Разрешимость проблемы эквивалентных преобразований в классе примитивных схем программ

А. Э. Молчанов <gurux13@gmail.com>
Московский государственный университет, факультет вычислительной математики и кибернетики, 119333 Москва, Ленинские Горы, 1с52.

Аннотация. Статья принадлежит теории схем программ. Схемы программ - это объекты, созданные для анализа формализованных программ. Одной из основных задач теории является создание полных конечных систем эквивалентных преобразований (Э.П.). В статье рассматриваются схемы программ с процедурами, с ограничением на перегородчатые модели. Такие модели тесно связаны с моделями программ без процедур. Даётся краткое описание систем Э.П., и описывается методология решения проблемы Э.П. Выделяется подкласс перегородчатых моделей программ, называемый примитивными моделями. Они находятся ещё ближе к моделям без процедур. Указанная методология успешно применяется для построения полной конечной системы Э.П. в примитивном подклассе перегородчатых уравновешенных полугрупповых моделей программ с левым сокращением. Этот класс является широко изученным классом моделей программ, и при построении системы Э.П. используется известная конечная полная система Э.П. для похожих моделей без процедур. Используется вспомогательный вид схем, называемый многовыходными. В результате, строится канонический представитель класса эквивалентности схем программ, а также последовательность преобразований, приводящая произвольную схему этой модели в её каноническую форму. Такой способ решения считается полным решением проблемы Э.П. в классе моделей программ. В заключение приводятся направления для дальнейшего исследования.
Ключевые слова: алгебраические модели программ с процедурами; система эквивалентных преобразований; уравновешенная модель программ с левым сокращением**.**

DOI: 10.15514/ISPRAS-2015-27(2)-11

Для цитирования: Молчанов А.Э. Разрешимость проблемы эквивалентных преобразований в классе примитивных схем программ. Труды ИСП РАН, том 27, вып. 2, 2015 г., стр. 173-188. DOI: 10.15514/ISPRAS-2015-27(2)-11.

1. Введение

Статья относится к теории алгебраических моделей программ, предназначенной для разработки эквивалентных преобразований программ на создаваемых для них схемах программ, и обобщает результат, полученный в [1].

Концепции, на основе которых строится теория, подробно изложены в [2]. Коротко перечислим их.

Программы берутся изначально формализованными. Схема программы по своей структуре совпадает с программой, для которой она создаётся. Отсюда всякое преобразование схемы одновременно является и преобразованием самой программы.

Моделью называется множество схем, построенных над некоторым базисом операторов, с введённым в нем отношением эквивалентности схем. Последнее вводится параметрически, что приводит к иерархии моделей. В теории рассматриваются только аппроксимирующие модели, т.е. такие, каждая из которых обладает свойством: из эквивалентности схем в модели следует эквивалентность программ, принадлежащих некоторому их классу, и совпадающих по структуре со схемами. В [2] описаны условия, налагаемые на параметры модели, при выполнении которых она является аппроксимирующей.

Обратимся теперь к вопросу о том, какое место в теории алгебраических моделей программ занимает рассматриваемая в статье задача.

Изучаемые модели подразделяются на модели программ, не использующих процедур, и модели программ, использующих их. Первые называются простыми, вторыми они обобщаются. Ключевой является проблема эквивалентных преобразований (э.п.) схем в модели. Она состоит в поиске системы э.п. схем, обладающей свойством: какими бы ни были две эквивалентные схемы из этой модели, конечной цепочкой преобразований, принадлежащих системе, одна из схем алгоритмически транслируется в другую. Естественно, что исследование проблемы э.п. проводятся в моделях с разрешимой эквивалентностью схем.

Первоначально рассматривались простые модели, и для них получен богатый арсенал результатов как по проблеме эквивалентности в модели, так и по проблеме э.п. в модели. Возникла задача – использовать эти результаты в моделях программ с процедурами.

