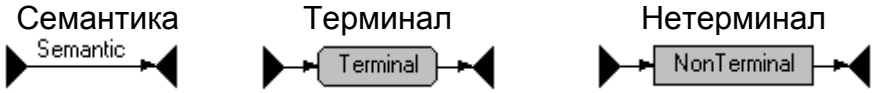


Рис.1. Графическое представление элементарных регулярных выражений

На рисунках 2, 3 и 4 показаны графические представления основных операций над элементарными регулярными выражениями в SynGT. Таким образом в SynGT задаются базисные элементы, к которым рекурсивно сводятся более сложные случаи. Рекурсия по структуре регулярного выражения устанавливается с учетом старшинства операций и их левой ассоциативности.

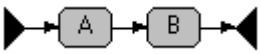


Рис. 2. Конкатенация

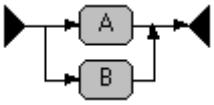


Рис. 3. Объединение

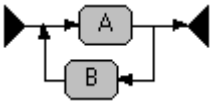


Рис. 4. Итерация

На рисунках 2, 3 и 4 показаны графические представления основных операций над элементарными регулярными выражениями в SynGT. Таким образом в SynGT задаются базисные элементы, к которым рекурсивно сводятся более сложные случаи. Рекурсия по структуре регулярного выражения устанавливается с учетом старшинства операций и их левой ассоциативности.

Графические представления более сложных функций приведены на рис. 5–7.

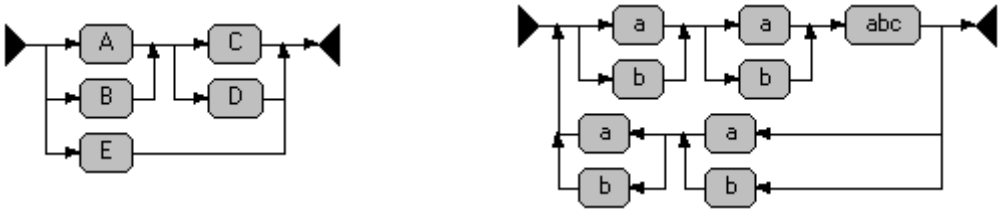


Рис. 5. Графические представления выражений ('A';'B'),('C';'D');'E' и ((('a';'b'),('a';'b'),'abc')#(('a';'b'),('a';'b'))

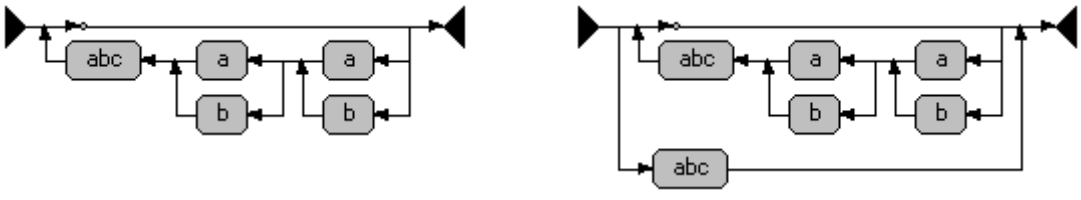


Рис. 6. Графические представления выражений "#((('a';'b'),('a';'b'),'abc') и "#((('a';'b'),('a';'b'),'abc');'abc'

Труды СПИИРАН. 2002. Вып. 1. Т. 1. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2002. Issue 1. V. 1. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Примеры выражений с семантиками (S1, S2, S3,S4 — семантические символы) на рис. 7.

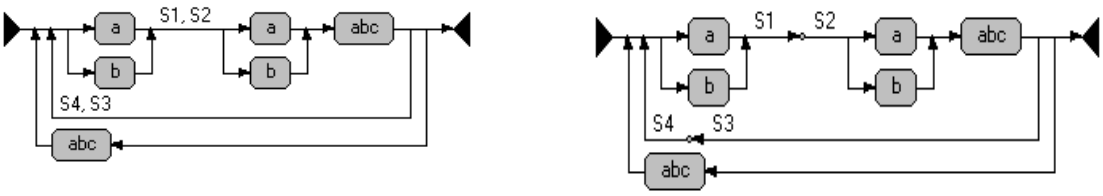


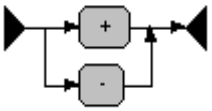
Рис. 7. Графические представления выражений (('a';'b'),S1,S2,('a';'b'),'abc';S3,'',S4)#'abc'
и (('a';'b'),S1,'',S2,('a';'b'),'abc')#(S3,'',S4)#'abc'

Последний граф демонстрирует то, что при нахождении нескольких семантик на линии идущей в обратном направлении, семантики отображаются в обратном порядке.

