Полиномиальный по времени алгоритм проверки логико-термальной эквивалентности программ

В.А.Захаров (ИСП РАН), Т.А. Новикова (ф-т ВМК МГУ им. М.В. Ломоносова)
{zakh@cs.msu.su} {taniaelf@mail.ru}

Аннотация. Логико-термальная эквивалентность программ – это одно из наиболее слабых отношений эквивалентности программ, аппроксимирующих отношение функциональной эквивалентности и обладающих разрешающим алгоритмом. В данной статье предложена новая модификация алгоритма проверки логико-термальной эквивалентности программ, основанная на операции вычисления точной нижней грани в решетке конечных подстановок. Показано, что трудоемкость предложенного алгоритма оценивается величиной $O(n^6)$, где n - размер анализируемых программ. Полученная верхняя оценка сложности задачи проверки логико-термальной эквивалентности программ существенно меньше ранее известных полиномиальных оценок сложности указанной задачи.

Ключевые слова: программа, логико-термальная эквивалентность программ, подстановка, антиунификация, сложность.

1. Введение

Две программы π_1 и π_2 считаются логико-термально эквивалентными, если для любой синтаксически допустимой трассы tr' в одной из программ существует такая трасса tr'' в другой программе, что в обеих трассах логические условия (предикаты) проверяются в одной и той же последовательности для одних и тех же наборов значений переменных. Логико-термальная (л-т) эквивалентность программ была введена в статье [1]. Интерес к л-т эквивалентности программ был обусловлен двумя ее свойствами:

1) л-т эквивалентность программ π_1 и π_2 влечет функциональную эквивалентность этих программ [2],

2) отношение л-т эквивалентности программ разрешимо за полиномиальное время [2,3].

Благодаря этим качествам алгоритмы проверки л-т эквивалентности программ можно использовать для решения многочисленных задач анализа поведения и

преобразования программ, включая задачу верификации [4-8], оптимизации [9,10] и реорганизации (рефакторинга) программ [11-13] .

Было установлено также, что л-т эквивалентность – это одно из наиболее слабых разрешимых отношений эквивалентности программ, аппроксимирующих отношение функциональной эквивалентности. Первый алгоритм, разрешающий л.-т. эквивалентность стандартных схем программ был предложен в статье [1]. Сложность его оценивается двойной экспонентой, зависящей от размера проверяемых программ. Впоследствии были созданы более эффективные алгоритмы [4,6]; в частности, в работах [2,3] (см. также [16]) был разработан алгоритм проверки л.-т. эквивалентности программ за время $O(n^7)$, где n - размер анализируемых программ.

Как было показано в статьях [14,15], л-т эквивалентность программ лежит на самой границе, отделяющей разрешимые виды эквивалентности программ от неразрешимых. Поскольку определение л-т эквивалентности не опирается на интерпретации базовых компонентов программ (функций и предикатов), задача проверки л-т эквивалентности программ может быть отнесена к задачам статического анализа программ, и для ее решения может быть привлечены методы вычисления неподвижных точек монотонных операторов на решетках [16]. На основании этих методов и строятся алгоритмы проверки л-т эквивалентности программ. В настоящей статье в качестве такой решетки была выбрана решетка конечных подстановок, исследованная в статьях [17,18]. В многочисленных работах (см. [17-23]) были предложены различные алгоритмы антиунификации подстановок, вычисляющие точные нижние грани в решетке подстановок. На основании этих алгоритмов в статье [24] был предложен новый алгоритм проверки л-т эквивалентности программ и показано его применение для решения новой задачи анализа программ – задачи унификации программ. В настоящей статье мы показываем, что верхняя оценка сложности предложенного алгоритма составляет величину $O(n^6)$, где n - размер анализируемых программ, что существенно улучшает ранее известные верхние оценки сложности задачи проверки л-т эквивалентности программ.

Содержание настоящей статьи таково. Во втором ее разделе приводятся основные понятия алгебры конечных подстановок с операциями композиции и антиунификации. В третьем разделе статьи коротко определяются модель стандартных схем программ и отношение л-т эквивалентности программ в этой модели, а также описан алгоритм проверки л-т эквивалентности программ в том виде, в котором он был предложен в статье [24]. Поскольку в этом алгоритме интенсивно используются операции композиции и антиунификации подстановок, сложность этих операций и самого алгоритма существенно зависят от способов представления этих подстановок. В четвертом разделе рассмотрены графовые способы представления конечных подстановок, приведены верхние и нижние оценки сложности указанных операций для подстановок, реализованных в виде конечных ациклических

ориентированных графов. Эти оценки свидетельствуют о том, что для повышения эффективности алгоритма проверки л-т эквивалентности программ класс подстановок, используемых в этом алгоритме, должен быть существенно сокращен. Поэтому в пятом разделе статьи вводится специальный подкласс редуцированных подстановок (определение 1), которые используются в дальнейшем в модифицированном алгоритме проверки л-т эквивалентности программ. В этом же разделе описана процедура редукции подстановок и исследованы некоторые алгебраические свойства редуцированных подстановок (утверждения 3-5 и теорема 6). В шестом разделе статьи введена операция редуцированной антиунификации (определение 3) и установлены некоторые ее алгебраические свойства (теоремы 7-9), необходимые для применения этой операции в модифицированном алгоритме проверки л-т эквивалентности программ. В заключительном седьмом разделе статьи описана процедура вычисления редуцированной антиунификации для подстановок, представленных в виде размеченных ациклических ориентированных графов. В теореме 10 показано, что время выполнения этой процедуры и размер графа, представляющего результат редуцированной антиунификации, ограничены величиной, линейно зависящей от размеров графов, представляющих подстановки-операнды. На основе этой оценки в теореме 11 последнего раздела статьи получен основной результат работы – верхняя оценка сложности по времени $O(n^6)$ для модифицированного алгоритма проверки л-т эквивалентности программ.

2. Алгебра конечных подстановок

Для заданных множеств переменных Var, функциональных символов F и предикатных символов P обозначим записью $Term(F,Var)$ множество термов, а записью $Atom(P,F,Var)$ множество атомарных формул (атомов), которые определяются обычным образом (см. [17,18]). Для каждого терма t обозначим записью Var_t множество переменных, входящих в терм t.

