МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

УДК 519.681:519.71 DOI 10.17223/20710410/70/5

ЭФФЕКТИВНЫЕ АЛГОРИТМЫ ПРОВЕРКИ ЭКВИВАЛЕНТНОСТИ ПРОПОЗИЦИОНАЛЬНЫХ ПРОГРАММ МИЛИ НА УРАВНОВЕШЕННЫХ ШКАЛАХ[1]

В. В. Подымов

МГУ имени М. В. Ломоносова, г. Москва, Россия

E-mail: valdus@yandex.ru

Предлагается и рассматривается модель программ, называемая далее моделью пропозициональных программ Мили (ППМ) и представляющая собой небольшое синтаксическое обобщение модели дискретных преобразователей Глушкова — Летичевского с «осовремененной» семантикой, основанной на понятиях, использующихся в модели пропозициональных последовательных программ, введённой В. А. Захаровым (ППЗ). Предлагается подход к построению эффективных алгоритмов проверки эквивалентности ППМ, являющийся адаптацией известного подхода к проверке эквивалентности ППЗ, основанного на анализе графа совместных вычислений программ. Демонстрируется применение этого подхода для получения эффективных алгоритмов проверки эквивалентности ППМ для некоторых видов семантик прикладного характера.

Ключевые слова: *проблема эквивалентности, проверка эквивалентности, модели программ, дискретные преобразователи, пропозициональные последовательные программы.*

EFFICIENT EQUIVALENCE-CHECKING ALGORITHMS FOR PROPOSITIONAL MEALY PROGRAMS OVER BALANCED FRAMES

V. V. Podymov

Lomonosov Moscow State University, Moscow, Russia

We propose and investigate propositional Mealy programs (PMPs), a model that is a slight syntactic generalization of the discrete processor model studied by V. M. Glushkov and A. A. Letichevsky. PMPs employ a "modernized" semantics based on notions used in the model of propositional sequential programs proposed by V. A. Zakharov (PSPZs). A technique for constructing efficient equivalence checking algorithms for PMPs is proposed, adapting a known technique for PSPZs based on analysis of a graph of consistent program computations. Efficient PMP equivalence-checking algorithms based on the proposed technique are obtained for some kinds of applied semantics.

[1]Исследования поддержаны Московским центром фундаментальной и прикладной математики МГУ имени М. В. Ломоносова по соглашению № 075-15-2025-345.

Keywords: *equivalence problem, equivalence checking, program models, discrete processors, propositional sequential programs.*

Введение

Данная работа посвящена исследованию проблемы эквивалентности программ: для двух заданных произвольных программ выяснить, имеют ли они одинаковое поведение. Из теоремы Райса — Успенского [1], констатирующей неразрешимость любого нетривиального свойства частично рекурсивных функций, и алгоритмической полноты этого класса функций [2] следует неразрешимость проблемы эквивалентности для любого достаточно выразительного (алгоритмически полного) класса программ. В связи с этим проблема эквивалентности исследуется, в числе прочего, для моделей программ с упрощённой семантикой, позволяющей избежать такой неразрешимости, с тем чтобы использовать решение этой проблемы в модели в качестве достаточного условия эквивалентности программ. Среди таких моделей нас интересуют дискретные преобразователи Глушкова — Летичевского (ДПГЛ) [3] и пропозициональные последовательные программы Захарова [4].

Общие черты моделей ДПГЛ и ППЗ таковы. Программа разделена на синтаксическую и семантическую части. Вычисление программы представляет собой взаимодействие этих двух частей, согласно которому выстраивается последовательность состояний управления и состояний данных программы, начиная с заданных входных значений и пока не будет достигнуто заданное выходное значение состояния управления либо до бесконечности. Результатом конечного вычисления объявляется последнее состояние данных. Построение вычисления основывается на операторах и логических условиях — символах, обозначающих соответственно способы изменения состояния данных программы и способы выбора следующего состояния управления в зависимости от текущих состояний управления и данных. Способ преобразования состояний данных операторами и выбор логических условий при продолжении вычисления задаются семантической частью. Выбор оператора, выполняющегося при продолжении вычисления, задаётся синтаксической частью.

Основные результаты, относящиеся к исследованию проблемы эквивалентности ДПГЛ [3, 5, 6], представляют собой разделение вариантов этой проблемы на разрешимые и неразрешимые. Но ввиду практической значимости алгоритмов проверки эквивалентности (подробнее о ней см., например, в [7]) интерес представляет не только разрешимость как таковая, но и эффективные решающие алгоритмы — полиномиальные достаточно низкой сложности. В [4] введена модель ППЗ, для неё предложена техника проектирования таких алгоритмов — техника совместных вычислений — и в конце введения коротко, без деталей и доказательств, отмечается, что модель ППЗ является обобщением модели ДПГЛ. В последующих работах, посвящённых проблеме эквивалентности ППЗ [7–21] (включая труды, упомянутые в списках литературы этих работ), это соотношение между моделями не обсуждается.

В данной работе обращается особое внимание на то, что пока ещё строго не установлено, имеет ли место включение между структурой или выразительными возможностями моделей ППЗ и ДПГЛ. Синтаксис этих моделей несравним: в модели ДПГЛ оператор обязан выполняться на каждом шаге вычисления, а в модели ППЗ может и не выполняться на последнем шаге; в модели ППЗ выполняющийся оператор однозначно задаётся следующим состоянием управления, а в модели ДПГЛ он может зависеть также и от выбора логического условия, аналогично тому, как выходной символ автомата Мура зависит только от состояния, тогда как в автомате Мили может

зависеть и от выбора входного символа [22]. При этом семантика ПППЗ, хотя и более «современна» в выборе терминологии по сравнению с семантикой ДПГЛ, но в конечном итоге не кажется более широкой, что коротко отмечается в том числе и в [23], хотя это сравнение требует более развёрнутого обсуждения.

В связи с обозначенным соотношением синтаксиса и семантики ДПГЛ и ПППЗ результаты, полученные для этих моделей, вообще говоря, следует считать независимыми, пока не будет строго установлена связь между этими моделями. В данной работе выполнен первый этап установления этой связи: предложена модель пропозициональных программ Мили, совмещающая синтаксис ДПГЛ и семантику ПППЗ, и к ней, в числе прочего, адаптированы результаты работы [4]: техника совместных вычислений и получающиеся с её помощью эффективные алгоритмы проверки эквивалентности для некоторых видов семантик прикладного характера. Применение предложенной техники, то есть эти виды семантик и детали соответствующих алгоритмов, описано в п. 7. Кроме того, получены некоторые побочные результаты, связанные с расширением и структурированием техники совместных вычислений, они обсуждаются в заключении.

Работа имеет следующую структуру. В п. 1 даются используемые понятия и обозначения общего характера. В п. 2 вводятся синтаксис и семантика ППМ, ставится рассматриваемая проблема эквивалентности и обсуждаются специальные понятия и факты, относящиеся к ППМ и требующиеся для формулировки результатов. В п. 3 приводится графовая конструкция, описывающая синхронное выполнение двух ППМ — граф совместных вычислений. В п. 4 вводится понятие критериальной системы, предназначенное для оценки каждой пары состояний данных ППМ характеристикой «удалённости» этих состояний друг от друга. В п. 5 обсуждается графовая конструкция, по сути представляющая собой граф совместных вычислений, снабжённый информацией об «удалённости» состояний данных описываемых пар вычислений, — критериальный граф. В п. 6 приводится алгоритм проверки эквивалентности ППМ, состоящий в обходе критериального графа и параметризованный выбором семантики и критериальной системы, с обоснованием корректности и оценкой сложности. Наконец, в п. 7 показывается, как критериальную систему из [4] можно переформулировать в виде критериальной системы данной работы, и обсуждаются алгоритмы проверки эквивалентности ППМ невысокой сложности, основанные на алгоритме п. 6 и критериальных системах из [4].

Некоторые утверждения в данной работе представляют собой адаптацию утверждений из [4] и смежные не очень сложно обосновываемые свойства рассматриваемых понятий. Такие утверждения, строго говоря, являются новыми и требуют обоснования, но эта новизна в основном техническая, а не содержательная, и эти утверждения озаглавлены словом «Утверждение». Утверждения, полагающиеся существенно новыми и нетривиальными, озаглавлены словами «Лемма» и «Теорема».

1. Общие понятия и обозначения

В связи с обилием областей, из которых далее обширно используются понятия и результаты, приведём названия этих понятий, кроме самых общеизвестных, со ссылками на работы, в которых можно их найти. Здесь же изложим сопутствующие обозначения, как взятые из упомянутых работ, так и отступающие от них.

Используемые понятия теории формальных языков [24]: алфавит; буква (символ) алфавита; слово в заданном алфавите; пустое слово; длина слова. Обозначения: λ —

пустое слово; Σ^* — множество всех слов в алфавите Σ; $|S|$ — размер множества S и длина последовательности S, в том числе длина слова.

Используемые понятия теории графов [25]: ориентированный граф с петлями и кратными дугами (далее — орграф); вершина и дуга орграфа; вершина, из которой исходит дуга и в которую заходит дуга; полный орграф; подграф; орграф, обратный к G; путь; длина пути; простой цикл; кратчайший путь от одной вершины до другой; достижимость одной вершины из другой. Рассмотрим также пути с вершинами заданного множества без упоминания графа, имея в виду пути в полном орграфе с этими вершинами. Про путь будем говорить, что он исходит из первой своей вершины и если он конечен, то он ведёт в последнюю свою вершину. Обозначения: $v_1 \to v_2$ — дуга (v_1, v_2) и часть пути $(v_1, (v_1, v_2), v_2)$; $\rho(i) = v_i$ и $\rho|^n = (v_0 \to v_1 \to \cdots \to v_n)$ для пути $\rho = v_0 \to v_1 \to \ldots$ Будем называть отрезком пути $v_0 \to v_1 \to \ldots$ всякую часть $v_i \to \cdots \to v_j$ этого пути и такой отрезок начальным, если $i = 0$. Как и в [25], рассмотрим также орграфы с помеченными вершинами и дугами, считая, что метки сохраняются при преобразовании графов и рассмотрении и преобразовании путей, и использовать в примерах понятие изоморфизма таких графов. Дугу $v_1 \to v_2$, помеченную значением x, будем записывать как $v_1 \xrightarrow{x} v_2$; кратные дуги $v_1 \to v_2$ с разными метками $x_1, \ldots, x_k$ — как одну помеченную дугу $v_1 \xrightarrow{x_1,\ldots,x_k} v_2$.

Будем использовать также понятия сужения функции $f : X \to Y$ на множество $Z \subseteq X$ [26] и частично определённой функции [2] и следующие обозначения: $f|_Z$ — сужение функции f на Z; $\perp$ — значение неопределённости, не входящее ни в одно из рассматриваемых множеств, кроме случаев, когда это сказано явно; $f : X \to Y \cup \{\perp\}$ — частично определённая функция, в которой $f(x) = \perp$ означает, что значение $f(x)$ не определено; $2^X = \{Y : Y \subseteq X\}$; $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$.

При обсуждении алгоритмов следуем терминологии и рекомендациям [27] и используем известные структуры данных: одномерный и двумерный массивы (далее — соответственно векторы и матрицы); односвязный список; список смежности ориентированного графа. Под сложностью алгоритма понимаем его сложность по времени в худшем случае в модели RAM [27, разд. 2.2]. Это означает, в частности, что при подсчёте сложности полагаем, что в распоряжении алгоритма есть любое необходимое количество пронумерованных ячеек памяти, способных хранить любые целые числа, и единица сложности отвечает выполнению любой простейшей команды: переход к заданной команде, безусловный или условный с проверкой значения в заданной ячейке и сравнениями ($=, \neq, <, \leqslant, >, \geqslant$) значений в ячейках; вычисление в заданной ячейке суммы, разности, произведения, частного или остатка от деления значений заданных ячеек; копирование значения из одной ячейки в другую. Для алгоритма $\mathcal{A}$ записью $\mathcal{A}(x)$ будем обозначать результат выполнения $\mathcal{A}$ на входе x.

Используемые понятия общей алгебры [28, 29]: моноид; образующие (порождающие) элементы; определяющие соотношения; подмоноид; конечно порождённый моноид; свободный моноид; свободный коммутативный моноид; частично коммутативный моноид; прямое произведение моноидов; гомоморфизм моноида A на моноид B (сюръективное отображение элементов A в элементы B, сохраняющее нейтральный элемент и операцию). Для краткости будем называть моноид с множеством образующих X просто X-моноидом. Обозначения: $\mathcal{M} = (M, \varepsilon, \circ)$ — моноид с множеством элементов M, нейтральным элементом ε и операцией $\circ$; $m \in \mathcal{M}$ — синоним записи $m \in M$; $\mathcal{M}(a_1 \ldots a_k) = a_1 \circ \cdots \circ a_k$, где $a_1, \ldots, a_k \in \mathcal{M}$; $m \circ \mathcal{U} = \{m \circ x : x \in \mathcal{U}\}$; $\mathcal{U} \circ m = \{x \circ m : x \in \mathcal{U}\}$; $\mathcal{M}_1 \times \mathcal{M}_2$ — прямое произведение моноидов $\mathcal{M}_1$ и $\mathcal{M}_2$.

2. Модель программ

2.1. Синтаксис

Символами $\mathfrak{A}$ и $\mathfrak{C}$ будем обозначать конечные непустые множества *операторов* и *логических условий* соответственно; эти два множества считаются заданными. Слова в алфавите $\mathfrak{A}$ будем называть (операторными) *цепочками*.

Пропозициональной программой Мили над $\mathfrak{A}$ и $\mathfrak{C}$ (далее — $(\mathfrak{A},\mathfrak{C})$-*программой* и просто *программой*) будем называть систему $\pi = (S, en, EX, T)$, где:

— S — непустое множество *состояний*;
— $en \in S$ — *вход*;
— $EX \subseteq S$ — множество *выходов*;
— $T : (S \setminus EX) \times \mathfrak{C} \to (\mathfrak{A} \times S) \cup \{\bot\}$ — частично определённая *функция переходов*.

Будем также использовать следующие обозначения: если $T(s,c) = (a,r)$, то $T^{\mathfrak{A}}(s,c) = a$ и $T^{S}(s,c) = r$. *Переходом* программы π будем называть четвёрку (s,c,a,r), для которой верно $T(s,c) = (a,r)$. Будем считать, что программа π представляет собой размеченный орграф, в котором S — множество вершин, «вход» и «выход» — метки вершин и каждый переход (s,c,a,r) представляет собой помеченную дугу $s \xrightarrow{c/a} r$.

Пример 1. На рис. 1 показаны следующие пропозициональные программы Мили над множествами $\mathfrak{A} = \{a,b\}$ и $\mathfrak{C} = \{c_1,c_0\}$: $\pi_1 = (\{s_1,s_2,s_3,s_4\}, s_1, \{s_3\}, T_1)$, $\pi_2 = (\{r_1,r_2,r_3,r_4\}, r_1, \{r_3,r_4\}, T_2)$, таблицы значений функций переходов T_1 и T_2 приведены на рис. 2. Здесь и далее вход помечается символом $*$, а выход — двойным контуром. Содержательное понимание элементов синтаксиса пропозициональных программ Мили совпадает с пониманием соответствующих элементов синтаксиса дискретных преобразователей Глушкова — Летичевского и может быть почерпнуто из [3, разд. 2]. В данной работе ограничимся небольшим примером для иллюстрации основных понятий и результатов. На рис. 3 приведён фрагмент кода на языке C++ [30], которому отвечает программа π_1, если буквами a и b обозначены соответственно присваивания «x = x + 1» и «y = y + 1», а буквами c_1 и c_0 — множества состояний данных, в которых соответственно истинно и ложно условие «y < 1».

