Унификация программ

Т.А. Новикова, В.А.Захаров
{taniaelf@mail.ru}{zakh@cs.msu.su}

Аннотация. Унифицировать два алгебраических выражения t_1 и t_2 означает отыскать такую подстановку θ термов вместо переменных этих выражений, чтобы оба терма $t_1\theta$ и $t_2\theta$ имели одинаковое значение. Задачу унификации можно распространить и на программы. Унифицировать две программы π_1 и π_2 означает отыскать такие цепочки присваиваний $\rho_1 : x_1 := t_1; x_2 := t_2; \ldots x_n := t_n$ и $\rho_2 : y_1 := s_1; x_2 := s_2; \ldots x_m := s_m$, для которых композиции программ $\rho_1;\pi_1$ и $\rho_2;\pi_2$ эквивалентны (т.е. вычисляют одну и ту же функцию). В данной работе в качестве эквивалентности программ рассматривается отношение логико-термальная эквивалентности, одно из наиболее слабых разрешимых отношений эквивалентности программ, аппроксимирующих отношение функциональной эквивалентности. Опираясь на алгоритм проверки логико-термальной эквивалентности программ, мы предлагаем полиномиальную по времени процедуру вычисления наиболее общего унификатора (ρ_1, ρ_2) для произвольной пары (π_1, π_2) последовательных императивных программ относительно логико-термальной эквивалентности.

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

1. Введение

Задача унификации выражений (формул, атомов, термов) языка предикатов первого порядка состоит в том, чтобы для заданной пары выражений $E_1(x_1,\ldots,x_n)$ и $E_2(x_1,\ldots,x_n)$ отыскать такую подстановку $\theta = \{x_1 / t_1,\ldots,x_n / t_n\}$, для которой выражения $E_2(x_1,\ldots,x_n)\theta$ и $E_1(x_1,\ldots,x_n)\theta$ становятся синтаксически одинаковыми. Впервые задачу унификации исследовал Дж. Робинсон в статье [1] при разработке метода

резолюций для систем автоматического доказательства теорем. Метод резолюций послужил отправной точкой для разработки концепции логического программирования, и алгоритмы унификации стали основным средством вычисления логических программ. За прошедшие годы задача унификации была детально исследована. В частности, были разработаны эффективные алгоритмы унификации [2-4], имеющие почти линейную сложность, а также были найдены подходящие структуры данных для практической реализации этих алгоритмов.

Наряду с задачей синтаксической унификации в алгебре свободных термов изучалась также и более общая задача семантической унификации (или, иначе, Е-унификации). В алгебре термов можно ввести определяющие тождества (например, законы коммутативности, ассоциативности, дистрибутивности и др.) для некоторых функций, и индуцировать тем самым отношение конгруэнтности на множестве термов. Тогда задача семантической унификации состоит в том, чтобы для заданной пары термов E_1 и E_2 отыскать такую подстановку θ, для которой термы $E_2\theta$ и $E_1\theta$ эквивалентны. Задача семантической унификации исследовалась для алгебраических операций, подчиняющихся законам ассоциативности и коммутативности [5-7]. Некоторые из методов, предложенных в этих статьях, сводят задачу унификации к решению систем диофантовых уравнений. Было установлено также, что задача унификации выражений в логиках высокого порядка алгоритмически неразрешима (см. [8]). С методами и результатами решения задачи унификации термов и формул можно ознакомиться в обзорной статье [9].

Алгебраический терм является, в определенном смысле, простейшей разновидностью вычислительных программ, при построении которых используется только операция композиции. Более сложные программы конструируются при помощи более широкого набора операций. Применительно к вычислительным императивным программам задачу унификации можно сформулировать в такой постановке. Предположим, что на множестве программ введено некоторое отношение эквивалентности и выделен некоторый класс «простых» программ Π. Тогда для произвольной заданной пары программ π_1 и π_2 требуется выяснить, существует ли две такие пары программ (π'_{in}, π''_{in}) и $(\pi'_{out}, \pi''_{out})$ из класса Π, для которой последовательные композиции программ $\pi'_{in};\pi_1;\pi'_{out}$ и $\pi''_{in};\pi_2;\pi''_{out}$ эквивалентны. В качестве отношения эквивалентности разумно выбрать отношение функциональной эквивалентности программ или любую его аппроксимацию; в этом случае из эквивалентности программ будет следовать равенство вычисляемых этими программами функций. Пары программ (π'_{in}, π''_{in}) и $(\pi'_{out}, \pi''_{out})$ в постановке задачи унификации могут мыслиться

как интерфейсы, преобразующие формат представления входных и выходных данных. Можно предполагать, что преобразования такого рода выполняются программами, имеющими сравнительно простое устройство. Тогда эквивалентность композиций $\pi'_{in};\pi_1;\pi'_{out}$ и $\pi''_{in};\pi_2;\pi''_{out}$ означает, что программы π_1 и π_2 вычисляют схожие функции, и отношение унифицируемости программ является одной из возможных формализаций отношения подобия, обобщающего отношение эквивалентности программ.

Различные отношения подобия программ находит применение в решении задач реорганизации (рефакторинга) программ (см. [10]). В частности, актуальными задачами являются проблемы формализации понятия программного клона и эффективного выделения клонов [11]. Содержательно, программный клон – это совокупность фрагментов программы, осуществляющих «похожие» преобразования данных. Корректное определение программного клона, сопровождаемое эффективным методом обнаружения клонов, позволяет проводить упрощения программного код путем замены нескольких больших фрагментов программы вызовом одной и той же процедуры или макроса [12]. Одно из возможных определений клона можно сформулировать на основе отношения унифицируемости: клоном называется совокупность фрагментов программы $\pi_0, \pi_1, \ldots, \pi_n$, обладающая следующим свойством – любая пара программ $\pi_0, \pi_i, \quad 1 \le i \le n,$ унифицируема, причем унифицирующими интерфейсами служат такие пары программ (π'_{in}, π''_{in}) и $(\pi'_{out}, \pi''_{out})$, в которых π'_{in} и π'_{out} – это пустые программы, вычисляющие тождественные функции. В этом случае при обнаружении клона мы можем ввести в программе новую процедуру $PROC$, телом которой служит фрагмент π_0, а все фрагменты $\pi_i, 1 \le i \le n,$ заменить вызовом этой процедуры в составе композиций вида $\pi'_{in};call\ PROC;\pi'_{out}, 0 \le i \le n$.