Пусть X и Y – два конечных множества переменных. Подстановкой назовем всякое отображение $\theta : X \to Term(F,Y)$, сопоставляющее каждой переменной из X некоторый терм из $Term(F,Y)$. Множество всех таких подстановок условимся обозначать записью $Subst(X,F,Y)$. Если $X = \{x_1, x_2, \ldots, x_n\}$ и $\theta(x_i) = t_i$ для всех i, $1 \le i \le n$, то подстановка θ однозначно определяется множеством (списком) пар $\{x_1 / t_1, x_2 / t_2, \ldots, x_n / t_n\}s$. Если $t_i = x_i s$, то пара (связка) $\{x_i / t_i\}$ может быть опущена в этом списке. Запись Dom_θ будет обозначать множество переменных $\{x : \theta(x) \ne x\}$, а запись Var_θ будет использоваться для обозначения множества переменных $\bigcup_{i=1}^{n} Var_{t_i} s$, входящих в

состав всех термов подстановки θ. Результатом применения подстановки θ к терму t является терм $t\theta$, получающийся одновременной заменой в t каждой переменной x_i термом $\theta(x_i)$. Композиция $\theta\eta$ подстановок $\theta \in Subst(X,F,Y)$, $\eta \in Subst(Y,F,Z)$ – это подстановка из множества $Subst(X,F,Z)$, которая определяется равенством $\theta\eta(x) = (\theta(x))\eta$ для каждой переменной $x, x \in X$. Всякая биекция $\theta : Y \to Y$ называется переименованием.

На множестве подстановок $Subst(X,F,Y)$ определим отношения предпорядка $\prec$ и эквивалентности $\approx$. Для пары подстановок θ_1, θ_2 отношение $\theta_1 \prec \theta_2$ выполняется, если есть такая подстановка $\eta \in Subst(Y,F,Y)$, что $\theta_2 = \theta_1\eta$, и отношение $\theta_1 \approx \theta_2$ выполняется, если $\theta_2 = \theta_1\rho$ для некоторого переименования ρ. Если для пары подстановок θ_1, θ_2 выполняется отношение $\theta_1 \prec \theta_2$, то подстановку θ_1 будем называть *прототипом* подстановки θ_2, а подстановку θ_2 – *примером* подстановки θ_1. Отношение предпорядка $\prec$ порождает на множестве классов эквивалентности $Subst(X,F,Y)/\approx$ отношение частичного порядка $\le$. Частично упорядоченное множество $(Subst(X,F,Y)/\approx, \le)$ образует решетку, наименьшим элементом которой является класс эквивалентности, порожденный так называемой пустой подстановкой $\{x_1 / y_1, x_2 / y_2, \ldots, x_n / y_n\}s$. Чтобы сделать решетку подстановок полной, добавим к множеству подстановок в качестве наибольшего элемента специальную мнимую подстановку τ, удовлетворяющую равенствам $\tau\theta = \theta\tau = \tau s$ и $E_1\tau = E_2\tau s$ для любой подстановки θ и термов t_1, t_2. Операция взятия точной нижней грани называется антиунификацией подстановок и обозначается символом $\downarrow$. Решетка $(Subst(X,F,Y)/\approx, \le)s$ удовлетворяет условию обрыва убывающих цепей. Далее запись $\theta_1 \downarrow \theta_2 s$ будем использовать для обозначения произвольной подстановки из классов эквивалентности $\theta_1^{\approx} \downarrow \theta_2^{\approx}$. Для любой пары подстановок $\theta', \theta' \in Subst(X',F,Y)$ и $\theta'', \theta'' \in Subst(X'',F,Y)$ в случае $X' \cap X'' = \varnothing$, мы будем использовать запись $\theta' \cup \theta''$ для обозначения подстановки ηs, которая представляет собой теоретико-множественное объединение связок подстановок θ' и θ''.

3. Модель последовательных императивных программ

В качестве математической модели последовательных императивных программ мы рассматриваем множество конечных размеченных ориентированных графов. Опишем вкратце устройство этой модели. Пусть

задано некоторое конечное множество переменных X. Каждой вершине v графа π, моделирующего императивную программу, приписана атомарная формула A_v из множества $Atom(P,F,X)$. Из каждой вершины исходят две дуги, одна из которых помечена символом 0, а другая – символом 1. Кроме того, каждой дуге, ведущей в графе π из вершины u в вершину v, приписана подстановка θ_{uv} из множества $Subst(X,F,X)$. Одна из вершин v_{in} графа π особо выделена в качестве входа в программу, а другая вершина v_{out} играет роль выхода из программы. Предполагается также, что через каждую вершину графа π проходит некоторый маршрут, ведущий из входа программы в ее выход.

Вершины графа π соответствуют точкам программы, в которых проводится проверка условий и ветвление потока управления программы. Атомарные формулы, приписанные вершинам, соответствуют тестам (предикатам), а дуги графа π - линейным участкам программы. Если линейный участок представляет собой последовательность операторов присваивания $x_{i_1} := t_1; x_{i_2} := t_2; \ldots x_{i_k} := t_k;$, то соответствующей дуге приписана подстановка $\theta = \{x_{i_1} / t_1\}\{x_{i_2} / t_2\}\cdots\{x_{i_k} / t_k\}$, представляющая собой композицию односвязочных подстановок для всех операторов этого линейного участка.

Пусть задан некоторый маршрут из входа в программу π в ее выход

$$w = v_0 \xrightarrow{\sigma_1,\theta_1} v_1 \xrightarrow{\sigma_2,\theta_2} v_2 \xrightarrow{\sigma_1,\theta_1} \cdots v_{n-1} \xrightarrow{\sigma_n,\theta_n} v_n$$

Тогда последовательность пар

$$lth(w) = (A_{v_0}\mu_0,\sigma_1),(A_{v_1}\mu_1,\sigma_2),\ldots,(A_{v_{n-1}}\mu_{n-1},\sigma_n),(A_{v_n}\mu_n,1),$$

где $\mu_0 = \varepsilon$ и $\mu_i = \theta_1\theta_2\cdots\theta_i$ для каждого $i, 1 \le i \le n,$ называется *логико-термальной историей* маршрута ws. *Детерминантом* программы π называется множество

$Det(\pi) = \{lth(w) : w$ - маршрут в программе πs из входа в выход$\}$.

Две программы π_1 и $\pi_2 s$ считаются *логико-термально* (*л-т*) *эквивалентными*, если справедливо равенство $Det(\pi_1) = Det(\pi_2)$. Отношение л-т эквивалентности программ является отношением эквивалентности, которое аппроксимирует функциональную эквивалентность программ, т.е. две л-т эквивалентные программы π_1 и π_2 для любой интерпретации и для любого начального значения переменных вычисляют одинаковые результаты (значения переменных при достижении вычислением выхода из программы). Более подробно об отношениях функциональной эквивалентности и л-т эквивалентности программ можно прочесть в статьях [1-10] и монографии [16]. Задача проверки л-т эквивалентности программ состоит в том, чтобы для

произвольной заданной пары программ π_1 и π_2 проверить, являются ли эти программы л-т эквивалентными.

