Модел. и анализ информ. систем. Т. **21**, № **2** (2014) 56–70

УДК 681.3.06

Разрешимость эквивалентности в перегородчатых моделях программ

Подловченко Р. И., Молчанов А. Э.

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

e-mail: podlovchenko.rimma@gmail.com, gurux13@gmail.com

получена 16 марта 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 формулируются проблемы либеризации и эквивалентности. Рассматривается подкласс моделей программ с процедурами – специальные перегородчатые модели. Для таких моделей программ улучшена оценка сложности алгоритма, решающего проблему либеризации. Введено понятие примитивных схем программ как подкласса специальных перегородчатых. Для них установлена разрешимость проблемы эквивалентности при разрешимости проблемы эквивалентности в специального вида моделях программ без процедур. Для некоторых случаев разрешимости проблемы эквивалентности приведены оценки сложности.

1. Введение

В данной статье продолжены исследования перегородчатых моделей программ, начатые в [1]. Остановимся коротко на тематике статьи [1].

Основная часть её посвящена описанию метатеории алгебраических моделей программ с процедурами и описанию проблематики самой теории этих моделей. При этом метатеория занимается отбором моделей, подлежащих изучению, а проблематика нацелена 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, а для таких моделей проблема эквивалентности достаточно хорошо изучена. Поэтому часть статьи [1] посвящена исследованию перегородчатых моделей.

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

Наложением определённых требований на параметры перегородчатой модели в [1] введены специальные перегородчатые модели и описан алгоритм построения по схеме эквивалентной ей схемы, претендующей на статус свободной в модели. Однако доказательство того, что претензия обоснована, опущено.

К отмеченному недочёту добавилось ошибочное утверждение о критерии разрешимости проблемы эквивалентности в специальной перегородчатой модели.

В связи с изложенным стали необходимыми дальнейшие исследования в моделях этого типа, чему и посвящена данная статья.

Здесь нами доказана теорема 1 о разрешимости проблемы либеризации в специальной перегородчатой модели, причём предложен разрешающий её алгоритм, имеющий меньшую сложность, чем алгоритм, рассмотренный в [1]. Далее введено понятие примитивной схемы в специальной перегородчатой модели и установлено, что в классе примитивных схем проблема эквивалентности разрешима, если она разрешима в модели, индуцирующей эту специальную перегородчатую (теорема 2). Дана и оценка сложности разрешающего алгоритма.

В заключение приведены случаи, когда в индуцирующих моделях решена проблема эквивалентности, причём с оценками сложности решения. Это – так называемые уравновешенные полугрупповые модели программ без процедур, исследованные в [2] – [4].

Нами воспроизведены определения всех рассматриваемых здесь моделей программ, дабы не отсылать читателя к статье [1].

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.

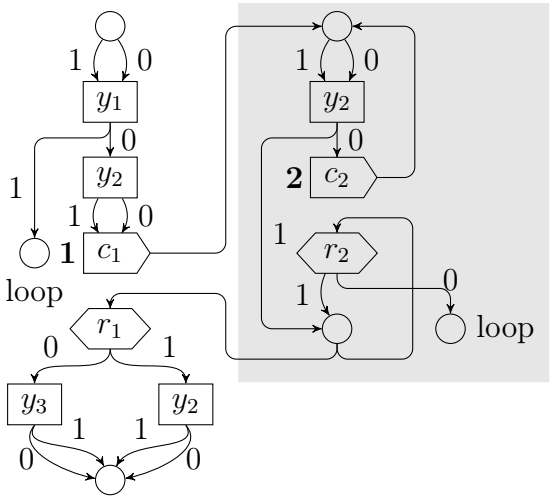

Рис. 1: Пример схемы программы

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, состоящий в путешествии по схеме и сопровождающийся построением слова из H; оно 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 $j, j = 0, \dots, k$, 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 его вершины, 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 либеризации в перегородчатой модели программ

