ISSN 2079-3316 **ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ** **т. 9**, № 2(37), с. 3–21

УДК 510.52

А. Н. Непейвода

Заметка об автоматическом решении квадратичных уравнений в словах

Аннотация. При анализе программ, оперирующих строками, естественным образом возникает задача решения уравнений в словах. На практике часто встречаются такие уравнения, содержащие, самое большее, два вхождения каждой переменной, — так называемые квадратичные уравнения. Для их решения Ю. И. Хмелевским в 1971 году предложен интуитивно ясный алгоритм, имеющий экспоненциальную сложность. В 1999 году В. Дьекертом показано, что задача решения квадратичного уравнения является NP-трудной. В данной заметке изложены и показаны на примерах способы упрощения классического алгоритма Хмелевского, позволяющие добиться лучшей его применимости в автоматическом анализе программ.

Ключевые слова и фразы: суперкомпиляция, уравнения в словах, анализ программ.

Введение

В последнее время возрос интерес исследователей к автоматическому анализу программ, оперирующих строковыми данными [1–5]. Простейший практический пример таких программ: скрипты анализа и изменения содержимого веб-страницы. Сложность анализа таких программ состоит в том, что язык, содержащий встроенный ассоциативный конструктор приписывания и оператор `replace`, алгоритмически полон. Поэтому в подобных исследованиях рассматриваются частные случаи языков над строковым типом данных, для которых задачу автоматической проверки свойств программ удается сделать разрешимой. В частности, таковы языки над строками не больше определенной длины и языки, содержащие регулярные операции.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта №17-07-00285_а и госзадания ФАНО России № АААА-А16-116021760039-0.

10.25209/2079-3316-2018-9-2-3-21

Существует еще один естественный класс языков, не являющийся Тьюринг-полным: языки уравнений в словах. Эти языки могут описывать связи между значениями строковых параметров более точно, чем регулярные языки.

Пример 1. Рассмотрим фрагмент программы:

Some Language

```
1  string x,y;
2  ...
3  if (x=y)
4  then
5          ...
6  else
7          ...
```

.

Пусть $\mathtt{x} = u\,v$, $\mathtt{y} = v\,u$, u и v — произвольные параметры типа строка. Тогда для разрешения вопроса, по какой ветке будет исполнена программа, необходимо найти множество таких пар $\langle u, v\rangle$, что $v\,u = u\,v$.

Равенство $v\,u = u\,v$ и есть уравнение в словах. Его решения (корни) — все такие строки — значения параметров u и v, при подстановке которых в уравнение оно превращается в буквальное равенство.

Уравнения в словах описывают класс языков, не поддающийся классификации в рамках иерархии Хомского [6]. Поэтому попытки описать множество решений уравнения, например, автоматными языками заведомо потребуют перехода к приближениям, причем даже в простейших случаях [4]. Нами предлагается альтернативный подход: рассматривать сами уравнения в словах как язык описания свойств значений параметров (подобно регулярным или контекстно-свободным языкам). В рамках этого подхода необходимо научиться выбирать наиболее простые выразительные средства внутри данного языка, то есть уметь упрощать уравнения.

В данной заметке будет рассмотрен один частный класс уравнений в словах: так называемые квадратичные уравнения, каждая переменная в которых имеет не более двух вхождений (в частности, таково и уравнение $v\,u = u\,v$). Хотя проблема существования корней квадратичного уравнения является NP-трудной [7], существует простой алгоритм решения таких уравнений, описанный впервые Ю. И. Хмелевским [8]. Поскольку квадратичные уравнения чаще прочих появляются при анализе программ над строковыми данными, имеет смысл рассмотреть

некоторые способы упрощения их решения методом Хмелевского. Нами будут предложены общие методы преобразования и упрощения уравнений, которые помогают ответить на вопрос о существовании их корней, а также найти более краткое описание множества этих корней. Все описанные ниже алгоритмы применены в модельном суперкомпиляторе `MSCP-A` для языка Рефал.

1. Квадратичные уравнения в словах

Пусть даны алфавит констант $\mathfrak{A}$ и алфавит строковых переменных[1] $\mathfrak{V}$.

Пусть Ψ и Φ (возможно, с индексами) — произвольные строки из алфавита $\mathfrak{A} \cup \mathfrak{V}$; ε — пустая строка; u, v (возможно, с индексами) — переменные; $\mathbf{A}, \mathbf{B}, \ldots$ — буквы. Запись $\mathtt{subst}(\Theta, \sigma)$ означает применение к слову Θ подстановки σ.

Определение 1. *Уравнение в словах* — равенство вида $\Psi = \Phi$, где $\Psi, \Phi \in \{\mathfrak{A} \cup \mathfrak{V}\}^*$.

Решить уравнение в словах — найти все такие подстановки $\sigma : \mathfrak{V} \to \{\mathfrak{A}\}^*$, что строки $\mathtt{subst}(\Psi, \sigma)$ и $\mathtt{subst}(\Phi, \sigma)$ совпадают побуквенно.

Уравнение $\Psi = \Phi$ *квадратичное*, если ни одна переменная из $\mathfrak{V}$ не входит в $\Psi = \Phi$ более, чем дважды.

Пример 2. Уравнение $\mathbf{A}\, u = u\, \mathbf{A}$ является квадратичным. Множество его решений (корней): $u \in \{\varepsilon, \mathbf{A}, \mathbf{A}\,\mathbf{A}, \mathbf{A}\,\mathbf{A}\,\mathbf{A}, \ldots, \underbrace{\mathbf{A} \ldots \mathbf{A}}_{k}, \ldots\}$. Слово $\underbrace{\mathbf{A}\,\mathbf{A} \ldots \mathbf{A}}_{n}$ также называется *n-ой степенью слова* $\mathbf{A}$. Если считать, что нулевая степень любого слова — пустое слово, то можно сказать, что множество решений уравнения $\mathbf{A}\, u = u\, \mathbf{A}$ есть множество степеней $\mathbf{A}$.

