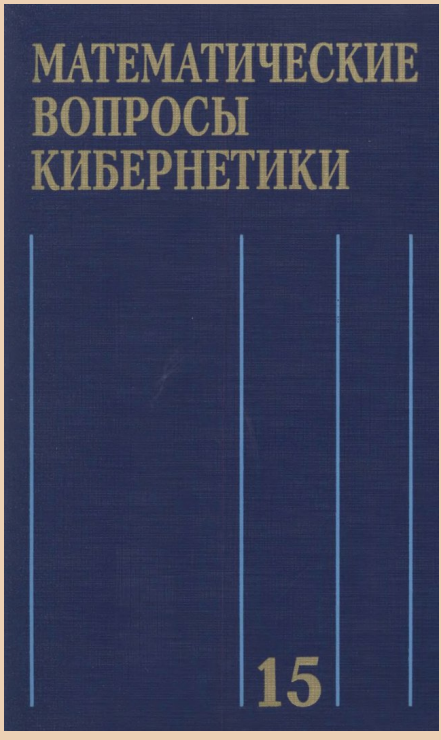
ИПМ им. М.В. Келдыша РАН • Электронная библиотека
Математические вопросы кибернетики • Выпуск 15



Р. И. Подловченко

Алгебраические модели программ и автоматы

Рекомендуемая форма библиографической ссылки:
Подловченко Р. И. Алгебраические модели программ и автоматы // Математические вопросы кибернетики. Вып. 15. — М.: Физматлит, 2006. — С. 199–216. URL: http://library.keldysh.ru/mvk.asp?id=2006-199

АЛГЕБРАИЧЕСКИЕ МОДЕЛИ ПРОГРАММ И АВТОМАТЫ

Р. И. ПОДЛОВЧЕНКО

(МОСКВА)

С появлением программирования, как самостоятельной области знания, возникла задача разработки эквивалентных преобразований программ. Наилучшим ее решением является построение системы преобразований, полной в рассматриваемом классе программ. Однако уже в работе [2] было показано, что богатый по выразительности класс программ представляет собой универсальную модель вычислений, для которой не существует полной системы преобразований.

Это обстоятельство привело к построению моделей программ, объектами которых являются схемы программ, и к переносу на них задачи построения эквивалентных преобразований программ.

Первыми моделями программ стали схемы Ляпунова–Янова [7, 20], исследование которых положило начало теоретическому программированию и, в частности, привело к возникновению теории схем программ.

Логическим развитием моделей Ляпунова–Янова явились алгебраические модели программ, введенные в [8, 10]; проблематика их теории подробно описана в [15]. Эти модели привлекательны тем, что образуют иерархию, в рамках которой можно отбирать модели, аппроксимирующие с желаемой степенью приближения функциональную эквивалентность реальных программ (см. [9]). Интерес к аппроксимирующим моделям основан на том, что всякое эквивалентное преобразование схем из такой модели автоматически является и эквивалентным преобразованием программ, управляющая структура которых совпадает с управляющей структурой этих схем.

Метатеория алгебраических моделей программ занимается отбором подходящих аппроксимирующих моделей (см. [3, 4, 9, 18, 19]), теория — решением проблем, группирующихся вокруг задачи построения систем эквивалентных преобразований схем, полных в рассматриваемых моделях. Методологические находки теории отражены в работах [5, 12, 14–17, 24, 25].

«Возмужание» теории алгебраических моделей программ поставило вопрос о месте, которое алгебраические модели программ занимают среди других моделей вычислений. Данная статья посвящена выявлению связей между первыми, с одной стороны, и такими моделями вычислений, как автоматы (конечные, магазинные, многоленточные, многоголовочные), — с другой. Нас интересует сводимость проблем включения, эквивалентности и пустоты из алгебраических моделей программ в автоматные модели. Поясним, в чем состоят эти проблемы.

Модель вычислений, в самом широком понимании, представляет собой множество конструктивных объектов с прилагаемой к нему процедурой, посредством которой каждому объекту приписывается порождаемое им множество; для объекта g будем обозначать его $Q(g)$.

Пусть M — класс объектов из некоторой модели. Объект g из M называется пустым, если $Q(g)$ пусто. Проблема пустоты в M заключается в отыскании алгоритма, который устанавливает, пуст объект из M или нет. По определению, объект g_1 из M включает в себя объект g_2 из M, если $Q(g_1) \subseteq Q(g_2)$; g_1, g_2 эквивалентны, если $Q(g_1) = Q(g_2)$. Проблема включения (эквивалентности) в M состоит в поиске алгоритма, разрешающего отношение включения (соответственно, эквивалентности) объектов из M.

Очевидно, что при разрешимости проблемы включения в M в нем разрешима и проблема эквивалентности; если M содержит пустой объект, то из разрешимости проблемы эквивалентности в M следует разрешимость в M проблемы пустоты, а из неразрешимости проблемы пустоты — неразрешимость проблем эквивалентности и включения в M.

Решение намеченной нами задачи излагается в §§ 1–5 статьи.

В § 1 вводятся алгебраические модели программ с процедурами; используется «устоявшееся» их определение, которое берется из [19]. Описывается изучаемая далее максимальная модель программ. Основным фактом, устанавливаемым в § 1, является сведение интересующих нас проблем из максимальной модели в принадлежащий ей класс схем (теорема 1).

§ 2 посвящен разрешимости проблемы включения в максимальной модели программ с процедурами (теорема 4); впервые описан разрешающий включение алгоритм, основной компонент которого имеет сложность, квадратичную относительно размеров сравниваемых схем.

В § 3 проблема включения в максимальной модели программ с процедурами сведена к проблеме включения контекстно-свободных грамматик (теорема 6). Из разрешимости первой следует разрешимость второй в полученном классе к.с. грамматик; легко просматривается и разрешающий алгоритм. Отмечаются функциональные характеристики полученного класса к.с. грамматик. Заметим, что ранее в [13] была показана сводимость проблемы эквивалентности в максимальной модели программ с процедурами к проблеме эквивалентности магазинных автоматов.

В § 4 и § 5 рассматриваются схемы программ без процедур.

В § 4 устанавливается взаимосводимость проблем включения в классе бинарных n-ленточных автоматов и специально построенной для него модели программ без процедур; здесь $n \geqslant 2$ (теорема 8). Таким образом, получены модели программ с неразрешимой проблемой включения и разрешимыми проблемами эквивалентности и пустоты.

Наконец, в § 5 для бинарных двухголовочных автоматов построена модель программ без процедур такая, что проблема включения для этих автоматов сводится к проблеме включения в построенной модели (теорема 11). Из этого результата вытекает, что в построенной модели неразрешима проблема пустоты, следовательно, неразрешимы и проблемы эквивалентности и включения.

В итоге: полученные в статье результаты свидетельствуют о глубокой связи между алгебраическими моделями программ и другими моделями вычислений и ставят вопрос о дальнейшем ее изучении.

