Модел. и анализ информ. систем. Т. **21**, № **4** (2014) 116–131

УДК 681.3.06

Исследование примитивных схем программ с процедурами

Подловченко Р. И.

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

e-mail: podlovchenko.rimma@gmail.com

получена 20 августа 2014

Ключевые слова: алгебраическая модель программ, эквивалентные преобразования схем программ

В статье рассматриваются алгебраические модели программ с процедурами, предназначенные для изучения семантических свойств программ на их схемах. Так возникают проблемы эквивалентности схем программ и проблема построения полной системы эквивалентных преобразований схем программ. Среди алгебраических моделей программ с процедурами выделены перегородчатые модели, индуцируемые моделями программ без процедур, и принадлежащие им примитивные схемы программ. Для них разрешима проблема эквивалентности. В данной статье в случае, когда индуцирующей является уравновешенная полугрупповая модель программ с левым сокращением, для определенного подкласса примитивных схем построена полная в нем система эквивалентных преобразований схем.

1. Введение

Данная статья является продолжением статьи [1]. В связи с этим коротко опишем содержание последней.

В [1] даётся определение алгебраической модели программ с процедурами. Если фиксирован базис, над которым строятся эти модели, то все они имеют одно и то же множество своих объектов, называемых схемами программ. Отличаются друг от друга модели только отношением эквивалентности схем. Отдельное отношение определяется двумя параметрами, которые специфицируют саму модель, называясь её параметрами.

Основными для алгебраических моделей программ с процедурами являются проблема эквивалентности схем в модели и проблема эквивалентных преобразований (э.п.) схем в ней. Первая состоит в поиске алгоритма, который, получив на свой вход две схемы из модели, распознаёт, эквивалентны они или нет. Вторая проблема заключается в построении системы э.п. схем, полной в модели, т.е. обладающей

свойством: какими бы ни были две эквивалентные схемы из модели, конечной цепочкой преобразований, принадлежащих данной системе, любая из них переводится в другую. При этом допускаются лишь структурные преобразования схем. Проблема э.п. рассматривается в моделях с разрешимой проблемой эквивалентности.

В любой модели программ с процедурами имеется подмодель, состоящая из схем, в которых отсутствуют обращения к процедурам; эти схемы и сама подмодель называются простыми.

В [1] описан приём, как по параметрам простой подмодели строятся параметры самой модели; последняя называется перегородчатой моделью, индуцируемой первой. В перегородчатой модели выделен класс так называемых примитивных схем и установлено, что в нём разрешима эквивалентность схем, если она разрешима в индуцирующей подмодели.

Возникает следующая задача: допустим, что в подмодели, индуцирующей перегородчатую, разрешима и проблема э.п. схем; найти случаи, когда она будет разрешимой и в некотором подклассе примитивных схем, принадлежащих этой перегородчатой модели.

В данной статье установлено следующее. Если в качестве индуцирующей брать уравновешенную полугрупповую модель программ с левым сокращением, в которой разрешимость проблемы э.п. доказана в [3], то существует эффективно описываемый подкласс примитивных схем с разрешимой в нём проблемой э.п. схем. Установленный факт является новым в теории алгебраических моделей программ с процедурами.

Статья состоит из шести разделов. Во втором в краткой форме восстанавливаются понятия общего вида алгебраической модели программ с процедурами и её частного вида – перегородчатой модели. Формулируется как теорема 2.1 факт, установленный в [1] для перегородчатой модели.

Третий раздел посвящён описанию примитивных схем, принадлежащих перегородчатой модели. В новой редакции даётся результат о разрешимости проблемы эквивалентности в классе примитивных схем (теорема 3.1). Здесь же вводится понятие особой примитивной схемы, эффективно распознаваемой в классе примитивных схем.

В четвертом разделе излагаются концепции, которыми руководствуется построение формального исчисления, создаваемого для решения проблемы э.п. схем. Предполагается, что эта проблема рассматривается либо в общего вида алгебраической модели программ, либо в классе принадлежащих ей схем. Описывается та часть формального исчисления, которая предшествует введению аксиом исчисления.

Поскольку далее рассматриваются примитивные схемы из перегородчатой модели, индуцированной уравновешенной полугрупповой моделью программ с левым сокращением, пятый раздел посвящён определению этой модели и описанию того, как отыскиваются аксиомы создаваемого для неё формального исчисления. Данный раздел представляет собой аналитический обзор статьи [3].

Наконец, в шестом разделе осуществляется построение аксиом формального исчисления, создаваемого для класса особых примитивных схем, и доказывается, что этими аксиомами индуцируется полная в этом классе система э.п. Теорема 6.2 о полноте построенной системы завершает раздел и статью в целом.