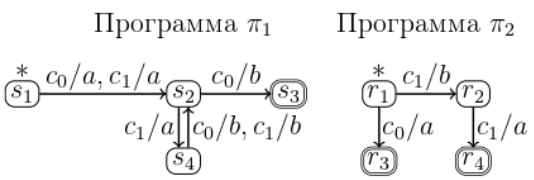

Рис. 1. Пропозициональные программы Мили (пример 1)

$T_1(\mathfrak{s},\mathfrak{c})$		$\mathfrak{c}$	
		c_0	c_1
$\mathfrak{s}$	s_1	a, s_2	a, s_2
	s_2	b, s_3	a, s_4
	s_4	b, s_2	b, s_2

$T_2(\mathfrak{s},\mathfrak{c})$		$\mathfrak{c}$	
		c_0	c_1
$\mathfrak{s}$	r_1	a, r_3	b, r_2
	r_2	$\bot$	a, r_4

Рис. 2. Таблицы значений функций переходов программ (пример 1)

```
x = x + 1;
while (y < 1) {
  x = x + 1;
  y = y + 1;
}
y = y + 1;
```

Рис. 3. Фрагмент кода на языке C++ (пример 1)

Путь в программе и в других графах будем называть *входным*, если он исходит из входа, и *выходным*, если он конечен и ведёт в выход. *Цепочкой пути* $\rho = (s_0 \xrightarrow{c_1/a_1} s_1 \xrightarrow{c_2/a_2} \dots)$ будем называть цепочку $a_1 a_2 \dots$ (для бесконечного пути — бесконечную последовательность операторов).

Программу будем называть *конечной*, если множество её состояний конечно, и *полной*, если её функция переходов всюду определена. *Размером* $|\pi|$ конечной программы $\pi = (S, en, EX, T)$ будем называть число $|S|$.

2.2. Семантика

Детерминированной динамической шкалой над $\mathfrak{A}$ (далее — $\mathfrak{A}$-*шкалой* и просто *шкалой*) называется система $\mathcal{F} = (D, d^0, \circ)$, где:

— D — непустое множество *состояний*;
— $d^0 \in D$ — *вход*;
— $\circ : D \times \mathfrak{A} \to D$ — *операция шкалы*.

Переходом шкалы $\mathcal{F}$ будем называть тройку (d, a, e), удовлетворяющую равенству $d \circ a = e$. Считаем, что шкала $\mathcal{F}$ представляет собой размеченный орграф, в котором D — множество вершин, «вход» — метка вершин и переход (d, a, e) представляет собой помеченную дугу $d \xrightarrow{a} e$. *Цепочкой пути* $\rho = (d_0 \xrightarrow{a_1} d_1 \xrightarrow{a_2} \dots)$ будем называть цепочку $a_1 a_2 \dots$ (для бесконечного пути — бесконечную последовательность операторов). Записью $\mathcal{F}(d, h)$ для $d \in D$ и $h \in \mathfrak{A}^*$ будем обозначать последнее состояние пути в $\mathcal{F}$, исходящего из d и имеющего цепочку h, и записью $\mathcal{F}(h)$ — состояние $\mathcal{F}(d^0, h)$.

Пример 2. На рис. 4 представлен фрагмент шкалы $\mathcal{F}^{x_0,y_0} = (\{x_0, x_0+1, x_0+2, \dots\} \times \{y_0, y_0+1, y_0+2, \dots\}, (x_0, y_0), \circ)$ над $\mathfrak{A} = \{a, b\}$, где $x_0, y_0 \in \mathbb{Z}$, $(x, y) \circ a = (x+1, y)$ и $(x, y) \circ b = (x, y+1)$. Этой шкалой определяется семантика операторов из примера 1 для (неограниченных) целочисленных переменных x, y, имеющих значения x_0 и y_0 соответственно в начале выполнения программы. Заметим, что для всех $x_0, y_0 \in \mathbb{Z}$ шкалы $\mathcal{F}^{x_0,y_0}$ изоморфны как размеченные графы. Поэтому далее в примерах в основном используем шкалу $\mathcal{F}^{0,0}$.

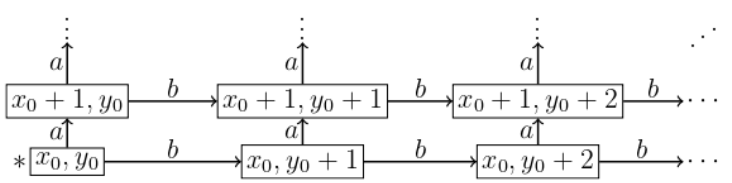

Рис. 4. Фрагмент детерминированной динамической шкалы (пример 2)

Графом $\mathcal{F}$-*вычислений* программы $\pi = (S, en, EX, T)$ назовём программу $\pi \oplus \mathcal{F} = (S \times D, (en, d^0), EX \times D, \mathcal{T})$, в которой функция переходов $\mathcal{T}$ задаётся так: если $T(s, c) = \perp$, то $\mathcal{T}((s, d), c) = \perp$, иначе $\mathcal{T}((s, d), c) = (T^{\mathfrak{A}}(s, c), (T^S(s, c), d \circ T^{\mathfrak{A}}(s, c)))$. Вершины графа $\pi \oplus \mathcal{F}$ и пути в нём будем называть соответственно $\mathcal{F}$-*конфигурациями* и $\mathcal{F}$-*трассами* программы π, элементы s и d конфигурации (s, d) — соответственно *состоянием управления* и *состоянием данных* этой конфигурации. Путь в программе и путь в шкале будем называть *парными*, если их цепочки одинаковы. Для парных путей $\rho_1 = (s_0 \xrightarrow{c_1/a_1} s_1 \xrightarrow{c_2/a_2} \dots)$ и $\rho_2 = (d_0 \xrightarrow{a_1} d_1 \xrightarrow{a_2} \dots)$ одинаковой длины в программе и шкале соответственно записью $\rho_1 \oplus \rho_2$ обозначаем путь $(s_0, d_0) \xrightarrow{c_1/a_1} (s_1, d_1) \xrightarrow{c_2/a_2} \dots$; пути ρ_1, ρ_2 называем соответственно *путём управления* и *путём данных* пути $\rho_1 \oplus \rho_2$.

Пример 3. На рис. 5 приведён фрагмент графа вычислений программы π_1 из примера 1 на шкале $\mathcal{F}^{0,0}$ из примера 2. Входной путь $(s_1,(0,0)) \xrightarrow{c_1/a} (s_2,(1,0)) \xrightarrow{c_0/b} (s_3,(1,1))$ в этом графе можно представить в виде $\rho_1 \oplus \rho_2$, где путь управления $\rho_1 = \big(s_1 \xrightarrow{c_1/a} s_2 \xrightarrow{c_0/b} s_3\big)$ — это входной путь в π_1 и путь данных $\rho_2 = \big((0,0) \xrightarrow{a} (1,0) \xrightarrow{b} (1,1)\big)$ — это парный входной путь в $\mathcal{F}^{0,0}$.

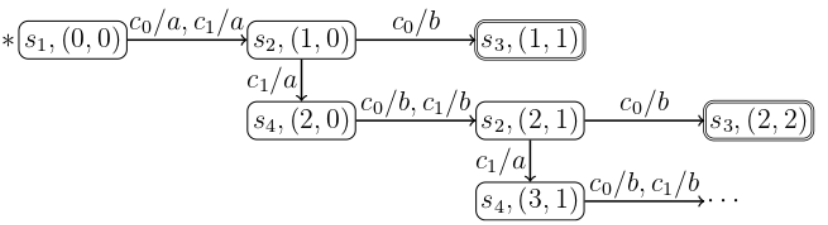

Рис. 5. Фрагмент графа вычислений (пример 3)

Утверждение 1. Для любых программы π и шкалы $\mathcal{F}$ входными путями в $\pi \oplus \mathcal{F}$ являются всевозможные пути $\rho_1 \oplus \rho_2$, где ρ_1 и ρ_2 — парные пути в π и $\mathcal{F}$, и только они.

Доказательство. Пусть $\pi = (S, en, EX, T)$ и $\mathcal{F} = (D, d^0, \circ)$. По устройству путей вида $\rho_1 \oplus \rho_2$ и графа $\pi \oplus \mathcal{F}$, во-первых, все такие пути начинаются со входа этого графа, и, во-вторых, множества переходов $(s,d) \xrightarrow{c/a} (r,e)$, исходящих из заданной конфигурации (s,d) в таких путях и в этом графе, равны: метка c/a произвольно выбирается среди меток переходов, исходящих из s в π; $r = T(s)$; $e = d \circ a$. ■

Детерминированной динамической моделью над $\mathfrak{A}$ и $\mathfrak{C}$ (далее — просто *моделью*) называется система $\mathcal{I} = (\mathcal{F}, L)$, где $\mathcal{F} = (D, d^0, \circ)$ — шкала и $L : D \to \mathfrak{C}$. Такую модель будем также называть *$\mathcal{F}$-моделью* и считать графом, получающимся из $\mathcal{F}$ пометкой каждой вершины d значением $L(d)$. Будем называть *$\mathcal{I}$-трассой*, а также трассой, *реализующейся в $\mathcal{I}$*, $\mathcal{F}$-трассу, для каждого перехода $(s,d) \xrightarrow{c/a} \sigma$ которой верно $c = L(d)$. Будем называть $\mathcal{I}$-трассу τ программы *полной*, если она является самой длинной среди всех $\mathcal{I}$-трасс этой программы, исходящих из $\tau(0)$. Полную входную $\mathcal{I}$-трассу программы будем называть *$\mathcal{I}$-вычислением* этой программы.

Пример 4. На рис. 6 приведён фрагмент модели $\mathcal{I}_1$ над $\mathfrak{A} = \{a, b\}$ и $\mathfrak{C} = \{c_0, c_1\}$, содержащей шкалу $\mathcal{F}^{0,0}$ из примера 2 и разметку её состояний логическими условиями согласно содержательной трактовке из примера 1. Рассмотрим также $\mathcal{F}^{0,0}$-модель $\mathcal{I}_2$, в которой каждое состояние помечено условием c_1, и программы π_1 и π_2 из примера 1. $\mathcal{I}_1$-вычисление τ_1 программы π_1 устроено так: $(s_1,(0,0)) \xrightarrow{c_1/a} (s_2,(1,0)) \xrightarrow{c_1/a} (s_4,(2,0)) \xrightarrow{c_1/b} (s_2,(2,1)) \xrightarrow{c_0/b} (s_3,(2,2))$. $\mathcal{I}_2$-вычисление τ_2 программы π_1 бесконечно и начинается так: $(s_1,(0,0)) \xrightarrow{c_1/a} (s_2,(1,0)) \xrightarrow{c_1/a} (s_4,(2,0)) \xrightarrow{c_1/b} (s_2,(2,1)) \xrightarrow{c_1/a} (s_4,(3,1)) \xrightarrow{c_1/b} \ldots$ $\mathcal{I}_1$-вычисление τ_3 программы π_2 устроено так: $(r_1,(0,0)) \xrightarrow{c_1/b} (r_2,(0,1))$.

Утверждение 2. Для любой программы π и любой модели $\mathcal{I}$ существует ровно одно $\mathcal{I}$-вычисление программы π.

Доказательство. Пусть $\mathcal{I} = (\mathcal{F}, L)$.

С у щ е с т в о в а н и е . Входной путь длины 0 в графе $\pi \oplus \mathcal{F}$ существует и является входной $\mathcal{I}$-трассой. Значит, существует и входная $\mathcal{I}$-трасса наибольшей длины.

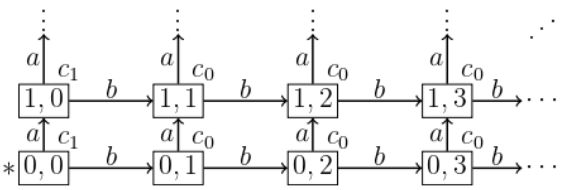

Рис. 6. Фрагмент детерминированной динамической модели (пример 4)

Е д и н с т в е н н о с т ь . Достаточно показать единственность входной $\mathcal{I}$-трассы τ программы π заданной длины. Для этого заметим, что 1) первая конфигурация каждой $\mathcal{F}$-трассы задана однозначно и 2) для каждого перехода $(s,d) \xrightarrow{c/a} (r,e)$ значениями s и d однозначно задаются остальные значения: $c = L(d)$, в π содержится не более одного перехода, исходящего из s и помеченного условием c, и этим переходом однозначно задаются a, r и $e = \mathcal{F}(d,a)$. ■

Далее будем без отсылки к утверждению 2 использовать понятие $\mathcal{I}$-вычисления программы для модели $\mathcal{I}$, имея в виду, что такое вычисление существует и единственно. Будем называть *итогом* конечной трассы состояние данных её последней конфигурации, *результатом* выходной трассы — её итог, а результатами трасс, не являющихся выходными, — значение $\perp$. Результат $\mathcal{I}$-вычисления программы π обозначим $\mathcal{I}(\pi)$. Для шкалы $\mathcal{F}$ будем называть *$\mathcal{F}$-вычислением* $\mathcal{I}$-вычисление для любой $\mathcal{F}$-модели $\mathcal{I}$.

Пример 5. Результаты вычислений τ_1, τ_2 и τ_3, приведённых в примере 4, равны соответственно $(2,2)$, $\perp$ и $\perp$.

2.3. Э к в и в а л е н т н о с т ь

Программы π_1, π_2 будем называть *эквивалентными в модели* $\mathcal{I}$, а также *$\mathcal{I}$-эквивалентными*, если $\mathcal{I}(\pi_1) = \mathcal{I}(\pi_2)$, и *эквивалентными на шкале* $\mathcal{F}$, а также *$\mathcal{F}$-эквивалентными*, если они эквивалентны в каждой $\mathcal{F}$-модели. Будем обозначать $\mathcal{F}$-эквивалентность программ π_1 и π_2 как $\pi_1 \sim_{\mathcal{F}} \pi_2$. *Проблема эквивалентности программ на шкале* $\mathcal{F}$ состоит в том, чтобы для заданных произвольных конечных программ π_1, π_2 проверить соотношение $\pi_1 \sim_{\mathcal{F}} \pi_2$. Будем называть программы *сильно эквивалентными*, если они эквивалентны в любой модели (а значит, и на любой шкале).

Пример 6. Эквивалентность программ на шкале означает, что результаты вычислений этих программ обязательно равны, если смысл логических условий неизвестен, а относительно операторов известны только свойства, задаваемые этой шкалой. В частности, шкала $\mathcal{F}^{0,0}$ из примера 2 отвечает свойству перестановочности (коммутативности) операторов a и b: итог трассы не зависит от порядка выполнения операторов. Этим свойством обладают присваивания, которым сопоставлены буквы a и b в примере 1. Для программ π_1 и π_2 из примера 1 верно $\mathcal{I}_1(\pi_1) \neq \mathcal{I}_1(\pi_2)$ (см. примеры 4 и 5), а значит, $\pi_1 \nsim_{\mathcal{F}^{0,0}} \pi_2$.

2.4. В с п о м о г а т е л ь н ы е п о н я т и я и с в о й с т в а

Будем считать заданной шкалу $\mathcal{F} = (D, d^0, \circ)$ и все утверждения сформулируем для произвольной такой шкалы.