КСР-правило в графическом вид — это регулярное выражение в графическом виде с пометкой входной и выходной вершин.

hexdigit_sequence: hexdigit# .

sign: '+';'-'.



qualified_identifier : identifier # '.'.

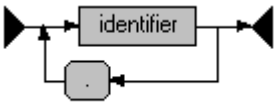


Рис.8. Примеры правил для языка программирования, представляемые в *SynGT*

Извлечение крайней рекурсии из КСР-правил. При построении языкового процессора часто требуется преобразовать КСР-грамматику так, чтобы в ней отсутствовали лево- или праворекурсивные нетерминалы. Эти ограничения вытекают из требования детерминированности механизма синтаксического разбора. Например, из алгоритма построения состояний распознавателя в [2] следует требование конечности разложения состояния любой нетерминальной вершины синтаксической граф-схемы, т.е. все списки маршрутов из нетерминальных вершин к терминальным или конечным вершинам должны иметь конечную длину. В случае бесконечного разложения состояния нетерминальной вершины необходимо преобразование исходной синтаксической граф-схемы Γ_G в граф-схему $\Gamma_{G'}$, эквивалентную данной, т.е. $L(\Gamma_G) = L(\Gamma_{G'})$.

Дадим формальное определение крайней рекурсии в КСР-грамматике.

Нетерминал A называется *самовставленным* (self embedding [4]), если существует вывод $A=>^{*}\alpha A\beta$, где α, β — непустые цепочки над $T\cup N\cup\Sigma$. Другими словами, нетерминал A самовставленный, если он выводится сам из себя в одновременно непустых левом (α) и правом (β) контекстах. Если нетерминал не является самовставленным, то он *несамовставленный*.

Нетерминал A называется *леворекурсивным,* если существует вывод $A=>^{*}A\beta$, где β — непустая цепочка над $T\cup N\cup\Sigma$.

Нетерминал A называется *праворекурсивным,* если существует вывод $A=>^{*}\alpha A$, где α, — непустая цепочка над $T\cup N\cup\Sigma$.

Если нетерминал A несамовставленный, то он не может быть одновременно и лево- и праворекурсивным, т.е. таким, что одновременно существуют выводы ви-

Труды СПИИРАН. 2002. Вып. 1. Т. 1. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2002. Issue 1. V. 1. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

да: $A=>^{*}\alpha A$ (право-рекурсивность) и $A=>^{*}A\alpha$ (леворекурсивность). И в самом деле, в противном случае из двух этих выводов можно было бы построить вывод вида $A=>^{*}\alpha A\beta$

Из теории формальных грамматик известно [1], что язык, порождаемый КС-грамматикой без самовставлений, является *регулярным*, т.е. распознаваемым некоторым конечным автоматом. По следствию из теоремы Клини в этом случае существует регулярное выражение над терминалами грамматики, представляющее тот же самый язык. Оно может быть получено посредством эквивалентных преобразований ее правил. Аналогичные эквивалентные преобразования можно применять и к КСР-грамматикам без самовставлений с целью получить регулярное выражение над терминалами и семантиками. Во всяком случае исключение из грамматики лево- или праворекурсивных нетерминалов всегда возможно [3]. Далее описывается алгоритм решения этой задачи.

Алгоритм исключения лево- и праворекурсивных нетерминалов из КСР-грамматики

Вход: $G = (N,T,\Sigma,P,S)$ приведенная КСР-грамматика с правилами, не содержащими операцию итерации #.

Выход: $G' = (N',T',\Sigma',P',S')$ КСР-грамматика, не содержащая лево- или праворекурсивных нетерминалов в правых частях правил, эквивалентная грамматике G.

