Конечные автоматы в теории алгебраических схем программ

Р.И. Подловченко <rimma.podlovchenko@gmail.com>
Московский Государственный Университет им. М.В. Ломоносова,
Россия, Москва, Ленинские горы, 1, стр. 52

Аннотация. Рассматриваемые в статье алгебраические модели программ обобщают две модели программ, введённые А.А. Ляпуновым и А.А. Летичевским. Показано, что алгебраические модели программ аппроксимируют компьютерные программы, через промежуточную формализацию. Центральное место в теории таких моделей занимает проблема эквивалентности схем программ. Существует достаточно много классов моделей, для которых эта проблема разрешима. Большинство разрешающих алгоритмов следуют структуре алгоритма проверки эквивалентности конечных автоматов. Целью данной статьи является выявление этой связи. Вводится эквивалентное представление моделей программ, называемое матричными схемами. Такое представление структурно ближе к конечным автоматам, и показано, что проблема эквивалентности в подклассе матричных схем программ сводится к таковой для конечных автоматов. Рассматривается алгоритм, решающий проблему эквивалентности КА; он формулируется в терминах требований к конечным участкам путей выполнения автоматов. В результате удаётся сформулировать более общий метод, применимый к другим моделям. Приводятся необходимые требования, предъявляемые к моделям для применимости к ним указанного метода. Приводятся две модели, уравновешенная полугрупповая модель с левым сокращением и коммутативная модель с монотонными операторами, в который разрешимость проблемы эквивалентности установлена указанным методом.

Ключевые слова: схема программ, алгебраическая модель программ, проблема эквивалентности, разрешающий алгоритм, конечный автомат.

DOI: 10.15514/ISPRAS-2015-27(2)-10

Для цитирования: Подловченко Р.И. Конечные автоматы в теории алгебраических схем программ. Труды ИСП РАН, том 27, вып. 2, 2015 г., стр. 161-172. DOI: 10.15514/ISPRAS-2015-27(2)-10.

1. Введение

Рассматриваемые нами алгебраические модели программ введены в [1] как обобщение моделей, исследованных в [2] и [3]. Эти модели предназначены для изучения семантических свойств реальных программ и, в первую очередь, решения для них проблемы эквивалентности. Она состоит в поиске алгоритма, который, получив на свой вход две программы, распознаёт, эквивалентны они функционально или нет.

Принципиальным отличием алгебраических моделей программ от моделей из [2] и [3] является то, что они введены для программ, предварительно формализованных. Центральное место в теории таких моделей принадлежит проблеме эквивалентности схем программ, заменивших формализованные программы. В [4] доказана разрешимость этой проблемы в широком классе алгебраических моделей программ. Обозревая методику, которой получен данный результат, можно прийти к выводу: проверяемые ею свойства алгебраических моделей подсказаны теми свойствами конечных автоматов [5], которыми обеспечивается разрешимость их эквивалентности.

Задача статьи состоит в следующем: отметить эти свойства конечных автоматов и перевести их на язык требований, предъявляемых к алгебраическим моделям программ и прогнозирующих разрешимость в них проблемы эквивалентности. В совокупности эти требования определят метод исследований, названный нами автоматным. Он действительно открыл один из путей, которыми распознаётся эквивалентность схем в алгебраических моделях программ.

Статья начинается экскурсом в теорию алгебраических моделей программ. В разделе 1описывются формализованные программы и определяются сами модели. Здесь же осуществляется отбор моделей, пригодных для распознавания семантических свойств формализованных программ (теорема 1). Отобранные модели называются строго аппроксимирующими, и только они рассматриваются в теории. В разделе 2 формулируется теорема 2, сводящая проблему эквивалентности в алгебраической модели программ к родственной ей модели, элементами которой являются матричные схемы. Демонстрируется, что по своей структуре матричная схема – это конечный автомат. Предварительно воспроизводится определение последнего. Раздел 3 посвящён описанию и анализу алгоритма, разрешающего эквивалентность конечных автоматов. Отмечаются те свойства автоматов, которые играют принципиальную роль в работе алгоритма. Сам автоматный метод изложен в разделе 4, а заключительный раздел 5 отводится применению метода для двух видов алгебраических моделей программ, что говорит о его действенности в вопросе разрешимости проблемы эквивалентности.

2. Формализованные программы и построенные для них модели программ.

