Модел. и анализ информ. систем. Т. **19**, № **5** (2012) 100–114

УДК 681.3.06

О теории алгебраических моделей программ с процедурами

Подловченко Р. И., Молчанов А. Э.

Московский государственный университет им. М.В. Ломоносова

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

получена 11 октября 2012

Ключевые слова: формализация программы, схема программы, эквивалентность схем программ, матричная схема, свободная схема

Алгебраические модели программ с процедурами предназначены для изучения семантических свойств самих программ на их образах - схемах программ. Излагаются концепции, лежащие в основе построения теории таких моделей, и описывается, как они реализуются. Центральное место в теории отводится проблеме эквивалентности схем программ, принадлежащих отдельной модели. Рассматривается класс специального вида алгебраических моделей программ с процедурами, называемых перегородчатыми моделями, и устанавливаются необходимые и достаточные условия разрешимости в этих моделях проблемы эквивалентности.

1. Введение

Статья принадлежит разделу теории схем программ. Назначение этой теории – изучать семантические свойства программ на их моделях – схемах программ.
Теория алгебраических моделей программ восходит к работам А.А. Ляпунова [1] и Ю.И. Янова [2]. Изложим концепции, на основе которых она строится:

- Исходной является формализация понятия программы. Формализованные программы конструируются над конечным базисом операторов и предикатов, индуцирующих семантику базиса. Выбранная семантика определяет класс программ, в котором программам приписываются вычисляемые ими функции. Эквивалентными считаются программы, вычисляющие одинаковые функции.

- Для изучения семантических свойств формализованных программ, направленных на построение эквивалентных преобразований (э.п.) структур программ, вводятся модели программ. Их объекты называются схемами программ. Они строятся над базисом, состоящим из символов операторов и обозначений логических переменных, заменивших собой предикаты. Структура

схемы берется совпадающей со структурой программы, для которой построена схема. Таким образом обеспечивается ситуация: всякое структурное преобразование схемы одновременно является и структурным преобразованием программы.

- Саму алгебраическую модель программ определяет выбор отношения эквивалентности в множестве схем программ над данным базисом. Такое отношение вводится посредством выбора двух параметров будущей модели. Первым фиксируется отношение эквивалентности цепочек операторных символов, вторым – допустимые ситуации, в которых выполняются схемы.

  В множестве моделей учреждается отношение: одна модель не слабее другой, если из эквивалентности схем в первой следует эквивалентность их во второй.

- Осуществляется отбор моделей, "полезных" для изучения. Говорится, что модель программ является *аппроксимирующей*, если для неё можно указать класс программ (над тем же базисом) такой, что из эквивалентности схем в модели следует эквивалентность программ в этом классе, по структуре совпадающих со схемами. Поскольку в аппроксимирующей модели всякое э.п. схемы является и э.п. программы, для которой построена схема, рассматриваются только аппроксимирующие модели. Для этого находятся условия, налагаемые на параметры модели и достаточные для того, чтобы она была аппроксимирующей.

- Ставится проблема построения в модели полной в ней системы э.п. – проблема э.п.; полнота системы трактуется так: какими бы ни были две эквивалентные схемы из данной модели, существует конечная цепочка э.п., принадлежащих этой системе, трансформирующая одну схему в другую. Очевидно, что решение проблемы э.п. в аппроксимирующей модели дает самое богатое множество э.п. программ в аппроксимируемом классе программ.

- Первоочередной становится проблема эквивалентности схем в модели. Она состоит в поиске алгоритма, который, получив на свой вход две схемы из этой модели, распознает, эквивалентны они или нет.

Концепции изложены.

Развитие теории алгебраических моделей программ в основном шло по руслу, когда при формализации программы использовались все композиции базисных операторов, принятые в алгоритмических языках программирования, кроме аппарата процедур. Конструируемые при этом модели именуются *простыми*. К настоящему времени получен большой арсенал результатов по решению проблемы эквивалентности и проблемы э.п. в простых моделях программ. Разработаны две методики распознавания эквивалентности – в [3] и в [4]. Описана методика решения проблемы э.п. [5].

Естественным направлением дальнейшего развития теории алгебраических моделей программ является включение аппарата процедур в композиции базисных операторов и предикатов, учитываемых при формализации понятия программы.

Так появляются алгебраические модели программ с процедурами – предмет рассмотрений в данной статье.

Кроме введения, статья содержит разделы 2, 3, 4.

В разделе 2 описывается расширенная формализация программы, т.е. определяется структура формализованной программы, описывается её семантика, вводится отношение эквивалентности программ. Выбранным базисом операторов и предикатов определяется свой класс формализованных программ.

Раздел 3 посвящен описанию алгебраических моделей программ с процедурами, т.е. для выбранного базиса символов операторов и логических переменных даются структура и семантика схемы программы с процедурами, и вводится множество моделей программ. Здесь же формулируются условия, при выполнении которых модель является аппроксимирующей. Дается определение чистой матричной схемы и доказывается, что в любой модели программ любую схему можно преобразовать в эквивалентную ей чистую матричную схему.

