Модел. и анализ информ. систем. Т. **21**, № **4** (2014) 104–115

УДК 681.3.06

Разрешимость эквивалентности в двухпараметрических перегородчатых моделях программ

Молчанов А. Э.

Московский государственный университет им. М.В. Ломоносова, 119991, Российская Федерация, Москва, Ленинские горы, д. 1

e-mail: gurux13@gmail.com

получена 31 августа 2014

Ключевые слова: формализация программы, схема программы, эквивалентность схем программ, перегородчатая модель программ

Алгебраические модели программ с процедурами предназначены для изучения семантических свойств самих программ на их образах – схемах программ. Для моделей программ с процедурами, в которых оба параметра могут быть выбраны независимо, формулируются проблемы процедурной либеризации и эквивалентности. Исследуются модели программ с процедурами, строящиеся по данной модели программ без процедур. Приводятся алгоритмы решения обеих задач для таких моделей при дополнительном ограничении на разрешимость проблемы непустоты пересечения в модели программ без процедур. Показана полиномиальная сложность этих алгоритмов. В заключение предложены задачи для дальнейшего исследования.

1. Введение

В данной статье продолжены исследования перегородчатых моделей программ, проводившиеся в [2] и в [1]

Основная тематика статей – исследование проблематики для алгебраических моделей программ с процедурами, нацеленной на исследование семантических свойств моделируемых программ.

Алгебраические модели программ с процедурами введены для программ, использующих в своей работе аппарат процедур. Они явились естественным обобщением моделей программ без процедур, широко применяемых в теории схем программ. Объекты этих моделей называются схемами программ.

В проблематике теории введённых моделей центральное место отводится проблеме эквивалентности схем, принадлежащих модели, и поэтому встал вопрос о выделении класса моделей, подающих надежду на разрешимость в них данной проблемы. В [1] в качестве выделяемых предложены так называемые перегородчатые модели. Интерес к ним основан на следующем: отдельная перегородчатая модель по

определению индуцируется некоторой моделью программ без процедур, и та является её подмоделью. Для моделей программ без процедур проблема эквивалентности достаточно хорошо изучена.

Для вводимых таким образом моделей в первую очередь рассмотрена проблема либеризации в модели, заключающаяся в поиске алгоритма, который, получив на свой вход схему из модели, строит эквивалентную ей свободную схему (так называется схема, все элементы которой участвуют в её функционировании). Дело здесь в том, что в подавляющем числе случаев разрешимости проблемы эквивалентности в модели предварительно устанавливалась разрешимость в ней проблемы либеризации.

Каждая алгебраическая модель программ определяется двумя параметрами. В [1], [2] рассматривались так называемые однопараметрические модели. В таких моделях из двух параметров произвольно выбран может быть только один, а второй определяется им однозначно. В приведенных доказательствах указанных статей это ограничение является существенным.

В настоящей статье это ограничение снято: модели предполагаются двухпараметрическими.

Несмотря на сделанное замечание о решении проблемы свободы перед решением проблемы эквивалентности, для решения проблемы эквивалентности не потребовалось решать проблему свободы. Она заменена проблемой процедурной свободы. Эта проблема заключается в поиске алгоритма, переводящего схему в эквивалентную ей, в которой все процедуры участвуют в функционировании.

Не удалось решить проблему эквивалентности в двухпараметрической модели программ без дополнительных ограничений, хотя введенное дополнительное ограничение нельзя назвать серьезным: требуется разрешимость проблемы непустоты пересечения в индуцирующей простой модели программ. Проблема непустоты пересечения может быть сформулирована так: найти алгоритм, который для двух схем определяет, есть ли какие-нибудь входные данные, на которых обе схемы закончат выполнение.

Отметим, что во всех рассматривавшихся простых моделях программ при разрешимости проблемы эквивалентности проблема непустоты пересечения разрешима алгоритмом, практически копирующим алгоритм проверки эквивалентности.

Подробное рассмотрение проблемы непустоты пересечения выходит за рамки данной статьи.

Здесь доказана теорема 1 о полиномиальной разрешимости проблемы процедурной либеризации в перегородчатой модели. Далее показано, что при упомянутых ограничениях в двухпараметрической модели программ разрешима проблема эквивалентности, причем разрешающий алгоритм полиномиален.

Нами воспроизведены определения всех рассматриваемых здесь моделей программ, дабы не отсылать читателя к статье [2].

2. Рассматриваемые модели программ

Здесь даются определения общего вида матричной модели программ с процедурами и её частного вида – перегородчатой модели.