1. Разобьем алфавит нетерминалов N данной грамматики G на два непересекающихся подмножества N_1 и N_2, включив в N_1 все несамовставленные нетерминалы, а в N_2 *все* остальные (самовставленные) нетерминалы: $N = N_1 \cup N_2$ Существует очевидный алгоритм, позволяющий определить, является ли данный нетерминал самовставленным или нет).

2. Упорядочим каким-либо образом нетерминалы из φ_{p+1}. Пусть $N_1 = \{A_1, A_2, ..., A_m\}$.

3. Рассмотрим правило для нетерминала $A_1 \in N_1$. В общем случае оно имеет вид:

$$A_1 : A_1\alpha_1; A_1\alpha_2; ...; A_1\alpha_n; \beta_1; \beta_2; ...; \beta_k. \quad (1)$$

или

$$A_1 : \alpha_1 A_1; \alpha_2 A_1; ...; \alpha_n A_1; \beta_1; \beta_2; ...; \beta_k., \quad (2)$$

где в первую очередь перечислены все явнорекурсивные альтернативы, в которых $\alpha_i, \beta_j \in (N \cup \Sigma)^*$, $(i = 1,2,...,n; j = 1,2,...,k)$ не содержат нетерминала A_1.

Очевидно, правило (1) эквивалентно правилу вида:

$$A_1 : (\beta_1;\beta_2;...;\beta_k.),(\alpha_1;\alpha_2;...;\alpha_n)^*. \quad (1')$$

а правило вида (2) эквивалентно правилу вида:

$$A_1 : (\alpha_1;\alpha_2;...;\alpha_n)^*,(\beta_1;\beta_2;...;\beta_k). \quad (2')$$

4. Пусть правила для первых p нетерминалов из N_1 уже приведены к виду

$$A_i : \varphi_i, \psi_i^* \text{ или } A_i : \psi_i^*, \varphi_i. \quad (3)$$

где φ_i, ψ_j — некоторые регулярные выражения над символами из $W = N \cup T \cup \Sigma$ не содержащие нетерминалов $A_1, A_2, ..., A_i$.$(i = 1,2,...,p)$, Преобразуем правило для A_{p+1} к такому же виду (3).

В правую часть правила для A_{p+1} вместо каждого вхождения A_j ($j = 1,2,...,p$) последовательно подставим правую часть определяющего его правила. Таким обра-

Труды СПИИРАН. 2002. Вып. 1. Т. 1. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2002. Issue 1. V. 1. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

зом, из правой части правила для A_{p+1} будут исключены нетерминалы $A_1, A_2,..., A_p$, причем, если A_{p+1} входит в полученное регулярное выражение, то только в одном из двух соответствующих контекстов· (В противном случае какие-то из A_i (i=1,2,...,p) оказались бы самовставленными):

$$A_{p+1}: A_{p+1},\psi_{p+1}; \varphi_{p+1} \text{ . или } A_{p+1}: \psi_{p+1}, A_{p+1}; \varphi_{p+1}$$

где φ_{p+1} и ψ_{p+1} - регулярные выражения, не содержащие нетерминалов $A_1, A_2, ..., A_{p+1}$.

После этого заменим крайние рекурсии на итерации, и правило для A_{p+1}окончательно преобразуем к виду

$$A_{p+1}: \varphi_{p+1} , \psi_{p+1}^{*}. \text{ или } A_{p+1}: \psi_{p+1}^{*}, \varphi_{p+1}.$$

5. Повторяя шаг 4 для p=1, 2, ... , m-1, получаем правила требуемого вида для всех несамовставленных нетерминалов из N_1.

Отметим, что правило для A_m в своей правой части не содержит ни одного несамовставленного нетерминала.

6. Для каждого p = m-1, m-2,..., 1, подставляя в правую часть правила для A_p полученные для $A_m, A_{m-1},..., A_{p+1}$ выражения, окончательно исключим несамовставленные нетерминалы из всех правил грамматики.