2. Основные понятия

Определим алгебраическую модель программ. Она строится над *базисом* Y, C, R, X, где Y, C, R – конечные алфавиты, элементы которых называются *символами операторов*, *вызовов* и *возвратов* соответственно, а X – конечное множество, элементы которого именуются *наборами*. Объектами модели являются схемы программ.

Схема программы представляет собой конечный ориентированный граф, состоящий из связанных между собой подграфов, один из которых называется *главным*, а остальные – *процедурными*. В главном подграфе выделены вершина – *вход* без входящих в неё дуг и вершина – *выход* без исходящих из неё дуг, в каждом процедурном подграфе выделены *инициальная* и *финальная* вершины. Всякому подграфу принадлежит своя "мёртвая" вершина *loop*, не имеющая исходящих из неё дуг. Связь между подграфами осуществляется с помощью вершин, называемых *вызовами* и парными им *возвратами*, а именно: из вызова исходит единственная дуга, которая ведет в инициальную вершину некоторого процедурного подграфа, и тогда из финальной вершины этого подграфа выходит дуга в возврат, парный данному вызову; только такие дуги исходят из финальных вершин. Вызов и парный ему возврат находятся в одном и том же подграфе. Все остальные вершины схемы, кроме упомянутых, называются *преобразователями*. Каждая вершина-преобразователь помечена символом из Y, вершина-вызов – символом из C, вершина-возврат – символом из R. Из входа, инициальной вершины, преобразователя и возврата исходят дуги в количестве, равном размеру X, каждая дуга помечена своим символом из X и находится в подграфе, которому принадлежит сама вершина. Структура схемы описана полностью.

В статье [1] дан пример схемы, состоящей из главного и одного процедурного подграфа.

В [1] детально описана процедура выполнения схемы на функции разметки, играющей роль входного данного. *Функция разметки* – это отображение множества цепочек, построенных из символов алфавитов Y, C, R, в множество X. Выполнение схемы на отдельной функции разметки представляет собой детерминированный обход схемы по некоторому ориентированному пути в ней, начинающемуся во входе схемы с пустой цепочкой. Обход сопровождается выписыванием друг за другом символов, метящих проходимые вершины. Выполнение схемы считается *результативным*, если обход достигает выхода схемы, и накопленная к этому моменту цепочка называется *результатом* выполнения. Таким образом, схеме приписывается отображение множества функций разметки в множество цепочек.

Отношение эквивалентности схем индуцируется двумя параметрами:

- отношением ν в множестве цепочек символов базиса; обозначим это множество H;
- множеством L допустимых функций разметки.

Допустимость функции разметки воспринимается как решение рассматривать выполнение схем только на таких функциях. Требуется, чтобы, какой бы ни была допустимая функция разметки, если одна из двух сравниваемых на эквивалентность схем завершает на ней выполнение, то и другая завершает его тоже. При сравнении

результатов выполнения используется отношение ν: требуется, чтобы результативные цепочки были ν-эквивалентными.

Моделью программ над заданным базисом называется множество построенных над ним схем, в котором введена эквивалентность схем. Если она индуцируется параметрами ν, L, то они называются *параметрами модели.*

Простой именуется модель, состоящая из схем без процедурных подграфов; такие схемы называются тоже *простыми.* Любой модели принадлежит подмодель, являющаяся простой.

Далее считаем базис фиксированным.

Перегородчатая модель программ с процедурами представляет собой тот частный случай описанной выше модели, когда её параметры ν, L определяются параметрами собственной простой подмодели; обозначим их τ, l и напомним (см. [1]), как строятся по ним ν, L.

Эквивалентность ν вводится следующим образом. Цепочки h_1, 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

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

где $b_1, b_2 \dots b_k$ есть общая проекция этих цепочек на множество, то при всех $j, j = 0, 1, \dots, k$, имеет место соотношение

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

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

Строгое определение множества L опустим, обращаясь лишь к следующему пояснению. Содержательно выполнение схемы на функции из L происходит так: когда встречается вершина схемы типа вызов или возврат, выбирается какая-либо функция из L и до встречи с вершиной того же типа схема выполняется фактически на этой функции.

Условимся говорить, что перегородчатая модель индуцируется простой моделью с параметрами τ, l.

Как отмечено во введении, для перегородчатых моделей программ основными являются проблема эквивалентности схем в модели и проблема э.п. схем. Напомним, что вторая проблема рассматривается в модели с разрешимой первой проблемой. Её исследованию в [1] предшествует решение так называемой проблемы либеризации в модели. Она заключается в поиске алгоритма, который, получив на свой вход схему из этой модели, строит эквивалентную ей свободную схему, принадлежащую той же модели. Схема программы называется *свободной в модели*, если каждая её вершина находится на пути выполнения схемы на какой-либо допустимой функции разметки.

