УДК 519.71

Р.И. ПОДЛОВЧЕНКО

К ВОПРОСУ О ПОЛИНОМИАЛЬНОЙ СЛОЖНОСТИ ПРОБЛЕМЫ ЭКВИВАЛЕНТНОСТИ В АЛГЕБРАИЧЕСКИХ МОДЕЛЯХ ПРОГРАММ

Ключевые слова: *алгебраическая модель программ, схема программ, эквивалентность, алгоритмическая разрешимость, вычислительная сложность.*

Настоящая статья примыкает к работе [1], в которой установлена разрешимость проблемы эквивалентности схем программ, принадлежащих алгебраическим моделям программ определенного типа, а именно: уравновешенным полугрупповым моделям программ с левым сокращением и разрешимой эквивалентностью операторных цепочек. Алгоритм, разрешающий эквивалентность схем программ, по временной сложности оказался экспоненциальным относительно размеров схем, т.е. малоэффективным в практике программирования.

В связи с этим возникает задача: выделить из найденного множества алгебраических моделей программ такие модели, для которых существует приемлемый по сложности алгоритм, распознающий эквивалентность схем программ.

Решение данной задачи проводится следующим образом. Для множества моделей из [1] конструируется алгоритм, распознающий эквивалентность схем в модели методом, отличающимся от рассмотренного в указанной работе. В основу нового метода положена идея, впервые воплощенная в алгоритме Мура, распознающем эквивалентность конечных автоматов. Эта идея состоит в следующем: просмотр процесса функционирования схемы осуществляется с помощью анализа пролагаемых маршрутов в схемах. Однако сконструированный алгоритм при этом также экспоненциален по сложности. Поэтому выделяется достаточно широкий класс моделей, для которых путем модификации данного алгоритма построен новый алгоритм, распознающий эквивалентность схем программ с полиномиальной сложностью относительно сложности разрешения эквивалентности операторных цепочек. Последнее означает, что в коэффициенты полинома, составляющего оценку сложности алгоритма, входят, в общем случае, оценки сложности алгоритма, распознающего эквивалентность операторных цепочек.

Изложению алгоритмов, разрешающих проблему эквивалентности для моделей программ и представляющих основные результаты проведенного исследования, предшествуют разд. 1 и 2.

В разд. 1 описываются основные понятия, используемые в данной статье. Исходным для этого является следующий факт (теорема 1): в любой алгебраической модели программ проблема эквивалентности сводится к проблеме эквивалентности в множестве принадлежащих модели матричных схем программ; сводящий алгоритм полиномиален по сложности. На этом основании определяются синтаксис и семантика матричных схем, являющихся объектами проводимых исследований. Дается описание уравновешенной полугрупповой модели программ с левым сокращением и формулируется теорема 2 о разрешимости в ней эквивалентности схем программ при условии, что эквивалентность операторных цепочек разрешима.

В разд. 2 приводятся необходимые условия эквивалентности матричных схем, установленные в [1]. Они используются в алгоритме, распознающем эквивалентность матричных схем в уравновешенной полугрупповой модели программ с левым сокращением и разрешимой эквивалентностью операторных цепочек (теорема 3). Доказательство теоремы 3 составляет разд. 3.

В разд. 4 к эквивалентности операторных цепочек предъявляется еще одно требование: обладать свойством частичного правого сокращения. Описывается

© Р.И. Подловченко, 2012

извлекаемое из этого свойства добавочное необходимое условие эквивалентности матричных схем, и с учетом его модифицируется алгоритм, представленный в теореме 3. Новый алгоритм распознает эквивалентность матричных схем в уравновешенной полугрупповой модели программ с левым и частичным правым сокращением и разрешимой эквивалентностью операторных цепочек с приемлемой сложностью (теорема 4).

К основным результатам, изложенным в разд. 3 и 4, относится и методология их получения, с помощью которой из множества уравновешенных полугрупповых моделей программ с левым и частичным правым сокращением можно выделять модели с приемлемой сложностью распознавания эквивалентности схем.

