Math-Net.Ru

Общероссийский математический портал

Е. В. Костылев, В. А. Захаров, О сложности задачи антиунификации, *Дискрет. матем.*, 2008, том 20, выпуск 1, 131–144

DOI: <https://doi.org/10.4213/dm996>

Использование Общероссийского математического портала Math-Net.Ru подразумевает, что вы прочитали и согласны с пользовательским соглашением

<http://www.mathnet.ru/rus/agreement>

Параметры загрузки:

IP: 54.80.97.219

26 апреля 2023 г., 16:57:35

О сложности задачи антиунификации

© 2008 г. Е. В. Костылев, В. А. Захаров

В статье представлен новый алгоритм антиунификации логических выражений, представленных ациклическими ориентированными графами, и оценена его сложность. Задача антиунификации состоит в том, чтобы для двух заданных выражений отыскать наименее общее выражение (шаблон), примерами которого являются оба исходных выражения. Предложен алгоритм антиунификации, сложность которого линейно зависит от размера вычисляемого им наименее общего шаблона. Таким образом, устанавливается, что при измерении сложности алгоритмов относительно размеров исходных данных и вычисляемого результата задача антиунификации имеет почти такую же сложность, что и задача унификации. Показано также, что существуют такие выражения, наименее общий шаблон которых имеет размер $O(n^2)$, где n — размер графов, представляющих исходные выражения.

1. Введение

В статье представлен новый алгоритм антиунификации логических выражений (термов, формул) и оценена его сложность. Задача антиунификации состоит в том, чтобы для двух заданных выражений E_1 и E_2 отыскать наименее общее выражение E_0 , примерами которого являются оба выражения E_1 и E_2 , то есть существуют такие подстановки ϑ_1 и ϑ_2 , для которых выполняются равенства $E_1 = E_0\vartheta_1$ и $E_2 = E_0\vartheta_2$. Такое выражение E_0 называется наименее общим шаблоном выражений E_1 и E_2 . Задача антиунификации двойственна задаче унификации. Последняя состоит в том, чтобы для двух заданных выражений E_1 и E_2 отыскать наиболее общее выражение E_0 , которое является примером обоих выражений E_1 и E_2 , то есть удовлетворяет равенствам $E_0 = E_1\eta_1$ и $E_0 = E_2\eta_2$ для некоторых подстановок η_1 и η_2 . Такое выражение E_0 называется наиболее общим примером выражений E_1 и E_2 . Задача унификации была впервые исследована Дж. Робинсоном [17] в 1965 г. в связи с созданием метода резолюций для автоматического доказательства теорем. В дальнейшем метод резолюций послужил отправной точкой для разработки концепции логического программирования, и алгоритмы унификации фактически стали основным средством вычисления логических программ. За прошедшие годы задача унификации была детально исследована. В частности, был разработан широкий спектр эффективных алгоритмов унификации [2, 3, 5, 10, 14], имеющих почти линейную сложность, а также были найдены подходящие структуры данных для практической реализации этих алгоритмов.

Проблема антиунификации возникает при решении некоторых задач анализа программ и, в частности, при вычислении инвариантов программ. Так, например, полиномиальный алгоритм проверки логико-термальной эквивалентности программ, предложенный в [18],

фактически проводит вычисление и сравнение наименее общих шаблонов термов, которые сопоставляются в качестве значений переменным анализируемых программ. Задача антиунификации (построения наименее общих шаблонов) рассматривалась в ряде работ [4, 7, 8, 9, 12, 13, 15, 16, 19, 20, 21]; в этих работах были предложены различные алгоритмы антиунификации, и в некоторых случаях была исследована сложность этой задачи. Так, например, в [8] было установлено, что задача антиунификации для термов, представленных в виде деревьев, являются NC -полной (в отличие от задачи унификации, которая относится к числу P -полных задач). В [4] были предложены параллельные алгоритмы антиунификации для термов, представленных в виде деревьев. Было показано, что антиунификацию термов, представленных деревьями размера n , можно провести за время $O(\log^2 n)$ с использованием $n / \log^2 n$ процессоров в модели параллельных вычислений EREW PRAM, не допускающей одновременное считывание и запись данных несколькими процессорами в одну и ту же ячейку памяти. В этой же работе было показано, что в том случае, когда нескольким процессорам разрешается одновременно проводить операции считывания и записи, относящиеся к одной и той же ячейке общей памяти (модель параллельных вычислений CRCW PRAM), антиунификацию древесных термов можно провести за время $O(\log n)$ с использованием $n / \log n$ процессоров. Однако, сложность задачи антиунификации для термов, представленных структурами данных, отличными от деревьев, ранее не исследовалась. Цель настоящей работы — выяснить, какова сложность задачи антиунификации в том случае, когда рассматриваемые термы представлены ациклическими ориентированными графами. Нам удалось построить алгоритм антиунификации, сложность которого линейно зависит от размера вычисляемого им наименее общего шаблона. Таким образом, устанавливается, что при измерении сложности алгоритмов относительно размеров исходных данных и вычисляемого результата задача антиунификации имеет почти такую же сложность, что и задача унификации. Нам удалось обнаружить также один необычный эффект, который подчеркивает существенное комбинаторное отличие задачи антиунификации от задачи унификации. Известно, что размер наиболее общего примера двух выражений может экспоненциально зависеть от размеров самих выражений, если эти выражения представлены в виде деревьев. Однако в том случае, когда выражения представляются ациклическими ориентированными графами, размер наиболее общего примера будет ограничен величиной, линейно зависящей от размеров исходных выражений. Для задачи антиунификации все происходит наоборот: если выражения представлены в виде деревьев, то размер наименее общего шаблона двух выражений не превосходит размеров самих исходных выражений, а если для представления выражений использовать ациклические ориентированные графы (более компактную структуру данных), то размер наименее общего шаблона может квадратично зависеть от размеров исходных выражений. Таким образом, сложность алгоритмов антиунификации существенно зависит от способа представления исходных данных.

Статья состоит из пяти разделов. Вначале приводятся определения основных понятий, используемых для формулировки задачи антиунификации, и отмечаются некоторые наиболее важные алгебраические свойства подстановок. В следующем разделе рассказывается о том, каким образом можно использовать алгоритмы антиунификации для вычисления инвариантов программ. Необходимость вычисления инвариантов возникает при решении многих задач верификации, оптимизации и синтеза программ [1], и поэтому эффективные алгоритмы антиунификации могут найти широкое практическое применение в составе инструментальных средств анализа программ. В четвертом разделе рассказывается о графовых способах представления подстановок. Далее приводится описание нового алгоритма антиунификации, обосновывается его корректность и оценивается его

сложность. В шестом разделе приводится пример последовательности пар подстановок, размер наименее общих шаблонов которых квадратично зависит от размеров исходных подстановок.

2. Основные понятия

Рассмотрим алфавит, состоящий из конечного множества $\mathcal{F} = \{f_1^{(n_1)}, \dots, f_m^{(n_m)}\}$ функциональных символов валентностей $n_1, \dots, n_m$ и множества $\mathcal{U}$ переменных. Множество термов $\text{Term}(\mathcal{U}, \mathcal{F})$ определим традиционным образом:

(1) $\mathcal{U} \subseteq \text{Term}(\mathcal{U}, \mathcal{F})$,

(2) если $f^{(n)} \in \mathcal{F}$ и $t_1, \dots, t_n \in \text{Term}(\mathcal{U}, \mathcal{F})$, то $f^{(n)}(t_1, \dots, t_n) \in \text{Term}(\mathcal{U}, \mathcal{F})$.