§ 1. Алгебраические модели программ с процедурами и максимальная модель

Введем понятие схемы программ с процедурами, являющейся объектом алгебраической модели программ. Краткости ради, будем называть ее схемой программы.

Схема программы строится над *базисом* B, состоящим из четырех конечных и непересекающихся алфавитов: Y, C, R и P. Элементы первых трех называются символами, соответственно — операторными, вызова и возврата; элементы алфавита P — логическими переменными; каждая из них принимает значения 0 и 1.

Структура схемы — это конечный ориентированный граф, размеченный над базисом B и построенный по следующим правилам. Граф распадается на подграфы с непересекающимися множествами вершин. Один из подграфов называется *главным*; в нем выделены две вершины: *вход* — вершина без заходящих в нее дуг и с одной исходящей и *выход* — вершина без исходящих из нее дуг. В любом неглавном подграфе тоже выделены две вершины — *инициальная* и *финальная*. Всякая вершина, кроме входа, выхода и финальных, имеет один из четырех типов, называемых *преобразователь*, *вызов*, *возврат* и *распознаватель*. Из распознавателя исходят две дуги, помеченные числами 0 и 1 соответственно; другие дуги пометок не имеют. Из преобразователя, вызова и возврата исходит по одной дуге; они помечены символами 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. Такая пара связана с некоторым неглавным подграфом следующим образом. Дуга из вызова ведет в инициальную вершину этого подграфа, а из его финальной вершины исходит дуга, ведущая в соответствующий вызову возврат, и это — единственная заходящая в него дуга. Из финальной вершины иных дуг, кроме описанных, т. е. ведущих в возвраты, не исходит. Пара вершин вызов — возврат и связанный с нею подграф называются *инцидентными* друг другу. Всякой паре вершин вызов–возврат присвоен номер, отличный от номеров других пар. Дуги, исходящие из вершин, отличных от вызова, ведут в вершины того же подграфа, которому принадлежит их начало.

На рис. 1 изображена схема программы, построенная над базисом B в предположении, что

$$y_1, y_2, y_3, y_4, y_5 \in Y,$$

$$c_1, c_2, c_3 \in C, \quad r_1, r_2, r_3 \in R,$$

$$p_1, p_2 \in P.$$

Обозначения вершин схемы опущены.

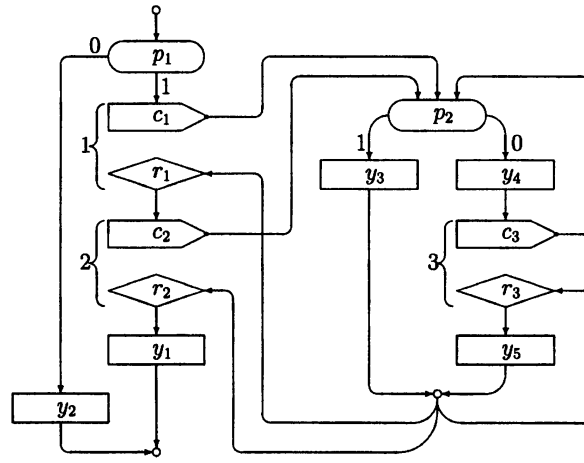


Рис. 1

Перейдем к описанию функционирования схемы над базисом B. Оно осуществляется на *функции разметки*, построенной над базисом B и, по определению, отображающей множество H, где

$$H = (Y \cup C \cup R)^*,$$

в множество X, где

$$X = \{x \mid x\colon P \to \{0, 1\}\}.$$

Множество всех функций разметки над B обозначаем $\mathscr{L}$. Слово из H называем *цепочкой*.

Пусть G — схема над B, и $\mu \in \mathscr{L}$.

Выполнение G на функции μ, $\mu \in \mathscr{L}$, состоит в путешествии по схеме G, идущем в направлении дуг схемы и сопровождающемся построением цепочки. Для однозначности выбора пути, кроме μ, используется магазин, в который загружаются номера вызовов в схеме. Путешествие начинается по дуге, исходящей из входа, при пустом магазине и пустой цепочке. Пусть при путешествии по схеме мы подошли к вершине V с текущей цепочкой h и некоторым состоянием магазина. Тогда различаем следующие случаи.

1. V — преобразователь с сопоставленным ему символом y из Y. При переходе через V к цепочке h справа приписывается символ y, состояние магазина не меняется. Путешествие продолжается по единственной дуге, исходящей из V.
2. V — распознаватель с сопоставленной ему переменной p из P. При переходе через V состояние магазина и текущая цепочка h не меняются, путешествие продолжается по дуге, исходящей из V и помеченной значением $\mu h(p)$.
3. V — вызов, помеченный символом c из C и имеющий номер n. При переходе через V в магазин заносится номер n, к текущей цепочке h справа приписывается символ c, путешествие продолжается по единственной дуге, исходящей из V и ведущей в инициальную вершину подграфа, инцидентного вызову V.
4. V — финальная вершина. При переходе через V с макушки магазина, который заведомо не пуст, снимается номер, и путешествие продолжается по дуге, ведущей к возврату с этим номером.
5. V — возврат с символом r из R. При переходе через V к текущей цепочке h справа приписывается символ r, состояние магазина не меняется, путешествие продолжается по единственной дуге, идущей из V.
6. V — выход. Путешествие заканчивается. В этом случае говорим, что схема G остановилась на μ, и построенную цепочку h называем *результатом выполнения* G на μ.

Путь, пройденный в схеме при ее выполнении на μ, именуем *прокладываемым в схеме функцией* μ. Если путешествие в схеме G бесконечно, то результат ее выполнения на μ считается неопределенным. Таким образом, схемой G реализуется частичная функция, отображающая $\mathscr{L}$ в H.

Эквивалентность схем над базисом B определяется двумя параметрами: эквивалентностью ν в множестве H и подмножеством L множества $\mathscr{L}$ функций разметки. Она вводится так. Схемы G_1, G_2 являются (ν, L)-*эквивалентными* в том и только в том случае, если, какой бы ни была функция из L, всякий раз, как на ней останавливается одна из схем G_1, G_2, останавливается и другая, и результаты их выполнения на этой функции разметки — это ν-эквивалентные цепочки.

Множество всех схем над B, в котором введена (ν, L)-эквивалентность схем, называется (ν, L)-*моделью программ над* B. Определенные таким образом модели программ именуются *алгебраическими*.

Все алгебраические модели программ над B используют общее множество схем над B. Говорят, что одна модель *аппроксимирует* другую, если из эквивалентности схем в первой вытекает их эквивалентность во второй.

Отношение аппроксимации превращает множество алгебраических моделей программ над B в верхнюю полурешетку. Ее наибольшей верхней гранью является модель, в которой эквивалентность схем индуцируется тождеством в H и всем множеством $\mathscr{L}$. Эта модель называется *максимальной над* B. Именно она и рассматривается в ближайших параграфах статьи.

Исходной при исследовании максимальной модели над B служит теорема 1.