Обозначим M рассматриваемую нами перегородчатую модель программ и определим проблему 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 *свободной в* M, если всякая её вершина, отличная от $loop$, находится на некотором реализуемом маршруте через G. *Проблема либеризации в* M состоит в поиске алгоритма, который для любой поступившей на его вход схемы из M строит эквивалентную ей свободную схему. Если такой алгоритм найден, проблема называется *разрешимой.*

Рассмотрим проблему либеризации для частного вида перегородчатых моделей, именуемых нами *специальными.* Так называется модель, индуцируемая простой моделью с параметрами τ и l, удовлетворяющими требованиям: какими бы ни были цепочки h_1, h_2 над Y и функция μ из l,

$$h_1 h_2 \overset{\tau}{\sim} h_2 \Rightarrow h_2 = \Lambda,$$

$$h_1 \overset{\tau}{\sim} h_2 \Rightarrow \mu h_1 = \mu h_2,$$

где $\mu \in l$.

В этом случае τ называется *эквивалентностью без циклов*, а μ – *согласованной с ней функцией разметки*. Обозначим M_1 специальную перегородчатую модель и M_0 – индуцирующую её простую модель. Подчеркнём, что M_0 – это подмодель модели M_1.

Основной в данном разделе является теорема 1.

Теорема 1. *Проблема либеризации в M_1 разрешима.*

Прежде всего установим следующее утверждение.

Утверждение 1. *В M_0 разрешима проблема либеризации.*

Действительно, пусть G – простая схема. Поскольку τ – эквивалентность без циклов, а l содержит все согласованные с ней функции, любой маршрут через G реализуем. Таким образом, трансформация схемы G в эквивалентную ей свободную осуществляется алгоритмом, который сначала отмечает вершины, достижимые маршрутами из входа схемы, удаляя из неё недостижимые вершины, а затем встречной разметкой вершин из выхода схемы находит те вершины, из которых выход недостижим, направляет в *loop* дуги, идущие в них из отмеченных вершин, а сами вершины удаляет. Очевидно, что полученная схема свободна и эквивалентна схеме G.

Теперь займёмся анализом маршрутов через схему из M_1, не являющуюся простой. Пусть G – такая схема. *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.

Легко увидеть, что для любой опорной вершины эффективно, то есть алгоритмом, выявляются все её преемники. При этом выход не имеет преемников вообще, преемниками входа могут быть вызов и выход, преемниками вызова – вызов и возврат, а преемниками возврата – вызов, выход схемы и возврат, и в случае, когда преемником является возврат, он определяется не всегда однозначно (из-за перехода через финальную вершину, из которой дуги ведут в общем случае в разные возвраты).

Любой реализуемый маршрут через схему из M_1 представляет собой последовательность примыкающих друг к другу е-маршрутов. Рассмотрим его проекцию на вызовы и возвраты. Если она не пуста, очевидными свойствами проекции являются следующие:

1) каждому вхождению вызова (возврата) соответствует вхождение парного с ним возврата (вызова), и вхождение вызова расположено раньше вхождения возврата;

2) имеются парные вызов и возврат, вхождения которых соседствуют;

3) если между вхождениями парных вызова и возврата находится вхождение какого-либо вызова (возврата), то между этими вхождениями присутствует и вхождение парного ему возврата (вызова).

При доказательстве теоремы 1 учитываем эти свойства.

Доказательство теоремы 1. Пусть G – схема из M_1, отличная от простой. Опишем алгоритм ρ_1, трансформирующий её в эквивалентную свободную.

На первом этапе своей работы алгоритм ρ_1 виртуально проводит в схеме дуги из вызовов в парные им возвраты и рассматривает полученные при этом ориентированные пути из входа схемы в её выход и из инициальной вершины подграфа в его финальную вершину. Он называет их *трассами*.