Уравнение $u_1\, u_1 = u_2\, u_1\, u_2$ не является квадратичным (переменная u_1 входит в него трижды).

Как установил Г. С. Маканин [9], проблема существования корней уравнений в словах разрешима. Множество решений уравнения в общем виде представляется графом сложной структуры, который мы в данной заметке будем называть *графом развертки* уравнения.

---

[1] В модельном суперкомпиляторе `MSCP-A` также используются символьные переменные, пробегающие все множество $\mathfrak{A}$. Все алгоритмы, описанные в данной заметке, могут быть применены к уравнениям, содержащим символьные переменные произвольной кратности.

Существует простая лемма[2] (см., например, [11]), с помощью которой можно решить вопрос о существовании корней квадратичного уравнения.

Предложение 1 (Лемма Леви). Рассмотрим уравнение

$$u\,\Psi = v\,\Phi.$$

Тогда либо[3] $u = v$, либо $u = v\,u_1$, либо $v = u\,v_1$.

В случае квадратичных уравнений леммы Леви достаточно, чтобы построить граф развертки уравнения.

Алгоритм 1 (Алгоритм Хмелевского решения квадратичного уравнения в словах). Пусть дано уравнение $\Psi = \Phi$. Поместим его в корневую вершину дерева и назначим корневую вершину дерева текущей вершиной. Далее $\Theta[i]$ — i-я буква слова Θ, $\mathtt{length}(\Theta)$ — длина Θ.

Рассмотрим первые буквы слов Ψ и Φ в алфавите $\mathfrak{A} \cup \mathfrak{V}$, $\Psi[1]$ и $\Phi[1]$ соответственно. Суффиксы Ψ и Φ длиной $\mathtt{length}(\Psi) - 1$ и $\mathtt{length}(\Phi) - 1$ обозначим как Ψ' и Φ'.

(1) Рассматриваем все вершины на пути от текущей к корню. Если хотя бы в одной из них записано уравнение $\Psi = \Phi$, строим обратное ребро от текущей вершины к данной (на рисунках — пунктиром), помечаем текущую вершину как вычисленную и пропускаем все шаги алгоритма, кроме двух последних. В противном случае переходим к *шагу развертки* уравнения (шагам 2-8).

(2) Если $\Psi = \varepsilon$, $\Phi = \varepsilon$, помещаем в вершину значение $\mathbf{T}$ и помечаем ее как вычисленную. Пути от данной вершины к корню соответствуют решениям исходного уравнения.

(3) Если $\Psi = \varepsilon$, $\Phi[1]$ — буква, уравнение не имеет решений. Удаляем текущую вершину. Аналогично если $\Phi = \varepsilon$, $\Psi[1]$ — буква.

---

[2] Данное наблюдение используется далеко не только при решении уравнений в словах. В частности, такое же правило описывается С. А. Романенко при построении правил прогонки для Рефала-4 [10].

[3] Эти три уравнения имеют непустое пересечение. Первое из них формально избыточно, но включено в формулировку леммы потому, что без него алгоритм решения квадратичного уравнения по Хмелевскому становится некорректен.

(4) Если $\Psi = \varepsilon$, $\Phi[1]$ — переменная, то переменная $\Phi[1]$ обязана принимать значение ε. Строим дочернюю вершину, содержащую уравнение $\varepsilon = \mathtt{subst}(\Phi', \Phi[1] \to \varepsilon)$. Случай $\Phi = \varepsilon$, $\Psi[1]$ — переменная обрабатывается аналогично.

(5) Если $\Psi[1]$ и $\Phi[1]$ неравные буквы, уравнение не имеет решений. Удаляем текущую вершину.

(6) Если $\Psi[1] = \Phi[1]$, сокращаем их и заменяем уравнение в узле на уравнение $\Psi' = \Phi'$.

(7) Если $\Psi[1]$ — буква $\mathbf{A}$, $\Phi[1]$ — переменная u, то либо переменная u пуста, либо она начинается на $\mathbf{A}$. Помечаем текущую вершину как вычисленную и строим к ней два дочерних узла:

- $\mathbf{A}\,\mathtt{subst}(\Psi', u \to \varepsilon) = \mathtt{subst}(\Phi', u \to \varepsilon)$;
- $\mathtt{subst}(\Psi', u \to \mathbf{A}\,u) = u\,\mathtt{subst}(\Phi', u \to \mathbf{A}\,u)$.

Случай, когда $\Psi[1]$ — переменная, $\Phi[1]$ — буква, обрабатывается аналогично.

(8) Если $\Psi[1]$ и $\Phi[1]$ — разноименные переменные u и v, согласно лемме Леви существует три возможности: либо u и v принимают одинаковые значения, либо значение одной из них является префиксом значения другой. Текущая вершина помечается как вычисленная и строятся три дочерних узла:

- $\mathtt{subst}(\Psi', u \to v) = \mathtt{subst}(\Phi', u \to v)$;
- $\mathtt{subst}(\Psi', v \to u\,v) = v\,\mathtt{subst}(\Phi', v \to u\,v)$.
- $u\,\mathtt{subst}(\Psi', u \to v\,u) = \mathtt{subst}(\Phi', u \to v\,u)$.

(9) Удаляем из графа все вычисленные листы, не содержащие значения $\mathbf{T}$ и не имеющие обратных ребер. Эти листы содержат уравнения, шаг развертки которых породил только противоречивые уравнения, следовательно, и сами эти уравнения противоречивы.

(10) Выбираем в качестве текущей вершины любой невычисленный лист. Если таковых не оказалось, граф развертки уравнения построен.

Если в итоговом графе развертки оказались листы, содержащие $\mathbf{T}$, у уравнения существуют решения. В противном случае решений нет.