При формализации понятия программы исходной является следующая установка: в программе, описанной на алгоритмическом языке типа Паскаль, сохранить лишь отдел операторов; это означает, что устраняется ввод начальных данных и вывод полученных результатов. Кроме того, предполагается, что все данные имеют общий тип. Вместе с тем, сохраняются все обычные композиции операторов за исключением аппарата процедур.

Согласно такой установке, формализованная программа строится над двумя конечными алфавитами Y и P; элементами алфавита Y являются обозначения операторов, а элементами алфавита P – обозначения логических условий.

Синтаксически *программа* представляет собой конечный ориентированный и размеченный граф; в нём выделены две вершины: *вход* без входящих в него дуг и с одной исходящей дугой и *выход* – вершина без исходящих из неё дуг. Остальные вершины, если они имеются, подразделяются на *преобразователи* и *распознаватели*; преобразователь помечается символом из Y, и из него исходит одна дуга; распознаватель помечается символом из P, и из него исходят две дуги с метками 0 и 1 соответственно.

Пример программы дан на рис.1; здесь y_1, y_2 – символы из Y, приписанные преобразователям, а p_1, p_2 – символы из P, сопоставленные распознавателям.

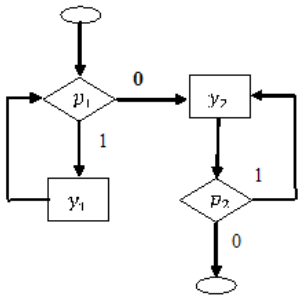

Рис. 1

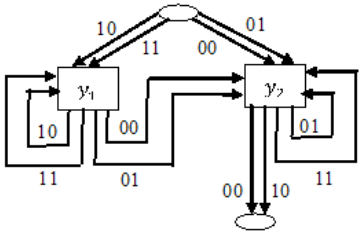

Рис. 2

Семантика формализованной программы (далее – просто программы) задаётся *семантикой* σ базиса Y, P; её составными частями являются:

- предметное множество Θ, элементы которого называются *состояниями*;
- отображения $\sigma_y: \Theta \to \Theta$, где $y \in Y$;
- отображения $\sigma_p: \Theta \to \{0,1\}$, где $p \in P$.

Для заданной семантики σ вводится процедура выполнения программы π на начальном состоянии ξ_0 из Θ. Она представляет собой детерминированный

обход программы π, который начинается в её входе при состоянии ξ_0 и подчиняется следующим правилам. Пусть при состоянии ξ достигнута вершина v программы π; если v – выход, то обход прекращается, а ξ воспринимается как *результат выполнения* π на состоянии ξ_0; если v – преобразователь с приписанным ему символом y, то состояние ξ перерабатывается в состояние $\sigma_y(\xi)$, и обход продолжается по дуге, исходящей из v; если же v – распознаватель с приписанным ему символом p, то состояние ξ сохраняется, а обход продолжается по дуге из v, помеченной числом $\sigma_p(\xi)$.

Таким образом, программе π сопоставляется реализуемая ею функция $\sigma\pi$, отображающая множество Θ в себя, в общем случае, частично. Программы π', π'' над Y, P называются *эквивалентными* при семантике σ, если $\sigma\pi' = \sigma\pi''$. Так возникает -класс программ, в котором рассматриваются проблема эквивалентности и проблема построения эквивалентных преобразований (э.п.) программ.

Перейдём к определению алгебраической модели программ над Y, P. Её элементами являются *схемы программ*. Чтобы всякое преобразование структуры схемы одновременно было бы преобразованием структуры программы, для которой построена схема, принимается решение: *синтаксически схема определяется так же, как программа*. Таким образом, предстоит определить лишь семантику схем программ.