Наиболее важным параметром в формулировке задачи унификации программ является отношение эквивалентности программ. Содержательный смысл задачи унификации требует, чтобы это отношение аппроксимировало отношение функциональной эквивалентности. В то же время, для эффективного решения задачи унификации необходимо, чтобы это выбранное отношение эквивалентности было разрешимым. Обоим требованиям удовлетворяет отношение логико-термальной (л-т) эквивалентности, введенное в статье [13]. Две программы π_1 и π_2 считаются л-т 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. Как было установлено в [13], л-т эквивалентность программ влечет их функциональную эквивалентность в любой интерпретации базовых операций и предикатов. В статье [14] показано, что проверку л-т эквивалентности программ можно провести за время, полиномиально зависящее от размеров программ. Еще более быстрый алгоритм проверки л-т эквивалентности программ был предложен в статьях [15,16]. Опираясь на эти результаты, мы покажем, что для одного класса интерфейсов-унификаторов задачу л-т унификации императивных программ также можно решить за полиномиальное время.

Содержание статьи таково. Во втором и третьем разделах мы напомним основные понятия алгебры конечных подстановок [17-18] и теории стандартных схем программ [20], которые понадобятся для формулировки и описания решения задачи л-т унификации программ. В четвертом разделе приведено упрощенное описание алгоритма вычисления наиболее общего л-т унификатора программ и обоснована его корректность. В пятом разделе показано, каким образом можно улучшить вычислительные качества описанного алгоритма л-т унификации программ, для того чтобы получить полиномиальную верхнюю оценку сложности модифицированного алгоритма. В заключение обсуждаются перспективы дальнейшего развития предложенного нами метода для решения более общих случаев проблемы унификации 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). Для каждого терма t обозначим записью Var_t множество переменных, входящих в терм t.

Для представления термов будем использовать размеченные ациклические графы (АОГ). 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, либо 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 $i, 1\le i\le n,$ 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 $t_v=t$.

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, которая за время, линейное относительно размера исходного графового представления строит приведенный АОГ, реализующий то же самое множество термов (см. [16]).

Пусть X и Y – два множества переменных. Подстановкой назовем всякое отображение $\theta: X\to Term(F,Y)$, сопоставляющее каждой переменной из множества X некоторый терм. Множество всех таких подстановок

обозначим 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 $\{x_1/t_1,x_2/t_2,\ldots,x_n/t_n\}$. Запись Var_θ будет использоваться для обозначения множества переменных $\bigcup_{i=1}^{n} Var_{t_i}$, 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 $\rho: Y\to Y$ называется переименованием. Подстановки θ_1 и θ_2 считаются эквивалентными (и это отношение обозначается записью $\theta_1\approx\theta_2$), если для некоторого переименования ρ выполняется равенство $\theta_2=\theta_1\rho$.

Размеченные АОГ 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 и использовать запись $|\theta|$ для обозначения размера этого графа.

На множестве подстановок $Subst(X,F,Y)$ определим отношения предпорядка $\prec$: 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 $\theta_2=\theta_1\eta$. В этом случае подстановку θ_1 будем называть *прототипом* подстановки θ_2, подстановку θ_2 – *примером* подстановки θ_1. Подстановку η будем называть *пополнением* прототипа θ_1 до примера θ_2 и использовать для ее обозначения запись $\frac{\theta_2}{\theta_1}$.

Прототип $\theta_1, \theta_1 \in Subst(X, F, X \cup Y),$ подстановки $\theta_2, \theta_2 \in Subst(X, F, X \cup Y)$ б назовем *редуцированным*, если для любой переменной y, $y \in Var_{\theta_1} \cap Y$, существует такая переменная x, для которой $\theta_1(x) = y$. *Наиболее специальной редукцией* подстановки θ_2 называется такая ее редукция θ_1, которая удовлетворяет отношению $\theta' \prec \theta_1$ для любой редукции θ' подстановки θ_2. В статье [16] показано, что для каждой подстановки θ существует единственная наиболее специальная редукция $msr(\theta)$ (с точностью до переименования переменных); $msr(\theta)$ может быть построена за время, линейное относительно размера графового представления подстановки θ.

Отношение предпорядка $\prec$ позволяет ввести на множестве подстановок $Subst(X, F, X \cup Y)$ операции точной верхней грани $\mathrm{lub}(\theta_1, \theta_2)$ и точной нижней грани $\mathrm{glb}(\theta_1, \theta_2)$. Первая из них вычисляет наиболее общий пример подстановок θ_1 и θ_2 и позволяет решать задачу унификации в свободной алгебре термов. Из описания алгоритма унификации термов [3] следует, что $|\mathrm{lub}(\theta_1, \theta_2)| = O(|\theta_1| + |\theta_2|)$ и сложность вычисления $\mathrm{lub}(\theta_1, \theta_2)$ пропорциональна максимальному из размеров подстановок $|\theta_1|$ и $|\theta_2|$. Вторая операция вычисляет наименее общий прототип подстановок θ_1 и θ_2 и называется антиунификацией. Чтобы операция антиунификации была всюду определенной, к множеству подстановок добавлена в качестве наибольшего элемента специальную мнимую подстановку τ, удовлетворяющую равенствам $\tau\theta = \theta\tau = \tau$ и $t_1\tau = t_2\tau$ для любой подстановки θ и термов t_1, t_2. В статье [20] показано, что сложность вычисления $\mathrm{glb}(\theta_1, \theta_2)$ и размер $|\mathrm{glb}(\theta_1, \theta_2)|$ оцениваются сверху величиной $O(|\theta_1| \times |\theta_2|)$. Такая сложность вычисления точной нижней грани подстановок затрудняет использование этой операции в итеративных алгоритмах статического анализа программ (см. [18]). Для преодоления этой трудности в статье [16] была предложена комбинированная операция *редуцированной антиунификации* подстановок $\theta_1 \Downarrow \theta_2 = msr(\mathrm{glb}(\theta_1, \theta_2))$. Как было установлено в [16], размер редуцированной антиунификации $\theta_1 \Downarrow \theta_2$ и сложность ее вычисления пропорциональны максимальному из размеров $|\theta_1|$ и $|\theta_2|$.

