УДК 519.766.23

Л.Н. Федорченко

РЕГУЛЯРИЗАЦИЯ КОНТЕКСТНО-СВОБОДНЫХ ГРАММАТИК НА ОСНОВЕ ЭКВИВАЛЕНТНЫХ ПРЕОБРАЗОВАНИЙ СИНТАКСИЧЕСКИХ ГРАФ-СХЕМ

Федорченко Л. Н. **Регуляризация контекстно-свободных грамматик на основе эквивалентных преобразований синтаксических граф-схем.**
Аннотация. В статье обосновывается актуальность проблемы быстрой настройки синтаксического определения реализуемого языка на основе применения метода регуляризации трансляционных контекстно-свободных грамматик с помощью эквивалентных преобразований их синтаксических граф-схем, позволяющего оптимизировать построенный синтаксический анализатор. Определяется понятие «регуляризации» для грамматического аспекта. Рассматривается одно из эквивалентных преобразований грамматики в процессе её регуляризации — алгоритм исключения лево- (право)рекурсивных нетерминальных символов из контекстно-свободной грамматики в регулярной форме (КСР-грамматики), который реализован в программном средстве SynGT (Syntax Graph Transformations).
Ключевые слова: КСР-грамматика, синтаксическая граф-схема, эквивалентные преобразования грамматик.

Fedorchenko L.N. **Regularization of context free grammars on the base of equivalent transformations of syntax graph-schemes.**
Abstract. In the article we validate the urgency of the problem rapid tuning a syntactic definition of language that implemented by applying the method of regularization of a translational context-free grammar using the equivalent transformations of their syntactic graph-scheme to optimize the built parser. In the paper we handle the method of regularization of a translational context-free grammar using the equivalent transformations of their syntactic graph-scheme to optimize the built parser. The notion "regularization of a CF grammar" is introduced. We consider a CF grammar in a regular form (CFR-grammar) supplied with extended set of operations in regular expressions in the right hand side of rules. The algorithm of extracting left (right) recursion nonterminals has been shown for CFR-grammar case. It is implemented in the system SynGT (Syntax Graph Transformations).
Keywords: CFR-grammar, syntax graph-scheme, grammars equivalent transformations.

Введение. При построении синтаксического анализатора как составной части транслятора необходимо проводить эквивалентные преобразования грамматики реализуемого языка. Сейчас языковые технологии активно включаются в различные сферы нашей жизни, что привело к развитию современных транслирующих систем, ориентированных на разнообразный ассортимент вычислительных устройств во многих предметных областях. При этом наиболее остро проявилась *проблема быстрой настройки* (преобразования) синтаксического определения языка на ту форму, которая допускает автоматическую

или ручную реализацию, а также проблема учёта *ограничений* выбранного метода синтаксического анализа. Первая проблема обусловлена разнообразием спецификаций реализуемых языков, диапазон которых простирается от обычной формы Бэкуса-Наура (БНФ) и языка разметки HTML, до двухуровневых и других видов грамматик. Вторая – ведёт либо к языковой неоднозначности, либо к недетерминированности распознающего автомата. Решение этих проблем связано с корректным отображением транслируемого языка во внутреннее машинное представление. Для этого следует эффективно использовать информацию о языке, определяя его синтаксис и статическую семантику *специальной КС-грамматикой* (трансляционной), которая помимо основной своей функции порождения цепочек языка позволяет задавать *трансляции*, необходимые разработчикам.

Существующие подходы к реализации языков и обширный набор средств автоматического построения трансляторов, как правило, используют встроенные в инструментальные системы разработки эквивалентные преобразования, часть из которых выполняются в ходе построения синтаксического анализатора, а часть – вручную, что существенно замедляет процесс разработки и приводит к большому числу ошибочных итераций. Стало очевидным, что необходимо выполнять предварительную обработку (препроцессинг) по приведению синтаксических определений языка к нужной форме с помощью инструментария, позволяющего заранее сформировать синтаксис языка и оптимизировать анализатор, применением процедуры *регуляризации* исходной трансляционной грамматики с помощью эквивалентных преобразований. Такой инструмент должен иметь удобный интерфейс и должен обладать достаточным набором функций по выполнению преобразований. Исследование этих преобразований и их применение в технологиях разработки трансляторов подробно представлено в работе [6], в которой также показано, как разрешать проблему языковой неоднозначности и недетерминированности при автоматическом построении синтаксического анализатора.