Матричные модели программ с процедурами строятся над четырьмя конечными и непересекающимися алфавитами Y, C, R и P. Элементы первых трех алфавитов называются символами операторов, вызовов и возвратов соответственно, элементы алфавита P именуются логическими переменными, каждая из которых принимает значение из множества $\{0, 1\}$. Алфавиты Y, C, R, P называются *базисом*.

Элементы матричной модели называются схемами программ. Определению схемы предпошлём введение множества X, где

$$X = \{x | x : P \to \{0, 1\}\} :$$

его элементы называются *наборами*.

Схема программы по своей структуре представляет размеченный конечный ориентированный граф следующего строения. Граф распадается на подграфы с непересекающимися множествами вершин. Один из подграфов называется *главным*, остальные – *процедурными*. В главном подграфе выделены вершина *вход* без входящих в неё дуг и вершина *выход* без исходящих из неё дуг. В любом процедурном подграфе тоже выделены две вершины – *инициальная* и *финальная*. В каждом подграфе имеется специальная вершина *loop* без исходящих из неё дуг. Перечисленные вершины не имеют меток, а все остальные помечены символами из Y, C и R, называясь соответственно *преобразователями*, *вызовами* и *возвратами*. Всякому вызову соответствует свой персональный возврат, именуемый *парным вызову*; тот и другой принадлежат общему для них подграфу. Все вызовы перенумерованы. Из всякой вершины, отличной от выхода схемы, вызова, финальной и вершины *loop*, исходят дуги, каждая из которых помечена своим набором из X, в количестве, равном числу наборов в X. Из вызова исходит единственная дуга, которая ведёт в инициальную вершину некоторого подграфа (он называется ассоциированным с данным вызовом), и тогда из финальной вершины этого подграфа идёт дуга в парный вызову возврат, и это – единственная приходящая в него дуга. В инициальную вершину могут входить дуги только из вызовов, а из финальной вершины могут исходить дуги только в возвраты. Так осуществляется связь между подграфами, ибо начало и конец дуги иного типа находятся в общем для них подграфе.

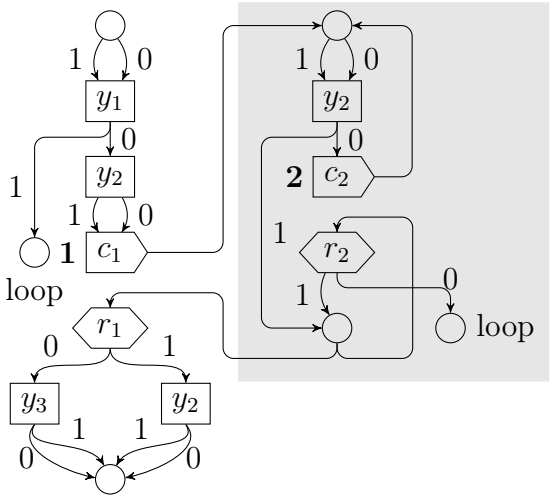

Рис. 1: Пример схемы программы

На рис. 1 приведен пример матричной схемы программы. Здесь $\{y_1, y_2, y_3\} \subseteq Y, \{c_1, c_2\} \subseteq C, \{r_1, r_2\} \subseteq R, P = \{p\}$. Схема содержит один процедурный подграф.

Функционирование схемы осуществляется на функциях разметки. *Функцией разметки* называется отображение множества всех слов, построенных над $Y \cup C \cup R$ (оно обозначается H), в множество X. Множество всех функций разметки обозначается $\mathfrak{L}$.

Пусть $\mu \in \mathfrak{L}$. *Выполнением схемы на функции* μ называется процесс, заключающийся в путешествии по схеме и сопровождающийся построением слова из H; это слово называется *цепочкой*.

Для однозначности выбора пути, кроме μ, используется магазин, в который загружаются номера вызовов в схеме. Путешествие начинается по дуге, исходящей из входа, при пустых магазине и цепочке. Переход через вершину с сопоставленным ей символом сопровождается приписыванием этого символа к текущей цепочке справа. Если переходимая вершина – вызов, то в магазин загружается его номер. При переходе через инициальную вершину и финальную вершину текущая цепочка не изменяется. Во втором случае из макушки магазина, который заведомо не пуст, извлекается номер, и путешествие продолжается по дуге, ведущей к возврату, парному вызову с этим номером. Из входа схемы путь идёт по дуге, помеченной набором $\mu(\Lambda)$, где Λ – пустая цепочка. При переходе через преобразователь, возврат, инициальную вершину тоже используется функция μ: в качестве следующей берется дуга, несущая метку $\mu(h)$, где h – построенная цепочка. При достижении выхода путешествие прекращается; говорим, что *схема остановилась на* μ, и построенную цепочку называем *результатом её выполнения на* μ. Альтернативным является случай бесконечного путешествия по схеме или случай попадания в вершину *loop*, когда путешествие прекращается без результата.