В этих целях в [3] среди последних выделены так называемые перегородчатые модели программ. Их характерной особенностью является то, что параметры такой модели индуцируются параметрами её простой подмодели. Существенно и то, что если простая подмодель – аппроксимирующая, то и индуцируемая ей модель тоже аппроксимирующая.

В [4] формулируется условие, когда проблема эквивалентности в перегородчатой модели сводится в её свободную подмодель; так называется множество принадлежащих модели схем, в которых нет бездействующих элементом (схемы этого множества называются свободными).

Далее в [4] описан класс так называемых примитивных схем, принадлежащих свободной подмодели, и получен следующий результат: эквивалентность в этом классе разрешима, если она разрешима в индуцирующей его простой модели.

Рассматриваемая нами задача состоит в следующем: выбрав в качестве индуцирующей простую модель, называемую уравновешенной полугрупповой моделью программ с левым сокращением (она рассматривалась в [5]), решить проблему э.п. в классе примитивных схем программ. Отметим, что выбранная простая модель является аппроксимирующей, и в ней разрешимы проблема эквивалентности и проблема э.п.

В [1] требуемый результат получен для подкласса примитивных схем, обсуждающихся в данной статье. Это полностью определяет план её изложения.

В разделе 2 описывается общий вид алгебраической модели программ с процедурами и их частный случай – перегородчатые модели.

Примитивные схемы программ определены в разделе 3. Здесь же формулируются факты, установленные для них в [4], и описываются свойства, присущие эквивалентным примитивным схемам.

Раздел 4 посвящён методу решения проблемы э.п. в алгебраических моделях программ – воспроизводится его изложение, данное в [1].

В разделе 5 даётся определение используемой нами уравновешенной полугрупповой модели программ с левым сокращением, формулируются полученные в [5] результаты и описываются свойства принадлежащих им эквивалентных схем.

Построение системы э.п., полной в классе примитивных схем, индуцируемых уравновешенными полугрупповыми моделями с левым сокращением, осуществляется в разделе 6. Описываются канонические формы примитивных схем и алгоритм, транслирующий эквивалентные примитивные схемы в каноническую форму. При изложении алгоритма выделяются э.п., входящие в полную систему. Они описываются содержательно со ссылкой на их формальное определение в [5].

2. Рассматриваемые алгебраические модели программ

2.1 Общего вида алгебраические модели программ с процедурами

Модели программ с процедурами строятся над четырьмя конечными и непересекающимися алфавитами Y, C, R и X. Элементы первых трёх алфавитов называются символами операторов, вызовов и возвратов соответственно, элементы алфавита X именуются наборами. Алфавиты Y, C, R, X называются базисом.

Элементы модели называются схемами программ.

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.

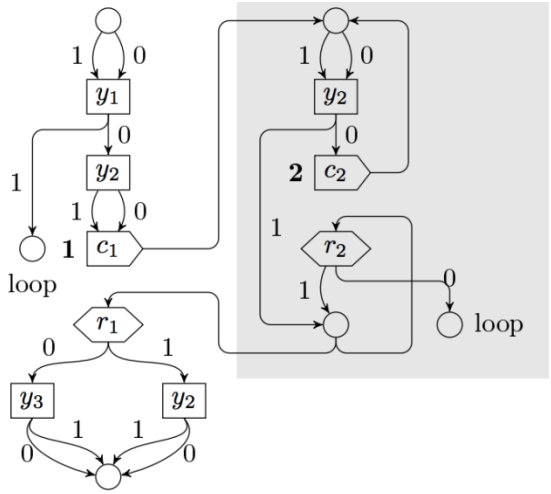

Рис. 1. Пример схемы программы

На рис. 1 приведён пример схемы программы. Здесь $\{y_1, y_2, y_3\} \subseteq Y, \{c_1, c_2\} \subseteq C, \{r_1, r_2\} \subseteq R, X = \{0,1\}$. 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 L.