1. ОСНОВНЫЕ ПОНЯТИЯ И ФАКТЫ

Алгебраическая модель программ, введенная в [1], строится над двумя конечными и непересекающимися алфавитами — Y и P. Элементы алфавита Y называются операторными символами, элементы алфавита P — логическими переменными; каждая переменная принимает значение из множества $\{0,1\}$. Объектами модели являются схемы программ. Модель представляет собой множество схем программ над Y, P с введенным на нем отношением эквивалентности схем.

Проблема проверки эквивалентности схем программ в модели, состоящая в построении разрешающего эквивалентность алгоритма, относится к числу фундаментальных. Имеет место следующая теорема.

Теорема 1. Проблема эквивалентности в любой алгебраической модели программ сводится к проблеме эквивалентности в множестве матричных схем, принадлежащих модели; при этом сводящий алгоритм полиномиален по сложности относительно размеров схем.

Доказательство теоремы 1 дано в [1].

Таким образом, поиск приемлемого по сложности алгоритма, разрешающего проблему эквивалентности в алгебраической модели программ, сводится к построению алгоритма, распознающего эквивалентность матричных схем из данной модели. Основываясь на этом, ограничимся описанием матричных схем (в [1] приведено определение схемы программы общего вида).

Синтаксически матричная схема над Y, P представляет собой конечный ориентированный и размеченный граф, вершины которого подразделяются на вход схемы, выход схемы, пустой цикл и преобразователи; при этом вход, выход и пустой цикл присутствуют обязательно. Из любой вершины, кроме выхода и пустого цикла, исходят дуги в количестве, равном числу элементов в множестве X, где $X=\{x \mid x\colon \mathrm{P} \to \{0,1\}\}$. Каждая дуга помечена элементом из X, причем различные дуги несут разные метки. Во входе схемы нет оканчивающихся в нем дуг. Каждый преобразователь в схеме имеет в качестве метки символ из Y.

Далее элементы множества X называются наборами значений логических переменных из P, или, кратко, наборами.

Семантически каждая матричная схема над Y, P является носителем отображения, в общем случае — частичного, множества $\mathcal{L}$, состоящего из всех функций разметки над Y, P, в множество Y^*, состоящее из всех слов над алфавитом Y. Функцией разметки над Y, P называется всюду определенное отображение множества Y^* в множество X. Далее элементы множества Y^* называются операторными цепочками.

Реализуемое матричной схемой отображение осуществляется посредством процедуры ее выполнения на функциях разметки.

Пусть $\mu \in \mathbf{L}$. Процедура выполнения схемы на μ, включающая обход схемы, начинающийся в ее входе, и построение операторной цепочки, осуществляется по следующим правилам.

Пусть v_0 — вход схемы. Тогда по функции μ определяется набор x_0, равный $\mu\Lambda$, где Λ — пустая операторная цепочка, и выход из вершины v_0 совершается по исходящей из нее дуге с меткой x_0.

Допустим, что при обходе достигнута вершина v схемы, h — построенная к этому моменту операторная цепочка. Тогда, если v — преобразователь с меткой y, цепочка h трансформируется в цепочку hy, находится набор $\mu(hy)$ и выход из вершины v совершается по дуге, помеченной этим набором. Если v — выход схемы или пустой цикл, то обход схемы прекращается. В первом случае говорится, что схема остановилась на функции μ с результатом h, во втором — выполнение схемы на μ считается безрезультатным. Таким образом определяется отображение схемой множества $\mathcal{L}$ в множество Y^*.

Как было отмечено, отдельная алгебраическая модель программ определяется введением отношения эквивалентности между схемами программ. Каждое отношение эквивалентности схем индуцируется двумя параметрами: отношением эквивалентности ν в множестве Y^*; подмножеством L множества $\mathcal{L}$.