Построение и описание математической модели определения и распознавания КС-языка на основе синтаксической граф-схемы с целью проверки важных для трансляции свойств входной грамматики и определения ограничений на нее для автоматического синтеза распознавателя языка было выполнено в 1980-х годах в работах [1–2]. В последующем в рамках построенной математической модели, разработан алгоритм регуляризации грамматики языка на основе эквивалентных преобразований синтаксической граф-схемы, позволяющего оптими-

зировать построенный синтаксический анализатор.[3–6]. Чтобы испытать на практике предложенные новые методы на эталонных примерах в сравнении с известными методиками в системах построения трансляторов Flex/Bison и Antlr, была построена экспериментальная инструментальная программная система, реализующая разработанные модели и алгоритмы [3,5,6]

В современных технологиях построения трансляторов предусматриваются различные способы задания трансляций. Разнообразие и многообразие в синтаксических определениях диктует необходимость настройки трансляционных грамматик на соответствующую трансляционную машину (МП-преобразователь), а следовательно, необходимость в *эквивалентных преобразованиях* этих грамматик, поскольку исходную грамматику можно рассматривать лишь как первоначальную форму спецификации синтаксиса языка. Взамен этой спецификации используют другую, эквивалентную ей, но выполненную в ином формализме.

Обзор литературы с 1960-х годов прошлого столетия и до наших дней по существующим методам синтаксического разбора, подробное исследование наиболее известных инструментальных систем построения компиляторов Lex/Yacc, Flex/Bison, Eli, Antlr, Форт-технология, ТК SYNTAX; оценивание влияния типа разбора на проблему эквивалентных преобразований для наиболее сложных спецификаций языка (двухуровневые грамматики Ван Вейнгаардена и аффиксные грамматики Костера) позволяют сделать вывод об *ограниченности применения* в существующих системах автоматических средств для выполнения эквивалентных преобразований правил грамматики и в необходимости поиска новых подходов решения проблемы эквивалентных преобразований, основанных на использовании синтаксических граф-схем для спецификации трансляций и построения оптимального анализатора, применяя регуляризацию трансляционной грамматики для минимизации управляющей таблицы синтаксического анализатора.

Существенным требованием к технологии разработки трансляторов (и в частности, синтаксических анализаторов) является *экономия ресурса* на разработку, это – *время* разработки, *трудозатраты* программиста по созданию соответствующего программного продукта, *компактность* анализатора в случае его встраивания в микропроцессоры и, в итоге, *стоимость* разработки (в виде трудозатрат высококвалифицированных системных программистов). Желательно минимизировать время написания новой грамматики языка, приспособленной

для выбранного метода анализа, разрабатывая специальное инструментальное средство для разработчика.

Характерной особенностью таких систем является то, что информация о конфликте является настолько избыточной, что программисту трудно понять, какое из диагностических сообщений является основным. Например, система Bison выдаёт иногда несколько страниц текста о всех предполагаемых ошибках в правилах грамматики, большая часть которых является наведёнными в результате автоматического исправления какой-либо одной первичной ошибки.

Другим характерным качеством таких систем является или их высокая стоимость (тысячи долларов США для коммерческих систем), или ограничения на тип входной грамматики, хотя современные системы уже расширили входной класс грамматик до класса *LALR* грамматик, порождающий почти все детерминированные языки.

Критический анализ существующих систем построения трансляторов (по информации из comp.compilers news group) позволяет отметить отсутствие полноценных методик, обеспечивающих продолжение анализа правил грамматики после обнаружения конфликтной ситуации (ошибки). Преобразование грамматики анализируемого языка проводится либо вручную, либо недостаточно полно; о приведении грамматики к такому виду, когда сгенерированный по ней анализатор минимален по числу состояний распознающего автомата, речи нет.

Стремясь поправить перечисленные выше недостатки существующих на практике систем построения трансляторов, приходим к необходимости разработать специальный инструментарий для автоматизированного выполнения эквивалентных преобразований синтаксиса реализуемого языка, основанный на новых, нетрадиционных подходах, которые исключают или ослабляют эти недостатки.

Отмечено, что наиболее перспективным подходом является метод эквивалентных преобразований синтаксической граф-схемы исходной трансляционной грамматики с последующей её регуляризацией.

Изложение математической формализации метода построения синтаксических распознавателей (анализаторов) даётся в работах [1,2,6], где определяются контекстно-свободные грамматики в регулярной форме (*КСР*-грамматикам), правые части которых интерпретируются в виде *регулярных выражений* над терминальными, нетерминальными символами и семантиками грамматики. Отправной точкой в изложении метода послужила простая модель: *регулярные выражения* – как средство описания бесконечного языка, и *конечный автомат* – в качестве адекватного средства его распознавания. Обос-

нованием этой модели является теорема Клини (*Stephen Cole Kleene*) о том, что класс регулярных множеств является минимальным классом, содержащим все конечные множества, замкнутым относительно операций объединения, конкатенации и замыкания, и следствия из этой теоремы о возможности представления регулярных множеств (множеств, распознаваемых конечными автоматами) регулярными выражениями.