Пусть G – схема над Y, P. Введём процедуру выполнения G на функции, которая отображает множество всех слов над алфавитом Y в множество X, где $X = \{x \mid x: P \to \{0,1\}\}$. Первое множество обозначим Y^*, а его элементы будем называть *цепочками*. Элементы второго множества назовём *наборами*, а саму функцию – *функцией разметки*, построенной над базисом Y, P. Обозначим $\mathcal{L}$ множество всех таких функций. Процедура выполнения схемы G на функции μ из $\mathcal{L}$ представляет собой детерминированный обход схемы G, который начинается в её входе с пустой цепочкой из Y^* и подчиняется следующим правилам. Допустим, что с цепочкой h достигнута вершина v схемы G. Если v – выход схемы, то обход её прекращается, а цепочка h называется *результатом выполнения схемы на* μ. Если h – преобразователь с приписанным ему символом y, то цепочка h трансформируется в цепочку hy, а обход схемы продолжается по дуге, исходящей из v. В случае, когда v – распознаватель, а p – сопоставленный ему символ, цепочка h остаётся неизменённой, а обход схемы продолжается по дуге, исходящей из v и помеченной числом, равным значению p в наборе $\mu(h)$. Таким образом схеме G приписывается отображение множества $\mathcal{L}$ в множество Y^*, частичное в общем случае.

Эквивалентность схем над Y, P индуцируется параметрами ν и L, где ν – отношение эквивалентности в Y^*, а L – подмножество множества $\mathcal{L}$, и определяется так: две схемы G_1, G_2 эквивалентны тогда и только тогда, когда,

какой бы ни была функция μ из L, если одна из схем останавливается на μ, то другая останавливается тоже, и результаты их выполнения на μ – это - эквивалентные цепочки. Множество схем над Y, P с введённой в нём эквивалентностью схем назовём *алгебраической моделью программ* над Y, P, а ν, L – её параметрами.

Напомним теперь, что введённые модели предназначены для исследования семантических свойств формализованных программ и для разработки на схемах эквивалентных преобразований программ. Это обязывает отобрать пригодные модели.

Пусть M – модель над Y, P, а σ – семантика базиса Y, P. Говорим, что M *аппроксимирует* –класс программ, если, какими бы ни были схемы G_1, G_2 из M, из их эквивалентности в M следует равенство функций $\sigma\pi_1, \sigma\pi_2$, где π_i – программа той же структуры, что и структура G_i, $i = 1,2$. Очевидно, что аппроксимирующая модель пригодна для объявленной выше задачи. Однако для отбора их пришлось предъявить дополнительные требования.

Модель M назовём *строго аппроксимирующей*, если существует такое множество S семантик базиса Y, P, что схемы из M эквивалентны тогда и только тогда, когда для любой семантики σ из S модель M аппроксимирует - класс программ. Достаточные условия строгой аппроксимируемости даёт

Теорема 1. Алгебраическая модель программ является строго аппроксимирующей, если её параметры ν и L удовлетворяют следующим требованиям: ν – это полугрупповая эквивалентность в Y^*, а множество L состоит из -согласованных функций разметки и является замкнутым по операции сдвига.

Теорема 1 доказана в [4]. Опишем используемые ею понятия. Эквивалентность ν называется *полугрупповой*, если, какими бы ни были цепочки h_1, h_2, h_3, h_4 из Y^*, из эквивалентности h_1, h_2 вместе с эквивалентностью h_3, h_4 следует эквивалентность цепочек h_1h_3 и h_2h_4. Функция разметки μ из $\mathcal{L}$ называется -*согласованной*, если она сохраняет своё значение на каждом классе ν-эквивалентных цепочек из Y^*. *Сдвигом функции* μ на цепочку h называется функция μ_h, обладающая свойством: для любой цепочки h' из Y^* верно равенство $\mu_h(h') = \mu(hh')$. Множество L, по определению, *замкнуто по операции сдвига*, если для любых μ из L и h из Y^* функция μ_h принадлежит L.

Как было отмечено во введении, рассматриваются только алгебраические модели, параметры которых удовлетворяют требованиям теоремы 1.

3. Матричные схемы и конечные автоматы

Связь между алгебраическими моделями программ и конечными автоматами, введёнными в [5], проявилась на основе теоремы 2, доказанной, в частности, в [4].

Теорема 2. Какой бы ни была алгебраическая модель программ M над базисом Y, P, проблема эквивалентности в ней сводится к родственной ей модели M_0 над тем же базисом.

Опишем модель M_0. Её элементами являются *матричные схемы*, построенные над Y, P. Структура матричной схемы выглядит так. Это – конечный ориентированный и размеченный граф, в котором выделены три вершины – *вход* без входящих в него дуг, *выход* и вершина loop, каждая без исходящих из неё дуг. Остальные вершины называются *преобразователями*. Каждому преобразователю приписан свой символ из Y. Из всякой вершины, отличной от выхода и loop, исходят дуги в количестве, равном числу наборов в X, причём каждая дуга помечена своим набором. Структура матричной схемы описана. На рис.2 приведена матричная схема, построенная по теореме 2 для схемы с рис. 1 и эквивалентная ей в любой модели M.