В разделе 4 рассматриваются специального вида алгебраические модели программ с процедурами; они названы перегородчатыми. Такая модель индуцируется некоторой простой моделью и содержит её как подмодель. Интерес к перегородчатым моделям обусловлен возможностью использовать при их изучении результаты исследований, полученные для простых моделей.

В данной статье для определённого класса перегородчатых моделей программ получены следующие результаты:

- найдено решение проблемы трансформации произвольной схемы программ, принадлежащей модели, в эквивалентную ей свободную схему, то есть схему, не содержащую нефункционирующих элементов;
- установлено, что проблема эквивалентности схем имеет решение, если решена проблема эквивалентности в той простой модели, которая индуцирует изучаемую перегородчатую модель.

Изложенные факты являются новыми в теории алгебраических моделей программ.

2. Формализованные программы с процедурами

Формализованная программа с процедурами (далее – просто программа) строится над конечным базисом, состоящим из элементов четырех непустых и непересекающихся алфавитов – Y, C, R и P. Элементы первых трех называются символами, элементы множества P – логическими переменными; каждая логическая переменная принимает значения из множества $\{0, 1\}$. Символами обозначаются операторы, логическими переменными – булевы выражения.

Программа представляет собой размеченный над базисом конечный ориентированный граф следующего строения. Граф распадается на подграфы с непересекающимися множествами вершин. Один из подграфов называется *главным*; в нем выделены *вход* – вершина без входящих в нее дуг и одной исходящей, *выход* – вершина без исходящих из нее дуг. В любом неглавном подграфе тоже выделены две

вершины – *инициальная* и *финальная*, из инициальной вершины исходит ровно одна дуга. Всякая вершина, кроме входа, выхода, инициальных и финальных, принадлежит одному из четырех типов – *преобразователь*, *распознаватель*, *вызов*, *возврат*. Из распознавателя исходят две дуги, помеченные числами 0 и 1 соответственно; из преобразователя, вызова, возврата – по одной дуге. Распознавателю сопоставлена переменная из P, преобразователю, вызову, возврату – символ из Y, C, R соответственно. Вызовы и возвраты находятся во взаимно–однозначном соответствии. Вызов и соответствующий ему возврат составляют пару, принадлежащую одному и тому же подграфу. Всякой такой паре присвоен номер, отличный от номеров других пар. Дуга из вызова ведет в инициальную вершину своего или чужого подграфа, и тогда из финальной вершины этого подграфа исходит дуга, ведущая в соответствующий вызову возврат, и она является единственной приходящей в него дугой. Из финальной вершины иных дуг, кроме упомянутых, не исходит. Дуги, исходящие из вершин иного типа, ведут в вершины того же подграфа, которому принадлежит их начало.

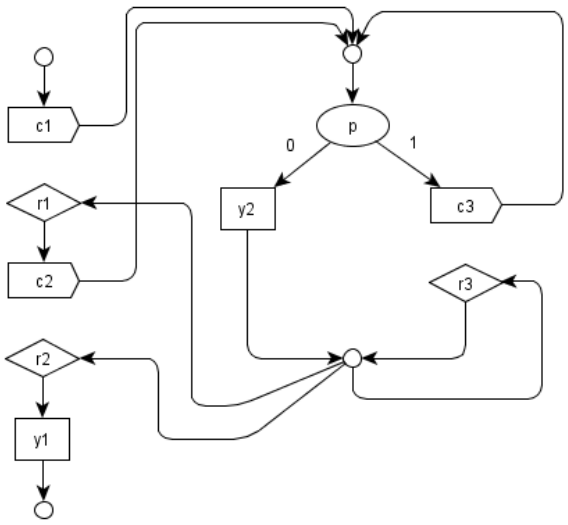

Рис. 1. Пример программы

На рис. 1 представлена программа, построенная над базисом

$$Y = \{y1, y2\}, C = \{c1, c2, c3\}, R = \{r1, r2, r3\}, P = \{p\}.$$

Она состоит из ведущей программы, носителем которой является главный подграф, и процедуры, носителем которой является неглавный подграф.

Функциональная трактовка программы основана на *семантике базиса*. Так называется алгебраическая система σ, использующая произвольно выбранное множество Ξ_σ и сопоставляющая каждому элементу b базиса всюду определенное отображение σb, тип которого определяется следующим образом:

$$\sigma b : \begin{cases} \Xi_\sigma \to \Xi_\sigma, & \text{если } b \in Y \cup C; \\ \Xi_\sigma \times \Xi_\sigma \to \Xi_\sigma, & \text{если } b \in R; \\ \Xi_\sigma \to \{0, 1\}, & \text{если } b \in P. \end{cases}$$

Элементы множества Ξ_σ называются *состояниями памяти* (без предъявления таковой). Множество всех семантик базиса обозначается Σ. Опишем теперь процедуру выполнения программы на паре (σ, ξ_0), где $\sigma \in \Sigma, \xi_0 \in \Xi_\sigma$. Процедура использует магазин, заполняемый парами (n, ξ), где n – натуральное число, ξ – состояние из Ξ_σ, и состоит в путешествии по программе, маршрут которого определяется последовательностью проходимых дуг. Путешествие сопровождается преобразованием состояния памяти и содержимого магазина.