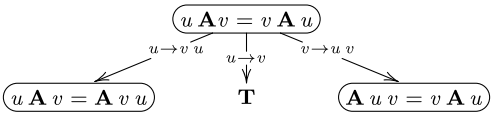

Рис. 1. Начало развертки уравнения $u\,\mathbf{A}v = v\,\mathbf{A}\,u$

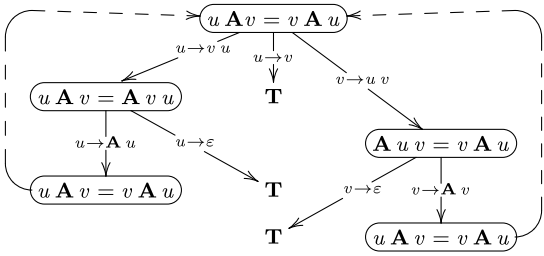

Рис. 2. Граф развертки уравнения $u\,\mathbf{A}v = v\,\mathbf{A}\,u$

Пример 3. Покажем, как работает Алгоритм 1, на примере уравнения $u\,\mathbf{A}v = v\,\mathbf{A}\,u$.

Первые термы левой и правой части уравнения — переменные, причем неравные. Поэтому применяем шаг 8 и строим тройное расщепление (см. рис. 1).

Уравнение $v\,\mathbf{A}\,v = v\,\mathbf{A}\,v$, полученное подстановкой $u \to v$, тривиально и порождает лист $\mathbf{T}$. К остальным двум уравнениям применяется шаг 7.

Оба уравнения, полученные подстановками $u \to \varepsilon$, $v \to \varepsilon$, тривиальны. Уравнения, полученные подстановками $u \to \mathbf{A}\,u$, $v \to \mathbf{A}\,v$, буквально повторяют исходное, поэтому можно провести из них обратные ребра к корню. Граф развертки уравнения показан на рисунке 2.

Данный граф содержит листы, содержащие значение $\mathbf{T}$, поэтому исходное уравнение имеет решения. Однако структура этих решений нерегулярна ($u = (w\,\mathbf{A})^n\,w$, $v = (w\,\mathbf{A})^m\,w$, w — строковый параметр), поэтому кратким описанием их множества может выступать само исходное уравнение.

Конечность шагов работы алгоритма Хмелевского вытекает из следующего утверждения.

Предложение 2. Пусть дано квадратичное уравнение $\Psi = \Phi$, $\mathtt{length}(\Psi) + \mathtt{length}(\Phi) = N$. После шагов 2-8 Алгоритма 1 все дочерние

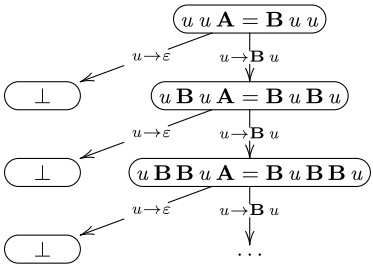

Рис. 3. Бесконечный граф развертки неквадратичного уравнения

уравнения остаются квадратичными, причем длина каждого из них не превышает N.

Поскольку ни на одном шаге работы Алгоритма 1 не порождается термов, отличных от тех, что присутствуют в исходном уравнении, Предложение 2 гарантирует, что в любой бесконечной цепочке уравнений, порожденных друг из друга алгоритмом, найдутся два одинаковых уравнения. Поэтому Алгоритм 1 завершает работу за конечное время. Если бы ограничений на кратность переменных в исходном уравнении не было, развертка могла бы продолжаться бесконечно, порождая все более и более длинные уравнения.

Пример 4. Пусть дано уравнение $u\,u\,\mathbf{A} = \mathbf{B}\,u\,u$. Начнем осуществлять его развертку по Алгоритму 1 (см. рис. 3). В данном случае на схеме ветви развертки, приведшие к противоречию, не удаляются, а помечаются знаком $\bot$.

По правой ветви развертки на каждом шаге получается все более длинное уравнение. Алгоритм не завершает работу.

2. Методы упрощения уравнений в применении к квадратичным

В данном разделе мы оценим способы упрощения уравнений общего вида, применяемые в `MSCP-A`, с точки зрения упрощения графов развертки для квадратичных уравнений. Пытаться упрощать уравнение имеет смысл на каждом шаге развертки графа: это может существенно сократить его размер. Поскольку задача решения квадратичного уравнения относится к переборным (NP-трудна), то поиск простых и эффективных эвристик для ее решения практически важен.

В работах [11, 12] предлагаются следующие общие способы упрощения уравнений. Они же применяются в `MSCP-A`.

(1) Расщеплять по префиксам или суффиксам. Если у левой и правой части уравнения выделяются префиксы (суффиксы) заведомо одинаковой длины, то такие префиксы (суффиксы) равны между собой, и уравнение можно разбить на несколько более простых.

(2) Оценивать длины. Можно написать линейное диофантово уравнение, описывающее взаимосвязь между длинами переменных в уравнении. Исходя из свойств этого уравнения, можно отсечь некоторые ветви графа развертки.

Проанализируем оба метода упрощения в применении к квадратичным уравнениям.

2.1. Расщепление

При расщеплении уравнений важнейшую роль играет свойство *равносоставленности* их частей.

Определение 2. Два слова Φ_1, Φ_2 *равносоставлены*, если они состоят из одного и того же набора переменных (с равным числом вхождений), и количество букв алфавита констант в Φ_1 и Φ_2 совпадает. Сами буквы совпадать не обязаны.

Если слова Φ_1 и Φ_2 равносоставлены, то $\mathtt{length}(\Phi_1) = \mathtt{length}(\Phi_2)$.

Рассмотрим следующие два способа расщепления уравнений.

(1) Можно выделять из левой и правой частей уравнения равносоставленные префиксы или суффиксы. В случае, например, равносоставленных префиксов такое уравнение имеет вид

$$\begin{gathered}\Psi_1\,\Psi_2 = \Phi_1\,\Phi_2\\ \mathtt{length}(\Psi_1) = \mathtt{length}(\Phi_1).\end{gathered}$$

Тогда $\Psi_i = \Phi_i$, $i \in \{1, 2\}$.