Матричной схеме сопоставляется отображение множества $\mathcal{L}$ функций разметки над Y, P в множество Y^*. Оно осуществляется процедурой выполнения схемы на функциях из $\mathcal{L}$. Описание процесса выполнения схемы из M_0 на функции μ отличается от того, как выполняется на μ схема из M, следующими деталями:

- обход схемы из её входа идёт по дуге, помеченной набором $\mu(\Lambda)$, где Λ – пустая цепочка из Y^*;
- из преобразователя с символом y, достигнутом при обходе с цепочкой h из Y^*, он продолжается по дуге, помеченной набором μ(hy);
- при достижении вершины loop обход прекращается без результата.

Параметрами модели M_0 являются параметры модели M, а отношение эквивалентности матричных схем в M_0 вводится также, как отношение эквивалентности схем в M. Поскольку далее будут рассматриваться модели типа M_0, их элементы называются просто схемами.

Модель M_0, параметрами которой являются тождество цепочек в Y^* и всё множество $\mathcal{L}$, называется *максимальной* в силу того, что из эквивалентности схем в этой модели следует эквивалентность их в любой модели из числа рассматриваемых нами. Имеет место лемма 1, доказанная в [6]. Она даётся в редакции, используемой в [7].

Лемма 1. Проблема эквивалентности в максимальной модели M_0 над базисом Y, P сводится к проблеме эквивалентности конечных автоматов над алфавитом $(Y \cup \{y_0\}) \times X$, где y_0 – добавочный символ.

Доказательству её предпошлём напоминание о том, как определяются конечные автоматы и отношение их эквивалентности.

Автоматом над алфавитом Σ называется конечный ориентированный граф с размеченными дугами. В нём одна вершина называется *инициальной*, и некоторые вершины (возможно, пустое множество) – *финальными*. Из каждой вершины исходят дуги в количестве, равном числу символов в Σ, и каждая

дуга помечена своим символом. Всякое слово над алфавитом Σ прокладывает в автомате маршрут, начинающийся в инициальной вершине и составленный дугами, метки которых при просмотре маршрута слагаются в это слово. По определению, оно *принимается автоматом*, если маршрут оканчивается в финальной вершине. Автоматы называются *эквивалентными*, если они принимают равные множества слов.

Доказательство леммы 1 состоит в построении алгоритма, который, получив на свой вход схему G из M_0, строит автомат $A(G)$, и в установлении корректности этого алгоритма. Ограничимся описанием алгоритма. Вершинами автомата $A(G)$ он объявляет образы всех вершин схемы G, при этом образ входа называет инициальной вершиной, образ выхода – единственной финальной вершиной, а образом loop – вершину, именуемую мёртвой, ибо все исходящие из неё дуги ведут в неё же.

Пусть v – вершина схемы G, а $\bar{v}$ – её образ в $A(G)$. Если v – это вход, то всякая дуга из него, помеченная набором x, порождает дугу из $\bar{v}$, помеченную парой (y_0, x) и ведущую в образ той вершины схемы, в которую ведёт первая дуга. Если v – это преобразователь с меткой y, то любая дуга из него, несущая метку x, порождает дугу из $\bar{v}$ с меткой (y, x), оканчивающуюся в образе той вершины схемы, в которой оканчивается первая дуга. Все остальные дуги, долженствующие быть в автомате $A(G)$, алгоритм направляет в мёртвую вершину. Этим автомат $A(G)$ построен. Обозревая описанный алгоритм, легко установить, что структура схемы из M_0 подобна структуре конечного автомата.

4. Разрешимость эквивалентности конечных автоматов.

Пусть конечные автоматы строятся над алфавитом Σ, где $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_\ell\}, \ell \geq 2$. Автомат над Σ назовём *приведённым*, если любая вершина в нем, за исключением мёртвой, находится на маршруте через него, то есть начинающимся в инициальной вершине и оканчивающимся в финальной. Мертвой называется вершина, в которую возвращаются все исходящие из неё дуги. Справедливо

Утверждение 1. Существует алгоритм, который поступивший на его вход автомат над Σ трансформирует в эквивалентный ему приведённый автомат над Σ.