Шкалу $\mathcal{F}$ будем называть *уравновешенной*, если для любых цепочек h, g из равенства $\mathcal{F}(h) = \mathcal{F}(g)$ следует $|h| = |g|$. Записями $\mathcal{R}_{\mathcal{F}}$ и $\mathcal{B}_{\mathcal{F}}$ обозначаем соответственно множества $\{\mathcal{F}(h) : h \in \mathfrak{A}^*\}$ и $\{(\mathcal{F}(h), \mathcal{F}(g)) : h, g \in \mathfrak{A}^*, |h| = |g|\}$.

Пример 7. Шкала $\mathcal{F}^{0,0}$ из примера 2 уравновешенна, так как для любых $k, m \in \mathbb{N}_0$ все слова h, для которых $\mathcal{F}^{0,0}(h) = (k,m)$, имеют одинаковую длину $(k+m)$.

Шкала $\mathcal{F} = (\{d\}, d, \circ)$ над $\mathfrak{A} = \{a, b\}$, в которой $d \circ a = d \circ b = d$, не уравновешенна: $\mathcal{F}(a) = \mathcal{F}(aa)$, но $|a| \neq |aa|$.

Пополнением программы $\pi = (S, en, EX, T)$ для значений $\mathrm{loop} \notin S$ и $a \in \mathfrak{A}$ будем называть программу $\pi^{\mathrm{loop},a} = (S \cup \{\mathrm{loop}\}, en, EX, T^{a,\mathrm{loop}})$, где $T^{\mathrm{loop},a}$ отличается от T только тем, что если $T(s,c) = \perp$ или $s = \mathrm{loop}$, то $T^{\mathrm{loop},a}(s,c) = (a, \mathrm{loop})$. Состояние s программы назовём *завершаемым*, если из него в этой программе достижим выход, иначе — *незавершаемым*. Весом $\|s\|_\pi$ состояния s программы π считаем длину кратчайшего пути из s в какой-либо выход в π (если такого пути нет, то вес бесконечен).

Пример 8. Пополнение $\pi_2^{\mathrm{loop},a}$ программы π_2 из примера 1 приведено на рис. 7.

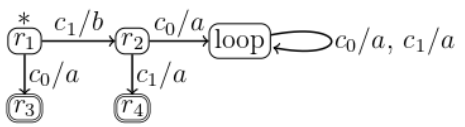

Рис. 7. Пополнение программы (пример 8)

Пример 9. Для программ π_1 из примера 1 и $\pi_2' = \pi_2^{\mathrm{loop},a}$ из примера 8 верно следующее: $\|s_3\|_{\pi_1} = \|r_3\|_{\pi_2'} = \|r_4\|_{\pi_2'} = 0$; $\|s_2\|_{\pi_1} = \|r_1\|_{\pi_2'} = \|r_2\|_{\pi_2'} = 1$; $\|s_1\|_{\pi_1} = \|s_4\|_{\pi_1} = 2$; $\|\mathrm{loop}\|_{\pi_2'} = \infty$. Состояние loop незавершаемо, остальные состояния завершаемы.

Назовём $\mathcal{F}$-трассы программ *совместными* (в том числе одну трассу — совместной), если существует $\mathcal{F}$-модель, в которой реализуются все эти трассы. *Произведением $\mathcal{F}$-трасс* $\tau_1 = ((s_0, d_0) \xrightarrow{c_1/a_1} (s_1, d_1) \xrightarrow{c_2/a_2} \dots)$ и $\tau_2 = ((r_0, e_0) \xrightarrow{\ell_1/b_1} (r_1, e_1) \xrightarrow{\ell_2/b_2} \dots)$ одинаковой длины будем называть путь $\tau_1 \otimes \tau_2 = (((s_0, r_0), (d_0, e_0)) \xrightarrow{(c_2,\ell_1)/(a_1,b_1)} ((s_1, r_1), (d_1, e_1)) \xrightarrow{(c_2,\ell_2)/(a_2,b_2)} \dots)$, такие трассы τ_1 и τ_2 будем называть соответственно *первой и второй проекциями* этого пути.

Пример 10. Рассмотрим модели $\mathcal{I}_1$, $\mathcal{I}_2$ и $\mathcal{F}^{0,0}$-трассы τ_1, τ_2, τ_3 из примера 4. Трассы τ_1 и τ_3 совместны, так как обе они реализуются в $\mathcal{I}_1$. Трассы τ_2 и τ_3 совместны, так как обе они реализуются в $\mathcal{I}_2$. Трассы τ_1 и τ_2 несовместны, так как если они обе реализуются в некоторой модели $\mathcal{I}$, то, согласно устройству τ_1, состояние $(2,1)$ в $\mathcal{I}$ должно быть помечено условием c_0, а согласно устройству τ_2, — условием c_1. Путь $\tau_1|^1 \otimes \tau_3$ имеет вид $((s_1, r_1), ((0,0), (0,0))) \xrightarrow{(c_1,c_1)/(a,b)} ((s_2, r_2), ((1,0), (0,1)))$.

Утверждение 3. Любые две программы $\mathcal{F}$-эквивалентны тогда и только тогда, когда результаты любых совместных $\mathcal{F}$-вычислений этих двух программ равны.

Доказательство. Напрямую следует из соответствующих определений. ■

Утверждение 4. Любая программа сильно эквивалентна любому своему пополнению.

Доказательство. Для любой модели $\mathcal{I}$ $\mathcal{I}$-вычисления программы и её пополнения отличаются только тем, что если вычисление программы конечно и ведёт не в выход, то вычисление её пополнения бесконечно. При этом результаты всех конечных $\mathcal{I}$-вычислений, оканчивающихся не в выходе, и бесконечных вычислений равны $\perp$. Значит, для любой модели $\mathcal{I}$ результаты $\mathcal{I}$-вычислений программы и её пополнения равны. ■

Утверждение 5. Любая конечная $\mathcal{F}$-трасса τ любой программы имеет итог $\mathcal{F}(d, h)$, где d — состояние данных конфигурации $\tau(0)$ и h — цепочка трассы τ.

Доказательство. Достаточно применить индукцию по длине трассы, заметив, что для любого перехода $(s, \mathcal{F}(d,h)) \xrightarrow{c/a} (r,e)$ трассы τ верно $e = \mathcal{F}(d,ha)$ (по определению трассы). ■

Утверждение 6. Итогом любой конечной входной $\mathcal{F}$-трассы τ любой программы является значение $\mathcal{F}(h)$, где h — цепочка трассы τ.

Доказательство. Следует из утверждения 5 и того, что состояние данных конфигурации $\tau(0)$ есть $\mathcal{F}(\lambda)$ (по соответствующим определениям). ■

Утверждение 7. Для любой вершины (ss, dd) произведения $\tau_1 \otimes \tau_2$ любых входных $\mathcal{F}$-трасс τ_1, τ_2 одинаковой длины любых программ верно $dd \in \mathcal{B}_{\mathcal{F}}$.

Доказательство. Достаточно показать, что утверждение верно для последней вершины (ss, dd) каждого конечного начального отрезка ρ пути $\tau_1 \otimes \tau_2$. Пусть путь ρ имеет длину n. По определению произведения трасс $\rho = \tau_1|^n \otimes \tau_2|^n$ и $dd = (d_1, d_2)$ для состояний данных d_1, d_2 конфигураций $\tau_1(n)$, $\tau_2(n)$. По утверждению 6 для некоторых цепочек h и g одинаковой длины n верно $d_1 = \mathcal{F}(h)$ и $d_2 = \mathcal{F}(g)$. Значит, $(d_1, d_2) \in \mathcal{B}_{\mathcal{F}}$ по определению этого множества. ■

Утверждение 8. Любой набор $\mathcal{F}$-трасс любых программ совместен в том и только в том случае, если для любых двух переходов $(s_1, d_1) \xrightarrow{c_1/a_1} \sigma_1$ и $(s_2, d_2) \xrightarrow{c_2/a_2} \sigma_2$ любых трасс этого набора из равенства $d_1 = d_2$ следует равенство $c_1 = c_2$.

Доказательство.

Необходимость. Если набор $\mathcal{F}$-трасс совместен, то существует модель $\mathcal{I} = (\mathcal{F}, L)$, в которой реализуются все трассы этого набора, и для любого состояния данных d и любых переходов $(s_1, d) \xrightarrow{c_1/a_1} \sigma_1$ и $(s_2, d) \xrightarrow{c_2/a_2} \sigma_2$ трасс этого набора $c_1 = L(d) = c_2$.

Достаточность. Пусть свойство переходов, сформулированное в условии, верно. Рассмотрим любую модель $\mathcal{I} = (\mathcal{F}, L)$, обладающую таким свойством: если в какой-либо из трасс содержится переход $(s, d) \xrightarrow{c/a} \sigma$, то $L(d) = c$. Согласно предполагаемому свойству переходов, для любого перехода $(r, d) \xrightarrow{\ell/b} \delta$ рассматриваемого набора трасс верно $\ell = c = L(d)$. Значит, интерпретация $\mathcal{I}$ задана корректно (существует). По определению все трассы рассматриваемого набора реализуются в $\mathcal{I}$. ■

Утверждение 9. Любые отрезки любых совместных $\mathcal{F}$-трасс совместны.

Доказательство. Следует из утверждения 8 и того, что все переходы отрезка трассы являются переходами этой трассы. ■

Утверждение 10. Любые $\mathcal{F}$-трассы любых программ совместны тогда и только тогда, когда совместна каждая пара конечных начальных отрезков этих трасс.

Доказательство.

Необходимость следует из утверждения 9.

Достаточность. Рассмотрим произвольный несовместный набор трасс. По утверждению 8 в этих трассах существуют переходы $t_1 = ((s_1, d_1) \xrightarrow{c_1/a_1} \sigma_1)$ и $t_2 = ((s_2, d_2) \xrightarrow{c_2/a_2} \sigma_2)$, для которых $d_1 = d_2$ и $c_1 \neq c_2$. Тогда существуют и конечные начальные отрезки τ_1', τ_2' некоторых трасс этого набора, содержащие переходы t_1 и t_2 соответственно. Трассы τ_1', τ_2' несовместны по утверждению 8. ■

3. Граф совместных вычислений

В данном пункте считаем заданными полные программы $\pi_1 = (S_1, en_1, EX_1, T_1)$ и $\pi_2 = (S_2, en_2, EX_2, T_2)$ и уравновешенную шкалу $\mathcal{F} = (D, d^0, \circ)$. Все утверждения формулируются для произвольных таких π_1, π_2 и $\mathcal{F}$.

Произведением программ π_1 и π_2 будем называть $(\mathfrak{A} \times \mathfrak{A}, \mathfrak{C} \times \mathfrak{C})$-программу $\pi_1 \otimes \pi_2 = (S_1 \times S_2, (en_1, en_2), (EX_1 \times S_2) \cup (S_1 \times EX_2), T)$, функция переходов которой задаётся так: если $T_1(s, c) \neq \perp$ и $T_2(r, \ell) \neq \perp$, то $T((s, r), (c, \ell)) = ((T_1^{\mathfrak{A}}(s, c), T_2^{\mathfrak{A}}(r, \ell)), (T_1^{S_1}(s, c), T_2^{S_2}(r, \ell)))$, иначе $T((s, r), (c, \ell)) = \perp$. *Произведением шкал* $\mathcal{F}_1 = (D_1, d_1^0, \circ_1)$ и $\mathcal{F}_2 = (D_2, d_2^0, \circ_2)$ называем $(\mathfrak{A} \times \mathfrak{A})$-шкалу $\mathcal{F}_1 \otimes \mathcal{F}_2 = (D_1 \times D_2, (d_1^0, d_2^0), \circ)$, в которой операция $\circ$ задаётся равенством $(d, e) \circ (a, b) = (d \circ_1 a, e \circ_2 b)$. Для $(\mathfrak{A} \times \mathfrak{A})$-шкалы $\mathcal{F}$ и цепочек $h = a_1 \dots a_k$ и $g = b_1 \dots b_k$, где $k \in \mathbb{N}_0$ и $a_1, \dots a_k, b_1, \dots, b_k \in \mathfrak{A}$, используем сокращения $\mathcal{F}(d, h_1, h_2) = \mathcal{F}(d, (a_1, b_1) \dots (a_k, b_k))$ и $\mathcal{F}(h_1, h_2) = \mathcal{F}(d^0, h_1, h_2)$. *Графом совместных вычислений* программ π_1 и π_2 на шкале $\mathcal{F}$ назовём подграф $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ графа $(\pi_1 \otimes \pi_2) \oplus (\mathcal{F} \otimes \mathcal{F})$, содержащий все вершины и дуги, кроме дуг вида $((s, r), (d, e)) \xrightarrow{(c,\ell)/(a,b)} v$, в которых $d = e$ и $c \neq \ell$.

Пример 11. Рассмотрим программы π_1 и $\pi_2' = \pi_2^{\mathrm{loop},a}$ и шкалу $\mathcal{F}^{0,0}$ из примеров 1, 2 и 8. Положим $\mathfrak{c}_{ij} = (c_i, c_j)$, $\mathfrak{a}_{xy} = (x, y)$ и $\mathfrak{d}_{ij\,km} = ((i, j), (k, m))$. На рис. 8–11 приведены соответственно программа $\pi_1 \otimes \pi_2'$, фрагмент шкалы $\mathcal{F}^{0,0} \otimes \mathcal{F}^{0,0}$, фрагмент графа $(\pi_1 \otimes \pi_2') \oplus (\mathcal{F}^{0,0} \otimes \mathcal{F}^{0,0})$ и фрагмент графа $\Gamma^{\mathcal{F}^{0,0}}_{\pi_1,\pi_2'}$.

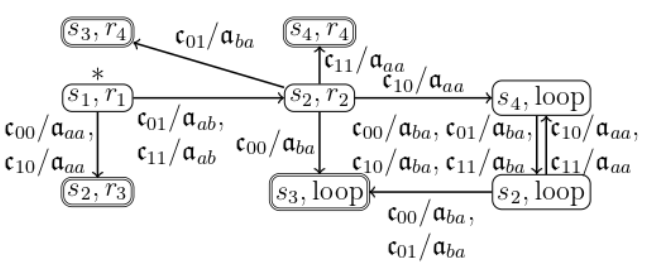

Рис. 8. Произведение программ (пример 11)

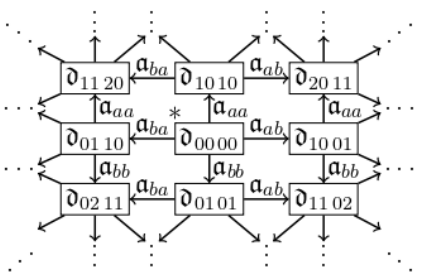

Рис. 9. Фрагмент произведения шкал (пример 11)

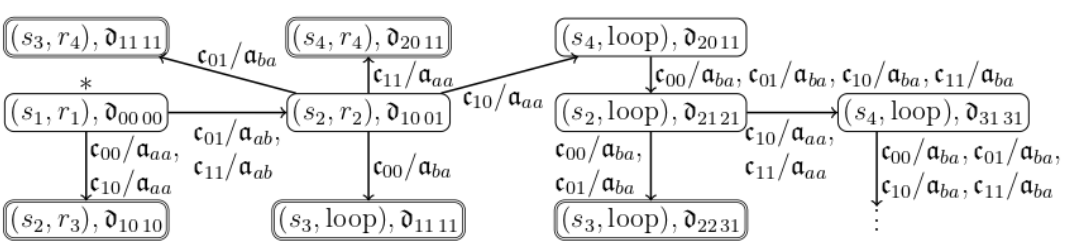

Рис. 10. Фрагмент графа вычислений произведения программ (пример 11)

Утверждение 11. Для любых совместных входных $\mathcal{F}$-трасс τ_1, τ_2 одинаковой длины программ π_1, π_2 соответственно путь $\tau_1 \otimes \tau_2$ — это входной путь в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$.