Схему из матричной модели, не содержащую процедурных подграфов, назовём *простой*. Также *простой* именуем матричную модель, построенную над базисом, в котором C и R пусты.

Пусть ν – эквивалентность в множестве H, и $L \subseteq \mathfrak{L}$. Схемы G_1, G_2 назовем (ν, L)-*эквивалентными* (обозначим: $G_1 \overset{\nu,L}{\sim} G_2$) тогда и только тогда, когда, какой бы ни была функция μ из L, всякий раз, как на ней останавливается одна из схем, останавливается и другая, и в этом случае результатами их выполнения являются ν-эквивалентные цепочки.

Множество всех схем над выбранным базисом, рассматриваемое вместе с (ν, L)-эквивалентностью схем, называется *матричной* (ν, L)-*моделью*, а ν, L – её *параметрами*.

Перегородчатая модель программ с процедурами представляет собой тот частный случай матричной модели над базисом Y, C, R, P, в которой параметры ν, L индуцируются параметрами τ, l некоторой простой модели над Y, P; последняя называется *индуцирующей первую*.

Эквивалентность ν вводится следующим образом. Цепочки h_1 и h_2 из H считаем ν-эквивалентными в том и только том случае, когда совпадают их проекции на множество $C \cup R$, и, кроме того, если представить цепочки h_i, $i = 1, 2$, в виде

$$h_i = h_{0i} b_1 h_{1i} b_2 \dots b_k h_{ki}, \quad k \geq 0,$$

где $b_1 b_2 \dots b_k$ есть общая проекция цепочек h_1, h_2 на множество $C \cup R$, то при всех

$j, j = 0, \ldots, k$, имеет место соотношение

$$h_{j1} \overset{\tau}{\sim} h_{j2}.$$

Здесь τ – эквивалентность цепочек g_1, g_2 над алфавитом Y записывается в виде $g_1 \overset{\tau}{\sim} g_2$.

Опишем, как строится множество L функций разметки.

Начнём с того, что задание отдельной функции разметки из $\mathfrak{L}$ фактически сводится к разметке наборами из X вершин бесконечного дерева, в котором из всякой вершины исходят дуги в количестве, равном общему числу символов в алфавитах Y, C, R, причём каждая дуга помечена своим символом. Таким образом, всякой вершине дерева соответствует цепочка, полученная выписыванием друг за другом символов, метящих дуги пути, идущего из корня дерева в данную вершину. Набор из X, сопоставленный этой вершине, воспринимается как значение функции разметки именно на этой цепочке, а разметка всех вершин дерева наборами из X определяет функцию разметки.

Заметим теперь, что всякая вершина дерева является корнем его поддерева, все дуги которого помечены только символами из Y и всеми такими символами. Выделим поддеревья описанного типа, вырастающие из корня дерева или из вершины его, в которую ведёт дуга с меткой из C или R. Всякое выделенное поддерево разметим наборами из X так, чтобы это определило функцию разметки из l. По определению, выполненная разметка даёт функцию разметки из L, и иных функций в L нет.

Итак, перегородчатая модель программ с параметрами ν, L построена.

В дальнейшем базис, над которым строятся перегородчатые модели, считаем фиксированным и поэтому его не упоминаем, а функции разметки из L называем просто *допустимыми для модели с этим параметром.*

3. Проблема процедурной либеризации в перегородчатой модели программ

Обозначим M рассматриваемую нами перегородчатую модель программ и определим проблему процедурной либеризации в M.

Пусть G – схема из M. *Маршрутом* в ней назовём ориентированный путь, начинающийся в её входе, именуя его *маршрутом через схему*, если он оканчивается в выходе схемы.

Говорим, что маршрут в схеме *реализуем*, если существует допустимая для M функция разметки, которая при выполнении на ней схемы прокладывает в ней путь, начинающийся этим маршрутом.

По определению, схема G называется *процедурно-свободной в* M, если всякая её вершина типа вызов, возврат, инициальная, финальная, находится на некотором реализуемом маршруте через G. *Проблема процедурной либеризации в* M состоит в поиске алгоритма, который для любой поступившей на его вход схемы из M строит эквивалентную ей процедурно-свободную схему. Если такой алгоритм найден, проблема называется *разрешимой.*

Схема G называется *пустой*, если на любой допустимой функции разметки результат выполнения схемы не определен. *Проблема пустоты* состоит в поиске ал-

горитма, который для поступившей на вход схемы G из некоторой модели M определяет, является ли она пустой. Если такой алгоритм найден, проблема разрешима.