Опишем его для матричных схем. Матричные схемы G_1, G_2 по определению (ν, L)-эквивалентны тогда и только тогда, когда, какой бы ни была функция μ из L, каждый раз при остановке одной из схем G_1, G_2 на μ останавливается и другая; результатами их выполнения являются ν-эквивалентные операторные цепочки.

Сформулируем теорему, доказанную в [1].

Теорема 2. В уравновешенной полугрупповой модели программ с левым сокращением и разрешимой эквивалентностью операторных цепочек разрешима проблема эквивалентности в множестве принадлежащих модели матричных схем; разрешающий алгоритм имеет экспоненциальную сложность относительно размеров сравниваемых схем.

Параметрами уравновешенной полугрупповой модели программ с левым сокращением являются ν и L, удовлетворяющие следующим требованиям: ν уравновешенная, полугрупповая, обладающая свойством левого сокращения; L состоит из всех ν-согласованных функций разметки из $\mathcal{L}$. Определим используемые здесь понятия.

Эквивалентность ν в множестве Y^* называется:

— уравновешенной, если, какими бы ни были цепочки h_1, h_2 из Y^*, выполняется соотношение $h_1 \overset{\nu}{\sim} h_2 \Rightarrow |h_1| = |h_2|$, где $|h|$ — длина цепочки h из Y^*;

— полугрупповой, если, какими бы ни были операторные цепочки h_1, h_2, h_3, h_4 из Y^*, выполняется соотношение $(h_1 \overset{\nu}{\sim} h_2) \wedge (h_3 \overset{\nu}{\sim} h_4) \Rightarrow h_1 h_3 \overset{\nu}{\sim} h_2 h_4$;

— обладающей свойством левого сокращения, если, какими бы ни были цепочки h_1, h_2, h_3, h_4, выполняется соотношение $(h_1 h_3 \overset{\nu}{\sim} h_2 h_4) \wedge (h_1 \overset{\nu}{\sim} h_2) \Rightarrow h_3 \overset{\nu}{\sim} h_4$.

Здесь и далее связка $\overset{\nu}{\sim}$ ставится между ν-эквивалентными цепочками.

Функция разметки μ из $\mathcal{L}$ называется ν-согласованной, если для любых h_1, h_2 из Y^* выполняется соотношение $h_1 \overset{\nu}{\sim} h_2 \Rightarrow \mu(h_1) = \mu(h_2)$.

2. НЕОБХОДИМЫЕ УСЛОВИЯ ЭКВИВАЛЕНТНОСТИ МАТРИЧНЫХ СХЕМ ИЗ УРАВНОВЕШЕННОЙ ПОЛУГРУППОВОЙ МОДЕЛИ ПРОГРАММ С ЛЕВЫМ СОКРАЩЕНИЕМ

Модель программ обозначим $M(\nu, L)$. Алгоритм, распознающий эквивалентность матричных схем из $M(\nu, L)$, строит в них сочетаемые маршруты, проверяя при этом необходимые условия эквивалентности схем.

Маршрутом в матричной схеме называется ориентированный путь в схеме, начинающийся в ее входе.

Пусть w_i — маршрут в матричной схеме G_i, $i = 1, 2$. Назовем w_1, w_2 сочетаемыми, если в L существует функция разметки такая, что выполнение на ней схем G_1, G_2 прокладывает в них пути, начинающиеся маршрутами w_1, w_2 соответственно (полагаем далее, что эта функция разметки прокладывает маршруты w_1, w_2).

Сформулируем необходимые условия эквивалентности матричных схем в модели $M(\nu, L)$.

Условие 1. Конечные равновеликие сочетаемые маршруты в эквивалентных матричных схемах из модели $M(\nu, L)$ оканчиваются в вершинах общего типа.

Всякому конечному ориентированному пути в матричной схеме сопоставим несомую им операторную цепочку, полученную выписыванием один за другим операторных символов, которые метят преобразователи, принадлежащие пути, при просмотре самого пути от начала к концу; если путь завершается в преобразователе, то метящий его операторный символ учитывать не будем. Если путь — это w, то несомую им цепочку обозначим $h(w)$.