Пусть $\mu \in L$. *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 берётся дуга, несущая метку $\mu(h)$, где h – построенная к этому моменту 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.

Пусть ν – эквивалентность в множестве H, и $L \subseteq L$. Схемы G_1, G_2 назовём (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 с результатом одна из схем, останавливается с результатом 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 (ν, L)-*моделью*, а ν, L – её *параметрами*.

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

2.2 Перегородчатые модели программ

Определение перегородчатой модели программ в точности соответствует данному в [2].

Перегородчатая модель программ с процедурами представляет собой тот частный случай модели над базисом Y, C, R, X, 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 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.

Легко доказуемо следующее утверждение.

Утверждение 1 *Перегородчатая модель программ является аппроксимирующей, если этим свойством обладает индуцирующая её простая модель.*

Затронем проблему либеризации для перегородчатой модели программ. Схема из неё называется *свободной*, если в ней любая вершина, кроме $loop$, принадлежит реализуемому маршруту через схему.

Пусть G – схема из рассматриваемой модели. *Маршрутом* в G назовём ориентированный путь w, начинающийся во входе схемы и заканчивающийся в некоторой вершине. Если последняя – это выход схемы G, то w именуем *маршрутом через схему*.

Маршрут в G назовём *реализуемым*, если в L существует функция разметки, которая его прокладывает при выполнении на ней схемы G.

Проблема либеризации – поиск алгоритма, который, получив на свой вход схему из рассматриваемой модели, трансформирует её в эквивалентную ей свободную схему из той же модели.

В [2] доказана

Теорема 1 *Проблема либеризации разрешима в перегородчатой модели программ, если она разрешима в индуцирующей её простой модели.*

Перегородчатая модель называется *специальной*, если она индуцируется простой моделью с разрешимой проблемой либеризации.

В [2] доказана

Теорема 2 *В специальной перегородчатой модели программ разрешима проблема либеризации.*

Назовём множество всех свободных схем в специальной перегородчатой модели программ её *свободной подмоделью*.

3. Примитивные схемы программ

Обозначим M свободную подмодель специальной перегородчатой модели над базисом Y, X, C, R; M_0 – индуцирующую её простую модель с параметрами τ, l.

Пусть G – 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.

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 [4] установлен факт, следствием которого является теорема 3, самостоятельно доказанная [1].

Теорема 3 *Если в модели M_0 разрешима проблема эквивалентности схем, то она разрешима в классе примитивных схем из M.*

Опираясь на доказательство теоремы 3, приведённое в [1], опишем необходимые признаки эквивалентности примитивных схем.

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. Сначала в ней оставляются все вершины, принадлежащие ориентированным путям из v в u. Если v – вход схемы G, то он объявляется входом создаваемой схемы, если u – выход схемы G, то он становится выходом создаваемой схемы. 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.

Маршруты в двух схемах из M назовём *сочетаемыми*, если они прокладываются при выполнении схем на общей и допустимой функции разметки. В эквивалентных примитивных схемах их опорные вершины назовём *сочетаемыми*, если ими оканчиваются сочетаемые маршруты, которые несут цепочки с равными проекциями на $C \cup R$. *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 установлен факт, формулируемый здесь как лемма 1.

Лемма 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.*

Всякой опорной вершине v схемы из M сопоставим так называемую *многовыходную схему* $\hat{G}(v)$, построенную над базисом Y, X, D, где D – конечный алфавит, который будет описан ниже.

Многовыходная схема по своей конструкции отличается от простой схемы из M_0 только тем, что в ней может быть больше одной вершины без исходящих из неё дуг и называемых ей выходами; при этом выходы помечаются символами из D без повторений.

Такая схема выполняется на функции разметки, допустимой для M_0, по тем же правилам, что и простая схема из M_0. Если выполнение завершается в одном из выходов схемы, и только в этом случае, оно признаётся результативным, причём результатом объявляется цепочка, несомая проложенным маршрутом и завершаемая символом, сопоставленным достигнутого выхода.