Для решения задачи унификации программ нам понадобится проводить унификацию атомов. Для заданного множества пар атомов $H = \{(A_i', A_i'') : i \in I\}$ из класса $Atom(P, F, X)$ *унификатором* этого множества называется подстановка θ, $\theta \in Subst(X, F, X)$, для которой равенство $A_i'\theta = A''\theta$ выполняется для любой пары атомов (A_i', A_i'') из множества H. Унификатор θ множества пар атомов H называется *наиболее общим унификатором*, если для любого унификатора θ' множества H выполняется соотношение $\theta \prec \theta'$. Задача унификации конечного множества пар атомов решается за время, линейное относительно суммарного размера атомов в множестве H (см. [3,4]).

3. Модель последовательных императивных программ

Мы рассматриваем модель последовательных императивных программ, заимствованную из работ [21,22] и адаптированную для работы с подстановками. Подробное описание этой модели приведено в статье [15].

Пусть задано некоторое конечное множество переменных X. Модель программы представляет собой размеченный ориентированный граф $\pi(\mathrm{X})$. Каждой вершине v этого графа 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 $\pi(X)$ из вершины u в вершину v, приписана подстановка $\theta[uv]$ из множества $Subst(X, F, X)$. Одна из вершин v_{in} графа $\pi(X)$ особо выделена в качестве входа в программу, а другая вершина v_{out} играет роль выхода из программы. Предполагается также, что через каждую вершину графа $\pi(X)$ проходит некоторый маршрут, ведущий из входа программы в ее выход.

Пусть задан некоторый маршрут (трасса) из входа в программу $\pi(X)$ в ее выход

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

где $\sigma_i \in \{0,1\}$, $\theta_i \in Subst(X, F, X), 1 \le i \le n$. Тогда последовательность пар

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

где $\mu_0 = \varepsilon$ и $\mu_i = \theta_i \mu_{i-1}$ для каждого $i, 1 \le i \le n,$ называется *логико-термальной историей* трассы α. *Детерминантом* программы $\pi(X)$ называется множество

$$Det(\pi(X)) = \{lth(\alpha) : \alpha \text{ - трасса в программе } \pi(X) \}.$$

Две программы $\pi_1(X)$ и $\pi_2(X)$ считаются *логико-термально* (*л-т*) *эквивалентными*, если справедливо равенство $Det(\pi_1(X)) = Det(\pi_2(X))$.

Опишем операцию применения подстановок к программам. Пусть задана программа $\pi(X)$ и подстановка $\theta, \theta \in Subst(X, F, X)$. Тогда *результатом применения подстановки* θ *к программе* $\pi(X)$ является программа $\pi(X)\theta$, которая получается из программы $\pi(X)$ введением

- новой входной вершины v_θ, которой приписывается 0-местный атом P_0;
- двух дуг, исходящих из v_θ и ведущих в вершину v_{in}; одна из этих дуг помечается символом 0, а другая – символом 1, и каждой из дуг приписывается подстановка θ.

Программу $\pi(X)\theta$ можно истолковывать как вызов процедуры с телом $\pi(X)$, в котором инициализация переменных осуществляется подстановкой θ.

Пусть конечное множество переменных X разбито на два непересекающихся множества X' и X''. Тогда подстановка $\theta, \theta \in Subst(X, F, X)$, называется *л-т унификатором программ* $\pi'(X')$ и $\pi''(X'')$, если программы $\pi'(X')\theta$ и $\pi''(X'')\theta$ являются л-т эквивалентными. Унифицируемость программ означает, что при некоторой инициализации переменных эти программы вычисляют одну и ту же функцию. Л-т унификатор θ программ $\pi'(X')$ и $\pi''(X'')$ называется *наиболее общим унификатором* (обозначается записью $НОУ(\pi'(X'), \pi''(X''))$), если для любого л-т унификатора θ' этих программ выполняется соотношение $\theta \prec \theta'$. Задача л-т унификации программ состоит в вычислении $НОУ(\pi'(X'), \pi''(X''))$.

4. Алгоритм логико-термальной унификации программ

Для решения задачи л-т унификации программ $\pi'(X')$ и $\pi''(X'')$, так же как и для решения задачи проверки л-т эквивалентности программ, мы воспользуемся графом 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''. Если в программе π' из вершины u' в вершину v' ведет дуга, помеченная символом $\sigma, \sigma \in \{0,1\}$, и этой дуге приписана подстановка $\theta' = \{x_1'/t_1', x_2'/t_2', \ldots, x_n'/t_n'\}$, а в программе π'' из вершины u'' в вершину v'' ведет дуга, помеченная тем же символом σ, и этой дуге приписана подстановка $\theta'' = \{x_1''/t_1'', x_2''/t_2'', \ldots, x_m''/t_m''\}$, то в графе $\Gamma[\pi', \pi'']$ из вершины (u', u'') в вершину (v', v'') ведет дуга с пометкой $\theta' \cup \theta'' = \{x_1'/t_1', x_2'/t_2', \ldots, x_n'/t_n', x_1''/t_1'', x_2''/t_2'', \ldots, x_m''/t_m''\}$.

Для каждого пути

$$path = (u'_{\text{in}}, u''_{\text{in}}) \xrightarrow{\theta_1} (u_1', u_1'') \xrightarrow{\theta_2} (u_2', u_2'') \xrightarrow{\theta_3} \cdots (u'_{k-1}, u''_{k-1}) \xrightarrow{\theta_k} (u_k', u_k''),$$

в графе логически совместных трасс $\Gamma[\pi', \pi'']$ запись $\theta[path]$ будет использоваться для обозначения композиции подстановок $\theta_k \cdots \theta_2 \theta_1$, приписанных дугам этого пути.

Полиномиальный по времени алгоритм проверки л-т эквивалентности программ, предложенный в статьях [15,16], основывается на следующем характеристическом свойстве графов логически совместных трасс программ.

Теорема 1 [15]. Программы $\pi'(X)$ и $\pi''(X)$ л-т эквивалентны в том и только том случае, когда для любой вершины каждого пути (u, v) графа логически совместных трасс $\Gamma[\pi', \pi'']$ и для любого пути $path$, ведущего из начальной вершины $(u'_{\text{in}}, u''_{\text{in}})$ в вершину (u, v), выполняется равенство $A[u]\theta[path] = A[v]\theta[path]$.