Теорема 1. *Проблемы включения, эквивалентности и пустоты в максимальной модели над B сводятся к одноименным проблемам в классе принадлежащих модели u-схем.*

Доказательство теоремы дается в [11] и воспроизводиться здесь не будет. Но, поскольку далее рассматриваются u-схемы, определим их. Это потребует понятий: маршрут в схеме, маршрут через схему, полное дерево распознавателей, пустой цикл.

Пусть G — схема над B. Назовем в ней *опорной* вершину типа вход, выход, вызов, возврат. Рассмотрим путь w из одной опорной вершины схемы G в другую, и в нем — какое-либо вхождение возврата. Этому вхождению поставим в соответствие такое вхождение в w вызова, которое удовлетворяет требованию: в w между двумя обсуждаемыми вхождениями имеется столько же вхождений вызовов, сколько вхождений возвратов. Очевидно, что искомого вхождения вызова может и не быть, но, если оно имеется, то единственное. Если w обладает свойством: он начинается во входе схемы, в нем всякому вхождению возврата соответствует вхождение вызова, и этот вызов парен данному возврату, то w назовем *маршрутом* в G. Маршрут, идущий в ее выход, называется *маршрутом через схему.*

Фрагмент схемы, изображенный на рис. 2, в предположении, что

$$P = \{p_1, \ldots, p_m\}, \quad m \geqslant 1,$$

называется *полным деревом распознавателей.* Число выходящих дуг фрагмента равно 2^m. Все входящие во фрагмент дуги оканчиваются в распознавателе с переменной p_1.

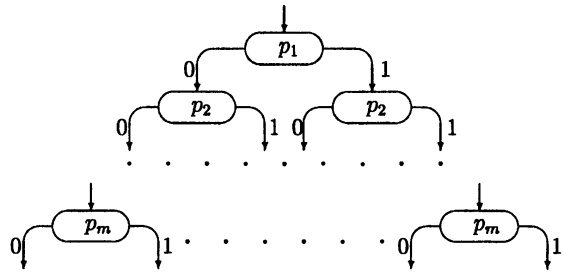


Рис. 2

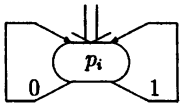


Рис. 3

Фрагмент схемы, изображенный на рис. 3, называется *пустым* циклом. Здесь двойной стрелкой изображается множество дуг, входящих в распознаватель извне; оно может быть и пустым.

Схема G над B называется *u-схемой*, если ее структура удовлетворяет следующим требованиям:

- из любой вершины типа вход, преобразователь, возврат дуга ведет в корень своего полного дерева распознавателя (о последнем говорим как о произрастающем из данной вершины);
- инициальной вершиной любого подграфа схемы является корень полного дерева распознавателей; в нем могут оканчиваться только дуги из вызовов (говорим, что дерево произрастает из этих вызовов);
- выходящая дуга любого дерева распознавателей оканчивается либо в вершине типа преобразователь, вызов, выход схемы, финальная вершина, либо в распознавателе пустого цикла, и тогда последний имеет единственную входящую извне дугу;
- иных распознавателей, кроме упомянутых выше, нет;
- любая вершина типа преобразователь, вызов, возврат лежит на маршруте через схему.

Отметим полезное в будущем свойство u-схемы.

Утверждение 1. *Любой маршрут в u-схеме можно продолжить до маршрута через нее.*

Заметим следующее. Частным случаем схемы программ над базисом B является тот, когда в ней отсутствуют вызовы и соответственно возвраты.

Такую схему будем называть *схемой программ без процедур* и считать, что она *строится над базисом, состоящим только из* Y *и* P.

Теорема 1, естественно, проецируется на схемы программ без процедур.

§ 2. Разрешимость проблемы включения в максимальной модели

Здесь мы установим, что проблема включения разрешима в максимальной модели, и опишем алгоритм, разрешающий ее в классе u-схем из этой модели.

Включение схемы G_1 в схему G_2 будем обозначать как $G_1 \sqsubseteq G_2$.

Начнем с введения понятий, используемых в теореме 2, которая трактует отношение включения схем в виде требований, предъявляемых к маршрутам в них.

Условимся каждую ветвь полного дерева распознавателей маркировать своим элементом x из X, удовлетворяющим требованию: каким бы ни был принадлежащий ветви распознаватель, если ему сопоставлена переменная p из P, то из него ветвь идет по дуге с меткой $x(p)$. Так промаркированную ветвь именуем *α-ветвью.*

Маршрут в u-схеме, завершающийся либо опорной вершиной, либо преобразователем, либо пустым циклом, назовем *закрытым.* Пусть w — закрытый маршрут. Будем изображать его в виде

$$v_0 \xrightarrow{x_0} v_1 \xrightarrow{x_1} \ldots v_{k-1} \xrightarrow{x_{k_1}} v_k, \qquad k \geqslant 0,$$

где v_0 — вход схемы, v_k — концевая вершина маршрута w, а символом $\xrightarrow{x_i}$ обозначена x_i-ветвь, которая произрастает из вершины v_i, ведет в вершину v_{i+1} и находится на маршруте w; здесь $i = 0, 1, \ldots, k-1$. Говорим, что v_i *запоминает i-ю позицию в w*; число k называем *нормой маршрута w* и обозначаем $\|w\|$.

Пусть b_i — символ, сопоставленный вершине v_i маршрута w, отличной от выхода и пустого цикла в случае, когда $i = k$. *Цепочкой, несомой маршрутом w*, назовем слово

$$b_1 b_2 \ldots b_{k-1},$$

если v_k — выход или пустой цикл, и слово

$$b_1 b_2 \ldots b_{k-1} b_k$$

в противном случае.

Закрытые и равновеликие по норме маршруты w_1, w_2 в u-схемах G_1, G_2 соответственно назовем *параллельными*, если любая пара находящихся на них и равноудаленных от их начала ветвей деревьев распознавателей маркирована общим для этой пары символом из X.

Теорема 2. *Пусть G_1, G_2 — u-схемы из максимальной модели над B. Отношение*

$$G_1 \sqsubseteq G_2 \tag{1}$$

имеет место тогда и только тогда, когда, каким бы ни был закрытый маршрут в схеме G_1, существует параллельный ему маршрут в схеме G_2, обладающий свойством: если первый завершается не в пустом цикле, то второй завершается в вершине того же типа, что и первый, и несет ту же цепочку, что и он.

Доказательство. Достаточность требования теоремы очевидна. Установим необходимость его.

Пусть имеет место (1), и w_1 — закрытый маршрут в G_1. Индукцией по норме маршрута w_1 покажем, что для него выполнено требование теоремы.

Базис индукции. Пусть $\|w_1\| = 1$.

Если w_1 завершается в выходе схемы G_1, то в силу (1) требование теоремы выполнено. Если w_1 завершается в вершине типа преобразователь, вызов, возврат, то, по утверждению 1, маршрут w_1 можно продолжить до маршрута через схему, и тогда опять-таки в силу (1) требование теоремы выполнено.