Пусть $\mathcal{X}$ — некоторое множество переменных. Подстановкой назовем всякое отображение $\vartheta : \mathcal{X} \rightarrow \text{Term}(\mathcal{U}, \mathcal{F})$, для которого $\text{Dom}(\vartheta) = \{x \in \mathcal{X} \mid \vartheta(x) \neq x\}$ — конечное множество. Подстановки удобно представлять в виде списка связок $\vartheta = \{x_1/\vartheta(x_1), \dots, x_n/\vartheta(x_n)\}$, где $\{x_1, \dots, x_n\} = \text{Dom}(\vartheta)$. Через $\text{Var}(\vartheta)$ обозначим множество переменных из $\mathcal{U}$, которые встречаются в термах $\vartheta(x_1), \dots, \vartheta(x_n)$. Для обозначения множества подстановок с областью определения $\mathcal{X}$ и областью значений $\text{Term}(\mathcal{U}, \mathcal{F})$ будем использовать запись $\text{Subst}_{\mathcal{X}, \mathcal{U}, \mathcal{F}}$. Результатом применения подстановки ϑ к терму $t \in \text{Term}(\mathcal{X}, \mathcal{F})$ называется терм $t\vartheta$, который получается одновременной заменой в t каждой переменной x_i на терм $\vartheta(x_i)$. Композиция $\vartheta\xi$ подстановок $\vartheta \in \text{Subst}_{\mathcal{X}, \mathcal{U}, \mathcal{F}}$, $\xi \in \text{Subst}_{\mathcal{U}, \mathcal{U}, \mathcal{F}}$ определяется следующим образом: $(\vartheta\xi)(x) = (\vartheta(x))\xi$. Подстановка ρ , которая является биекцией из $\mathcal{U}$ на $\mathcal{U}$, называется переименованием.

На множестве $\text{Subst}_{\mathcal{X}, \mathcal{U}, \mathcal{F}}$ определим отношение предпорядка $\sqsubseteq$ и отношение эквивалентности $\sim$: для пары подстановок ϑ_1, ϑ_2 условимся считать, что $\vartheta_1 \sqsubseteq \vartheta_2$, если существует такая $\xi \in \text{Subst}_{\mathcal{U}, \mathcal{U}, \mathcal{F}}$, что $\vartheta_2 = \vartheta_1\xi$, и $\vartheta_1 \sim \vartheta_2$, если $\vartheta_2 = \vartheta_1\rho$ для некоторого переименования ρ . Отношение предпорядка $\sqsubseteq$ индуцирует на множестве классов эквивалентности $\text{Subst}_{\mathcal{X}, \mathcal{U}, \mathcal{F}}^{\sim}$ отношение частичного порядка $\leq$. Частично упорядоченное множество $(\text{Subst}_{\mathcal{X}, \mathcal{U}, \mathcal{F}}^{\sim}, \leq)$ было исследовано в работах [6, 9, 13]. Это множество образует решетку, наименьшим элементом которой является класс эквивалентности, порожденный так называемой пустой подстановкой $\varepsilon = \{x_1/y_1, \dots, x_n/y_n\}$. Операция взятия точной нижней грани в этой решетке называется антиунификацией подстановок и обозначается символом $\downarrow$. Нетрудно убедиться (и это было отмечено в указанных работах), что операция композиции дистрибутивна относительно антиунификации, то есть для любой подстановки $\eta \in \text{Subst}_{\mathcal{X}, \mathcal{X}, \mathcal{F}}$ и подстановок $\vartheta_1, \vartheta_2 \in \text{Subst}_{\mathcal{X}, \mathcal{U}, \mathcal{F}}$ выполняется равенство $\eta(\vartheta_1^{\sim} \downarrow \vartheta_2^{\sim}) = \eta\vartheta_1^{\sim} \downarrow \eta\vartheta_2^{\sim}$. Кроме того, решетка $(\text{Subst}_{\mathcal{X}, \mathcal{U}, \mathcal{F}}^{\sim}, \leq)$ обладает тем свойством, что для всякой подстановки ϑ длина любой цепи между классами эквивалентности $\varepsilon^{\sim}$ и $\vartheta^{\sim}$ конечна. В дальнейшем запись $\vartheta_1 \downarrow \vartheta_2$ будет использоваться для обозначения произвольной подстановки из класса эквивалентности $\vartheta_1^{\sim} \downarrow \vartheta_2^{\sim}$.

3. Вычисление инвариантов программ при помощи антиунификации

В этом разделе мы покажем, каким образом задачу вычисления инвариантов программ можно свести к задаче антиунификации. Предлагаемый метод вычисления инвариантов

требует многократного итеративного вычисления точных нижних граней подстановок, размер которых может линейно зависеть от размера анализируемой программы. Таким образом, разработка эффективных алгоритмов антиунификации — это задача, имеющая важные практические приложения.

Пусть задано конечное множество переменных $\mathcal{X} = \{x_1, \dots, x_n\}$ и конечный алфавит $\mathcal{F} = \{f_1^{(n_1)}, \dots, f_m^{(n_m)}\}$ функциональных символов. Оператором назовем всякое выражение вида $x := \tau$, где $x \in \mathcal{X}$, $\tau \in \text{Term}(\mathcal{X}, \mathcal{F})$. Программа Π представляет собой конечный помеченный ориентированный граф, в котором выделена одна вершина v_1 (она будет называться начальной точкой программы), и каждой вершине графа приписан некоторый оператор $x := \tau$. В дальнейшем вершины указанного графа будем называть точками программы. Для исследования задачи вычисления инвариантов нас не будут интересовать логические условия (предикаты) и точки выхода из программы, поэтому предложенное здесь определение задает упрощенную модель программы. Всякий конечный ориентированный путь, исходящий из начальной точки, будем называть трассой. Для обозначения множества всех трасс программы Π , оканчивающихся в точке v , будем использовать запись $\text{Tr}(v)$.

Операционная семантика программы определяется следующим образом. Пусть задана некоторая структура первого порядка с равенством $I = (D_I, \bar{f}_1, \dots, \bar{f}_m, =)$ сигнатуры $\mathcal{F}$ с предметной областью D_I , интерпретацией функциональных символов $\bar{f}_1, \dots, \bar{f}_m$ и предикатом равенства $=$. Оценкой переменных $\mathcal{X}$ в структуре I назовем всякое отображение $\sigma: \mathcal{X} \rightarrow D_I$, сопоставляющее переменным множества $\mathcal{X}$ элементы предметной области D_I в качестве их значений. Для каждого терма $\tau \in \text{Term}(\mathcal{X}, \mathcal{F})$ и оценки σ условимся использовать запись $\tau[\sigma]_I$ для указания того значения, которое принимает в структуре I терм τ в том случае, когда значения переменных терма τ определяются оценкой σ . Вычислением программы Π с начальной точкой v_1 в структуре I для некоторой оценки σ_0 называется всякая конечная последовательность пар

$$(v_1, \sigma_1), (v_2, \sigma_2), \dots, (v_k, \sigma_k), \quad (1)$$

которая удовлетворяет следующим двум требованиям.