Повторим определения, данные в статье [1], которые потребуются для дальнейшего изложения.

Пусть G – схема из рассматриваемой модели. *Опорной* в ней назовём вершину, имеющую тип: вход, выход, вызов, возврат. *Преемником опорной вершины* v_1 назовём такую опорную вершину v_2, в которой завершается ориентированный путь из v_1, не содержащий внутри опорных вершин. Сам путь именуем *е-маршрутом* из v_1 в v_2.

Пусть v_1 – опорная вершина в схеме из M и v_2 – её преемник. Обозначим $G(v_1, v_2)$ простую схему, построенную по следующим правилам: в ней оставляются все вершины, принадлежащие е-маршрутам в рассматриваемой схеме из v_1 в v_2, и если v_1 – это возврат, то он заменяется входом создаваемой схемы, если v_2 – это вызов, то её выходом; при этом инициальную вершину считаем входом схемы, финальную – её выходом. Схему $G(v_1, v_2)$ назовем вырастающей из v_1.

Основной в данном разделе является теорема 1.

Теорема 1. *Проблема процедурной либеризации в M разрешима, если разрешима проблема пустоты в индуцирующей модели программ.*

Доказательство.

Отметим, что любая простая схема процедурно свободна. Следовательно, можно считать, что на вход алгоритма подается схема, не являющаяся простой.

Назовем алгоритм, разрешающий проблему процедурной либеризации, $pfree$. Этот алгоритм строится на основе алгоритма ρ_1 из [1], разрешающего проблему *либеризации* схем программ с процедурами. Опуская определение этой проблемы, приведем выполняемые алгоритмом ρ_1 действия.

Алгоритм ρ_1 работает в два этапа.

На первом для испытуемой схемы G строятся все последовательности её опорных вершин (без повторения вершин в последовательности), являющиеся проекциями на них реализуемых маршрутов через схему. Если

$$v_1 v_2 \dots v_k, k \geq 2 \tag{1}$$

– такая последовательность, то для всех $i, i = 1, \dots, k-1$, схема $G(v_i, v_{i+1})$ должна быть непустой.

На втором этапе алгоритм ρ_1 удаляет те опорные вершины схемы G, которые не вошли ни в одну последовательность вида (1), вместе с вырастающими из них простыми схемами.

Алгоритм $pfree$ отличается от ρ_1 только тем, что ему приходится проверять на непустоту схемы типа $G(v_1, v_2)$ специальным алгоритмом, тогда как алгоритму ρ_1 было достаточно убедиться в существовании е-маршрута из v_i в v_{i+1}.

Исходя из этого, можно привести оценку сложности алгоритма $pfree$.

Операция построения простой схемы $G(v_1, v_2)$ для выбранных v_1, v_2 оценивается как $O(n \log n)$, где n – число вершин в схеме. Такая операция производится в худшем случае для всех пар вершин, которых $O(n^2)$, то есть общая сложность построения простых схем $G(v_1, v_2)$ оценивается как $O(n^3 \log n)$.

На каждой простой схеме $G(v_1, v_2)$, которых $O(n^2)$, применяется алгоритм проверки пустоты. В предположении о том, что сложность алгоритма проверки пустоты схемы размера k равна $f(k)$, общая сложность проверки пустоты составляет $O(n^2 \cdot f(n))$.

Операции, проводимые алгоритмом дополнительно, совпадают с операциями, которые проводит алгоритм ρ_1. Следовательно, их суммарная сложность не превышает сложности алгоритма ρ_1, которая равна $O(n^3 \log n)$.

Следовательно, сложность алгоритма составляет

$$O(n^3 \log n + n^2 \cdot f(n)),$$

где $f(n)$ – сложность проверки пустоты простых схем.

4. Проблема эквивалентности

Задача данного раздела – исследовать проблему эквивалентности в двухпараметрической перегородчатой модели программ. Она по-прежнему обозначается M. Обозначим M_0 индуцирующую простую модель программ.

Основной результат устанавливается теоремой 2.

Теорема 2. *Если в простой модели M_0, индуцирующей перегородчатую, разрешимы проблемы эквивалентности и непустоты пересечения, то проблема эквивалентности разрешима в индуцируемой перегородчатой модели M, причем разрешающий алгоритм полиномиален как алгоритм с двумя оракулами.*

Проблема непустоты пересечения заключается в поиске алгоритма, который для пары схем определяет, существует ли допустимая функция разметки, на которой обе схемы останавливаются.