Имеет место теорема 2.1.

Теорема 2.1. *Если проблема либеризации разрешима в простой модели, индуцирующей перегородчатую, то она разрешима и в этой перегородчатой модели.*

Доказательство её полностью повторяет выполненное в [1] доказательство о разрешимости проблемы либеризации в перегородчатой модели программ. Отличие

одной теоремы от другой состоит в формулировке их посылок: требования к параметрам перегородчатой модели, гарантирующие разрешимость проблемы либеризации в индуцирующей её простой модели, теперь заменены требованием просто её разрешимости.

3. Примитивные схемы программ

В данной статье исследуется проблема э.п. в некотором подмножестве схем программ, принадлежащих перегородчатой модели (обозначим её M), индуцируемой простой моделью M_0. Предполагаем, что в M_0 разрешима проблема либеризации, следовательно, согласно теореме 2.1, она разрешима и в M. Схемы из рассматриваемого нами подмножества именуются примитивными. Здесь мы определим их и изложим результат, полученный для них в [1].

Определению примитивной схемы предпошлём используемые в нём понятия.

Пусть G – свободная схема из M. *Опорной* в G назовём вершину типа вход, вызов, возврат, выход схемы. *Преемником опорной вершины* v, отличной от выхода, назовём опорную вершину u, достижимую из v ориентированным путём, который не содержит внутри опорных вершин. Отметим, что в силу свободности схемы G любая её опорная вершина имеет алгоритмически определяемые преемники. Отметим тип преемника u опорной вершины v:

- если v – вход схемы, то u – либо её выход, либо вызов;
- если v – вызов, то u – либо парный ему возврат, либо вызов;
- если v – возврат, то u – либо выход схемы, либо вызов.

Свободная схема из M называется *примитивной*, если преемники любой её опорной вершины не имеют повторяющихся меток. Легко увидеть, что класс примитивных схем из M является разрешимым, то есть всегда можно установить, принадлежит ему или нет схема из M. В [1] доказана следующая теорема.

Теорема 3.1. *Если в модели M_0 разрешима проблема эквивалентности схем, то она разрешима в классе примитивных схем из M.*

При доказательстве её использовались необходимые признаки эквивалентности примитивных схем. Приведём их.

Пусть G – примитивная схема из M, v – опорная в ней вершина, u – преемник вершины v. Обозначим $G(v,u)$ простую схему, построенную по следующим правилам. В ней оставляются все вершины, принадлежащие ориентированным путям из v в u. Если v – возврат, то он заменяется входом создаваемой схемы, если u – вызов, то он заменяется её выходом. Инициальную вершину всегда считаем входом создаваемой схемы, устраняя вызов v, из которого идёт дуга в данную инициальную вершину, а финальную – выходом, устраняя возврат u, в который из неё идёт дуга. Полагаем, что все оставленные преобразователи и наследуемые из G дуги сохраняют свои метки. Теперь введём в создаваемую схему вершину *loop*, если её там не было, устраним дуги, приходящие в оставленные вершины извне, а дуги, ведущие в вершины наружу, направим в вершину *loop*. Схема $G(v,u)$ построена. Именуем её вырастающей из вершины v.

В эквивалентных примитивных схемах их опорные вершины назовём *сочетаемыми*, если они достижимы из входов схем такими путями, прокладывемыми при выполнении схем на допустимой и общей для них функции разметки, которые несут цепочки, имеющие равные проекции на множество $C \cup R$. *Несомая* путём цепочка строится выписыванием друг за другом символов, метящих вершины этого пути при просмотре его от начала к концу. Отметим, что сочетаемые вершины, если помечены, то общим для них символом.

Пусть v_1, v_2 – сочетаемые вершины эквивалентных примитивных схем G_1, G_2 соответственно, и u_i – преемник вершины $v_i, i = 1, 2$. Если преемники имеют общую для них метку в случае, когда они отличны от выходов схем, то схемы $G_1(v_1, u_1), G_2(v_2, u_2)$ называем *сочетаемыми*.

При доказательстве теоремы 3.1 установлен факт, формулируемый здесь как лемма 3.1.

Лемма 3.1 *В эквивалентных примитивных схемах из модели M каждой опорной вершине в одной из них соответствует единственная сочетаемая с ней опорная вершина в другой, а вырастающие из них сочетаемые схемы эквивалентны.*

Введём понятие особой примитивной схемы. Так называется примитивная схема, в которой каждая её опорная вершина имеет только одного преемника. Алгоритмическая распознаваемость такой схемы следует из её свободности.

