УДК 681.51
DOI: 10.18101/2304-5728-2019-2-44-60

ГОМОМОРФИЗМ ЭКВИВАЛЕНТНЫХ ПРЕОБРАЗОВАНИЙ ГРАММАТИК, ПРИМЕНЯЕМЫЙ ПРИ ГЕНЕРАЦИИ АНАЛИЗАТОРОВ ЯЗЫКА

© Федорченко Людмила Николаевна
кандидат технических наук, доцент, старший научный сотрудник
Санкт-Петербургский институт информатики и автоматизации Российской академии наук
Россия, 199178, г. Санкт-Петербург, 14 Линия В.О., 39;

Санкт-Петербургский государственный университет,
Россия, 199034, г. Санкт-Петербург, Университетская наб., 7/9
E-mail: LNF@iias.spb.su

При построении транслятора, как правило, необходимо проводить эквивалентные преобразования грамматики реализуемого языка, преобразующие синтаксическую спецификацию языка в тот вид, который допускает автоматическую или ручную реализацию исходного языка, и решающие проблемы учёта ограничений выбранного метода синтаксического анализа. Эти проблемы возникают как из-за разнообразия способов определения реализуемых языков, так и из-за языковой неоднозначности или недетерминированности распознающего автомата. Каждое преобразование, выполняемое на трансляционных КС-грамматиках, может задаваться как отношение подобия (гоморфизма), определяемое на классах грамматик. Такие отношения имеют непосредственную связь с сохранением семантического значения грамматических конструкций при эквивалентных преобразованиях синтаксиса, так как конечной целью трансляции является получение последовательности действий, предписываемых вычислительной среде.
В статье представлен краткий обзор различных типов отношений подобия на грамматиках, применяемых при построении трансляторов с 1970-х гг.
Представлена схема построения анализаторов *LL*-языков с использованием синтаксических граф-схем (СГС). Даны формальное понятие маршрута (пути) в СГС и пример автоматического преобразования КСР-грамматики формального языка *CIAO*, выполненный в системе эквивалентных преобразований SynGT (Syntax Graph Transformations).
***Ключевые слова**:* эквивалентные преобразования грамматик; отношение подобия; гомоморфизм покрытия; КСР-грамматики; синтаксическая граф-схема; регуляризация грамматик.

Для цитирования:
Федорченко Л. Н. Гомоморфизм эквивалентных преобразований грамматик, применяемый при генерации анализаторов языка // Вестник Бурятского государственного университета. Математика, информатика. 2019. № 2. С. 44–60.

Введение

При построении компилятора (или любого транслятора) разработчики реализуемого языка проводят эквивалентные преобразования грамматики исходного языка, необходимые, чтобы получить ту форму синтаксического определения языка, которая допускает автоматическую или ручную реализацию языка, а также учесть ограничения, диктуемые методом синтаксического анализа. Первая проблема возникает из-за разнообразия вида определений реализуемых языков, вторая — из-за языковой неоднозначности или недетерминированности распознающего автомата. Для этого следует так использовать информацию о языке, определяя его синтаксис и статическую семантику в виде специальной КС-грамматики (трансляционной) [1], чтобы помимо основной своей функции порождения цепочек языка грамматика позволяла задавать трансляции, необходимые разработчикам [1–3].

1 Эквивалентность КС-грамматик и гомоморфизмы

Под *трансляцией* из языка $L_1 \subseteq \Sigma^*$ в язык $L_2 \subseteq \Delta^*$ называется отношение $\tau \subseteq L_1 \times L_2$. Здесь Σ — *входной алфавит*, L_1 — *входной язык*, Δ — *выходной алфавит*, L_2 — *выходной язык.* Другими словами, трансляция есть некоторое множество пар предложений (x, y), где $x \in L_1$ — *входное*, а $y \in L_2$ — *выходное предложение*.

Каждое преобразование, выполняемое на трансляционных КС-грамматиках [1–4] может задаваться как отношение подобия (гоморфизма) [6], определяемое на классах грамматик. Такие отношения имеют непосредственную связь с сохранением семантического значения грамматических конструкций при эквивалентных преобразованиях синтаксиса (теорема Флойда), так как конечной целью трансляции является получение последовательности действий, предписываемых вычислительной среде (Environment), как одного из разделов спецификации трансляционной грамматики.

Из грамматик наиболее простыми для разработки являются регулярные грамматики типа 3 (по иерархии Хомского) [3], которые порождают регулярные языки, обрабатываемые конечными автоматами. На практике широко применяется грамматики типа 2-*контекстно-свободные грамматики* (КСГ), относительная простота которых позволяет применять для анализа КС-языков эффективные полиномиальные алгоритмы. Например, алгоритм Эрли (Early) [5] для КСГ имеет вычислительную сложность $O(n^3)$, где n — длина входной цепочки. Более сильные ограничения на КС-языки классов *LL*, *LR, LALR* дают возможность синтезировать анализаторы, имеющие линейную сложность разбора.

Перевод грамматик из одного класса в другой — это задача эквивалентных преобразований синтаксического описания языка.

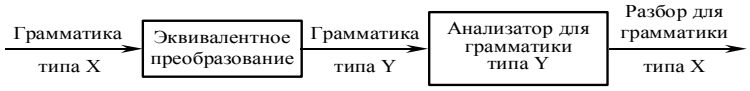

Рис. 1. Схема применения эквивалентного преобразования при разборе

Таким образом, разработчики трансляторов и инструментальных систем, автоматизирующих процесс разработки, при настройке синтаксиса всегда стремятся преобразовать его таким образом, чтобы получить грамматику, принадлежащую к наиболее простому классу иерархии Хомского или к тому подклассу КСГ, для которого есть используемый в системе автоматического построения трансляторов метод синтеза анализатора языка.

При автоматизированном построении транслятора дополнительно к задаче приведения грамматики к необходимому способу спецификации приходится решать задачу разрешения конфликтов, связанных с языковой неоднозначностью или недетерминированностью распознающего устройства. Снять эти проблемы можно только с помощью эквивалентного преобразования грамматики или такого преобразования, которое восстановит первоначальную семантику с помощью гомоморфизмов (отношений на грамматиках).