В статье [24] был предложен следующий алгоритм проверки л-т эквивалентности программ. Предположим, что задана пара программ π_1 и π_2 над множеством переменных $X = \{x_1, x_2, \ldots, x_n\}$. Работа алгоритма проверки л-т эквивалентности этих программ состоит из четырех этапов.

На первом этапе проводится переименование переменных в этих программах: все вхождения переменных $x_1, x_2, \ldots, x_n$ в программу π_1 заменяются на переменные $x_1', x_2', \ldots, x_n'$ соответственно, а все вхождения переменных $x_1, x_2, \ldots, x_n$ в программу π_2 заменяются на переменные $x_1'', x_2'', \ldots, x_n''$ соответственно. Получившиеся в результате указанного переименования переменных варианты программ π_1 и π_2 условимся обозначать π' и π''.

На втором этапе для пары программ π' и π'' строится граф логически совместных трасс $\Gamma[\pi', \pi'']$. Его устройство таково. Вершинами графа $\Gamma[\pi', \pi'']$ служат всевозможные пары (u', u''), где u' - вершина программы π', а u'' - вершина программы π''. Каждой вершине (u', u'') приписывается пара атомов $(A_{u'}, A_{u''})$, которыми были помечены вершины u' и u'' в программах π' и π''. В том случае, если в программе π' из вершины u' в вершину v' ведет дуга с пометками σ, θ', где $\sigma \in \{0,1\}$, и $\theta' = \{x_1' / t_1', x_2' / t_2', \ldots, x_n' / t_n'\}$, а и в программе π'' из вершины u'' в вершину v'' ведет дуга с пометками σ, θ'', где $\theta'' = \{x_1'' / t_1'', x_2'' / t_2'', \ldots, x_n'' / t_n''\}$, то в графе $\Gamma[\pi', \pi'']$ из вершины (u', u'') в вершину (v', v'') ведет дуга с пометкой $\theta = \theta' \cup \theta'' = \{x_1' / t_1', \ldots, x_n' / t_n', x_1'' / t_1'', \ldots, x_n'' / t_n''\}$.

На третьем этапе применяется процедура вычисления разметки вершин графа $\Gamma[\pi', \pi'']$ подстановками из множества $Subst(X' \cup X'', F, Y)$, где $Y = \{y_1, y_2, \ldots\}$ - бесконечное множество переменных, отличных от переменных множеств $X' = \{x_1', x_2', \ldots, x_n'\}$ и $X'' = \{x_1'', x_2'', \ldots, x''\}_n$. В начале работы этой процедуры вершина (u_{in}', u_{in}'') графа $\Gamma[\pi', \pi'']$, где u_{in}' и u_{in}'' - точки входа в программы π' и π'', помечается подстановкой $\eta_0 = \{x_1' / y_1, x_1'' / y_1, x_2' / y_2, x_2'' / y_2, \ldots, x_n' / y_n, x_n'' / y_n\}$, а все остальные вершины графа $\Gamma[\pi', \pi'']$ помечаются максимальной в решетке подстановок $(Subst(X' \cup X'', F, Y) / \approx, \le)$ подстановкой τ. Далее, для каждой пары вершин (u', u'') и (v', v''), помеченных подстановками $\eta_{u'u''}$ и $\eta_{v'v''}$ и соединенных

дугой, которой приписана подстановка θ, вычисляется подстановка $\mu = \theta\eta_{u'u''} \downarrow \eta_{v'v''}$, которая приписывается вершине (v', v'') вместо подстановки $\eta_{v'v''}$. Эта процедура выполняется до тех пор, пока в графе $\Gamma[\pi', \pi'']$ для каждой пары вершин (u', u'') и (v', v'') не будет выполняться равенство $\eta_{v'v''} \approx \theta\eta_{u'u''} \downarrow \eta_{v'v''}$.

На последнем этапе работы алгоритма проводится проверка выполнимости следующих двух условий:

1. В графе $\Gamma[\pi', \pi'']$ из вершины (u'_{in}, u''_{in}) достижима хотя бы одна из вершин вида (u'_{out}, u'') или (u', u''_{out}), где $u' \neq u'_{out}, u'' \neq u''_{out}$;
2. Некоторая вершина (u', u'') графа $\Gamma[\pi', \pi'']$ помечена подстановкой $\eta_{u'u''}$, для которой справедливо соотношение $A_{u'}\eta_{u'u''} \neq A_{u''}\eta_{u'u''}$.

Если хотя бы одно из этих условий выполнено, то исходные программы π_1 и π_2 признаются л-т неэквивалентными. В противном случае рассматриваемые программы л-т эквивалентны.

Завершаемость описанной процедуры вычисления разметки вершин графа $\Gamma[\pi', \pi'']$ следует из свойства фундированности отношения порядка в решетке подстановок $(Subst(X' \cup X'', F, Y)/\approx, \leq)$. Корректность и полнота алгоритмы проверки л-т эквивалентности программ следуют из двух теорем, в которых установлены алгебраические свойства операции антиунификации подстановок.

Теорема 1. [24] Для любой подстановки $\theta \in Subst(X, F, X)$ и пары подстановок $\eta_1, \eta_2 \in Subst(X, F, Y)$ справедливо равенство

$\theta\eta_1 \downarrow \theta\eta_2 = \theta(\eta_1 \downarrow \eta_2)$.

Теорема 2. [24] Для любой пары атомов $A, B \in Atom(P, F, X)$ любой пары подстановок $\eta_1, \eta_2 \in Subst(X, F, Y)$ справедливо соотношение

$A\eta_1 = B\eta_1 \wedge A\eta_2 = B\eta_2 \quad \Leftrightarrow \quad A(\eta_1 \downarrow \eta_2) = B(\eta_1 \downarrow \eta_2)$.

Для того чтобы оценить сложность предложенного алгоритма проверки л-т эквивалентности программ, необходимо рассмотреть некоторые способы представления подстановок и оценить сложность операций композиции и антиунификации в зависимости от этих представлений.

4. Графовые представления подстановок

Эффективность алгоритмов вычисления композиции и антиунификации подстановок существенно зависит от тех структур данных, которые используются для представления термов и подстановок. Для представления

термов целесообразно использовать графовые конструкции, которые легко моделируются ссылочными структурами данных. Опишем общий принцип, положенный в основу этих конструкций.

Рассмотрим произвольный конечный ациклический ориентированный граф $G = (V, E)$ с множеством вершин V и множеством дуг E. Размером $|G|$ графа G будем считать суммарное количество его вершин и дуг. *Разметкой* АОГ G назовем отображение, которое приписывает каждой вершине графа v символ s_v, который является либо функциональным символом из множества F или переменной из множества Var, а каждой дуге – некоторое натуральное число. *Правильной разметкой* АОГ G назовем *разметку*, удовлетворяющую следующим двум требованиям:

- если из вершины v не исходит ни одной дуги, то эта вершина помечена либо константой, либо переменной;
- если из вершины v исходят n дуг, где $n > 0$, то эта вершина помечена функциональным символом местности n, а все исходящие дуги помечены попарно различными числами из множества $\{1, 2, \ldots, n\}$.