Из этой теоремы два важных для решения задачи л-т унификации программ следствия.

Следствие 1. Подстановка η является л-т унификатором программ $\pi'(X')$ и $\pi''(X'')$ тогда и только тогда, когда для любой вершины (u, v) графа логически совместных трасс $\Gamma[\pi', \pi'']$ и для любого пути $path$, ведущего из

начальной вершины $(u'_{\text{in}}, u''_{\text{in}})$ в вершину (u, v), выполняется равенство $A[u]\theta[path]\eta = A[v]\theta[path]\eta$.

Следствие 2. Пусть H - это множество пар $(A[u]\theta[path], A[v]\theta[path])$ для всех вершин (u, v) графа логически совместных трасс $\Gamma[\pi', \pi'']$ и всех путей $path$, ведущих в эти вершины. Тогда $НОУ(\pi'(X'), \pi''(X'')) = НОУ(H)$.

На основании этих следствий мы адаптируем алгоритм проверки л-т эквивалентности, предложенный в статье [16], для вычисления $НОУ(\pi'(X'), \pi''(X''))$.

Вначале опишем этот алгоритм в упрощенном виде, удобном для доказательства его корректности, а затем покажем, каким образом его можно модифицировать, чтобы достичь полиномиальной по времени трудоемкости.

Алгоритм вычисляет монотонно возрастающую последовательность подстановок $\rho_0 \prec \rho \prec \ldots \prec \rho_n$ из множества $Subst(X, F, X)$, начинающуюся тождественной подстановкой ρ_0. На каждом n-ом этапе его работы начальная вершина (u'_{in}, u''_{in}) графа $\Gamma[\pi', \pi'']$ помечается подстановкой ρ_n, и вслед за тем запускается процедура вычисления стационарной разметки вершин графа $\Gamma[\pi', \pi'']$ подстановками. По окончании работы процедуры очередная подстановка ρ_{n+1} указанной последовательности вычисляется путем композиции ρ_n и наиболее общего унификатора некоторого конечного множества пар атомов. Алгоритм завершает работу, когда $\rho_{n+1} \approx \rho_n$.

Процедура вычисления стационарной разметки вершин графа $\Gamma[\pi', \pi'']$ сопоставляет каждой вершине (u', u'') графа

1) подстановку $\eta[u'u'']$ из множества $Subst(X, F, X \cup Y)$, где $Y = \{y_1, y_2, \ldots\}$ – бесконечное множество переменных, отличных от переменных множества X, и
2) некоторое множество подстановок $S[u'u'']$ из класса $Subst(X, F, X)$.

В начале каждого n-го этапа работы алгоритма, $n \geq 0$, вершина (u'_{in}, u''_{in}) графа $\Gamma[\pi', \pi'']$, помечается подстановкой ρ_n, а все остальные вершины графа $\Gamma[\pi', \pi'']$ помечаются максимальным в квазирешетке подстановок

$(Subst(X' \cup X'', F, Y), \prec)$ элементом τ. Кроме того, вершине (u'_{in}, u''_{in}) приписывается множество подстановок $S[u'_{in}, u''_{in}] = \{\rho_n\}$, а всем остальным вершинам (u', u'') приписывается пустое множество подстановок $S[u', u''] = \varnothing$. Далее выполняется процедура вычисления стационарной разметки вершин графа. 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 $\mu = \theta\eta[u'u''] \Downarrow \eta[v'v'']$. Если не выполняется эквивалентность $\mu \approx \eta[v'v'']$, то вершине (v', v'') вместо подстановки $\eta[v'v'']$ приписывается подстановка μ, а вместо множества подстановок $S[v'v'']$ приписывается множество подстановок $S[v'v''] \cup \{\theta\lambda : \lambda \in S[u'u'']\}$. Выполнение процедуры продолжается до тех пор, пока в графе $\Gamma[\pi', \pi'']$ не будет построена такая разметка, что для каждой пары вершин (u', u'') и (v', v'') выполняется эквивалентность $\eta[v'v''] \approx \theta\eta[u'u''] \Downarrow \eta[v'v'']$.

По завершении процедуры разметки для каждой вершины (u, v) графа $\Gamma[\pi', \pi'']$ проверяется равенство $A[u]\eta[uv] = A[v]\eta[uv]$. Если каждое из указанных равенств выполняется, то подстановка ρ_n объявляется наиболее общим унификатором программ $\pi'(X')$ и $\pi''(X'')$. В ином случае вычисляется наиболее общий унификатор

$$\tau = НОУ(\bigcup_{(u,v)\in\Gamma(\pi',\pi'')} \bigcup_{\lambda\in S[uv]} \{(A[u]\lambda, A[v]\lambda)\}).$$

Если указанное множество пар атомов неунифицируемо, то объявляется, что программы $\pi'(X')$ и $\pi''(X'')$ также не имеют унификатора. В противном случае вычисляется подстановка $\rho_{n+1} = \rho_n\tau$, и алгоритм л-т унификации программ переходит к следующему (n+1)-ому этапу вычисления.

Как было показано в статье [16], процедура вычисления стационарной разметки графа $\Gamma[\pi', \pi'']$ всегда завершает работу. Покажем, что последовательность подстановок $\rho_0, \rho_1, \ldots, \rho_n, \ldots$ является конечной.

Лемма 1. Предположим, что на n-ом этапе работы алгоритма вершине (v', v'') приписано множество подстановок $S[v'v'']$. Тогда для любой подстановки $\lambda, \lambda \in S[v'v'']$, существует такой путь $path$, ведущий в графе логически

совместных трасс из начальной вершины $(u'_{\text{in}}, u''_{\text{in}})$ в вершину (v', v''), для которого выполняется равенство $\lambda = \theta[path]\rho_n$.

Доказательство. Проводится индукцией по числу шагов работы процедуры вычисления стационарной разметки на n-ом этапе работы алгоритма. □

Лемма 2. Предположим, что на n-ом этапе работы алгоритма вершине (v', v'') приписана подстановка $\eta[v'v'']$ и множество подстановок $S[v'v'']$. Тогда $\eta[v'v''] = \Downarrow\limits_{\lambda \in S[v'v'']} \lambda$.

