УДК 004.423.25

А.С. Станкевич

ИСПОЛЬЗОВАНИЕ АЛГОРИТМОВ АНАЛИЗА ЛЕВОКОНТЕКСТНЫХ ТЕРМИНАЛЬНЫХ ГРАММАТИК В ЗАДАЧАХ АВТОМАТИЧЕСКОГО ТЕСТИРОВАНИЯ ПРОГРАММ

Станкевич А.С. **Использование алгоритмов анализа левоконтекстных терминальных грамматик в задачах автоматического тестирования программ.**
Аннотация. При построении сценариев тестирования программ может возникнуть потребность анализа их свойств. Для формального задания сценариев можно применить левоконтекстные терминальные грамматики. В работе доказывается эквивалентность порождающей мощности левоконтекстных терминальных грамматик и контекстно-свободных грамматик и рассматриваются алгоритмы анализа левоконтекстных грамматик, которые могут быть использованы для анализа свойств сценариев тестирования.
Ключевые слова: алгоритм, сценарий тестирования, тестирование программ, формальная грамматика, левоконтекстная грамматика.

Stankevich A.S. **Algorithms of analyzing left context terminal grammars used for automated programs testing.**
Abstract. After designing testing scripts, sometimes it is needed to analyze their properties. To make script specification formal, one can use left context terminal grammars.
The article gives proof of left context terminal grammars and context free grammars equivalence and develops algorithms for analyzing properties of such grammars that are useful in analyzing testing script properties.
Keywords: algorithm, testing script, program testing, formal grammar, left context grammar.

1. Введение. Проблема тестирования программ является актуальной в различных областях прикладной математики и информатики. Помимо непосредственно тестирования программ в рамках разработки программного обеспечения проблема тестирования возникает, например, в процессе обучения при проверке правильности выполнения заданий.

В соответствии с теоремой Успенского—Райса [11], анализ исходного кода программы не позволяет сделать выводов о ее поведении. Для решения этой проблемы используются два основных метода:

1) тестирование посредством запуска программы либо ее подпрограмм на наборе тестов (на таком подходе, в частности, основывается широко распространенное модульное тестирование) [8];

2) построение по программе формальной математической модели и дальнейший анализ и доказательство ее свойств (верификация) [5].

Рассмотрим процесс тестирования программы. Сценарий тестирования может сам по себе иметь достаточно сложную логику и содер-

жать нетривиальные зависимости между действиями, выполняемыми при различных исходах проведенных тестов. При построении сценариев для тестирования программ в автоматическом режиме возникает потребность в анализе свойств получившихся сценариев. Для этого необходимо использовать формальный метод записи сценариев. Одним из таких методов является задание сценариев с использованием левоконтекстных терминальных грамматик [7, 9].

В работе показано, что этот класс грамматик эквивалентен по порождающей мощности контекстно-свободным и любая левоконтекстная терминальная грамматика может быть автоматически преобразована в контекстно-свободную, порождающую тот же язык. При этом нетерминалы представляют этапы тестирования программы, терминалы соответствуют результатам тестирования, а выведенное слово содержит исходы всех этапов тестирования и представляет собой отчет о тестировании [9]

Задание сценария тестирования с использованием левоконтекстной терминальной грамматики позволяет свести вопросы о свойствах сценария к вопросам о языке, порождаемом грамматикой и о процессах вывода в соответствующей грамматике. В данной работе представлены алгоритмы анализа левоконтекстных терминальных грамматик и соответствующих им контекстно-свободных грамматик, которые используются для анализа свойств сценариев тестирования.

2. Левоконтекстные терминальные грамматики. Рассмотрим специальный класс контекстно-зависимых грамматик, вывод строки из терминала в которых можно естественным образом сопоставить с процессом выполнения сценария, в котором очередное действие может зависеть от исхода предыдущих действий.

Определение 1. Левоконтекстной грамматикой называется формальная грамматика, в которой все правила имеют вид $\alpha S \to \alpha\xi$, где α и ξ могут состоять из терминалов и нетерминалов. Строка α называется левым контекстом правила.

Известно, что левоконтекстные грамматики эквивалентны по порождающей мощности произвольным формальным грамматикам [3]. Как следствие, множество языков, порожденных левоконтестными грамматиками, совпадает с множеством всех перечислимых языков [11]. Однако если потребовать от грамматики дополнительного свойства, что α состоит только из терминалов, то оказывается, что класс порождаемых ими языков сужается, и порождаются только контекстно-свободные языки.