Если в правильно помеченном АОГ G из вершины v в вершину u ведет дуга, помеченная числом m, то вершину u будем называть *m-наследником* вершины v. Каждой вершине v правильно размеченного АОГ G можно однозначно сопоставить терм t_v, который *реализуется в вершине* v, руководствуясь следующими правилами:

- если из вершины v не исходит ни одной дуги, то в этой вершине реализуется терм s_v;
- если из вершины v исходят n дуг, ведущие в вершины $u_1, u_2, \ldots, u_n$, и при этом для любого $i, 1 \leq i \leq n,$ дуга, ведущая в вершину u_i, помечена натуральным числом i, то в вершине v реализуется терм $s_v(t_{u_1}, t_{u_2}, \ldots, t_{u_n})$.

Правильно размеченный АОГ G *реализует* конечное множество термов $T, T \subseteq Term(F, Var),$ если для каждого терма $t, t \in T,$ в графе G существует вершина v, для которой $t_v = t$.

Среди различных способов реализации множеств термов T, посредством правильно размеченных АОГ, наиболее широкое распространение получили два способа – *древесный* и *приведенный*.

В первом случае АОГ G представляет собой конечное множество деревьев (лес), и каждому терму из множества T взаимно однозначно соответствует дерево из этого леса, в корне которого реализуется этот терм. Это самый

простой способ графового представления множеств термов: древесная реализация множества термов легко строится по строковой записи термов и легко получается из произвольного АОГ, реализующего множество T, путем «расклейки» вершин этого графа.

АОГ G, реализующий множество термов T, называется *приведенным*, если он удовлетворяет двум требованиям:

- в каждой вершине АОГ G реализуется некоторый подтерм какого-либо терма из множества T;
- в разных вершинах АОГ G реализуются разные термы.

Для построения приведенного АОГ из произвольного графового представления множеств термов нужно выполнить специальную процедуру редукции. Эта процедура заключается в удалении всех вершин, которые не удовлетворяют первому из указанных выше требований и в применении к АОГ (до тех пор пока это возможно) следующего правила: если две вершины u и v одинаково помечены, и при этом m-наследник вершины u совпадает m-наследником вершины v для каждого натурального числа m, то эти вершины необходимо склеить, т.е. удалить одну из этих вершин вместе с исходящими из нее дугами, а все ведущие в эту вершину дуги направить в другую вершину рассматриваемой пары. Нетрудно видеть, что в результате применения описанной процедуры к АОГ, реализующему множество термов T, будет построен приведенный граф, реализующий то же самое множество термов. Процедура может быть осуществлена за время, линейное относительно размеров исходного АОГ.

Приведенные АОГ - наиболее компактный способ представления множеств термов. Существуют такие последовательности термов, размер древесного представления которых оценивается экспонентой, зависящей от размеров приведенного представления этих же термов.

Размеченные АОГ, реализующие множества термов, можно использовать также и для реализации подстановок. Пусть заданы подстановка $\theta = \{x_1 / t_1, x_2 / t_2, \ldots, x_n / t_n\}$ из множества $Subst(X, F, Y)$ и АОГ G, реализующий множество термов $\{t_1, t_2, \ldots, t_n\}$. Сопоставим каждой переменной $x_i, 1 \le i \le n$, ту вершину v графа G, в которой реализуется соответствующий терм t_i; переменную x_i будем называть *заголовком* вершины v. Размеченный таким образом АОГ G будем называть *графовой реализацией* подстановки θ.

Графовое представление подстановок оказывает существенное влияние на сложность задач вычисления операций композиции и антиунификации подстановок. Если подстановки θ и η имеют древесные реализации G_θ и G_η, то

- размер и сложность вычисления древесной реализации $G_{\theta\eta}$ композиции этих подстановок оценивается величиной $\Omega(|G_\theta \| G_\eta|)$,
- размер древесной реализации $G_{\theta\downarrow\eta}$ антиунификации этих подстановок оценивается величиной $\Omega(\min(|G_\theta|, |G_\eta|))$, а сложность ее вычисления оценивается величиной $\Omega(|G_\theta \| G_\eta|)$.

Если же подстановки θ и η имеют приведенные реализации H_θ и H_η, то

- размер и сложность вычисления приведенной реализации $H_{\theta\eta}$ композиции этих подстановок оценивается величиной $\Omega(|H_\theta| + |H_\eta|)$,
- размер и сложность приведенной реализации $H_{\theta\downarrow\eta}$ антиунификации этих подстановок оценивается величиной $\Omega(|H_\theta \| H_\eta|)$ (см. [22]).

В статье [24] был предложен алгоритм проверки логико-термальной эквивалентности программ. Этот алгоритм представляет собой итеративную процедуру, в которой на каждом шаге для тройки подстановок $\theta_1, \theta_2, \theta_3$ вычисляется значение $(\theta_1\theta_2)\downarrow\theta_3$. Таким образом, при любом из двух рассмотренных способе графовой реализации подстановок размер АОГ, представляющего подстановку $(\theta_1\theta_2)\downarrow\theta_3$, оценивается величиной, пропорциональной произведению размеров АОГ, представляющих подстановки-операнды. Поскольку в худшем случае число итераций, которые должен выполнить процедура проверки эквивалентности, пропорционально размеру анализируемых программ, алгоритм, предложенный в работе [25] имеет экспоненциальную сложность. Один из способов преодоления возникшей трудности состоит в ограничении класса рассматриваемых подстановок и замене операции антиунификации более простой операцией, которая вычисляет некоторую нижнюю грань двух подстановок, но при этом сохраняет справедливость всех вспомогательных утверждений, представленных в статье [24] и обеспечивающих корректность процедуры проверки логико-термальной эквивалентности программ.

5. Редуцированные подстановки

Далее мы будем рассматривать множество подстановок $Subst(X, F, Y)$; переменные из конечной области определения $X = \{x_1, x_2, \ldots, x_n\}$ этих подстановок будем называть *главными переменными*, а переменные из бесконечного множества Y, используемые для построения термов, – *вспомогательными переменными*. Для каждой подстановки θ множество переменных Var_θ может быть разбито на два подмножества

$HVar_\theta = \{y : \exists x\,(x \in X \wedge \theta(x) = y)\}$ и $NVar_\theta = Var_\theta \setminus HVar_\theta$. В графовой реализации подстановки θ все вершины, помеченные вспомогательными переменными из множества $HVar_\theta$, имеют заголовки. В множестве $Subst(X,F,Y)$ выделим подкласс редуцированных подстановок.