Доказательство. Проводится индукцией по числу шагов работы процедуры вычисления стационарной разметки. Покажем справедливость индуктивного перехода. Предположим, что $\eta[v'v''] = \Downarrow\limits_{\lambda \in S[v'v'']} \lambda$, $\eta[u'u''] = \Downarrow\limits_{\lambda \in S[u'u'']} \lambda$, и пусть из вершины (u', u'') в вершину (v', v'') ведет дуга, которой приписана подстановка θ. Тогда на основании закона левой дистрибутивности операции композиции подстановок относительно операции редуцированной антиунификации (см. [16]), справедливо равенство $\theta\eta[u'u''] = \Downarrow\limits_{\lambda \in S[u'u'']} \theta\lambda$. Поэтому верна цепочка равенств $\mu = \theta\eta[u'u''] \Downarrow \eta[v'v''] = (\Downarrow\limits_{\lambda \in S[u'u'']} \theta\lambda) \Downarrow (\Downarrow\limits_{\lambda \in S[v'v'']} \lambda) = \Downarrow\limits_{\lambda \in S[v'v''] \cup \{\theta\lambda \,:\, \lambda \in S[u'u'']\}} \lambda$.

□

Лемма 3. Пусть $\rho_0, \rho_1, \ldots, \rho_n, \rho_{n+1}, \ldots$ – последовательность подстановок, которую вычисляет описанный алгоритм л-т унификации. Тогда для любого n, $n \geq 0$, справедливо соотношение $Var(\rho_{n+1}) \subset Var(\rho_n)$.

Доказательство. Рассмотрим n-ый этап работы алгоритма. Как утверждает лемма 1, для любой вершины (v', v'') каждая подстановка $\lambda, \lambda \in S[v'v'']$, представима в виде композиции $\lambda = \theta[path]\rho_n$ для некоторого пути $path$ в графе $\Gamma[\pi', \pi'']$. Это означает, что $Var_\lambda \subseteq Var_{\rho_n}$. Если алгоритм л-т унификации не завершил работу на n-ом этапе, то для некоторой вершины (u, v) графа $\Gamma[\pi', \pi'']$ имеет место соотношение $A[u]\eta[uv] \neq A[v]\eta[uv]$. Как следует из леммы 2, верно равенство $\eta[v'v''] = \Downarrow\limits_{\lambda \in S[v'v'']} \lambda$. В статье [16] было установлено, что 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 $A\eta_1 = B\eta_1 \wedge A\eta_2 = B\eta_2 \;\Leftrightarrow\; A(\eta_1 \Downarrow \eta_2) = B(\eta_1 \Downarrow \eta_2)$. Отсюда следует, что для некоторой подстановки $\lambda, \lambda \in S[v'v'']$, выполняется

соотношение $A[u]\lambda \neq A[v]\lambda$. Поэтому наиболее общий унификатор $\tau = НОУ(\bigcup\limits_{(u,v) \in \Gamma(\pi', \pi'')} \bigcup\limits_{\lambda \in S[uv]} \{(A[u]\lambda, A[v]\lambda)\})$ отличен от тождественной подстановки. Тогда, как видно из описания любого алгоритма л-т унификации (см., например, [4]), множество переменных Var_τ является собственным подмножеством множества переменных Var_λ хотя бы для одной подстановки $\lambda, \lambda \in S[v'v'']$. Следовательно, Var_τ является собственным подмножеством множества переменных Var_{ρ_n}, и справедливо строгое включение $Var(\rho_{n+1}) = Var(\rho_n\tau) \subset Var(\rho_n)$. □

Теорема 2. Алгоритм л-т унификации программ завершает работу для любой пары программ $\pi'(X)$ и $\pi''(X)$.

Доказательство. Из леммы 3 следует, что для последовательности подстановок $\rho_0, \rho_1, \ldots, \rho_n, \rho_{n+1}, \ldots$, вычисляемой алгоритмом л-т унификации, выполняется цепочка строгих включений $X = Var_{\rho_0} \supset Var_{\rho_1} \supset \ldots \supset Var_{\rho_n} \supset Var_{\rho_{n+1}} \supset \ldots$. Поскольку множество переменных X конечно, указанная последовательность подстановок также является конечной. □

Лемма 4. Если алгоритм л-т унификации программ завершает работу, вычислив подстановку ρ_n, то эта подстановка является л-т унификатором программ $\pi'(X)$ и $\pi''(X)$.

Доказательство. Как показано в статье [16], если вычислена стационарная разметка графа, и для некоторой вершины (u, v) выполняется равенство $A[u]\eta[uv] = A[v]\eta[uv]$, то для любого пути $path$, ведущего из начальной вершины $(u'_{\text{in}}, u''_{\text{in}})$ в вершину (u, v), выполняется равенство $A[u]\theta[path]\rho_n = A[v]\theta[path]\rho_n$. Тогда, согласно следствию 1 из теоремы 1, успешное завершение работы алгоритма л-т унификации с результатом ρ_n означает, что подстановка ρ_n является л-т унификатором программ $\pi'(X)$ и $\pi''(X)$. □

Лемма 5. Если программы $\pi'(X)$ и $\pi''(X)$ л-т унифицируемы, то алгоритм л-т унификации вычисляет подстановку ρ_n, и при этом $\rho_n \prec НОУ(\pi'(X'), \pi''(X''))$.

Доказательство. Нетрудно заметить, что наиболее общие унификаторы произвольных множеств пар атомов H_1 и H_2 обладают следующими двумя свойствами.

1. Если H_2 – унифицируемое множество пар атомов, и $H_1 \subseteq H_2$, то $НОУ(H_1) \prec НОУ(H_2)$;
2. Если $НОУ(H_1) = \tau$, то $НОУ(H_1 \cup H_2) \approx НОУ(H_1\tau \cup H_2\tau)$.

Таким образом, на основании следствия 2 теоремы 1, а также леммы 1 можно легко показать, применяя математическую индукцию по числу этапов работы алгоритма л-т унификации, что для л-т унифицируемых программ $\pi'(X)$ и $\pi''(X)$ каждая подстановка в последовательности $\rho_0, \rho_1, \ldots, \rho_n, \rho_{n+1}, \ldots$, которая вычисляется алгоритмом л-т унификации, является прототипом $НОУ(\pi'(X'), \pi''(X''))$. □

Из лемм 3, 4 и 5 вытекает теорема корректности предложенного нами алгоритма л-т унификации программ.