Путешествие по программе начинается по дуге, исходящей из ее входа, при состоянии памяти ξ_0 и пустом магазине и подчиняется следующим правилам. Пусть в некоторый момент времени путешествие вывело на дугу, ведущую в вершину v, и ξ – текущее состояние памяти. Различаем случаи:

1. v – преобразователь с символом y; тогда состояние памяти ξ преобразуется в состояние $\sigma y(\xi)$, содержимое магазина остается прежним, и путешествие продолжается по дуге, исходящей из v;

2. v – распознаватель с символом p; тогда состояние памяти ξ и содержимое магазина не изменяются, и путешествие продолжается по дуге, исходящей из v и помеченной значением $\sigma p(\xi)$;

3. v – вызов с символом c и номером n; тогда в магазин заносится пара (n, ξ), состояние памяти ξ преобразуется в состояние $\sigma c(\xi)$, и путешествие продолжается по дуге, исходящей из v;

4. v – инициальная вершина; тогда состояние памяти ξ и содержимое магазина не изменяются, и путешествие продолжается по дуге, исходящей из v;

5. v – финальная вершина; в этом случае магазин заведомо не пуст: тогда из него снимается верхний элемент, пусть это – (n', ξ'), состояние памяти ξ не изменяется, путешествие продолжается по дуге, ведущей к возврату с номером n';

6. v – возврат с символом r, и перед приходом в него была снята из магазина пара (n', ξ'); тогда состояние памяти ξ преобразуется в состояние $\sigma r(\xi', \xi)$, и путешествие продолжается по дуге, исходящей из v;

7. v – выход; тогда путешествие заканчивается; в этом случае говорим, что программа остановилась на паре (σ, ξ_0) с заключительным состоянием ξ; заметим, что магазин опустел.

Процедура выполнения программы описана.

При заданной семантике σ программой реализуется частичное отображение множества Ξ_σ в себя: оно определено для ξ_0 из Ξ_σ тогда и только тогда, когда программа останавливается на паре (σ, ξ_0), и в этом случае состоянию ξ_0 сопоставляет заключительное состояние.

Программы G_1, G_2 назовем *σ–эквивалентными* (обозначим: $G_1 \overset{\sigma}{\sim} G_2$) в том и только в том случае, если они осуществляют одно и то же отображение Ξ_σ в себя.

В [6] описана связь реальных программ с формализованными.

3. Алгебраические модели программ с процедурами

Все такие модели строятся над базисом $Y \cup C \cup R \cup P$ и отличаются друг от друга отношением эквивалентности своих объектов, именуемых схемами программ.

Схема программы по своей структуре совпадает с программой над тем же базисом, что и схема.

Функционирование схемы осуществляется на функциях разметки. Введем понятие последней.

Слово в алфавите $Y \cup C \cup R$ называется *цепочкой*; цепочка считается *правильной*, если в любом ее префиксе число вхождений символов из R не превосходит числа вхождений символов из C. Обозначим H множество всех правильных цепочек. Пусть

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

Функцией разметки назовем отображение $\mu : H \to X$. Множество всех функций разметки обозначим $\mathfrak{L}$.

Пусть $\mu \in \mathfrak{L}$. Выполнением схемы на функции μ назовем процесс, состоящий в путешествии по схеме и сопровождающийся построением правильной цепочки. 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, финальную вершину и распознаватель текущая цепочка не изменяется. Во втором случае из макушки магазина, который заведомо не пуст, извлекается номер, и путешествие продолжается по дуге, ведущей к возврату с этим номером. При переходе через распознаватель используется функция μ: в качестве следующей берется дуга, несущая метку $\mu h(p)$, где p – сопоставленная распознавателю переменная, а h – текущая цепочка. При достижении выхода путешествие прекращается; говорим, что *схема остановилась на* μ, и построенную цепочку называем *результатом ее выполнения на* μ.

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 *алгебраической* (ν, L)*–моделью*. Последнюю назовем *простой*, если $C = \varnothing, R = \varnothing$. Параметрами (ν, L)–модели назовём ν, L.

(ν, L)–модель назовем *строго аппроксимирующей*, если существует такое непустое множество S семантик базиса, что для любых схем G_1, G_2 выполняется

$$G_1 \overset{\nu,L}{\sim} G_2 \iff \forall \sigma \in S \; (G_1 \overset{\sigma}{\sim} G_2).$$

В [7] доказана теорема 1.