Заметим, что если в модели программ разрешима проблема эквивалентности, то в ней разрешима и проблема пустоты (путем проверки эквивалентности схемы пустой схеме). Следовательно, применима теорема 1, и можно проверять эквивалентность только процедурно-свободных схем из M.

Доказательству теоремы предпошлём следующие понятия.

Трасса строится по маршруту в схеме, оканчивающемуся в опорной вершине, и представляет собой проекцию этого маршрута на опорные вершины.

Маршруты схем G_1, G_2 называются *сочетаемыми*, если они прокладываются общей для них функцией разметки.

Трассы называются *сочетаемыми*, если они построены по сочетаемым маршрутам.

Опорные вершины *сочетаемые*, если они являются концами равновеликих сочетаемых трасс.

Отметим, что сочетаемые вершины эквивалентных схем – либо входы, либо выходы, либо одинаково помеченные вершины. Действительно, пусть две сочетаемые опорные вершины разнотипны или разнопомечены. Пусть они разнопомечены. Рассмотрим сочетаемые маршруты, по которым были построены трассы. В силу процедурной свободы схем, эти маршруты можно продолжить до выходов схем. Результаты вычислений содержат символы вершин на одинаковых местах в проекции на

$C \cup R$, и по определению эквивалентности должны совпадать. Очевидно, что два входа сочетаемы только друг с другом. Если же выход сочетаем с некоторой вершиной другого типа, то можно получить проекции на $C \cup R$ разной длины, что тоже невозможно.

Если w – преемник v в схеме $G \in M$, и схема $G(v, w)$ не пуста, то w называется *достижимым* преемником.

Пусть G – процедурно свободная схема из M, v – её опорная вершина. Обозначим $T(G, v)$ множество символов из $C \cup R$, метящих достижимые преемники v. Пусть $t \in T(G, v)$. Обозначим $G(v, t)$ простую схему, построенную по маршрутам из v, ведущим в опорные вершины с меткой t. При этом все вершины с меткой t склеиваются в одну с переносом на неё всех входивших дуг.

Рассмотрим процедурно-свободные схемы $G_1, G_2 \in M$ и сочетаемые опорные вершины $v_1 \in G_1, v_2 \in G_2$.

Полагаем, что модель M имеет параметры (ν, L), а индуцирующая модель – (τ, l).

Утверждение 1. *Схемы G_1, G_2 не эквивалентны, если существует символ t, который принадлежит одному из множеств $T(G_1, v_1), T(G_2, v_2)$ и не принадлежит другому.*

Доказательство. Действительно, если $t \in T(G_1, v_1)$ и $t \notin T(G_2, v_2)$, где v_1, v_2 – сочетаемые опорные вершины, то можно построить функцию разметки, на которой обе схемы останавливаются, и в схеме G_1 следующей за v_1 проходимой опорной вершиной является вершина с меткой t. Для v_2 такого преемника нет, и это приводит к несовпадению проекций на $C \cup R$.

Утверждение 2. *В эквивалентных процедурно-свободных схемах G_1, G_2 простые схемы $G_1(v_1, t), G_2(v_2, t) \in M_0$, где v_1, v_2 – сочетаемые опорные вершины, а t – символ из $T(G_1, v_1) = T(G_2, v_2)$, являются эквивалентными.*

Доказательство. Пусть существуют две неэквивалентные простые схемы $G_1(v_1, t), G_2(v_2, t_2)$. Рассмотрим функцию разметки, которая прокладывает маршруты, подтверждающие согласованность v_1 и v_2. После прохода v_1, v_2, согласно определению перегородчатой модели, можно выбрать новую функцию разметки из l. Выберем ту, которая подтверждает неэквивалентность $G_1(v_1, t), G_2(v_2, t_2)$. Так как хотя бы одна из простых схем остановится, можно продолжить функцию разметки так, чтобы хотя бы одна из схем G_1, G_2 остановилась. При этом, очевидно, результаты выполнения схем не будут эквивалентными.

Перейдем к доказательству теоремы 2.

Алгоритм, проверяющий эквивалентность процедурно-свободных схем G_1, G_2, строит конечный ориентированный граф $D(G_1, G_2)$, вершинами которого являются пары сочетаемых опорных вершин.

Пару вершин (v_1, v_2) назовем заключительной, если хотя бы одна из вершин – выход схемы.