Использование этой модели возможно лишь в применении к регулярным языкам; например, на фазе лексического анализа в компиляторе. Кроме того, всё множество регулярных выражений над фиксированным алфавитом также можно рассматривать как регулярный язык, который описывается контекстно-свободной грамматикой (метаграмматикой) с одним правилом, и тогда этот язык распознается конечным автоматом, который можно минимизировать по числу состояний и получить абсолютно оптимальный верификатор регулярных выражений. В системе SynGT, разработанной автором, такой конечный автомат реализован и используется для проверки правильности записи регулярных выражений в правилах трансляционной грамматики.

Напомним вариант определения регулярных выражений, отличный от определений Клини, позволяющий частично решить задачу минимизации регулярных выражений и упростить их обработку. В соответствии с данными определениями множество в конечном алфавите V регулярно тогда и только тогда, когда оно либо $\varnothing$, либо $\{\varepsilon\}$,($\varepsilon-$ пустая цепочка символов), либо $\{a\}$ для некоторого $a \in V$, либо его можно получить из этих множеств, применяя конечное число раз операции объединения, конкатенации, унарной и бинарной обобщённой итерации $(\#)$.Она может быть определена через традиционную (унарную) операцию Клини $(*)$ как $P\#Q = P,(Q,P)*$, и частично решает задачу минимизации регулярного выражения по числу вхождений символов из объединённого алфавита, уменьшая вероятность появления конфликта Shift/Reduce при построении распознавателя.

Определение обобщенной итерации $(\#)$ (итерации с разделителем или итерации Цейтина) над множествами слов в алфавите V выглядит следующим образом. Пусть дан алфавит символов $V = \{a_1, a_2, ..., a_n\}$. V^* – множество всех слов в алфавите V.

Определение 1. Обобщенной итерацией множеств слов P и Q называется множество $C = P\#Q$, состоящее из слов вида $x_1y_1x_2y_2\ldots y_{n-1}x_n$, где $x_i \in P, i = 1,\ldots,n, \quad y_j \in Q, j = 1,\ldots,n-1, \quad n > 0$.

Для представления регулярного множества Клини ввел понятие регулярного выражения в алфавите V. Дадим определение регулярного выражения для расширенного набора операций над множествами слов.

Определение 2. *Регулярным выражением множества* A называется слово $r(A)$ над расширенным алфавитом W, где $W = V \cup \{\#, *, ^+, ;, (,), \varepsilon, \varnothing\}$, и, если $A = \varnothing$, то $r(A) = \varnothing$, если $A = e$, то $r(A) = \varepsilon$, если $A = \{a\}$, где $a \in V$, то $r(A) = a$, если $r(P) = p$ и $r(Q) = q$ – регулярные выражения для множеств P и Q. Тогда: $r(A) = (p;q)$ для множества $P \cup Q$; $r(A) = (p,q)$, для множества PQ; $r(A) = (p^*)$, для множества P^*; $r(A) = (p^+)$, для множества P^+; $r(A) = (p\#q)$, для множества $P\#Q$.

Таким образом, в обобщённом регулярном выражении операции являются бинарными. Это удобно в работе со стеком при построении обратной Польской формы записи регулярного выражения.

Для удаления из выражений лишних скобок описываются приоритеты операций: унарные операции $\{*\}$ и $\{^+\}$ обладают наивысшим приоритетом, затем идет итерация с разделителем или обобщенная итерация $\{\#\}$, затем конкатенация $\{,\}$, и далее объединение $\{;\}$. Например, регулярное выражение $p^*;q, p\#q$ с учетом приоритетов означает $((p^*);(q,(p\#q)))$.

Существенным различием между регулярным множеством слов и регулярным выражением является то, что регулярное выражение - это описание регулярного множества, а регулярное множество слов – это язык, порождаемый данным регулярным выражением, его значение.

Формулируется понятие эквивалентности для регулярных выражений: Два регулярных выражения $r(A)$ и $r(B)$ из множества $\Re(V)$ будем считать эквивалентными, если они представляют одно и тоже регулярное множество слов над алфавитом V.

Регулярные выражения, соответствующие одному и тому же регулярному множеству слов, могут быть получены друг из друга с помощью эквивалентных преобразований, например,

$A,(\varepsilon;B\#(A;C),A)\equiv(A\#(B\#C))$. Отметим, что выражение в правой части эквивалентности содержит меньше операндов, а значит меньше вероятность появления конфликта Shift/Reduce при построении распознавателя.

Исходя из сформулированной цели регуляризации грамматик, следующим шагом на пути обобщения регулярной модели является определение контекстно-свободной грамматики в регулярной форме (КСР-грамматики), правила которой задаются в виде множества пар вида (A,r), где A – нетерминальный символ грамматики (нетерминал), а r – регулярное выражение, представляющее множество слов, являющихся значением данного регулярного выражения.

Из определения КСР-грамматики следует, что каждое правило традиционной КС-грамматики автоматически является правилом в КСР-грамматике. Обратное неверно.