Обозначим $[w, x]$ продолжение конечного маршрута w, не оканчивающегося в выходе схемы или в пустом цикле, на дугу с меткой x, где $x \in X$.

Условие 2. Пусть w_1, w_2 — конечные равновеликие сочетаемые маршруты в эквивалентных матричных схемах из модели $M(\nu, L)$, имеющие продолжение. Тогда, если $h(w_1) \overset{\nu}{\sim} h(w_2)$, сочетаемыми в схемах являются маршруты $[w_1, x]$, $[w_2, x]$ при любом наборе x из X, в противном случае сочетаемы маршруты $[w_1, x_1]$, $[w_2, x_2]$ для любых наборов x_1, x_2 из X.

Сопряженными в матричных схемах из модели $M(\nu, L)$ называются их вершины v_1, v_2, в которых оканчиваются сочетаемые маршруты, несущие ν-эквивалентные цепочки.

Условие 3. Пусть v_1, v_2 — сопряженные преобразователи в эквивалентных матричных схемах из модели $M(\nu, L)$, помеченные различными операторными символами. Тогда из вершин v_1, v_2 в схемах вырастают кусты общего маркера.

Кустом в матричной схеме, вырастающим из преобразователя v, называется подграф схемы, индуцируемый множеством V преобразователей схемы и содержащий все инцидентные им дуги. При этом как вершины из V, так и инцидентные им дуги сохраняют присвоенные им в схеме метки, а сам подграф удовлетворяет следующим требованиям: вершина v принадлежит множеству V; все вершины из V разбиваются на уровни по признаку их удаленности от v; уровни нумеруются числами $0, 1, \ldots, k$, где $k \geq 1$; уровень с номером 0 состоит из вершины v; какой бы ни была вершина, принадлежащая уровню с номером i, $i = 0, 1, \ldots, k-1$, все дуги из нее ведут в вершины уровня с номером $i+1$; все пути из вершины v к вершинам k-го уровня несут полные операторные цепочки, ν-эквивалентные между собой. Класс ν-эквивалентности этих цепочек называется маркером данного куста.

Полной цепочкой, которую несет путь, оканчивающийся в преобразователе, называется цепочка, полученная приписыванием справа символа, сопоставленного этому преобразователю, к цепочке, которую несет данный путь.

Далее кусты, вырастающие из сопряженных преобразователей v_1, v_2 согласно условию 3, называются сопряженными.

Условимся называть x-преемником куста, вырастающего из вершины v, ту вершину, в которую ведет дуга с меткой x из вершины, принадлежащей последнему уровню куста; здесь $x \in X$.

Условие 4. Пусть F_1, F_2 — сопряженные кусты в эквивалентных матричных схемах из модели $M(\nu, L)$. Тогда, каким бы ни был набор x из X, x-преемниками кустов F_1, F_2 являются сопряженные вершины общего типа.

Условия 1–3 доказаны в [1], условие 4 — очевидное следствие условия 3.

3. АЛГОРИТМ РАСПОЗНАВАНИЯ ЭКВИВАЛЕНТНОСТИ МАТРИЧНЫХ СХЕМ ИЗ УРАВНОВЕШЕННОЙ ПОЛУГРУППОВОЙ МОДЕЛИ ПРОГРАММ С ЛЕВЫМ СОКРАЩЕНИЕМ

Теорема 3. Для уравновешенной полугрупповой модели $M(\nu, L)$ с левым сокращением и разрешимой эквивалентностью ν существует алгоритм, распознающий эквивалентность ее матричных схем путем сравнения сочетаемых маршрутов в схемах.

Доказательство теоремы состоит из описания алгоритма τ.