Теорема 1. *Алгебраическая* (ν, L) *– модель программ является строго аппроксимирующей, если*

1. ν *– полугрупповая эквивалентность в* H*,*
2. *множество* L *состоит из* ν*–согласованных функций разметки и замкнуто по сдвигу.*

Опуская доказательство теоремы 1, введем используемые в ней понятия.
Эквивалентность ν является *полугрупповой*, если

1. $\forall(h_1, h_2, h_3, h_4) \in H^4 : (h_1 \overset{\nu}{\sim} h_2)\&(h_3 \overset{\nu}{\sim} h_4) \Rightarrow (h_1 h_3 \overset{\nu}{\sim} h_2 h_4)$;
2. $\forall(h_1, h_2) \in H^2, \forall r \in R : (h_1 \overset{\nu}{\sim} h_2) \rightarrow (h_1 r \in H \iff h_2 r \in H)\&((h_1 r, h_2 r \in H) \rightarrow (h_1 r \overset{\nu}{\sim} h_2 r))$.

Здесь факт ν – эквивалентности цепочек h_1, h_2 из H записывается в виде $h_1 \overset{\nu}{\sim} h_2$.
Функцию разметки $\mu, \mu \in \mathfrak{L}$, назовем ν *– согласованной*, если для любых цепочек h_1, h_2 из H справедливо

$$h_1 \overset{\nu}{\sim} h_2 \Rightarrow \mu h_1 = \mu h_2.$$

Сдвигом функции μ *на цепочку* $h, h \in H$, назовем функцию $\mu', \mu' \in \mathfrak{L}$, определенную следующим образом: какой бы ни была цепочка h' из H, $\mu' h' = \mu h h'$; функцию μ' будем обозначать μ_h. Множество $L, L \subseteq \mathfrak{L}$, по определению, *замкнуто по сдвигу*, если для любых μ из L и любых h из H функция μ_h принадлежит L.

Далее рассматриваются только модели, удовлетворяющие требованиям теоремы 1.

Покажем (теорема 2), что проблема эквивалентности в произвольной алгебраической модели программ с процедурами сводится к проблеме эквивалентности в родственной ей модели, объекты которой именуются матричными схемами.

Как и алгебраические модели, так и определяемые ниже модели строятся над базисом Y, P, C, R, который далее обозначается как B.

Опишем матричную схему. По структуре это – конечный ориентированный граф, который отличается от графа, задающего схему программ над B, только в одном пункте: в нём нет распознавателей, а из каждой вершины типа вход, инициальная, преобразователь, возврат исходят дуги в количестве, равном числу элементов в X, и каждая дуга помечена своим элементом x из X; кроме того, в любом подграфе имеется специальная вершина *loop*, из которой не исходят дуги.

Пусть G – матричная схема над базисом B. Используем введенное ранее определение функции разметки и опишем процесс выполнения схемы G на функции μ из $\mathfrak{L}$. Он заключается в путешествии по схеме, сопровождающемся построением правильной цепочки. Кроме функции μ, используется магазин, в который загружаются номера пар вызов–возврат. 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 – это преобразователь, инициальная, вход, возврат, и h – это полученная приписыванием цепочка, то переход из вершины осуществляется по дуге с меткой μh. Если переходимая вершина – вызов, то после приписывания к текущей цепочке сопоставленного ему символа в магазин загружается его номер. При переходе через финальную вершину из макушки магазина, который заведомо не пуст, извлекается номер, и путешествие продолжается по дуге, ведущей к возврату с этим номером. Путешествие по схеме G завершается либо попаданием в выход схемы, либо достижением вершины *loop*. В первом случае построенная цепочка называется *результатом выполнения схемы на функции* μ, а во втором случае выполнение схемы на μ считается *безрезультатным*.

По полной аналогии со случаем алгебраической модели программ вводится понятие (ν, L)–эквивалентности матричных схем, где ν – эквивалентность в множестве H, а $L \in \mathfrak{L}$.

Таким образом, нами введена (ν, L)–модель матричных схем над B.

Теорема 2. *Проблема эквивалентности в* (ν, L)*–модели программ над базисом* B *сводится к проблеме эквивалентности в* (ν, L)*–модели матричных схем над базисом* B.

Доказательство.
Доказательство состоит в описании алгоритма, который по схеме из первой модели строит схему из второй модели и делает это так, что эквивалентные схемы переходят в эквивалентные схемы.