7. Положим $N'=\{S\}\cup N_2$, $T'=T$, $\Sigma'=\Sigma$, $S'=S$. В правые части правил, полученных для нетерминалов из N', подставим выражения для несамовставленных нетерминалов, полученные на предыдущих шагах алгоритма. В результате будут получены

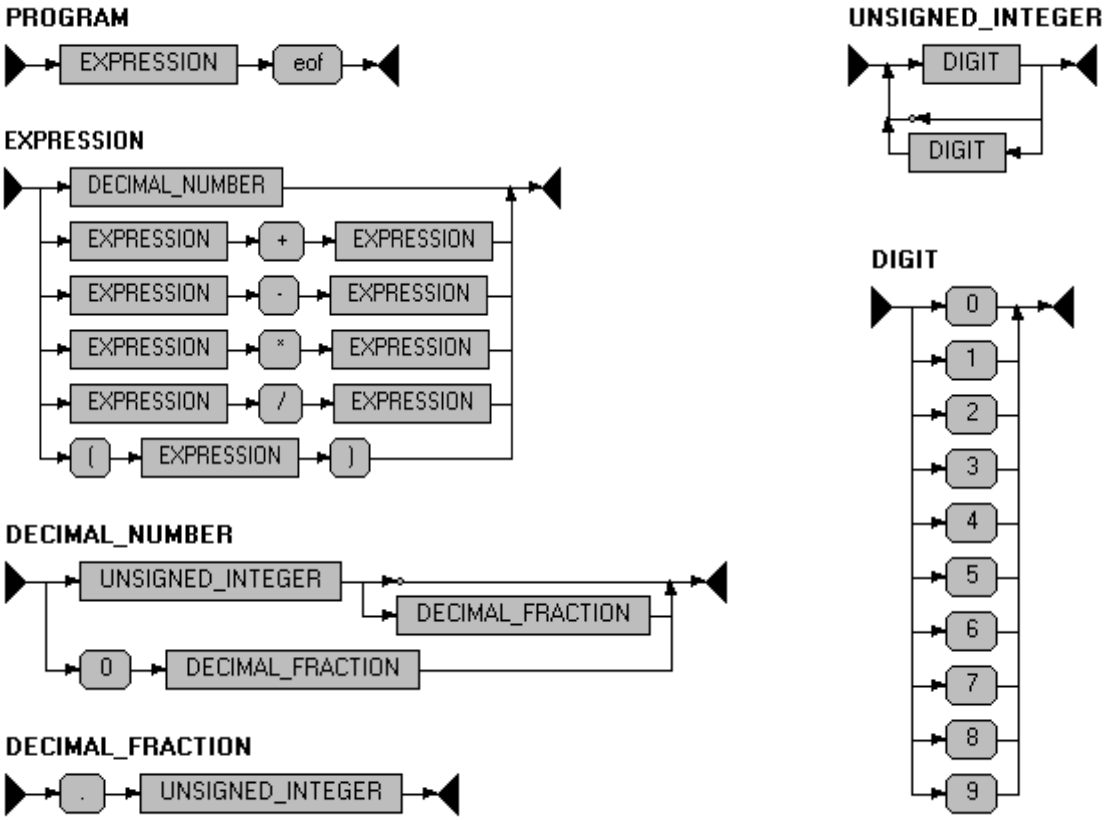


Рис. 9. Синтаксическая граф-схема исходной КСР-грамматики с леворекурс. нетермина-

искомые правила P'. Очевидно, если $N_2=\varnothing$, то грамматика G' — регулярна: в ней имеется единственное правило для нетерминала S, правая часть которого есть регулярное выражение над терминальными и семантическими символами.

Труды СПИИРАН. 2002. Вып. 1. Т. 1. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2002. Issue 1. V. 1. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

На рис. 9 и 10 дан пример синтаксической граф-схемы КСР-грамматики, содержащей леворекурсивный нетерминал EXPRESSION до и после применения алгоритма извлечения левой рекурсии.