Алгоритм τ строит списки S_1, S_2, состоящие из пар сопряженных вершин схем G_1, G_2, помещая в S_1 пару из входов схем, а также некоторые пары одинаково помеченных преобразователей, в S_2 — пары разнопомеченных преобразователей. Списки S_1, S_2 не содержат повторяющихся пар; в каждой паре первая компонента — вершина из G_1.

Вначале алгоритм τ помещает в S_1 пару, состоящую из входов схем G_1, G_2, а список S_2 объявляет пустым. Работа алгоритма подразделяется на шаги. На каждом шаге обрабатывается пара, входящая либо в список S_1, либо в список S_2. Шаг может закончиться остановкой алгоритма τ при выявлении неэквивалентности схем G_1, G_2; такой шаг считается неудачно завершенным. В случае положительного завершения шага списки S_1, S_2 пополняются, если это возможно. Если каждый шаг, выполняемый алгоритмом τ, успешно завершен и в списках S_1, S_2 не осталось необработанных пар, то алгоритм τ объявляет схемы G_1, G_2 эквивалентными и прекращает работу. Конечность списков S_1, S_2 гарантирует завершение работы алгоритма.

Опишем шаг работы алгоритма τ.

Случай 1. Обрабатываемая пара (v_1, v_2) входит в список S_1. Тогда алгоритм τ выполняет действие по продолжению сочетаемых маршрутов, завершающихся в вершинах v_1, v_2. Перебираются один за другим наборы из X и определяются x-преемники вершин v_1, v_2. Если они разнотипны, то нарушено условие 1 и алгоритм τ останавливается, выявив неэквивалентность схем G_1, G_2. Если для любого x x-преемники вершин v_1, v_2 однотипны, то те из них, которые являются преобразователями, составляют пары сопряженных вершин, подлежащие проверке: нужно ли включить пару в список S_1 или в список S_2. При пополнении списков включаются только не содержащиеся в них пары.

Случай 2. Обрабатываемая пара (v_1, v_2) входит в список S_2. Тогда в целях продолжения сочетаемых маршрутов, заканчивающихся в вершинах v_1, v_2, алгоритм τ проверяет выполнение условия 3. Строится таблица, элементами которой являются четверки вида

$$(u_1, u_2, h_1, h_2)\,, \tag{1}$$

в которых u_1, u_2 — вершины схем G_1, G_2 соответственно, h_1, h_2 — цепочки из Y^*. Первая строка таблицы содержит единственную четверку (v_1, v_2, y_1, y_2), где y_i — метка вершины v_i, $i = 1, 2$.

Пусть построена j-я строка таблицы, $j \geq 1$. Она объявляется последней в таблице, если в принадлежащих ей четверках все участвующие в них операторные цепочки эквивалентны между собой. Для проверки применяется алгоритм, распознающий эквивалентность отношения ν. Тогда условие 3 на данном шаге выполнено.

Если j-я строка не последняя, предпринимаются действия по построению $(j+1)$-й строки. Просматриваются одна за другой четверки, принадлежащие j-й строке.

Пусть (1) — достигнутая в процессе построения четверка. Если $h_1 \overset{\nu}{\sim} h_2$, то для (1) перебираются все наборы из X; это подсказывает условие 2 для продолжения сочетаемых маршрутов, оканчивающихся в вершинах u_1, u_2. Для очередного набора x находятся x-преемники вершин u_1, u_2; обозначим их u_1', u_2' соответственно. Если они не преобразователи, то нарушено условие 3 и алгоритм τ останавливается, выявив неэквивалентность схем G_1, G_2.

Допустим, что u_1', u_2' — преобразователи, y_1', y_2' — приписанные им операторные символы. Тогда формируется четверка

$$(u_1', u_2', h_1', h_2')\,, \tag{2}$$

где $h_i' = h_i y_i'$, $i = 1, 2$. В случае, когда четверка (2) не совпадает ни с одной уже построенной четверкой $(j+1)$-й строки, условием ее включения в эту строку является требование: u_1' не совпадает с первой компонентой ни в одной из четверок всех предыдущих строк таблицы, а u_2' — со второй компонентой. Очевидно, что нарушение этого требования означает нарушение условия 3, т.е. алгоритм τ останавливается, констатируя неэквивалентность схем G_1, G_2.