Доказательство. Докажем утверждение для конечных трасс индукцией по длине, после чего — для бесконечных трасс.

Б а з а: τ_1 и τ_2 имеют длину 0. Тогда τ_i — это трасса из одной конфигурации (en_i, d^0), $i \in \{1, 2\}$, и $\tau_1 \otimes \tau_2$ — путь из одной вершины $((en_1, en_2), (d^0, d^0))$, являющейся входом графа $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$, то есть входной путь в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$.

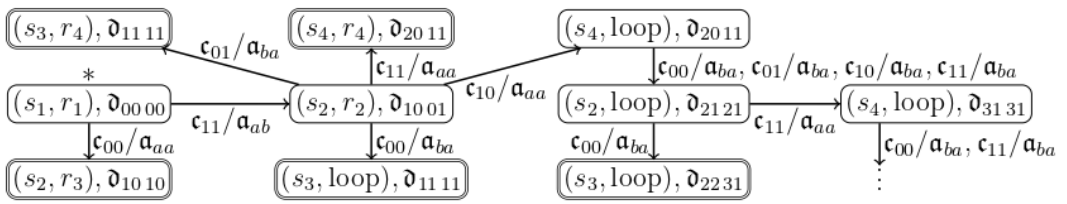

Рис. 11. Фрагмент графа совместных вычислений программ (пример 11)

И н д у к т и в н ы й п е р е х о д: τ_1 и τ_2 имеют длину $n+1$, $n \in \mathbb{N}_0$, и $\rho' = \tau_1|^n \otimes \tau_2|^n$ — это входной путь в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$. Положим для ясности, что $(s_i, d_i) \xrightarrow{c_i/a_i} (r_i, e_i)$ — последний переход трассы τ_i, $i \in \{1,2\}$. По устройству произведения трасс последней вершиной пути ρ' является $((s_1,s_2),(d_1,d_2))$. По устройству графа $(\pi_1 \otimes \pi_2) \oplus (\mathcal{F} \otimes \mathcal{F})$ в нём содержится дуга $t = (((s_1,s_2),(d_1,d_2)) \xrightarrow{(c_1,c_2)/(a_1,a_2)} ((r_1,r_2),(e_1,e_2)))$. Если $d_1 = d_2$, то по утверждению 8 $c_1 = c_2$. Значит, в любом случае дуга t содержится в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$, тогда $\tau_1 \otimes \tau_2 = (\rho' \xrightarrow{(c_1,c_2)/(a_1,a_2)} ((r_1,r_2),(e_1,e_2)))$ — это входной путь в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$.

П е р е х о д к б е с к о н е ч н ы м т р а с с а м: пусть τ_1 и τ_2 имеют бесконечную длину и утверждение справедливо для любых конечных трасс указанного в условии вида. Тогда для каждого $n \in \mathbb{N}_0$ в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ содержится входной путь $\rho_n = \tau_1|^n \otimes \tau_2|^n$ и ρ_{n+1} получается из ρ_n добавлением в конец одного перехода этого графа, обозначим этот переход t_n. Значит, в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ существует бесконечный путь ρ из входа по переходам $t_0, t_1, t_2, \ldots$ и по устройству произведения трасс $\rho = \tau_1 \otimes \tau_2$. ■

Утверждение 12. Любой входной путь ρ в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ представим в виде $\rho = \tau_1 \otimes \tau_2$, где τ_1 и τ_2 — совместные входные $\mathcal{F}$-трассы программ π_1, π_2 соответственно.

Доказательство. Докажем утверждение для конечных путей индукцией по длине, после чего — для бесконечных путей.

Б а з а: ρ имеет длину 0. Тогда $\rho = \tau_1 \otimes \tau_2$, где τ_i состоит из одной вершины (en_i, d^0), $i \in \{1,2\}$, и по утверждению 8 τ_1 и τ_2 — совместные входные $\mathcal{F}$-трассы программ π_1, π_2 соответственно.

И н д у к т и в н ы й п е р е х о д: ρ имеет длину $n+1$, $n \in \mathbb{N}_0$, и существуют совместные входные $\mathcal{F}$-трассы τ_1', τ_2' программ π_1, π_2, удовлетворяющие равенству $\rho|^n = \tau_1' \otimes \tau_2'$. Положим, что $((s_1,s_2),(d_1,d_2)) \xrightarrow{(c_1,c_2)/(a_1,a_2)} ((r_1,r_2),(e_1,e_2))$ — последняя дуга пути ρ. Для каждого $i \in \{1,2\}$ верно следующее. По определению произведения трасс трасса τ_i' ведёт в конфигурацию (s_i, d_i). По устройству графа вычислений, программы $\pi_1 \otimes \pi_2$ и шкалы $\mathcal{F} \otimes \mathcal{F}$ переход $(s_i,d_i) \xrightarrow{c_i/a_i} (r_i,e_i)$ входит в граф $\pi_i \oplus \mathcal{F}$, а значит, $\tau_i = (\tau_i' \xrightarrow{c_i/a_i} (r_i,e_i))$ — входная $\mathcal{F}$-трасса программы π_i. По утверждению 6 и уравновешенности шкалы $\mathcal{F}$ состояние данных d_i отличается от состояний данных остальных конфигураций трасс τ_1' и τ_2', кроме, быть может, состояния d_{3-i}. По определению графа совместных вычислений если $d_1 = d_2$, то $c_1 = c_2$. По последнему соотношению, утверждению 8 и совместности трасс τ_1', τ_2' трассы $\tau_1 = (\tau_1' \xrightarrow{c_1/a_1} (r_1,e_1))$ и $\tau_2 = (\tau_2' \xrightarrow{c_2/e_2} (r_2,e_2))$ совместны. По определению произведения трасс $\rho = \tau_1 \otimes \tau_2$.

П е р е х о д к б е с к о н е ч н ы м п у т я м: ρ имеет бесконечную длину; полагаем, что утверждение справедливо для любых конечных путей указанного в условии вида. Тогда для каждого $n \in \mathbb{N}_0$ путь $\rho_n = \rho|^n$ представим в виде $\rho_n = \tau_1^n \otimes \tau_2^n$, где τ_1^n и τ_2^n — совместные входные $\mathcal{F}$-трассы программ π_1, π_2, и для каждого $i \in \{1,2\}$

трасса τ_i^{n+1} получается из τ_i^n добавлением в конец одного перехода программы π_i (обозначим этот переход t_i^n), значит, бесконечный путь τ_i из конфигурации (en_i, d^0) по переходам $t_i^0, t_i^1, t_i^2, \ldots$ является бесконечной входной $\mathcal{F}$-трассой программы π_i. При этом трассы τ_1 и τ_2 совместны — иначе по утверждению 10 существуют несовместные конечные начальные отрезки этих трасс и по утверждению 9 для некоторого n несовместны трассы τ_1^n и τ_2^n, чего быть не может по полученному выше. ■

Вершину $((s_1, s_2), (d_1, d_2))$ графа $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ назовём *достижимой* в этом графе, если она достижима из входа, и *опровергающей*, если верно одно из следующих условий:

1) $s_1 \in EX_1$ и $s_2 \notin EX_2$;
2) $s_1 \notin EX_1$ и $s_2 \in EX_2$;
3) $s_1 \in EX_1$, $s_2 \in EX_2$ и $d_1 \neq d_2$.

Пример 12. Среди вершин графа совместных вычислений на рис. 11 опровергающими являются все выходы, кроме $((s_3, r_4), \mathfrak{d}_{11\,11})$, и только они. В этом графе содержится бесконечно много неизображённых опровергающих вершин — например, недостижимая вершина $((s_3, r_4), \mathfrak{d}_{20\,02})$.

Утверждение 13. Для любой достижимой вершины (ss, dd) графа $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ верно $dd \in \mathcal{B}_{\mathcal{F}}$.

Доказательство. Следует из утверждений 7 и 12. ■

Утверждение 14. Соотношение $\pi_1 \nsim_{\mathcal{F}} \pi_2$ верно тогда и только тогда, когда в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ существует достижимая опровергающая вершина.

Доказательство.

Необходимость. Пусть $\pi_1 \nsim_{\mathcal{F}} \pi_2$. По утверждению 3 существуют совместные $\mathcal{F}$-вычисления τ_1, τ_2 программ π_1, π_2 соответственно, имеющие различные результаты. Это означает, в частности, что одно из этих вычислений конечно. Положим без ограничения общности, что длина n трассы τ_1 конечна и не превосходит длину τ_2 (иначе достаточно поменять местами индексы 1 и 2). По утверждению 9 трассы τ_1 и $\tau_2' = \tau_2|^n$ совместны. По утверждению 11 $\rho = \tau_1 \otimes \tau_2'$ — входной путь в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$, по устройству произведения трасс этот путь конечен. Положим, что ρ ведёт в вершину $v = ((s_1, s_2), (d_1, d_2))$. Значит, вершина v достижима в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$. По выбору τ_1 верно $s_1 \in EX_1$. По устройству произведения трасс и неравенству результатов вычислений τ_1 и τ_2, если $s_2 \in EX_2$, то $d_1 \neq d_2$. Значит, в любом случае вершина v является опровергающей.

Достаточность. Положим, что в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ по некоторому пути ρ достижима некоторая опровергающая вершина $v = ((s_1, s_2), (d_1, d_2))$. По утверждению 12 и устройству произведения трасс существуют конечные совместные входные трассы τ_1, τ_2 одинаковой длины (обозначим её n) программ π_1, π_2, ведущие в конфигурации (s_1, d_1) и (s_2, d_2) соответственно.

Случай 1: $s_1 \in EX_1$ и $s_2 \in EX_2$. Тогда d_1 и d_2 — результаты совместных $\mathcal{F}$-вычислений τ_1, τ_2 соответственно. Так как вершина v является опровергающей, верно $d_1 \neq d_2$. Значит, по утверждению 3 $\pi_1 \nsim_{\mathcal{F}} \pi_2$.

Случай 2: $s_1 \in EX_1$ и $s_2 \notin EX_2$. По совместности трасс τ_1, τ_2 существует модель $\mathcal{I}$, в которой реализуются эти трассы. Положим, что τ_2' — $\mathcal{I}$-вычисление программы π_2. Так как $s_1 \in EX_1$, верно $\mathcal{I}(\pi_1) = d_1$. Так как $s_2 \notin EX_w$, длина τ_2' больше длины τ_2. Если длина τ_2' бесконечна, то $\mathcal{I}(\pi_2) = \bot \neq d_1 = \mathcal{I}(\pi_1)$. Иначе длина τ_2' конечна, по утверждению 6 $\mathcal{I}(\pi_1) = \mathcal{F}(h_1)$ и $\mathcal{I}(\pi_2) = \mathcal{F}(h_2)$, где h_1 и h_2 — цепочки трасс τ_1 и τ_2', и из этого, неравенства $|h_1| < |h_2|$ (так как τ_2' длиннее τ_2 и длина τ_2 равна длине τ_1) и уравновешенности шкалы $\mathcal{F}$ следует $\mathcal{I}(\pi_1) \neq \mathcal{I}(\pi_2)$, а значит, $\pi_1 \nsim_{\mathcal{F}} \pi_2$.

С л у ч а й 3: $s_1 \notin EX_1$ и $s_2 \in EX_2$ — повторяет случай 2 с взаимной заменой индексов 1 и 2. ■

4. Критериальная система

Для шкалы $\mathcal{F} = (D, d^0, \circ)$ и $(\mathfrak{A} \times \mathfrak{A})$-шкалы $\mathcal{W} = (W, w^0, \odot)$ *критериальным морфизмом* $\varphi : \mathcal{F} \to \mathcal{W}$ будем называть функцию $\varphi : \mathcal{B}_\mathcal{F} \to W$, заданную так:

— $\varphi(d^0, d^0) = w^0$;
— если $(d, e) \in \mathcal{B}_\mathcal{F}$ и $a, b \in \mathfrak{A}$, то $\varphi(d, e) \odot (a, b) = \varphi(d \circ a, e \circ b)$.

Записью $\mathcal{E}_\varphi$ будем обозначать множество $\{\varphi(d, d) : d \in \mathcal{R}_\mathcal{F}\}$. *Критериальной системой* шкалы $\mathcal{F} = (D, d^0, \circ)$ назовём пару $(\mathcal{W}, \varphi)$, где $\mathcal{W}$ — $(\mathfrak{A} \times \mathfrak{A})$-шкала, $\varphi : \mathcal{F} \to \mathcal{W}$ и для любой пары $(d, e) \in \mathcal{B}_\mathcal{F}$ из соотношения $\varphi(d, e) \in \mathcal{E}_\varphi$ следует $d = e$. Шкалу $\mathcal{W}$ системы $\mathcal{K}$ будем называть *критериальной* и её состояния — *критериями*; систему $\mathcal{K}$ — $\mathfrak{k}$-*ограниченной*, если для любых цепочек h_1, h_2 одинаковой длины существует не более $\mathfrak{k}$ критериев w, для которых $\mathcal{W}(w, h_1, h_2) \in \mathcal{E}_\varphi$.

Пример 13. Рассмотрим шкалу $\mathcal{F}^{0,0}$ с операцией $\circ$ из примера 2, $(\mathfrak{A} \times \mathfrak{A})$-шкалу $\mathcal{W} = (\mathbb{Z}, 0, \odot)$, где $m \odot (a, a) = m \odot (b, b) = m$, $m \odot (a, b) = m + 1$ и $m \odot (b, a) = m - 1$, и функцию $\varphi : \mathcal{B}_\mathcal{F} \to \mathbb{Z}$, заданную равенством $\varphi((n_1, m_1), (n_2, m_2)) = n_1 - n_2$. Тогда нетрудно убедиться, что $(\mathcal{W}, \varphi)$ — 1-ограниченная критериальная система для $\mathcal{F}^{0,0}$:

— $\varphi((0, 0), (0, 0)) = 0$;
— $\varphi((n_1, m_1), (n_2, m_2)) \odot (a, a) = n_1 - n_2 = \varphi((n_1, m_1) \circ a, (n_2, m_2) \circ a)$;
— $\varphi((n_1, m_1), (n_2, m_2)) \odot (b, b) = n_1 - n_2 = \varphi((n_1, m_1) \circ b, (n_2, m_2) \circ b)$;
— $\varphi((n_1, m_1), (n_2, m_2)) \odot (a, b) = (n_1 + 1) - n_2 = \varphi((n_1, m_1) \circ a, (n_2, m_2) \circ b)$;
— $\varphi((n_1, m_1), (n_2, m_2)) \odot (b, a) = n_1 - (n_2 + 1) = \varphi((n_1, m_1) \circ b, (n_2, m_2) \circ a)$;
— $\mathcal{E}_\varphi = \{0\}$ и при этом $\varphi((n, m), (n, m)) = n - n = 0$;
— для любых цепочек h_1 и h_2, содержащих соответственно n_1 и n_2 букв a, только для критерия $w = (n_2 - n_1)$ верно $\mathcal{W}(w, h_1, h_2) = 0$.

Утверждение 15. Для любой критериальной системы $(\mathcal{W}, \varphi)$ любой шкалы $\mathcal{F}$ и для любой пары $(d, e) \in \mathcal{B}_\mathcal{F}$ верно следующее: $d = e$ тогда и только тогда, когда $\varphi(d, e) \in \mathcal{E}_\varphi$.