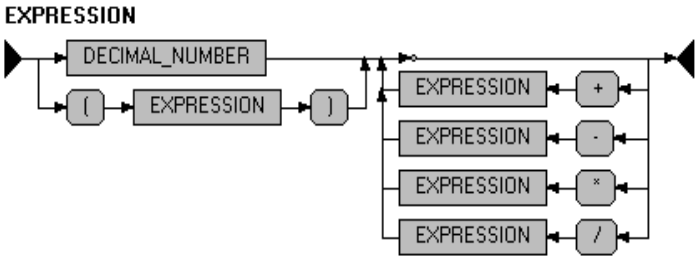


Рис. 10. Пример преобразования леворекурсивного нетерминала

На рис.11–12 показано главное окно системы эквивалентных преобразований КСР-грамматик SynGT и результат преобразований синтаксиса языка Си (фрагмент грамматики). Здесь b1–b7 — новые нетерминалы, возникающие в результате извлечения левой рекурсии из правил и других эквивалентных преобразований.

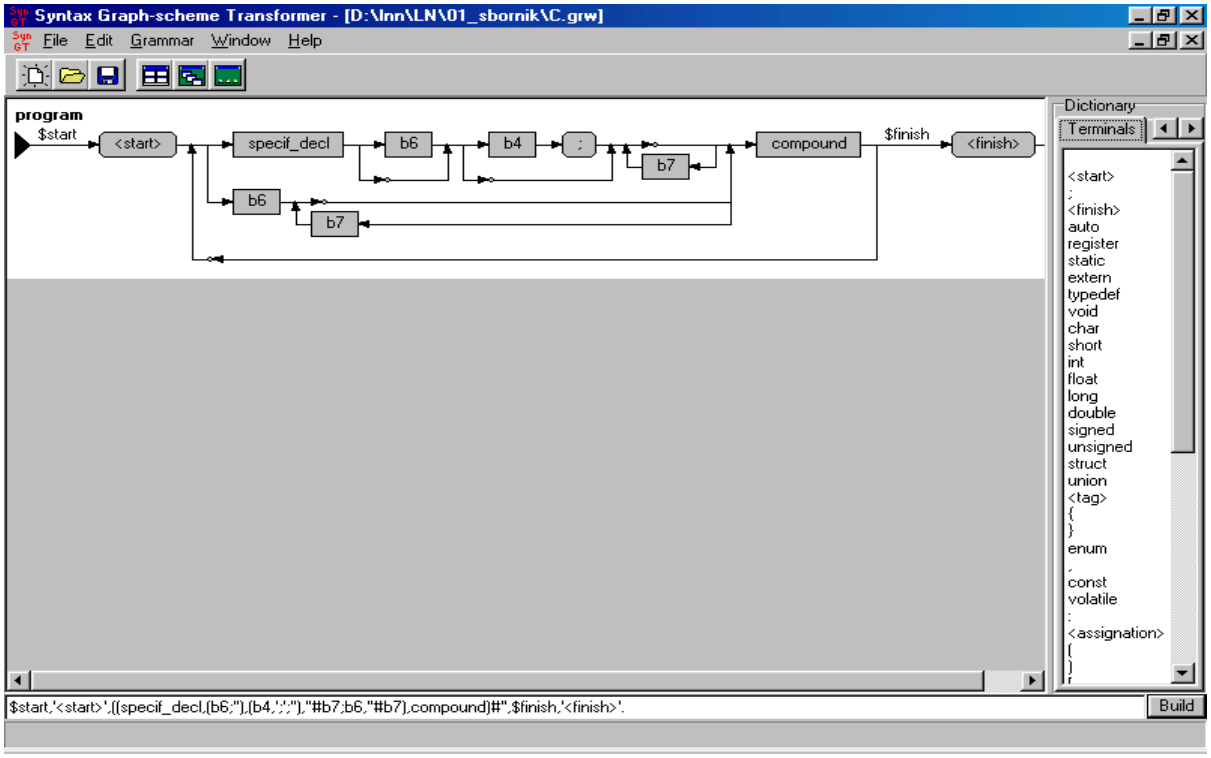


Рис. 11. Главное окно SynGT

Труды СПИИРАН. 2002. Вып. 1. Т. 1. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2002. Issue 1. V. 1. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