Включением четверки (2) в $(j+1)$-ю строку операции алгоритма τ с выбранным набором x для четверки (1) завершены. Если не исчерпаны наборы в X, то для четверки (1) рассматривается другой набор, а если они исчерпаны, то из j-й строки берется следующая четверка, и так до полного исчерпания j-й строки в ситуации, когда алгоритм τ не остановился, обнаружив неэквивалентность схем G_1, G_2.

Если в четверке (1) отношение $h_1 \overset{\nu}{\sim} h_2$ не выполняется, то работа алгоритма τ отличается от описанной выше только тем, что согласно условию 2 перебираются пары из $X \times X$ и для очередной пары x_1, x_2 строятся x_1-преемник вершины u_1 и x_2-преемник вершины u_2.

Предположим, что условие 3, проверяемое для пары (v_1, v_2), выполнено. Тогда алгоритм τ проверяет, выполнено ли условие 4. Просматриваются наборы из X, и для очередного набора x устанавливается, имеют ли общий тип x-преемники всех вершин, принадлежащих четверкам последней строки построенной таблицы. Если не имеют, то алгоритм τ останавливается, констатируя неэквивалентность схем G_1, G_2.

Допустим, что условие 4 для пары (v_1, v_2) выполнено. Тогда алгоритмом построено множество пар сопряженных вершин схем G_1, G_2. Пары сопряженных преобразователей, не содержащиеся в списках S_1, S_2, пополняют эти списки.

Очевидно, что алгоритм τ действительно распознает, являются или нет эквивалентными схемы G_1, G_2.

Теорема 3 доказана.

Рассмотрим теперь оценку временной сложности алгоритма τ. Из описания данного алгоритма видно, что в нем наиболее емкой является работа по проверке условия 3. Создаваемая при этом таблица четверок такова, что в ней каждая следующая строка по своей длине может превышать длину предыдущей строки в $|X|^2$ раз, где $|X|$ — число элементов в множестве X. Поскольку количество строк в таблице может достигать величины n, где n — максимум числа вершин в схемах G_1, G_2, приходим к заключению: в общем случае алгоритм τ экспоненциален по сложности.

В следующем разделе рассматривается подмножество уравновешенных полугрупповых моделей программ с левым сокращением, для которого алгоритм τ при подходящей его модификации будет иметь приемлемую сложность.

4. СЛУЧАЙ УРАВНОВЕШЕННОЙ ПОЛУГРУППОВОЙ МОДЕЛИ ПРОГРАММ С ЛЕВЫМ И ЧАСТИЧНЫМ ПРАВЫМ СОКРАЩЕНИЕМ

Введем еще одно ограничение на параметр ν модели $M(\nu, L)$. Полагаем, что эквивалентность ν обладает свойством частичного правого сокращения, если, какими бы ни были операторные цепочки h_1, h_2 и h из Y^*, выполняется импликация

$$h_1 h \overset{\nu}{\sim} h_2 h \Rightarrow h_1 \overset{\nu}{\sim} h_2. \tag{3}$$

Имеет место следующая лемма.

Лемма 1. Пусть G — матричная схема из модели $M(\nu, L)$, для которой ν обладает свойством частичного правого сокращения, F — куст в схеме G, v — преобразователь, являющийся его внутренней вершиной. Тогда, какими бы ни были пути, ведущие из корня куста F в преобразователь v, они несут ν-эквивалентные операторные цепочки.

Доказательство. Обозначим w_1, w_2 пути, ведущие в преобразователь v из корня куста F, и пусть h_1, h_2 — операторные цепочки, несомые путями w_1, w_2 соответственно. Обозначим w какой-либо путь из v к последнему уровню куста F, и пусть h — несомая им полная операторная цепочка. Согласно определению куста выполняется отношение $h_1 h \overset{\nu}{\sim} h_2 h$, откуда на основании (3) следует утверждение леммы. Лемма 1 доказана.