В работах [7], [8], [9], [10], [11], [12], [13] подробно рассмотрены различные типы эквивалентных отношений — слабая и структурная эквивалентность, сильная структурная эквивалентность. Показано, что проблема сильной структурной эквивалентности алгоритмически разрешима, так как существует только конечное число нумераций, а для всякой такой нумерации скобочная версия $G_{[\,]}$ определяет простой детерминированный язык (*S*-язык). Преобразование $G_{[\,]}$ в S-грамматику прямое. Кореньяк, Хопкрофт в [14] и Саломаа в [15] доказали, что проблема, порождают ли две S-грамматики один и тот же язык, алгоритмически разрешима. Все эти и другие типы эквивалентных преобразований рассмотривались с точки зрения сложности грамматического разбора. С целью выразить отношение грамматического подобия более точно появилось много новых формальных понятий. В работах [16; 17] даны определения и описаны результаты для грамматических гомоморфизмов, изоморфизмов, слабых покрытий и т. д. В работах [18–21] даны определения и описаны результаты для грамматических гомоморфизмов, изоморфизмов, слабых покрытий и покрытий по Рейнольдсу. В этих определениях сделан акцент на правила грамматик, а не на множества выводов или грамматических разборов. Так, отношение гомоморфной эквивалентности КС-грамматик, являясь более тонким, чем отношение структурной, сильно структурной и частично структурной эквивалентности, может быть определено следующим образом:

КС-грамматика $G=(N,T,P,S)$ называется гомоморфно эквивалентной грамматике $G'=(N',T',P',S')$, если существует гомоморфизм

$\psi : G \to G'$ и $L(G) = L(G')$. В этом случае гомоморфизм ψ представляется тройкой отображений $\psi_1 : G \to G'$, $\psi_2 : T \to T'$, $\psi_3 : P \to P'$ и $\psi_1(S) = S'$.

В работе [20] Грей и Харрисон определили понятие грамматического покрытия (разновидность гомоморфизма), которое описывает отношение между множествами деревьев вывода двух КС-грамматик. Интерес авторов к этому понятию был основан исключительно на его применении в области синтаксического анализа языков.

Если ограничиться моделью анализа, в которой каждое предложение исходного языка даётся описанием его деревьев вывода в виде строки правил грамматики (или меток, идентифицирующих эти правила), то соответствие двух КС-грамматик, которое описывается грамматическим покрытием, является отношением между такими описаниями деревьев вывода для данного предложения.

Результат синтаксического анализа можно считать аргументом семантического отображения. В случае когда КС-грамматика G' покрывает КС-грамматику G, разбор предложения w относительно G' может быть отображён гомоморфизмом на разбор той же w относительно G. Отсюда следует вывод, что язык $L(G)$ можно анализировать относительно преобразованной грамматики G', а затем использовать первоначальное семантическое отображение, как на рис. 1.

Неформально говоря, основная идея в том, что та КС-грамматика G, которая «трудна» для анализа или не удовлетворяет определённым ограничениям, диктуемым методом синтаксического анализа, преобразуется в КС-грамматику G', которую можно анализировать более простым анализатором. Последующий синтаксический анализ проводится относительно преобразованной G', а после этого результат анализа (разбора) отображается покрывающим гомоморфизмом на соответствующий разбор относительно G.

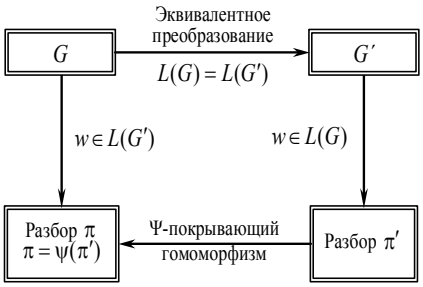

Рис. 2. Схема эквивалентного преобразования с использованием покрытий

Подводя итог вышесказанному, можно сказать, что в том случае, если синтаксические правила, которые подаются на вход инструментальной системе, не специфицируют грамматику, для которой выбранный тип анализатора может быть построен, возникают следующие ситуации:

– система диагностирует о своей неспособности построить анализатор и выдает информацию, почему она терпит неудачу; эта информация может быть использована разработчиком языка для того, чтобы изменить (не обязательно эквивалентно преобразовать) эти правила (и связанные с ними семантики) с целью сделать их подходящими для системы;
– система может применить эквивалентные преобразования к входной грамматике для того, чтобы сделать её адекватной методу. Но это должно делаться так, чтобы первоначальные семантики сохранялись;
– синтаксические правила на входе системы не специфицируют грамматику, которая годится для метода анализа, заложенного в инструментальной системе, но у разработчика есть возможность обеспечить систему дополнительной информацией (например, пополнить новыми правилами), достаточной для построения правильного анализатора.

Если синтаксические правила (с дополнительной информацией) не специфицируют грамматику желаемого типа, то система может принять решение, которое ведёт к успешному построению анализатора. Но эквивалентные преобразования могут изменять структуру грамматики. Это означает то, что преобразованная грамматика не обязательно выполняет тот же самый перевод к семантическим действиям, что и первоначальная (исходная грамматика). Для этого необходим покрывающий гомоморфизм, при котором, если преобразование делается таким образом, что можно провести разбор в терминах исходной грамматики, можно как бы «обмануть» пользователя инструментальной системы. Эта идея была проиллюстрирована на рис. 2.

2 Обобщение регулярной модели

Можно отказаться от модели с деревьями вывода в КС-грамматиках и обратиться к модели с вычислением значений регулярных выражений.

Формальной базой для построения трансляторов является методика, основанная на использовании КС-грамматик с обобщёнными регулярными выражениями и атрибутами в виде семантик для представления контекстных ограничений. Эта методика развивается в курсе лекций студентам СПбГУ и поддерживается пакетом программных приложений, реализованных на платформе Delphi. В качестве основы взят код специального инструментального средства SynGT, разрабатываемого с 2000 г. [2].

Формальной моделью методики построения языковых процессоров, описанной в [1–2], являются регулярные выражения и конечный автомат — в качестве адекватного средства их обработки.

Данная модель обобщена до применения для класса КСР-языков с использованием специальных магазинных автоматов [19; 21; 22]. КСР-язык порождается КСР-грамматикой — обобщением КС-грамматики. Основные определения были даны в работах [1; 2; 4]. Из теории [2] формальным языком является набор цепочек языковых токенов (лексем или терминалов) в соответствии с грамматикой языка.

Синтаксис языка описывается его контекстно-свободной составляющей в виде КС-правил и контекстно-зависимой (ограничения).