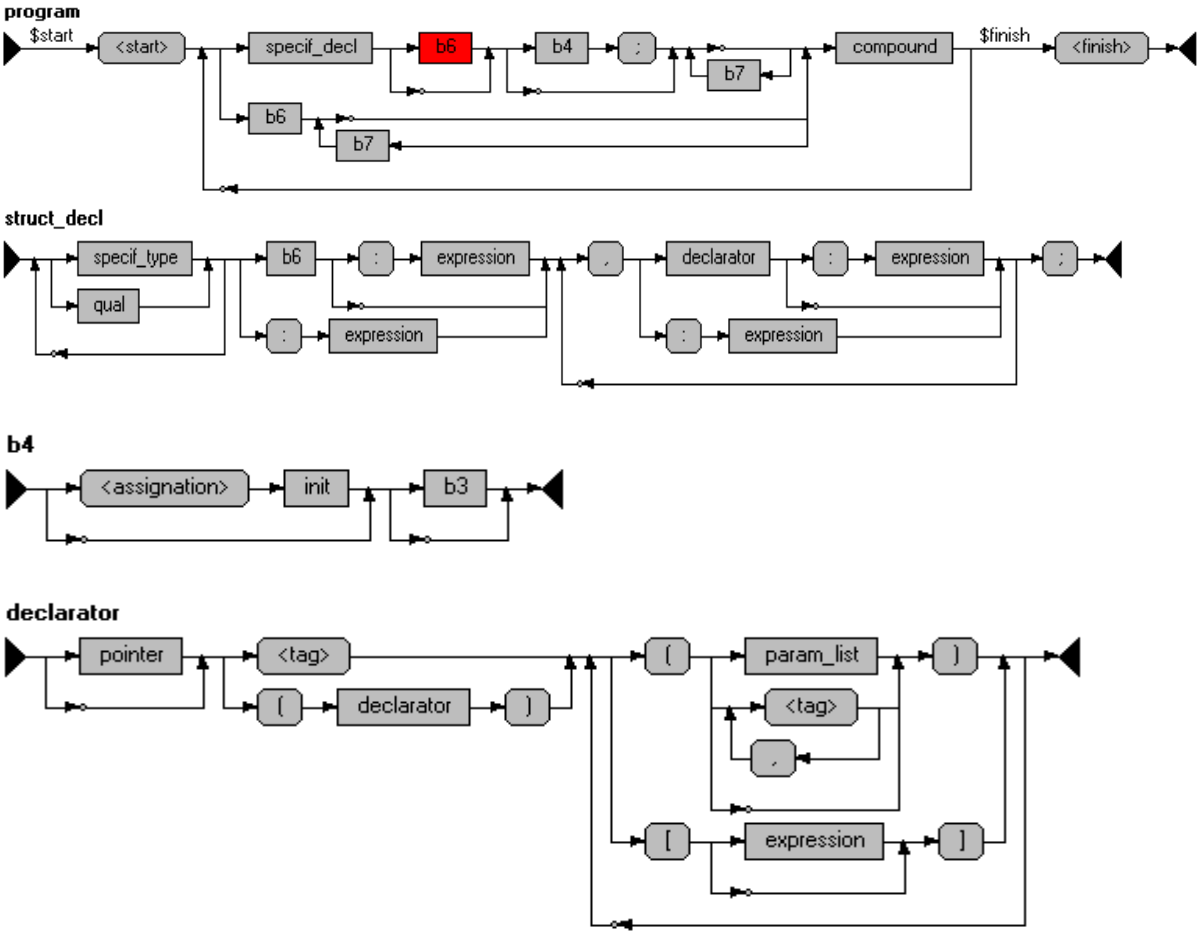


Рис. 12 .Пример графического представления КСР-правил из языка Си

Литература

[1] *Ахо А., Ульман Дж.* Теория синтаксического анализа, перевода и компиляции. т.1. Синтаксический анализ, — 612 с., т.2. Компиляция, — М.:Мир,1978. — 487 с.

[2] *Федорченко Л. Н., Мартыненко Б. К.* Эквивалентные преобразования КСР-грамматик в регулярной форме в практике построения языковых процессоров. Часть первая. Определение и распознавание КСР-языков посредством синтаксических граф-схем. — Ленинград: Академия Наук СССР, Ленинградский Научно-исследовательский Вычислительный Центр, 1983 г.

[3] *Фостер Дж.* Автоматический синтаксический анализ. — М.: Мир, 1975. — 72 с.

[4] *Ginsburg S.* The mathematical theory of Context-free languages. Mc Graw-Hill Inc., 1966.