Теорема 3. Алгоритм л-т унификации программ объявляет о том, что подстановка ρ_n является наиболее общим л-т унификатором программ $\pi'(X)$ и $\pi''(X)$ тогда и только тогда, когда $\rho_n \in НОУ(\pi'(X'), \pi''(X''))$.

5. Сложность логико-термальной унификации программ

Описанный в предшествующем разделе алгоритм л-т унификации программ не является оптимальным, поскольку требует построения в явном виде множеств подстановок $S[uv]$, ассоциированных с вершинами. Как видно из описания алгоритма, размер этих множеств может оказаться величиной, экспоненциально зависящей от размеров программ $\pi'(X)$ и $\pi''(X)$. В этом разделе мы покажем, что подстановки из множеств $S[uv]$ можно представлять в неявном виде, используя лишь полиномиальный относительно размеров программ объем памяти.

Обозначим записью N суммарный размер программ $\pi'(X)$ и $\pi''(X)$. Тогда граф $\Gamma[\pi', \pi'']$ имеет не более N^2 вершин, и максимальный размер каждой подстановки, приписанной вершине этого графа, не превосходит величины $O(N^2)$.

На каждом этапе работы алгоритма л-т унификации программ выполняется процедура вычисления стационарной разметки вершин графа логически

совместных трасс $\Gamma[\pi', \pi'']$ при помощи операции редуцированной антиунифкации: 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 (в случае неэквивалентности подстановок μ и $\eta[v'v'']$) приписывается вершине (v', v'') вместо прежней подстановки $\eta[u'u'']$. Обозначим записью $list[v'v'']$ список подстановок, которые были последовательно приписаны вершине (v', v'') по ходу работы алгоритма. Как следует из утверждения 10, приведенного в статье [16], длина такого списка (равная количеству изменений пометки вершины графа $\Gamma[\pi', \pi'']$) не превосходит величины $O(N^2)$.

При вычислении подстановки $\mu = \theta\eta[u'u''] \Downarrow \eta[v'v'']$ также вычисляются две подстановки-пополнения $\xi' = \dfrac{\mu}{\theta\eta[u'u'']} = \{y_1 / t_1, \ldots, y_k / t_k\}$ и $\xi'' = \dfrac{\mu}{\eta[v'v'']} = \{y_1 / s_1, \ldots, y_k / s_k\}$. Эти подстановки применяются к одному и тому же множеству вспомогательных переменных $\{y_1, \ldots, y_k\}$, которые вводятся в термах подстановки μ при выполнении редуцированной антиунификации (подробности см. в [15,16,20]). Будем далее говорить, что подстановки ξ' и ξ'' *определяют множество переменные* $y_1, \ldots, y_k$. Если подстановка μ приписывается вершине (v', v''), то между подстановками μ, ξ' и ξ'' установим отношение предшествования: подстановки ξ' и ξ'' объявляются предшественниками подстановки μ (для обозначения отношения предшествования будем использовать записи $\mu \to \xi'$ и $\mu \to \xi''$). В свою очередь, предшественниками подстановки ξ' объявляются все предшественники (если таковые есть) подстановки $\eta[u'u'']$, а предшественниками подстановки ξ'' объявляются все предшественники подстановки $\eta[v'v'']$. Обозначим транзитивное замыкание отношения предшествования записью $\xrightarrow{*}$. Общее число подстановок-пополнений, порождаемых по ходу работы процедуры вычисления стационарной разметки графа $\Gamma[\pi', \pi'']$, ограничено величиной $O(N^4)$, и суммарный размер их оценивается величиной $O(N^6)$.

Лемма 6. Предположим, что на некотором шаге работы процедуры вычисления стационарной разметки графа логически совместных трасс $\Gamma[\pi',\pi'']$ вершине (v',v'') приписана подстановка $\eta[v'v'']$ и множество подстановок $S[v'v'']$. Тогда:

1. для любой подстановки λ из множества $S[v'v'']$ существует такая последовательность подстановок-пополнений, связанных отношением предшествования $\eta[v'v''] \to \xi_1 \to \xi_2 \to \cdots \to \xi_m$, для которой справедливо равенство $\lambda = \eta[v'v'']\xi_1\xi_2 \ldots \xi_m$;
2. для любой последовательность подстановок-пополнений, связанных отношением предшествования $\eta[v'v''] \to \xi_1 \to \xi_2 \to \cdots \to \xi_m$, существует такая подстановка λ из множества $S[v'v'']$, для которой справедливо равенство $\lambda = \eta[v'v'']\xi_1\xi_2 \ldots \xi_m$.

Доказательство. Проводится индукцией по числу шагов работы процедуры вычисления стационарной разметки графа $\Gamma[\pi',\pi'']$. При обосновании индуктивного перехода ограничимся рассмотрением первого утверждения леммы. Обоснование второго утверждения проводится сходным образом.

Предположим, что в вершине (v',v'') произошло изменение разметки: подстановка $\eta[v'v'']$ была заменена подстановкой $\mu = \theta\eta[u'u''] \Downarrow \eta[v'v'']$, а множество подстановок $S[v'v'']$ было заменено множеством подстановок $S[v'v''] \cup \{\theta\lambda' : \lambda' \in S[u'u'']\}$. В этом случае вводятся новые подстановки дополнения ξ' и ξ'', удовлетворяющие равенствам $\theta\eta[u'u''] = \mu\xi'$ и $\mu = \eta[v'v'']\xi''$. Тогда по индуктивному предположению для любой подстановки λ из множества $S[v'v'']$ верны равенства $\lambda = \eta[v'v'']\xi_1\xi_2 \ldots \xi_m = \eta[v'v'']\xi''\xi_1\xi_2 \ldots \xi_m$, а для любой подстановки λ из множества $\{\theta\lambda' : \lambda' \in S[u'u'']\}$ верны равенства $\lambda = \theta\lambda' = \theta\eta[u'u'']\xi_1\xi_2 \ldots \xi_m = \mu\xi'\xi_1\xi_2 \ldots \xi_m$, подтверждающие первое утверждение леммы для новой разметки вершины (v',v''). □

Таким образом, вся информация о подстановках из множеств, ассоциированных с вершинами графа $\Gamma[\pi',\pi'']$, содержится в подстановках-пополнениях, связанных отношением предшествования. Покажем теперь, как можно быстро вычислять «на лету» наиболее общий унификатор

$\tau = НОУ(\bigcup_{(u,v)\in\Gamma(\pi',\pi'')} \bigcup_{\lambda\in S[uv]} \{(A[u]\lambda, A[v]\lambda)\})$, используя введенное неявное описание подстановок.