Определение 2. Левоконтекстная грамматика называется *терминальной*, если во всех правилах левый контекст состоит только из терминалов.

Теорема 1. Язык, порождаемый левоконтекстной терминальной грамматикой, является контекстно-свободным.

Доказательство. Рассмотрим все левые контексты. По предположению они состоят только из терминалов. Найдем среди них самый длинный и обозначим его длину за k. Тогда, если из стартового символа выведена сентенциальная форма $\tau X\alpha$, где τ состоит только из терминалов, то для принятия решения, какое правило можно применить к символу X, требуется знать последние k символов строки τ.

Для каждого нетерминала введем $(|S|+1)^{2k}$ нетерминалов. Будем обозначать их как $[pXq]$, где p и q — строки из терминалов длины не больше k. Смысл этого нетерминала будет состоять в следующем: если он встречается в сентенциальной форме, то перед ним следует строка p, а после того, как из него будет выведена строка из терминалов, она будет заканчиваться на q. Обозначим соответствующее множество нетерминалов как $[*X*]$

Теперь преобразуем грамматику. Рассмотрим правило $\alpha A \to \alpha\xi$. Заменим его сначала на множество правил вида $[pAq] \to \xi$, где p заканчивается на α, а q — произвольная строка. Теперь обратим внимание на правую часть ξ. Заменим в ней каждый нетерминал B на все возможные нетерминалы $[pBq]$. Теперь оставим среди получившегося множества правил те, которые являются согласованными.

Правило $[pAq] \to \xi$ будем называть согласованным, если выполнены следующие условия:

1) пусть $[rBs]$ — первый нетерминал в правой части: $[pAq] \to \tau[rBs]\zeta$, тогда должно выполняться условие: r — суффикс $p\tau$.

2) пусть $[rBs]$ и $[uCv]$ — два нетерминала в правой части, между которыми расположены только терминалы: $[pAq] \to \zeta[rBs]\tau[uCv]\eta$. тогда должно выполняться условие: u — суффикс $s\tau$ (τ может быть и пустой строкой).

3) пусть $[rBs]$ — последний нетерминал в правой части: $[pAq] \to \zeta[rBs]\tau$, тогда q — суффикс $s\tau$.

4) если правая часть не содержит нетерминалов: $[pAq] \to \tau$, то должно выполняться условие: q — суффикс $p\tau$.

Также введем новый стартовый символ S' и добавим правила $S' \to [St]$ для всех возможных строк t.

Получившаяся грамматика порождает тот же язык, что и исходная и является контекстно-свободной.

Отметим, что все правила преобразования формальны и носят синтаксический характер, поэтому преобразование левоконтекстной терминальной грамматики в эквивалентную ей контекстно-свободную легко реализовать автоматически.

Вообще говоря, в отличие от контекстно-свободных грамматик, в левоконтекстных грамматиках нельзя потребовать, чтобы при выводе всегда применялось правило для самого левого нетерминала в сентенциальной форме. Однако для случая левоконтекстных терминальных грамматик это требование не изменяет выводимого языка. Действительно, возможность применения правила зависит только от строки терминалов слева от нетерминала, а слева от самого левого нетерминала в сентенциальной форме множество терминалов изменяться не будет. Назовем соответствующий вывод *левосторонним* по аналогии с контекстно-свободными грамматиками.

Определение 3. Левоконтекстная терминальная грамматика называется *контекстно-однозначной*, если для всех нетерминалов A выполняется следующее свойство. Рассмотрим все правила, в которых A встречается в левой части: $\alpha A \to \alpha\xi$.

Тогда выполнено одно из двух:

1) для любых двух таких правил $\alpha A \to \alpha\xi$ и $\beta A \to \beta\eta$ выполнены условия: α не является суффиксом β и β не является суффиксом α;

2) во всех таких правилах $\alpha = \varepsilon$ и правые части начинаются с различных терминалов.

Важным свойством контекстно-однозначных грамматик является однозначность левостороннего вывода любого слова.

Теорема 2. Пусть Γ — контекстно-однозначная левоконтекстная терминальная грамматика и w — слово из терминалов. Тогда существует не более одного левостороннего вывода w в Γ.

Доказательство. Пусть найдется два различных левосторонних вывода для некоторого слова w. Рассмотрим первое место, в котором они различаются:

$$S \Rightarrow^* zA\xi \Rightarrow z\gamma\xi \Rightarrow^* w,$$
$$S \Rightarrow^* zA\xi \Rightarrow z\delta\xi \Rightarrow^* w.$$

Здесь строка z состоит только из терминалов.