Определение 1. Подстановка θ из множества $Subst(X,F,Y)$ называется *редуцированной подстановкой*, если $Var_\theta = HVar_\theta$.

Исследуем некоторые характеристические свойства редуцированных подстановок.

Как видно из определения 1, характеристическое свойство редуцированных подстановок состоит в том, что в их приведенном АОГ, реализующем редуцированную подстановку, все листовые вершины, которым приписаны вспомогательные переменные, озаглавлены. Очевидно также, что каждая подстановка θ имеет хотя бы один редуцированный прототип (таковым является, например, пустая подстановка $\{x_1 / y_1, x_2 / y_2, \ldots, x_n / y_n\}$). Пусть задана некоторая подстановка $\theta = \{x_1 / t_1, x_2 / t_2, \ldots, x_n / t_n\}$ и пусть y - некоторая вспомогательная переменная, не принадлежащая множеству Var_θ. Предположим, что терм t_1 содержит подтерм $f(y_1, y_2, \ldots, y_k)$ и при этом $y_j \notin HVar_\theta$ для некоторого $j, 1 \le j \le k$. Тогда для каждого $i, 1 \le i \le k$, обозначим записью t_i' терм, который либо совпадает с термом t_i в том случае, когда t_i не содержит подтерма $f(y_1, y_2, \ldots, y_k)$, либо получен из терма t_i заменой всех вхождений подтерма $f(y_1, y_2, \ldots, y_k)$ на переменную y в том случае, когда указанный подтерм содержится в терме t_i. Рассмотрим новую подстановку $\theta' = \{x_1 / t_1', x_2 / t_2', \ldots, x_n / t_n'\}$. Как видно из приведенного выше описания подстановки θ', имеет место равенство $\theta = \theta'\{y / f(y_1, y_2, \ldots, y_k)\}$. Покажем, что для пары подстановок θ и θ' и произвольной редуцированной подстановки η справедливо

Утверждение 3. Редуцированная подстановка η является прототипом подстановки θ тогда и только тогда, когда η является прототипом подстановки θ'.

Доказательство. Поскольку $\theta = \theta'\{y / f(y_1, y_2, \ldots, y_k)\}$, каждый прототип подстановки θ' является прототипом подстановки θ. Рассмотрим теперь произвольный редуцированный прототип η подстановки θ. Тогда для некоторой подстановки $\rho, \rho \in Subst(Y,F,Y)$, верны равенства $\eta\rho = \theta = \theta'\{y / f(y_1, y_2, \ldots, y_k)\}$. Без ограничения общности будем считать, что подстановка ρ не содержит связок-переименований вида $\{y'/ y''\}$. Тогда терм

$f(y_1, y_2, \ldots, y_k)$ должен входить в состав либо одного из термов в подстановке η, либо в состав одного из термов подстановки ρ. Поскольку подстановка η является редуцированной, а переменная y_j не содержится в множестве $HVar_\theta$, терм $f(y_1, y_2, \ldots, y_k)$ не содержится ни в одном из термов подстановки η. Значит, выражение $f(y_1, y_2, \ldots, y_k)$ входит только в состав термов подстановки ρ. В этом случае подстановку ρ можно представить в виде композиции $\rho = \rho'\{y / f(y_1, y_2, \ldots, y_k)\}$, где ρ' - это подстановка из класса $Subst(Y,F,Y)$, ни один терм которой не содержит вхождений выражения $f(y_1, y_2, \ldots, y_k)$. Ввиду того, что это выражение также не входит в состав ни одного из подстановки θ', приходим к заключению о выполнимости равенства $\eta\rho' = \theta'$. Оно свидетельствует о том, что редуцированная подстановка η является прототипом подстановки θ'. □

Определение 2. Редуцированная подстановка θ' называется *наиболее специальной редукцией* подстановки θ (обозначается $msr(\theta)$), если θ' является прототипом θ, и каждая редуцированная подстановка, являющаяся прототипом подстановки θ, является также примером подстановки θ'.

Непосредственно из утверждения 3 вытекает, что для каждой подстановки θ существует ее наиболее специальная редукция $msr(\theta)$, и эта редукция единственна (с точностью до переименования переменных). Опишем алгоритм построения $msr(\theta)$. Этот алгоритм работает с АОГ G_θ, реализующим подстановку θ, проводя его раскраску.

Все вершины графа G_θ, имеющие заголовок, окрашиваются в синий цвет, а все вершины, которым приписана вспомогательные переменные из множества $NVar_\theta$, окрашиваются в красный цвет. Далее строим раскраску графа, руководствуясь следующими правилами:

1. каждая красная вершина окрашивает в красный цвет все входящие в нее дуги;
2. если из вершины исходит хотя бы одна красная дуга, то и все исходящие из этой вершины дуги окрашиваются в красный цвет;
3. если вершина еще не окрашена, а все исходящие из нее дуги – красные, то эта вершина окрашивается в красный цвет;
4. все перечисленные выше действия повторяются до тех пор, пока возможно применение хотя бы одного из них.

После завершения раскраски графа

- для каждой вершины, у которой имеется заголовок и из которой исходят только красные дуги, вводится новая вспомогательная переменная, которой помечается эта вершина;
- из графа удаляются все вершины и дуги окрашенные в красный цвет.

В результате этого построения получаем размеченный граф, который является реализацией подстановки $msr(\theta)$. Время работы предложенного алгоритма пропорционально размеру графа G_θ.

Выделим несколько важных свойств, которыми обладает операция редукции подстановок.

Утверждение 4. Для любых двух подстановок θ_1 и θ_2 таких, что $\theta_1 \le \theta_2$ верно, что их наиболее специальные редукции находятся в соотношении $msr(\theta_1) \le msr(\theta_2)$.

Доказательство. Заметим, что из выполнимости отношения $\theta_1 \le \theta_2$ следует, что существует такая подстановка $\rho, \rho \in Subst(Y,F,Y)$, что выполнено равенство $\theta_2 = \theta_1\rho$. Тогда для обоснования справедливости леммы достаточно показать справедливость неравенства $msr(\theta_1) \le msr(\theta_1\rho)$. Справедливость этого отношения следует непосредственно из описания алгоритма построения редуцированных подстановок, описанного выше. □

Свойства операции редукции относительно композиции подстановок устанавливает следующее

Утверждение 5. Пусть $\theta \in Subst(X,F,X), \eta \in Subst(X,F,Y)$. Тогда $msr(\theta\eta) = msr(\theta\, msr(\eta))$.