После того, как построена стационарная разметка $\eta[uv]$, для вычисления подстановки τ можно воспользоваться следующей процедурой $UNIF$, которая перемежает выполнение алгоритмом унификации Мартелли-Монтанари (см. [4,15]) и применение подстановок-пополнений. Вначале этот алгоритм применяется для решения системы уравнений $\{A[u]\eta[uv] = A[v]\eta[uv] : (u,v) \in \Gamma[\pi',\pi'']\}$ и конструирует равносильную приведенную систему уравнений. Если такой системы получить не удается, то унификация исходных пар атомов невозможна. Если такая приведенная система уравнений E может быть построена, то рассматривается множество вспомогательных переменных $\{y_1, y_2, \ldots, y_m\}$ и среди подстановок-пополнений выбирается максимальная по отношению $\xrightarrow{*}$ пара подстановок ξ' и ξ'', определяющих переменную одну из переменных y_i рассматриваемого множества $\{y_1, y_2, \ldots, y_m\}$. После этого система уравнений E преобразуется в систему уравнений $E\xi' \cup E\xi''$, в которой переменная y_i уже будет отсутствовать. Далее процедура $UNIF$ вновь применяется к полученной системе алгоритм Мартелли-Монтанари. Поскольку отношение $\xrightarrow{*}$ является отношением строгого частичного порядка на множестве подстановок-пополнений, и каждая переменная определяется лишь в паре таких подстановок ξ' и ξ'', удаленная вспомогательная переменная y_i уже не появится в системах уравнений на последующих этапах выполнения процедуры $UNIF$. Выполнение описанного процесса унификации завершается, когда в очередной приведенной системе уравнений E не остается вспомогательных переменных, т.е. $E = \{x = t : x \in \hat{X}, \hat{X} \subseteq X' \cup X''\}$. В этом случае подстановка $\{x/t : x \in \hat{X}\}$ объявляется результатом его работы.

На основании леммы 6 и свойств корректности и полноты алгоритма унификации Мартелли-Монтанари справедлива.

Лемма 7. Описанный процесс последовательного применения процедуры $UNIF$ вычисляет подстановку $\{x/t : x \in \hat{X}\}$ тогда и только тогда, когда

наиболее общий унификатор $НОУ(\bigcup_{(u,v)\in\Gamma(\pi',\pi'')}\bigcup_{\lambda\in S[uv]}\{(A[u]\lambda, A[v]\lambda)\})$ существует. При этом $\tau = \{x/t : x \in \hat{X}\}$.

Нетрудно видеть, что процедура $UNIF$ $O(N^4)$ раз применяет алгоритм унификации Мартелли-Монтанари к системам, содержащим $O(N^4)$ уравнений, и размер термов в каждом из уравнений оценивается величиной $O(N^2)$. Таким образом, вычисление $НОУ(\bigcup_{(u,v)\in\Gamma(\pi',\pi'')}\bigcup_{\lambda\in S[uv]}\{(A[u]\lambda, A[v]\lambda)\})$ можно осуществить за время $O(N^{10})$.

Теорема 4. Алгоритм л-т унификации программ $\pi'(X)$ и $\pi''(X)$ завершает работу за время, полиномиальное относительно суммарного размера этих программ.

Доказательство. Пусть N – суммарный размер программ $\pi'(X)$ и $\pi''(X)$.Как следует из леммы 3, алгоритм л-т унификации программ $\pi'(X)$ и $\pi''(X)$ завершает работу за $O(N)$ этапов. На каждом этапе выполняется процедура вычисления стационарной разметки графа логически совместных трасс $\Gamma[\pi',\pi'']$, которая завершает работу за $O(N^4)$ шагов. На каждом шаге выполняется операция редуцированной антиунификации, которая применяется к подстановкам размера $O(N^2)$. По завершении построения стационарной разметки наиболее общий унификатор $НОУ(\bigcup_{(u,v)\in\Gamma(\pi',\pi'')}\bigcup_{\lambda\in S[uv]}\{(A[u]\lambda, A[v]\lambda)\})$ вычисляется за время $O(N^{10})$. Таким образом, общее время работы описанного в данной статье алгоритма л-т унификации оценивается величиной $O(N^{11})$. □

6. Заключение

Успешное решение задачи л-т унификации программ в простой постановке, когда в качестве интерфейсов-инициализаторов (π'_{in}, π''_{in}) используются лишь цепочки операторов присваивания, создает предпосылки для изучения более общего случая, когда в качестве интерфейсов (π'_{in}, π''_{in}) и $(\pi'_{out}, \pi''_{out})$ можно использовать произвольные ациклические программы. Другое направление исследований – это изучение возможности понижения сложности алгоритма л-т унификации программ.

Список литературы

[1] Robinson J.A. A machine-oriented logic based on the resolution principle // Journal of the ACM. 1965 — v. 12, N 1. — p. 23-41.

[2] Baxter L.D. An efficient unification algorithm // Technical Report CS-73-23, Dep. of Analysis and Comp. Sci., University of Waterloo, Ontario, Canada, 1973.

[3] Paterson M.S., Wegman M.N. Linear unification // The Journal of Computer and System Science. — 1978. — v. 16, N 2 — p. 158-167.

[4] Martelli A., Montanari U. An efficient unification algorithm //ACM Transactions on Program, Languages and Systems. — 1982. — v. 4, N 2 — p. 258-282.

[5] Stickel E.M. A unification algorithm for associative-commutative functions // Journal of the association for Computing Machinary. — 1981. — v. 28, N 5 — p. 423-434.

[6] Herold A., Sieckmann J. Unification in Abelean semigroups // Jornal of Automated Reasoning. — 1983. — p. 247-283.