- (1) Последовательность точек $v_1, v_2, \dots, v_k$ образует трассу в программе Π .
- (2) Последовательность оценок $\sigma_1, \sigma_2, \dots, \sigma_k$ такова, что для любой точки v_i , $1 \leq i \leq k$, которой приписан оператор $x := \tau$, оценки σ_{i-1} и σ_i связаны соотношением

$$\sigma_i(x') = \begin{cases} \sigma_{i-1}(x'), & \text{если } x' \neq x, \\ t[\sigma_{i-1}]_I, & \text{если } x' = x. \end{cases}$$

Приведенное здесь определение вычисления является наиболее простой формализацией привычного представления о вычислении как о последовательности переходов из одной точки программы в другую, сопряженных изменением значения переменных, стоящих в левых частях операторов присваивания. Мы будем говорить, что вычисление (1) завершается в точке v_k с результатом σ_k . Как видно из приведенного определения, между множеством трасс программы Π и множеством вычислений программы Π в структуре I существует взаимно однозначное соответствие.

Пусть задана программа Π и структура I . Тогда всякая формула $\Phi(x_1, \dots, x_n)$, свободные переменные которой содержатся в множестве $\mathcal{X}$, называется инвариантом программы Π в точке v_k для структуры I , если для любого вычисления (1), оканчивающегося в точке v_k , формула $\Phi(x_1, \dots, x_n)$ выполняется для оценки переменных σ_k , то есть

$I \models \Phi(x_1, \dots, x_n)[\sigma_k]$. Практическая важность задачи вычисления инвариантов обусловлена тем, что многие методы верификации программ (см. [1, 11]) предусматривают сведение задачи проверки правильности программ к вычислению подходящих инвариантов в различных точках программы.

Задача вычисления инвариантов программ нуждается в уточнении.

1. Как видно из приведенного определения, в одной и той же точке программы существует бесконечно много разных инвариантов для заданной структуры. В частности, любая тождественно истинная формула может выступать в роли инварианта. Поэтому интерес представляют только нетривиальные инварианты, существенно зависящие от специфики анализируемой программы Π . Будем говорить, что инвариант $\Phi(x_1, \dots, x_n)$ в точке v является наиболее специальным, если всякий инвариант $\Psi(x_1, \dots, x_n)$ программы Π в точке v для структуры I логически следует из формулы $\Phi(x_1, \dots, x_n)$, то есть имеет место выполнимость $I \models \Phi(x_1, \dots, x_n) \rightarrow \Psi(x_1, \dots, x_n)$.

2. Инвариант может существенно зависеть от той структуры I , которая определяет интерпретацию базовых операций $f_1^{(n_1)}, \dots, f_m^{(n_m)}$. Поскольку в большинстве реальных программ широко используются арифметические операции, задача проверки того, что некоторая формула является инвариантом программы для структуры I , оказывается алгоритмически неразрешимой. Поэтому разумно усилить требования, предъявляемые к вычисляемым инвариантам: будем говорить, что формула $\Phi(x_1, \dots, x_n)$ является сильным инвариантом в точке v программы Π , если $\Phi(x_1, \dots, x_n)$ является инвариантом в точке v для любой структуры I . Тогда вычисление сильных инвариантов можно проводить в свободной алгебре термов сигнатуры $\mathcal{F}$, и это гораздо более простая задача.

Таким образом, рассматриваемая нами задача формулируется следующим образом: для заданной программы Π вычислить наиболее специальные сильные инварианты в каждой точке программы.

Для вычисления сильных инвариантов программы можно использовать подстановки, руководствуясь следующими правилами:

- каждой точке v , которой приписан оператор присваивания $x_i := \tau$, сопоставим подстановку $\eta_v = \{x_i/\tau\}$;
- каждой трассе T , проходящей через точки $v_1, \dots, v_k$, сопоставим композицию подстановок $\vartheta_T = \eta_{v_k} \dots \eta_{v_1}$;
- каждой подстановке $\vartheta = \{x_1/t_1, \dots, x_n/t_n\} \in \text{Subst}_{\mathcal{X}, \mathcal{Y}, \mathcal{F}}$ сопоставим формулу

$$\Phi_{\vartheta}(x_1, \dots, x_n) = \exists y_1 \exists y_2 \dots \exists y_m (x_1 = t_1 \wedge \dots \wedge x_n = t_n), \quad (2)$$

где $y_1, y_2, \dots, y_m$ — все переменные из $\mathcal{Y}$, входящие в термы $t_1, \dots, t_n$.

Тогда справедливы следующие утверждения.

Теорема 1. Пусть задана программа Π и подстановка $\vartheta = \{x_1/t_1, \dots, x_n/t_n\}$ из $\text{Subst}_{\mathcal{X}, \mathcal{Y}, \mathcal{F}}$. Тогда формула Φ_{ϑ} является сильным инвариантом программы в точке v тогда и только тогда, когда для любой трассы $T \in \text{Tr}(v)$ выполняется отношение $\vartheta \sqsubseteq \vartheta_T$.

Доказательство. Достаточно показать, что утверждение верно для свободной алгебры термов $I_E = (\text{Term}(\mathcal{Y}, \mathcal{F}), f_1^{(n_1)}, \dots, f_m^{(n_m)}, =)$ с начальной оценкой σ_0 такой, что $\sigma_0(x_i) = y_i^0$, $1 \leq i \leq n$. Каждая оценка в структуре I_E является подстановкой из множества $\text{Subst}_{\mathcal{X}, \mathcal{Y}, \mathcal{F}}$. Поэтому, как видно из определения вычисления программы, для всякой

трассы T из $\text{Tr}(v)$ соответствующее этой трассе вычисление $(v_1, \sigma_1), (v_2, \sigma_2), \dots, (v_k, \sigma_k)$ оканчивается в точке $v_k = v$ с оценкой $\sigma_k = \vartheta_T$.

Рассмотрим произвольную трассу T из $\text{Tr}(v)$. Формула $\Phi_\vartheta(x_1, \dots, x_n)$ является сильным инвариантом программы в точке v в том и только том случае, если справедливо соотношение

$$\Phi_\vartheta(x_1, \dots, x_n)[\sigma_k] = \exists y_1 \exists y_2 \dots \exists y_m (\vartheta_T(x_1) = t_1 \wedge \dots \wedge \vartheta_T(x_n) = t_n),$$

то есть существует такая подстановка $\xi = \{y_1/t'_1, \dots, y_m/t'_m\} \in \text{Subst}_{\mathcal{U}, \mathcal{F}}$, для которой выполняются соотношения $I_E \models (\vartheta_T(x_i) = t_i)[\xi]_{I_E}$ для всех i , $1 \leq i \leq n$. Так как I_E — свободная алгебра, последнее равносильно тому, что $\vartheta\xi = \vartheta_T$, то есть $\vartheta \sqsubseteq \vartheta_T$. Доказательство завершено.

Теорема 2. Пусть задана точка v в программе Π , и пусть $\vartheta_0 = \downarrow_{T \in \text{Tr}(v)} \vartheta_T$. Тогда формула Φ_{ϑ_0} является сильным инвариантом программы Π в точке v . При этом Φ_{ϑ_0} — это наиболее специальный инвариант среди всех сильных инвариантов вида (2) в точке v .