КС-правила представлены в виде обобщённых регулярных выражений (КСР-правила):

Нетерминал : *обобщённое регулярное выражение* .

Регулярные выражения состоят из токенов (лексем), нетерминалов, регулярных операций (конкатенация, альтернативный выбор и обобщённая итерация), пустой цепочки ("") и скобок ("(" и ")").

- Конкатенация (“,”): A,B = AB.
- Альтернативный выбор (“;”): A;B = либо A, либо B.
- Обобщённая итерация (“#“): A#B = A; ABA; ABABA; ABABABA;
- Унарная итерация Клини (“*“): A* =…; A; AA; AAA;…
- Возможно пустое множество цепочек X: [X] = (X;)

Обобщение конечно автоматной модели обработки языков сводится к следующему:

- вводится итерация с разделителем (#) или обобщённая итерация, или итерация, которая не расширяет множество регулярных слов и может быть определена через традиционную (одноместную) операцию Клина (*) как (P#Q)=P,(Q,P;
- КСР-грамматика — обобщение КС-грамматики. Класс языков не расширяет, но снимает лишнюю структурированность языка;
- синтаксическая граф-схема (СГС) — графический аналог КСР-грамматики, стартовый объект для синтеза распознавателя (анализатора) языка, в терминах вершин которой удобно формулировать свойство детерминированности языка и возможные типы конфликтов (Shift/Reduce). СГС является промежуточной моделью между порождающей структурой, грамматикой и распознающей машиной, МП-автоматом.

3 Синтаксическая граф-схема (СГС)

Регулярные выражения в правых частях КСР-правил определяют бесконечное множество слов (регулярные множества), которые удобнее представлять в виде конечного ориентированного графа с метками в вершинах.

Такой граф считается порождающей конечно-автоматной схемой. В нём каждая вершина соответствует состоянию конечного автомата, а ее метка специфицирует порождаемый символ.

В синтаксической граф-схеме (СГС) каждому нетерминалу соответствует одна компонента — граф для нетерминала или одно правило КСР-грамматики, правая часть которого является обобщённым регулярным выражением над символами объединенного алфавита грамматики. В объединённый алфавит грамматики входят алфавиты терминалов, нетерминалов, семантик и, возможно, предикатов. СГС создается рекурсивно из графов для элементарных регулярных выражений и бинарных операций, подробно описанных в [1].

Одним из основных понятий в данной модели является *понятие достижимости,* с помощью которого формулируется вычисление множества слов (язык), порождаемых СГС. Дадим определения.

4 Достижимые вершины и понятие маршрута (пути) в синтаксической граф-схеме

Пусть дан алфавит $V = \{\xi_1, \xi_2, \ldots \xi_n\}$, $n > 0$.

Определение 4.1. *Граф-схемой* над алфавитом V называется конечная совокупность ориентированных графов $\Gamma = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_k\}$, $(k = 1,2,\ldots,n)$, удовлетворяющая следующим условиям:

– в каждом графе Γ_i, $\forall i \in \{1,2,\ldots,k\}$, выделены две вершины — *входная и выходная*, причем во входную вершину не входит, а из выходной вершины не выходит ни одна из дуг, а каждая из вершин, отличных от этих двух, помечена буквой из алфавита V;

– графы Γ_i для $\forall i \in \{1,2,\ldots,k\}$ не имеют между собой общих вершин и называются *компонентами граф-схемы*.

Входную и выходную вершины в каждом графе Γ_i $\forall i \in \{1,2,\ldots,k\}$ обозначим E_i и F_i соответственно. Пусть вершина α принадлежит графу $\Gamma_i \in \Gamma$.

Определение 4.2. Буква ξ из алфавита V, которой помечена вершина α, называется *меткой вершины* α и обозначается $m(\alpha)$, $\xi = m(\alpha)$.

Каждой вершине α такой, что $(\alpha \neq F_i)$, $\forall i \in \{1,2,\ldots,k\}$, в граф-схеме Γ поставим в соответствие множество вершин $succ(\alpha)$, смежных с данной вершиной: $succ(\alpha) = \{\beta \mid \beta \in \Gamma\}$, причем существует дуга, началом которой является вершина α, а концом — вершина β}.

Любым двум вершинам α и β, принадлежащим графу $\Gamma_i \in \Gamma$ в граф-схеме Γ, $(\alpha, \beta \neq F_i)$, можно поставить в соответствие множество слов $L_i(\alpha,\beta)$, порождаемых маршрутами от вершины α к вершине β.

Определение 4.3. *Множество слов* $L_i(\alpha,\beta)$, *порождаемых маршрутами от вершины* α *к вершине* β, *– это:*

1. Пустое слово $\varepsilon \in L_i(\alpha,\beta)$, если $\alpha = \beta$;
2. Слово $x\xi \in L_i(\alpha,\beta)$, если $x \in L_i(\alpha,\gamma)$, и существует вершина β графа Γ_i такая, что $\beta \in succ(\gamma)$ и $m(\beta) = \xi$.

Лемма 4.1. Слово $x \in V^*$ принадлежит множеству слов $L_i(\alpha,\beta)$ тогда и только тогда, когда существует последовательность вершин $\alpha_0, \alpha_1, \ldots, \alpha_k$, $(k \geq 0)$, графа Γ_i, такая, что $\alpha = \alpha_0$, $\beta = \alpha_k$, $\alpha_i \in succ(\alpha_{i-1})$, $m(\alpha_i) = \xi_i$, $\forall i \in \{1,2,\ldots,k\}$ и $x = \xi_1\xi_2\ldots\xi_k$.

Обозначим через L_i – множество слов, порождаемых всеми маршрутами в графе $\Gamma_i \in \Gamma$, начинающихся из входной вершины E_i графа Γ_i,

$$L_i = \bigcup_{F_i \in succ(\beta)} L_i(E_i, \beta).$$

Следствие 4.1. Слово x в алфавите V принадлежит множеству L_i тогда и только тогда, когда существует последовательность вершин графа Γ_i, $\alpha_0, \alpha_1, \ldots, \alpha_k$ $(k \geq 0)$ такая, что $\alpha_0 = E_i$, $F_i \in succ(\alpha_k)$, $\alpha_i \in succ(\alpha_{i-1})$, $m(\alpha_i) = \xi_i$, $\forall i \in \{1,2,\ldots,k\}$ и $x = \xi_1 \xi_2 \ldots \xi_k$.