Тогда либо λ и δ начинаются с различных терминалов (что невозможно), либо было применено два различных правила с непустым левым контекстом:

$$\alpha A \to \alpha\gamma,$$
$$\beta A \to \beta\delta.$$

Однако в этом случае строки α и β должны оба быть суффиксами z. Следовательно, одна из них является суффиксом другой, что противоречит определению контекстно-однозначной грамматики.

Отметим, что контекстная однозначность является синтаксическим свойством и может быть легко проверена автоматически (в отличие от полноценной однозначности, проверка которой для контекстно-сводобных грамматик — алгоритмически неразрешимая задача [11]).

Сформулируем некоторые свойства контекстно-свободной грамматики, полученной из левоконтекстной терминальной с помощью процесса, описанного в доказательстве теоремы 1.

Предложение 1. Пусть нетерминал A левоконтекстной терминальной грамматики встречается в левой части только в правилах вида $A \to \xi$, где ξ состоит только из терминалов. Тогда соответствующие ему нетерминалы $[pAq]$ контекстно-свободной грамматики Γ' также встречаются в левой части только правил вида $[pAq] \to \xi$, где ξ состоит только из терминалов, причем множество правых частей соответствующих правил в Γ' совпадает с множеством правых частей исходных правил Γ.

Предложение 2. Пусть исходная левоконтекстная терминальная грамматика Γ является контекстно однозначной. Тогда соответствующая ей контекстно-свободная грамматика Γ' является однозначной.

Доказательство. Заметим, что существует взаимно-однозначное соответствие между левосторонними выводами в исходной грамматике и левосторонними выводами в полученной контекстно-свободной грамматике. Следовательно, если существует не более одного левостороннего вывода любого слова в Γ, то существует не более одного левостороннего вывода любого слова в Γ', т.е. Γ' однозначна.

3. Алгоритмы анализа контекстно-свободных грамматик с возможностью бесконечных цепочек порождений. Отметим особенности, которые возникают при анализе сценариев тестирования, опи-

санных с использованием левоконтекстных грамматик, по сравнению с анализом свойств контекстно-свободных грамматик. При анализе контекстно-свободных грамматик обычно анализируют только выводы слов из терминалов, и соответствующие им деревья разбора. При анализе сценария тестирования ограничиваться этим нельзя, поскольку тестирование может не завершиться, что соответствует бесконечной цепочке порождений в рассматриваемой грамматике. Из-за этого часто приходится анализировать именно левосторонние выводы, поскольку от порядка применения правил к нетерминалам начинает зависеть результат, а процесс тестирования, описанный в работе [9], применяет правило к самому левому нетерминалу в сентенциальной форме. Кроме того, многие алгоритмы приходится адаптировать для случаев бесконечного вывода.

Рассмотрим алгоритмы анализа контекстно-свободных грамматик, которые используются для верификации свойств сценариев тестирования.

Свойство 1. Форсирование одним нетерминалом другого.

По двум заданным нетерминалам A и B в заданной контекстно-свободной грамматике Γ выяснить: верно ли, что если в выводе слова, состоящего только из терминалов, встречается A, то встретится и B. Будем говорить, что в этом случае A форсирует B.

Алгоритм проверки свойства 1. Удалим в грамматике нетерминал B и все правила, в которых он встречается. В получившейся грамматике удалим все бесполезные символы. Обозначим полученную контекстно-свободную грамматику как Γ'.

Предложение 3. A форсирует B если и только если после указанных преобразований A оказался удален из грамматики.

Доказательство. Пусть A не был удален из грамматики. Тогда он не является бесполезным, следовательно, в Γ' существует вывод слова из терминалов стартового символа, в котором используется нетерминал A. Однако в этом выводе не встречается B, следовательно, A не форсирует B.

Пусть A не форсирует B. Тогда найдется вывод в грамматике Γ, содержащий A, но не содержащий B. Ни одно из правил, использованных в этом выводе, не содержит B, и, следовательно, не было удалено при преобразовании Γ в Γ'. Все нетерминалы, использованные в этом выводе, являются достижимыми и порождающими, следовательно, рассматриваемый вывод является корректным выводом в Γ'. Поэтому нетерминал A не удален при построении Γ'. Полученным противоречием предложение доказано.