Эквивалентными, по определению, являются две схемы, удовлетворяющие требованию: какой бы ни была допустимая для M_0 функция разметки, если выполнение на ней одной из схем результативно, то результативно и выполнение другой, и при этом результаты оканчиваются общим для них символом, а в остальном это τ-эквивалентные цепочки.

Для опорной вершины v схемы из M сопоставляемая ей многовыходная схема строится объединением в один граф всех простых схем, вырастающих из v, сопровождаемая приписыванием выходу схемы метки преемника, для которого она построена; если же преемником является выход самой схемы из M, то в многовыходной схеме он помечается специальным символом y_0. Таким образом, в качестве алфавита D берётся сумма C и R, дополняемая этим

специальным символом. Будем говорить, что построенная многовыходная схема вырастает из v.

В [1] установлена справедливость утверждения 2

Утверждение 2 *Какими бы ни были сочетаемые опорные вершины эквивалентных приведённых схем, вырастающие из них многовыходные схемы эквивалентны.*

4. Методика решения проблемы э.п. в алгебраической модели программ

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.

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.

Определения фрагмента, его вхождения в схему, согласованных фрагментов и подстановки фрагмента подробно представлены в [1]. Приведём здесь лишь содержательное описание.

Фрагментом называется часть схемы, для которой указана связь с остальной частью схемы.

Вхождение фрагмента F в схему G – это фрагмент схемы G, изоморфный F.

Согласованность фрагментов обеспечивает корректную выполнимость правила подстановки вместо одного фрагмента согласованного с ним; корректность трактуется так: результатом подстановки является фрагмент, в частности, схема, если подставляемый фрагмент входил в неё.

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 отметим, что в разделе 6 данной статьи изучается решение проблемы э.п. в классе примитивных схем в случае, когда индуцирующая модель M_0 – это уравновешенная полугрупповая модель программ с левым сокращением.

5. Уравновешенные полугрупповые модели программ с левым сокращением

Описывая эти модели, мы фактически даём аналитический обзор статьи [5].

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

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

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

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

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

Множество l состоит из всех τ-согласованных функций разметки.

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 4 и теорема 5.

Теорема 4 *В модели M_0 разрешима проблема эквивалентности схем.*

Теорема 5 *В модели M_0 разрешима проблема э.п. схем.*

Полагаем схемы в M_0 свободными.

Остановимся на описании необходимых свойств эквивалентности схем из модели M_0, формулируя их как утверждения 3-5

Утверждение 3 *Равновеликие сочетаемые маршруты в эквивалентных схемах оканчиваются в вершинах общего типа.*

Утверждение 4 *Из сопряжённых и различно помеченных вершин эквивалентных схем вырастают кусты общего маркера.*

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 $i, i = 0,1, \dots, k-1$, ведут в вершины $i+1$-ого уровня; таким образом, из вершины v на k-ый 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*.

Здесь *сопряжёнными* в двух схемах называются преобразователи, в которых завершаются сочетаемые маршруты, несущие цепочки, которые являются τ-эквивалентными по устранении из них последнего символа.

Кусты, о которых говорится в утверждении 4, также называются *сопряжёнными*.

Утверждение 5 *В эквивалентных схемах для любого набора $x \in X$ x–выходы сопряжённых кустов эквивалентны.*

По определению, *x–выходом куста* в схеме называется вершина, в которую из последнего уровня куста ведёт дуга с меткой x; $x \in X$.

Введём отношение эквивалентности вершин в схеме из M_0.

Любой вершине v схемы G из M_0 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 схемы G из M_0 назовём *эквивалентными*, если эквивалентны схемы $G(v_1), G(v_2)$.