Действительно, такой алгоритм сначала оставляет в автомате все вершины, достижимые из инициальной, затем все вершины, из которых достижима какая-либо финальная вершина, заменяя при этом прочие вершины мёртвой.

Воспользуемся алгоритмом, распознающим эквивалентность приведённых автоматов над Σ и известным как алгоритм Мура. Обозначим его для определённости символом ρ. Пусть A_1, A_2 – приведённые автоматы над Σ, поступившие на вход алгоритма ρ. Он строит таблицу, в которой, кроме ведущего столбца, имеется ℓ столбцов. Элементами таблицы будут пары

a_1, a_2 где a_i – вершина автомата $A_i, i = 1,2$. Элементы ведущего столбца таблицы называются *ведущими*, а позиция в строке таблицы, находящаяся в -ом столбце, – j-ой позицией. .

Сначала алгоритм ρ записывает в ещё пустую таблицу в качестве первого ведущего элемента пару (a_{10}, a_{20}), где a_{i0} – инициальная вершина автомата $A_i, i = 1,2$, и приступает к заполнению первой строки.

Предположим, что алгоритм ρ добрался до заполнения i-ой позиции строки с ведущим элементом (a_1, a_2). Тогда он строит пару (b_1, b_2), где b_i – вершина, в которую идёт дуга с меткой σ_j из вершины $a_i, i = 1,2$. В случае, когда вершины b_1, b_2 разнотипны (типы – финальная, мёртвая, прочая), алгоритм ρ останавливается, справедливо заявляя, что автоматы A_1, A_2 не эквивалентны. В ином случае он проверяет, имеется ли в ведущем столбце пара (b_1, b_2), и, если её нет, записывает её в первую свободную позицию ведущего столбца и продолжает свою работу, переходя к заполнению следующей позиции той же строки, если таковая имеется. Когда вся строка заполнена, а в таблице имеются незаполненные строки, алгоритм ρ работает со следующей строкой, останавливаясь с ответом «A_1, A_2 эквивалентны» в случае, когда таблица полностью построена. Этим алгоритм ρ описан.

Проанализируем работу алгоритма ρ, введя предварительно нужные понятия. Легко видеть, что всякий маршрут в автомате является *реализуемым*, то есть прокладывается некоторым словом над Σ. Равновеликие по длине маршруты в автоматах назовём *сочетаемыми*, если они прокладываются общим для них словом. Если в сочетаемых маршрутах повторяется пара равноудалённых от их начала вершин, то отрезки маршрутов, соединяющие эти вершины, назовём *сочетаемыми интервалами*.

Обратимся теперь к алгоритму ρ. Из его описания следует, что он строит сочетаемые маршруты в автоматах A_1, A_2, постепенно увеличивая их длину. При этом алгоритм ρ учитывает два свойства, присущие автоматам, а именно:

– если маршруты оканчиваются в вершинах разного типа, то автоматы не эквивалентны;

– в эквивалентных автоматах сочетаемые интервалы несут равные слова, и сокращение маршрутов на эти интервалы приводит к сочетаемым маршрутам.

Второе свойство позволяет не повторять пары в ведущем столбце, что и обеспечивает завершаемость построения таблицы.

5. Автоматный метод распознавания эквивалентности схем программ

Этот метод, отталкиваясь от свойств, на которые опирается алгоритм ρ, предписывает требования к модели с матричными схемами программ, выполнение которых прогнозирует разрешимость в ней проблемы эквивалентности. Областью его применимости являются только те модели, в

которых ν-эквивалентные цепочки над Y равны по длине; здесь ν – первый параметр модели.

Пусть M_0 – модель программ из этой области. Изложению автоматного метода предпошлём описание некоторых характеристик схем из M_0.

Маршрутом в схеме называется ориентированный путь в ней, начинающийся во входе схемы; если путь завершается в выходе схемы, то он называется *маршрутом через схему*. Всякому конечному маршруту в схеме сопоставим *несомую им цепочку* над Y, которая строится выписыванием друг за другом меток преобразователей при просмотре маршрута от начала к концу; если маршрут оканчивается в преобразователе, то его символ не учитывается. Маршрут в схеме называется *реализуемым*, если он прокладывается при выполнении схемы на некоторой функции разметки, допустимой для M_0. Маршруты в двух схемах называются *сочетаемыми*, если они прокладываются общей для них функцией разметки. В двух сочетаемых маршрутах, в которых повторяется пара равноудалённых от их начала вершин, отрезки их, соединяющие повторяющиеся вершины, будем называть *сочетаемыми интервалами*.