Доказательство. Необходимость следует из определения множества $\mathcal{E}_\varphi$, достаточность — из определения критериальной системы. ■

При обсуждении алгоритмов, использующих критериальную систему $\mathcal{K} = (\mathcal{W}, \varphi)$, где $\mathcal{W} = (W, w^0, \odot)$, будем полагать заранее заданной *алгоритмическую составляющую* этой системы — представление элементов шкалы $\mathcal{W}$, алгоритмы $\mathcal{A}_0^\mathcal{K}$, $\mathcal{A}_e^\mathcal{K}$, $\mathcal{A}_\odot^\mathcal{K}$, $\mathcal{A}_=^\mathcal{K}$ и $\mathcal{A}_\leftarrow^\mathcal{K}$ и функции $\mathfrak{f}_e^\mathcal{K}, \mathfrak{f}_\odot^\mathcal{K}, \mathfrak{f}_=^\mathcal{K}, \mathfrak{f}_\leftarrow^\mathcal{K} : \mathbb{N}_0 \to \mathbb{N}_0$ следующего вида:

— $\mathcal{A}_0^\mathcal{K}() = w^0$;
— $\mathcal{A}_e^\mathcal{K}(w) =$ «Да», если $w \in \mathcal{E}_\varphi$, и «Нет» иначе; $\mathfrak{f}_e^\mathcal{K}(n)$ — сложность этого алгоритма на значениях вида $w = \mathcal{W}(h_1, h_2)$, где $|h_1| = |h_2| \leqslant n$;
— $\mathcal{A}_\odot^\mathcal{K}(w, a, b) = w \odot (a, b)$; $\mathfrak{f}_\odot^\mathcal{K}(n)$ — сложность этого алгоритма на значениях вида $w = \mathcal{W}(h_1, h_2)$, где $|h_1| = |h_2| \leqslant n$;
— $\mathcal{A}_=^\mathcal{K}(w, u) =$ «Да», если $w = u$, и «Нет» иначе; $\mathfrak{f}_=^\mathcal{K}(n)$ — сложность этого алгоритма на значениях вида $w = \mathcal{W}(h_1, h_2)$ и $u = \mathcal{W}(g_1, g_2)$, где $|h_1| = |h_2| \leqslant n$, $|g_1| = |g_2| \leqslant n$;
— выполнение $\mathcal{A}_\leftarrow^\mathcal{K}(x, w)$ устанавливает (копирует) в x значение w; $\mathfrak{f}_\leftarrow^\mathcal{K}(n)$ — сложность этого алгоритма на значениях вида $w = \mathcal{W}(h_1, h_2)$, где $|h_1| = |h_2| \leqslant n$.

Сложностной характеристикой критериальной системы для такой алгоритмической составляющей будем называть функцию $\mathfrak{f}^\mathcal{K} : \mathbb{N}_0 \to \mathbb{N}_0$, задающуюся равенством

$\mathfrak{f}^{\mathcal{K}}(n) = \max(\mathfrak{f}^{\mathcal{K}}_e(n), \mathfrak{f}^{\mathcal{K}}_{\odot}(n), \mathfrak{f}^{\mathcal{K}}_{=}(n), \mathfrak{f}^{\mathcal{K}}_{\leftarrow}(n))$. Далее для наглядности вместо вызовов вспомогательных алгоритмов будем записывать соответственно значения и соотношения w^0, $w \in \mathcal{E}_\varphi$, $w \odot (a, b)$, $w = u$ и описание установки критерия.

Пример 14. Для критериальной системы из примера 13 с алгоритмической составляющей, естественно отвечающей определению этой критериальной системы, сложностная характеристика имеет порядок $O(1)$, так как такой порядок сложности имеют все требуемые алгоритмы: проверка равенства числа-критерия нулю, прибавление или вычитание единицы в зависимости от пары операторов, проверка равенства критериев, копирование числа.

5. Критериальный граф

Предположим заданными полные программы $\pi_1 = (S_1, en_1, EX_1, T_1)$ и $\pi_2 = (S_2, en_2, EX_2, T_2)$, уравновешенную шкалу $\mathcal{F} = (D, d^0, \circ)$ и её критериальную систему $\mathcal{K} = (\mathcal{W}, \varphi)$, где $\mathcal{W} = (W, w^0, \odot)$. Все утверждения формулируются для произвольных таких π_1, π_2, $\mathcal{F}$ и $\mathcal{K}$.

Критериальным графом программ π_1, π_2 и системы $\mathcal{K}$ назовём подграф $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ графа $(\pi_1 \otimes \pi_2) \oplus \mathcal{W}$, содержащий все вершины и все дуги, кроме дуг $((s_1, s_2), w) \xrightarrow{(c_1,c_2)/(a_1,a_2)} v$, в которых $w \in \mathcal{E}_\varphi$ и $c_1 \neq c_2$. Вершины критериального графа будем называть *узлами*, пути в нём — *маршрутами*, значения s_1, s_2 и w — соответственно *состояниями и критерием узла* $((s_1, s_2), w)$. Назовём *φ-образом* вершины $v = ((s_1, s_2), (d_1, d_2))$ графа $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ и пути $v_0 \xrightarrow{(c^1_1,c^1_2)/(a^1_1,a^1_2)} v_1 \xrightarrow{(c^2_1,c^2_2)/(a^2_1,a^2_2)} \dots$ в этом графе соответственно набор $\varphi(v) = ((s_1, s_2), \varphi(d_1, d_2))$ и путь $\varphi(v_0) \xrightarrow{(c^1_1,c^1_2)/(a^1_1,a^1_2)} \varphi(v_1) \xrightarrow{(c^2_1,c^2_2)/(a^2_1,a^2_2)} \dots$

Пример 15. Рассмотрим программы π_1, π'_2 и критериальную систему $\mathcal{K} = (\mathcal{W}, \varphi)$ из примеров 11 и 13. На рис. 12 показан фрагмент графа $\Gamma^{\mathcal{K}}_{\pi_1,\pi'_2}$, содержащий φ-образы всех вершин на рис. 11.

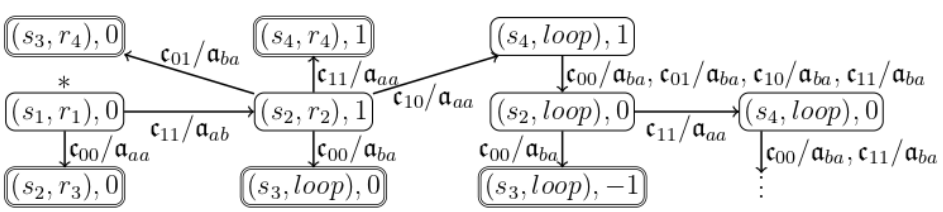

Рис. 12. Фрагмент критериального графа (пример 15)

Утверждение 16. Входными путями в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ являются все φ-образы входных путей в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$, и только они.

Доказательство. Достаточно обосновать следующее: 1) φ-образы всех достижимых вершин графа $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ являются вершинами графа $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$; 2) φ-образом входа в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ является вход в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$; 3) метки дуг, исходящих в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ из достижимой вершины v, совпадают с метками дуг, исходящих из $\varphi(v)$ в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$; 4) если в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ дуга из достижимой вершины v заходит в δ, то в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ дуга из $\varphi(v)$ с той же меткой заходит в $\varphi(\delta)$. Всё это следует из определений графов $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$, $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ и критериальной системы и утверждений 13 и 15. ■

Узел $((s_1, s_2), w)$ графа $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ назовём *достижимым* в этом графе, если он достижим из входа, *нейтральным*, если $w \in \mathcal{E}_\varphi$, и *опровергающим*, если верно одно из следующих условий:

1) $s_1 \in EX_1$ и $s_2 \notin EX_2$;
2) $s_1 \notin EX_1$ и $s_2 \in EX_2$;
3) $s_1 \in EX_1$, $s_2 \in EX_2$ и $w \notin \mathcal{E}_\varphi$.

Пример 16. С учётом изложенного в примере 13 среди узлов критериального графа на рис. 12 нейтральными являются все помеченные числом 0, и только они, опровергающими — все выходы, кроме $((s_3, r_4), 0)$. В этом графе содержится бесконечно много неизображённых опровергающих узлов — например, недостижимый узел $((s_3, r_4), 2)$.

Утверждение 17. Достижимыми опровергающими узлами в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ являются φ-образы всех достижимых опровергающих вершин графа $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$, и только они.

Доказательство. С учётом утверждения 16 и устройства φ-образов путей достаточно показать, что достижимая вершина v графа $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ является опровергающей тогда и только тогда, когда узел $\varphi(v)$ опровергающий. Положим, что $v = ((s_1, s_2), (d_1, d_2))$. Тогда по утверждению 13 верно $(d_1, d_2) \in \mathcal{B}_{\mathcal{F}}$, а значит, $\varphi(v) = ((s_1, s_2), w)$, где $w = \varphi(d_1, d_2)$. Осталось заметить, что состояния программ в v и $\varphi(v)$ одинаковы и из утверждения 15 следует равносильность $(d_1 = d_2 \Leftrightarrow w \in \mathcal{E}_\varphi)$. ■

Утверждение 18. Соотношение $\pi_1 \nsim_{\mathcal{F}} \pi_2$ верно тогда и только тогда, когда в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ существует достижимый опровергающий узел.

Доказательство. Следует из утверждений 17 и 14. ■

Утверждение 19. Если система $\mathcal{K}$ $\mathfrak{k}$-ограниченная, в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ достижимы узлы $v_1, \ldots, v_{\mathfrak{k}+1}$, где $v_i = ((s_1, s_2), w_i)$, $i \in \{1, \ldots, \mathfrak{k}+1\}$, хотя бы одно из состояний s_1, s_2 завершаемо и критерии w_i, $i \in \{1, \ldots, \mathfrak{k}+1\}$, попарно различны, то $\pi_1 \nsim_{\mathcal{F}} \pi_2$.

Доказательство. Положим, что $\|s_1\|_{\pi_1} \leqslant \|s_2\|_{\pi_2}$ (если $\|s_1\|_{\pi_1} > \|s_2\|_{\pi_2}$, то далее достаточно поменять местами индексы 1 и 2). Тогда в π_1 существует путь из s_1 в выход. Рассмотрим кратчайший такой путь $p_0 \xrightarrow{c_1/a_1} p_1 \xrightarrow{c_2/a_2} \ldots \xrightarrow{c_n/a_n} p_n$, $p_0 = s_1$, $p_n \in EX_1$. Тогда по устройству критериального графа, утверждению 12 и неравенству в начале доказательства в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ существуют пути $v_i \xrightarrow{(c_1,c_1)/(a_1,b_1)} ((p_1, q_1), w_1^i) \xrightarrow{(c_2,c_2)/(a_2,b_2)} \ldots \xrightarrow{(c_n,c_n)/(a_n,b_n)} ((p_n, q_n), w_n^i)$, $i \in \{1, \ldots, \mathfrak{k}+1\}$, для некоторых $q_1, \ldots, q_n, b_1, \ldots, b_n, w_1^1, \ldots, w_n^1, \ldots, w_1^{\mathfrak{k}+1}, \ldots, w_n^{\mathfrak{k}+1}$ и при этом $w_n^i = \mathcal{W}(w_i, a_1 \ldots a_n, b_1 \ldots b_n)$. Значит, по $\mathfrak{k}$-ограниченности системы $\mathcal{K}$ хотя бы для одного $i \in \{1, \ldots, \mathfrak{k}+1\}$ верно $w_n^i \notin \mathcal{E}_\varphi$. Тогда хотя бы одна из вершин $v_i' = ((p_n, q_n), w_n^i)$ является опровергающей, при этом все вершины v_i' достижимы в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ и из утверждения 18 следует $\pi_1 \nsim_{\mathcal{F}} \pi_2$. ■

6. Алгоритм проверки эквивалентности программ

В описании и анализе алгоритма проверки эквивалентности программ (алгоритма 4) и вспомогательных алгоритмов 1–3 считаем заданными конечные множества операторов $\mathfrak{A}$ и логических условий $\mathfrak{C}$, шкалу $\mathcal{F} = (D, d^0, \circ)$ и её $\mathfrak{k}$-ограниченную критериальную систему $\mathcal{K} = (\mathcal{W}, \varphi)$ со шкалой $\mathcal{W} = (W, w^0, \odot)$ и некоторой алгоритмической составляющей. Для сохранения понятности изложения действия алгоритмов описываются «высокоуровнево» с использованием математической терминологии. В комментариях приводятся «низкоуровневые» детали, необходимые, в числе прочего, для оценки сложности алгоритмов.

Будем использовать следующие способы представления данных: $\mathfrak{A} = \{1, 2, \ldots, n_{\mathfrak{a}}\}$, $\mathfrak{C} = \{1, 2, \ldots, n_{\mathfrak{c}}\}$; множество состояний S каждой программы имеет вид $\{1, 2, \ldots, n_S\}$;

множество вида $\{1, 2, \ldots, n\}$ представляется числом n; одноместная функция $f : \{1, 2, \ldots, n\} \to X$ — вектором $(f(1), f(2), \ldots, f(n))$; множество $X \subseteq \{1, 2, \ldots, n\}$ — так же, как функция $f : \{1, 2, \ldots, n\} \to \{0, 1\}$, задающаяся равносильностью $f(x) = 1 \Leftrightarrow x \in X$. Двухместная функция $f : \{1, 2, \ldots, n_1\} \times \{1, 2, \ldots, n_2\} \to X$ (в том числе функция переходов программы) представляется матрицей, где $f(i, j)$ — значение элемента матрицы в i-й строке и j-м столбце. Последовательность нефиксированной длины задаётся списком элементов, за исключением цепочки: она представляется вектором элементов. Для цепочки переменной длины сразу выделяется столько ячеек памяти, сколько необходимо для хранения любой цепочки длины, равной наибольшему из размеров рассматриваемых программ. Конечное множество других видов по умолчанию представляется списком своих элементов — добавление элемента происходит в конец списка, проверка принадлежности элемента множеству состоит в проходе по списку с проверкой равенства.

Алгоритм 1. Пополнение программы

Вход: конечная программа $\pi = (S, en, EX, T)$.
Выход: пополнение π' программы π.

1: Произвольно выбрать значения $\mathrm{loop} \notin S$ и $a \in \mathfrak{A}$.
// По выбору представления множества состояний $\mathrm{loop} = n_S + 1$. *Для определённости выбирается* $a = 1$.
2: **Вернуть** $(S \cup \{\mathrm{loop}\}, en, EX, T^{\mathrm{loop},a})$.
// Матрицу $T^{\mathrm{loop},a}$ *можно вычислить так: установить во всех строках этой матрицы, кроме последней, те же значения, что и в* T, *а в каждой ячейке последней строки — значение* (loop, a). *Затем заменить каждое значение* $T^{\mathrm{loop},a}(s, c) = \bot$, *где* $s \notin EX$, *на* $T^{\mathrm{loop},a}(s, c) = (\mathrm{loop}, a)$.

Алгоритм 2. Вычисление завершаемых состояний программы

Вход: конечная полная программа $\pi = (S, en, EX, T)$.
Выход: множество S^f всех завершаемых состояний программы π.

1: Вычислить орграф G, обратный к орграфу π. Добавить в G произвольный простой цикл, содержащий все вершины множества EX и только их.
// Для определённости выбирается цикл, в котором выходы соединяются дугами по возрастанию и самый большой выход соединяется с самым маленьким. Граф G *представляется списком смежности. Вычисление графа* G *производится полным перебором ячеек матрицы* T *с добавлением соответствующих дуг и затем добавлением дуг цикла.*
2: Произвольно выбрать вершину $s \in EX$ и применить к ней и графу G поиск в ширину [27], вычисляющий множество X состояний, достижимых из s.
3: **Вернуть** $S^f = X$.