Доказательство. Справедливость утверждения вытекает из утверждения 1 и описания алгоритма построения наиболее специальных редукций подстановок. □

Теорема 6. Для любой пары атомарных формул A, B из множества $Atom(P,F,X)$ и для любой редуцированной подстановки θ из множества $Subst(X,F,Y)$ выполняется соотношение

$$A\theta = B\theta \quad \Leftrightarrow \quad Amsr(\theta) = Bmsr(\theta).$$

Доказательство. Поскольку подстановка $msr(\theta)$ является прототипом подстановки θ, равенство $Amsr(\theta) = Bmsr(\theta)$ очевидно влечет за собой равенство $A\theta = B\theta$. С другой стороны, равенство $A\theta = B\theta$ означает, что подстановка θ является унификатором атомов A и B. Коль скоро эти атомы унифицируемы, они имеют наиболее общий унификатор. Так как атомы A и

B принадлежат классу $Atom(P,F,X)$, любой из алгоритмов унификации, описанных в работах [18-24], построит для этих атомов наиболее общий унификатор η, принадлежащий классу подстановок $Subst(X,F,X)$ и удовлетворяющий соотношению $Dom_\eta \cap Var_\eta = \varnothing$, т.е. $\eta = \{x_{i_1} / t_1, x_{i_2} / t_2, \ldots, x_{i_k} / t_k\}$ и при этом ни один терм $t_1, t_2, \ldots, t_k$ не содержит переменные из множества $\{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\}$. Пусть $Var_\theta = \{x_{j_1}, x_{j_2}, \ldots, x_{j_m}\}$; рассмотрим переименование $\lambda = \{x_{j_1} / y_1, x_{j_2} / y_2, \ldots, x_{j_m} / y_m\}$. Как известно (см. [19]), всякая подстановка, эквивалентная наиболее общему унификатору двух атомов, также является наиболее общим унификатором той же пары атомов. Поэтому подстановка $\theta' = \eta\lambda = \{x_{i_1} / t_1\lambda, x_{i_2} / t_2\lambda, \ldots, x_{i_k} / t_k, x_{j_1} / y_1, x_{j_2} / y_2, \ldots, x_{j_m} / y_m\}$ является

1. редуцированной подстановкой из класса $Subst(X,F,Y)$,
2. наиболее общим унификатором атомов A и B.

Последнее означает, что $A\theta' = B\theta'$ и для некоторой подстановки $\rho, \rho \in Subst(Y,F,Y)$, верно равенство $\theta = \theta'\rho$, и, следовательно, подстановка θ' является редуцированным прототипом подстановки. Но в таком случае для некоторой подстановки $\rho', \rho' \in Subst(Y,F,Y)$, выполняется равенство $msr(\theta) = \theta'\rho'$, и поэтому справедливо равенство $Amsr(\theta) = Bmsr(\theta)$. □

6. Редуцированная антиунификация подстановок

Определение 3. Редуцированной антиунификацией двух подстановок θ_1 и θ_2 из класса $Subst(X,F,Y)$ назовем такую подстановку θ_0 которая удовлетворяет равенству $\theta_0 = msr(\theta_1 \downarrow \theta_2)$.

Для обозначения редуцированной антиунификации двух подстановок будем использовать запись $\theta_0 = \theta_1 \Downarrow \theta_2$. Операция редуцированной антиунификации обладает определенным набором свойств, которые позволяют использовать ее для проверки логико-термальной эквивалентности.

Теорема 7. Для любой пары подстановок $\eta_1, \eta_2 \in Subst(X,F,Y)$ справедливо равенство

$$\eta_1 \Downarrow \eta_2 \approx msr(\eta_1) \Downarrow msr(\eta_2).$$

Доказательство. Заметим, что $msr(\eta_1) \le \eta_1$ и $msr(\eta_2) \le \eta_2$. Поэтому $msr(\eta_1) \downarrow msr(\eta_2) \le \eta_1 \downarrow \eta_2$. На основании утверждения 4 отсюда следует неравенство $msr(\eta_1) \Downarrow msr(\eta_2) \le \eta_1 \Downarrow \eta_2$.

Покажем теперь, что каждый редуцированный прототип подстановки $\eta_1 \downarrow \eta_2$ является также редуцированным прототипом подстановки $msr(\eta_1) \downarrow msr(\eta_2)$. Рассмотрим произвольную редуцированную подстановку θ из класса $Subst(X,F,Y)$, являющуюся прототипом подстановки $\eta_1 \downarrow \eta_2$. Тогда θ также является редуцированным прототипом обеих подстановок η_1 и η_2. Но в этом случае верны соотношения $\theta \le msr(\eta_1)$ и $\theta \le msr(\eta_2)$, и, следовательно, соотношение $\theta \le msr(\eta_1) \downarrow msr(\eta_2)$.

Таким образом, наиболее специальный редуцированный прототип подстановки $\eta_1 \downarrow \eta_2$ является некоторым редуцированным прототипом подстановки $msr(\eta_1) \downarrow msr(\eta_2)$. Отсюда следует неравенство $\eta_1 \Downarrow \eta_2 \le msr(\eta_1) \Downarrow msr(\eta_2)$.

Неравенства $msr(\eta_1) \Downarrow msr(\eta_2) \le \eta_1 \Downarrow \eta_2$ и $\eta_1 \Downarrow \eta_2 \le msr(\eta_1) \Downarrow msr(\eta_2)$ означают, что подстановки $\eta_1 \Downarrow \eta_2$ и $msr(\eta_1) \Downarrow msr(\eta_2)$ эквивалентны. □

Теорема 8. Для любой подстановки $\theta \in Subst(X,F,X)$ и пары подстановок $\eta_1, \eta_2 \in Subst(X,F,Y)$ справедливо равенство

$\theta\eta_1 \Downarrow \theta\eta_2 = msr(\theta(\eta_1 \Downarrow \eta_2))$.

Доказательство. Согласно определению редуцированной антиунификации и теореме 1 (см. [24]) верны равенства

$\theta\eta_1 \Downarrow \theta\eta_2 = msr(\theta\eta_1 \downarrow \theta\eta_2) = msr(\theta(\eta_1 \downarrow \eta_2))$.

Как следует из утверждения 5,

$msr(\theta(\eta_1 \downarrow \eta_2)) = msr(\theta\, msr(\eta_1 \downarrow \eta_2)) = msr(\theta(\eta_1 \Downarrow \eta_2))$. □

Теорема 9. Для любой пары атомов $A, B \in Atom(P,F,X)$ и любой пары подстановок $\eta_1, \eta_2 \in Subst(X,F,Y)$ справедливо соотношение

$A\eta_1 = B\eta_1 \wedge A\eta_2 = B\eta_2 \quad \Leftrightarrow \quad A(\eta_1 \Downarrow \eta_2) = B(\eta_1 \Downarrow \eta_2)$.