Необходимость в рассмотрении сочетаемых маршрутов в схемах из M_0 обоснована самим определением эквивалентности схем, которое можно дать в следующей редакции: схемы в M_0 эквивалентны тогда и только тогда, когда, каким бы ни был маршрут через одну из них, всякая функция разметки из L, прокладывающая этот маршрут, вместе с тем прокладывает в другой схеме маршрут через неё, несущий цепочку, -эквивалентную цепочке, несомой первым маршрутом.

Поскольку в модели M_0 имеются схемы с нереализуемыми маршрутами, автоматный метод выставляет

Требование 1. Доказать, что проблема эквивалентности в модели M_0 сводится к множеству принадлежащих ей схем со свойствами: любой маршрут в схеме реализуем, а каждый преобразователь находится на маршруте через неё.

Схема с указанным здесь свойством называется *свободной в* M_0.

Предположим, что это требование выполнено, и обозначим M_1 подмодель модели M_0, состоящую из всех свободных в ней схем. Теперь проблема эквивалентности рассматривается в M_1, и возникает

Требование 2. Сформулировать алгоритмически проверяемый критерий сочетаемости маршрутов в схемах из M_1.

Наконец, формулируется

Требование 3. Выбрать параметры модели M_1 такими, чтобы необходимыми условиями эквивалентности схем в ней являлись два условия:

1) равновеликие сочетаемые маршруты в схемах оканчиваются в вершинах общего типа;

2) сочетаемые интервалы в схемах несут -эквивалентные цепочки, и сокращение маршрутов на сочетаемые интервалы приводит к сочетаемым маршрутам.

Автоматный метод, которым прогнозируется разрешимость эквивалентности в модели M_0, описан.

6. Заключение

Первым примером применимости автоматного метода являются уравновешенные полугрупповые модели программ с левым сокращением, рассмотренные в [4]. Опишем их параметры ν и L.

Эквивалентность ν является полугрупповой, а множество L состоящим из всех -согласованных функций разметки. Таким образом, модель является аппроксимирующей. Уравновешенность модели трактуется как требование: -эквивалентные цепочки равновелики по длине. Это позволило применить к модели автоматный метод. Свойство левого сокращения формулируется так: какими бы ни были цепочки h_1, h_2, h_3, h_4 из Y^*, выполняется импликация $(h_1h_3 \sim h_2h_4) \wedge (h_1 \sim h_2) \Rightarrow h_3 \sim h_4$.

Доказательство разрешимости проблемы эквивалентности в этих моделях выполнено автоматным методом. Таблица, аналогичная той, что строится для конечных автоматов алгоритмом ρ, состоит из пар так называемых сопряжённых вершин схем: такие вершины достижимы равновеликими сочетаемыми маршрутами, которые несут -эквивалентные цепочки.

Другим примером является коммутативная модель программ с монотонными операторами, исследованная в [8]. В ней -эквивалентными объявляются цепочки, любая из которых получается из другой перестановками соседних символов. А множество L состоит из всех таких -согласованных функций разметки μ, которые удовлетворяют требованию : какими бы ни были цепочка h из Y^* и символ y из Y, верно соотношение $\mu(h) \leq \mu(hy)$; по определению, наборы x_1, x_2 находятся в отношении $\leq$, если для любого p из P выполняется: $x_1(p) \leq x_2(p)$.

Описанная модель является аппроксимирующей. Очевидно, что она входит в область применимости автоматного метода. При распознавании эквивалентности схем из этой модели таблицу, аналогичную таблице алгоритма ρ, приходится строить для каждого из маршрутов через схемы, ограничиваясь маршрутами, длина которых определяется размерами самих схем.

В [4] и [8] обсуждается и сложность алгоритма, распознающего эквивалентность схем.

Список литературы

[1]. Подловченко Р.И. Иерархия моделей программ. Программирование, 1981, №2, стр. 3-14.

[2]. Ляпунов А.А. О логических схемах программ. Проблемы кибернетики. М.: Физматгиз, Вып.1, 1958, стр. 46-74.