Сначала в $D(G_1, G_2)$ вносится пара (вход схемы G_1, вход схемы G_2). Пусть (v_1, v_2) – пара из $D(G_1, G_2)$, не являющаяся заключительной. Алгоритм исследует, можно ли продолжить трассы, ведущие в v_1, v_2, на единицу длины, сохранив их сочетаемость.

Для этого сначала он просматривает $T(G_1, v_1), T(G_2, v_2)$. Если выполнено утверждение 1, алгоритм останавливается с сообщением о неэквивалентности. В противном случае для каждого символа из $T(G_1, v_1) = T(G_2, v_2)$ он проверяет утверждение 2. Если проверка не выявляет неэквивалентности, то для пары (v_1, v_2) строятся пары вида (u_1, u_2), где u_1, u_2 – преемники вершин v_1, v_2, имеющие общую метку, и такие, что $G(v_1, u_1), G(v_2, u_2)$ в пересечении не пусты. Пара (u_1, u_2) – новая пара сочетаемых вершин, которая добавляется в $D(G_1, G_2)$, если её там не было.

Очевидно, что процесс построения $D(G_1, G_2)$ завершаем.

Если в процессе построения этого графа не была установлена неэквивалентность схем, схемы эквивалентны.

Покажем, что это действительно так.

Пусть $G_1 \nsim G_2$. Существует функция разметки из L, которая подтверждает их неэквивалентность. На этой функции разметки результаты выполнения схем либо отличаются в проекции на $C \cup R$, либо есть участок между символами этой проекции, на котором неэквивалентны слова из Y^*. В первом случае, согласно определению, существует пара разнопомеченных сочетаемых вершин. Во втором случае существует пара неэквивалентных схем вида $G_1(v_1, t), G_2(v_2, t)$, где v_1, v_2 – сочетаемые опорные вершины. Следовательно, неэквивалентность G_1, G_2 будет установлена.

Очевидно, что сформулированные необходимые условия являются алгоритмически проверяемыми. Покажем, что сложность проверки всех необходимых условий полиномиальна, как алгоритм с двумя оракулами, решающими проблемы непустоты пересечения и эквивалентности в индуцирующей простой модели.

Пусть общее число вызовов (а значит, и возвратов) схем равно k. Пусть также число вершин в подграфе с наибольшим числом вершин равно s.

Алгоритм последовательно обходит графы программ, выявляя сочетаемые вершины. Всего таких пар может быть не больше k^2. Для каждой пары строятся маршруты в опорные вершины того же подграфа и проверяется непустота получаемых схем. Такая операция занимает $O(s^2 + f(s))$ времени, где $f(s)$ – сложность проверки пустоты простой схемы размера s. Эта операция выполняется в худшем случае для всех пар опорных вершин, то есть, не более k^2 раз. Затем для всех простых схем проверяется непустота пересечения, что занимает $O(k^4 g(s))$, где $g(s)$ – сложность проверки непустоты пересечения простых схема размера s.

Далее, для каждой построенной простой схемы из G_1 проверяется эквивалентность простой схеме из G_2. Сложность этой операции – $O(k^4 e(s))$, где $e(s)$ – сложность проверки эквивалентности простых схем.

В итоге, общая сложность алгоритма составляет

$$O(k^4(g(s) + e(s)) + k^2 s^2).$$

5. Заключение

В заключение предложим задачи, которые планируется решать в дальнейшем.

Во-первых, недостаточно исследована проблема непустоты пересечения в простых моделях программ. Поскольку от решения этой проблемы существенно зависят полученные результаты, проведение исследований оправдано.

Во-вторых, для моделей программ, в которых разрешима проблема эквивалентности, естественным образом возникает проблема *эквивалентных преобразований*. Эта проблема состоит в поиске такой системы преобразований, меняющих структуру схем, чтобы любую схему можно было перевести в эквивалентную ей с помощью конечной цепочки преобразований. В настоящее время проблема эквивалентных преобразований решена во многих простых моделях программ (например, [22], [23]), но нет ни одного результата для схем с процедурами. Поскольку система эквивалентных преобразований позволяет проводить структурный анализ эквивалентных схем, исследования в этом направлении также обоснованы.

Список литературы

1. Подловченко Р.И., Молчанов А.Э. О теории алгебраических моделей программ с процедурами // Моделирование и анализ информационных систем. 2012. Т. 19, №5. С. 100–114. [Podlovchenko R.I., Molchanov A.E. About Algebraic Program Models with Procedures // Modeling and Analysis of Information Systems. 2012. V. 19, No 5. P. 100–114 (in Russian)].