Алгоритм 3. Обход критериального графа

Вход: конечные полные программы $\pi_1 = (S_1, en_1, EX_1, T_1)$ и $\pi_2 = (S_2, en_2, EX_2, T_2)$ и соответствующие множества S_1^f и S_2^f всех завершаемых состояний этих программ.
Выход: ответ «Да», если $\pi_1 \sim_{\mathcal{F}} \pi_2$, и «Нет» иначе.

// В алгоритме используются следующие вспомогательные значения:

— *набор* $(s_1, s_2, w) \in S_1 \times S_2 \times W$, *в начале выполнения равен* (en_1, en_2, w^0);

— *разметка* $F : S_1 \times S_2 \to 2^W$, *в начале выполнения все значения* F *равны* $\varnothing$ *(используются только конечные подмножества* W*);*

— *множество* $X \subseteq \mathfrak{C} \times \mathfrak{C}$, *значение в начале выполнения неважно;*

— *конечная последовательность* P *элементов множества* $S_1 \times S_2 \times W \times 2^{\mathfrak{C}\times\mathfrak{C}}$, *в начале выполнения эта последовательность пуста.*

1: **Если**
а) ($s_1 \notin S_1^f$ и $s_2 \notin S_2^f$) или
б) $w \in F(s_1, s_2)$, **то**
2: **выход**, ответ «Да».
3: **Если**
а) ($s_1 \in EX_1$ и $s_2 \notin EX_2$) или ($s_1 \notin EX_1$ и $s_2 \in EX_2$) или ($s_1 \in EX_1$, $s_2 \in EX_2$ и $w \notin \mathcal{E}_\varphi$) или
б) $|F(s_1, s_2)| = \mathfrak{k}$, **то**
4: **выход**, ответ «Нет».
5: Добавить в множество $F(s_1, s_2)$ критерий w.
6: Вычислить множество пар $X \subseteq \mathfrak{C} \times \mathfrak{C}$: **если** $w \in \mathcal{E}_\varphi$, **то** $X = \{(c, c) : c \in \mathfrak{C}\}$, **иначе** $X = \mathfrak{C} \times \mathfrak{C}$.
// Для проверки $w \in \mathcal{E}_\varphi$ *используется результат проверки* $w \notin \mathcal{E}_\varphi$ *с шага 3.*
7: **Для всех** $(c_1, c_2) \in X$:
// Значения (c_1, c_2) *перебираются в порядке расположения в списке и удаляются из списка при рассмотрении.*
8: Вычислить значения $(a_1, r_1) = T_1(s_1, c_1)$ и $(a_2, r_2) = T_2(s_2, c_2)$.
9: Заменить набор (s_1, s_2, w) на $(r_1, r_2, w \odot (a_1, a_2))$.
// Перед заменой набор (s_1, s_2, w, X) *для текущего значения* X *добавляется в конец последовательности* P.
10: Выполнить тело алгоритма 3 (основной рекурсивный вызов).
Если его результат «Нет», **то выход**, ответ «Нет».
11: Восстановить значения (s_1, s_2, w) и X, которые были в начале выполнения шага 9.
// Значения берутся из последнего элемента P, *этот элемент удаляется из* P.
12: **Вернуть** «Да».

Лемма 1. Алгоритм 1 имеет сложность $O(\mathfrak{n})$, где $\mathfrak{n}$ — размер программы на входе.

Доказательство. На шаге 1 выполняется $O(1)$ действий. На шаге 2 перебираются $(\mathfrak{n}+1)n_{\mathfrak{c}} = O(\mathfrak{n})$ ячеек матрицы $T^{\mathrm{loop},a}$ и для каждой ячейки выполняется $O(1)$ действий. Значит, суммарно получаем $O(\mathfrak{n})$ действий. ■

Лемма 2. Для любой конечной программы π результат выполнения алгоритма 1 на входе π — это пополнение программы π.

Доказательство. Пусть $\pi = (S, en, EX, T)$. Алгоритм 1 выдаёт программу $(S \cup \{\text{loop}\}, en, EX, T^{\text{loop},a})$ для некоторых $\text{loop} \notin S$ и $a \in \mathfrak{A}$, то есть, по определению, пополнение $\pi^{\text{loop},a}$ программы π. ■

Лемма 3. Алгоритм 2 имеет сложность $O(\mathfrak{n})$, где $\mathfrak{n}$ — размер программы на входе.

Доказательство. На шаге 1 просматривается $\mathfrak{n}n_{\mathfrak{c}} = O(\mathfrak{n})$ ячеек матрицы T и для каждой ячейки выполняется $O(1)$ действий, после чего за $O(\mathfrak{n})$ действий добавляется цикл. Поиск в ширину на шаге 2 имеет сложность $O(\mathfrak{n})$, согласно [27] и тому, что программа имеет $O(\mathfrak{n})$ переходов. Значит, суммарно получаем $O(\mathfrak{n})$ действий. ■

Лемма 4. Для любой конечной полной программы π результат выполнения алгоритма 2 на входе π — это множество всех завершаемых состояний программы π.

Доказательство. Достижимость вершины t из $s \in EX$ на шаге 2 равносильна достижимости t хотя бы из одного выхода в графе, обратном к π, а это равносильно достижимости хотя бы одного выхода из t в π, то есть завершаемости t. ■

Пусть задано выполнение $\mathfrak{E}$ алгоритма 3. Будем называть *итерацией* выполнения $\mathfrak{E}$ однократное последовательное выполнение шагов 1–12, кроме действий в основных рекурсивных вызовах на шаге 10, записью $\mathfrak{E}(i)$ обозначим i-ю итерацию выполнения $\mathfrak{E}$ при нумерации с единицы с упорядочиванием по времени начала. *Узлом итерации* $\mathfrak{I}$ назовём значение $[\mathfrak{I}] = ((s_1, s_2), w)$ в начале итерации $\mathfrak{I}$. Итерацию $\mathfrak{I}_c$ будем считать *ребёнком* итерации $\mathfrak{I}_p$, если итерация $\mathfrak{I}_c$ начинается основным рекурсивным вызовом на шаге 10 итерации $\mathfrak{I}_p$, набор значений $(c_1, c_2, a_1, a_2, r_1, r_2)$ в начале этого шага будем называть *переходным набором* итерации $\mathfrak{I}_c$. *Итерационным путём* назовём путь вида $\mathfrak{I}_0 \xrightarrow{(c_1,\ell_1)/(a_1,b_1)} \mathfrak{I}_1 \xrightarrow{(c_2,\ell_2)/(a_2,b_2)} \dots$, где $\mathfrak{I}_{i+1}$ — ребёнок итерации $\mathfrak{I}_i$ и c_i, ℓ_i, a_i, b_i — первые четыре элемента переходного набора $\mathfrak{I}_{i+1}$. Итерационный путь будем называть *входным*, если он исходит из итерации $\mathfrak{E}(1)$. *Трасса итерационного пути* P — путь, получающийся из P заменой каждой итерации на её узел; *трасса итерации* $\mathfrak{I}$ — трасса входного итерационного пути, ведущего в $\mathfrak{I}$.

Лемма 5. Для любой итерации $\mathfrak{I}$ любого выполнения алгоритма 3 верно следующее: если $[\mathfrak{I}] = ((s_1, s_2), w)$ и в начале итерации $\mathfrak{I}$ имеет место $u \in F(s_1, s_2)$, то существует итерация $\mathfrak{I}'$ с меньшим номером, для которой $[\mathfrak{I}'] = ((s_1, s_2), u)$.

Доказательство. Получить элемент u в множестве $F(s_1, s_2)$ в начале итерации $\mathfrak{I}$ можно, только добавив его выполнением шага 5 до начала этой итерации, то есть на шаге 5 некоторой итерации $\mathfrak{I}'$ с меньшим номером. Для такого добавления u необходимо равенство $[\mathfrak{I}'] = ((s_1, s_2), u)$. ■

Лемма 6. В начале любой итерации любого выполнения алгоритма 3 верно $|F(s_1, s_2)| \leqslant \mathfrak{k}$, где s_1 и s_2 — состояния узла итерации.

Доказательство. В начале выполнения алгоритма $|F(s_1, s_2)| = 0$ для всех $s_1 \in S_1$ и $s_2 \in S_2$. Значения F изменяются только на шаге 5. При выполнении этого шага размер значения $F(s_1, s_2)$ для состояний s_1, s_2 узла итерации увеличивается на 1, а остальные значения F не изменяются. При этом увеличение размера $F(s_1, s_2)$ на шаге 5 возможно только в том случае, если на этой итерации не выполнено условие 3б, это возможно, только если $|F(s_1, s_2)| < \mathfrak{k}$, тогда размер этого множества после добавления элемента не превосходит $\mathfrak{k}$. ■

Лемма 7. Трасса любой итерации $\mathfrak{I}$ любого выполнения алгоритма 3 является входным маршрутом в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$.

Доказательство. Рассмотрим произвольное выполнение $\mathfrak{E}$ алгоритма 3 и применим индукцию по структуре входного итерационного пути, ведущего в $\mathfrak{I}$.

Б а з а: если $\mathfrak{I} = \mathfrak{E}(1)$, то $[\mathfrak{I}] = ((en_1, en_2), w^0)$, трасса итерации $\mathfrak{I}$ состоит из одного узла, и этот узел — вход графа $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$.

И н д у к т и в н ы й п е р е х о д: $\mathfrak{I}$ — ребёнок итерации $\mathfrak{I}_p$, и трасса итерации $\mathfrak{I}_p$ является входным маршрутом в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$. Пусть $[\mathfrak{I}_p] = ((s_1, s_2), w)$ и $(c_1, c_2, a_1, a_2, r_1, r_2)$ — переходный набор итерации $\mathfrak{I}$. По устройству шага 10 и соответствующим определениям трасса итерации $\mathfrak{I}$ получается из трассы итерации $\mathfrak{I}_p$ продолжением на одну дугу $t = (((s_1, s_2), w) \xrightarrow{(c_1,c_2)/(a_1,a_2)} ((r_1, r_2), w \odot (a_1, a_2)))$. Значит, достаточно показать, что t входит в граф $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$.

По устройству шага 9 $(a_1, r_1) = T_1(s_1, c_1)$ и $(a_2, r_2) = T_2(s_2, c_2)$. Значит, t входит в граф $(\pi_1 \otimes \pi_2) \oplus \mathcal{W}$ по определению этого графа. По устройству множества X на шаге 6 если $w \in \mathcal{E}_\varphi$, то $c_1 = c_2$. Значит, в любом случае t содержится в графе $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ по определению этого графа. ■

В леммах 8–10 символом $\mathfrak{n}$ обозначено значение $\max(|\pi_1|, |\pi_2|)$ для программ π_1 и π_2, подающихся на вход алгоритму 3.

Лемма 8. В любом выполнении $\mathfrak{E}$ алгоритма 3 содержится $O(\mathfrak{n}^2)$ итераций.

Доказательство. Если на итерации $\mathfrak{I}$ выполняется шаг 5, то все значения F, кроме $F(s_1, s_2)$, где s_1 и s_2 — состояния узла $[\mathfrak{I}]$, не изменяются, а размер множества $F(s_1, s_2)$ увеличивается на 1 (так как если выполняется шаг 5, то не верно условие 1б, т. е. добавляемый критерий не содержится в $F(s_1, s_2)$). Из этого и леммы 6 следует, что в $\mathfrak{E}$ шаг 5 выполняется не более $\mathfrak{k}\,\mathfrak{n}^2$ раз. На каждой итерации шаги 5 и 7 либо оба не выполняются, либо оба выполняются по одному разу. На шаге 7 не более $|X|$ раз, т. е. не более $|\mathfrak{C}|^2$ раз, выполняется основной рекурсивный вызов, отвечающий одному очередному ребёнку. Следовательно, не более $\mathfrak{k}\,\mathfrak{n}^2$ итераций имеют детей и каждая такая итерация имеет не более $|\mathfrak{C}|^2$ детей. Значит, всего в $\mathfrak{E}$ содержится не более $(1 + |\mathfrak{C}|^2\,\mathfrak{k}\,\mathfrak{n}^2) = O(\mathfrak{n}^2)$ итераций. ■

Лемма 9. Критерий w узла любой итерации $\mathfrak{I}$ любого выполнения $\mathfrak{E}$ алгоритма 3 представим в виде $w = \mathcal{W}(h_1, h_2)$, где $h_1, h_2 \in \mathfrak{A}^*$ и $|h_1| = |h_2| = O(\mathfrak{n}^2)$.

Доказательство. Если $\mathfrak{I} = \mathfrak{E}(1)$, то $[\mathfrak{I}] = ((en_1, en_2), w^0)$, $w^0 = \mathcal{W}(\lambda, \lambda)$ и длины цепочек λ равны 0. Если $\mathfrak{I}$ — ребёнок итерации $\mathfrak{I}_p$, $[\mathfrak{I}_p] = ((s_1, s_2), \mathcal{W}(g_1, g_2))$ и $(a_1, a_2, c_1, c_2, r_1, r_2)$ — переходный набор итерации $\mathfrak{I}$, то $[\mathfrak{I}] = ((r_1, r_2), \mathcal{W}(g_1a_1, g_2a_2))$, и если длины цепочек g_1 и g_2 равны k, то длины цепочек g_1a_1 и g_2a_2 равны $(k+1)$. Значит, критерий w узла любой итерации $\mathfrak{I}$ представим в виде $w = \mathcal{W}(h_1, h_2)$, где длины цепочек h_1, h_2 равны длине входного итерационного пути, ведущего в $\mathcal{I}$. Длина этого пути оценивается как $O(\mathfrak{n}^2)$ по лемме 8 и попарной различности итераций в любом итерационном пути. ■

Лемма 10. Алгоритм 3 имеет сложность $O(\mathfrak{n}^2 f(\mathfrak{n}^2))$ для некоторой функции $f : \mathbb{N}_0 \to \mathbb{N}_0$, удовлетворяющей равенству $f(m) = \mathfrak{f}^{\mathcal{K}}(O(m))$, где $\mathfrak{f}^{\mathcal{K}}$ — сложностная характеристика алгоритмической составляющей системы $\mathcal{K}$, используемой в алгоритме.

Доказательство. По лемме 8 в любом выполнении $\mathfrak{E}$ алгоритма 3 содержится $O(\mathfrak{n}^2)$ итераций. Следовательно, достаточно показать, что любая итерация $\mathfrak{I}$ выполнения $\mathfrak{E}$ имеет сложность $O(f(\mathfrak{n}^2))$ для подходящей функции f.

Положим, что $[\mathfrak{I}] = ((s_1, s_2), w)$. По лемме 9 $w = \mathcal{W}(h_1, h_2)$ для некоторых $h_1, h_2 \in \mathfrak{A}^*$, удовлетворяющих равенствам $|h_1| = |h_2| = O(\mathfrak{n}^2)$, и из этого и устройства алгоритма 3 и его шага 5 следует, что все элементы множества $F(s_1, s_2)$ представимы в виде $\mathcal{W}(g_1, g_2)$, где $g_1, g_2 \in \mathfrak{A}^*$ и $|g_1| = |g_2| = O(\mathfrak{n}^2)$. Положим $f(m) = \mathfrak{f}^{\mathcal{K}}(\widetilde{f}(m))$, где $\widetilde{f}(\mathfrak{n}^2) = O(\mathfrak{n}^2)$ — обозначенная выше оценка длин цепочек h_1, h_2, g_1, g_2. По лемме 6 на каждой итерации проверка условия 3б содержит не более $\mathfrak{k} = O(1)$ проверок равенства критериев и при этой проверке вычисляемый размер множества не превосходит $\mathfrak{k} = O(1)$. На этом основываются оценки сложности, изложенные далее.