Например, правило для процедурного вызова в языке Ada может быть записано в следующем виде (символы терминального алфавита взяты в кавычки):

procedure_call: procedure_name,
("(",((formal_parameter, "⇒";ε),
actual_parameter) # ",")";),";" .

Чтобы представить то регулярное множество слов, которое порождается данным правилом, необходимо записать не одно, а несколько контекстно-свободных правил.

Принципиально новым качеством, отличающим КСР-грамматику от других подобных грамматик, является принятый в ней механизм вывода цепочек языка. Если в традиционной КС-грамматике на каждом шаге вывода происходит замена одной цепочки на другую, то в КСР-грамматике цепочка заменяется значением регулярного выражения, языком $L(A)$ – бесконечным множеством цепочек. Для представления этого множества цепочек используется понятие синтаксической граф-схемы (СГС), являющейся графовым аналогом КСР-правил, а вывод в грамматике заменяется более простой структурой – маршрутом (или путём) в СГС. Под синтаксической граф-схемой понимают набор конечных ориентированных графов с помеченными вершинами и дугами. Каждый граф соответствует одному правилу КСР-грамматики и называется графом для нетерминала, определяющего КСР-правило. Две вершины графа Γ_A для нетерминала A являются входными и выходными с метками E_A и F_A. Внутренние вершины помечены терминальными и нетерминальными символами – операн-

дами регулярного выражения правой части правила, определяющего нетерминал A, а дуги помечены контекстными символами (семантиками) – именами семантических процедур, которые должны исполняться по ходу синтаксического анализа.

На рис. 3–5 показаны графические представления основных операций над элементарными регулярными выражениями, которые реализованы в программной системе эквивалентных преобразований SynGT:

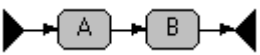

Рис. 3. Конкатенация

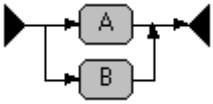

Рис. 4. Объединение

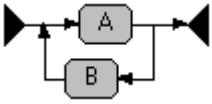

Рис. 5. Итерация

Так задаются базисные элементы, к которым рекурсивно сводятся более сложные случаи, например, такие как на рис. 6–7.

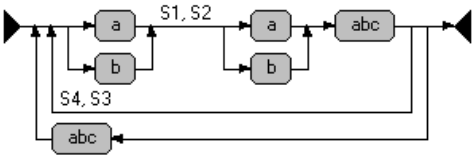

Рис.6. Граф, соответствующий регулярному выражению
(('a';'b'),S1,'',S2,('a';'b'),'abc')#(S3,'',S4)#'abc'

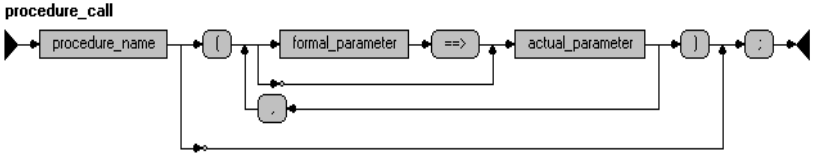

Рис. 7. Граф, соответствующий правилу для нетерминала «Procedure_call»

Для порождения цепочек языка вводится отношение достижимости на вершинах СГС Γ_G, которое отражает возможную последовательность символов в порождаемой цепочке. Тогда маршрут P_x, определяющий процесс порождения цепочки x из языка L_G, содержит последовательность вершин в Γ_G, первый и последний элементы ко-

торой являются входной и конечной вершинами в графе для начального нетерминала КСР-грамматики, причём, каждый элемент достижим из предыдущего (если таковой имеется), а соответствующая последовательность меток вершин представляет символы цепочки x.

Введение понятия *синтаксической граф-схемы* в качестве порождающего механизма для языка вызвано следующими соображениями:

– являясь адекватным регулярному выражению, граф нагляднее отражает структуру *КСР*-грамматики и не вводит избыточную рекурсивность в KC-язык;

– в терминах вершин удобней исследовать и проверять свойства автомата, распознающего язык, порождаемый синтаксической граф-схемой на наличие конфликтных состояний;

– с дугами граф-схемы легко связать вызовы процедур и делать проверки на семантическую эквивалентность;

– синтаксическая граф-схема позволяет использовать более простую структуру, чем вывод в традиционной *КС*-грамматике, а именно маршруты или путь, в терминах которого удобней формулировать и проверять ограничения на класс грамматик с эффективным анализом.

В качестве иллюстрации рассматривается построение простейшего конечного магазинного распознавателя, синтезируемого по *СГС*. Определяется понятие *состояния*, играющее основную роль в синтезе распознающих автоматов. Известному из литературы понятию состояние МП-автомата, придается внутреннее содержание. Состоянию, в котором находится автомат, приписывается определенное количество информации о языке. В данном случае под состоянием в синтаксической граф-схеме Γ_G понимается множество вершин в Γ_G.