2. Подловченко Р.И., Молчанов А.Э. Разрешимость эквивалентности в перегородчатых моделях программ // Моделирование и анализ информационных систем. 2014. Т. 21, №2. С. 56–70. [Podlovchenko R.I., Molchanov A.E. Equivalence Problem Solvability in Gateway Program Models // Modeling and Analysis of Information Systems. 2014. V. 21, No 2. P. 56–70 (in Russian)].

3. Подловченко Р.И. К вопросу о полиномиальной сложности проблемы эквивалентности в алгебраических моделях программ // Кибернетика и системный анализ. Киев, 2012. №5. С. 17–24. [Podlovchenko R.I. K voprosu o polinomialnoj slozhnosti problemy ekvivalentnosti v algebraicheskih modeljah programm // Kibernetika i sistemnyj analiz. Kiev, 2012. N5. S. 17–24 (in Russian)].

4. Подловченко Р.И. Об одной методике распознавания эквивалентности в алгебраических моделях программ // Программирование. 2011. №6. С. 33–43. (Podlovchenko R.I. On an equivalence checking technique for algebraic models of programs // Programming and Computer Software. 2011. No 6. P. 292–298).

5. Подловченко Р.И. Об одном классе алгебраических моделей программ, представляющем практический интерес // Программирование. 2013. №3. С. 15–28. (Podlovchenko R.I. On a Class of Algebraic Models of Programs of Practical Interest // Programming and Computer Software. 2013. No 3. P. 124–134).

6. Ляпунов А.А. О логических схемах программ // Проблемы кибернетики. Вып. 1. М.: Физматгиз, 1958. С. 46–74 [Ljapunov A.A. O logicheskih shemah programm // Problemy kibernetiki. Vyp. 1. M.: Fizmatgiz, 1958. S. 46–74 (in Russian)].

7. Янов Ю.И. О логических схемах алгоритмов // Проблемы кибернетики. Вып. 1. М.: Физматгиз, 1958. С. 75–127 [Janov Ju.I. O logicheskih shemah algoritmov // Problemy kibernetiki. Vyp. 1. M.: Fizmatgiz, 1958. S. 75–127 (in Russian)].

8. Глушков В.М., Летичевский А.А. Теория дискретных преобразователей // Избранные вопросы алгебры и логики: сб. статей. Новосибирск: Наука, 1973. С. 5–39

(Glushkov V.M., Letichevskij A.A. Teorija diskretnyh preobrazovatelej // Izbrannye voprosy algebry i logiki: sb. statej. Novosibirsk: Nauka, 1973. S. 5–39 (in Russian)].

9. Ершов А.П., Сабельфельд В.К. Очерки схемной теории рекурсивных программ // Трансляция и модели программ. Новосибирск, 1980. С. 23–53 [Ershov A.P., Sabelfeld V.K. Ocherki shemnoj teorii rekursivnyh programm // Transljacija i modeli programm. Novosibirsk, 1980. S. 23–53 (in Russian)].

10. Захаров В.А. Быстрые алгоритмы разрешения эквивалентности операторных программ на уравновешенных шкалах // Математические вопросы кибернетики. Вып. 7. М.: Физматлит, 1998. С. 303–324 [Zaharov V.A. Bystrye algoritmy razreshenija ekvivalentnosti operatornyh programm na uravnoveshennyh shkalah // Matematicheskie voprosy kibernetiki. Vyp. 7. M.: Fizmatlit, 1998. S. 303–324 (in Russian)].

11. Захаров В.А. Проверка эквивалентности программ при помощи двухленточных автоматов // Кибернетика и системный анализ. 2010. N 4. С. 39–48 [Zaharov V.A. Proverka ekvivalentnosti programm pri pomoschi dvuhlentochnyh avtomatov // Kibernetika i sistemnyj analiz. 2010. N 4. S. 39–48 (in Russian)].

12. Котов В.Е., Сабельфельд В.К. Теория схем программ. М.: Наука, 1991. 348 с. [Kotov V.E., Sabelfeld V.K. Teorija shem programm. M.: Nauka, 1991. 348 p. (in Russian)].

13. Лисовик Л.П. Металинейные рекурсивные схемы над размеченными деревьями // Программирование. 1983. N 5. С. 13–22 [Lisovik L.P. Metalinejnye rekursivnye shemy nad razmechennymi derevjami // Programmirovanie. 1983. N 5. S. 13–22 (in Russian)].