Шаг индукции. Предположим, что для маршрута w_1 с нормой, не меньшей 1, требование теоремы выполнено, и w_1 не завершается в пустом цикле. Продолжим w_1 до маршрута w_1' такого, что $\|w_1'\| = \|w_1\| + 1$. Для w_1' применимо то же рассуждение, что и в случае базиса индукции, т. е. для w_1' выполнено требование теоремы.

Итак, необходимость этого требования установлена, и вместе с этим теорема доказана.

Размером и-схемы из максимальной модели над B назовем число вершин в ней, имеющих тип: преобразователь, вызов, возврат.

Теорема 3. *Для и-схем G_1, G_2 из максимальной модели над B требование теоремы 2 выполнено, если оно выполнено для закрытых маршрутов схемы G_1, норма которых не превышает числа $n_1 n_2$, где n_i — размер схемы G_i, $i = 1, 2$.*

Доказательство. Предположим, что посылка теоремы верна, и рассмотрим маршрут w_1 в схеме G_1 такой, что $\|w_1\| > n_1 n_2$. Покажем, что в схеме G_2 имеется маршрут w_2, параллельный маршруту w_1 и удовлетворяющий требованию теоремы 2.

Доказательство этого осуществляется техникой следов.

Пусть w_1^0 — подмаршрут маршрута w_1, имеющий норму, равную $n_1 n_2$. Из посылки теоремы следует существование в G_2 маршрута w_2^0, параллельного маршруту w_1^0 и обладающего свойствами, зафиксированными теоремой 2.

В маршрутах w_1^0, w_2^0 имеется хотя бы одна пара позиций, которые заняты одной и той же парой вершин. Пусть j_1, j_2, где $j_1 < j_2$, — номера самых последних позиций с таким свойством, и u_1, u_2 — повторяющаяся на них пара вершин, где u_i принадлежит G_i, $i = 1, 2$.

Возможны случаи:

1) u_1, u_2 — вершины типа преобразователь, вызов;

2) u_1, u_2 — вершины типа возврат.

В каждом из них выполним сокращение маршрутов w_1^0, w_2^0 до параллельных маршрутов, сохраняющих свойство, зафиксированное теоремой 2.

В случае 1) выбросим из w_1^0, w_2^0 отрезки, идущие от позиции j_1 до позиции j_2 включительно. В случае 2) предварительно отыщем в w_1^0, w_2^0 позиции i_2, i_1, где $i_2 < i_1$, удовлетворяющие требованию: они заняты вызовами, парными возвратам u_1, u_2; и в каждом из интервалов $[i_2, j_2]$, $[i_1, j_1]$ число вызовов совпадает с числом возвратов. После этого выбросим из w_1^0, w_2^0 отрезки: от позиции i_2 до позиции i_1 включительно и от позиции j_1 до позиции j_2 включительно.

Легко увидеть, что описанное сокращение маршрутов w_1^0, w_2^0 действительно дает маршруты, параллельные и обладающие свойством, зафиксированным теоремой 2.

По выполнении этой операции маршрут w_1 перешел в новый закрытый маршрут уменьшенной нормы. Далее описанная процедура повторяется до тех пор, пока w_1 не укоротится до маршрута нормы, не превышающей числа $n_1 n_2$. Тут вступает в силу посылка теоремы, и фиксируется параллельный ему маршрут в G_2, обладающий требуемыми свойствами. Наконец, корректными вставками выброшенных интервалов восстанавливается исходный маршрут w_1 и синхронно с этим строится требуемый маршрут w_2.

Теорема 3 доказана.

Из теорем 1–3 следует теорема 4.

Теорема 4. *В максимальной модели программ над B разрешима проблема включения, следовательно, разрешимы проблемы эквивалентности и пустоты.*

Доказательство справедливости теоремы 4 очевидно.

Теоремы 2 и 3 позволяют придти к утверждению 2.

Утверждение 2. *Отношение включения для u-схем G_1, G_2 из максимальной модели над B распознаваемо со сложностью $O(n_1 n_2)$, где n_i — размер схемы G_i, $i = 1, 2$.*

Опишем теперь алгоритм, распознающий отношение включения u-схем G_1, G_2 из максимальной модели программ над B.

Алгоритм строит серию таблиц, в которых, кроме ведущего столбца, имеется ровно по 2^m столбцов, и каждый из них маркирован своим элементом x из X. Элементами таблицы являются пары вершин (v_1, v_2), где v_1 принадлежит схеме G_1, v_2 — схеме G_2, и типы этих вершин следующие: вход, выход, преобразователь, вызов, возврат, финальная вершина, пустой цикл.

Первым элементом ведущего столбца является пара, которая состоит либо из входов схем, либо из вызовов, либо из возвратов, и в последних двух случаях вершины несут общую метку. Остальные элементы ведущего столбца состоят из одинаково помеченных преобразователей.

Пара (v_1, v_2), находящаяся не в ведущем столбце, считается допустимой, если либо v_1 — это пустой цикл, либо вершина v_1 отлична от пустого цикла, и тогда v_2 однотипна с v_1 и в случае, когда v_1, v_2 помечены, они несут общую метку.

При построении таблиц используется понятие x-преемника вершины v типа вход, вызов, возврат, преобразователь; так называется вершина, которой заканчивается x-ветвь дерева распознавателей, произрастающего из вершины v.

Перейдем к построению таблиц.

Первой строится таблица, в которой ведущий столбец начинается парой, состоящей из входов схем G_1, G_2. Пусть при ее построении уже заполнены строки, предшествующие строке с ведущей парой (v_1, v_2), и заполняется позиция в этой строке, принадлежащая столбцу с маркером x. Сначала находим x-преемников вершин v_1, v_2; пусть это — v_1', v_2' соответственно. Если пара (v_1', v_2') — недопустимая, то алгоритм объявляет, что G_1 не включается в G_2, и прекращает свою работу.

Предположим, что (v_1', v_2') — допустимая пара. Тогда различаем случаи:

1) (v_1', v_2') состоит из преобразователей; эта пара вносится в ведущий столбец, если ее там не было;
2) (v_1', v_2') состоит из вызовов; тогда отыскиваются возвраты, парные вызовам v_1', v_2'; пусть это — вершины v_1'', v_2'' соответственно; если v_1'', v_2'' несут различные метки, то алгоритм объявляет, что G_1 не включается в G_2, и прекращает свою работу; в противном случае он проверяет, имеются ли таблицы, в которых ведущие столбцы начинаются парами (v_1', v_2'), (v_1'', v_2''), и, если их нет, то заводит такие таблицы.

Таблица считается построенной, если в ней заполнены все позиции.

Работа алгоритма с таблицами, ведущие столбцы которых начинаются парами, состоящими из вызовов или возвратов, проводится так же, как только что описанная.

И только, если все таблицы построены, алгоритм объявляет, что $G_1 \sqsubseteq G_2$.

Корректность предложенного алгоритма следует из теорем 2 и 3.