Например, если $u_1\,\mathbf{A}\,u_2 = \mathbf{B}\,u_1\,u_3$, то $u_1\,\mathbf{A} = \mathbf{B}\,u_1$.

Такой способ расщепления назовем *расщеплением по равносоставленности*.

(2) Уравнение можно расщепить, если в обоих его частях можно выделить префикс, суффикс и центр такие, что центры двух частей равносоставлены по отношению друг к другу, а префиксы

равносоставлены с суффиксами. Такое уравнение имеет вид

$$\Psi_1\,\Psi_2\,\Psi_3 = \Phi_1\,\Phi_2\,\Phi_3$$
$$\mathtt{length}(\Psi_1) = \mathtt{length}(\Psi_3), \quad \mathtt{length}(\Phi_1) = \mathtt{length}(\Phi_3),$$
$$\mathtt{length}(\Phi_2) = \mathtt{length}(\Psi_2).$$

Значит, $\mathtt{length}(\Psi_1) = \mathtt{length}(\Phi_1)$, и исходное уравнение расщепляется на три: $\Psi_i = \Phi_i$ $(i \in \{1, 2, 3\})$.

Например, если $u_1\,w\,\mathbf{A}\,u_1 = u_2\,\mathbf{A}\,w\,u_2$, то $u_1 = u_2$, $w\,\mathbf{A} = \mathbf{A}\,w$.

Такой способ расщепления назовем *расщеплением по кратности*[4].

Расщепив уравнение на несколько, проще всего решать их по отдельности. Однако разрешимость каждого отдельного уравнения не гарантирует разрешимости их системы.

Пример 5. Уравнение $u\,u = \mathbf{A}\,v\,\mathbf{B}\,v$ разбивается по кратности на два: $u = \mathbf{A}\,v$ и $u = \mathbf{B}\,v$. Каждое из них имеет корни, а исходное уравнение корней не имеет.

Второй очевидный подход — хранить в конфигурации узла графа развертки все уравнения и решать их одно за другим. Тогда подстановки, осуществляемые после шага развертки, должны применяться не только к решаемому уравнению, а ко всем уравнениям в узле. Однако этот подход создает трудности, если граф развертки какого-либо уравнения, за исключением последнего в очереди на решение, содержит обратные ребра.

Пример 6. Пусть решается система $u\,\mathbf{A} = \mathbf{A}\,u$ и $v\,\mathbf{A}\,v = u$, причем развертка начинается с первого уравнения. Результат развертки изображен на рис. 4. Видно, что лист после подстановки $u \to \varepsilon$ ведет к противоречию (хотя исходная система имеет корни). Происходит это потому, что вложение узла после подстановки $u \to \mathbf{A}\,u$ в родительский не может быть осуществлено простой переименовкой: хотя первое уравнение системы $\{u\,\mathbf{A} = \mathbf{A}\,u,\ v\,\mathbf{A}\,v = \mathbf{A}\,u\}$ точно повторяет исходное, второе имеет дополнительное вхождение буквы $\mathbf{A}$.

[4]Расщепление квадратичных уравнений по кратности может происходить только указанным выше способом. Но в общем случае число равносоставленных подслов в одной части уравнения может быть и больше двух. Например, уравнение $u\,v\,v\,u\,v\,u = \mathbf{A}\,w\,w\,w\,w\,\mathbf{B}\,w\,\mathbf{C}\,w$ расщепляется на три: $u\,v = \mathbf{A}\,w\,w$, $v\,u = w\,w\,\mathbf{B}$, $v\,u = w\,\mathbf{C}\,w$.

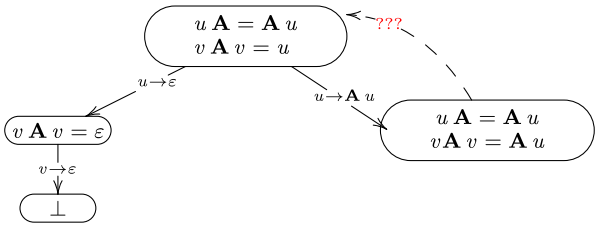

Рис. 4. Нетривиальная совместная развертка уравнений

Заметим, что в Примере 6 переменная u входит в систему уравнений трижды. Именно это создает затруднения при совместной развертке уравнений в системе. Удобное свойство квадратичных уравнений — в том, что и расщепление по равносоставленности, и расщепление по кратности позволяют корректно решать полученные системы уравнений внутри одного графа развертки. Покажем это.

Предложение 3.

(1) Пусть квадратичное уравнение $\Psi_1\,\Psi_2 = \Phi_1\,\Phi_2$ было расщеплено по равносоставленности префиксов Ψ_1 и Φ_1. Тогда множество переменных, входящих в $\Psi_1 = \Phi_1$, не пересекается с множеством переменных, входящих в $\Psi_2 = \Phi_2$.

(2) Пусть квадратичное уравнение $\Psi_1\,\Psi_2\,\Psi_3 = \Phi_1\,\Phi_2\,\Phi_3$ было расщеплено по кратности. Тогда граф развертки уравнения $\Psi_1 = \Phi_1$ (и $\Psi_3 = \Phi_3$) не содержит обратных ребер.

Доказательство.

(1) Поскольку слова Ψ_1 и Φ_1 равносоставлены, то каждая переменная входит либо в оба этих слова, либо ни в одно. Если она входит в оба слова Ψ_1 и Φ_1, то она имеет уже два вхождения в исходное уравнение, а значит, не может входить в Ψ_2 или Φ_2.