Переход из одного состояния в другое, которое также является множеством (возможно пустым) вершин в *СГС*, управляется *текущим символом*, поступающим из входного текста. Переходное состояние содержит информацию о том, какие символы допустимы для следующего перехода. Поэтому состояние содержит информацию о том, какие текущие символы разрешены в данный момент, а следовательно, какие подцепочки допустимы к моменту перехода в это состояние. Отсюда, состояние позволяет установить принадлежность слова языку, порождаемому данной синтаксической граф-схемой.

Разнообразные способы определения понятия состояния в СГС и переходного состояния позволяют промоделировать известные из литературы МП-преобразователи.

В работах [1–6] описываются свойства синтаксических граф-схем, в которых состояния любых вершин **существуют всегда** (т.е. процесс построения переходного состояния не зацикливается и заканчивает работу), и формулируются условия детерминированности для выполнения операции перехода по текущему символу в процессе синтеза МП-распознавателя. Точки (набор вершин в граф-схеме) нарушения этих свойств определяют то, какую операцию эквивалентного преобразования необходимо применять.

Описание модели завершается формулировкой теоремы о языке, допускаемом описанным распознавателем, которая дает алгоритм, с помощью которого за число шагов, пропорциональное длине слова, можно проверить, принадлежит ли данное слово языку, порождаемому синтаксической граф-схемой. Сложность шага (расчет переходного состояния) в общем случае велика, поэтому данный алгоритм представляет чисто теоретический интерес. Он является основой для моделирования известных из научной литературы *LL*- распознавателей.

Под *регуляризацией КС*-грамматики понимается процесс применения цепочки базисных эквивалентных преобразований исходной *КС*-грамматики G в новую *КС*-грамматику G_1 в регулярной форме (КСР-грамматику), причем такую, что из нее исключаются подстановками несамовставленные нетерминалы. Алгоритм исключения приводится в [4,6]. Если все нетерминалы КС-грамматики G не являются самовставленными, то их можно исключить из правых частей правил на начальном этапе обработки. Определяющие правила для этих нетерминалов исключаются также. В этом случае грамматика G преобразуется в грамматику G_1 с одним правилом, правая часть которого является сложным по композиции операций регулярным выражением над терминалами и контекстными символами (семантиками). Значением этого регулярного выражения является регулярный язык $L = L(G_1) = L(G)$.

При регуляризации грамматики правые части её правил попрежнему являются регулярными выражениями над объединённым алфавитом грамматики, а синтезированный МП-преобразователь может выродиться в конечноавтоматный.

Алгоритм регуляризации грамматик, реализованный в программной системе SynGT, состоит из последовательности следующих этапов обработки трансляционной грамматики:

1. Выделение контекстно-свободной составляющей языка, в том случае, если синтаксис языка задан в виде двухуровневой грамматики

(аффиксной или Ван Вейнгаардена) или какой-либо другой, отличной от КС-грамматики. Такая КС-составляющая будет входной грамматикой для системы эквивалентных преобразований SynGT.

2. Эквивалентное преобразование входной грамматики в приведённую *КСР*-грамматику, когда удаляются пустые порождающие правила, непродуктивные нетерминальные символы, тупиковые и циклические порождения.

3. Исключение лево- (право) рекурсивных нетерминалов из правил грамматики.

4. Выполнение операции глобальной подстановки вместо нетерминальных вершин их порождений.

5. Минимизация регулярных выражений в правых частях правил.

Рассмотрим подробнее алгоритм исключения лево- (право)рекурсивных нетерминалов из КС-грамматики с обобщенной итерацией.

Для простоты изложения рассмотрим A-правило в КСР-грамматике, когда нетерминал A является одновременно и лево, и праворекурсивным, и рекурсия прямая. Хорошо известно, что косвенная рекурсия методом подстановок сводится к прямой. В отличие от всех известных алгоритмов извлечения крайних рекурсий, в SynGT реализован алгоритм прямого эквивалентного преобразования лево-(право)рекурсивного нетерминала с использованием операции обобщённой итерации.

Рассмотрим A-правило для нетерминала A вида

$$A: A, r_{11}, A; \quad A, r_{12}; \quad r_{21}, A; \quad r_{22}., \tag{1}$$

где $r_{11}, r_{12}, r_{21}, r_{22}$ – регулярные выражения над алфавитом $N \cup T \cup \Sigma$.

Данное A–правило (1) состоит из четырёх частей (A_i – фрагментов), $i = 1,2,3,4$:

$$A, r_{11}, A \; ; \quad A, r_{12} \; ; \quad r_{21}, A \; ; \quad r_{22} \; ,$$

где

A_1 – фрагмент содержит одновременно и лево, и праворекурсивное вхождение нетерминала A;

A_2 – фрагмент содержит левоорекурсивное вхождение нетерминала A;

A_3 – фрагмент содержит праворекурсивное вхождение нетерминала A;

A_4 – фрагмент не содержит рекурсивных вхождений нетерминала A.