Согласно изложенным ранее замечаниям о реализуемых маршрутах через схему, имеет место следующее: если в главном подграфе отсутствуют трассы, содержащие вызов, или нет процедурных подграфов с трассами без вызовов, то свободная схема, эквивалентная схеме G, является простой. Она пуста, если в главном подграфе нет трасс вообще, а в противном случае состоит из вершин, принадлежащих е-маршрутам из входа схемы в выход, соскальзывающие с которых дуги ведут в $loop$. Построением такой схемы алгоритм ρ_1 завершает работу.

Если же имеются отмеченные выше трассы, алгоритм переходит ко второму этапу работы, выполняя разметку вызовов в схеме.

Он называет *изначально сквозной* трассу в процедурном подграфе, не содержащую вызовов. Для всякой такой трассы алгоритм отмечает вызовы, ассоциируемые с его подграфом. Если после этого появляются трассы, в которых отмечены все принадлежащие ей вызовы, то алгоритм называет их тоже *сквозными* и работает с ними также как с изначально сквозными. Процесс выявления сквозных трасс продолжается до тех пор, пока появляются новые сквозные трассы. С их исчерпанием разметка закончена.

Если в главном подграфе не оказалось сквозных трасс, то свободная схема, эквивалентная схеме G, проста, и алгоритм строит её по описанным выше правилам.

В противном случае он проводит выделение сквозных трасс, принадлежащих реализуемым маршрутам через схему, начиная с тех из них, что находятся в главном подграфе. *Выделенной* является сквозная трасса, находящаяся в подграфе, в который ведёт дуга из вызова, принадлежащего предыдущей выделенной трассе.

Закончив этот процесс, алгоритм выполняет заключительную работу, удаляя из схемы не функционирующие в ней элементы. Он оставляет в схеме только те процедурные подграфы, в которых имеются выделенные сквозные трассы. В каждом таком подграфе алгоритм удаляет вершины, не принадлежащие е-маршрутам из вызовов, находящихся в выделенной трассе, и из парных им возвратов; при этом дуги, ведущие из оставленных вершин в удаляемые, направляет в $loop$. Подобным образом алгоритм обрабатывает и главный подграф, рассматривая в нём добавочно е-маршруты из входа в выход. Свободная схема, эквивалентная схеме G, построена, и алгоритм останавливается.

Теорема 1 доказана.

Обратимся к вопросу о сложности алгоритма ρ_1, сосредоточив внимание на оценке старшего члена в верхней оценке алгоритма.

Пусть G – схема, поступившая на вход алгоритма ρ_1. Она задаётся таблицей переходов, в которой строки идентифицируются вершинами схемы, а столбцы – наборами из X. Предположим, что в G нет процедурных подграфов без входящих в них дуг, и есть хотя бы один процедурный подграф.

Обозначим l – число подграфов в схеме G, m – число вызовов в ней и n – максимум количества преобразователей в отдельных подграфах схемы. Отметим, что $l \leq m + 1$.

В работе алгоритма ρ_1 следует рассмотреть этапы, состоящие в построении трасс в схеме, в выделении сквозных трасс и в освобождении схемы от неработающих элементов.

При построении трасс для всякой вершины v типа вход, инициальная, возврат отыскиваются достижимые из неё по преобразователям вершины типа выход, финальная, вызов. При этом для v просматриваются ориентированные пути из неё, идущие по преобразователям и содержащие не более одного витка цикла. Это требует для v

$$b_0 n \log n$$

операций, где b_0 – константа. Таким образом, старший член в верхней оценке сложности построения трасс в схеме выражается числом

$$b_1(m + l)n \log n, \tag{1}$$

где b_1 – константа.

На этом этапе в таблице переходов отмечаются специальным символом вызовы и возвраты, принадлежащие трассам. Этап завершается вычёркиванием в таблице переходов строк, соответствующих неотмеченным вызовам и возвратам, а также тем инициальным вершинам, которые оказались без входящих в них дуг; вместе с вычёркиванием строки, соответствующей инициальной вершине, удаляется и строка, соответствующая финальной вершине того же подграфа.