В дальнейшем класс особых примитивных схем из модели M обозначается K.

4. Формальное исчисление, создаваемое для решения проблемы э.п. схем

Обсуждаемой является проблема э.п. схем в алгебраической модели программ или в классе принадлежащих ей схем программ. В обоих случаях предполагается разрешимой проблема эквивалентности.

Для решения проблемы э.п. схем строится специального вида формальное исчисление, то есть описываются его элементы и вводятся действующие в нём правила вывода и аксиомы.

Здесь мы изложим концепции, которыми руководствуется такое построение, обобщая при этом результаты исследований, выполненных в [2] и [3].

Итак, пусть S – рассматриваемая алгебраическая модель программ или рассматриваемый класс принадлежащих ей схем программ, и в S разрешима проблема эквивалентности. Говорим, что тип S определяется отношением эквивалентности схем программ из S.

В самом начале построения формального исчисления для S независимо от типа выполняется следующее:

- вводится понятие фрагмента схемы, и фрагменты объявляются элементами создаваемого исчисления;

- определяется единственное в исчислении правило вывода, состоящее в допустимой подстановке во фрагмент вместо его подфрагмента некоторого другого фрагмента; при этом подстановка является обратимой операцией.

Далее используется отношение эквивалентности схем из S и, опираясь на него, выполняется следующее:

- определяется отношение эквивалентности фрагментов с классификацией на безусловную и условную эквивалентность;
- объявляется, что аксиомой исчисления может быть разрешимое множество пар эквивалентных фрагментов, причём состоящее либо из пар безусловно эквивалентных, либо из пар условно эквивалентных фрагментов;
- вводится конечное множество аксиом, и формальное исчисление считается построенным.

Отметим, когда оно служит решению проблемы э.п. в S.

Применяя правило подстановки, каждой аксиоме сопоставляется множество э.п. схем, реализуемых её использованием; оно именуется индуцируемым аксиомой и является разрешимым. Таким образом, конечным числом аксиом в исчислении индуцируется в совокупности разрешимое множество э.п. схем. Если доказывается его полнота в S, то в S решена проблема э.п. схем.

Теперь – о поиске нужных для S аксиом. Он подчиняется стратегии, рекомендации которой состоят в следующем:

- ввести каноническую форму схем из S, подчинив это требованию, чтобы каноническая форма была алгоритмически распознаваема и была единственной с точностью до изоморфизма в своём классе эквивалентных схем из S;
- построить алгоритм, который, получив на свой вход две эквивалентные схемы из S, трансформирует каждую в каноническую, применяя только эквивалентные преобразования их структуры;
- проанализировать этот алгоритм, переводя выполняемые им преобразования на язык фрагментных, и скомпоновать таким образом аксиомы.

Заметим, что анализ алгоритма сопровождает его построение.

Если изложенные рекомендации реализованы и получено конечное число аксиом, то в силу обратимости выполненных алгоритмом преобразований будет получено решение проблемы э.п. схем в S.

Далее опишем ту часть создаваемого формального исчисления, которая предваряет построение аксиом, ибо этот процесс индивидуален для рассматриваемого S.

Начнём с определения фрагмента. Пусть G – схема из S, и V – некоторое множество её вершин. *Фрагментом, порождённым множеством* V, назовём подграф схемы G, который состоит из вершин множества V и всех инцидентных им дуг; вершины и дуги фрагмента наследуют приписанные им в схеме G метки. *Входом во фрагмент* назовём вершину, не принадлежащую V и такую, из которой идёт дуга в вершину из V; *выходом* – вершину, не принадлежащую V и такую, в которой оканчивается дуга из вершины, входящей в V. Обозначим Ф множество фрагментов схем из S. Согласно их определению, каждая схема из S является частным случаем фрагмента из Ф.

Пусть F_1, F_2 – фрагменты из Ф. Говорим, что F_1 *входит* во фрагмент F_2, если в нём имеется подфрагмент, изоморфный фрагменту F_1.

Фрагменты F_1, F_2 назовём *согласованными*, если выполнено следующее: между входами одного и другого установлено взаимно-однозначное соответствие, при котором совпадают множества меток дуг, исходящих из соответствующих друг другу входов, и между выходами фрагментов тоже установлено взаимно-однозначное соответствие.