Приведенный алгоритм можно обобщить, если вместо нетерминалов A и B в предыдущем рассуждении использовать некоторые множества нетерминалов U и V.

Свойство 2. Форсирование множеством нетерминалов множества нетерминалов.

Пусть заданы множества нетерминалов U и V в некоторой контекстно-свободной грамматике Γ. Требуется выяснить: верно ли, что если в выводе слова, состоящего только из терминалов, встречается нетерминал из U, то встретится и нетерминал из V. Будем говорить, что в этом U форсирует V.

Алгоритм проверки свойства 2. Алгоритм аналогичен алгоритму проверки свойства 1. Отличие состоит в том, что необходимо удалить из грамматики все нетерминалы из множества V и проверить, что в результирующей грамматике после удаления множества бесполезных символов остается хотя бы один нетерминал из U. Доказательство корректности алгоритма выполняется аналогично доказательству предложения 3.

Рассмотренные алгоритмы успешно справляются с анализом форсирования одного нетерминала другим (одной альтернативы другой) при выводе слова в контекстно-свободных грамматиках. Однако при анализе сценариев тестирования возможно, что тестирование не окажется завершено. При этом, тем не менее, вопрос о форсировании одного действия другим может быть актуален. Поэтому при анализе сценариев тестирования необходимо учитывать возможность бесконечных выводов.

Свойство 3. Усиленное форсирование одним нетерминалом другого.

Будем говорить, что нетерминал A усиленно форсирует нетерминал B, когда для любой конечной или бесконечной левосторонней цепочки порождений выполняется условие: если встречается нетерминал A, то встречается и нетерминал B.

Алгоритм проверки свойства 3. Рассмотрим грамматику Γ. Для каждого нетерминала C выясним две характеристики. Будем называть нетерминал C *порождающим без* B, если из C можно вывести строку из терминалов, не используя нетерминал B. Будем называть нетерминал C порождающим без B через A, если из C можно вывести строку из терминалов, не используя нетерминал B и используя нетерминал A.

Для того чтобы определить, обладает ли нетерминал C этими свойствами, удалим из грамматики нетерминал B и, считая терминал C стартовым терминалом грамматики, удалим все недостижимые и непорождающие символы. Нетерминал C является порождающим без B, если он не был удален из грамматики. Нетерминал C является порождающим без B через A, если оба нетерминала A и C не были удалены из грамматики.

Теперь построим ориентированный граф, в котором каждому нетерминалу грамматики будет соответствовать вершина. Рассмотрим правила $C \to \xi$, где в ξ не встречается нетерминал B. Для каждого нетерминала D в ξ, перед которым встречаются только порождающие без B нетерминалы, проведем ребро из C в D. При этом, если перед D встречается нетерминал, порождающий B через A, либо если ξ содержит A, пометим это ребро.

Обозначим получившийся граф как $F_B(G)$.

Теорема 3. Нетерминал A грамматики Γ усиленно форсирует нетерминал B, если и только если выполнены следующие два условия:

1) A форсирует B;

2) в графе $F_B(G)$ не существует бесконечного пути, начинающегося в вершине S и проходящего через помеченное ребро, где S — начальный символ грамматики Γ.

Доказательство. Пусть неверно, что A сильно форсирует B. Тогда выполнено одно из двух условий: 1) либо существует конечный вывод, в котором встречается A, но не встречается B; 2) либо имеет место бесконечная последовательность порождений, в которой встречается A, но не встречается B.

В первом случае A не форсирует B, и поэтому не выполнено условие 1.

Во втором случае рассмотрим бесконечную цепочку порождений. Рассмотрим соответствующее ей (бесконечное) дерево разбора. В этом дереве все степени вершин конечны, поэтому по лемме о бесконечном дереве в нем существует бесконечный путь P. Поскольку рассматривается левосторонний вывод, то все нетерминалы в вершинах слева от этого пути порождают строки из терминалов. Тогда для любой пары отец—сын X — Y в дереве на рассматриваемом пути в графе $F_B(G)$ найдется ребро $X \to Y$. Следовательно, в графе существует бесконечный путь, соответствующий рассматриваемому пути в дереве. Дока-

жем, что на этом пути встречается помеченное ребро. Действительно, в дереве встречается нетерминал A. Пусть A встречается не на пути P. Рассмотрим X — самую нижнюю вершину на P, содержащую A в поддереве. Ребро, которое ведет из X в следующую вершину на пути P, будет помеченным в $F_B(G)$. Если же A встречается на P, то ребро, ведущее в A, будет помеченным.