Заметим, что экстремальным при оценке сложности алгоритма ρ_1 является случай, когда таблица переходов осталась неизменной, т.е. сохранились её параметры l, m, n. Далее будем отталкиваться от этого случая.

Выделение сквозных трасс осуществляется просмотром в таблице переходов строк, идентифицируемых инициальными вершинами или входом. Предположим, что обнаружены изначально сквозные трассы. Тогда на каждом шаге алгоритм ρ_1 отмечает хотя бы один вызов, т.е. число шагов не превышает m. Поскольку на отдельном шаге просматривается в таблице переходов не более l строк, старший член в верхней оценке сложности второго этапа имеет вид

$$b_2 ml, \tag{2}$$

где b_2 – константа.

Наконец, переходим к третьему этапу, т.е. освобождению схемы от неработающих элементов. Тогда для вершин типа вход, инициальная, возврат строятся простые схемы, ведущие в вершины типа выход, финальная, вызов. Число таких схем не превышает $(m + l)^2$. При построении каждой схемы выполняется

$$b_3 n \log n$$

операций. Таким образом, старший член в верхней оценке сложности третьего этапа имеет вид

$$b_3(m + l)^2 n \log n, \tag{3}$$

где b_3 – константа.

Обозревая (1), (2), (3) и учитывая, что $l \leq m+1$, приходим к тому, что старший член в верхней оценке сложности алгоритма ρ_1 имеет вид

$$b_4(m+1)^2 n \log n,$$

где b_4 – константа.

4. Классы примитивных схем программ

Задача данного раздела – исследовать проблему эквивалентности в специальной перегородчатой модели программ. Она по-прежнему обозначается M_1.

Основной результат устанавливается теоремой 2.

Теорема 2. *Если проблема эквивалентности разрешима в простой модели, индуцирующей специальную перегородчатую, то она разрешима в классе примитивных схем, принадлежащих этой модели.*

Примитивной в модели M_1 называется свободная схема программы, удовлетворяющая требованию: какой бы ни была её опорная вершина, все её преемники помечены различающимися символами. Очевидно, что всякая свободная простая схема примитивна.

Доказательству теоремы 2 предпошлём следующие понятия. Пусть v_1 – опорная вершина в схеме из M_1 и v_2 – её преемник. Обозначим $G(v_1, v_2)$ простую схему, построенную по следующим правилам: в ней оставляются все вершины, принадлежащие e-маршрутам рассматриваемой схемы 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, финальную – выходом.

Маршруты в схемах из M_1 называются *сочетаемыми*, если они прокладываются общей для них допустимой для M_1 функцией разметки.

Цепочкой несомой маршрутом схемы назовём ту, что получается выписыванием друг за другом символов, сопоставленных вершинам маршрута, при просмотре его от начала к концу.

Сопряжёнными в схемах из M_1 назовём опорные вершины, которые являются концами сочетаемых маршрутов, несущих эквивалентные цепочки.

Доказательство теоремы 2. Оно опирается на следующие факты, извлекаемые из самого определения эквивалентности свободных схем из модели M_1. Пусть рассматриваются две эквивалентные свободные схемы, тогда

1) для всякой опорной вершины в одной из них имеется сопряжённая с ней вершина в другой схеме, и если это – вызовы, то парные им возвраты помечены одинаково;

2) пары сопряженных вершин могут повторяться, но число различных пар конечно;

3) если сочетаемые маршруты оканчиваются в паре сопряженных вершин, отличных от выходов схем, то они без потери сочетаемости однозначно продолжаются до следующей пары сопряженных вершин.