(2) Поскольку слова Ψ_1 и Ψ_3 равносоставлены, каждая переменная, входящая в них, входит в них по разу и не имеет других вхождений в исходное уравнение. Аналогичное замечание справедливо в отношении Φ_1 и Φ_3. Поэтому в уравнении $\Psi_1 = \Phi_1$ в каждой части не будет повторных переменных, и не будет переменных, входящих сразу в правую и левую часть. Значит, уравнение $\Psi_1 = \Phi_1$ линейно (все переменные в нем имеют кратность 1). Шаг

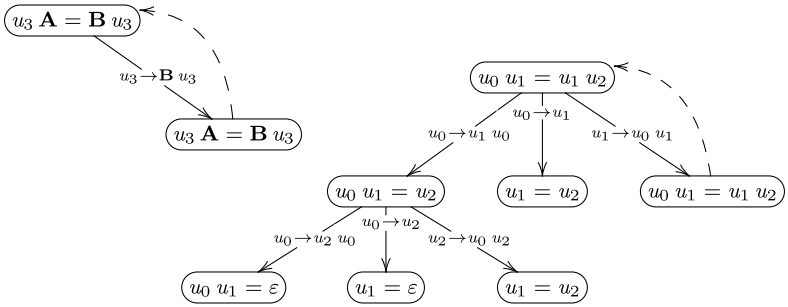

Рис. 5. Графы развертки для уравнений $\mathbf{A}\,u_3 = u_3\,\mathbf{B}$ и $u_0\,u_1 = u_1\,u_2$

развертки линейного уравнения всегда укорачивает либо левую, либо правую часть, поэтому граф развертки такого уравнения не может иметь обратных ребер.

□

Предложение 3 показывает, как можно работать с расщепленными квадратичными уравнениями. Уравнения, полученные расщеплением по равносоставленности, можно решать по очереди независимо друг от друга. Уравнения, полученные расщеплением по кратности, решаются совместно (в системе)[5]. И в том, и в другом случае получается существенный выигрыш по размеру графов развертки уравнений по сравнению с графом развертки нерасщепленного уравнения.

Пример 7. Уравнение $u_0\,u_1\,\mathbf{A}\,u_3 = u_1\,u_2\,u_3\,\mathbf{B}$ разбивается на два по равносоставленности: $u_0\,u_1 = u_1\,u_2$ и $\mathbf{A}\,u_3 = u_3\,\mathbf{B}$. Графы развертки этих уравнений приведены на рис. 5. Для уравнения $u_0\,u_1 = u_1\,u_2$ граф неполный, однако по нему видно, что существуют минимум четыре[6] вершины в дереве, содержащие значение $\mathbf{T}$. Поскольку уравнение $\mathbf{A}\,u_3 = u_3\,\mathbf{B}$ не порождает вершин, содержащих $\mathbf{T}$, то корней у него нет.

Если бы мы решали вместо двух разделенных уравнений одно слитное, то каждый лист графа развертки, соответствующий листу графа развертки уравнения $u_0\,u_1 = u_1\,u_2$, содержащему значение $\mathbf{T}$, либо такому, уравнение в котором содержит хотя бы с одной

[5]При разбиении по кратности может получиться три уравнения, но третье (уравнение на центральные части) состоит всегда из равносоставленных частей и может решаться по отдельности от двух прочих.

[6]Проведя полную развертку по Алгоритму 1, можно убедиться, что таких вершин восемь. Оба уравнения $u_1 = u_2$ порождают по три такие вершины.

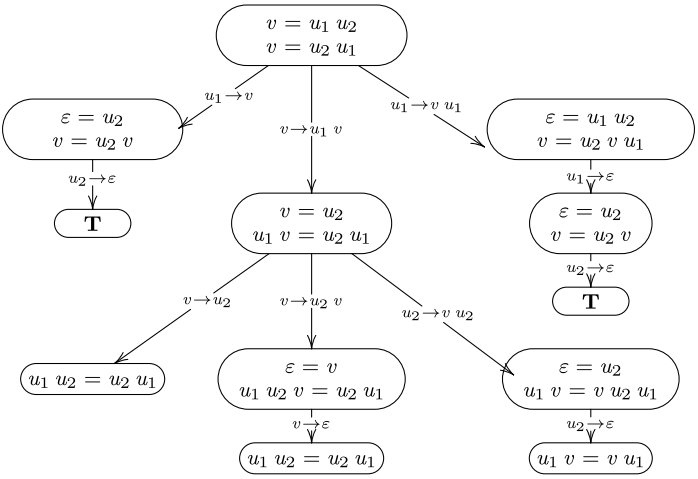

Рис. 6. Граф развертки для совместного решения уравнений $v = u_1\ u_2$ и $v = u_2\ u_1$

из сторон ε, продолжался бы разверткой уравнения $\mathbf{A}\,u_3 = u_3\,\mathbf{B}$ (возможно, с некоторым префиксом). Граф развертки уравнения (свидетельствующий об отсутствии решений) получился бы значительно больше.

Пример 8. Уравнение $u_1\ u_2\ \mathbf{A}\ u_2\ u_1 = v\ \mathbf{A}\ v$ удовлетворяет условию разбиения по кратности: $\mathtt{length}(u_1\ u_2) = \mathtt{length}(u_2\ u_1)$, длины центральных частей (буква $\mathbf{A}$) очевидно равны, так же, как и префикс и суффикс правой части. Исходя из этого, $u_1\ u_2 = v$ и $u_2\ u_1 = v$ (и $\mathbf{A} = \mathbf{A}$, что тривиально). Граф развертки системы этих уравнений приведен на рис. 6. Он неполон: вершины, в которых остается только одно уравнение, дальше не раскрываются. По графу видно, что, если произвести все указанные на ребрах подстановки, он описывает два простых семейства решений уравнения (первое из которых является подмножеством второго):

$$\{\langle u_1, u_2, v\rangle | v = u_1\ \&\ u_2 = \varepsilon\},$$

$$\{\langle u_1, u_2, v\rangle | v = u_1\ u_2\ \&\ u_1\ u_2 = u_2\ u_1\}.$$

Поскольку граф развертки системы уравнений не имеет обратных ребер, семейство его решений описывается как дизъюнкция семейств решений уравнений, находящихся в его узлах. Поэтому исходная

система уравнений может быть автоматически сведена к гораздо более простой системе, задающей множества, указанные выше.