Очевидно, что для любого графа Γ_i множество слов L_i регулярно.

В работе [1] есть строгие доказательства этих утверждений и подробно рассмотрен вопрос однозначности регулярных выражений.

Далее построим граф-схему для нетерминала в КСР-грамматике G. Для этого условимся в дальнейшем алфавитом V считать объединение алфавитов нетерминалов, терминалов и пустой буквы, то есть $V = N \cup T \cup \{\lambda\}$. В дальнейшем алфавит V будет расширяться за счет добавления новых символов-понятий — вспомогательных нетерминалов, семантик, предикатов и других.

Тогда если метка вершины α, $m(\alpha)$ из алфавита N, то α называется *нетерминальной вершиной*; если $m(\alpha)$ из алфавита T, то α — *терминальная вершина*; если $m(\alpha) = \lambda$, то α — *пустая вершина*.

Определение синтаксической граф-схемы для KCP-грамматики G будем вести последовательно: сначала опишем способ построения графа Γ_A для всякого регулярного выражения A над алфавитом V, затем определим граф для нетерминала из алфавита N KCP-грамматики G.

Построение графа Γ_A проводится индуктивно следующим образом: для элементарных выражений $(\varepsilon \, и \, \xi \in V)$ соответствующие графы показаны на рис. 3, 4. Семантики будем отображать на дугах, для них графическое представление дано на рис. 5.

В работе [1] дан рекуррентный алгоритм построения синтаксической граф-схемы. В начале строится граф Γ_A для регулярных выражений с операциями *конкатенации, объединения и обобщённой итерации*, формулируются леммы о равенстве множества слов (языка), порождаемого регулярным выражением A и графом Γ_A.

Введение синтаксической граф-схемы в качестве механизма порождения цепочек языка вызвано следующими соображениями:

– являясь адекватным регулярному выражению (Лемма 2.2.) в [1], граф нагляднее отражает синтаксическую структуру KC-языка и не вводит лишнюю рекурсивность (структурированность) в его спецификацию;

– из граф-схемы проще извлекать информацию о порождающих цепочках, рассматривая маршруты в графах, а не деревья вывода в традиционных грамматиках;

– в терминах вершин удобней описать состояния автомата (анализатора) языка, порождаемого синтаксической граф-схемой;
– с дугами граф-схемы удобно связать вызовы процедур из специального раздела(Environment) спецификации грамматики;
– использовать более простую структуру, чем вывод в KC-грамматике, а именно *маршрут* или *путь* в синтаксической граф-схеме, в терминах которого формулируются и проверяются ограничения на класс грамматик с эффективным анализом (линейной сложности разбора).

Определим множество слов (язык) — L_G, порождаемых синтаксической граф-схемой Γ_G для КСР-грамматики $G = \{N, T, P, S\}$.

Пусть Γ_A — граф для нетерминала A из словаря N. Определим рекурсивно множество $H(A)$ — начальных вершин графа Γ_A:

Определение 4.1

1. Если α — вершина графа Γ_A и $\alpha \in succ(E_A)$, то α — начальная вершина графа Γ_A для нетерминала A, $\alpha \in H(A)$;

2. Если $\alpha \in H(A)$ — нетерминальная вершина в Γ_G, $m(\alpha)=B$ и β — начальная вершина графа Γ_B, то есть $\beta \in H(B)$, то β — начальная вершина графа Γ_A, т.е. $\beta \in H(A)$.

Аналогично определим множество конечных вершин $K(A)$ графа Γ_A:

Определение 4.2

1. Если α — вершина графа Γ_A и выходная вершина $F_A \in succ(\alpha)$, то α — *конечная* вершина графа Γ_A для нетерминала A, $\alpha \in K(A)$;

2. Если $\alpha \in K(A)$, α — нетерминальная вершина в Γ_G, $m(\alpha) = B$ и вершина $\beta \in K(B)$, то вершина $\beta \in K(A)$.

Введем понятие *пути (маршрута)*, соединяющего две вершины в синтаксической граф-схеме Γ_G и определим отношение достижимости на множестве терминальных вершин.

Определение 4.3

Между терминальными вершинами α_1 и α_2 в синтаксической граф-схеме Γ_G, существует путь (маршрут), если выполняется одно из следующих условий:

1. Вершины α_1, α_2 — смежные, принадлежат одной и той же компоненте Γ_A для некоторого $A \in N$, и $\alpha_2 \in succ(\alpha_1)$. Тогда путь $P_{\alpha_1\alpha_2}$ от вершины α_1 к вершине α_2 имеет вид:

$$P_{\alpha_1\alpha_2} = \{\alpha_2\}.$$

2. Существует нетерминальная вершина β, смежная с вершиной α_1, $\beta \in succ(\alpha_1)$, такая, что $m(\beta) = B$, а терминальная вершина α_2 — начальная вершина в графе для нетерминала B, $\alpha_2 \in H(B)$.

В этом случае путь $P_{\alpha_1\alpha_2}$ от вершины α_1 к вершине α_2 имеет вид:

$$P_{\alpha_1\alpha_2} = \{\beta\} \cdot P_\beta, \text{ где}$$

а) если α_2 принадлежит графу Γ_B, то есть $\alpha_2 \in succ(E_B)$, то

$$P_\beta = P_{E_B \alpha_2} = \{\omega_H \alpha_2, \},$$

где ω_H — специальный символ, обозначающий след прохождения через входную вершину графа для нетерминала;

b) если существует нетерминальная вершина $\gamma \in \Gamma_B$, $m(\gamma) = C$, $\gamma \in succ(E_B)$, и вершина $\alpha_2 \in H(C)$, то

$$P_\beta = P_{E_B \gamma} \cdot P_\gamma = \{\omega_H \gamma\} \cdot P_\gamma .$$