Доказательство. Пусть $\vartheta_0 = \{x_1/t_1^0, \dots, x_n/t_n^0\}$. Так как $\vartheta_0 \sqsubseteq \vartheta_T$ для любой трассы $T \in \text{Tr}(v)$, по теореме 1 подстановка ϑ_0 является сильным инвариантом.

Рассмотрим некоторую подстановку $\vartheta = \{x_1/t_1, \dots, x_n/t_n\}$ такую, что Ψ_ϑ — сильный инвариант в точке v . По теореме 1 выполняется соотношение $\vartheta \sqsubseteq \vartheta_0$, из чего следует, что существует подстановка $\xi \in \text{Subst}_{\mathcal{U}, \mathcal{F}}$ такая, что $\vartheta\xi = \vartheta_0$, то есть $t_i = t_i^0\xi$, $1 \leq i \leq n$. Но это значит, что верна формула

$$I_E \models \exists y_1^0 \dots \exists y_m^0 (x_1 = t_1^0 \wedge \dots \wedge x_n = t_n^0) \rightarrow \exists y_1 \dots \exists y_m (x_1 = t_1 \wedge \dots \wedge x_n = t_n),$$

то есть Ψ_{ϑ_0} является более специальным инвариантом, чем Ψ_ϑ . Теорема доказана.

Таким образом, вычисление наиболее специальных сильных инвариантов можно свести к вычислению точных нижних граней в решетке подстановок. Как известно из теории статического анализа программ [11], в том случае, если операция композиции элементов нижней полурешетки дистрибутивна относительно операции взятия точной нижней грани (как это имеет место для решетки подстановок), то задачу вычисления точных нижних граней вида $\downarrow_{T \in \text{Tr}(v)} \vartheta_T$ в каждой точке программы можно свести к задаче вычисления наименьшего решения специальной системы уравнений. Уравнения этой системы формируются следующим образом. Каждой точке v анализируемой программы сопоставляется переменная X_v , принимающая значения в некоторой полурешетке (в нашем случае, в решетке подстановок). Пусть $\{u_1, \dots, u_k\}$ — это множество всех точек программы, из которых исходят дуги (переходы) в точку v . Тогда в точке v формируется уравнение $\mathcal{E}_v: X_v = \vartheta_v(X_{u_1} \downarrow \dots \downarrow X_{u_k})$. Существуют различные методы решения полученных таким образом систем уравнений (см. [11]); эффективность каждого из этих методов в значительной мере определяется эффективностью алгоритма вычисления точных нижних граней элементов полурешетки (в нашем случае, эффективностью алгоритмов антиунификации).

4. Графовые представления подстановок

Как известно, любой терм t из множества $\text{Term}(\mathcal{U}, \mathcal{F})$ можно представить в виде упорядоченного ориентированного помеченного дерева. Каждая вершина этого дерева взаимно

однозначно соответствует некоторому вхождению подтерма t' в терм t . При этом вершина v' , соответствующая какому-либо вхождению подтерма $t' = f^{(n)}(t_1, \dots, t_n)$ или $t' = y$, помечена символом $f^{(n)}$ (соответственно, y); из вершины v' исходит n упорядоченных дуг, которые ведут в вершины дерева $v_1, \dots, v_n$, соответствующие вхождениям подтермов $t_1, \dots, t_n$ в терм t' . Таким образом, листья дерева будут помечены переменными из множества $\mathcal{U}$ и 0-местными функциональными символами (константами).

Еще более компактным способом представления термов могут служить ациклические ориентированные графы (АОГ), в которых нескольким различным вхождениям одного и того же подтерма $t' = f^{(n)}(t_1, \dots, t_n)$ или $t' = y$ сопоставляется одна и та же вершина, помеченная символом $f^{(n)}$ или y . АОГ, представляющий терм t , можно получить, если в соответствующем упорядоченном ориентированном дереве склеить друг с другом изоморфные размеченные поддеревья. Если в АОГ, представляющем терм t , уже не существует изоморфных подграфов, то такой АОГ назовем редуцированным.

В дальнейшем условимся использовать запись $\text{mark}_G(v)$ для обозначения того символа из множества $\mathcal{U} \cup \mathcal{F}$, который приписан вершине v в графе G , представляющем некоторый терм, запись $\text{desc}_G(v, i)$ — для обозначения i -го наследника вершины v , а запись $T_G(v)$ — для обозначения того терма, который представлен вершиной v в графе G . Индекс G может быть опущен, если из контекста ясно, какому графу принадлежит вершина v .

В дальнейшем мы будем полагать, что рассматриваемые нами термы представлены в виде редуцированных АОГ. Каждому набору термов можно сопоставить редуцированный АОГ, образующийся в результате склейки изоморфных подграфов из леса тех деревьев, которые соответствуют термам этого набора. В частности, если термы t_1 и t_2 не содержат ни одной общей переменной или константы (функционального символа валентности 0), редуцированный АОГ, соответствующий паре термов t_1, t_2 , является простым объединением пары АОГ, представляющих каждый из этих термов. Мы воспользуемся этим наблюдением при доказательстве теоремы 4.

Всякую подстановку $\vartheta = \{x_1/t_1, \dots, x_n/t_n\}$ из множества $\text{Subst}_{\mathcal{X}, \mathcal{U}, \mathcal{F}}$ также можно представить в виде АОГ $\mathcal{G}(\vartheta)$, снабдив граф, представляющий набор термов $t_1, \dots, t_n$, дополнительными метками $x_1, \dots, x_n$, при помощи которых выделены вершины, соответствующие термам $t_1, \dots, t_n$. Размером подстановки $N(\vartheta)$ назовем число вершин в представляющем ее редуцированном АОГ $\mathcal{G}(\vartheta)$.

5. Алгоритм антиунификации

Предложенный нами алгоритм вычисления точной нижней грани двух подстановок представлен на рис. 1. Его принцип работы таков. На вход процедуре MST (most specific template) подаются редуцированные АОГ $\mathcal{G}(\vartheta')$, $\mathcal{G}(\vartheta'')$, представляющие подстановки $\vartheta' = \{x_1/t_1, \dots, x_n/t_n\}$ и $\vartheta'' = \{x_1/s_1, \dots, x_n/s_n\}$ из $\text{Subst}_{\mathcal{X}, \mathcal{U}, \mathcal{F}}$. Записи $v'(x)$ и $v''(x)$ обозначают те вершины графов $\mathcal{G}(\vartheta')$ и $\mathcal{G}(\vartheta'')$, соответственно, которые имеют дополнительную метку x из множества $\{x_1, \dots, x_n\}$. В алгоритме используются вспомогательные тройки элементов вида (y, v', v'') , где y — переменная из множества $\mathcal{U}$, а v' и v'' — вершины графов $\mathcal{G}(\vartheta')$ и $\mathcal{G}(\vartheta'')$, соответственно. Множества таких троек составляют списки $\mathcal{Q}$ и $\widehat{\mathcal{Q}}$. Алгоритм строит АОГ $\mathcal{G}_\eta$, представляющий точную нижнюю грань $\vartheta' \downarrow \vartheta''$ подстановок. Запись $v(x)$ обозначает вершину графа $\mathcal{G}_\eta$, помеченную переменной x . Каждая новая вершина графа $\mathcal{G}_\eta$ вводится вместе с новой переменной z из множества $\mathcal{U}$, и запись $\text{mem}(z)$ будет обозначать ту вершину АОГ $\mathcal{G}_\eta$, которая была введена вместе с переменной z .