Отсюда следует, как построить алгоритм, распознающий эквивалентность примитивных схем. Обозначим его ρ_2, и пусть на его вход поступили две примитивные схемы G_1, G_2. Тогда, начиная с пары сопряжённых вершин, состоящей из входов схем, алгоритм ρ_2 для каждой очередной пары сопряженных вершин выполняет следующее. Он находит всех преемников этих вершин и проверяет, выполняется ли требование: для каждого преемника одной вершины имеется преемник другой вершины с той же меткой. Если оно не выполнено, то алгоритм ρ_2 квалифицирует схемы как неэквивалентные и останавливается. Если это требование выполнено, то на основании примитивности схем между найденными преемниками имеется взаимно-однозначное соответствие, и алгоритм ρ_2 проверяет, выполняется ли требование 2. Если нет, то он останавливается, объявляя схемы неэквивалентными. Иначе он в общем случае пополняет список пар сопряженных вершин новыми парами и переходит к рассмотрению следующей необработанной пары. Согласно 1, в списке конечное число пар, и если алгоритм ρ_2 исчерпал его без остановки на одной из них, то он установил эквивалентность схем и завершил свою работу.

Теорема 2 доказана.

При оценке сложности алгоритма ρ_2 по-прежнему выявляем старший член в верхней оценке сложности.

Пусть G_1, G_2 – примитивные схемы, поступившие на вход алгоритма ρ_2. Каждая из них задана таблицей переходов, которая построена описанным ранее способом. Обозначим l_i – число подграфов, m_i – число вызовов и n_i – максимум числа преобразователей в отдельных подграфах схемы G_i, $i = 1, 2$. Пусть

$$l = \max\{l_1, l_2\},$$
$$m = \max\{m_1, m_2\},$$
$$n = \max\{n_1, n_2\}.$$

Обозначим ρ_0 алгоритм, распознающий эквивалентность простых схем из M_0, и пусть $f(k)$ – сложность алгоритма ρ_0 при подаче на него схем, в которых число преобразователей не превосходит k; здесь $k \geq 0$.

Заметим, что случаем максимальной сложности алгоритма ρ_2 является эквивалентность схем G_1, G_2, и рассмотрим этот случай.

В своей работе алгоритм ρ_2 для пар сопряженных вершин схем G_1, G_2, а их не более $2m + 1$, выполняет по три этапа действий для каждой пары: строит преемников вершин, входящих в пару, проверяет, совпадают ли метки преемников одной вершины из пары с метками преемников второй вершины, и если совпадают, то проверяет эквивалентность простых схем, соответствующих преемникам с одинаковой меткой.

На первом этапе старший член в верхней оценке его сложности имеет вид

$$d_1 n \log n,$$

на втором этапе –

$$d_2(m+1)\log(m+1),$$

на третьем этапе –

$$(m+1)(d_3 n \log n + f(n)),$$

где d_1, d_2, d_3 – константы.

Здесь мы применяем рассуждения, выполненные при оценке сложности алгоритма ρ_1.

Таким образом, старший член в оценке сложности алгоритма ρ_2 имеет вид

$$(2m+1)[d_2(m+1)\log(m+1) + d_3(m+1)n\log n + (m+1)f(n)],$$

т.е. для $\hat{m} = m+1$ не превышает числа

$$d\hat{m}^2(\log \hat{m} + n\log n + f(n)),$$

где d – константа.

5. Заключение. Случаи разрешимости проблемы эквивалентности в простой модели программ

Все приводимые ниже оценки относятся к уравновешенным полугрупповым моделям программ с левым сокращением. Такие модели рассматривались в [2] и [3]. Параметры τ и l отдельной модели удовлетворяют следующим требованиям.

Эквивалентность ν является полугрупповой, уравновешенной и обладающей свойством левого сокращения, т.е.