3. α_1 — конечная вершина графа Γ_B, и $\alpha_2 \in succ(\beta)$, где β — нетерминальная вершина и $m(\beta) = B$. Тогда путь от вершины α_1 к вершине α_2 имеет вид:

$$P_{\alpha_1 \alpha_2} = P^{\beta}_{\alpha_1} \cdot \{\alpha_2\}, \text{где}$$

а) если α_1 — вершина графа Γ_B, т. е. $F_B \in succ(\alpha_1)$, то путь

$$P^{\beta}_{\alpha_1} = \{\omega_K \beta\},$$

где ω_K — специальный символ, обозначающий след прохождения через выходную вершину графа для соответствующего нетерминала;

b) если α_1 — вершина графа Γ_C, т. е. $F_C \in succ(\alpha_1)$ и существует нетерминальная вершина γ, $m(\gamma) = C$ и $\gamma \in K(B)$, то

$$P^{\beta}_{\alpha_1} = \{\omega_K \gamma\} \cdot P^{\beta}_{\gamma} .$$

4. Вершина α_1 — конечная в графе Γ_B, то есть $\alpha_1 \in K(B)$, α_2 — начальная вершина графа Γ_C, т. е. $\alpha_2 \in H(C)$ и существуют нетерминальные вершины β и γ в графе Γ_A, такие что $m(\beta) = B$, $m(\gamma) = C$ и $\gamma \in succ(\beta)$, тогда путь между вершинами α_1 и α_2 имеет вид:

$$P_{\alpha_1 \alpha_2} = P^{\beta}_{\alpha_1} \cdot \{\gamma\} \cdot P_\gamma .$$

Определение 4.4

Терминальная вершина α_2 достижима из терминальной вершины α_1 в синтаксической граф-схеме Γ_G, (обозначение $\alpha_1 \underset{G}{\rightarrow} \alpha_2$ или $\alpha_1 \rightarrow \alpha_2$, если G подразумевается), если существует путь (маршрут) $P_{\alpha_1\alpha_2}$ из вершины α_1 к вершине α_2.

Заданное таким образом отношение достижимости на множестве вершин в граф-схеме Γ_G позволяет рассматривать Γ_G как порождающую схему, в которой путь определяется множеством цепочек вершин, являющихся классом эквивалентности по отношению равенства $\cong$.

4.1 Отношение эквивалентности

Рассмотрим множество всех вершин в синтаксической граф-схеме Γ_G. Исключим из него все входные и выходные вершины компонент графов и добавим два новых элемента ω_H, и ω_K, где ω_H, заменяет все входные, а ω_K — выходные вершины. Полученное множество обозначим $\mathfrak{I}$.

Рассмотрим расширенный алфавит $\mathfrak{I} \cup \{\Omega\}$. Определим отношение эквивалентности $\cong$ на множестве слов в этом расширенном алфавите следующей системой равенств.

Определение 4.5

Пусть α и β — любые нетерминальные вершины в множестве $\mathfrak{I}$, x — цепочка терминальных вершин, X, Y — произвольные цепочки в алфавите $\mathfrak{I} \cup \{\Omega\}$. Тогда отношение эквивалентности $\cong$ на множестве всех маршрутов синтаксической граф-схемы имеет вид:

$$X\omega_H x\omega_K Y \cong XxY$$

$$X\alpha\omega_H x\omega_K \beta Y \cong \begin{cases} XxY & \text{при } \alpha = \beta \\ \Omega & \text{при } \alpha \neq \beta \end{cases}$$

$$X\Omega Y \cong \Omega, \quad X \cong X.$$

По определению для каждого класса эквивалентности существует свой представитель. Будем считать путь в синтаксической граф-схеме Γ_G множеством цепочек вершин, являющихся классом эквивалентности по данному отношению эквивалентности $\cong$, а представлять его представителем данного класса. Тогда произведение путей P_1 и P_2 в синтаксической граф-схеме Γ_G — это множество цепочек $\mathrm{P}_1 \cdot \mathrm{P}_2$, которое также является путем в граф-схеме Γ_G.

Обозначим через P_x — путь $P_{\alpha_0 \alpha_1} \cdot P_{\alpha_1 \alpha_2} \cdot \ldots \cdot P_{\alpha_{n-1} \alpha_n}$, где $\alpha_{i-1} \underset{G}{\to} \alpha_i$, $m(\alpha_i) = \xi_i \ \ \forall i \in \{1,2,\ldots,k\}$ и $x = \xi_1 \xi_2 \ldots \xi_n$.

Тогда P_x — путь, порождающий слово x — это множество цепочек, состоящих из нетерминальных, терминальных вершин и символов ω_H, ω_K.

Двум терминальным вершинам α и β, принадлежащим синтаксической граф-схеме Γ_G, поставим в соответствие множество слов $L_{\Gamma_G}(\alpha,\beta)$, определяемое следующим образом:

1. $\varepsilon \in L_{\Gamma_G}(\alpha,\beta)$, если $\alpha = \beta$, в этом случае $\mathrm{P}_{\alpha\beta} \cong e$,
2. Если $x \in L_{\Gamma_G}(\alpha,\gamma)$, существует вершина β, такая что $\gamma \underset{G}{\to} \beta$, $m(\beta) = \xi$, и существует путь $\mathrm{P}_{\gamma\beta}$, такой что $\mathrm{P}_x \cdot \mathrm{P}_{\gamma\beta} \neq \Omega$ то $x\xi \in L_{\Gamma_G}(\alpha,\beta)$.

Обозначим $L_{\Gamma_G}(E_A) = \bigcup_{\beta \in K(A)} L_{\Gamma_G}(E_A, \beta)$. Если $A = S$, то $L_{\Gamma_G}(E_S)$ будем обозначать L_{Γ_G}.

Лемма 4.1. Слово x в алфавите терминалов T принадлежит множеству $L_{\Gamma_G}(\alpha,\beta)$ тогда и только тогда, когда существует последовательность терминальных вершин $\alpha_0, \alpha_1, \ldots, \alpha_k$ и последовательность путей $\mathrm{P}_1, \mathrm{P}_2, \ldots, \mathrm{P}_k$ в синтаксической граф-схеме Γ_G, такие что $\alpha_0 = \alpha$, $\alpha_k = \beta$, $m(\alpha_i) = \xi_i$, $\alpha_{i-1} \underset{G}{\to} \alpha_i$ $\mathrm{P}_i \cong \mathrm{P}_{\alpha_{i-1}\alpha_i}$, $\forall i \in \{1,2,\ldots,k\}$, $x = \xi_1\xi_2\ldots\xi_k$ и $\mathrm{P}_x \cong \mathrm{P}_1 \cdot \mathrm{P}_2 \ldots \mathrm{P}_k$, причем $P_x \neq \Omega$.

Строгое доказательство леммы можно найти в [1] и в брошюре [19].