Условие 1а проверяется за $O(1)$ действий, 1б — за $O(f(\mathfrak{n}^2))$, 3а — за $O(f(\mathfrak{n}^2))$, 3б — за $O(1)$ действий. Шаг 5 выполняется за $O(f(\mathfrak{n}^2))$ действий. На шаге 6 за $O(1)$ действий вычисляется множество X, содержащее $O(1)$ элементов, и для каждого элемента на шагах 7–11 итерации выполняется $O(f(\mathfrak{n}^2))$ действий. Значит, суммарно на всех шагах итерации выполняется $O(f(\mathfrak{n}^2))$ действий. ■

Лемма 11. Для любых конечных полных программ π_1 и π_2 и множеств их завершаемых состояний S_1^f и S_2^f выполнение алгоритма 3 на входе $(\pi_1, \pi_2, S_1^f, S_2^f)$ имеет результат «Нет» тогда и только тогда, когда $\pi_1 \nsim_{\mathcal{F}} \pi_2$.

Доказательство. Обозначим символом $\mathfrak{E}$ выполнение алгоритма 3 на входе $(\pi_1, \pi_2, S_1^f, S_2^f)$.

Необходимость. Положим, что выполнение $\mathfrak{E}$ имеет результат «Нет». Пусть $\mathfrak{E}(n)$ — последняя итерация выполнения $\mathfrak{E}$ и $[\mathfrak{E}(n)] = ((s_1, s_2), w)$. Из ответа «Нет» и устройства алгоритма следует, что на итерации $\mathfrak{E}(n)$ не выполнены условия шага 1 и выполнено хотя бы одно из условий шага 3. По лемме 7 в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ достижим узел $[\mathfrak{E}(n)]$.

Если на итерации $\mathfrak{E}(n)$ выполнено условие 3а, то узел $[\mathfrak{E}(n)]$ опровергающий и соотношение $\pi_1 \nsim_{\mathcal{F}} \pi_2$ следует из утверждения 18. Далее полагаем, что на $\mathfrak{E}(n)$ выполнено $|F(s_1, s_2)| = \mathfrak{k}$. Так как не выполнено условие 1б, все элементы $F(s_1, s_2)$ отличны от w (и попарно различны). По лемме 5 существуют итерации выполнения $\mathfrak{E}$ с узлами $((s_1, s_2), u)$ для всех $u \in F(s_1, s_2)$. По лемме 7 все эти узлы достижимы в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$. Так как не выполнено условие 1а, хотя бы одно из состояний s_1, s_2 завершаемо. Тогда соотношение $\pi_1 \nsim_{\mathcal{F}} \pi_2$ следует из утверждения 19.

Достаточность. Положим, что $\pi_1 \nsim_{\mathcal{F}} \pi_2$. По устройству алгоритма 3 достаточно показать: а) на шаге 2 итерации $\mathfrak{E}(1)$ не выдаётся ответ «Да»; б) на шаге 4 хотя бы одной итерации $\mathfrak{I}$ выполнения $\mathfrak{E}$ выдаётся ответ «Нет» — тогда, согласно шагу 10, на всех итерациях входного итерационного пути в $\mathfrak{I}$ выдаётся ответ «Нет», в том числе на итерации $\mathfrak{E}(1)$, начинающей этот путь и предоставляющей ответ алгоритма. Покажем это:

а) Условие 1а не выполнено на итерации $\mathfrak{E}(1)$ — иначе входы обеих программ незавершаемы и все вычисления этих программ имеют результат $\perp$, а значит, программы $\mathcal{F}$-эквивалентны, что противоречит предположению достаточности. По описанию алгоритма в начале итерации $\mathfrak{E}(1)$ верно $F(s_1, s_2) = \varnothing$. Значит, и условие 1б не выполнено на итерации $\mathfrak{E}(1)$, и ответ «Да» на шаге 1 этой итерации не выдаётся.

б) По утверждению 18 в графе $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ существует входной маршрут $\rho = (v_0 \xrightarrow{(c_1,\ell_1)/(a_1,b_1)} v_1 \xrightarrow{(c_2,\ell_2)/(a_2,b_2)} \ldots \xrightarrow{(c_n,\ell_n)/(a_n,b_n)} v_n)$ в некоторый опровергающий узел v_n. Положим, что $v_i = ((s_i, r_i), w_i)$ и $s_n \in EX_1$ (иначе по определению опровергающего узла $r_n \in EX_2$ и в дальнейших рассуждениях достаточно поменять ролями (s_i, c_i, a_i) и (r_i, ℓ_i, b_i) и индексы 1 и 2, относящиеся к программам). Возможны два случая:

С л у ч а й 1: v_n является узлом какой-либо итерации выполнения $\mathfrak{E}$. Рассмотрим итерацию $\mathfrak{E}(m)$ с наименьшим номером m, узлом которой является v_n. Условие 1а не выполнено на $\mathfrak{E}(m)$, так как s_n завершаемо. Условие 1б также не выполнено на $\mathfrak{E}(m)$ — иначе по лемме 5 существовала бы итерация $\mathfrak{E}(m')$ с номером $m' < m$ и узлом v_n, чего не может быть по выбору m. Значит, на итерации $\mathfrak{E}(m)$ на шаге 2 не выдаётся ответ «Да» и выполняется шаг 3. Так как узел v_n опровергающий, на итерации $\mathfrak{E}(m)$ выполнено условие 3а и на шаге 4 выдаётся ответ «Нет».

С л у ч а й 2: v_n не является узлом ни одной итерации выполнения $\mathfrak{E}$. Рассмотрим наименьший номер k, для которого v_{k+1} не является узлом ни одной итерации выполнения $\mathfrak{E}$. Так как $v_0 = [\mathfrak{E}(1)]$, то $k \in \{0, 1, \ldots, n-1\}$. Рассмотрим итерацию $\mathfrak{E}(m)$ с наименьшим номером m, узлом которой является v_k. По утверждениям 16, 12 и 1 в π_1 существует путь из s_k в s_n. Значит, состояние s_k завершаемо и условие 1а не выполнено на $\mathfrak{E}(m)$. Условие 1б не выполнено на $\mathfrak{E}(m)$ по тем же соображениям, что и в случае 1. Значит, на шаге 2 итерации $\mathfrak{E}(m)$ не выдаётся ответ «Да».

Предположим от противного, что на шаге 4 итерации $\mathfrak{E}(m)$ не выдаётся ответ «Нет». Тогда на итерации $\mathfrak{E}(m)$ выполняются шаги 6–11. Согласно устройству шагов 6–11 и определению графа $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$, верно $(c_{k+1}, \ell_{k+1}) \in X$ и у итерации $\mathfrak{E}(m)$ есть ребёнок $\mathfrak{I}_c$, для которого логическими условиями переходного набора являются c_{k+1} и ℓ_{k+1} и $[\mathfrak{I}_c] = v_{k+1}$. Получено противоречие с тем, что v_{k+1} по выбору k не является узлом ни одной итерации. ■

Алгоритм 4. Проверка эквивалентности программ

Вход: конечные программы $\pi_1 = (S_1, en_1, EX_1, T_1)$ и $\pi_2 = (S_2, en_2, EX_2, T_2)$.
Выход: ответ «Да», если $\pi_1 \sim_{\mathcal{F}} \pi_2$, и «Нет» иначе.

1: Вычислить значения $\pi_1' = \mathcal{A}_1(\pi_1)$, $\pi_2' = \mathcal{A}_1(\pi_2)$, $S_1^f = \mathcal{A}_2(\pi_1')$ и $S_2^f = \mathcal{A}_2(\pi_2')$, где $\mathcal{A}_1$ и $\mathcal{A}_2$ — алгоритмы 1 и 2 соответственно.
2: Вычислить и **вернуть** значение $\mathcal{A}_3(\pi_1', \pi_2', S_1^f, S_2^f)$, где $\mathcal{A}_3$ — алгоритм 3.

Теорема 1. Алгоритм 4 имеет сложность $O(\mathfrak{n}^2 f(\mathfrak{n}^2))$ для некоторой функции $f : \mathbb{N}_0 \to \mathbb{N}_0$, удовлетворяющей равенству $f(m) = \mathfrak{f}^{\mathcal{K}}(O(m))$, где $\mathfrak{n} = \max(|\pi_1|, |\pi_2|)$ для программ π_1 и π_2 на входе и $\mathfrak{f}^{\mathcal{K}}$ — сложностная характеристика алгоритмической составляющей системы $\mathcal{K}$, используемой в алгоритме 4.

Доказательство. По леммам 1 и 3, шаг 1 имеет сложность $O(\mathfrak{n})$. По устройству пополнения программы $|\pi_1'| = |\pi_1| + 1$ и $|\pi_2'| = |\pi_2| + 1$. Значит, на шаге 2 алгоритм 3 выполняется на программах размера не более $\mathfrak{n}+1$; по лемме 10 шаг 2 имеет сложность $O((\mathfrak{n}+1)^2 f(\mathfrak{n}^2)) = O(\mathfrak{n}^2 f(\mathfrak{n}^2))$ для некоторой f, для которой $f(m^2) = \mathfrak{f}^{\mathcal{K}}(O((m+1)^2)) = \mathfrak{f}^{\mathcal{K}}(O(m^2))$, а значит, $f(m) = \mathfrak{f}^{\mathcal{K}}(O(m))$. ■

Теорема 2. Для любых конечных программ π_1 и π_2 выполнение алгоритма 4 на входе (π_1, π_2) имеет результат «Да» тогда и только тогда, когда $\pi_1 \sim_{\mathcal{F}} \pi_2$.

Доказательство. Следует из устройства алгоритма, утверждения 4 и лемм 2, 4 и 11. ■

Пример 17. Рассмотрим программы π_1, π_2 из примера 1, шкалу $\mathcal{F}^{0,0}$ из примера 2, 1-ограниченную критериальную систему $\mathcal{K} = (\mathcal{W}, \varphi)$ из примера 13 и алгоритмическую составляющую этой системы со сложностной характеристикой $O(1)$, как отмечено в примере 14. Тогда алгоритм проверки эквивалентности (алгоритм 4) выполняется следующим образом. На шаге 1 вычисляются программы $\pi_1' = \pi_1^{\mathrm{loop}',a}$ и

$\pi_2' = \pi_2^{\text{loop},a}$ и множества $S_1^f = \{\text{loop}'\}$ и $S_2^f = \{\text{loop}\}$ (программа π_2' представлена на рис. 7). На шаге 2 выполнение алгоритма 3 представляет собой обход графа $\Gamma^{\mathcal{K}}_{\pi_1',\pi_2'}$ в глубину [27] с произвольным выбором порядка исходящих дуг. Состояние loop' недостижимо из входа в π_1', поэтому обходятся только узлы графа $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2'}$ (фрагмент этого графа представлен на рис. 12), и условие 1а алгоритма 3 не выполняется ни для одного посещённого узла. Если узел посещается больше одного раза, то по условию 1б алгоритма 3 дуги, исходящие из этого узла, не исследуются, что соответствует обходу в глубину. Если посещается опровергающий узел (как, например, отмеченные в примере 16), то по условию 3а алгоритма 3 алгоритм 4 завершается с ответом «Нет». Если посещаются узлы вида $((s,r),n_1)$ и $((s,r),n_2)$, где $n_1 \neq n_2$ (например, узлы $((s_3,\text{loop}),0)$ и $((s_3,\text{loop}),-1)$ на рис. 12), то по 1-ограниченности системы $\mathcal{K}$ и условию 3б алгоритма 3 алгоритм 4 завершается с ответом «Нет». Если обход графа завершается без ответа «Нет» по указанным причинам, то алгоритм 4 завершается с ответом «Да». В данном примере алгоритм обязательно завершается с ответом «Нет» в связи с наличием отмеченных выше узлов, этим обосновывается соотношение $\pi_1 \nsim_{\mathcal{F}^{0,0}} \pi_2$.

7. Применение алгоритма проверки эквивалентности программ

Рассмотрим $\mathfrak{A}$-моноид $\mathcal{M} = (M, \varepsilon, \circ)$. Записью $\mathcal{B}_{\mathcal{M}}$ обозначим подмоноид моноида $\mathcal{M} \times \mathcal{M}$ с множеством элементов $\{(\mathcal{M}(h), \mathcal{M}(g)) : h, g \in \mathfrak{A}^*,\ |h| = |g|\}$; записью $\mathcal{F}_{\mathcal{M}}$ — *шкалу моноида* $\mathcal{M}$, получающуюся из $\mathcal{M}$ заменой операции $\circ$ на её сужение на множество $M \times \mathfrak{A}$. *Моноидальной системой* для $\mathcal{M}$ назовём систему $\mathcal{K} = (\mathcal{W}, \mathcal{U}, w^+, w^*, \varphi)$, где $\mathcal{W} = (W, \epsilon, \odot)$ — конечно порождённый моноид; U — его подмоноид; $w^+, w^* \in \mathcal{W}$; φ — гомоморфизм моноида $\mathcal{B}_{\mathcal{M}}$ на $\mathcal{U}$ и для любой пары $(m_1, m_2) \in \mathcal{B}_{\mathcal{M}}$ справедлива равносильность $m_1 = m_2 \Leftrightarrow w^+ \odot \varphi(m_1, m_2) \odot w^* = \epsilon$. Такую систему $\mathcal{K}$ назовём $\mathfrak{k}$-*ограниченной*, если для любого элемента $w \in U \cdot w^*$ существует не более $\mathfrak{k}$ элементов $u \in w^+ \cdot U$, удовлетворяющих равенству $u \odot w = \epsilon$. *Производной системой* для $\mathcal{K}$ назовём пару $\mathcal{K}' = (\mathcal{W}', \psi)$, в которой $\mathcal{W}' = (W', w^+, \ominus)$, $W' = w^+ \odot U$, $\psi : \mathcal{B}_{\mathcal{F}_{\mathcal{M}}} \to W'$ задаётся равенством $\psi(m_1, m_2) = w^+ \odot \varphi(m_1, m_2)$ и $\ominus : W' \times (\mathfrak{A} \times \mathfrak{A}) \to W'$ — равенством $w \ominus (a, b) = w \odot \varphi(a, b)$.

Пример 18. Шкала $\mathcal{F}^{0,0}$ из примера 2 — это шкала свободного коммутативного моноида $\mathcal{M}$, порождённого множеством $\mathfrak{A} = \{a, b\}$. 1-Ограниченная моноидальная система $\mathcal{K} = (\mathcal{W}, \mathcal{U}, w^+, w^*, \varphi)$ для $\mathcal{M}$ может быть устроена так: $\mathcal{W} = \mathcal{U} = (\mathbb{Z}, 0, +)$; $w^+ = w^* = 0$; $\varphi(\mathcal{M}(h), \mathcal{M}(g)) = (n - m)$, где n и m — количество букв a в словах h и g соответственно. Критериальная система из примера 13 является производной системой для $\mathcal{K}$.

Лемма 12. Пусть $\mathcal{M}$ — $\mathfrak{A}$-моноид, $\mathcal{K}$ — его моноидальная система и $\mathcal{K}'$ — система, производная для $\mathcal{K}$. Тогда $\mathcal{K}'$ — это критериальная система шкалы $\mathcal{F}_{\mathcal{M}}$, и система $\mathcal{K}$ $\mathfrak{k}$-ограниченна тогда и только тогда, когда $\mathfrak{k}$-ограниченна система $\mathcal{K}'$.