Докажем следующую теорему.

Теорема 4. Для уравновешенной полугрупповой модели $M(\nu, L)$ с левым и частичным правым сокращением существует алгоритм, распознающий эквивалентность ее матричных схем с полиномиальной сложностью относительно сложности распознавания эквивалентности операторных цепочек.

Доказательство. Опишем алгоритм $\bar{\tau}$, на вход которого поступают матричные схемы G_1, G_2 из модели $M(\nu, L)$.

Алгоритм $\bar{\tau}$ является модификацией алгоритма τ, а именно: при проверке условия 3 дополнительно к действиям алгоритма τ добавляются действия, обусловленные тем, что эквивалентность ν обладает свойством частичного правого сокращения, а значит, применима лемма 1.

Опишем дополнительные действия. Пусть при формировании $(j+1)$-й строки построена четверка (2). Тогда просматриванием четверок, уже содержащихся в $(j+1)$-й строке, устанавливается, имеется ли четверка с первой компонентой u_1'. Если такие четверки есть, то фиксируется ближайшая из них к началу строки. Пусть третья компонента выбранной четверки — цепочка h. Тогда если отношение $h_1' \overset{\nu}{\sim} h$ не выполняется, то алгоритм τ останавливается, констатируя неэквивалентность схем G_1, G_2. В противном случае в четверке (2) цепочка h_1' заменяется цепочкой h. Аналогичные действия выполняются алгоритмом $\bar{\tau}$ для компоненты u_2'. Алгоритм $\bar{\tau}$ описан.

Рассмотрим верхнюю оценку его временной сложности. Пусть $n = \max(n_1, n_2)$, где n_i — число вершин в схеме G_i, $i = 1, 2$. Обозначим τ_0 алгоритм, распознающий ν-эквивалентность операторных цепочек из Y^*. Предполагаем известной функцию $f(k)$, определяющую оценку сложности алгоритма τ_0 при распознавании им ν-эквивалентности цепочек, длина которых равна k, где k — натуральное число. Отметим, что функция $f(k)$ неубывающая.

Выделим в алгоритме $\bar{\tau}$ его подалгоритмы:

— τ', проверяющий выполнимость условия 3;

— τ'', проверяющий выполнимость условия 4.

Отнесем к атомарным следующие операции алгоритма $\bar{\tau}$: установление типа вершины схемы; сравнение типов вершин на равенство; сравнение вершин схем на равенство; отыскание x-преемника вершины схемы; приписывание к цепочке операторного символа.

Поскольку интерес представляет верхняя оценка сложности алгоритма $\bar{\tau}$, а наиболее сложным в $\bar{\tau}$ является подалгоритм τ', предполагаем, что список S_1 пуст, а список S_2 содержит $n_1 n_2$ элементов.

Таким образом, алгоритм τ', а также алгоритм τ'' применяются $n_1 n_2$ раз.

Оценим сверху число действий, выполняемых алгоритмом τ'. В таблице четверок, которую строит алгоритм τ', не более n строк и не более n^2 элементов; первое очевидно, а второе следует из дополнительных действий алгоритма $\bar{\tau}$.

Пусть (1) — рассматриваемая алгоритмом τ' четверка, не принадлежащая ни первой, ни последней строке таблицы. Для нее проверяется, эквивалентны или нет цепочки h_1, h_2, затем отыскиваются x-преемники, составляющие пары в количестве, не большем $|X|^2$. Для каждой пары применяется операция, устанавливающая, состоит ли она из преобразователей. Значит, общее количество действий при положительном исходе не превышает числа N_0, $N_0 = f(n) + 2|X|^2$. Учитываем, что длины цепочек h_1, h_2 не превышают числа n; к $f(n)$ добавляется число атомарных операций.