Отметим, что доказательство теоремы 4 выполнимо проверкой утверждений 3-5 при просмотре пар сопряжённых вершин испытуемых схем.

6. Решение проблемы э.п. в классе примитивных схем, индуцированных моделью M_0

Следуя методике, изложенной в разделе 4, определим каноническую схему в классе K примитивных схем, индуцированных моделью M_0.

Схему G из K назовём *канонической*, если выполнены условия:

- в G отсутствуют эквивалентные опорные вершины;
- какой бы ни была опорная вершина v схемы G, многовыходная схема $\hat{G}(v)$ не содержит кустов и сильно эквивалентных вершин;
- для разных опорных вершин v_1, v_2 многовыходные схемы $\hat{G}(v_1), \hat{G}(v_2)$ не имеют общих вершин, кроме опорных.

Определим используемые здесь понятия. Пусть G – схема из K.

Опорной вершине v, являющейся вызовом, сопоставим схему $\nabla G(v)$. Эта схема строится следующим образом. Главный подграф G удаляется и заменяется на главный подграф следующего вида. Он содержит входную и выходную вершины, все дуги из входной вершины ведут в единственный вызов, который помечен тем же символом, что и v, и инцидентен тому же

подграфу, что и v. Единственный возврат, принадлежащий главному подграфу, парен этому вызову, и все дуги из него ведут в выход. Других вершин в главном подграфе нет. В остальном схема $\nabla G(v)$ совпадает с G.

Вершины-вызовы v_1, v_2 схемы G назовём *эквивалентными*, если эквивалентны схемы $\nabla G(v_1), \nabla G(v_2)$.

Схемы G_1, G_2 из M_0 назовём *сильно эквивалентными*, если они эквивалентны, и сочетаемые маршруты через них несут равные цепочки.

Вершины v_1, v_2 схемы G из M_0 назовём *сильно эквивалентными*, если сильно эквивалентны схемы $G(v_1), G(v_2)$.

Лемма 2 *Каноническая схема из K алгоритмически распознаваема и единственна с точностью до изоморфизма в своём классе эквивалентности.*

Proof. Очевидно, что в силу разрешимости отношений эквивалентности и сильной эквивалентности, свойства 1-2 канонической схемы алгоритмически распознаваемы.

Свойство 3 распознаваемо обходом графа схемы из каждой опорной вершины. Следовательно, каноническая форма является алгоритмически распознаваемой.

Пусть существует две канонические формы $G_1, G_2, G_1 \sim G_2$, не являющиеся изоморфными. Пусть в G_1 существует вершина v, не являющаяся опорной, которая не имеет сопряжённой в G_2. В этом случае, в силу свободы схем, они не эквивалентны. Если вершина v схемы G_1 сопряжена более чем с одной вершиной G_2, то эти вершины в G_2 эквивалентны (тогда G_2 – не каноническая форма), либо принадлежат различным схемам вида $\widehat{G_2}(v_1), \widehat{G_2}(v_2)$ (тогда G_1 не является канонической формой, поскольку содержит пересекающиеся схемы, вырастающие из различных опорных вершин).

Если же схемы G_1, G_2 отличаются опорными вершинами, то они неэквивалентны в силу примитивности схем.

Лемма доказана.

Теперь опишем алгоритм ρ, который, получив на свой вход две эквивалентные схемы G_1, G_2, строит эквивалентную им каноническую схему G.

Сначала алгоритм ρ в каждой из схем G_1, G_2 просматривает пары одинаково помеченных вызовов и устанавливает, эквивалентны они или нет. В случае эквивалентности алгоритм ρ выполняет склеивание этих вершины путём переброса на оставляемую вершину дуг, приходящих в другую.

Это преобразование поддерживается условной аксиомой $B1$, формальное определение которой совпадает с определением условной аксиомы $A6$ в [5]. Здесь и далее мы будем ссылаться на аксиомы из [5], определяющие эквивалентные преобразования системы э.п., полной в M_0, ограничиваясь лишь содержательным описанием самих преобразований.