Пусть G – схема из алгебраической модели программ. Введем некоторые её характеристики. Пусть v – вершина схемы G типа вход, инициальная, возврат, преобразователь, и $x \in X$. Тогда x*–путем из* v назовем ориентированный путь в G, начинающийся в v и содержащий в качестве внутренних вершин только распознаватели; при этом выполнены требования: если распознаватель с переменной p принадлежит пути, то путь идет из этого распознавателя по дуге с меткой $x(p)$; путь завершается либо в вершине, отличной от распознавателя, либо в уже пройденном распознавателе; в первом случае достигнутую вершину назовем x–преемником вершины v, а во втором случае x–преемником v будем называть пустой цикл, т.е. распознаватель, обе дуги которого возвращаются к нему.

Требуемый теоремой алгоритм по схеме G строит матричную схему G', применяя следующие правила. Он переносит в G' образы всех вершин из G, кроме распознавателей, и добавляет в каждый подграф вершину *loop*. Для всякой вершины v из G, имеющей тип: вход, инициальная, преобразователь, возврат, и для любого x из X алгоритм находит x–преемника вершины v и тогда из образа вершины v в G' направляет дугу с меткой x в образ этого x–преемника (он называется x–преемником образа вершины v в G'); при этом образом пустого цикла считается вершина *loop*.

Схема G' построена.

Легко проверить, что путь выполнения схемы G на функции разметки μ из $\mathfrak{L}$ полностью копируется путем выполнения матричной схемы G', если любой x–путь в G, исходящий из вершины v, заменить x–дугой из образа вершины v.

Таким образом, алгоритм выполняет задачу, о которой говорится в теореме 2.

□

Пусть G – матричная схема над B. Вершину в G, имеющую тип вход, выход, вызов, возврат, назовем *опорной*. Говорим, что из опорной вершины v в опорную вершину v' идет *единичный маршрут* (*е. маршрут*), если в G имеется ориентированный путь из v в v', не содержащий опорных вершин.

Матричную схему G назовем *чистой*, если любой принадлежащий ей преобразователь находится на каком-либо е. маршруте.

Теорема 3. *Существует алгоритм, который произвольную матричную схему над B трансформирует в чистую матричную схему, эквивалентную первой в любой модели матричных схем.*

Доказательство.
Пусть G – матричная схема над B, поступившая на вход требуемого алгоритма. Тогда алгоритм осуществляет две разметки вершин схемы G – прямую и обратную. Прямая разметка применяется к самой схеме G; по завершении её алгоритм эквивалентными преобразованиями трансформирует схему G в схему G', удаляя, возможно, некоторые фрагменты схемы G. Обратная разметка применяется к схеме G', и по завершении её алгоритм трансформирует схему G' в эквивалентную ей схему G'', возможно, опять-таки удаляя некоторые фрагменты схемы G'.

Из построений будет видно, что схема G'' является чистой матричной.

Опишем действия, выполняемые алгоритмом.

Прямая разметка начинается в вершинах типа вход, инициальная, возврат, которые первыми получают метку. Далее метку получает всякий x–преемник, где $x \in X$, вершины, уже имеющей метку и отличной от вершины без x–преемников. Разметка завершается, когда не появляются новые помеченные вершины.

Легко увидеть, что вершины, не получившие метку, не участвуют в функционировании схемы G, а поэтому могут быть устранены из схемы корректным образом, т.е. преобразованием схемы опять-таки в схему.

Алгоритм осуществляет устранение вершин схемы, придерживаясь следующих правил.

Если выход схемы не получил метки, то схема G эквивалентна пустой схеме, т.е. такой, которая состоит из входа, выхода и вершины *loop*, причем все дуги из входа ведут в вершину *loop*. Тогда алгоритм объявляет, что схема G' – это пустая схема, и прекращает свою работу.

Если финальная вершина некоторого подграфа g схемы G не получила метку, то этот подграф устраняется целиком из схемы как не участвующий в функционировании схемы. Более того, устранению подлежат и все инцидентные подграфу g вызовы и возвраты (которые могут находиться и не в подграфе g).

В случае, когда неотмеченным оказался вызов, принадлежащий некоторому подграфу схемы, еще сохранившемуся в ней, алгоритм устраняет и этот вызов, и парный ему возврат.

Устранение вызова происходит так: все приходящие в него дуги, если таковые имеются, направляются в вершину *loop* подграфа, которому принадлежит этот вызов. После этого вызов устраняется вместе с исходящей из него дугой.

Устранение возврата осуществляется по правилу: он устраняется вместе с приходящей в него дугой и всеми исходящими из него дугами.

После выполнения описанных операций могут появиться подграфы без входящих в них дуг, и они устраняются алгоритмом из схемы. Теперь неотмеченными, может быть, остались преобразователи. Они устраняются, и схема G' построена.

Обратная разметка начинается в вершинах типа вызов, выход, финальная, которые отмечаются первыми. Далее метку получает всякая вершина, x–преемником которой оказалась помеченная вершина, $x \in X$. Разметка завершается, когда не появляются новые помеченные вершины.

Нетрудно увидеть, что вершины, не получившие метку, хотя и могут в некоторых случаях участвовать в функционировании схемы, но не могут принадлежать путям выполнения схемы, в случаях, когда она останавливается с результатом. Поэтому алгоритм устраняет их, придерживаясь следующих правил.