Определим *операцию подстановки вместо фрагмента F_1 согласованного с ним фрагмента F_2*. Пусть F – фрагмент из Ф, имеющий вхождение фрагмента F_1. Обозначим F_1' подфрагмент фрагмента F, изоморфный фрагменту F_1. Заменим фрагмент F_2 изоморфным ему фрагментом F_2', вершины которого отличны от вершин фрагмента F, и перенесём на F_1', F_2' согласованность фрагментов F_1, F_2. После этого во фрагменте F заменим входы фрагмента F_1' соответствующими им входами фрагмента F_2', а выходы фрагмента F_1' – соответствующими им выходами фрагмента F_2', и устраним внутренние вершины фрагмента F_1'. Операция подстановки в F вместо фрагмента F_1 фрагмента F_2 завершена. Очевидно, что результатом её является фрагмент из Ф, причём схема, если F – схема.

Описанная операция подстановки объявляется единственным правилом вывода в строящемся исчислении.

Согласованные фрагменты F_1, F_2 называются *безусловно эквивалентными*, если выполняется следующее: какой бы ни была схема из S и каким бы ни было вхождение в неё одного из фрагментов F_1, F_2, подстановка вместо этого вхождения другого фрагмента является эквивалентным в S преобразованием данной схемы. *Условная эквивалентность* этих *фрагментов* имеет место в том случае, когда существует алгоритм, определяющий допустимые вхождения подставляемого фрагмента.

В заключение данного раздела отметим, что в последующих разделах статьи для рассматриваемых в них S завершен процесс построения формального исчисления.

5. Уравновешенные полугрупповые модели программ с левым сокращением

Поскольку в исследуемом нами классе K особых примитивных схем в качестве модели M_0, индуцирующей модель M, берётся модель указанного в заглавии типа, дадим её определение и отметим основные установленные для неё факты. Это фактически будет аналитическим обзором статьи [3].

Простая алгебраическая модель программ над базисом Y, P называется *уравновешенной полугрупповой моделью программ с левым сокращением*, если её параметры τ и l удовлетворяют следующим требованиям.

Эквивалентность τ является *полугрупповой*, то есть какими бы ни были цепочки h_1, h_2, h_3, h_4 над Y, выполняется импликация

$$(h_1 \overset{\tau}{\sim} h_2)\&(h_3 \overset{\tau}{\sim} h_4) \Rightarrow h_1h_3 \overset{\tau}{\sim} h_2h_4;$$

и обладающей *левым сокращением*, если для любых цепочек h_1, h_2, h_3, h_4 над Y выполняется импликация

$$(h_1h_3 \overset{\tau}{\sim} h_2h_4)\&(h_1 \overset{\tau}{\sim} h_2) \Rightarrow h_3 \overset{\tau}{\sim} h_4.$$

Кроме того, эквивалентность τ является уравновешенной, то есть τ-эквивалентные цепочки равны по длине.

Множество l состоит из всех τ-согласованных функций разметки, то есть $\mu \in l$, где μ – функция разметки, тогда и только тогда, когда для любых h_1, h_2 над Y выполняется импликация

$$h_1 \overset{\tau}{\sim} h_2 \Rightarrow \mu(h_1) = \mu(h_2).$$

По-прежнему обозначаем M_0 описанную модель программ.

Легко доказуемо, что для модели M_0 разрешима проблема либеризации. В [3] установлены факты, излагаемые ниже как теорема 5.1 и теорема 5.2.

Теорема 5.1. *В модели M_0 разрешима проблема эквивалентности схем.*

Теорема 5.2. *В модели M_0 разрешима проблема э.п. схем.*

Остановимся на решении в M_0 проблемы э.п. и опишем, каким образом в M_0 реализованы рекомендации стратегии поиска аксиом.

Канонической формой в M_0 названа свободная схема из M_0, в которой отсутствуют кусты и, кроме того, нет различающихся и сильно эквивалентных вершин.

Определим понятия, используемые в этом определении.

Пусть G – схема из M_0, и v – преобразователь в ней. *Кустом, произрастающим из v*, называется фрагмент схемы G, состоящий из преобразователей, включая v, и удовлетворяющий следующим требованиям. Все вершины фрагмента распределены по уровням, на каждом из которых находятся равноудалённые от v преобразователи, при этом, если k – число уровней, то дуги, исходящие из вершин, принадлежащих уровню $i, i = 0, 1, \ldots, k-1$, ведут в вершины $i+1$-го уровня; таким образом, из вершины v на k-й уровень ведут простые ориентированные пути; требуется, чтобы все они несли эквивалентные цепочки. Класс эквивалентности этих цепочек называется *маркером* данного *куста*, а вершина v – его *корнем*.

Введём отношение *сильной эквивалентности схем* из M_0: в этом отношении находятся две схемы тогда и только тогда, когда, какой бы ни была допустимая для M_0 функция разметки, если выполнение на ней одной из схем результативно, результативно и выполнение на ней другой схемы, причём результатами являются равные цепочки. Очевидно

Утверждение 5.1. *Из сильной эквивалентности схем в модели M_0 следует их эквивалентность.*

Введём понятие сильной эквивалентности вершин свободной схемы из M.