Обратимся к схемам программ без процедур, принадлежащим максимальной модели над B. Будучи спроецирован на них, описанный выше алгоритм строит только первую таблицу.

§ 3. Сведение проблемы включения в максимальной модели программ к проблеме включения для контекстно-свободных грамматик

Для решения задачи, сформулированной в заголовке параграфа, прежде всего, нужно сопоставить всякой схеме из максимальной модели над B порождаемый ею язык. Сделаем это.

Слово над алфавитом $(Y \cup \{y_0\}) \times X$, где y_0 — добавочный к Y символ, назовем *конфигурацией*, если оно удовлетворяет требованию: первая его буква содержит y_0, и никакая другая не содержит y_0.

Пусть G — u-схема из максимальной модели над B, и w — маршрут через G, имеющий вид

$$v_0 \xrightarrow{x_0} v_1 \xrightarrow{x_1} \ldots v_k \xrightarrow{x_k} v_{k+1}, \qquad k \geqslant 0.$$

Если вершине v_i, $i = 1, \ldots, k$, приписан символ b_i, то слово

$$(y_0, x_0)(b_1, x_1) \ldots (b_k, x_k)$$

назовем *конфигурацией, порожденной маршрутом* w.

Множество конфигураций, порожденных всеми маршрутами через G, обозначим $K(G)$.

Справедлива теорема 5.

Теорема 5. *Для произвольных u-схем G_1, G_2 из максимальной модели над B выполняется*

$$G_1 \sqsubseteq G_2 \iff K(G_1) \subseteq K(G_2).$$

Доказательство теоремы опускаем, так как оно технически повторяет проведенное в [13] доказательство совпадения множеств $K(G_1)$, $K(G_2)$ при эквивалентности u-схем G_1, G_2 и обратное этому утверждение.

Обратимся теперь к контекстно-свободным (к.с.) грамматикам.

Напомним (см., например, [1]), что к.с. грамматика Г над конечным алфавитом Σ представляет собой тройку

$$(S, s_0, \pi),$$

где

S — конечный алфавит символов, называемых нетерминалами;

s_0 — начальный нетерминал, $s_0 \in S$;

π — конечный набор продукций, т.е. конструкций вида $\alpha \leftarrow \beta$, где $\alpha \in S$, $\beta \in (S \cup \Sigma)^*$.

Язык, порождаемый грамматикой Г, обозначим $l(\Gamma)$.

Имеет место

Теорема 6. *Проблема включения в максимальной модели программ над B сводится к проблеме включения для к.с. грамматик над алфавитом* $(Y \cup \{y_0\}) \times X$.

Доказательство. На основании теоремы 1 достаточно провести его для u-схем, принадлежащих максимальной модели над B.

Рассматривая u-схемы, условимся помечать символом y_0 вход схемы.

Покажем, что для u-схемы G можно построить грамматику Г над алфавитом $(Y \cup \{y_0\}) \times X$ такую, что

$$K(G) = l(\Gamma). \tag{2}$$

Это и будет доказательством теоремы.

Грамматика Г строится следующим образом.

В качестве нетерминалов берутся образы всех вершин схемы G, имеющих следующий тип: вход, преобразователь, вызов, возврат. Образ вершины v обозначается S_v.

Начальным считается нетерминал S_{v_0}, где v_0 — вход схемы G.

Для всякого нетерминала S_v в грамматику Г вносится серия продукций, руководствуясь следующими правилами.

Пусть b — символ, сопоставленный вершине v, и x — произвольный элемент из X. Предположим, что x-преемник вершины v отличен от пустого цикла и обозначим его u. Тогда в Г помещаем: продукцию

$$S_v \leftarrow (b, x) S_u,$$

если u — преобразователь или возврат;

$$S_v \leftarrow (b, x) S_u S_{u'},$$

если u — вызов, а u' — парный ему возврат;

$$S_v \leftarrow (b, x),$$

если u — выход или финальная вершина.

Кроме описанных, других продукций в Г нет.

Проверку того, что для Г действительно выполняется (2), опускаем ввиду ее легкости.

Теорема 6 доказана.

Обозначим T множество к.с. грамматик, построенных для u-схем из максимальной модели над B согласно алгоритму, изложенному в теореме 6.

Следствие теорем 4 и 6. *В множестве T к.с. грамматик над $(Y \cup \{y_0\}) \times X$ разрешима проблема включения, следовательно, разрешимы проблемы эквивалентности и пустоты.*

Отметим, что множество к.с. языков, порождаемых грамматиками из T,

а) замкнуто по операции пересечения языков;

б) незамкнуто по операции сложения языков.

Оба утверждения легко доказуемы и являются важными характеристиками обсуждаемого множества.

В заключение обратимся к случаю, когда изучаются схемы программ без процедур. Легко усмотреть, что следствием теоремы 6 является утверждение 3.

Утверждение 3. *Если G — и-схема программы без процедур, то $K(G)$ — регулярный язык.*

§ 4. Многоленточные автоматы и схемы программ без процедур

Основная цель данного параграфа — доказать теорему 8, утверждающую существование такой модели программ без процедур, для которой проблема включения в ней и проблема включения в множестве n-ленточных бинарных автоматов, где $n \geqslant 2$, взаимосвязаны.

Сначала дадим используемое нами определение n-ленточного автомата над алфавитом Σ; оно взято из [22].

Такой автомат задается множеством Q состояний автомата, в котором выделены

$\bar{q}$ — инициальное состояние;

a — финальное состояние; $a \neq \bar{q}$;

r — мертвое состояние; $r \neq a$, $r \neq \bar{q}$.

Множество Q представлено в виде

$$Q = Q' \cup \{a, r\}.$$

Задается функция δ: $Q' \times \Sigma \to Q$ переходов автомата из состояния в состояние под воздействием считанного с ленты символа из Σ.

Множество Q' разбивается на n непересекающихся подмножеств $Q_1, Q_2, \ldots, Q_n$.

В состав автомата входят n лент, нумеруемых числами $1, 2, \ldots, n$. Лента состоит из однородных ячеек, в каждой из которых может быть записан символ из Σ. Лента ограничена слева и не ограничена справа; все ее ячейки заполнены.

Каждой ленте приписывается своя считывающая головка; ей присваивается номер ее ленты. Считывающая головка в любой момент работы автомата обозревает некоторую ячейку своей ленты и обладает способностью сдвинуться на следующую ячейку ленты.

Функционирование автомата осуществляется на заданном наборе лент.

Первый шаг работы автомата происходит в условиях, когда он находится в состоянии $\bar{q}$, и каждая головка обозревает первую ячейку своей ленты.

Опишем общий шаг работы. Пусть автомат находится в состоянии q, где $q \in Q$, и каждая головка обозревает некоторую ячейку на своей ленте. Тогда

1) если $q \in \{r, a\}$, то данный шаг считается завершенным, ибо автомат останавливается;
2) пусть $q \in Q_j$, $j \in \{1, \ldots, n\}$, и σ — символ, обозреваемый j-й головкой; предположим, что

$$\delta(q, \sigma) = q';$$

в этом случае j-я головка считывает символ σ и передвигается на следующую ячейку, а автомат переходит в состояние q'.