Рассмотрим, какие строки могут порождаться с помощью A-правила (1).

Шаг 1. Рассмотрим A_1 – фрагмент A-правила: A, r_{11}, A. Используя данный фрагмент A-правила мы можем вывести строки

$$A,\ Ar_{11}A,\ Ar_{11}Ar_{11}A, \ldots ,$$

Из определения операции # это множество строк совпадает с множеством строк, порождаемым регулярным выражением

$$A\#r_{11} \ . \tag{2}$$

Шаг 2. Рассмотрим A_2 – фрагмент A – правила: A, r_{12}.

Здесь мы можем вывести строки

$$A\,r_{12},\ \ Ar_{12}r_{12},\ Ar_{12}r_{12}r_{12},\ \ldots \ . \tag{2.1}$$

Из определения унарной операции * следует, что множество (2.1) порождается регулярным выражением

$$A, r_{12}{}^* \ . \tag{2.2}$$

Подставив вместо A в выражении (2) A_2 –фрагмент A-правила (2.2), получим регулярное выражение

$$(A, r_{12}{}^*)\#r_{11} \ . \tag{3}$$

Шаг 3. Аналогично для праворекурсивного вхождения нетерминала A в A_3 – фрагменте. Здесь мы выводим множество строк $r_{21}A,\ r_{21}r_{21}A, r_{21}r_{21}r_{21}A,\ \ldots$, которые порождаются регулярным выражением $r_{21}{}^*, A$. Сделав подстановку вместо вхождения нетерминала A в (3) выражения $r_{21}{}^*, A$, получим регулярное выражение

$$(r_{21}{}^*, A, r_{12}{}^*)\#r_{11} . \tag{4}$$

Шаг 4. Окончательно, подставляя вместо вхождения нетерминала A в регулярное выражение (4) A_4 – фрагмент r_{22}, получим эквивалентное регулярное выражение

$$(r_{21}{}^*, r_{22}, r_{12}{}^*)\#r_{11} . \tag{5}$$

Таким образом, правую часть A – правила (1) можно заменить на регулярное выражение (5).

Если предварительно воспользоваться преобразованиями регулярных выражений и привести правило к такому виду, что множество $L(r_{12})$ не содержит цепочек вида αA, множество $L(r_{21})$ не содержит цепочек вида $A\alpha$, а множество $L(r_{22})$ не содержит цепочек вида $A\alpha$ и вида αA, где α произвольная цепочка, тогда прямая левая и правая рекурсия для нетерминала A отсутствуют.

Перенумеровав все нетерминалы в грамматике G и последовательно применив данное преобразование для каждого нетерминала, подставляя результаты преобразований в оставшиеся правила, получим грамматику G', эквивалентную исходной грамматике G без крайних рекурсий.

Если r_{11}, r_{12}, r_{21}, r_{22} регулярные выражения, не содержащие вхождений нетерминала A, то нетерминал A и правило для него можно удалить из грамматики G', а все его вхождения заменить на выражение $(r_{21}{*}, r_{22}, r_{12}{*})\#r_{11}$.

Алгоритм подготовки исходной грамматики с целью её регуляризации предполагает использование следующих базисных эквивалентных преобразований над регулярными выражениями и КСР-правилами.

1. $T_1(A)$ – подстановка вместо нетерминала A его порождения.
2. $T_2(A)$ – удаление вхождения леворекурсивного или праворекурсивного нетерминала A в правой части правила.
3. $T_3(A)$ – объединение общих префиксов в графе для нетерминала A.
4. $T_4(A)$ – удаление повторяющихся альтернатив для нетерминала A
5. $T_5(A)$ – удаление крайних рекурсий для самовложенных нетерминалов.
6. $T_6(A)$ – сведение регулярного подвыражения в новый нетерминал в графе для A и создание нового правила в грамматике.
7. T_7 – удаление лишних правил.

На основании описанной в [1–2,6] модели, сформулированных утверждений и алгоритмов в работах [1–6] описывается программная реализация – система эквивалентных преобразований грамматик SynGT, которая подтверждает теоретические исследования и результаты, изложенные в [6].

В качестве основных информационных компонент система SynGT содержит следующие: SynGT – документ, рабочее поле (desktop) с областями, синтаксическая граф-схема, подграф, дуга (семантика), нетерминальная, вспомогательная и терминальная вершины, правило КСР-грамматики. В SynGT – документе содержится информация о синтаксической граф-схеме, состоянии рабочего поля, соответствующей синтаксической граф-схеме КСР-грамматики, комментарии к текущему SynGT - документу в текстовой форме.