Предварительно любой вершине v свободной схемы G сопоставим простую и свободную схему $G(v)$, построенную по следующим правилам. Берётся фрагмент схемы G, порождённый всеми её вершинами, достижимыми из v ориентированными путями. Среди них, в силу свободности схемы G, находится её выход, который будем считать выходом схемы $G(v)$. Реконструируем этот фрагмент, внеся в него вершину *loop*, если её не было, затем устраним все входящие во фрагмент извне его дуги и направим в вершину *loop* все исходящие из него наружу дуги. В заключение добавим вершину, именуемую входом схемы $G(v)$, отправив в вершину v все исходящие из неё дуги. Схема $G(v)$ построена.

Вершины v_1, v_2 свободной схемы G назовём *сильно эквивалентными*, если сильно эквивалентны схемы $G(v_1), G(v_2)$. Справедливо

Утверждение 5.2. *Сильная эквивалентность вершин схем из модели M_0 алгоритмически распознаваема.*

Итак, алгоритмическая распознаваемость канонической формы в M_0 установлена.

Чтобы доказать единственность канонической формы в своём классе эквивалентных схем из M_0, воспроизведём результаты, полученные в [3].

Лемма 5.1. *В свободных эквивалентных схемах из модели M_0 из разнопомеченных их сопряженных вершин вырастают кусты общего для них маркера.*

Здесь *сопряжёнными* в двух схемах называются преобразователи, в которых завершаются такие пути, идущие из входов схем и пролагаемые общей для них допустимой функцией разметки, которые несут цепочки, являющиеся эквивалентными по устранении в них последнего символа. Кусты, о которых говорится в лемме 5.1, называются тоже *сопряжёнными*.

Приведём ещё один факт, установленный в [3] для схем из M_0.

Пусть G – схема из M_0 и F – куст в ней. Его *x-выходом*, где x – набор из X, назовём вершину в G, в которую ведёт дуга с меткой x из вершины, принадлежащей последнему уровню куста F.

Лемма 5.2. *В эквивалентных свободных схемах из M_0 выполняется следующее: каким бы ни был набор x из X, x-выходы сопряжённых кустов эквивалентны.*

Докажем теперь лемму 5.3.

Лемма 5.3. *В каждом классе эквивалентных свободных схем из M_0 каноническая форма единственна с точностью до изоморфизма.*

Доказательство. Пусть эквивалентные свободные схемы G_1, G_2 являются каноническими формами в M_0. Покажем, что они изоморфны.

Действительно, рассмотрим для каждого x из X вершины, в которые из входов схем ведут дуги с меткой x. По теореме 5.1 эти вершины эквивалентны. Если это – преобразователи, то их метки должны совпадать, иначе они были бы корнями сопряжённых кустов, а кустов в схемах нет. Аналогичным образом для любой пары сочетаемых и равновеликих маршрутов в схемах G_1, G_2 устанавливается эквивалентность концов маршрутов и совпадение их меток, если концы – преобразователи. Итак, схемы G_1, G_2 изоморфны. Но тогда они, будучи сильно эквивалентными, подобны эквивалентным конечным автоматам, которые являются минимальными, следовательно, совпадают с точностью до изоморфизма.

Лемма 5.3 доказана.

Обратимся теперь к алгоритму, на вход которого поступают свободные эквивалентные схемы G_1, G_2 и который трансформирует каждую из них в каноническую форму, применяя только эквивалентные преобразования их структур. Обозначим его ρ_1. Условимся выявляемые им аксиомы описывать неформально, ссылаясь на [3], где они описаны строго.

В своей работе алгоритм ρ_1 опирается на результаты алгоритма, установившего эквивалентность схем G_1, G_2. Последним составлен список всех пар сопряжённых и разнопомеченных вершин этих схем. Таким образом, для каждой из них известны кусты, вырастающие из вершин, принадлежащих этим парам и подлежащих обработке алгоритмом ρ_1. Она проводится в пять этапов.

Пусть F_0 – обрабатываемый куст и v – его корень. На первом этапе куст F_0 трансформируется в *чистый куст*; так называется куст, в котором все входящие в него извне дуги ведут в корень куста. Для этого применяется безусловная аксиома A1. Пусть в вершину u куста F_0, отличную от вершины v, приходят извне дуги. Тогда рассматривается подфрагмент куста F_0, состоящий из вершин, достижимых из u, создаётся его копия, и она корректно по аксиоме A1 пристраивается к фрагменту F_0, устраняя в нём дуги, ведущие в вершину u. Конечным числом применений аксиомы A1 куст F_0 превращается в чистый куст F_1.