Граф развертки нерасщепленного уравнения имеет значительно больший размер и запутанную структуру с множеством обратных ребер, из которой подобным наблюдением извлечь простое описание множества его решений невозможно.

2.2. Оценка длин значений переменных

Как в Примере 7, так и в Примере 8 граф развертки уравнения дополнительно разрастается из-за существования общих корней по различным его ветвям. В частности, на рис. 5 первая и вторая ветви, исходящие из узла $u_0\, u_1 = u_2$, описывают одно и то же решение, как и на рис. 6 все три ветви, исходящие из узла $v = u_2,\ u_1\, v = u_2\, u_1$. Причина такой избыточности кроется в формулировке леммы Леви (Предложение 1), используемой в алгоритме Хмелевского: подстановки $u \to v$, $u \to v\, u$, $v \to u\, v$ имеют непустое пересечение. Можно сделать его пустым, введя понятие буквенной переменной s и с помощью него дополнительно уточнив подстановки до $u \to v\, s\, u$, $v \to u\, s\, v$. Но в этом случае графы развертки многих уравнений становятся более громоздкими[7] (в чем можно убедиться, например, для уравнения $u\, v = v\, u$).

От ряда повторных разверток можно избавиться, если заметить, что некоторые ветви развертки можно сразу обрезать исходя из оценки длин значений переменных. Например, если $u_0\, u_1 = u_2$, ветвь $u_0 \to u_2\, u_0$ порождает то же множество решений, что и ветвь $u_0 \to u_2$ (а значит, может быть обрезана), потому что длина значения переменной u_0 заведомо не больше длины значения u_2. Анализ длин левой и правой части показывает, что уравнение $u_0\, u_1\, \ldots\, u_n\, \mathbf{A} = u_1\, \ldots\, u_n\, u_0$ не имеет корней (его левая часть имеет заведомо большую длину из-за вхождения константы $\mathbf{A}$). Поэтому, чтобы сократить развертку уравнения $\Psi = \Phi$, имеет смысл воспользоваться результатами анализа линейного диофантова уравнения $\mathtt{length}(\Psi) = \mathtt{length}(\Phi)$.

[7] К тому же с таким уточнением уравнение после шага развертки может стать длиннее, чем до него. Правда, удлинение может произойти лишь фиксированное число раз и лишь за счет введения новых буквенных переменных, поэтому граф развертки все равно будет конечен. Доказательство этого факта здесь приводить не будем, но заметим, что он выводится из наблюдения, что каждую строковую переменную может предварять, самое большее, одна буквенная.

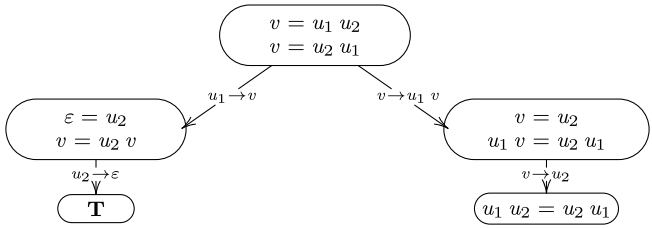

Рис. 7. Сокращенный граф развертки для совместного решения уравнений $v = u_1\ u_2$ и $v = u_2\ u_1$

С точностью до перестановки левой и правой частей, такое уравнение имеет вид

$$a + k_1 * \mathtt{length}(u_1) + \cdots + k_n * \mathtt{length}(u_n) = l_1 * \mathtt{length}(v_1) + \cdots + l_m * \mathtt{length}(v_m),$$

причем $k_i, l_j \in \{1, 2\}$ и $u_i \neq v_j$ (повторные вхождения одной и той же переменной в левую и правую части сокращаются).

Если в таком уравнении $m = 1$ и для некоторого i $k_i \leq l_1$, то можно сделать вывод, что $\mathtt{length}(u_i) \leq \mathtt{length}(v_1)$. Если к тому же $a \neq 0$, то неравенство строгое. Это позволяет обрезать недостижимые и повторные ветви развертки, если уравнение $\Psi = \Phi$ имеет вид $u_i\ \Psi' = v_1\ \Phi'$.

Пример 9. После оценки длин значений переменных граф развертки системы уравнений $v = u_1\ u_2$ и $v = u_2\ u_1$ стал значительно меньше (рис. 7).

Наконец, уравнение $\mathtt{length}(\Psi) = \mathtt{length}(\Phi)$ может порождать разбиение уравнения $\Psi = \Phi$, являющееся в некотором роде обобщением описанных ранее разбиений. А именно, если $\Psi = \Psi_1\ \Psi_2$, $\Phi = \Phi_1\ \Phi_2$ и исходя из анализа уравнения $\mathtt{length}(\Psi) = \mathtt{length}(\Phi)$ может быть доказано уравнение $\mathtt{length}(\Psi_1) = \mathtt{length}(\Phi_1)$, то $\Psi_1 = \Phi_1$, $\Psi_2 = \Phi_2$.

Пример 10. Рассмотрим уравнение $u_1\ v\ u_2\ w\ u_2\ u_1 = v\ u_3\ u_3\ w$. По кратности или равносоставленности оно не разбивается, но после сокращения подобных членов диофантова уравнения на длины имеем:

$$2 * \mathtt{length}(u_1) + 2 * \mathtt{length}(u_2) = 2 * \mathtt{length}(u_3).$$

Таким образом, $\mathtt{length}(u_1) + \mathtt{length}(u_2) = \mathtt{length}(u_3)$, а следовательно, $u_1\ v\ u_2 = v\ u_3$ и $w\ u_2\ u_1 = u_3\ w$. Начало развертки системы этих уравнений приведено на рис. 8.