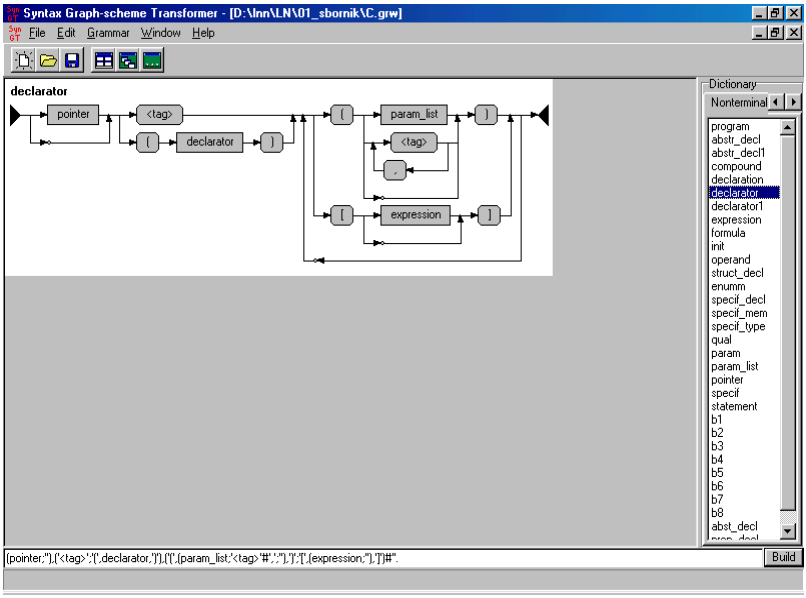

Рис. 8. Рабочее поле системы SynGT

Заключение. Подводя итог проделанной многолетней работы[1–6], можно сделать следующие выводы:

1. Разработаны и реализованы алгоритмы эквивалентных преобразований грамматик с целью построения компактных и эффективных анализаторов языков, с улучшенными на 10-15-20%, по сравнению с известными системами из семейства компиляторов GNU характеристиками по эффективности и объему памяти. На этих алгоритмах построена схема регуляризации трансляционной грамматики языка. Эквивалентные преобразования и метод регуляризации на их основе являются единственным средством расширить область применимости метода синтаксического анализа языка, который фиксирован в системе построения трансляторов.
2. Предложен метод построения простых и естественных распознавателей языков для достаточно большого класса грамматик. Выбор метода синтаксического анализа был продиктован следующими соображениями:

– эффективностью разбора текста (линейное время);

– достаточно широким классом допустимых грамматик с исчерпывающим определением его грамматических свойств (порождающие детерминированные языки).

– простым набором базисных эквивалентных преобразований.

3. Разработана специальная инструментальная система эквивалентных преобразований КСР-грамматик SynGT, обеспечивающая автоматизированную настройку синтаксиса реализуемого языка в рамках простого метода синтаксического разбора. Система является автономной и может быть использована и в других технологиях разработки. В настоящий момент система SynGT позволяет применять эквивалентные преобразования к правилам любой приведённой КС-грамматики в регулярной форме. Класс трансляционных грамматик, удовлетворяющих выбранному методу синтаксического анализа, порождает детерминированные языки, определение которых дано Д.Кнутом.

4. Эффективность метода построения анализатора достигается в результате предварительного этапа обработки входной трансляционной КСР-грамматики.

Практическая ценность проделанной работы состоит в следующем:

– разработанные модели и алгоритмы положены в основу методики эквивалентных преобразований и регуляризации трансляционных грамматик, в рамках которой построена экспериментальная программная система, позволяющая, по сравнению с известными подходами, повысить скорость подготовки корректной спецификации реализуемого языка в 3-4 раза и точнее определить разработчику языкового процессора необходимые затраты экономических, временных и кадровых ресурсов;

– разработанный алгоритм регуляризации КСР-грамматики позволяет проверить, является ли порождаемый ею язык регулярным, и тем самым оптимизировать синтезируемый языковой процессор по числу внутренних состояний соответствующего конечного автомата;

– разработанный способ спецификации языка с использованием синтаксической граф-схемы может быть применён для тестирования корректной работы анализатора. Алгоритмы для генерации множества тестовых предложений языка могут быть использованы при проведении тестирования и в других подобных действующих системах построения трансляторов;