Если вход схемы G' оказался без метки, то схема G' эквивалентна пустой; тогда алгоритм объявляет, что G'' – пустая схема, и завершает работу.

Если инициальная вершина некоторого подграфа g схемы G' не получает метку, то подграф g устраняется вместе с инцидентными ему вызовами и возвратами.

В случае, когда не получил метку возврат, он устраняется вместе с парным ему вызовом. Устранение вызова и возврата проводится описанными выше правилами.

Если по выполнении этих операций появился подграф без приходящих в него дуг, то он тоже устраняется из схемы. В результате в отдельных подграфах могли остаться непомеченные преобразователи, составляющие в подграфе в совокупности фрагмент без исходящих из него дуг. Тогда все приходящие во фрагмент дуги алгоритм направляет в вершину *loop* в данном подграфе, после чего устраняет фрагмент.

Схема G'' построена.

□

4. Перегородчатые модели программ

В множестве алгебраических моделей программ с процедурами выделим перегородчатые модели.

Перегородчатая модель над B строится по заданной простой модели над Y, P. Пусть τ и l – параметры последней; тогда по τ определяется эквивалентность ν в H, а по l – множество функций разметки L, где $L \subseteq \mathfrak{L}$. Это и будут параметры перегородчатой модели.

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

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

Опишем, как строится множество L функций разметки.

Начнём с того, что задание отдельной функции разметки заключается в разметке элементами из X вершин бесконечного дерева, из каждой вершины которого исходят дуги, помеченные символами из множества $Y \cup C \cup R$, причём каждому символу выделяется в точности одна дуга. В этом дереве оставляются только ветви, несущие правильные цепочки.

Заметим, что всякая дуга дерева определяет начинающееся ею бесконечное поддерево, дуги которого помечены только символами из Y. Располагая множеством l функций разметки, отображающих Y^* в X, при построении функции из L осуществим следующую разметку вершин дерева: если дуга помечена символом из $C \cup R$, то индуцируемое ею поддерево размечается как некоторая функция из l; подобным образом размечается и поддерево с дугами, несущими метки из Y, которое вырастает из корня всего дерева. В L включаются только функции, размеченные описанным образом.

Легко устанавливается следующее утверждение.

Утверждение 1. *Если простая* (τ, l) *– модель над* (Y, P) *является аппроксимирующей, то таковой будет и построенная по ней перегородчатая* (ν, L) *– модель над* B*.*

Основные результаты, полученные в этом разделе, относятся к перегородчатым моделям специального вида, а именно: если $\mathfrak{M}'(\tau, l)$ – простая модель, для которой построена перегородчатая модель, то эквивалентность τ удовлетворяет требованию: любая цепочка из Y^* τ–эквивалентна пустой цепочке тогда и только тогда, когда сама пуста; а l состоит из всех τ–согласованных функций разметки. Описанная перегородчатая модель называется *моделью над полугруппой операторов, не имеющей циклов.*

Опираясь на теорему 3, будем рассматривать далее только чистые матричные схемы в модели над полугруппой без циклов; эти схемы составляют подмодель $\mathfrak{M}$.

Для подмодели $\mathfrak{M}$ установим следующие факты:

1. каждая схема из $\mathfrak{M}$ эквивалентными и эффективными преобразованиями трансформируется в так называемую свободную схему (теорема 4);

2. если построенные для свободных схем из $\mathfrak{M}$ диаграммы этих схем не изоморфны, то схемы не эквивалентны (лемма 1);

3. для свободных схем из $\mathfrak{M}$ с изоморфными диаграммами проблема эквивалентности сводится к проблеме эквивалентности простых свободных схем из $\mathfrak{M}'(\tau, l)$ (следствие теоремы 5).

Установлению фактов 1 – 3 предпошлем необходимые понятия.

Пусть G – схема из $\mathfrak{M}$. *Маршрутом* в G назовем ориентированный путь w, начинающийся во входе схемы и заканчивающийся в некоторой опорной вершине. Если последняя – это выход схемы G, то w именуем *маршрутом через схему.*

Всякому маршруту w в схеме G сопоставим *несомую им цепочку* $h(w)$ из $(Y \cup C \cup R)^*$, которая получается выписыванием друг за другом символов, сопоставленных вершинам маршрута w, при просмотре его от начала к концу.

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

Заметим теперь, что всякий маршрут в G представляет собой последовательность примыкающих друг к другу е. маршрутов, и займемся характеристиками е.маршрутов.

Рассмотрим е.маршрут u в схеме G, идущий из опорной вершины v_1 в опорную вершину v_2. Построим простую схему $G(u)$: в ней входом является вершина v_1, выходом – вершина v_2, и сохранен единственный путь из v_1 в v_2, идущий по е.маршруту u, а все ответвления от него направлены в пустой цикл.