Пусть построена четверка типа (2). Алгоритм τ' осуществляет первую проверку: принадлежит ли хотя бы одна из вершин u_1', u_2' какой-либо четверке из строк таблицы, предшествующих рассматриваемой строке. При положительном исходе выполняются операции в количестве, не превышающем числа N_1, $N_1 = 2\log(n^2)$, поскольку в предшествующих строках не более чем n^2 четверок.

Далее алгоритм τ' выполняет вторую проверку: совпадает ли u_1' с первой компонентой какой-либо четверки, принадлежащей рассматриваемой строке; если совпадает и в обнаруженной четверке третья компонента — цепочка h, то цепочки h_1' и h сравниваются на ν-эквивалентность. В случае, если они ν-эквивалентны, осуществляется замена цепочки h_1' цепочкой h. При положительном исходе как второй проверки, так и последующей за ней замены цепочки алгоритм τ' выполняет количество операций, не превышающее числа N_2, $N_2 = \log(n^2)(f(n)+n)$. Здесь учитывается, что количество проверяемых четверок не превышает числа n^2, а цепочки h_1', h имеют длину, не превышающую n; замена цепочки h_1' цепочкой h выполняется не более чем за n операций.

Третья проверка, осуществляемая алгоритмом τ', отличается от предыдущей тем, что вместо u_1' рассматривается u_2'. Она потребует количество операций, не превышающее числа N_2.

При положительном исходе второй и третьей проверок в случае, когда обнаруженные для u_1' и u_2' четверки не совпадают, четверка (2), возможно подправленная, включается в рассматриваемую строку.

Таким образом, работа алгоритма τ' с четверкой (2) завершена.

Если рассматриваемая строка полностью построена, то алгоритм τ' выполняет четвертую проверку: является ли эта строка последней в таблице. При этом просматриваются все четверки данной строки (их не более n^2), и для каждой, возможно дважды, выполняется алгоритм τ_0 (если в четверке третья и четвертая компоненты ν-эквивалентны, то одна из них сравнивается на ν-эквивалентность с операторной цепочкой, входящей в первую четверку данной строки). При положительном исходе четвертой проверки алгоритм τ' выполняет количество операций, не превышающее числа N_3, $N_3 = 2n^2 f(n)$.

Таким образом, сложность алгоритма τ' ограничена сверху числом $n^2(N_0 + N_1 + 2N_2 + N_3)$, поскольку при построении таблицы каждая из четверок типа (1) или (2) встречается не более n^2 раз.

Рассмотрим алгоритм τ''. Он анализирует все четверки последней строки таблицы; для первой из них фиксируется тип x-преемников вершин, входящих в четверку, и так для любого x из X (в предположении, что типы совпадают). Далее для каждой последующей четверки и любого x из X сравниваются типы x-преемников вершин, входящих в эту четверку, и проверяется, совпадает ли их общий тип с типом x-преемников вершин из первой четверки. Описанные действия потребуют количество операций, не превышающее числа N_4, $N_4 = 2n^2|X|$.

Кроме того, алгоритм τ'' решает вопрос: включить ли пару x-преемников вершин в случае, когда она составлена преобразователями, в список S_2. На это требуется количество операций, не превосходящее числа N_5, $N_5 = \log(n^2)\, n^2 \,|X|$.

Анализируя полученные верхние оценки сложности алгоритмов τ' и τ'' и учитывая, что алгоритмы выполняются не более n^2 раз, приходим к заключению: утверждение теоремы 4 о сложности алгоритма $\overline{\tau}$ справедливо.

Отметим, что старшее (по зависимости от n) слагаемое в полученной оценке имеет вид $Cn^6 f(n)$, где C — константа, не зависящая от n.

Теорема 4 доказана.

СПИСОК ЛИТЕРАТУРЫ

1. Подловченко Р.И. Техника следов в разрешении проблемы эквивалентности в алгебраических моделях программ // Кибернетика и системный анализ. — 2009. — № 5. — С. 25–37.

Поступила 18.02.2011