Литература

1. *Федорченко Л.Н.* Об одном алгоритме синтаксического анализа языков, порождаемых R-грамматиками: сб. научных трудов ЛНИВЦ АН СССР, “Алгоритмы и системы автоматизации исследований и проектирования” / М: Наука, 1980 – С. 146–155.
2. *Федорченко Л.Н.* Алгоритм синтаксического анализа языков, порождаемых R-грамматиками – "Algorithms and Systems of the Automation Research and Designing", Наука, М., 1983. С.15-20
3. *L. Fedorchenko,* Syntax Graph Transformations in the System SynGT and Regularization of Grammars, Procs of Intern Multi-Conference on Advanced Computer Systems (ACS–CISIM 2004), 14–16 June, Elk, Poland [электронный ресурс] http://acs.wi.ps.pl/info.php
4. *Федорченко Л.Н.* О регуляризации контекстно-свободных грамматик. / Изв. вузов. Приборостроение, 2006. Т.49, №11. С.50–-54.
5. *Федорченко Л.Н.* Метод регуляризации грамматик в системах трансляции языков. // VI Юбилейная международная научная конференция “Инновации в науке и образовании–2008,” посвящённая 50-летию пребывания КГТУ на Калининградской земле. Труды конференции в 3-х частях, часть 2 – Калининград: 2008 – С. 305. ISBN 978–5–94826–217–8. С.294–297.
6. *Федорченко Л.Н.* Регуляризация контекстно-свободных грамматик на основе эквивалентных преобразований синтаксических граф-схем, Дисс… канд. тех. наук по специальности 05.13.11 – "Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей". Библиотека СПИИРАН, 2009. 160 с.

Федорченко Людмила Николаевна — к.т.н.; старший научный сотрудник лаборатории прикладной информатики учреждения Российской академии наук Санкт-Петербургского института информатики и автоматизации РАН (СПИИРАН). Область научных интересов: синтаксически ориентированная обработка данных; регуляризация грамматик; разработка программного обеспечения, поддерживающего технологию синтаксически ориентированной обработки данных. Число научных публикаций — 51. lnf@iias.spb.su; СПИИРАН, 14 линия, 39, Санкт-Петербург, 199178, РФ. Р.т. +7(812)328-1919. Факс +7(812)328-4450.

Ludmila Fedorchenko — PhD; senior researcher of the laboratory of applied informatics of institution of the Russian Academy of Sciences St. Petersburg Institute for Informatics and Automation of RAS (SPIIRAS). Research area: Syntax-directed data processing; methods and algorithms of grammar regularization; parsing; software SynGT (Syntax Graph Transformations). Number of publications — 51. lnf@iias.spb.su; SPIIRAS, 14 line, 39, Saint-Petersburg, 199178, Russia; tel +7(812)328-1919, fax +7(812)328-4450.

Рекомендовано лабораторией ЛПИ, зав. лаб. Юсупов Р.М.., чл.-корр РАН, проф.
Статья поступила в редакцию 22.12.2010.

РЕФЕРАТ

Федорченко Л. Н. **Регуляризация контекстно-свободных грамматик на основе эквивалентных преобразований синтаксических граф-схем.**

Регуляризация грамматики – это метод подготовки исходной грамматики с помощью применения эквивалентных преобразований правил контекстно-свободной грамматики в новую грамматику в регулярной форме (КСР-грамматику), причем такую, что из нее исключаются подстановками несамовставленные нетерминалы. Процесс преобразований проводится на синтаксической граф-схеме – графическом аналоге КСР-грамматики.

Новый алгоритм регуляризации трансляционных контекстно-свободных грамматик с помощью эквивалентных преобразований их синтаксических граф-схем, позволяет аппроксимировать входной язык регулярными множествами слов. Показана его применимость на фрагментах реальных языков программирования, обеспечивающая высокий процент повышения эффективности (15-20%), по сравнению с известными методами в системах построения трансляторов Flex/Bison и Antlr. Алгоритм регуляризации грамматики может применяться для создания миниатюрных языковых процессоров во встраиваемое программное обеспечение.

Достигнутый уровень автоматизации при использовании системы эквивалентных преобразований совместно с выбранной технологией автоматизации проектирования трансляторов составляет от 50% до 100% в зависимости от специфики проекта. Снижение трудозатрат – примерно в 3–4 раза.

SUMMARY

Fedorchenko L.N. **Regularization of context free grammars on the base of equivalent transformations of syntax graph-schemes.**

The regularization of grammar is a method of preliminary design phase of preparing source grammar by equivalent transformations of the rules of the CF-grammar into a new grammar in a regular form (CFR-grammar), and such that all non-self-embedded nonterminals are excluded by substitutions. The transformation process is carried out on the syntactical graph diagram, which is a graphical analog of CFR-grammar.

A new algorithm for the regularization of the translational context-free grammars by means of equivalent transformations of their syntactic-classical graph-schemes that can approximate the input language by regular sets of words. Shown its applicability to the real language fragments, providing a high percentage increase efficiency (15-20%) compared with the known methods in Flex / Bison and Antlr. Algorithm for the regularization of grammar can be used to create miniature language processors for embedded software.

Using the system for equivalent transformations in conjunction with the chosen technology the level of automation of compiler developing is obtained from 50% to 100% depending on the specifics of the project. Reduced labor costs is about 3-4 times.