Обратно, пусть A не форсирует B или в графе $F_B(G)$ существует бесконечный путь, содержащий помеченное ребро. В первом случае существует конечный вывод, содержащий A, но не содержащий B. Во втором случае рассмотрим этот путь и построим соответствующий ему бесконечный вывод. Начнем из стартового нетерминала. Инвариантом будет то, что сентенциальная форма имеет вид $\tau X \upsilon$, где τ состоит только из терминалов, а X — нетерминал, которому соответствует текущая вершина. Рассмотрим очередное ребро пути $X \to Y$. Пусть оно было добавлено в граф благодаря правилу $X \to \xi Y \zeta$. Применим это правило, затем породим все нетерминалы из ξ строки из терминалов, не используя нетерминал B (при этом, если какой-либо из них является порождающим без B через A, то породим строку из терминалов с использованием A). Затем переместимся в вершину Y. Построенный таким образом бесконечный вывод не содержит B, но содержит A, который встретится в выводе при рассмотрении помеченного ребра.

Теорема доказана.

Покажем, как проверить, что в графе существует бесконечный путь, начинающийся в стартовой вершине и проходящий по помеченному ребру. Для каждого помеченного ребра с использованием поиска в глубину проверим, достижим ли из него цикл [6]. После этого проверим, существует ли помеченное ребро, из которого достижим цикл, и которое, в свою очередь, достижимо из стартовой вершины.

Свойство 4. Усиленное форсирование одним множеством нетерминалов другого.

Множество нетерминалов U усиленно форсирует множество нетерминалов V, если в любой (конечной или бесконечной) левосторонней цепочке порождений выполняется условие: если встречается нетерминал из множества U, то встречается и нетерминал из множества V.

Алгоритм проверки свойства 4. Алгоритм строится аналогично алгоритму для проверки свойства 3. Для этого в приведенном алгоритме необходимо удалять из грамматики все вершины, соответству-

ющие нетерминалам из V, и аналогично рассматривать нетерминалы, порождающие без V через U, для которых существует вывод, содержащий хотя бы один нетерминал из U и не содержащий терминалов из V.

Аварийное завершение тестирования, когда для самого левого нетерминала не существует ни одного правила с соответствующим левым контекстом, при рассмотрении эквивалентной контекстно-свободной грамматики соответствует нетерминалу, для которого нет вообще ни одного правила. Такие нетерминалы будем называть *тупиковыми*.

Определение 4. Нетерминал в контекстно-свободной грамматике называется *тупиковым*, если не существует ни одного правила, в котором он является левой частью.

Определение 5. Нетерминал в контекстно-свободной грамматике называется *потенциально тупиковым*, если из него существует частичное порождение, завершающееся сентенциальной формой, в которой самый левый нетерминал является тупиковым.

Свойство 4. Терминал является потенциально тупиковым.

Алгоритм проверки свойства 4. Алгоритм аналогичен алгоритму поиска порождающих нетерминалов.

Пометим все тупиковые нетерминалы как потенциально тупиковые. Затем будем последовательно выполнять следующую операцию. Если нетерминал Y является потенциально тупиковым и в грамматике есть правило $X \to \xi Y \zeta$, где X пока не отмечен как потенциально тупиковый, а в ξ все нетерминалы являются порождающими, то пометим нетерминал X как потенциально тупиковый. Когда ни одной такой ситуации больше нет, то отмечены в точности все потенциально тупиковые нетерминалы.

Предложение 4. Приведенный алгоритм корректно строит множество потенциально тупиковых нетерминалов.

Доказательство. Пусть некоторый нетерминал X был отмечен как потенциально тупиковый. Докажем по индукции, что он действительно является таковым. В качестве параметра индукции будем использовать число действий по добавлению отметки нетерминалу. Исходно помечены все тупиковые нетерминалы, которые по определению являются потенциально тупиковыми.