Доказательство. Как гласит теорема 2 (см. [24]),

$A\eta_1 = B\eta_1 \wedge A\eta_2 = B\eta_2 \quad \Leftrightarrow \quad A(\eta_1 \downarrow \eta_2) = B(\eta_1 \downarrow \eta_2)$.

Согласно утверждению 6

$A(\eta_1 \downarrow \eta_2) = B(\eta_1 \downarrow \eta_2) \quad \Leftrightarrow \quad A(\eta_1 \Downarrow \eta_2) = B(\eta_1 \Downarrow \eta_2)$. □

Обратимся теперь к алгоритму проверки л-т эквивалентности программ, описанному в разделе 3 и обоснованному в статье [24]. Заменим в процедуре разметки графа логически совместных трасс $\Gamma[\pi', \pi'']$ операцию антиунификации $\downarrow$ на операцию редуцированной антиунификации $\Downarrow$. Тогда

доказанные выше теоремы 7, 8 и 9 обеспечивают корректность и полноту модифицированного таким образом алгоритма.

7. Сложность вычисления редуцированной антиунификации подстановок

Покажем, что переход от операции унификации $\downarrow$ к операции редуцированной антиунификации $\Downarrow$ позволяет существенно повысить эффективность алгоритма проверки л-т эквивалентности программ. Прежде всего, оценим сложность построения и размер графовой реализации редуцированной антиунификации двух подстановок $\eta_1 \Downarrow \eta_2$. Для вычисления редуцированной антиунификации можно воспользоваться композицией двух алгоритмов: алгоритма вычисления антиунификации двух подстановок, описанного в статьях [13, 23], и алгоритма построения наиболее специальной редукции, описанного в разделе 3 настоящей работы. Однако оценка сложности вычисления и размера построенной таким образом подстановки $\theta = \eta_1 \Downarrow \eta_2$ может иметь квадратичную зависимость от размера графовой реализации подстановок η_1, η_2. Такие оценки неприемлемы для нашей цели – построения полиномиального по времени алгоритма проверки л-т эквивалентности программ. Поэтому ниже описан алгоритм редуцированной антиунификации подстановок, имеющий линейную сложность по времени.

Пусть заданы приведенные размеченные АОГ $G_{\eta 1}$ и $G_{\eta 2}$, реализующие пару подстановок η_1, η_2. Построение приведенного размеченного АОГ G_θ, реализующего подстановку $\theta = \eta_1 \Downarrow \eta_2$, проведем в три этапа. Вершинами графа G_θ будут упорядоченные пары (u,v), где u - вершина АОГ $G_{\eta 1}$, а v - вершина графа $G_{\eta 2}$.

На первом этапе построения введем множество вершин графа G_θ, имеющих заголовки. Чтобы осуществить это, для каждой переменной $x_i, x_i \in X$, рассмотрим пару вершин $w_i = (u_i, v_i)$, где u_i - вершина АОГ $G_{\eta 1}$, озаглавленная переменной x_i, а v_i - вершина АОГ $G_{\eta 2}$, озаглавленная той же переменной x_i. Все указанные вершины w_i входят в состав графа G_θ; при этом каждая такая вершина w_i считается озаглавленной переменной x_i. Возможно, что некоторые из указанных вершин могут быть озаглавлены несколькими различными переменными из множества X.

На втором этапе множество вершин графа G_θ пополняется новыми вершинами при помощи следующей процедуры. Предположим, что u_0 и v_0 - это вершины в АОГ $G_{\eta 1}$ и $G_{\eta 2}$ соответственно, помеченные одним и тем же

функциональным символом f, $f \in F$, местности m. Предположим также, что для любого натурального числа $i, 1 \le i \le m$, пара (u_i', v_i'), где u_i' - i-наследник вершины u_0 в АОГ $G_{\eta 1}$, а v_i' - i-наследник вершины v_0 в АОГ $G_{\eta 2}$, уже внесена в состав вершин графа G_θ. Тогда к множеству вершин графа G_θ добавляется вершина (u_0, v_0), которая помечается функциональным символом f. При этом для каждого $i, 1 \le i \le m$, из вершины (u_0, v_0) в вершину (u_i', v_i') проводится дуга, помеченная числом i. В том случае, если вершина (u_0, v_0) была введена ранее на первом этапе построения графа G_θ, осуществляется только проведение дуг из этой вершины. Применение описанной процедуры прекращается, как только к графу G_θ уже не удается добавить ни одной новой вершины и ни одной новой дуги.

На третьем этапе все листовые вершины, озаглавленные переменными из множества X (к их числу относятся только некоторые из вершин $w_i = (u_i, v_i)$, введенных на первом этапе построения графа G_θ) помечаются попарно различными вспомогательными переменными из множества Y, и после этого применяется процедура редукции, описанная в разделе 4.

Из описания процедуры построения размеченного АОГ G_θ видно, что реализуемая этим графом подстановка θ обладает следующими свойствами:

1. подстановка θ является редуцированной (это следует из замечания к определению 1 редуцированной подстановки);
2. подстановка θ является прототипом каждой из подстановок η_1, η_2;
3. каждый редуцированный прототип обеих подстановок η_1, η_2 является также и прототипом подстановки θ.

Таким образом, построенный АОГ G_θ является приведенной графовой реализацией редуцированной антиунификации $\eta_1 \Downarrow \eta_2$.

На первом этапе построения АОГ G_θ алгоритм вычисления редуцированной антиунификации совершает $|X|$ действий. Число итераций, совершаемых процедурой пополнения множества вершин графа G_θ на втором этапе его построения, не превосходит числа вершин в каждом из АОГ $G_{\eta 1}$ и $G_{\eta 2}$. Поэтому время построения АОГ G_θ можно оценить сверху величиной

$$O(|X| + \min(|G_{\eta 1}|, |G_{\eta 2}|)).$$

Чтобы оценить размер графа, реализующего результат редуцированной антиунификации двух подстановок, введем в рассмотрение еще одну

сложностную характеристику АОГ для редуцированных подстановок – сложность редуцированной подстановки. Пусть η – произвольная редуцированная подстановка из класса $Subst(X, F, Y)$, и G_η – приведенный АОГ, реализующий подстановку η. Тогда *сложностью* подстановки η назовем величину $\ell(\eta) = N_1(\eta) + |X| - N_2(\eta)$, где $N_1(\eta)$ – количество вершин в графе G_η, помеченных функциональными символами, а $N_2(\eta)$ – количество вершин в графе G_η, озаглавленных переменными из множества X.