Следствие 2.4. Слово x в алфавите T принадлежит множеству $L_G(E_A)$ тогда и только тогда, когда существует последовательность терминальных вершин $\alpha_0, \alpha_1, \ldots, \alpha_k$ в Γ_G и последовательность путей $\mathrm{P}_1, \mathrm{P}_2, \ldots, \mathrm{P}_k$ в Γ_G, такие что $\alpha_0 = E_A$, $\alpha_k \in K(A)$, $\alpha_{i-1} \underset{G}{\to} \alpha_k$, $m(\alpha_i) = \xi_i$, $\mathrm{P}_i \cong \mathrm{P}_{\alpha_{i-1}\alpha_i}$, $\forall i \in \{1,2,\ldots,k\}$, $x = \xi_1\xi_2\ldots\xi_k$ и $\mathrm{P}_x \cong \mathrm{P}_1 \cdot \mathrm{P}_2 \ldots \mathrm{P}_k$, причем существуют нетерминальные вершины $\beta_1, \beta_2, \ldots \beta_l$, в Γ_G, $\forall i \in \{1,2,\ldots,l\}$, что $\mathrm{P}_x \cdot \omega_K\beta_1 \cdot \omega_K\beta_2 \cdot \ldots \cdot \omega_K\beta_l = \alpha_1\alpha_2\ldots\alpha_k$

Множество L_G — это множество слов, порождаемых синтаксической граф-схемой Γ_G. Из вышесказанного можно заключить, что отношение достижимости на вершинах Γ_G отражает возможную последовательность букв в порождаемом слове.

Путь P_x содержит последовательность вершин в Γ_G, первый и последний элементы которой являются входной и конечной вершинами в Γ_S и каждый элемент достижим из предыдущего (если таковой имеется), а соответствующая последовательность меток вершин представляет буквы слова x.

5 Эквивалентное преобразование КСР-грамматик

При построении транслятора разработчики решают следующие *задачи: упрощение грамматики; регуляризация грамматики; разрешение конфликтов,* возникающих в процессе построения анализатора и связанных с языковой неоднозначностью (как синтаксической, так и семантической) и недетерминированностью магазинного автомата.

Упрощение грамматики — это предварительная подготовка грамматики, проверка на правильность записи регулярных выражений, формирование приведённой грамматики (well-formed);

Регуляризация грамматики — это необходимая часть полного цикла разработки транслятора, которая заключается в пошаговом выполнении базисных преобразований исходной грамматики в новую КСР-грамматику, причем такую, что из нее исключаются леворекурсивные и несамовставленные нетерминалы. Определяющие правила для этих нетерминалов исключаются также. В этом случае грамматика преобразуется в грамматику 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. Напомним, что вывод в КСР-грамматике заменяется вычислением значений соответствующих регулярных выражений. Алгоритмы исключения рассмотрены в работах [1; 2; 4]. При этом синтезированный МП-преобразователь может выродиться в конечно-автоматный. Далее дадим пример такого автоматического исключения несамовставленных нетерминалов для версии языка CIAO [23], выполненного в системе SynGT.

Правила грамматики языка CIAO

```
P : N E [A] [R] U W [D] L .
N : '<p_num>' '<p_nm>' $open .
E : 'EVENT' ( '<e_nm>' $e1 '(' ( $at1 ; '<tag>' ':' '<type>' $at4 ) ')' )# .
A : 'ACTION' ( '<a_nm>' $ac1 '(' ( $at1 ; '<tag>' ':' '<type>' $at4 ) ')' )# .
R : 'REQUEST' ( '<r_nm>' $r1 '(' ( $at1 ; '<tag>' ':' '<type>' $at4 ) ')'
   ':'  '<type>' $r2 )# .
U : 'STATE' ('<u_nm>' $st1 '->' (( '<e_nm>' $u1 '(' ($at1 ; '<val>' $at2) ')'
   ) ; ( 'after' '(' '<val>' 's' ')' $after )) ($ac2 ; Action) '->'
   ('<u_nm>' ; '<d_nm>') $st2)# $u2 .
W : 'WAGGLY' 'entry' '->' '->' '<u_nm>' $entry .
D : 'DECISION' ( '<d_nm>' $d1 '->' '(' '[' '<r_nm>' $c1 '(' ($at1 ; '<val>‘
   $at2) ')' ']' ($ac2 ; Action) '->' ( '<u_nm>' ; '<d_nm>' $st3 ) $st2 '|'
   $else ($ac2 ; Action) '->' ( '<u_nm>' ; '<d_nm>' $st3 ) $st2 ')' )# $d2 .
L : 'LINK' ( '<f_num>' '<p_num>' '<e_nm>' $lnk1 )#
   ( ; '<p_num>' '<t_num>' ( '<a_nm>' ; '<r_nm>' ) $lnk2 )# $close .
Action : '/' ( '<a_nm>' $ac3 '(' ( $at1 ; '<val>' $at2 ) ')' )#( ',' ) .
```

Грамматика CIAO (после эквивалентных преобразований) с одним правилом для нетерминала P

```
P : '<p_num>' , '<p_nm>' , $open , 'EVENT' ,
( '<e_nm>' , $e1 , '(' , ( $at1 ; '<tag>' , ':' , '<type>' , $at4 ) , ')' )*@ ,
( @ ; 'ACTION' , ( '<a_nm>' , $ac1 , '(' , ( $at1 ; '<tag>' , ':' , '<type>' , $at4 )
, ')' )*@ ) ,
( @ ; 'REQUEST' , ( '<r_nm>' , $r1 , '(' , ( $at1 ; '<tag>' , ':' , '<type>' , $at4 )
, ')' , ':' ,
'<type>' , $r2 )*@ ) ,'STATE' , ( '<u_nm>' , $st1 , '->' , ( '<e_nm>' , $u1 ,
'(' , ( $at1 ; '<val>' , $at2 ) , ')' ; 'after' , '(' , '<val>' , 's' , ')' , $after ) ,
( $ac2 ; '/' , ( '<a_nm>' , $ac3 , '(' , ( $at1 ; '<val>' , $at2 ) , ')' )*',' ) ,
```

```
    '->' , ( '<u_nm>' ; '<d_nm>' ) , $st2 )*@ , $u2 , 'WAGGLY' , 'entry' , '->' , '-
>' ,
    '<u_nm>' , $entry , ( @ ; 'DECISION' , ( '<d_nm>' , $d1 , '->' ,
    '(' , '[' , '<r_nm>' , $c1 , '(' , ( $at1 ; '<val>' , $at2 ) , ')' , ']' ,
    ( $ac2 ; '/' , ( '<a_nm>' , $ac3 , '(' , ( $at1 ; '<val>' , $at2 ) , ')' )*',' ) ,
    '->' , ( '<u_nm>' ; '<d_nm>' , $st3 ) , $st2 , '|' , $else ,
    ( $ac2 ; '/' , ( '<a_nm>' , $ac3 , '(' , ( $at1 ; '<val>' , $at2 ) , ')' )*',' ) ,
    '->' , ( '<u_nm>' ; '<d_nm>' , $st3 ) , $st2 , ')' )*@ , $d2 ) , 'LINK' ,
    ( '<f_num>' , '<p_num>' , '<e_nm>' , $lnk1 )*@ ,
    ( @ ; ('<p_num>' , '<t_num>' , ( '<a_nm>' ; '<r_nm>' ) , $lnk2 )*@ ) , $close
.
```