Пусть некоторый нетерминал X был помечен благодаря нетерминалу Y и правилу $X \to \xi Y \zeta$. Нетерминал Y является потенциально тупиковым в соответствии с предположением индукции. Рассмот-

рим следующее порождение: $X \Rightarrow \xi Y \zeta \Rightarrow^* \tau Y \zeta$, где τ состоит только из терминалов (такое порождение существует, поскольку все нетерминалы из ξ являются порождающими. Поскольку Y является потенциально тупиковым, то для него существует порождение $Y \Rightarrow^* \sigma Q \chi$, где Q является тупиковым, а σ состоит только из терминалов. Но тогда, продолжая рассмотренное порождение, получим $X \Rightarrow \xi Y \zeta \Rightarrow^* \tau Y \zeta \Rightarrow^* \tau \sigma Q \chi \zeta$, где Q является тупиковым, а $\tau\sigma$ состоит только из терминалов. Это по определению означает, что X является потенциально тупиковым.

Пусть, наоборот, X является потенциально тупиковым. Предположим от противного, что он не был помечен описанным алгоритмом. Среди таких нетерминалов найдем тот, для которого вывод $X \Rightarrow^* \tau Q \zeta$ для тупикового нетерминала Q и строки τ, состоящей только из терминалов, является самым коротким. Рассмотрим первое правило в этом выводе: $X \to \alpha$, и представим α как $\xi Y \chi$, где Y — нетерминал, из которого выводится Q. Тогда Y является потенциально тупиковым, причем вывод сентенциальной формы, в которой самый левый нетерминал является тупиковым для Y, является более коротким, чем для X. Следовательно, Y помечен как потенциально тупиковый приведенным алгоритмом. Однако тогда, поскольку из ξ выводится префикс τ и, следовательно, все нетерминалы в ξ являются порождающими, алгоритм должен был пометить X как потенциально тупиковый.

Полученным противоречием предложение доказано.

Определение 5. По заданной контекстно-свободной грамматике Γ построим граф следующим образом: вершинами графа будут нетерминалы грамматики, из вершины A в вершину B ведет ребро, если в грамматике существует правило $A \to \alpha B \beta$ для некоторых α и β. Этот граф называют *графом порождений* заданной грамматики.

Граф порождений при анализе сценариев мог бы соответствовать возможности выполнения одного действия в рамках выполнения другого. Однако бывают ситуации, когда, несмотря на наличие пути из A в B в графе порождений, в рамках действия A невозможно выполнение действия B. Это связано с тем, что возможно зацикливание выполнения сценария в рамках одного из действий до выполнения дей-

ствия B. Для того чтобы разрешить подобные ситуации, модифицируем метод построения графа порождений.

Проведем ребро из A в B, только если в грамматике есть правило $A \to \alpha B \beta$, где все нетерминалы в α являются порождающими. При таком построении путь в графе соответствует возможному порождению одним нетерминалом другого (т.е. выполнению одного действия в рамках другого) при левостороннем выводе даже при возможности бесконечных выводов.

В дальнейшем будем рассматривать именно такой граф порождений (это не противоречит обычному анализу контекстно-свободных грамматик, поскольку при их анализе обычно непорождающие нетерминалы удаляются из грамматики).

Теорема 4. Из нетерминала A достижим нетерминал B в графе порождений тогда и только тогда, когда существует левосторонний вывод из A сентенциальной формы ξ, в которой B является самым левым нетерминалом.

*Доказательство***.** Пусть из A достижим B. Применим индукцию по длине пути. Если длина пути равна нулю, то $A = B$ и предложение доказано. В ином случае рассмотрим первый нетерминал на пути. Пусть это C. По правилам построения графа это значит, что в грамматике присутствует правило $A \to \alpha C \beta$, где все нетерминалы в α являются порождающими. Рассмотрим вывод $A \Rightarrow \alpha C \beta \Rightarrow^* \tau C \beta$, где τ состоит только из терминалов. По предположению индукции из C можно вывести форму, где B является самым левым нетерминалом, сделаем это. Получим $A \Rightarrow \alpha C \beta \Rightarrow^* \tau C \beta \Rightarrow^* \tau \upsilon B \gamma \beta$, где $\tau\upsilon$ состоит только из терминалов.

Обратно, рассмотрим вывод $A \Rightarrow^* \tau B \alpha$, где τ состоит только из терминалов. Снова проведем индукцию по длине вывода. Если $A = B$, то все доказано. В ином случае рассмотрим первое примененное правило: $A \to \beta$. Пусть B выводится из нетерминала C в β: $A \Rightarrow^* \beta_1 C \beta_2$. Поскольку из β_1 выводится префикс τ, все нетерминалы в β_1 являются порождающими, и поэтому в графе порождений есть ребро из A в B. По индукционному предположению в графе порождений существует путь из C в B, в итоге заключаем, что в графе есть путь из A в B, что и требовалось доказать.