Говорят, что *автомат принимает заданный набор лент*, если он при работе на нем останавливается в состоянии a. Результатом работы считаем кортеж

$$< w_1, w_2, \ldots, w_n >,$$

где w_i — слово, считанное с i-й ленты; $i = 1, \ldots, n$.

Пусть A_1, A_2 — автоматы над Σ. Согласно общему определению, данному во введении для любой модели вычислений, *автомат A_1 включается в автомат A_2* (обозначается это в виде $A_1 \sqsubseteq A_2$), если, каким бы ни был набор лент, всякий раз, как он принимается автоматом A_1, он принимается и автоматом A_2, и при этом с тем же результатом, что и у автомата A_1.

Имеет место теорема 7.

Теорема 7. *Для n-ленточных автоматов с $n \geqslant 2$, рассматриваемых над не менее, чем двубуквенным алфавитом, неразрешима проблема включения и разрешима проблема эквивалентности.*

Первое из утверждений теоремы 7 установлено в [23], второе — в [21].

Используем факт, установленный в [22], формулируя его как лемму 1.

Лемма 1. *Проблема включения для n-ленточных автоматов с $n \geqslant 2$, рассматриваемых над не менее, чем двубуквенным алфавитом, сводится к проблеме включения бинарных n-ленточных автоматов.*

На основании леммы 1 нами рассматриваются бинарные n-ленточные автоматы, причем используется алфавит $\{0, 1\}$. Их множество обозначаем $\mathfrak{M}_n$.

Как было отмечено, основной в данном параграфе является теорема 8.

Теорема 8. *Для всякого n, где $n \geqslant 2$, можно построить модель M_n программ без процедур, обладающую свойством: проблемы включения в $\mathfrak{M}_n$ и M_n взаимосводимы.*

Доказательство теоремы 8 проведем в случае $n = 2$, ибо для больших значений n используются те же идеи, но возникают дополнительные трудности чисто технического порядка.

Сначала мы опишем модель M_2. Затем покажем, что в M_2 имеются подклассы M_2^x, $x \in X$, такие, что проблема включения в M_2 разрешима тогда и только тогда, когда она разрешима в каждом из подклассов M_2^x.

Наконец, далее опишем алгоритм ρ, который по схеме из M_2^x, $x \in X$, строит автомат из $\mathfrak{M}_2$ и делает это так, что проблема включения в M_2^x разрешима тогда и только тогда, когда она разрешима в множестве построенных автоматов.

И так как последнее составляет все множество $\mathfrak{M}_2$, то этим будет доказана теорема 8.

Итак, определим модель M_2. Она строится над базисом Y, P, где

$$Y = \{y_1, y_2\}, \qquad P = \{p_1, p_2\}.$$

Эквивалентность в Y^* определяется следующим образом: цепочки из Y^* эквивалентны тогда и только тогда, когда совпадают их проекции на символ y_1 и соответственно — на символ y_2.

Допустимой считаем функцию разметки μ, удовлетворяющую требованию: для любой цепочки h из Y^*

$$i \neq j \Longrightarrow \mu h y_i(p_j) = \mu h(p_j), \qquad i = 1, 2; \quad j = 1, 2. \tag{3}$$

Построение класса M_2^x осуществляется в четыре этапа. Опишем их.

Этап 1. Применяя алгоритм, упомянутый в § 1, сведем проблему включения в M_2 к проблеме включения u-схем, принадлежащих M_2.

Этап 2. Применяя алгоритм, изложенный в [12], сведем проблему включения u-схем из M_2 к проблеме включения принадлежащих M_2 схем, названных маркированными. По определению, *маркированной* называется u-схема, всякий преобразователь которой маркируется элементом x из X,

руководствуясь тем, что он может быть только x-преемником других вершин схемы.

Этап 3. Всякой маркированной схеме G из M_2 сопоставим набор маркированных схем G_x, где x пробегает все множество X. Схема G_x строится по схеме G сохранением в ней только тех маршрутов через нее, которые начинаются x-ветвью произрастающего из входа дерева распознавателей; при этом ветви деревьев, не принадлежащие таким маршрутам, направляются в пустой цикл. Если x-ветвь из входа ведет в пустой цикл, то G_x полагается пустой схемой.

Очевидным является утверждение: для любых маркированных схем G_1, G_2 из M_2

$$G_1 \sqsubseteq G_2 \iff \forall x \in X \ [G_{1x} \sqsubseteq G_{2x}].$$

Этап 4. Трансформируем маркированную схему G_x, $x \in X$, в так называемую приведенную схему $G_{x'}$.

Для этого проанализируем выполнение схемы G_x на допустимых функциях разметки.

Предположим, что на какой-либо из таких функций выполнение схемы G_x привело к преобразователю v с символом y_i, $i = 1, 2$, и маркером $\bar{x}$. Тогда дальнейший путь из v пойдет по ветви дерева, в которой переменная p_{3-i} равна числу $\bar{x}(p_{3-i})$; это диктуется условием (3). Преемники вершины v по ветвям дерева, соответствующим другому значению переменной p_{3-i}, не достижимы, какой бы ни была рассматриваемая допустимая функция разметки. В связи с этим, полное дерево распознавателей, произрастающее из v, можно заменить деревом, вид которого представлен на рис. 4. Это будет эквивалентным преобразованием схемы G_x. Выполнив его для каждого ее преобразователя, мы получим, в общем случае, схему, содержащую фрагменты, недостижимые маршрутами, и фрагменты, из которых ориентированными путями недостижим выход схемы. Фрагменты первого типа устраним, а дуги, ведущие во фрагменты второго типа, направим в пустой цикл, введя его в случае, когда он отсутствовал в схеме, после чего тоже устраним и эти фрагменты. Описанные операции с фрагментами являются эквивалентными преобразованиями схемы. Результатом всех проведенных трансформаций схемы G_x будет схема, эквивалентная ей. Она и называется приведенной и обозначается G'_x.

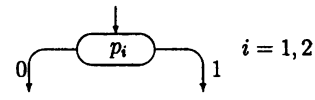


Рис. 4

По определению, множество M_2^x состоит из всех приведенных схем G'_x.

Таким образом, проблема включения в M_2 сведена к проблеме включения в каждом из классов M_2^x, $x \in X$.

Опишем теперь алгоритм ρ.

Пусть на вход его поступила схема G из M_2^x. Если в G отсутствуют преобразователи, то, по определению, $\rho(G)$ состоит только из инициального и финального состояний, причем все переходы из первого ведут во второй, и инициальному состоянию сопоставлена лента 1. В остальных случаях всякому преобразователю v схемы G алгоритм ρ ставит в соответствие состояние q_v автомата $\rho(G)$, выходу схемы G — финальное состояние автомата $\rho(G)$, обозначаемое a, пустому циклу, если он имеется в G, — мертвое состояние. Инициальным в $\rho(G)$ назначается состояние, соответствующее преобразователю, который является x-преемником входа схемы G. Если преобразователю v сопоставлен символ y_i, $i = 1, 2$, то в состоянии q_v автомат $\rho(G)$ будет считывать i-ю ленту.