На втором этапе куст F_1 трансформируется в *блок*; так называется чистый куст, в котором последний уровень состоит из одной вершины. Здесь алгоритм ρ_1 использует условную аксиому A2, позволяющую склеивать эквивалентные вершины в схеме, а таковыми, согласно лемме 5.2, являются x-выходы куста F_1 для любого x из X. При склейке по аксиоме A2 одной вершины с другой эквивалентной ей вершиной все дуги, приходящие в первую вершину, перебрасываются на вторую. Конечным числом применений аксиомы A2 куст F_1 превращается в блок; обозначим его F_2.

Но так как при склейке вершин появляются вершины без входящих в них дуг, то после второго этапа преобразуемая схема перестаёт быть свободной. Поэтому алгоритмом ρ_1 проводится третий этап, на котором схеме возвращается свободность. Здесь применяется безусловная аксиома A3, позволяющая устранять из схемы фрагмент, составленный из недостижимых из её входа вершин.

На четвёртом этапе блок F_2 трансформируется в цепь преобразователей; так называется блок, в котором каждый уровень состоит из одной вершины, а цепочка, несомая этой последовательностью вершин, является фиксированной для маркера блока. Преобразование блока F_2 в цепь преобразователей осуществляется по безусловной аксиоме A4.

Описанные четыре этапа выполняются алгоритмом ρ_1 для каждого ранее отмеченного куста в одной и другой схеме, поступившей на его вход. В результате этого исходные схемы G_1, G_2 трансформированы в сильно эквивалентные схемы. На завершающем пятом этапе в каждой из них по условной аксиоме A5 осуществляются склейки сильно эквивалентных вершин, сопровождаемые восстановлением свободности схемы путём применения аксиомы A3. Полученные схемы действительно являются каноническими.

Таким образом, при доказательстве теоремы 5.2 установлено, что система э.п. схем, индуцируемая аксиомами A1 – A5, является полной в модели M_0.

6. Построение аксиом формального исчисления фрагментов для класса K

Рассмотрим класс K из модели программ M, полагая, что индуцирующая M простая модель M_0 – это описанная в предыдущем разделе модель. В целях решения в K проблемы э.п. достроим аксиомами изложенную в разделе 4 часть формального исчисления. Руководствуемся рекомендациями стратегии поиска аксиом.

Назовём *канонической в* K схему, в которой отсутствуют эквивалентные вызовы, а каждая из простых схем, вырастающих из опорных вершин схемы, представлена в канонической форме в M_0.

Введём отношение эквивалентности вызовов в схеме из K. Пусть G – такая схема, и v – вызов в ней. Построим схему $G_1(v)$, принадлежащую K. Её главным подграфом назовём подграф, в котором, кроме положенных входа, выхода и вершины *loop*, имеются всего две вершины: вызов v и парный ему в схеме G возврат u, которые наследуют из G свои метки, и при этом все дуги из входа идут в вершину v, а все дуги из u – в выход. Сохраним имеющиеся в схеме G процедурные подграфы, применяя правило: первым сохраняемым является подграф, в который идёт дуга из вызова v; далее, если в сохраняемом подграфе имеется вызов, то сохранению подлежит подграф, в который из него идёт дуга. Итак, схема $G_1(v)$ построена. Отметим, что она свободна. Говорим, что вызовы v_1, v_2 схемы G эквивалентны, если эквивалентны схемы $G_1(v_1)$, $G_1(v_2)$.

Это отношение эффективно проверяемо, поскольку в модели M разрешима проблема эквивалентности. И так как имеет место алгоритмическая распознаваемость канонической формы в M_0, верно утверждение 6.1.

Утверждение 6.1. *Каноничность схемы в K алгоритмически распознаваема.*

Чтобы установить единственность канонической схемы в своём классе эквивалентных схем из K, введём несколько понятий.

Пусть v_1, v_2 – опорные вершины схемы из K. *Склейкой* первой из них со второй вершиной назовём операцию, состоящую в переброске дуг, приходящих в первую, на вторую вершину. По выполнении её первая вершина становится недостижимой из входа схемы, и поэтому тут же следует операция по удалению недостижимых из входа вершин.

Пусть G – схема из K. Сопоставим ей конечный ориентированный граф, полученный из неё проекцией на её опорные вершины, при которой из опорной вершины идут дуги в её преемники. Назовём этот граф диаграммой схемы. Из леммы 3.1 следует утверждение 6.2.