4. Заключение. При задании сценариев тестирования с использованием левоконтекстных терминальных грамматик задачи верифика-

ции различных свойств сценариев естественным образом переносятся на анализ грамматики сценария и языка, порождаемого этой грамматикой.

Эквивалентность левоконтекстных терминальных грамматик контекстно-свободным, а также простая синтаксическая процедура получения эквивалентной контекстно-свободной, позволяют для решения задач, связанных исключительно с порождаемым языком, вместо разработки алгоритмов для левоконтекстных терминальных грамматик, разрабатывать алгоритмы для контекстно-свободных грамматик, либо использовать существующие алгоритмы. В некоторых случаях алгоритмы необходимо адаптировать на случай бесконечных цепочек выводов, что сделано в разделе 3.

Рассмотрим некоторые примеры анализа свойств сценариев с использованием алгоритмов проверки свойствам соответствующих грамматик, рассмотренных в разделе 3.

Пример 1. Обязательное выполнение действия.

Свойство сценария: при любых исходах будет выполнено действие, которому соответствует нетерминал A.

Анализ свойства: преобразуем грамматику, задающую сценарий, в эквивалентную контекстно-свободную. Удалим все недостижимые символы. Если множество $[*A*]$ усиленно форсируется стартовым нетерминалом S, то действие, соответствующее нетерминалу A обязательно будет выполнено.

Пример 2. Обязательное выполнение действия при завершении тестирования.

Свойство сценария: при любых исходах если тестирование завершится, то будет выполнено действие, которому соответствует нетерминал A.

Анализ свойства: преобразуем грамматику, задающую сценарий, в эквивалентную контекстно-свободную. Удалим все недостижимые символы. Если множество $[*A*]$ форсируется стартовым нетерминалом S, то действие, соответствующее нетерминалу A обязательно будет выполнено при завершении тестирования. Отметим отличие от предыдущего примера: может случиться, что $[*A*]$ усиленно не форсируется S, и тогда если тестирование зациклится, то действие, соответствующее A, может и не выполниться.

Возможен анализ и других свойств: завершение тестирования, выполнение действия при завершении тестирования, выполнение действия после другого действия, выполнение действия при условии и др.

Таким образом, разработанные алгоритмы позволяют осуществить автоматизированный анализ свойств сценариев, заданных левоконтекстными грамматиками.

Литература

1. *Ахо А., Сети Р., Ульман Дж.* Компиляторы. Принципы, технологии, инструменты. М.: Вильямс, 2003. 768 с.
2. *Ахо А., Ульман Дж.* Теория синтаксического анализа, перевода и компиляции. Т. 1. Синтаксический анализ, М.: Мир, 1978. 612 с.
3. *Гладкий А.В.* Формальные грамматики и языки. М.: Наука, 1973. 368 с.
4. *Касьянов В.Н.* Лекции по теории формальных языков, автоматов и сложности вычислений. Новосибирск: Изд. НГУ, 1995. 112 с.
5. *Кларк Э.М., Грамберг О., Пелед Д.* Верификация моделей программ. Model Checking М.: Изд. МЦНМО, 2002. 416 с.
6. *Кормен Т.Х., Лейзерсон Ч.И., Ривест Р.Л., Штайн К.* Алгоритмы: Построение и анализ. М.: Вильямс, 2005. 1296 с.
7. *Корнеев Г.А., Маврин П.Ю., Станкевич А.С.* Использование конечных автоматов с магазинной памятью для автоматизации тестирования программных решений // Тр. XI Всерос. науч.-методич. конф. «Телематика-2005» СПб.: СПбГУ ИТМО. С. 510–511
8. *Котляров В.П., Коликова Т.В.* Основы тестирования программного обеспечения. М.: Бином, 2006. 288 с.
9. *Маврин П.Ю., Парфенов В.Г., Станкевич А.С.* Использование левоконтекстных грамматик для описания сценариев автоматического тестирования программных решений // Тр. XVII Всерос. науч.-методич. конф. «Телематика-2010» СПб.: СПбГУ ИТМО. С.197–198
10. *Хомский Н., Миллер Дж.* Введение в формальный анализ естественных языков // Кибернетический сборник. М.: Мир, 1965.
11. *Хопкрофт Дж., Мотвани Р., Ульман Дж.* Введение в теорию автоматов, языков и вычислений. М.: Вильямс, 2002. 528 с.