Всякое множество троек $\mathcal{Q} = \{(y_1, v'_1, v''_1), \dots, (y_k, v'_k, v''_k)\}$, которое формируется


```

procedure MST ( $\mathcal{G}(\vartheta'), \mathcal{G}(\vartheta'')$ )
L0: выбрать новые переменные  $y_1, \dots, y_n$  из множества  $\mathcal{Y}$ , не входящие в  $\vartheta', \vartheta''$ ;
 $\mathcal{G}_\eta := \mathcal{G}(\{x_1/y_1, \dots, x_n/y_n\})$ ;  $\mathcal{Q} := \emptyset$ ;  $\widehat{\mathcal{Q}} := \emptyset$ ;
for  $i := 1$  to  $n$  do
L1: if  $\exists y : (y, v'(x_i), v''(x_i)) \in \mathcal{Q}$  then
    в  $\mathcal{G}_\eta$  удалить вершину  $v(x_i)$ ;
    в  $\mathcal{G}_\eta$  выбрать вершину  $v$  такую, что  $\text{mark}(v) = y$ , и пометить ее  $x_i$ 
else
     $\mathcal{Q} := \mathcal{Q} \cup \{(y_i, v'(x_i), v''(x_i))\}$ ;  $\text{mem}(y_i) = v(x_i)$ 
fi
od;
L2: while  $\exists y : (y, v', v'') \in \mathcal{Q}$  and  $\text{mark}(v') = \text{mark}(v'')$  do
L3: выбрать произвольный элемент  $(y, v', v'') \in \mathcal{Q}$ ,
    у которого  $\text{mark}(v') = \text{mark}(v'')$ ;
 $\mathcal{Q} := \mathcal{Q} \setminus \{(y, v', v'')\}$ ;  $\widehat{\mathcal{Q}} := \widehat{\mathcal{Q}} \cup \{(y, v', v'')\}$ ;
    if  $\text{mark}(v') = f^{(m)} \in \mathcal{F}$  then
L4: в  $\mathcal{G}_\eta$  выбрать вершину  $v = \text{mem}(y)$ ;  $\text{mark}(v) := f^{(m)}$ ;
    for  $i := 1$  to  $m$  do
L5: if  $\exists z_i : (z_i, \text{desc}(v', i), \text{desc}(v'', i)) \in \mathcal{Q} \cup \widehat{\mathcal{Q}}$  then
L6: в  $\mathcal{G}_\eta$  выбрать вершину  $w_i = \text{mem}(z_i)$ 
    else
L7: добавить к АОГ  $\mathcal{G}_\eta$  новую вершину  $w_i$ ;
    выбрать в множестве  $\mathcal{Y}$  новую переменную  $z_i$ ;
     $\text{mark}(w_i) := z_i$ ;  $\text{mem}(z_i) := w_i$ ;
L8:  $\mathcal{Q} := \mathcal{Q} \cup \{(z_i, \text{desc}(v', i), \text{desc}(v'', i))\}$ 
    fi;
L9: направить  $i$ -ю дугу, исходящую из вершины  $v$ , в вершину  $w_i$ 
    od
fi
od;
 $\mathcal{G}(\vartheta' \downarrow \vartheta'') := \mathcal{G}_\eta$ 
end of MST.

```

Рис. 1. Алгоритм антиунификации MST

в процессе работы алгоритма, представляет пару идемпотентных подстановок $\xi'_Q = \{y_1/T(v'_1), \dots, y_k/T(v'_k)\}$ и $\xi''_Q = \{y_1/T(v''_1), \dots, y_k/T(v''_k)\}$ из $\text{Subst}_{\mathcal{Y}, \mathcal{Y}, \mathcal{F}}$. Идемпотентность подстановки $\xi = \{y_1/t_1, \dots, y_k/t_k\}$ подразумевает выполнимость равенства $\xi\xi = \xi$, благодаря которому ξ можно представить в виде композиции отдельных связей $\xi = \{y_1/t_1\} \dots \{y_k/t_k\}$.

Заметим, что благодаря проверкам в точках L1 и L5 в множестве $\mathcal{Q} \cup \widehat{\mathcal{Q}}$ никогда не появляется двух различных троек (y_i, v'_i, v''_i) и (y_j, v'_j, v''_j) , таких что $v'_i = v'_j$ и $v''_i = v''_j$.

Для обоснования корректности и завершаемости алгоритма MST, нам потребуется следующая лемма.

Лемма 1. *Соотношение*

$$\forall z((z, u', u'') \in \mathcal{Q} \cup \widehat{\mathcal{Q}} \Rightarrow (T(u') = T(\text{mem}(z))\xi'_Q \wedge T(u'') = T(\text{mem}(z))\xi''_Q)) \quad (3)$$

является инвариантом основного цикла L2 алгоритма MST.

Доказательство. Обоснование проведем индукцией по числу итераций основного цикла, ограничившись исследованием равенства $T(v') = T(\text{mem}(z))\xi'_Q$. Равенство $T(v'') = T(\text{mem}(z))\xi''_Q$ доказывается аналогично.

Перед началом работы основного цикла

$$\widehat{\mathcal{Q}} = \emptyset, \quad \mathcal{Q} = \{(y_{i_1}, v'(x_{i_1}), v''(x_{i_1})), \dots, (y_{i_l}, v'(x_{i_l}), v''(x_{i_l}))\}, \quad 1 \leq i_1 < \dots < i_l \leq n,$$

где для каждого j , $j \notin \{i_1, \dots, i_l\}$, существует i , $i \in \{i_1, \dots, i_l\}$, такое, что $v'(x_j) = v'(x_i)$ и $v''(x_j) = v''(x_i)$. Тогда для любого $i = i_1, \dots, i_l$ верны соотношения $\text{mem}(y_i) = v(x_i)$ и $T(\text{mem}(y_i)) = T(v(x_i)) = y_i$. Вследствие этого

$$T(v'(x_i)) = t_i = y_i \{y_i / t_i\} = y_i \xi'_Q = T(\text{mem}(y_i)) \xi'_Q.$$

Предположим, что по окончании k -й итерации $\mathcal{Q} = Q^k$, $\mathcal{G} = G^k$ и при этом выполняется соотношение (3). Допустим, что в начале следующей $(k+1)$ -й итерации из множества Q^k была выбрана тройка (y, v', v'') , и по окончании $(k+1)$ -й итерации $\mathcal{Q} = Q^{k+1}$, $\widehat{\mathcal{Q}} = \widehat{Q}^{k+1}$, $\mathcal{G} = G^{k+1}$. Рассмотрим произвольную тройку (z, u', u'') из $Q^{k+1} \cup \widehat{Q}^{k+1}$ и вершину $w = \text{mem}(z)$. Возможны следующие варианты расположения вершины w в АОГ G^{k+1} .