[7] Lincoln P., Christian J. Adventures in associative-commutative unification //Journal of Symbolic Computation. — 1989. — v.8. — p. 393 — 416.

[8] Baader F., Snyder W. Unification theory // In J.A. Robinson and A. Voronkov, editors, Handbook of Automated Reasoning. — 2001. — v. 1 — p. 447-533.

[9] Goldfarb W.D. The undecidability of the second-order unification problem // Theoretical Computer Science. — 1981. — v. 13, N 2. — p. 225-230.

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

[11] Roy C. K., Cordy J. R. A survey on software clone detection research // Technical report TR 2007-541, School of Computing, Queen's University. — 2007. — v. 115.

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

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

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

[16] Захаров В.А., Новикова Т.А.. Полиномиальный по времени алгоритм проверки логико-термальной эквивалентности программ. Труды Института системного программирования РАН, том 22, 2012 г. ISSN 2220-6426 (Online), ISSN 2079-8156 (Print). DOI: 10.15514/ISPRAS-2012-22-23.

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

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

[21] 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.

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

On Program Unification

V.A.Zakharov (ISP RAS, Moscow Russia),
T.A. Novikova (Faculty of Computational Mathematics and Cybernetics,
Lomonosov Moscow State University, Moscow, Russia)
{zakh@cs.msu.su} {taniaelf@mail.ru}

Annotation. To unify a pair of algebraic expressions t_1 and t_2 is to find out such a substitution θ that both terms $t_1\theta$ and $t_2\theta$ have the same meaning. Unification problem can be extended to computational programs. To unify a pair of programs π_1 and π_2 is to build two sequences of assignment statements $\rho_1 : x_1 := t_1; x_2 := t_2; \ldots x_n := t_n$ and $\rho_2 : y_1 := s_1; x_2 := s_2; \ldots x_m := s_m$, such that both compositions of programs $\rho_1;\pi_1$ и $\rho_2;\pi_2$ are equivalent (i.e. they compute the same function). In this paper we deal with logic-and-term equivalence introduced in 1972 by V. Itkin. This is the most weak decidable equivalence on programs that approximates the functional equivalence. Based on the polynomial-time equivalence checking algorithm for logic-and-term equivalence in the first-order model of imperative programs we introduce a polynomial-time unification algorithm which computes the most general unifier (ρ_1, ρ_2) for every pair of sequential imperative programs (π_1, π_2) w.r.t. logic-and-term equivalence. After discussing the importance of unification proble for program refactoring and optimization we briefly recall the basic concepts of term substitution theory and theory of program schemata. Then we introduce a formal model of sequential programs we deal with, and finally we present our unification algorithm, prove its correctness and show that its complexity is polynomial on the size of programs to be unified

Keywords: program, logic-and-term equivalence, substitution, unification, complexity.

References

[1]. Robinson J.A. A machine-oriented logic based on the resolution principle. Journal of the ACM. 1965, v. 12, N 1, p. 23-41.
[2]. Baxter L.D. An efficient unification algorithm. Technical Report CS-73-23, Dep. of Analysis and Comp. Sci., University of Waterloo, Ontario, Canada, 1973.
[3]. Paterson M.S., Wegman M.N. Linear unification. The Journal of Computer and System Science. 1978, v. 16, N 2, p. 158-167.
[4]. Martelli A., Montanari U. An efficient unification algorithm. ACM Transactions on Program, Languages and Systems. 1982, v. 4, N 2, p. 258-282.
[5]. Stickel E.M. A unification algorithm for associative-commutative functions. Journal of the association for Computing Machinary. 1981, v. 28, N 5, p. 423-434.
[6]. Herold A., Sieckmann J. Unification in Abelean semigroups. Journal of Automated Reasoning. 1983, p. 247-283.

[7]. Lincoln P., Christian J. Adventures in associative-commutative unification. Journal of Symbolic Computation. 1989, v. 8, p. 393-416.
[8]. Baader F., Snyder W. Unification theory. In J.A. Robinson and A. Voronkov, editors, Handbook of Automated Reasoning. 2001, v. 1, p. 447-533.
[9]. Goldfarb W.D. The undecidability of the second-order unification problem. Theoretical Computer Science. 1981, v. 13, N 2, p. 225-230.
[10]. Fowler M. Refactoring. Improving the Design of Existing Code. Addison-Wesley, 1999.
[11]. Roy C. K., Cordy J. R. A survey on software clone detection research. Technical report TR 2007-541, School of Computing, Queen's University. 2007, v. 115.
[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]. Itkin V.E.: Local- termal equivalence of program schemata. Proceedings of the International Sympoisum on Theoretical Programming 1972: p. 127-143.
[14]. Sabelfeld V.K.: The Logic-Termal Equivalence is Polynomial-Time Decidable. Information Processing Letters. 1980, v.10, N 2, p. 57-62.
[15]. Zakharov V.A., Novikova T.A. Primineniye algebry podstanovok dly unifikatcii program [On the application of substitution algebra to program unification]. Trudy ISP RAN [The Proceedings of ISP RAS], 2011, v. 21, p. 141-166.
[16]. Zakharov V.A., Novikova T.A.. Polynomialny po vremeni algoritm proverki logiko-termalnoy ekviviletnosti program [Polynomial time algorithm for checking strong equivalence of programs]. Trudy ISP RAN [The Proceedings of ISP RAS], 2012, v. 22, p. 435-455. DOI: 10.15514/ISPRAS-2012-22-23.
[17]. Eder E. Properties of substitutions and unifications. Journal of Symbolic Computations. 1985. v. 1, p. 31-46.
[18]. Palamidessi C. Algebraic properties of idempotent substitutions. Lecture Notes in Computer Science. 1990, v. 443, p. 386-399.
[19]. V.E., Sabelfeld V.K. Teorija shem programm [Theory of program schemes] . — M.:Nauka, 1991. — 348 s.
[20]. Zaharov V.A., Kostylev E.V. O slozhnosti zadachi antiunifikacii [On the complexity of anti-unification]. Diskretnaja matematika [Discrete Mathematics]. — 2008. — t. 20, N 1. — s. 131-144.
[21]. 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.
[22]. Kotov V.E., Sabelfeld V.K. Teorija shem program [Theory of program schemes]. — M.:Nauka, 1991. — 348 s.