EOGram!

Заключение

В работе представлена схема автоматического построения анализаторов языков класса LL(1) с использованием синтаксических граф-схем. Обосновано понятие вывода в грамматиках в регулярной форме как класса эквивалентности на всех маршрутах в СГС. Подчёркнута важность эквивалентных преобразований, выполняемых автоматически. В качестве примера приведено эквивалентное преобразование с удалением несамовставленных нетерминалов в КСР-грамматике формального языка CIAO, выполненное в системе SynGT.

Литература

1. Fedorchenko L. Regularization of Context-Free Grammars. Saarbrucken: LAP LAMBERT Academic Publishing, 2011. 180 p.

2. Fedorchenko L., Baranov S. Equivalent Transformations and Regularization in Context-Free Grammars // Cybernetics and Information Technologies (CIT). Sofia, 2015. Vol. 14, No. 4. P. 11–28.

3. Ахо А. В., Сети Р., Ульман Д. Д. Компиляторы: принципы, технологии и инструменты. М.: Вильямс, 2001. 768 с.

4. Федорченко Л. Н. Регуляризация контекстно-свободных грамматик на основе эквивалентных преобразований синтаксических граф-схем // Труды СПИИРАН. 2010. Вып. 4(15). С. 213–230.

5. Early J. Ambiguity and precedence in syntax description // Acta Informatica. 1975. Vol. 4, Iss. 2. P. 183–192.

6. Федорченко Л. Н. Эквивалентность как отношение подобия в трансляции языков // Вестник Бурятского государственного университета. 2014. № 9–3. С. 49–53.

7. Mc Naughton R. The development of formal language theory since 1956 // Internat. J. Found. Comput. Sci. 1990. Vol. 1, No. 4. P. 355–368.

8. Paull M. C., Unger S. H. Structural equivalence of context-free grammars // Journal of Computer and System Sciences. 1968. Vol. 2, Iss. 4. P. 427–463.

9. Paull M. C., Unger S. H. Structural Equivalence of LL-k Grammars // IEEE Conference Record of Ninth Annual Symposium on Switching and Automata Theory. 1968. P. 160–175.

10. Prather R. E. Regular Expressions for Program Computations // American Mathematical Monthly. 1997. Vol. 104, No. 2. P. 120–130.

11. Kenichi T., Tadao K. A Result on the Equivalence Problem for Deterministic Pushdown Automata // J. Comput. Syst. Sci. 1976. Vol. 13, Iss. 1. P. 38–50.

12. Ginsburg S. A survey of grammar forms // Acta Cybernetica. 1977. Vol. 3, No. 4. P. 269–280.

13. Ginsburg S., Harrison M. A. Bracketed context free languages // J. Comput. Syst. Sci. 1967. Vol. 1, Iss. 1. P. 1–23.

14. Korenjak A. J., Hopcroft J. E. Simple deterministic languages // IEEE Conf. Record of 7th Annual Symposium on Switching and Automata Theory. 1966. P. 36–46.

15. Salomaa K., Yu S. Decidability of structural equivalence of E0L grammars // Theoretical Computer Science. 1991. Vol. 82, Iss. 1. P. 131–139.

16. Cremers A. B., Ginsburg S. Context-free grammar forms // J. Comput. Syst. Sci. 1975. Vol. 11, Iss. 1. P. 86–117.

17. Hotz G. Normal-form transformations of context-free grammars // Acta Cybernetica. 1978. Vol. 4, No. 1. P. 65–84.

18. Hunt III H. B., Rosenkrantz D. J. Complexity of grammatical similarity relations // Procs of a Conf. on Theoretical Computer Science. Waterloo, Canada, 1977. P. 139–145.

19. Федорченко Л. Н., Мартыненко Б. К. Эквивалентные преобразования КСР-грамматик в регулярной форме в практике построения языковых процессоров. Ч. 1. Определение и распознавание КСР-языков посредством синтаксических граф-схем. Л., 1983. 46 с.

20. Gray J. N., Harrison M. A. On the covering and. reduction problems for context free grammars // Journal of the Association for Computing Machinery. 1972. Vol. 19, Issue 4. P. 675–698.

21. Martynenko B. K. Regular Languages and CF Grammars // Computer Tools in Education. 2012. No. 1. P. 14–20.

22. Федорченко Л. Н. Алгоритмы построения состояний анализатора для КСР-языка // Вестник Бурятского государственного университета. Математика, информатика. 2016. № 4. С. 23–33.

23. Федорченко Л. Н., Афанасьева И. В. Метод описания систем со сложным поведением на принципах обобщённых автоматов // Вестник Бурятского государственного университета. Математика, информатика. 2018. № 4. С. 22–36. DOI: 10.18101/2304-5728-2018-4-22-36.