Утверждение 6.2. *Диаграммы эквивалентных схем из класса K изоморфны.*

Отметим, что после склейки в эквивалентных схемах их эквивалентных вызовов, что сопровождается удалением вершин, недостижимых из входов схем, изоморфизм диаграмм сохраняется. Это заключение основано на лемме 3.1, согласно которой в эквивалентных примитивных схемах, какой бы ни была пара эквивалентных вызовов в одной из них, сочетаемые с ними вызовы в другой схеме тоже эквивалентны.

В итоге справедливо утверждение 6.3.

Утверждение 6.3. *В модели K каноническая схема единственна с точностью до изоморфизма в своём классе эквивалентных схем из K.*

Действительно, пусть G_1, G_2 – эквивалентные схемы, являющиеся каноническими в K. По утверждению 6.2 и сделанному к нему замечанию можно заключить, что их диаграммы изоморфны. Но тогда по лемме 3.1 сами схемы изоморфны.

Теперь приступим к реализации другой рекомендации стратегии построения аксиом.

Теорема 6.1. *Существует алгоритм, который, получив на свой вход две эквивалентные схемы из класса K, путём эквивалентных преобразований их структуры трансформирует их в канонические.*

Доказательство. Опишем алгоритм ρ_2 и докажем, что он – требуемый теоремой.

Сначала алгоритм ρ_2, применяя условную аксиому А6, склеивает в схемах их эквивалентные вызовы. По завершении этих действий обрабатываемые схемы пе-

рестают быть свободными. Поэтому алгоритм ρ_2 применяет безусловную аксиому A7, представляющую собой модификацию на случай схем из K аксиомы A3, описанной в разделе 4, и возвращает им свободу. Далее он для преобразованных схем рассматривает сочетаемые в них опорные вершины, и для каждой из них – пары вырастающих из них и сочетаемых простых схем. В такой паре, согласно лемме 3.1, схемы эквивалентны, следовательно, к ней применим алгоритм ρ_1, который для них определяет сопряжённые кусты и после этого с каждой схемой работает отдельно. И алгоритм ρ_2, выполняя функции алгоритма ρ_1, осуществляет канонизацию каждой из них. Рассматривая все сочетаемые и опорные вершины в постепенно преобразуемых схемах, алгоритм ρ_2 достигает канонизации исходных схем.

Теорема 6.1 доказана.

Вместе с тем установлена справедливость теоремы 6.2.

Теорема 6.2. *Система э.п. схем, индуцируемая аксиомами A1 – A7, полна в классе K.*

Этим завершён раздел 6 и статья в целом.

Список литературы

1. 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)].

2. Подловченко Р.И. Эквивалентные преобразования в математических моделях вычислений: Учебное пособие. М.: МАКС-Пресс, 2011. 72 с. [Podlovchenko R.I. Ekvivalentnie preobrazovanija v matematicheskih modelah vichislenij: Uchebnoe posobie. M.: MAKS-Press, 2011. 72 s. (in Russian)].

3. 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).

4. 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]).

5. 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)].

6. 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).

7. 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)].

8. 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)].

9. 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)].

10. 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)].

11. 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)].

12. 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. S. 39–48 (in Russian)].

13. 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 s. (in Russian)].

14. 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. S. 13–22 (in Russian)].

15. 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. S. 38–46 (in Russian)].

16. 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)].

17. 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. S. 275–295 (in Russian)].

18. 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. S. 47–56 (in Russian)].

19. 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. S. 47–56 (in Russian)].

20. 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.

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

22. 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.

23. De Bakker J.W., Scott D.A. Theory of programs. Unpublished notes // Vienna: IBM seminar, 1969.

24. Valiant L.G. The equivalence problem for deterministic finite-turn push-down automata // Information and Control. 1974. V. 25, No. 2. P. 123–133.

25. Nielson F., Nielson H.R., Hankin C. Principles of Program Analysis // Springer, 2004.

Primitive Program Schemes with Procedures

Podlovchenko R.I.

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

Keywords: algebraic program model, program schemes equivalent transformations

The paper considers algebraic program models with procedures designed to analyze program semantic properties based on program schemes. This leads to the problem of program scheme equivalence and the problem of constructing a complete system of equivalent program scheme transformations. Among algebraic program models with procedures, we focus on gateway models induced by program models without procedures and primitive program schemes belonging to these gateway models. The equivalence problem is decidable for these schemes. In the case where the inducer is a special type of program model without procedures, we construct a complete system of equivalent scheme transformations for a particular subclass of primitive program schemes.

Сведения об авторе:
Подловченко Римма Ивановна,
МГУ им. М.В. Ломоносова,
проф., д-р физ.-мат. наук