После применения аксиомы $B1$ схема перестаёт быть свободной, и поэтому применяется преобразование, описываемое безусловной аксиомой $B2$. Оно

состоит в удалении из схемы вершин, не достижимых из её входа; это преобразование обобщает описанное в [5] аксиомой $A7$.

Обозначим $G_{1,}, G_{2,}$ схемы, эквивалентные исходным схемам G_1, G_2, и не содержащие эквивалентных вызовов.

Далее, алгоритм ρ работает с каждой парой многовыходных схем, вырастающих из сочетаемых опорных вершин схем $G_{1,}, G_{2,}$. При этом, если существует преобразователи, принадлежащие нескольким многовыходным схемам, они *расклеиваются*, то есть, создаются копии преобразователей так, чтобы каждая копия принадлежала своей многовыходной схеме. Это преобразование поддерживается безусловной аксиомой $A1$ из [5].

Обозначим ρ_1 выполняемый для каждой пары многовыходных схем алгоритм.

Пусть v_1, v_2 – рассматриваемая пара вершин схем $G_{1,}, G_{2,}$ соответственно. Поскольку многовыходные схемы $\widehat{G_{1,}}(v_1)$ и $\widehat{G_{2,}}(v_2)$ эквивалентны, то известны принадлежащие им сопряжённые вершины, следовательно, и сопряжённые кусты в них.

На основании утверждения 5 справедлива

Лемма 3 *Если F_1, F_2 – сочетаемые кусты из схем $\widehat{G_{1,}}(v_1)$ и $\widehat{G_{2,}}(v_2)$, то при любом $x \in X$ все x–выходы этих кустов находятся в сочетаемых схемах, вырастающих из v_1, v_2 соответственно.*

Для каждой пары сочетаемых кустов проводится трансформация, подробно описанная в [1]. Приведём её кратко.

В первую очередь, сопряжённые кусты переводятся в *чистые* кусты, то есть, кусты, в которых все входящие извне дуги ведут в корень куста. Эта трансформация проводится путём *отклеивания* подкуста с формированием новой пары сопряжённых кустов. Это преобразование обеспечивается аксиомой $A1$.

Далее обеспечивается следующее свойство сопряжённых кустов: для каждого $x \in X$ все x–выходы куста ведут в одну и ту же вершину. Это требование выполняется путём склеивания x–выходов кустов при помощи аксиомы $B1$.

После этого кусты трансформируется в *линейные*, то есть, просто в цепочки преобразователей, несущие операторные символы, соответствующие маркерам кустов. Преобразование проводится при помощи безусловной аксиомы $A7$ из [5].

В процессе преобразований могли получаться несвободные схемы, поэтому алгоритм повторяет трансформацию схем в свободные согласно аксиоме $B2$.

В результате имеет место следующее утверждение.

Утверждение 6 *Преобразованные многовыходные схемы сильно эквивалентны.*

Здесь сильно эквивалентными называются эквивалентные схемы, составными частями которых являются сильно эквивалентные простые схемы.

Осталось применить аксиому $A6$, склеивающую сильно эквивалентные вершины.

Алгоритм ρ_1 описан.

Применением алгоритма ρ пара схем приводится к паре изоморфных канонических схем, следовательно, верна

Теорема 6 *Система э.п., индуцируемая аксиомами*

$$B1, B2, A1 - A7,$$

является полной в классе K.

7. Заключение

Дальнейшие исследования в этой области можно вести в следующих направлениях:

- исследование других простых моделей программ, в которых разрешима проблема эквивалентных преобразований, для применения решения к многовыходным схемам и, соответственно, к перегородчатым моделям программ;
- поиск представлений схем из перегородчатых моделей, которые могут быть альтернативой многовыходным схемам; результаты в этой области помогут решать проблему эквивалентных преобразований для схем, которые не являются примитивными;
- исследование многовыходных схем как самостоятельного объекта.