Станкевич Андрей Сергеевич — доцент кафедры «Компьютерные технологии» Санкт-Петербургского государственного университета информационных технологий, механики и оптики. Область научных интересов: информационные технологии в образовании, теория формальных языков, верификация программ, алгоритмы. Число научных публикаций — 13. stankev@gmail.com, neerc.ifmo.ru/~sta; каф. КТ, СПбГУ ИТМО, Кронверкский пр., 49, Санкт-Петербург, 197101, РФ; р.т. +7(812)242-4620, факс +7(812)242-4620. Научный руководитель — В.Г. Парфенов.

Stankevich Andrey Sergeevich — associate processor, computer technologies department, St Petersburg State University of Information Technology, Mechanics and Optics. Research interests: information technologies in education, formal languages, program verification, algorithms. The number of publications — 13. stankev@gmail.com, neerc.ifmo.ru/~sta; SPbSU ITMO, Kronverkskiy pr. 49, St Petersburg, 197101, Russia; office phone. +7(812)242-4620, fax +7(812)242-4620. Scientific advisor— V.G. Parfenov.

РЕФЕРАТ

Станкевич А.С. **Использование алгоритмов анализа левоконтекстных терминальных грамматик в задачах автоматического тестирования программ**

В статье рассматриваются алгоритмы анализа левоконтекстных терминальных грамматик и эквивалентных им контекстно-свободных грамматик с целью анализа сценариев тестирования, формально заданных с их использованием.

В работе вводится понятие левоконтекстной терминальной грамматики как контекстно-зависимой грамматики, в которой у всех правил правый контекст пуст, а левый состоит только из терминалов. Доказывается, что левоконтекстные грамматики эквивалентны по порождающей мощности контекстно-свободным грамматикам, причем соответствующая контекстно-свободная грамматика строится по заданной грамматике с помощью простой синтаксической процедуры.

Затем вводится понятие контекстно-однозначной левоконтекстной терминальной грамматики, доказывается, что соответствующая ей контекстно свободная грамматика является однозначной. Для этого строится взаимно однозначное соответствие между левосторонними выводами в исходной левоконтекстной терминальной и соответствующей контекстно-свободной грамматике.

При анализе выводов, соответствующих сценариям тестирования, возникает необходимость изучения не только порождений конечных слов, но и бесконечных выводов, что соответствует зацикливанию сценариям тестирования. Поэтому необходимо рассматривать бесконечные цепочки выводов и соответствующие им бесконечные деревья разбора.

Для анализа левоконтекстных терминальных грамматик разработаны алгоритмы для решения следующих задач: форсирование одним нетерминалом другого, форсирование множеством нетерминалов множества нетерминалов, усиленной форсирование одним нетерминалом (множеством нетерминалов) другого, построение множества потенциально тупиковых нетерминалов, связь путей в модифицированном графе порождений с частичными выводами в грамматике. Все алгоритмы разработаны с учетом необходимости анализа бесконечных выводов.

В заключении приведены примеры анализа свойств сценариев с использованием разработанных алгоритмов.

SUMMARY

Stankevich A.S. **Algorithms of analyzing left context terminal grammars used for automated programs testing.**

The article considers algorithms of analyzing left context terminal grammars and equivalent context free grammars for the purpose of analyzing formally specified testing scripts.

The notion of left context terminal grammar is introduced as a contest dependent grammar with empty right context and left context composed exclusively of terminal characters. The theorem about generating power equivalence of left context terminal grammars and context free grammars is proved. The process of creating equivalent context free grammar from left context terminal grammar is purely syntactic and can be easily implemented in software.

After that the notion of context unambiguous grammar is introduced. It is proved that the corresponding context free grammar is also unambiguous. To prove this fact the one-to-one correspondence between left derivations in left context terminal grammar and left derivations in corresponding context free grammars.

When analyzing derivations corresponding to execution of testing scripts, it is not enough to consider derivations of finite words because execution of the script can get into infinite loop. Therefore infinite derivations and corresponding infinite parse trees are considered.

Algorithms for testing the following properties were developed for analyzing left context terminal grammars: forcing of one nonterminal by another, forcing of one set of nonterminals by another, strict forcing of one nonterminal (set of nonterminals) by another, building a set of potentially deadlock nonterminals, correspondence of paths in modified generation graph to partial left derivations in grammar. All algorithms consider possibility of infinite derivations.

In conclusion some examples of analyzing testing scripts with developed algorithms are presented.