Е.маршрут u назовем *реализуемым*, если в множестве l существует функция разметки, которая при выполнении схемы $G(u)$ на ней прокладывает фактически u.

Утверждение 2. *В любой схеме G из $\mathfrak{M}$ всякий е.маршрут является реализуемым.*

Доказательство.
Действительно, реализации е.маршрута u могли бы препятствовать имеющиеся в схеме $G(u)$ подмаршруты, несущие эквивалентные цепочки из Y^* и продолжающиеся по дугам с различными метками, но в силу отсутствия циклов в полугруппе операторов этого препятствия нет; к тому же множество l состоит из всех τ–согласованных функций разметки.

□

Утверждение 3. *Всякий маршрут в схеме G из $\mathfrak{M}$ является реализуемым, если составляющие его е.маршруты реализуемы, а он сам несет правильную цепочку.*

Справедливость этого утверждения очевидна.

Схему G из $\mathfrak{M}$ назовем *свободной*, если в ней любой е.маршрут принадлежит какому-либо реализуемому маршруту.

Теорема 4. *Существует алгоритм, который произвольную схему G из $\mathfrak{M}$ трансформирует в эквивалентную ей свободную схему.*

Доказательство.
Требуемый алгоритм сначала проводит анализ схемы G.

Пусть g – подграф схемы G. *Трассой* в g назовем путь из инициальной вершины (входа, если g – главный подграф) в финальную вершину (соответственно, в выход схемы), построенный с соблюдением правила: за всяким вызовом идет парный ему возврат.

Алгоритм выявляет, имеется ли в схеме G подграф с трассой, в который нет вызовов–возвратов. Такая трасса именуется *сквозной*.

Если в схеме G нет подграфов со сквозными трассами, то алгоритм трансформирует G в пустую схему, ибо в схеме G нет реализуемых маршрутов через неё, и прекращает свою работу.

Пусть в G имеются подграфы со сквозными трассами, и g – один из них. Тогда получают метку вызовы и возвраты, инцидентные подграфу g.

Если в процессе разметки обнаружился подграф g', в котором в некоторой принадлежащей ему трассе получили метку все вызовы и возвраты, то подграф g' возводится в ранг располагающего сквозной трассой, и с ним алгоритм работает так же, как с подграфом g, продолжая разметку вызовов и возвратов, инцидентных подграфу g'.

Очевидно, что разметка завершается, когда исчерпаны подграфы со сквозными трассами.

Тогда алгоритм приступает к анализу главного подграфа. В нём сквозной объявляется трасса, в которой каждые вызов–возврат, если они имеются, инцидентны некоторому подграфу со сквозной трассой.

Если главный подграф не располагает сквозными трассами, то алгоритм трансформирует схему G в пустую схему, так как в G отсутствуют реализуемые маршруты через неё. После этого алгоритм считает завершенной свою работу.

В ином случае алгоритм устраняет в схеме G все элементы, не функционирующие при выполнении схемы на функциях разметки из L. К ним принадлежат, в первую очередь, подграфы без сквозных маршрутов. Они устраняются по правилу, описанному в теореме 3. Далее алгоритм устраняет вызовы и парные им возвраты во всех трассах, не являющихся сквозными и не принадлежащие сквозным трассам. Это осуществляется так, как описано в теореме 3.

Теперь в схеме G остались только такие е.маршруты, каждый из которых принадлежит некоторому реализуемому маршруту через схему, и алгоритм заканчивает свою работу.

□

Обратимся к проблеме эквивалентности в классе свободных схем из подмодели $\mathfrak{M}$.

Пусть G_1, G_2 – свободные схемы из $\mathfrak{M}$, и w_1, w_2 – маршруты в G_1, G_2 соответственно. Назовем w_1, w_2 *сочетаемыми*, если они прокладываются общей для них функцией разметки из L. *Длиной маршрута* будем называть число составляющих его е.маршрутов.

Утверждение 4. *Равновеликие сочетаемые маршруты в эквивалентных схемах из $\mathfrak{M}$ несут эквивалентные цепочки, т.е., в частности, цепочки с равными проекциями на алфавит $C \cup R$.*

Доказательство утверждения очевидно.

Учитывая это, сопоставим свободной схеме G из $\mathfrak{M}$ *диаграмму* $D(G)$ – ориентированный граф, вершинами которого являются все опорные вершины схемы G, и каждому е.маршруту из v_1 в v_2 соответствует дуга из вершины v_1 в вершину v_2.