- какими бы ни были цепочки h_1, h_2, h_3, h_4 над Y, выполняется импликация

$$(h_1 \overset{\tau}{\sim} h_2)\&(h_3 \overset{\tau}{\sim} h_4) \Rightarrow h_1 h_3 \overset{\tau}{\sim} h_2 h_4;$$

- для любых цепочек над Y из их τ-эквивалентности следует равенство их длин;
- какими бы ни были цепочки h_1, h_2, h_3, h_4 над Y, выполняется импликация

$$(h_1 h_3 \overset{\tau}{\sim} h_2 h_4)\&(h_1 \overset{\tau}{\sim} h_2) \Rightarrow h_3 \overset{\tau}{\sim} h_4.$$

Множество l состоит из всех τ-согласованных функций разметки.

Пусть M_0 – простая уравновешенная полугрупповая модель программ с левым сокращением. В [2] и [3] установлена разрешимость проблемы эквивалентности в M_0, причём разрешающий алгоритм имеет экспоненциальную сложность относительно размеров испытуемых схем.

В связи с этим в [2] и [3] введено дополнительное требование к τ: говорим, что τ обладает свойством правого сокращения, если для любых цепочек h_1, h_2, h_3, h_4 над Y выполняется импликация

$$(h_1 h_3 \overset{\tau}{\sim} h_2 h_4)\&(h_3 \overset{\tau}{\sim} h_4) \Rightarrow h_1 \overset{\tau}{\sim} h_2.$$

Тогда в верхней оценке сложности алгоритма, распознающего эквивалентность схем, старшие члены имеют вид $C_5 n^5 \log n$ и $C_6 n^4 q(n) \log n$, где n – максимум размеров испытуемых схем, $q(n)$ – сложность распознавания τ–эквивалентности цепочек над Y, имеющих длину n, а C_5, C_6 – константы. Этот результат получен тоже в [2], [3].

Наконец, в [4] рассмотрены частные случаи уравновешенной полугрупповой модели с левым и правым сокращением – коммутативные модели. В такой модели эквивалентность τ определяется так. Задается множество T, состоящее из пар (y_1, y_2), где $y_1, y_2 \in Y$; множество T непременно содержит все пары вида (y, y), где $y \in Y$, в любой паре (y_1, y_2) из T символы y_1, y_2 называются *перестановочными*. Две цепочки над Y объявляются τ-эквивалентными, если любая из них может быть получена из другой перестановками соседних перестановочных символов.

Для коммутативной модели программ различаются случаи: перестановочность символов транзитивна или нет. В первом случае выполняется условие: если $(y_1, y_2), (y_2, y_3)$ принадлежат множеству T, то ему принадлежит и пара (y_1, y_3).

В [4] установлено, что в первом случае старшим членом в верхней оценке сложности алгоритма, разрешающего эквивалентность схем, является $C_7 n^2 \log n$, а во втором – $C_8 n^3 \log n$. Здесь n – максимум размеров испытуемых схем, а C_7, C_8 – константы.

На основании полученной оценки для алгоритма ρ_2 и приведенных оценок возможно установить оценки для алгоритмов проверки эквивалентности в классах примитивных схем из перегородчатых моделей, индуцированных перечисленными классами простых моделей.

Пусть $\hat{m} = \max\{m_1, m_2\} + 1, n = \max\{n_1, n_2\}$, где m_i – число вызовов в схеме G_i, n_i – максимальное по подграфам число преобразователей в схеме G_i $i = 1, 2$.

Утверждение 2. *В уравновешенной полугрупповой перегородчатой модели программ с левым и правым сокращением существует алгоритм проверки эквивалентности со сложностью не больше*

$$K_1 \hat{m}^2 (\log \hat{m} + n^5 \log n + n^4 q(n)),$$

где $q(n)$ – сложность распознавания τ-эквивалентности цепочек над Y, имеющих длину n.