1. Вершина w не содержится в подграфе G^k графа G^{k+1} . Это означает, что $w = w_i$, где w_i — одна из новых вершин, добавленных к АОГ G^k при выполнении действий в точке Л7. Тогда, согласно описанию алгоритма MST, верны соотношения $z = z_i = T_{G^{k+1}}(w)$ и $(z, u', u'') \in Q^{k+1}$. Поэтому

$$T(u') = z_i \{z_i / T(u')\} = T_{G^{k+1}}(w) \xi'_{Q^{k+1}}.$$

2. Вершина w содержится в АОГ G^k , но терм $T_{G^{k+1}}(w)$ не содержит вхождений переменной y . Тогда из описания алгоритма следует, что в результате выполнения действий в точках Л4, Л6 и Л7, преобразующих граф $\mathcal{G}_\eta$, терм $T(w)$ не подвергается изменениям, то есть $T_{G^k}(w) = T_{G^{k+1}}(w)$. А поскольку терм $T_{G^k}(w)$ не содержит ни переменной y , ни новых переменных, которые могли быть введены в результате выполнения действий в точке Л7, справедливо соотношение $T_{G^k}(w) \xi'_{Q^k} = T_{G^k}(w) \xi'_{Q^{k+1}}$. В таком случае искомое равенство $T(u') = T_{G^{k+1}}(w) \xi'_{Q^{k+1}}$ непосредственно следует из индуктивной гипотезы $T(u') = T_{G^k}(w) \xi'_{Q^k}$.

3. Вершина w содержится в АОГ G^k , и терм $T_{G^{k+1}}(w)$ содержит вхождения переменной y . Пусть $\text{mem}(y) = w_0$, и $f^{(m)} = \text{mark}(w_0)$ — это символ, приписанный в АОГ G^{k+1} вершине w_0 . Из описания алгоритма в точке Л4 следует равенство $\text{mark}(v') = \text{mark}(w_0)$, и поэтому

$$T(v') = f^{(m)}(T(\text{desc}(v', 1)), \dots, T(\text{desc}(v', m))). \quad (4)$$

Рассмотрим в графе G^{k+1} вершины $w_1, w_2, \dots, w_m$, в которые по порядку были проведены дуги из вершины w_0 при выполнении действий в точке Л9. Тогда

$$T_{G^{k+1}}(w) = T_{G^k}(w) \{y / f^{(m)}(T_{G^{k+1}}(w_1), \dots, T_{G^{k+1}}(w_m))\}. \quad (5)$$

Заметим, что поскольку G^{k+1} — ациклический граф, для тех вершин w_i , которые присутствовали в графе G^k , термы $T_{G^{k+1}}(w_i)$ не содержат вхождений переменной y . Поэтому на основании пп. 1 и 2 доказательства леммы для каждой вершины w_i , $i = 1, \dots, m$, справедливо равенство

$$T(\text{desc}(v', i)) = T_{G^{k+1}}(w_i) \xi'_{Q^{k+1}}. \quad (6)$$

Кроме того, учитывая идемпотентность рассматриваемых подстановок, а также тот факт, что $(y, v', v'') \in Q^k$, можно заключить, что верны равенства

$$\xi'_{Q^k} = \{y/T(v')\}\xi'_{Q^k}, \quad \xi'_{Q^{k+1}}\xi'_{Q^{k+1}} = \xi'_{Q^{k+1}}. \quad (7)$$

Таким образом, с использованием (4)–(7) мы приходим к цепочке равенств

$$\begin{aligned} T(u') &= T_{G^k}(w)\xi'_{Q^k} = T_{G^k}(w)\{y/T(v')\}\xi'_{Q^k} \\ &= T_{G^k}(w)\{y/f^{(m)}(T(\text{desc}(v', 1)), \dots, T(\text{desc}(v', m)))\}\xi'_{Q^k} \\ &= T_{G^k}(w)\{y/f^{(m)}(T_{G^{k+1}}(w_1)\xi'_{Q^{k+1}}, \dots, T_{G^{k+1}}(w_m)\xi'_{Q^{k+1}})\}\xi'_{Q^k} \\ &= T_{G^k}(w)\{y/f^{(m)}(T_{G^{k+1}}(w_1)\xi'_{Q^{k+1}}, \dots, T_{G^{k+1}}(w_m)\xi'_{Q^{k+1}})\}\xi'_{Q^{k+1}} \\ &= T_{G^k}(w)\{y/f^{(m)}(T_{G^{k+1}}(w_1), \dots, T_{G^{k+1}}(w_m))\}\xi'_{Q^{k+1}}\xi'_{Q^{k+1}} \\ &= T_{G^k}(w)\{y/f^{(m)}(T_{G^{k+1}}(w_1), \dots, T_{G^{k+1}}(w_m))\}\xi'_{Q^{k+1}} = T_{G^{k+1}}(w)\xi'_{Q^{k+1}}. \end{aligned} \quad (8)$$

Здесь следует иметь в виду, что равенство (8) выполняется, поскольку терм $T_{G^k}(w)\{y/f^{(m)}(T_{G^{k+1}}(w_1)\xi'_{Q^{k+1}}, \dots, T_{G^{k+1}}(w_m)\xi'_{Q^{k+1}})\}$ не содержит ни переменной y , ни новых переменных, введенных в результате выполнения действий в точке L7, и поэтому подстановки ξ'_{Q^k} и $\xi'_{Q^{k+1}}$ оказывают одинаковое воздействие на этот терм. Лемма доказана.

Теорема 3. *Существует алгоритм антиунификации, который для любых двух подстановок ϑ' , ϑ'' , представленных в виде редуцированных ациклических ориентированных графов, вычисляет редуцированный ациклический ориентированный граф, представляющий точную нижнюю грань $\vartheta' \downarrow \vartheta''$, за время $O(N \log N)$, где $N = N(\vartheta' \downarrow \vartheta'')$.*

Доказательство. Алгоритм MST, примененный к паре подстановок ϑ' , ϑ'' , всегда завершает свою работу, поскольку из описания алгоритма видно, что на протяжении всей работы алгоритма в множество $\mathcal{Q}$ может быть добавлено не более $N(\vartheta')N(\vartheta'')$ элементов; на каждой итерации основного цикла алгоритма из множества $\mathcal{Q}$ всегда удаляется один элемент.

Пусть $\vartheta' = \{x_1/t_1, \dots, x_n/t_n\}$ и $\vartheta'' = \{x_1/s_1, \dots, x_n/s_n\}$. Как следует из леммы 1 и описания действий алгоритма MST до точки L2, для любого $i = 1, \dots, n$ на каждой итерации основного цикла имеют место равенства

$$t_i = T(v(x_i))\xi'_{Q^i}, \quad s_i = T(v(x_i))\xi''_{Q^i}.$$

Поэтому подстановка $\eta = \{x_1/T_{\mathcal{G}_\eta}(v(x_1)), \dots, x_n/T_{\mathcal{G}_\eta}(v(x_n))\}$, которую на каждом шаге итерации представляет граф $\mathcal{G}_\eta$, формируемый алгоритмом MST, удовлетворяет равенствам

$$\vartheta' = \{x_1/t_1, \dots, x_n/t_n\} = \{x_1/T_{\mathcal{G}_\eta}(v(x_1))\xi'_{Q^1}, \dots, x_n/T_{\mathcal{G}_\eta}(v(x_n))\xi'_{Q^n}\} = \eta\xi'_{Q^n}, \quad (9)$$

$$\vartheta'' = \{x_1/s_1, \dots, x_n/s_n\} = \{x_1/T_{\mathcal{G}_\eta}(v(x_1))\xi''_{Q^1}, \dots, x_n/T_{\mathcal{G}_\eta}(v(x_n))\xi''_{Q^n}\} = \eta\xi''_{Q^n}. \quad (10)$$

Это означает, что подстановка η , вычисляемая алгоритмом MST, является нижней гранью подстановок ϑ' и ϑ'' .