Лемма 1. *Если свободные схемы G_1, G_2 из $\mathfrak{M}$ эквивалентны, то диаграммы $D(G_1)$, $D(G_2)$ изоморфны.*

Доказательство.
Пусть u_1 – е.маршрут в схеме G_1, индуцирующий дугу в $D(G_1)$ из вершины v_{11} в вершину v_{12}. Так как G_1 – свободная схема, то существует реализуемый маршрут

w_1 через G_1, содержащий е.маршрут u_1. Но тогда в эквивалентной схеме G_2 имеется маршрут w_2 через неё, сочетаемый с маршрутом w_1. Условимся называть подобными дуги диаграмм $D(G_1), D(G_2)$, начало и конец которых, будучи помеченными, имеют равные метки. Поскольку $h(w_1)$ и $h(w_2)$ эквивалентны, дуге из v_{11} в v_{12} соответствует в $D(G_2)$ подобная ей дуга u_2 из v_{21} в v_{22}. Поскольку дуга u_1 выбрана произвольно, а в рассуждениях схема G_1 может быть заменена схемой G_2, изоморфность диаграмм $D(G_1), D(G_2)$ установлена.

□

Пусть v_1, v_2 – опорные вершины в схеме G из $\mathfrak{M}$, и из v_1 в v_2 идет е.маршрут. Построим простую схему $G(v_1, v_2)$. Включим в неё все вершины и дуги, принадлежащие е.маршрутам из v_1 в v_2, за исключением дуги, исходящей из финальной вершины, если только таковая им принадлежит. Вершина v_1 объявляется входом схемы $G(v_1, v_2)$, т.е. лишается сопоставленного ей символа, если он имелся. Выходом схемы $G(v_1, v_2)$ объявляется финальная вершина, если она принадлежит рассматриваемым е.маршрутам; в противном случае выходом объявляется вершина v_2, то есть она лишается сопоставленного ей символа, если он был. Все остальные вершины и дуги, вошедшие в $G(v_1, v_2)$, наследуют приписанные им в G символы и метки. Если при этом в $G(v_1, v_2)$ оказались преобразователи, исходящие из которых дуги не принадлежат рассматриваемым е.маршрутам, то эти дуги направляются в пустой цикл, который вносится в схему $G(v_1, v_2)$, если его там не было. Схема $G(v_1, v_2)$ построена.

Теорема 5. *Свободные схемы G_1, G_2 из $\mathfrak{M}$ с изоморфными диаграммами эквивалентны тогда и только тогда, когда, какими бы ни были изоморфные дуги (v_{11}, v_{12}) и (v_{21}, v_{22}) в диаграммах $D(G_1)$, $D(G_2)$ соответственно, схемы $G_1(v_{11}, v_{12})$ и $G(v_{21}, v_{22})$ эквивалентны.*

Доказательство.
Необходимость этого требования вытекает из эквивалентности схем G_1, G_2, их свободности и из изоморфности их диаграмм.

Достаточность требований теоремы следует из определения сочетаемых маршрутов в G_1, G_2 и утверждения 4.

□

Следствие теоремы 5. Проблема эквивалентности свободных схем из $\mathfrak{M}$ с изоморфными диаграммами сводится к проблеме эквивалентности схем из модели $\mathfrak{M}'(\tau, l)$, индуцирующей модель $\mathfrak{M}$.

Список литературы

1. Ляпунов А.А. О логических схемах программ // Проблемы кибернетики. Вып. 1. М.: Физматгиз, 1958. С. 46–74.

2. Янов Ю.И. О логических схемах алгоритмов // Проблемы кибернетики. Вып. 1. М.: Физматгиз, 1958. С. 75–127.

3. Подловченко Р.И. Об одной методике распознавания эквивалентности в алгебраических моделях программ // Программирование. 2011. №6. С. 33–43.

4. Захаров В.А. Проверка эквивалентности программ при помощи двухленточных автоматов // Кибернетика и системный анализ. 2010. №4. С. 39–48.

5. Подловченко Р.И. Эквивалентные преобразования в математических моделях вычислений // Учебное пособие. М.: Изд. отдел ф-та ВМиК МГУ; МАКС ПРЕСС, 2011. 72 с.

6. Подловченко Р.И. Абстрактные программы с процедурами и конечные автоматы с магазином // Интеллектуальные системы. 1997. Т. 2, вып. 1–4. С. 275–295.

7. Подловченко Р.И. От схем Янова к теории моделей программ // Математические вопросы кибернетики. 1998. Вып. 7. С. 281–302.

8. Подловченко Р.И. Специальные перегородчато-автоматные модели рекурсивных программ // Программирование. 1994. №3. С. 3–26.

About Algebraic Program Models with Procedures

Podlovchenko R.I., Molchanov A.E.

Keywords: program formalization, program scheme, program schemes equivalence, matric scheme, free scheme

Algebraic program models with procedures are designed to analyze program semantic properties on their models called program schemes. The concepts that give foundation to the theory of such models are stated along with a description of their implementation. The key point of the theory is the equivalence of program schemes that belong to a particular model. A class of special algebraic models with procedures, called gateway models, is studied. Necessary and sufficient conditions of the equivalence problem decidability in such models are proposed.

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