14. Подловченко Р.И., Попов С.В. Аппроксимируемость одних моделей другими // Вестник Московского университета. Сер. 15: Вычислительная математика и кибернетика. 2001. N 2. С. 38–46 [Podlovchenko R.I., Popov S.V. Approksimiruemost odnih modelej drugimi // Vestnik Moskovskogo universiteta. Ser. 15: Vychislitelnaja matematika i kibernetika. 2001. N 2. S. 38–46 (in Russian)].

15. Подловченко Р.И. От схем Янова к теории моделей программ // Математические вопросы кибернетики. М.: Наука, Физматлит, 1998. Вып. 7. С. 281–302 [Podlovchenko R.I. Ot shem Janova k teorii modelej programm // Matematicheskie voprosy kibernetiki. M.: Nauka, Fizmatlit, 1998. Vyp. 7. S. 281–302 (in Russian)].

16. Подловченко Р.И. Абстрактные программы с процедурами и конечные автоматы с магазином // Интеллектуальные системы. М.: изд-во МГУ, 1997. Т. 2, вып. 1–4. С. 275–295 [Podlovchenko R.I. Abstraktnye programmy s procedurami i konechnye avtomaty s magazinom // Intellektualnye sistemy. M.: izd-vo MGU, 1997. T. 2, vyp. 1–4. S. 275–295 (in Russian)].

17. Подловченко Р.И., Долгих Б.А. Двухступенчатое моделирование программ с процедурами // Математические вопросы кибернетики. М.: Физматгиз, 2003. Вып. 12. С. 47–56 [Podlovchenko R.I., Dolgih B.A. Dvuhstupenchatoe modelirovanie programm s procedurami // Matematicheskie voprosy kibernetiki. M.: Fizmatgiz, 2003. Vyp. 12. S. 47–56 (in Russian)].

18. Подловченко Р.И. Алгебраические модели программ и автоматы // Математические вопросы кибернетики. М.: Физматгиз, 2003. Вып. 12. С. 47–56 (Podlovchenko R.I. Algebraicheskie modeli programm i avtomaty // Matematicheskie voprosy kibernetiki. M.: Fizmatgiz, 2003. Vyp. 12. S. 47–56 (in Russian)].

19. Cousot P. Constructive design of a hierarchy of semantics of transition system by abstract interpretation // Theoretical Computer Science. 2002. V. 277, N 86. P. 47–103.

20. Senizergues G. The equivalence problem for deterministic pushdown automata is decidable // Lectoure Notes in Computer Science. 1997. V. 1256. P. 271–281.

21. Zakharov V.A., Kuzurin N.N., Podlovchenko R.I., Scherbina V.V. Using algebraic models of programs for detecting metamorfic malwares // Труды Института Системного Программирования. М.: ИСП РАН, 2007. Т. 12. С. 77–94.

22. Подловченко Р.И. Методология построения системы эквивалентных преобразований, полной в модели вычислений, и её применение для алгебраических моделей программ // Труды семинара «Семантика, спецификация и верификация программ: теория и приложения» (Казань, 14–15 июня 2010). Казань: Отечество, 2010. С. 82–87 [Podlovchenko R.I. Metodologija postroenija sistemy ekvivalentnyh preobrazovanij, polnoj v modeli v ychislenij, i ee primenenie dlja algebraicheskih modelej programm // Trudy seminara "Semantika, specifikacija i verifikacija programm: teorija i prilozhenija". Kazan, 2010 (in Russian)].

23. Подловченко Р.И. Полные системы эквивалентных преобразований в уравновешенных полугрупповых моделях программ с левым сокращением // Программирование. 2010. №3. С. 3–18 (Podlovchenko R.I. Complete systems of equivalent transformations in balanced semigroup models of programs with left cancellation // Programming and Computer Software. 2010. No 3. P. 125–137).

Equivalence Problem Solvability in Biparametric Gateway Program Models

Molchanov A.E.

Lomonosov Moscow State University,
GSP-1, Leninskie Gory, Moscow, 119991, Russian Federation

Keywords: program formalization, program scheme, program schemes equivalence, gateway program model

Algebraic program models with procedures are designed to analyze program semantic properties on their models called program schemes. Procedural liberisation problem and equivalence problem are stated for program models with procedures in which both defining parameters are chosen independently. Program models with procedures built over a given program model without procedures are investigated. Algorithms for both stated tasks are proposed for models where an additional restriction is applied: the intersection emptiness problem is solvable in the program model without procedures. Polynomial estimates for the complexity of the algorithms are shown. Some topics for further investigation are proposed.

Сведения об авторе:
Молчанов Андрей Эрикович,
Московский государственный университет им. М.В. Ломоносова,
аспирант