[3]. Глушков В.М., Летичевский А.А. Теория дискретных преобразователей. Избранные вопросы алгебры и логики. Новосибирск. ИМ СО АН СССР, 1973, стр. 5-39.

[4]. Подловченко Р.И. Об одной методике распознавания эквивалентности в алгебраических моделях программ. Программирование, 2011, № 6, с. 33-43.

[5]. Rabin M.O., Scott D. Finite automata and their decision problems. IBM Journal of Research and Development, 1959, vol.3, №2, p. 114-125.

[6]. Rutledge J.D. On Ianovs program schemata. Journal of the ACM, 1964, Vol. 11, № 1, p. 1-9.

[7]. Подловченко Р.И. Эквивалентные преобразования в математических моделях вычислений. Учебное пособие. М.: Издат. отдел факультета ВМиК МГУ им. М.В.Ломоносова. МАКС-Пресс, 2011, 72 с.

[8]. Подловченко Р.И. О схемах программ с перестановочными и монотонными операторами. Программирование, 2003, №5, стр. 46-54.

Finite State Automata in the Theory of Algebraic Program Schemata

R. I. Podlovchenko <rimma.podlovchenko@gmail.com>
Lomonosov Moscow State University,
1/52, Leninskie Gory, Moscow, Russia

Annotation. Algebraic program models considered in this paper generalize two models of programs introduced by A.A. Lyapunov and A.A. Letichevsky. The algebraic models are shown to approximate real-world programs via an intermediate formalization. The theory of program models focuses on the equivalence problem for program schemata. There are many results of decidability of this problem for various classes of algebraic program models. Most of these deciding algorithms derive from the equivalence checking algorithm for finite state automata. The aim of this paper is to reveal this relationship. An equivalent representation of schemes called matrix schemes is introduced. This representation is structurally closer to FSA, and it is proven that the equivalence problem in a subclass of program models represented as matrix models is reducible to one in FSA. The algorithm that checks an equivalence of FSA is studied and stated in terms of requirements applied to finite execution paths in automata. This results in a more general method which can be applied to other models, and necessary requirements models must satisfy for this method to be applicable are stated. Two models, namely the balanced semigroup model with left cancellation and the monotonous commutative model, are shown to have the equivalence problem decidable by the proposed method.

Keywords: program scheme, algebraic model of program, equivalence checking problem, decision procedure, finite state automaton.

DOI: 10.15514/ISPRAS-2015-27(2)-10

For citation: Podlovchenko R. I. Finite State Automata in the Theory of Algebraic Program Schemata. *Trudy ISP RAN/Proc. ISP RAS*, vol. 27, issue 2, 2015, pp. 161-172 (in Russian). DOI: 10.15514/ISPRAS-2015-27(2)-10.

References

[1]. Podlovchenko R.I. Ierarchiya modeley program [Hierarchy of program models]. Programmirovanije [Programming and Computer Software], 1981, № 2, p. 3-14 (in Russian).

[2]. Lyapunov A.A. O logicheskih shemah program [On the logical program schemata]. Problemy Kibernetiki [Problems of Cybernetics] Mosocow. Physmathgiz, vol. 1, 1958, p. 46-74 (in Russian).

[3]. Glushkov V.M., Letichevsky A.A. Teoriya discretnyh preobrazovateley [Theory of discrete transducers] Izbranniye voprosy algebry i logiki [Selected issues in algebra and logics] Novosibirsk, 1973, p. 5-39 (in Russian).

[4]. Podlovchenko R.I. On one equivalence checking technique for algebraic models of programs Programming and Computer Software, 2011, vol. 37, № 6, p. 292-298.

[5]. Rabin M.O., Scott D. Finite automata and their decision problems. IBM Journal of Research and Development, 1959, vol.3, №2, p. 114-125.

[6]. Rutledge J.D. On Ianovs program schemata. Journal of the ACM, 1964, vol. 11, № 1, p. 1-9.

[7]. Podlovchenko R.I. Equivalent transformations in mathematical models of computations. Tetxbook, Moscow, CMC Faculty, Lomonosov Moscow State University, 2011, 72 p. (in Russian)

[8]. Podlovchenko R.I. On program schemes with commuting and monotone operators. Programming and Computer Software, 2003, vol. 29, № 5, p. 270-276.