Утверждение 3. *В коммутативной перегородчатой модели программ существует алгоритм проверки эквивалентности со сложностью не больше*

$$K_2 \hat{m}^2 (\log \hat{m} + n^3 \log n).$$

Утверждение 4. *В транзитивно коммутативной перегородчатой модели программ существует алгоритм проверки эквивалентности со сложностью не больше*

$$K_3 \hat{m}^2 (\log \hat{m} + n^2 \log n).$$

Здесь K_1, K_2, K_3 – константы.

Этими рассуждениями подтверждается, что коммутативные модели представляют практический интерес.

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. 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. No 5. P. 17–24 (in Russian)].

3. Подловченко Р.И. Об одной методике распознавания эквивалентности в алгебраических моделях программ // Программирование. 2011. Т. 37, №6. С. 33–43. [English trans.: Podlovchenko R.I. On an equivalence checking technique for algebraic models of programs // Programming and Computer Software. 2011. V. 37, No 6. P. 292–298].

4. Подловченко Р.И. Об одном классе алгебраических моделей программ, представляющем практический интерес // Программирование. 2013. №3. С. 15–28. [English trans.: Podlovchenko R.I. On a class of algebraic models of programs of practical interest // Programming and Computer Software. 2013. V. 39, No 3. P. 124–134].

5. 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. P. 46–74 (in Russian)].

6. 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. P. 75–127 (in Russian)].

7. 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. P. 5–39 (in Russian)].

8. 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. P. 23–53 (in Russian)].

9. 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. P. 303–324 (in Russian)].

10. Захаров В.А. Проверка эквивалентности программ при помощи двухленточных автоматов // Кибернетика и системный анализ. 2010. №4. С. 39–48 [Zaharov V.A. Proverka ekvivalentnosti programm pri pomoschi dvuhlentochnyh avtomatov // Kibernetika i sistemnyj analiz. 2010. №4. P. 39–48 (in Russian)].

11. 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)].

12. Лисовик Л.П. Металинейные рекурсивные схемы над размеченными деревьями // Программирование. 1983. №5. С. 13–22 [Lisovik L.P. Metalinejnye rekursivnye shemy nad razmechennymi derevjami // Programmirovanie. 1983. №5. P. 13–22 (in Russian)].

13. Подловченко Р.И., Попов С.В. Аппроксимируемость одних моделей другими // Вестник Московского университета. Сер. 15: Вычислительная математика и кибернетика. 2001. №2. С. 38–46 [Podlovchenko R.I., Popov S.V. Approksimiruemost odnih modelej drugimi // Vestnik Moskovskogo universiteta. Ser. 15: Vychislitelnaja matematika i kibernetika. 2001. №2. P. 38–46 (in Russian)].

14. 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. P. 281–302 (in Russian)].

15. Подловченко Р.И. Абстрактные программы с процедурами и конечные автоматы с магазином // Интеллектуальные системы. М.: 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.: MGU, 1997. T. 2, vyp. 1–4. P. 275–295 (in Russian)].

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

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

18. Cousot P. Constructive design of a hierarchy of semantics of transition system by abstract interpretation // Theoretical Computer Science. 2002. V. 277, №86. P. 47–103.

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

20. 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. Т. 12. М.: ИСП РАН, 2007. С. 77–94.

Equivalence Problem Solvability in Gateway Program Models

Podlovchenko R.I. 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. Liberisation and equivalence problems are stated for program models with procedures. A subclass of program models with procedures called special gateway models is investigated. A better complexity algorithm for the liberisation in such models is proposed. Primitive program schemes are defined as a subclass of special gateway models. It is shown that the equivalence problem in such models is decidable if the equivalence problem is decidable in special program models without procedures. For some cases of decidability the complexity is evaluated.

Сведения об авторах:
Подловченко Римма Ивановна,
МГУ им. М.В. Ломоносова,
д-р физ.-мат. наук, проф., ведущий научный сотрудник НИВЦ;
Молчанов Андрей Эрикович,
МГУ им. М.В. Ломоносова,
аспирант ф-та ВМК