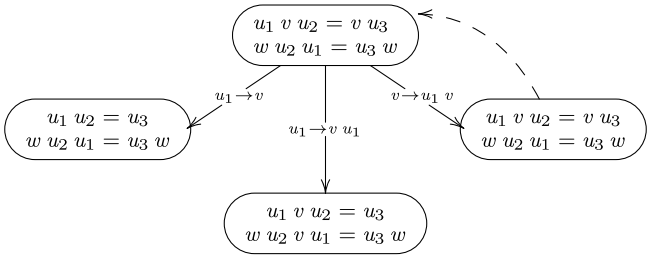

Рис. 8. Начало развертки уравнений $u_1\ v\ u_2 = v\ u_3$ и $w\ u_2\ u_1 = u_3\ w$

Две ветви развертки приводят к уже рассмотренному случаю, когда система состоит из двух уравнений, первое из которых — линейное. Но третья ветвь развертки приводит к повторению исходной системы уравнений, что порождает обратное ребро. Поэтому исходная система уравнений не сводится к дизъюнкции систем, полученных на ациклических путях графа.

Пример 10 показывает, что возможности анализа уравнений после расщепления по длинам значений переменных несколько ограничены. Полной разверткой такой системы можно установить, имеются ли корни у исходного уравнения, но не всегда возможно свести исходное уравнение к системе более простых. Типичен тот факт, что путь между вершинами, связанными обратным ребром, не содержит подстановок, изменяющих второе уравнение. Это происходит потому, что каждая переменная, входящая во второе уравнение, входит в первое, самое большее, один раз. А значит, любая подстановка такой переменной будет укорачивать первое уравнение и запрещать появление обратных ребер между вершинами, путь между которыми содержит такую подстановку. Поэтому, если в некоторых двух узлах графа развертки системы повторится первое уравнение, то в них повторится и второе.

3. Заключение

Мы привели и обосновали несколько элементарных методов упрощения квадратичных уравнений в словах, делающих алгоритм их решения более применимым на практике. При обсуждении даже этих методов видно, что как только область их применения выходит за множество квадратичных уравнений, возникают трудности с их применением (что иллюстрирует, в частности, Пример 6). Если алгоритм решения квадратичных уравнений можно сравнить с элементарным

алгоритмом суперкомпиляции без обобщения [13], то расширение класса уравнений, для которых строится граф развертки, требует более сильных средств анализа. В частности, аналогом обобщения для графа развертки уравнений может выступать алгоритм рекомпрессии (recompression) [14], но данная аналогия требует тщательного анализа.

Сходство методов упрощения уравнений в словах путем анализа графа развертки уравнения и методов преобразования программ, основанных на развертке графа вычислений, позволяет использовать общие алгоритмы анализа для обоих классов графов. Подобные аналогии неоднократно возникали при развитии методов суперкомпиляции. Помимо уже упомянутого использования леммы Леви в прогонке [10], укажем также, что идея анализа систем диофантовых уравнений на длины переменных использовалась при построении быстрого отождествления для языка Рефал [15].

Список литературы

[1] А. П. Немытых. *Суперкомпилятор SCP4: Общая структура*, УРСС, М., 2007. ↑3

[2] N. Bjorner, N. Tillmann, A. Voronkov. “Path feasibility analysis for string-manipulating programs”, *Tools and Algorithms for the Construction and Analysis of Systems*, TACAS 2009, Lecture Notes in Computer Science, vol. **5505**, eds. Kowalewski S., Philippou A., Springer, Berlin, pp. 307–321. ↑3

[3] M. T. Trinh, D. H. Chu, J. Jaffar. “Progressive reasoning over recursively-defined strings”, *Computer Aided Verification*, CAV 2016, Lecture Notes in Computer Science, vol. **9779**, eds. Chaudhuri S., Farzan A., Springer, 2016, pp. 218–240. ↑3

[4] F. Yu, T. Bultan, O. H. Ibarra. “Relational string verification using multi-track automata”, *Implementation and Application of Automata*, CIAA 2010, Lecture Notes in Computer Science, vol. **6482**, eds. Domaratzki M., Salomaa K., Springer, 2010, pp. 290–299. ↑3,4

[5] T. Liang, A. Reynolds, N. Tsiskaridze, C. Tinelli, C. Barrett, M. Deters. “An efficient SMT solver for string constraints”, *Formal Methods in System Design*, **48**:3 (2016), pp. 206–234. ↑3

[6] J. Karhumaki, W. Plandowski, F. Mignosi. “The expressibility of languages and relations by word equations”, *Automata, Languages and Programming*, ICALP 1997, Lecture Notes in Computer Science, vol. **1256**, eds. Degano P., Gorrieri R., Marchetti-Spaccamela A., Springer, 1997, pp. 98–109. ↑4

[7] J. Karhumaki, H. Maurer, G. Paun, G. Rozenberg (eds.). *Jewels are forever. Contributions on theoretical computer science in honor of Arto Salomaa*, Springer, 1999, 379 p. ↑4

[8] Ю. И. Хмелевский. «Уравнения в свободной полугруппе», Тр. МИАН СССР, т. **107**, 1971, с. 3–288. ↑4

[9] Г. С. Маканин. «Проблема разрешимости уравнений в свободной полугруппе», *Матем. сб.*, **103(145)**:2(6) (1977), с. 147–236. ↑5

[10] С. А. Романенко. *Прогонка для программ на Рефале-4*, Препринт № 211, Институт Прикладной Математики им. М. В. Келдыша АН СССР, 1987, 23 с. ↑6,18

[11] J. Karhumaki. *Combinatorics of words.* ↑6,9

[12] M. Huova. *Combinatorics of words. New aspects on avoidability, defect effect, equations and palindromes*, Ph.D. Thesis, Turku Centre for Computer Science, 2014, 140 p. ↑9

[13] M. H. Sørensen. *Turchin's supercompiler revisited*, Ms. Thesis, Department of Computer Science, University of Copenhagen, 1994, 143 p. ↑18