Функция переходов δ автомата $\rho(G)$ строится следующим образом.

Пусть преобразователю v схемы G сопоставлен символ y_i, $i = 1, 2$; обозначим v_0, v_1 вершины, достижимые из v по дуге произрастающего из v дерева, помеченной числом 0 и соответственно числом 1. Тогда

$$\delta(q_v, 0) = q_{v_0}, \qquad \delta(q_v, 1) = q_{v_1}.$$

Автомат $\rho(G)$ построен.

Покажем теперь, что что для любых G_1, G_2 из M_2^x выполняется

$$G_1 \sqsubseteq G_2 \iff \rho(G_1) \sqsubseteq \rho(G_2). \tag{4}$$

Для этого введем отображение λ множества, состоящего из наборов двух лент над алфавитом $\{0, 1\}$, в множество допустимых функций разметки.

Пусть набор ω состоит из лент

$$\varepsilon_{i_1} \varepsilon_{i_2} \ldots \varepsilon_{i_k} \ldots,$$

где $i = 1, 2$. Функцию разметки $\lambda(\omega)$, обозначаемую как μ, определим следующим образом:

1) $\mu e = x$;

2) пусть h из Y^* содержит l_i вхождений символа y_i, $i = 1, 2$; тогда

$$\mu h(p_1) = \varepsilon_{1l_1}, \qquad \mu h(p_2) = \varepsilon_{2l_2}.$$

Легко убедиться в том, что μ — допустимая функция.

Отображение λ обладает следующими свойствами.

Пусть автомат $\rho(G)$ принимает набор ω, и при этом в автомате прокладывается путь, проходящий последовательность состояний

$$q_1, q_2, \ldots, q_k, a.$$

Тогда функция $\lambda(\omega)$ прокладывает маршрут w через G, проходящий последовательность вершин

$$v_0, v_1, \ldots, v_k, v_{k+1},$$

где v_0 — вход схемы, v_{k+1} — ее выход, а q_j и v_j для $j = 1, 2, \ldots, k$ связаны равенством

$$q_j = q_{v_j}.$$

Если при этом с i-й ленты, $i = 1, 2$, прочитан кусок размера t_i, то в слове, несомом маршрутом w, имеется в точности t_i вхождений символа y_i.

Заметим теперь, что отображение λ является взаимно-однозначным, и его значениями являются все допустимые функции разметки μ, для которых $\mu e = x$ (на других функциях разметки схемы из M_2^x не останавливаются). Кроме того, индуцируемое λ соответствие между путями через автомат $\rho(G)$ и маршрутами через G является взаимно-однозначным, а все последние прокладываются на допустимых функциях разметки.

Из всего сказанного следует справедливость (4).

Теорема 8 доказана.

Из теорем 7 и 8 следует теорема 9.

Теорема 9. *В модели M_2 программ без процедур неразрешима проблема включения и разрешимы проблемы эквивалентности и пустоты.*

§ 5. Двухголовочные автоматы и схемы программ без процедур

В этом параграфе мы установим сводимость проблемы включения для двухголовочных автоматов к проблеме включения в некоторой модели программ без процедур.

Исходим из определения двухголовочного автомата над алфавитом Σ, данного в [22]. Оно отличается от определения двухленточного автомата только тем, что теперь используется одна лента вместо двух, и обе головки считывают символы с нее. Остальные понятия, относящиеся к структуре автомата и его функционированию, сохраняются.

В [22] доказана теорема 10.

Теорема 10. *Для бинарных двухголовочных автоматов неразрешима проблема пустоты, следовательно, неразрешимы проблемы включения и эквивалентности.*

Общность этого результата следует из леммы 2, установленной в [22].

Лемма 2. *Проблема включения для двухголовочного автомата над произвольным алфавитом Σ, где $|\Sigma| \geqslant 2$, сводится к проблеме включения для бинарных двухголовочных автоматов.*

Обозначим $\mathfrak{M}$ множество бинарных двухголовочных автоматов и докажем теорему 11.

Теорема 11. *Существует модель M программ без процедур такая, что проблема включения в $\mathfrak{M}$ сводится к проблеме включения в M.*

Доказательство. Сначала определим модель M, затем выделим в этой модели класс M_2^x, где $x \in X$, и покажем, что проблемы включения в $\mathfrak{M}$ и M_2^x взаимосводимы.

Модель M схем программ строится над базисом Y, P, где

$$Y = \{y_1, y_2\}, \qquad P = \{p_1, p_2\}.$$

Эквивалентность в Y^* вводится требованием: цепочки h_1, h_2 из Y^* эквивалентны тогда и только тогда, когда совпадают их проекции на символ y_1 и проекции на символ y_2.

Допустимая функция разметки μ определяется следующим образом. Пусть $\|h\|_i$ — число вхождений символа y_i, $i = 1, 2$, в цепочку h, $h \in Y^*$. Тогда для любых h_1, h_2 из Y^*

$$\|h_1\|_i = \|h_2\|_{3-i} \Longrightarrow \mu h_1(p_i) = \mu h_2(p_{3-i}), \qquad i = 1, 2.$$

Модель M описана.

Класс M_2^x описан в § 4. Отметим, что теперь эквивалентность схем из M_2^x принципиально отличается от эквивалентности их в модели M_2, рассмотренной в § 4, так как используется другое множество допустимых функций разметки.

Опишем теперь алгоритм τ, который сводит проблему включения в $\mathfrak{M}$ к проблеме включения в M_2^x (x — фиксированный элемент из X).

Пусть на вход алгоритма τ поступил автомат A, который задается пятеркой

$$(Q, \bar{q}, a, r, \delta),$$

где

$Q = Q' \cup \{a, r\}$;

$\bar{q}$ — инициальное состояние, и $\bar{q} \in Q'$;

a — финальное состояние;

r — мертвое состояние,

и Q' делится на непересекающиеся подмножества Q_1, Q_2. Полагаем, что

$$\delta\colon\ Q' \times \{0, 1\} \to Q,$$

и запишем δ в удобном для нас виде. Пусть

$$Q' = \{q_1, q_2, \ldots, q_N\}, \qquad N \geqslant 1;$$

используем обозначение q_i^j в случае, когда $q_i \in Q_j$ (здесь $i = 1, \ldots, N$ и $j = 1, 2$), и пусть

$$\delta(q_i^j, 0) = r_{i0}, \qquad \delta(q_i^j, 1) = r_{i1}.$$

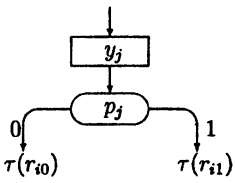


Рис. 5

Автомат A описан.