Предположим, что подстановка η , полученная в результате применения алгоритма MST, не является точной нижней гранью. Тогда существует подстановка $\pi = \{y_j/t\}$

такая, что $t \neq y'$, $y' \notin \text{Var}(\eta)$ (то есть $(\eta\pi)^\sim \neq \eta^\sim$), для которой выполняются неравенства $\eta\pi \sqsubseteq \vartheta'$, $\eta\pi \sqsubseteq \vartheta''$. Учитывая равенства $\vartheta' = \eta\xi'_Q$ и $\vartheta'' = \eta\xi''_Q$, мы получаем соотношения $\pi \sqsubseteq \xi'_Q$, $\pi \sqsubseteq \xi''_Q$. На основании определения подстановок ξ'_Q и ξ''_Q , это означает, что в множестве $\mathcal{Q}$, которое сформировалось по завершении работы алгоритма MST, содержится такая тройка (y_j, v', v'') , для которой $T(v') = t\xi'$ и $T(v'') = t\xi''$ для некоторых ξ' и ξ'' . Однако последнее противоречит тому, что по завершении работы алгоритма условие в точке L2 не выполняется и в $\mathcal{Q}$ нет различных троек (y_i, v'_i, v''_i) и (y_j, v'_j, v''_j) таких, что $v'_i = v'_j$ и $v''_i = v''_j$. Следовательно, η — точная нижняя грань подстановок ϑ' и ϑ'' .

Чтобы оценить сложность алгоритма MST, достаточно заметить, что граф $\mathcal{G}_\eta$ имеет столько вершин, сколько элементов (троек) (y, v', v'') , $v' \in \mathcal{G}_{\vartheta'}$, $v'' \in \mathcal{G}_{\vartheta''}$ содержится в множестве $\mathcal{Q} \cup \mathcal{Q}$ в конце работы алгоритма, и на каждой итерации основного цикла в это множество заносится в точности одна новая тройка. Следовательно, число итераций основного цикла в точности равно размеру подстановки $\eta = \vartheta' \downarrow \vartheta''$. На каждой итерации проводится поиск нужного элемента среди троек множества $\mathcal{Q} \cup \mathcal{Q}$. Мощность этого множества не превосходит величины $rN(\vartheta' \downarrow \vartheta'')$, где r — наибольшая валентность функциональных символов алфавита $\mathcal{F}$. Отсюда получается требуемая оценка сложности. Теорема доказана.

6. Максимальный размер наименее общих шаблонов

Остается выяснить, каким может быть размер точной нижней грани. Из описания алгоритма видно, что каждая вершина графа $\mathcal{G}_\eta$ взаимно однозначно соответствует некоторой паре вершин графов $\mathcal{G}_{\vartheta'}$, $\mathcal{G}_{\vartheta''}$. Поэтому справедлива верхняя оценка $N(\vartheta' \downarrow \vartheta'') \leq N(\vartheta')N(\vartheta'')$. Нижнюю оценку размера точной нижней грани дает следующая теорема.

Теорема 4. Если в алфавите $\mathcal{F}$ найдется хотя бы один функциональный символ валентности m , $m > 1$, то существует такая последовательность пар подстановок $(\vartheta'_1, \vartheta''_1), (\vartheta'_2, \vartheta''_2), \dots, (\vartheta'_n, \vartheta''_n), \dots$, что для любого $n > 0$ справедливо неравенство $(1/6)N(\vartheta'_n)N(\vartheta''_n) \leq N(\vartheta'_n \downarrow \vartheta''_n)$.

Доказательство. Не ограничивая общности, будем считать, что в алфавите $\mathcal{F}$ есть двухвалентный функциональный символ f .

Пусть $\mathcal{Y}' = \{y'_1, y'_2, \dots\}$ и $\mathcal{Y}'' = \{y''_1, y''_2, \dots\}$ — два непересекающихся бесконечных множества переменных. Введем в рассмотрение две последовательности термов $t'_1, t'_2, \dots$ и $t''_1, t''_2, \dots$, устроенных следующим образом:

$$\begin{aligned} t'_1(y'_1, y'_2) &= f(f(y'_1, y'_2), f(y'_1, y'_2)), \\ t'_{n+1}(y'_1, \dots, y'_{2n+1}) &= t'_1(t'_n(y'_1, \dots, y'_{2n}), t'_n(y'_{2n+1}, \dots, y'_{4n+1})), \quad n \geq 1; \\ t''_1(y''_1, y''_2) &= f(f(y''_1, y''_2), f(y''_2, y''_1)), \\ t''_{n+1}(y''_1, \dots, y''_{2n+1}) &= t''_1(t''_n(y''_1, \dots, y''_{2n}), t''_n(y''_{2n+1}, \dots, y''_{4n+1})), \quad n \geq 1. \end{aligned}$$

Графические представления термов t'_1, t'_2, t''_1, t''_2 этих последовательностей изображены на рис. 2. Для большей наглядности термы t'_i представлены нередуцированными графами.

Искомые подстановки $\vartheta'_n, \vartheta''_n$ определяются так:

$$\vartheta'_n = \{x/t'_n(y'_1, \dots, y'_{2n})\}, \quad \vartheta''_n = \{x/t''_n(y''_1, \dots, y''_{2n})\}.$$

Очевидно, что

$$N(\vartheta'_1) = 4, \quad N(\vartheta'_{n+1}) = 2 + 2N(\vartheta'_n)$$

Рис. 2. Термы $t'_i, t''_i, i = 1, 2$

и

$$N(\vartheta'_1) = 5,$$

$$N(\vartheta''_{n+1}) = 3 + 2N(\vartheta'_n).$$

Отсюда следует, что

$$N(\vartheta'_n) = 3 \cdot 2^n - 2,$$

$$N(\vartheta''_n) = 4 \cdot 2^n - 3.$$

Далее рассмотрим последовательность термов $t_1, t_2, \dots$, которые устроены следующим образом:

$$\begin{aligned} t_1(y_1, y_2, y_3, y_4) &= f(f(y_1, y_2), f(y_3, y_4)), \\ &\dots \\ t_{n+1}(y_1, \dots, y_{4^{n+1}}) &= f(f(t_n(y_1, \dots, y_{4^n}), t_n(y_{4^n+1}, \dots, y_{2 \cdot 4^n})), \\ &\quad f(t_n(y_{2 \cdot 4^n}, \dots, y_{3 \cdot 4^n}), t_n(y_{3 \cdot 4^n+1}, \dots, y_{4^{n+1}}))). \end{aligned}$$

АОГ, представляющий каждый из термов t_n , — это полное двоичное дерево глубины $2(n+1)$. Графические представления термов t_1, t_2 этой последовательности изображены на рис. 3. Этой последовательности термов отвечает последовательность подстановок $\eta_1, \eta_2, \dots$, где $\eta_n = \{x/t_n(y_1, \dots, y_{4^n})\}$. Очевидно, $N(\eta_n) = 2 \cdot 4^n - 1$, и следовательно, выполняется неравенство $(1/6)N(\vartheta'_n)N(\vartheta''_n) \leq N(\eta_n)$.

Нетрудно заметить также, что каждая подстановка η_n является нижней гранью подстановок ϑ'_n и ϑ''_n . Остается убедиться в том, что η_n — это точная нижняя грань ϑ'_n и ϑ''_n . Для этого воспользуемся индукцией по n . Очевидно, $\vartheta'_1 \downarrow \vartheta''_1 = \eta_1$. Учитывая строение

Рис. 3. Термы t_1, t_2 .