[14] A. Jez. "Recompression: a simple and powerful technique for word equations", *J. ACM*, **63**:1 (March 2016), 4, 51 p. ↑18

[15] С. М. Абрамов, А. Ю. Орлов. «Компиляция в императивные языки синтаксического отождествления языка Рефал», *Труды международной конференции «Программные системы: теория и приложения»*. Т. 1 (ИПС РАН, г. Переславль-Залесский, май 2004), ред. С. М. Абрамов, Физматлит, М., 2004, с. 403–448. ↑18

Рекомендовал к публикации *д.ф.-м.н. С. М. Абрамов*

Пример ссылки на эту публикацию:

А. Н. Непейвода. «Заметка об автоматическом решении квадратичных уравнений в словах». *Программные системы: теория и приложения*, 2018, **9**:2(37), с. 3–21. 10.25209/2079-3316-2018-9-2-3-21
http://psta.psiras.ru//read/psta2018_2_3-21.pdf

Об авторе:

Антонина Николаевна Непейвода

Младший научный сотрудник. Работает в сфере суперкомпиляции, частичных вычислений и вычислительной сложности процессов преобразования программ.

0000-0003-3949-2164
e-mail: a_nevod@mail.ru

UDC 510.52

Antonina Nepeivoda. *On solving quadratic word equations.*

Abstract. Word equations are natural constraints in an automatic analysis of string manipulating programs. In particular, equations with at most two occurrences of each variable (quadratic word equations) are of interest of the analysis. The algorithm solving such equations with the exponential complexity is given by Yu. Hmelevskij in 1971. V. Diekert in 1999 proved that the satisfiability problem for the quadratic word equations is NP-hard. In this paper we suggest some refinements of Hmelevskij's algorithm to make it more applicable in the automatic analysis of programs. We consider the length analysis and splitting procedures and show when these refinements can be used to extract explicit solutions of the equations and when they can be used only for deciding satisfiability. (*In Russian*).

Key words and phrases: supercompilation, word equations.

References

[1] A. P. Nemytykh. *Supercompiler SCP4: general structure*, URSS, M., 2007 (in Russian). ↑3

[2] N. Bjorner, N. Tillmann, A. Voronkov. "Path feasibility analysis for string-manipulating programs", *Tools and Algorithms for the Construction and Analysis of Systems*, TACAS 2009, Lecture Notes in Computer Science, vol. **5505**, eds. Kowalewski S., Philippou A., Springer, Berlin, pp. 307–321. ↑3

[3] M. T. Trinh, D. H. Chu, J. Jaffar. "Progressive reasoning over recursively-defined strings", *Computer Aided Verification*, CAV 2016, Lecture Notes in Computer Science, vol. **9779**, eds. Chaudhuri S., Farzan A., Springer, 2016, pp. 218–240. ↑3

[4] F. Yu, T. Bultan, O. H. Ibarra. "Relational string verification using multi-track automata", *Implementation and Application of Automata*, CIAA 2010, Lecture Notes in Computer Science, vol. **6482**, eds. Domaratzki M., Salomaa K., Springer, 2010, pp. 290–299. ↑3,4

[5] T. Liang, A. Reynolds, N. Tsiskaridze, C. Tinelli, C. Barrett, M. Deters. "An efficient SMT solver for string constraints", *Formal Methods in System Design*, **48**:3 (2016), pp. 206–234. ↑3

[6] J. Karhumaki, W. Plandowski, F. Mignosi. "The expressibility of languages and relations by word equations", *Automata, Languages and Programming*, ICALP 1997, Lecture Notes in Computer Science, vol. **1256**, eds. Degano P., Gorrieri R., Marchetti-Spaccamela A., Springer, 1997, pp. 98–109. ↑4

10.25209/2079-3316-2018-9-2-3-21

[7] J. Karhumaki, H. Maurer, G. Paun, G. Rozenberg (eds.). *Jewels are forever. Contributions on theoretical computer science in honor of Arto Salomaa*, Springer, 1999, 379 p. ↑4

[8] Yu. I. Khmelevskii. "Equations in a free semigroup", Proceedings of the Steklov Institute of Mathematics, vol. **107**, 1971, pp. 1–270 (in Russian). ↑4

[9] G. S. Makanin. "The problem of solvability of equations in a free semigroup", *Mathematics of the USSR-Sbornik*, **32**:2 (1977), pp. 129–198. ↑5

[10] C. A. Romanenko. *The driving algorithm for programs in Refal-4*, Preprint no. 211, Institut Prikladnoy Matematiki im. M. V. Keldysha AN SSSR, 1987 (in Russian), 23 p. ↑6,18

[11] J. Karhumaki. *Combinatorics of words.* ↑6,9

[12] M. Huova. *Combinatorics of words. New aspects on avoidability, defect effect, equations and palindromes*, Ph.D. Thesis, Turku Centre for Computer Science, 2014, 140 p. ↑9

[13] M. H. Sørensen. *Turchin's supercompiler revisited*, Ms. Thesis, Department of Computer Science, University of Copenhagen, 1994, 143 p. ↑18

[14] A. Jez. "Recompression: a simple and powerful technique for word equations", *J. ACM*, **63**:1 (March 2016), 4, 51 p. ↑18

[15] S. M. Abramov, A. Yu. Orlov. "Compilation of Refal pattern matching to imperative languages", *Trudy mezhdunarodnoy konferentsii "Programmnyye sistemy: teoriya i prilozheniya"*. V. 1 (IPS RAN, g. Pereslavl'-Zalesskiy, may 2004), ed. S. M. Abramov, Fizmatlit, M., 2004, pp. 403–448 (in Russian). ↑18

Sample citation of this publication:

Antonina Nepeivoda. "On solving quadratic word equations". *Program Systems: Theory and Applications*, 2018, **9**:2(37), pp. 3–21. (*In Russian*).

10.25209/2079-3316-2018-9-2-3-21

http://psta.psiras.ru//read/psta2018_2_3-21.pdf