Алгоритм τ строит по автомату A схему $\tau(A)$ из M_2^x и делает это следующим образом. Состояниям a и r он сопоставляет выход схемы и, соответственно, пустой цикл. Состоянию q_i^j алгоритм τ сопоставляет фрагмент, изображенный на рис. 5. Здесь висящие на дугах обозначения $\tau(r_{i0})$, $\tau(r_{i1})$ отмечают фрагменты, в которые направлены эти дуги; двойной стрелкой изображено множество входящих дуг. Фрагмент, сопоставленный состоянию $\bar{q}$, алгоритм τ объявляет x-преемником входа схемы. Схема $\tau(A)$ построена.

Исследуем функционирование A и $\tau(A)$.

Зададимся лентой ω, имеющей вид бесконечной последовательности

$$\varepsilon_1 \varepsilon_2 \ldots \varepsilon_k \ldots,$$

где $\varepsilon_i \in \{0, 1\}$, $i = 1, 2, \ldots$. Построим по ленте ω функцию разметки $\lambda(\omega)$, обозначив ее μ:

1) $\mu e = x$;

2) для любого h из Y^*, отличного от пустого слова e,

$$\|h\|_j = i \to \mu h(p_j) = \varepsilon_i,$$

где $j = 1, 2$, $i = 1, 2, \ldots$

Очевидно, что μ — допустимая функция.

Нетрудно убедиться в том, что справедливо утверждение: автомат A принимает ленту ω тогда и только тогда, когда схема $\tau(A)$ останавливается на функции $\lambda(\omega)$; при этом, если j-ая головка, $j = 1, 2$, прочитала кусок ленты длины l_j, то цепочка, с которой остановилась схема $\tau(A)$, имеет в точности l_j вхождений символа y_j.

Отсюда уже следует, что для любых A_1, A_2 из $\mathfrak{M}$

$$A_1 \sqsubseteq A_2 \to \tau(A_1) \sqsubseteq \tau(A_2).$$

Отношение

$$\tau(A_1) \sqsubseteq \tau(A_2) \to A_1 \sqsubseteq A_2$$

следует из того, что областью значений отображения λ является множество всех допустимых функций разметки μ, где $\mu e = x$, и само отображение λ обратимо.

Взаимосводимость проблемы включения в $\mathfrak{M}$ и M_2^x устанавливается тем, что алгоритм τ обратим.

Теорема 11 доказана.

В заключение подчеркнем мысль, высказанную во введении: выявленные связи между алгебраическими моделями программ и автоматами выдвигают задачу дальнейшего изучения места, которое занимают первые среди математических моделей вычислений.

СПИСОК ЛИТЕРАТУРЫ

1. Ахо А., Ульман Д. Теория синтаксического анализа, перевода и компиляции. — М.: Мир, 1978.
2. Ершов А. П. Операторные алгоритмы. 2. Описание основных конструкций программирования // Проблемы кибернетики. Вып. 8. — М.: Физматгиз, 1962. — С. 211–233.
3. Захаров В. А. Об отношении аппроксимируемости семантик операторных программ // Вестник МГУ. Вычислительная математика и кибернетика. — 1994. — № 3. — С. 54–60.
4. Захаров В. А. Аппроксимация абстрактных семантик формальными моделями программ // Дискретная математика. — 1998. — Т. 10, вып. 4. — С. 119–141.
5. Захаров В. А. Быстрые алгоритмы разрешения эквивалентности операторных программ на уравновешенных шкалах // Математические вопросы кибернетики. Вып. 7. — М.: Наука, 1998. — С. 257–280.
6. Летичевский А. А. Практические методы распознавания эквивалентности дискретных преобразователей и схем программ // Кибернетика. — 1973. — № 4. — С. 15–26.
7. Ляпунов А. А. О логических схемах программ // Проблемы кибернетики. Вып. 1. — М.: Физматгиз, 1958. — С. 46–74.
8. Подловченко Р. И. Иерархия моделей программ // Программирование. — 1981. — № 2. — С. 3–14.
9. Подловченко Р. И. Полугрупповые модели программ // Программирование. — 1981. — № 4. — С. 3–13.
10. Подловченко Р. И. Рекурсивные программы и иерархия их моделей // Программирование. — 1991. — № 6. — С. 44–51.
11. Подловченко Р. И. Трансформация схем рекурсивных программ в эквивалентные им свободные // Программирование. — 1994. — № 6. — С. 3–22.
12. Подловченко Р. И., Айрапетян М. А. О построении полных систем эквивалентных преобразований схем программ // Программирование. — 1996. — № 1. — С. 3–29.
13. Подловченко Р. И. Абстрактные программы с процедурами и конечные автоматы с магазинами // Интеллектуальные системы. — 1997. — Т. 2, вып. 1–4. — С. 275–295.
14. Подловченко Р. И., Захаров В. А. Полиномиальный по сложности алгоритм, распознающий коммутативную эквивалентность схем программ // Докл. РАН. — 1998. — Т. 362, № 6.
15. Подловченко Р. И. От схем Янова к теории моделей программ // Математические вопросы кибернетики. Вып. 7. — М.: Наука, 1998. — С. 281–302.
16. Подловченко Р. И. Об одном массовом решении проблемы эквивалентных преобразований схем программ. I // Программирование. — 2000. — № 1. — С. 64–77.
17. Подловченко Р. И. Об одном массовом решении проблемы эквивалентных преобразований схем программ. II // Программирование. — 2000. — № 2. — С. 3–11.
18. Подловченко Р. И., Попов С. В. Аппроксимируемость одних моделей программ другими // Вестник МГУ. Вычислительная математика и кибернетика. — 2001. — № 2. — С. 38–46.
19. Подловченко Р. И., Долгих Б. А. Двухступенчатое моделирование программ с процедурами // Математические вопросы кибернетики. Вып. 12. — М.: Физматлит, 2003. — С. 283–300.
20. Янов Ю. И. О логических схемах алгоритмов // Проблемы кибернетики. Вып. 20. — М.: Наука, 1968. — С. 201–216.
21. Harju T., Karhumaki J. The equivalence of multi-tape finite automata // Theor. Comp. Sci. — 1991. — V. 78. — P. 347–355.
22. Luckham D. C., Park D. M., Paterson M. S. On formalized computer programs // J. Computer and System Sci. — 1970. — V. 4, № 3. — P. 220–249.
23. Rabin M. O., Scott D. Finite automata and their decision problems // IBM J. Res. Develop. — 1959. — V. 3, № 2. — P. 114–125.
24. Zakharov V. A. An efficient and unified approach to the decidability of equivalence of propositional program schemes // Lect. Notes Comp. Sci. — 1998. — V. 1443. — P. 246–258.

25. Z a k h a r o v V. A. On the decidability of the equivalence problem for monadic recursive programs // Theoretical Informatics and Applications. — 2000. — V. 34. — P. 157–171.

Поступило в редакцию 20 VIII 2005