HOMOMORPHISM OF EQUIVALENT TRANSFORMATIONS OF GRAMMARS IN PARSER GENERATION

Lyudmila N. Fedorchenko
Cand. Sci. (Engineering), A/Prof., Senior Researcher,
St. Petersburg Institute for Informatics and Automation RAS (SPIIRAS),
39 14th Line of V. O., St. Petersburg 199178, Russia

St. Petersburg State University
7/9 Universitetskaya Emb., St. Petersburg 199034, Russia
E-mail: lnf@iias.spb.su

When constructing a translator, as a rule, it is necessary to apply the equivalent transformations of grammar of the implemented language, which convert the syntactic specification of the language into a form that allows for automatic or manual implementation of the source language, and solves the problem of constraint satisfaction of the chosen parsing technique. These problems arise both because of the variety of ways of defining the implemented languages, and because of the language ambiguity or nondeterminism of the recognizing automaton. Each transformation performed on translational CF-grammars can be defined as a likelihood ratio (homomorphism), determined on classes of grammars. Such relations preserve the semantic meaning of grammatical structures in equivalent syntax transformations, since the ultimate purpose of translation is to obtain a sequence of actions prescribed by the computing environment.

The article presents a brief overview of various types of likelihood ratio over grammars used in constructing translators since the 1970s.

A flow chart for LL parsing with use of syntactic graph scheme (SGS) is presented. We have given a formal notion of a route (path) in the SGS and an example of the automatic CFR-grammar transformation of CIAO formal language in SynGT (Syntax Graph Transformations) system of equivalent transformations.

Keywords: equivalent transformations of grammars; likelihood ratio; cover homomorphism; CF-grammar in a regular form (CFR-grammar); syntactic graph scheme (flow-chart); grammar regularization.

References

1. Fedorchenko L. *Regularization of Context-Free Grammars.* Saarbrucken: LAP LAMBERT Academic Publishing, 2011.

2. Fedorchenko L., Baranov S. Equivalent Transformations and Regularization in Context–Free Grammars. *Cybernetics and Information Technologies (CIT).* V. 14. No. 4. Pp.11–28.

3. Aho A. V., Sethi R., Ullman J. D. *Compilers, Principles, Techniques, and Tools.* Addison-Wesley, 1986.

4. Fedorchenko L. N. Regulyarizatsiya kontekstno-svobodnykh grammatik na osnove ekvivalentnykh preobrazovanii sintaksicheskikh graf-skhem [Regularization of Context-Free Grammars Based on Equivalent Transformations of Syntactic Graph-Schemes]. *Trudy SPIIRAN.* 2010. V. 4 (15). Pp. 213–230. ISSN 2078-9181.

5. Early J. Ambiguity and Precedence in Syntax Description. *Acta Informatica.* 1975. No. 4. Iss. 2. Pp. 183–192.

6. Fedorchenko L. N. Ekvivalentnost kak otnoshenie podobiya v translyacii yazykov [Equivalence as a Likelihood Ratio in Language Translation]. *Vestnik Buryatskogo gosudarstvennogo universiteta.* 2014. No. 9–3. Pp. 49–53.

7. Mc Naughton R. The Development of Formal Language Theory Since 1956. *Int. J. Found. Comput. Sci.* 1990. V. 1. No. 4. Pp. 355–368.

8. Paull M., Unger S. Structural Equivalence of Context-Free Grammars. *Journal of Computer and System Sciences.* 1968. Vol. 2. Pp. 427–463.

9. Paull M. C., Unger S. H. Structural Equivalence of LL-k Grammars. *IEEE Conference Record of Ninth Annual Symposium on Switching and Automata Theory.* 1968. Pp. 160–175.

10. Prather R. E. Regular Expressions for Program Computations. *American Mathematical Monthly.* 1997. V. 104. No. 2.

11. Kenichi Taniguchi, Tadao Kasami. A Result on the Equivalence Problem for Deterministic Pushdown Automata. *J. Comput. Syst. Sci.* 1976. V. 13. No. 1. Pp. 38–50.

12. Ginsburg S. A Survey of Grammar Forms. *Acta Cybernetica.* 1977. V. 3. No. 4. Pp. 269–280.

13. Ginsburg S., Harrison M. A. Bracketed Context Free Languages. *J. Comput. Syst. Sci.* 1967. V. 1, Iss. 1. Pp. 1–23.

14. Korenjak A. J., Hopcroft J. E. Simple Deterministic Languages. *IEEE Conf. Record of 7th Annual Symposium on Switching and Automata Theory.* 1966. Pp. 36–46.

15. Salomaa K., Yu S. Decidability of Structural Equivalence of E0L Grammars. *Theoretical Computer Science.* 1991. V. 82. Iss. 1. Pp. 131–139.

16. Cremers A. B., Ginsburg S. Context-Free Grammar Forms. *J. Comput Syst. Sci.* 1975. Vol. 11, Iss.1. Pp. 86–117.

17. Hotz G. Normal-form Transformations of Context-Free Grammars. *Acta Cybernetica.* 1978. Vol. 4. No. 1. Pp. 65–84.

18. Hunt III H. B., Rosenkrantz D. J. Complexity of Grammatical Similarity Relations. *Proc. Conf. on Theoretical Computer Science* (Waterloo, Canada). 1977. Pp. 139–145.

19. Fedorchenko L. N., Martynenko B. K. *Ekvivalentnye preobrazovaniya KSR grammatik v regulyarnoi forme v praktike postroeniya yazykovykh protsessorov. Chast pervaya. Opredelenie i raspoznavanie KSR-yazykov posredstvom sintaksicheskikh grafskhem* [Equivalent Transformations of CF-Grammars in a Regular Form in the Practice of Constructing Language Processors. Part one. Definition and Recognition of CFR-languages by Means of Syntactic Graph-Schemes]. Leningrad, 1983.

20. Gray J. N., Harrison M. A. On the Covering and Reduction Problems for Context Free Grammars. *Journal of the ACM.* 1972. V. 19. Iss. 4. Pp. 675–698.

21. Martynenko B. K. Regular Languages and CF-Grammars. *Computer Tools in Education.* 2012. No. 1. Pp. 14–20.

22. Fedorchenko L. N. Algoritmy postroeniya sostoyanii analizatora dlya KSR-yazyka [Algorithms for Construction of Analyzer States for CFR-Language]. *Vestnik Buryatskogo gosudarstvennogo universiteta. Matematika, informatika.* 2016. No. 4. Pp. 23–33.

23. Fedorchenko L. N., Afanasyeva I. V. Metod opisaniya sistem so slozhnym povedeniem na printsipakh obobshchennykh avtomatov [A Method for Describing Systems with Complex Behavior on the Principles of Generalized Automata]. *Vestnik Buryatskogo gosudarstvennogo universiteta. Matematika, informatika.* 2018. No. 4. Pp. 22–36.