Утверждение 10. Пусть $\theta = \eta_1 \Downarrow \eta_2$. Тогда

1. $\ell(\theta) \le \min(\ell(\eta_1), \ell(\eta_2))$;
2. Если $\ell(\theta) = \ell(\eta_1)$, то $\theta = \eta_1$.

Доказательство. Вершины АОГ G_θ, озаглавленные переменными из множества X, строятся на первом этапе. Из описания алгоритма видно, что $N_2(\theta) \ge max(N_2(\eta_1), N_2(\eta_2))$.

Каждая вершина, которая добавляется в АОГ G_θ на втором этапе вычисления редуцированной антиунификации, взаимно однозначно соответствует одной вершине графа $G_{\eta 1}$ и одной вершине графа $G_{\eta 2}$. Таким образом, верно неравенство

$N_1(\theta) \le \min(N_1(\eta_1), N_1(\eta_2))$.

Из полученных неравенств следует, что $\ell(\theta) \le \min(\ell(\eta_1), \ell(\eta_2))$. Кроме того, равенство $\ell(\theta) = \ell(\eta_1)$ означает, что графы G_θ и $G_{\eta 1}$ изоморфны, и поэтому $\theta = \eta_1$. □

Оценки трудоемкости вычисления редуцированной антиунификации двух подстановок и размеров ее графовой реализации, установленные выше, позволяют оценить трудоемкость решения задачи проверки л-т эквивалентности программ.

Теорема 11. Существует алгоритм, разрешающий задачу проверки л-т эквивалентности произвольной пары программ π_1 и π_2 за время $O(n^6)$, где $n = \max(|\pi_1|, |\pi_2|)$.

Доказательство. Воспользуемся модифицированным алгоритмом проверки л-т эквивалентности программ, описанным в разделе 3 и уточненным в разделе 6. Очевидно, что граф логически совместных трасс $\Gamma[\pi_1, \pi_2]$ имеет не более n^2. Максимальная сложность подстановки $\eta_{u'u''}$, приписанной

любой вершине (u',u'') этого графа, также не превосходит величины n^2. Как следует из утверждения 10, на каждой итерации алгоритма проверки л-т эквивалентности программ сложность хотя бы одной из подстановок, приписанных вершине графа логически совместных трасс $\Gamma[\pi_1,\pi_2]$, уменьшается. Поэтому число итераций алгоритма ограничено величиной $O(n^4)$. При этом трудоемкость вычисления редуцированной антиунификации $\mu = \theta\eta_{u'u''} \Downarrow \eta_{v'v''}$ на каждой итерации алгоритма проверки л-т эквивалентности программ ограничено величиной, которая пропорциональна максимальной сложности подстановок, приписанных вершинам графа. Отсюда следует оценка сложности алгоритма, заявленная в теореме. □

Список литературы

[1] Иткин В.Э. Логико-термальная эквивалентность схем программ // Кибернетика. — 1972. — N 1. — с. 5-27.

[2] Сабельфельд В.К. Полиномиальная оценка сложности распознавания логико-термальной эквивалентности // ДАН СССР. — 1979. — т. 249, N 4. — с. 793-796.

[3] Sabelfeld V.K. The logic-termal equivalence is polynomial-time decidable // Information Processing Letters. — 1980 — v. 10, N 2. — p. 57-62.

[4] Буда А.О., Иткин В.Э. Сводимость эквивалентности программ к термальной эквивалентности // Системное и теоретическое программирование. Сборник статей. Т. 1. — Кишенев, 1974. — с. 293-324.

[5] Буда А.О. Об одной модели вычислений над памятью с разрешимой проблемой эквивалентности // Доклады АН СССР — 1979. — т. 245, N 1.

[6] Буда А.О. Канонизация логической структуры логико-термально эквивалентных схем программ // Трансляция и модели программ. — Новосибирск, 1980.

[7] Годлевский А.Б. О связи логико-термальной эквивалентности схем программ с одним случаем специальной проблемы эквивалентности дискретных преобразователей // Кибернетика. — 1975. — N 5. — с. 32-34.

[8] Хачатрян В.Е. Логико-термальная эквивалентность в классе схем над сильно невырожденным базисом // Программирование. — 1977. — N 3. — с. 20-27.

[9] Сабельфельд В.К. Некоторые оптимизирующие преобразования для стандартных схем программ // Языки и системы программирования. — Новосибирск, 1979.

[10] Син Мен Де. Логико-термально эквивалентные преобразования схем программ // Кибернетика. — 1976 — N 5.

[11] Фаулер М. Рефакторинг. Улучшение существующего кода. — Символ-Плюс, 2008. — 432 с.

[12] Komondoor R., Horwitz S. Using slicing to identify duplication in source code // Proceedings of the 8th International Symposium on Static Analysis. — Springer-Verlag, 2001. — p. 40–56.

[13] Bulychev P., Kostylev E., Zakharov V. Anti-unification algorithms and their applications in program analysis // Proceedings of the 7th International Conference “Perspectives of System Informatics”, June 15-19, 2009, Novosibirsk. — 2009. — p. 82-89.

[14] Luckham D.C., Park D.M., Paterson M.S., On formalized computer programs // Journal of Computer and System Science — 1970. — v.4, N 3. — p. 220-249.

[15] Itkin V.E., Zwinogrodski Z. On program schemata equivalence // Journal of Computer and System Science. — 1972. — v. 6, N 1. — p. 88-101.

[16] Котов В.Е., Сабельфельд В.К. Теория схем программ. — М.:Наука, 1991. — 348 с.

[17] Eder E. Properties of substitutions and unifications // Journal of Symbolic Computations. — v. 1. — 1985. — p. 31-46.

[18] Palamidessi C. Algebraic properties of idempotent substitutions // Lecture Notes in Computer Science — v. 443 — 1990. — p. 386-399.

[19] Plotkin G.D. A note on inductive generalization // Machine Intelligence. — 1970. — v. 5, N 1. — p. 153-163.

[20] Sorensen M. H., Gluck. R. An algorithm of generalization in positive supercompilation // Proceedings of the 1995 International Symposium on Logic Programming._MIT Press. — 1995. — p. 465-479.

[21] Захаров В.А., Костылев Е.В. О сложности задачи антиунификации // Дискретная математика. — 2008. — т. 20, N 1. — с. 131-144.

[22] Zakharov V.A. On the refinement of logic programs by means of anti-unification // Proceedings of the 2nd Panhellenic Logic Symposium, Delphi, Greece. — 1999. — p. 219-224.

[23] В.А. Захаров, Т.А. Новикова. Применение алгебры подстановок для унификации программ. Труды Института системного программирования РАН, том 21, 2011 г. ISSN 2220-6426 (Online), ISSN 2079-8156 (Print).