Список литературы

[1]. Подловченко Р.И. Исследование примитивных схем программ с процедурами // Моделирование и анализ информационных систем, т. 21, 4, 2014, с. 116–131.

[2]. Подловченко Р.И., Молчанов А.Э. О теории алгебраических моделей программ с процедурами // Моделирование и анализ информационных систем, т.19, №5, 2012, с.100–114.

[3]. Подловченко Р.И. Специальные перегородчато-автоматные модели рекурсивных программ // Программирование, 1994, №3, с.3 - 26.

[4]. Молчанов А.Э. Разрешимость эквивалентности в двухпараметрических перегородчатых моделях программ // Моделирование и анализ информационных систем, т.21, №4, 2014, с. 104-115.

[5]. Подловченко Р.И. Полные системы эквивалентных преобразований в уравновешенных полугрупповых моделях программ с левым сокращением // Программирование, 2010, № 3, с. 3-18.

A Solution to the Equivalent Transformation Problem in a Class of Primitive Program Schemes

A. Molchanov <gurux13@gmail.com>

Faculty of Computatational Mathematics and Cybernetics of Moscow State University, 1b52, Leninskie Gory str., Moscow, 119333, Russian Federation

Abstract. The article belongs to the theory of program schemes. Program schemes are objects designed for the analysis of formalized programs. One of the key tasks of the theory is to create a finite full system of equivalent transformations (E.T.). The article deals with program schemes with procedures, limited only to gateway program models. Such models derive highly from program models without procedures. A brief description of E.T. systems is given, followed by a methodology for the construction of full systems of E.T. A subclass of gateway program models, called primitive schemes, is introduced. Such schemes are closer to procedure-free schemes. The methodology is then successfully applied to construct a full finite system of E.T. in the primitive subclass of balanced gateway semigroup program models with left cancellation, which is a well-studied class of program models. The construction is based on the known system of E.T. for similar program models without procedures. An auxiliary type of schemes, called multiexit schemes, is used. As a result, canonical schemes for classes of equivalence within the models are defined, and a sequence of transformations resulting in the transition of any scheme from this model into its canonical form is stated. This is considered a complete E.T. problem solution for a class of programs. Further research topics are given in the conclusion.

Keywords: algebraic program models; equivalent transformations; gateway program models; primitive program models; left cancellation.

DOI: 10.15514/ISPRAS-2015-27(2)-11

For citation: Molchanov A. A Solution to the Equivalent Transformation Problem in a Class of Primitive Program Schemes. *Trudy ISP RAN/Proc. ISP RAS*, vol. 27, issue 2, 2015, pp. 173-188 (in Russian). DOI: 10.15514/ISPRAS-2015-27(2)-11.

References

[1]. Podlovchenko R.I. Primitive program schemata with procedures. *Automatic Control and Computer Sciences, Allerton Press*, 2014, vol. 48, N 7, pp. 615-622.

[2]. Podlovchenko R.I., Molchanov A.E. About algebraic program models with procedures. *Automatic Control and Computer Sciences, Allerton Press*, 2013, vol. 47, N 7, pp. 385-392.

[3]. Podlovchenko R.I. [Special gateway-automaton models of recursive programs]. *Programmirovanie [Programming],* 1994, No. 3, pp. 3-26 (in Russian).

[4]. Molchanov A.E. [Equivalence Problem Solvability in Biparametric Gateway Program Models]. *Modelirovanie I analiz informacionnikh sistem [Information systems analysis and modelling],* 2014, vol.21, No. 4, pp. 104-115 (in Russian).

[5]. Podlovchenko R.I. Complete Systems of Equivalent Transformations in Balanced Semigroup Models of Programs with Left Cancellation. *Programming and Computer Software, 2010, vol. 36, No. 3, pp. 125-137*