подстановок ϑ'_{n+1} и ϑ''_{n+1} и закон дистрибутивности операции композиции подстановок относительно взятия точной нижней грани, можно заметить, что

$$\vartheta'_{n+1} \downarrow \vartheta''_{n+1} = \{x/f(f(x_1, x_2), f(x_3, x_4))\}(\rho' \downarrow \rho''),$$

где

$$\begin{aligned} \rho' &= \{x_1/t'_n(y'_1, \dots, y'_{2^n}), x_2/t'_n(y'_{2^n+1}, \dots, y'_{2^{n+1}}), \\ &\quad x_3/t'_n(y'_1, \dots, y'_{2^n}), x_4/t'_n(y'_{2^n+1}, \dots, y'_{2^{n+1}})\}, \\ \rho'' &= \{x_1/t''_n(y''_1, \dots, y''_{2^n}), x_2/t''_n(y''_{2^n+1}, \dots, y''_{2^{n+1}}), \\ &\quad x_3/t''_n(y''_{2^n+1}, \dots, y''_{2^{n+1}}), x_4/t''_n(y''_1, \dots, y''_{2^n})\}. \end{aligned}$$

Термы, фигурирующие в подстановках ρ' и ρ'' , позволяют воспользоваться индуктивной гипотезой для вычисления точной нижней грани этих подстановок. В результате мы получаем, что

$$\begin{aligned} \rho' \downarrow \rho'' &= \{x_1/t_n(y_1, \dots, y_{4^n}), x_2/t_n(y_{4^n+1}, \dots, y_{2 \cdot 4^n}), \\ &\quad x_3/t_n(y_{2 \cdot 4^n}, \dots, y_{3 \cdot 4^n}), x_4/t_n(y_{3 \cdot 4^n+1}, \dots, y_{4^{n+1}})\}. \end{aligned}$$

Здесь следует обратить особое внимание на то, что все четыре множества переменных, от которых зависят термы, подставляемые вместо x_1, x_2, x_3, x_4 , попарно не пересекаются. Это обусловлено тем, что в подстановках ρ' и ρ'' не повторяется ни одна пара переменных (y'_i, y''_j) , стоящих на соответствующих позициях в термах $t'_n(y'_1, \dots, y'_{2^n}), t'_n(y'_{2^n+1}, \dots, y'_{2^{n+1}}), t'_n(y''_1, \dots, y''_{2^n}), t'_n(y''_{2^n+1}, \dots, y'_{2^{n+1}}), t''_n(y''_1, \dots, y''_{2^n}), t''_n(y''_{2^n+1}, \dots, y''_{2^{n+1}}), t''_n(y''_{2^n+1}, \dots, y''_{2^{n+1}}), t''_n(y''_1, \dots, y''_{2^n})$. Принимая во внимание замечание о строении АОГ, представляющего совместно термы с непересекающимися множествами свободных переменных, приходим к выводу о том, что АОГ, задающий подстановку ρ , состоит из четырех деревьев глубины $2n$. Таким образом АОГ, задающий подстановку $\eta_n = \vartheta'_n \downarrow \vartheta''_n$, является деревом глубины $2n + 2$. Доказательство теоремы завершено.

7. Заключение

На основании теорем, представленных выше, можно утверждать, что сложность задачи вычисления наименее общего шаблона термов, представленных ациклическими ориентированными упорядоченными графами, оценивается сверху величиной $\Theta(n^2 \log n)$, а снизу величиной $\Theta(n^2)$, где n — суммарный размер графов, задающих исходные термы. Этот

алгоритм можно использовать для вычисления инвариантов последовательных императивных программ. В дальнейшем мы планируем исследовать задачу антиунификации в алгебрах, операции которых удовлетворяют законам коммутативности, ассоциативности, идемпотентности и другим законам.

Список литературы

1. Apt K. R., Olderog E.-R., *Verification of sequential and concurrent Programs*. Springer, Berlin, 1997.
2. Baxter L. D., *An efficient unification algorithm*, Techn. Rep. CS-73-23, Dept. Anal. Comp. Sci. Univ. Waterloo, Ontario, Canada, 1973.
3. Годлевский А. Б., Кривой С. Л., О проектировании эффективных алгоритмов приведения автоматов для некоторых отношений эквивалентности. *Кибернетика* (1989) №6, 54–61.
4. Delcher A. L., Kasif S., Efficient parallel term matching and anti-unification. *J. Autom. Reasoning* (1992) **9**, №3, 391–406.
5. Дегтярев А. И., *Об использовании аксиом функциональной рефлексивности в (P & R) процедуре опровержения*, Препринт Ин-та Кибернетики АН УССР, №75-28, 1975.
6. Eder E., Properties of substitutions and unifications. *J. Symb. Comput.* (1985) **1**, 31–46.
7. Костылев Е. В., Захаров В. А., Быстрые алгоритмы антиунификации и их применение при анализе программ. В сб.: *Материалы 13 Международной школы-семинара «Синтез и сложность управляющих систем»*, **1**, Пенза, 2002, с. 76–81.
8. Kuper G. M., McAloon K. W., Palem K. V., Perry K. J., A note on the parallel complexity of anti-unification. *J. Autom. Reasoning* (1992) **9**, №3, 381–389.
9. Lassez J. I., Maher M. J., Marriot K., Unification revisited. *Lect. Notes Comput. Sci.* (1988) **306**, 67–113.
10. Martelli A., Montanari U., An efficient unification algorithm. *ACM Trans. Program. Lang. Syst.* (1982) **4**, №2, 258–282.
11. Nielson F., Nielson H., Hankin C., *Principles of program analysis*. Springer, Berlin, 1999.
12. Oancea C. E., So C., Watt S. M., Generalization in Maple. *Maple Conf.* (2005) 277–382.
13. Palamidessi C., Algebraic properties of idempotent substitutions. *Lect. Notes Comput. Sci.* (1990) **443**, 386–399.
14. Paterson M. S., Wegman M. N., Linear unification. *J. Comput. Syst. Sci.* (1978) **16**, №2, 158–167.
15. Plotkin G. D., A note on inductive generalization. *Machine Intelligence* (1970) **5**, №1, 153–163.
16. Reynolds J. C., Transformational systems and the algebraic structure of atomic formulas. *Machine Intelligence* (1970) **5**, №1, 135–151.
17. Robinson J. A., A machine-oriented logic based on the resolution principle. *J. ACM* (1965) **12**, №1, 23–41.
18. Сабельфельд В. К., Полиномиальная оценка сложности распознавания логико-термальной эквивалентности. *Докл. АН СССР* (1979) **249**, №4, 793–796.
19. Sorensen M. H., Gluck R., An algorithm of generalization in positive supercompilation. In: *Proc. 1995 Intern. Symp. Logic Programming*. MIT Press, 1995, pp. 465–479.
20. Watt S. M., Algebraic generalization. *ACM SIGSAM Bull.* (2005) **39**, №3, 93–94.
21. Zakharov V. A., On the refinement of logic programs by means of anti-unification. In: *Proc. 2nd Panhellenic Logic Symp*, Delphi, Greece, 1999, pp. 219–224.

Статья поступила 14.06.2007.