Доказательство. Положим, что $\mathcal{M}$, $\mathcal{K}$ и $\mathcal{K}'$ имеют такой вид, как в определениях перед леммой, и $\mathcal{F}_{\mathcal{M}} = (M, \varepsilon, \cdot)$. Заметим, что $\mathcal{R}_{\mathcal{F}_{\mathcal{M}}} = M$ (так как $\mathcal{M}$ порождён множеством $\mathfrak{A}$) и множество элементов моноида $\mathcal{B}_{\mathcal{M}}$ есть $\mathcal{B}_{\mathcal{F}_{\mathcal{M}}}$. Тогда верно следующее (по соответствующим определениям):

1) $\psi(\varepsilon, \varepsilon) = w^+ \odot \varphi(\varepsilon, \varepsilon) = w^+ \odot \epsilon = w^+$;
2) для любых $m_1, m_2 \in \mathcal{M}$ имеет место $\psi(m_1, m_2) \ominus (a, b) = w^+ \odot \varphi(m_1, m_2) \odot \odot \varphi(a, b) = w^+ \odot \varphi(m_1 \circ a, m_1 \circ b) = \psi(m_1 \cdot a, m_2 \cdot b)$;
3) для любых $m_1, m_2 \in \mathcal{B}_{\mathcal{M}}$ если $w^+ \odot \varphi(m_1, m_2) \in \mathcal{E}_{\psi}$, то существует $m \in \mathcal{M}$, удовлетворяющий равенству $w^+ \odot \varphi(m_1, m_2) = \psi(m, m)$, т. е. $w^+ \odot \varphi(m_1, m_2) =$

$= w^+ \odot \varphi(m,m)$, а значит, $w^+ \odot \varphi(m_1,m_2) \odot w^* = w^+ \odot \varphi(m,m) \odot w^*$, при этом по определению моноидальной системы $w^+ \odot \varphi(m,m) \odot w^* = \epsilon$, а значит, $w^+ \odot \varphi(m_1,m_2) \odot w^* = \epsilon$; по тому же определению $m_1 = m_2$;

4) для любых $(m_1,m_2) \in \mathcal{B}_\mathcal{M}$ если $\psi(m_1,m_2) \in \mathcal{E}_\psi$, то $w^+ \odot \varphi(m_1,m_2) \in \mathcal{E}_\psi$ и по п. 3 верно $m_1 = m_2$.

Из пп. 1, 2 и 4 следует, что $\mathcal{K}'$ — критериальная система для $\mathcal{F}_\mathcal{M}$.

Положим, что система $\mathcal{K}'$ не $\mathfrak{k}$-ограниченна. Тогда существуют цепочки $h = a_1 \dots a_k$, $g = b_1 \dots b_k$ и попарно различные критерии $w_1, \dots, w_{\mathfrak{k}+1} \in W' = w^+ \odot U$, удовлетворяющие соотношению $\mathcal{W}'(w_i,h,g) \in \mathcal{E}_\psi$ для всех $i \in \{1,\dots,\mathfrak{k}+1\}$. При этом $\mathcal{W}'(w_i,h,g) = w_i \ominus (a_1,b_1) \ominus \dots \ominus (a_k,b_k) = w_i \odot \varphi(a_1,b_1) \odot \dots \odot \varphi(a_k,b_k) = w_i \odot \varphi(m_1,m_2)$ для $m_1 = \mathcal{M}(h)$ и $m_2 = \mathcal{M}(g)$. Соотношение $w_i \in w^+ \odot U$ означает, что w_i представим в виде $w^+ \odot \varphi(m_1^i,m_2^i)$ для некоторой пары $(m_1^i,m_2^i) \in \mathcal{B}_\mathcal{M}$. Значит, $w_i \odot \varphi(m_1,m_2) = w^+ \odot \varphi(m_1^i,m_2^i) \odot \varphi(m_1,m_2) = w^+ \odot \varphi(m_1^i \circ m_1, m_2^i \circ m_2) \in \mathcal{E}_\psi$, по доказанному п. 3 $m_1^i \circ m_1 = m_2^i \circ m_2$ и по определению моноидальной системы $w^+ \odot \varphi(m_1^i,m_2^i) \odot \varphi(m_1,m_2) \odot w^* = \epsilon$. То есть существуют $w' = w \odot w^* \in U \odot w^*$ и попарно различные $w_i \in w^+ \odot U$, $i \in \{1,\dots,\mathfrak{k}+1\}$, для которых $w_i \odot w' = \epsilon$. Следовательно, система $\mathcal{K}$ не $\mathfrak{k}$-ограниченна.

Положим, что система $\mathcal{K}$ не $\mathfrak{k}$-ограниченна. Тогда существуют $w \in U \odot w^*$ и попарно различные $w_1, \dots, w_{\mathfrak{k}+1} \in w^+ \odot U$, для которых $w_i \odot w = \epsilon$. По устройству классов $U \odot w^*$ и $w^+ \odot U$, $w = \varphi(m_1,m_2) \odot w^*$ для некоторой пары $(m_1,m_2) \in \mathcal{B}_\mathcal{M}$ и $w_i = w^+ \odot \varphi(m_1^i,m_2^i)$ для некоторой пары $(m_1^i,m_2^i) \in \mathcal{B}_\mathcal{M}$ для каждого $i \in \{1,\dots,\mathfrak{k}+1\}$. Тогда для каждого такого i верно следующее (по соответствующим определениям): $w^+ \odot \varphi(m_1^i \circ m_1, m_2^i \circ m_2) \odot w^* = w^+ \odot \varphi(m_1^i,m_2^i) \odot \varphi(m_1,m_2) \odot w^* = w_i \odot w = \epsilon$; $m_1^i \circ m_1 = m_2^i \circ m_2$ по определению моноидальной системы; $\psi(m_1^i \circ m_1, m_2^i \circ m_2) \in \mathcal{E}_\psi$ по утверждению 15; $m_1 = \mathcal{M}(h)$ и $m_2 = \mathcal{M}(g)$ для некоторых цепочек h, g одинаковой длины, так как $\mathfrak{A}$ — множество образующих моноида $\mathcal{M}$ и $(m_1,m_2) \in \mathcal{B}_\mathcal{M}$. Положим, что $h = a_1 \dots a_k$ и $g = b_1 \dots b_k$, где $k \in \mathbb{N}_0$ и $a_1,\dots,a_k,b_1,\dots,b_k \in \mathfrak{A}$; $\psi(m_1^i \circ m_1, m_2^i \circ m_2) = w^+ \odot \varphi(m_1^i \circ m_1, m_2^i \circ m_2) = w^+ \odot \varphi(m_1^i,m_2^i) \odot \varphi(a_1,b_1) \odot \dots \odot \varphi(a_k,b_k) = w_i \ominus (a_1,b_1) \ominus \dots \ominus (a_k,b_k) = \mathcal{W}'(w_i,h,g)$. Значит, $\mathcal{W}'(w_i,h,g) \in \mathcal{E}_\psi$, и так как это верно для всех $i \in \{1,\dots,\mathfrak{k}+1\}$, то система $\mathcal{K}'$ не является $\mathfrak{k}$-ограниченной. ■

Моноидом условной эквивалентности относительно множества $J \subseteq \mathfrak{A} \times \mathfrak{A} \times \mathfrak{A}$ назовём $\mathfrak{A}$-моноид с определяющими соотношениями $\{ca = cb : (a,b,c) \in J\}$. Из леммы 12, устройства и свойств моноидальных систем, описанных в [4, разд. 5] (там они называются критериальными системами), и устройства соответствующих производных систем следуют приведённые далее леммы 13–16.

Лемма 13. Для свободного $\mathfrak{A}$-моноида $\mathcal{M}$ существуют 1-ограниченная критериальная система $\mathcal{K}$ шкалы $\mathcal{F}_\mathcal{M}$ и её алгоритмическая составляющая со сложностной характеристикой $\mathfrak{f}^\mathcal{K}(n) = O(1)$.

Лемма 14. Для свободного коммутативного $\mathfrak{A}$-моноида $\mathcal{M}$ существуют 1-ограниченная критериальная система $\mathcal{K}$ шкалы $\mathcal{F}_\mathcal{M}$ и её алгоритмическая составляющая со сложностной характеристикой $\mathfrak{f}^\mathcal{K}(n) = O(1)$.

Лемма 15. Для любого частично коммутативного $\mathfrak{A}$-моноида $\mathcal{M}$ существуют 1-ограниченная критериальная система $\mathcal{K}$ шкалы $\mathcal{F}_\mathcal{M}$ и её алгоритмическая составляющая со сложностной характеристикой $\mathfrak{f}^\mathcal{K}(n) = O(n)$.

Лемма 16. Для любого моноида условной эквивалентности $\mathcal{M}$ существуют $2^{|\mathfrak{A}|}$-ограниченная критериальная система $\mathcal{K}$ шкалы $\mathcal{F}_\mathcal{M}$ и её алгоритмическая составляющая со сложностной характеристикой $\mathfrak{f}^\mathcal{K}(n) = O(1)$.

На основании этих лемм и алгоритма 4 можно получить соответствующие результаты о проверке эквивалентности программ, приведённые далее в теоремах 3–6. Прикладные причины рассмотрения шкал, для которых сформулированы эти теоремы, можно подробно изучить, например, в [4].

Теорема 3. Существует алгоритм проверки сильной эквивалентности пропозициональных программ Мили, имеющий сложность $O(n^2)$.

Доказательство. Заметим, что сильная эквивалентность программ равносильна их эквивалентности на шкале свободного моноида: необходимость — по определению сильной эквивалентности, достаточность сформулирована в [4, следствие 3]. Справедливость теоремы 3 следует из этого, теорем 1 и 2 и леммы 13. ■

Теорема 4. Для любого свободного коммутативного $\mathfrak{A}$-моноида существует алгоритм проверки эквивалентности пропозициональных программ Мили на шкале этого моноида, имеющий сложность $O(n^2)$.

Доказательство. Следует из теорем 1 и 2 и леммы 14. ■

Теорема 5. Для любого частично коммутативного $\mathfrak{A}$-моноида существует алгоритм проверки эквивалентности пропозициональных программ Мили на шкале этого моноида, имеющий сложность $O(n^4)$.

Доказательство. Следует из теорем 1 и 2 и леммы 15. ■

Теорема 6. Для любого моноида условной эквивалентности существует алгоритм проверки эквивалентности пропозициональных программ Мили на шкале этого моноида, имеющий сложность $O(n^2)$.

Доказательство. Следует из теорем 1 и 2 и леммы 16. ■

Заключение

Как отмечалось во введении, соотношение между ПППЗ и ППМ схоже с соотношением между автоматами Мура и Мили, то есть модель ППМ можно считать в некотором роде более общей по сравнению с ПППЗ. Но всё же эти модели, вообще говоря, несравнимы по тем же причинам, отмеченным во введении, по которым несравнимы модели ПППЗ и дискретных преобразователей Глушкова — Летичевского. Исследование соотношения между этими моделями оставлено на будущее.

Ключевые результаты данной работы — это переложение результатов [4] с ПППЗ на ППМ: техники совместных вычислений (понятие критериального графа, алгоритм 4 и теоремы 1 и 2) и эффективных алгоритмов проверки эквивалентности ППМ на некоторых полезных шкалах, получающихся применением этой техники (теоремы 3–6). Ещё один результат, представляющий интерес, — это лемма 12, позволяющая при получении алгоритмов проверки эквивалентности ПППЗ на основе техники из [4] немедленно в качестве следствия получать настолько же эффективные аналогичные алгоритмы проверки эквивалентности ППМ. Кроме того, можно выделить ещё несколько особенностей полученных результатов, показывающих их ценность.

В работах, использующих технику совместных вычислений с критериальными системами [4, 8–11, 14, 20, 21, 31–37], рассматриваются только шкалы $\mathfrak{A}$-моноидов, имеющие критериальные системы, основывающиеся на конечно порождённых моноидах аналогично тому, как в данной работе задание моноидальной системы начинается с такого моноида. В работе показано, что можно применять эту технику и к шкалам, не базирующимся на моноидах (согласно теоремам 1 и 2), и не основывать понятие критериальной системы на моноидах. В лемме 12 показано, что понятие критериальной

системы из [4], опирающееся на моноиды, является частным случаем понятия критериальной системы, введённого в данной работе.

Техника совместных вычислений в данной работе по сравнению с [4] заметно приближена к технике проверки эквивалентности конечных автоматов с помощью их декартова произведения [22, 38], что выражается, в числе прочего, в использовании в ключевых определениях операций $\oplus$ и $\otimes$, являющихся по сути разновидностями декартова произведения вычислителей автоматного типа.

Кроме того, в работе исправлен ряд огрехов, содержащихся в [4] и проявляющихся в остальных работах, посвящённых ПППЗ и технике совместных вычислений:

1. Явно указан способ подсчёта сложности алгоритмов, включая модель сложности и способы представления данных. Для результатов, констатирующих или опровергающих полиномиальную разрешимость, это было бы неважно, но когда речь идёт о более точных оценках сложности, это становится важным.
2. Рассуждения о сравнении преобразователей за логарифмическое время в доказательстве теоремы 7 работы [4] склоняют к тому, что в [4] для подсчёта сложности используется модель машин Тьюринга или родственная ей. В данной работе вместо неё используется более широко применяющаяся на практике модель RAM-машин и в связи с этим получаются оценки сложности, более близкие к практике.
3. Алгоритмы снабжены всеми подробностями, необходимыми для анализа и подсчёта сложности, и не содержат существенных недосказанностей, которые в [4] приводят, например, к тому, что:
   - в формулировке теоремы 7 используется сложность $\mathfrak{f}^{\mathcal{K}}_{=}$, а следовало бы использовать $\mathfrak{f}^{\mathcal{K}}$ (если применить обозначения данной работы по аналогии);
   - после изучения доказательства теоремы 7 остаётся сомнение, не потерян ли в оценке сложности какой-либо дополнительный множитель, проистекающий из копирования данных и особенностей работы со структурами данных (достоверный вывод, что не потерян, можно сделать только после дополнительного не очень тривиального анализа обоснования);
   - в теореме 10 без достаточных пояснений приводится оценка $O(n^3 \log n)$, тогда как, согласно лемме 15 данной работы, разумно было бы предположить оценку не лучше чем $O(n^4)$ (найти, откуда следовала бы оценка $O(n^3 \log n)$, не удалось).

В будущем планируется: а) обосновать, что модель ППМ можно считать обобщением модели ПППЗ; б) усовершенствовать технику совместных вычислений для получения более низкого порядка сложности соответствующих алгоритмов проверки эквивалентности; в) применить полученные наработки для аналогичного усовершенствования известных смежных результатов и затем для установления новых фактов об эффективной разрешимости проблемы эквивалентности в моделях программ.

ЛИТЕРАТУРА

1. *Rice H. G.* Classes of recursively enumerable sets and their decision problems // Trans. AMS. 1953. V. 74. P. 358–366.
2. *Клини С. К.* Введение в метаматематику. М.: ИЛ, 1957.
3. *Глушков В. М., Летичевский А. А.* Теория дискретных преобразователей // Избранные вопросы алгебры и логики. Новосибирск: Наука, Сибирское отделение, 1973. С. 5–39.

4. *Захаров В. А.* Быстрые алгоритмы разрешения эквивалентности операторных программ на уравновешенных шкалах // Матем. вопр. кибернетики. 1998. Вып. 7. С. 303–324.
5. *Летичевский А. А.* Функциональная эквивалентность дискретных преобразователей III // Кибернетика. 1972. №1. С. 1–4.
6. *Летичевский А. А., Смикун Л. Б.* Об одном классе групп с разрешимой проблемой эквивалентности // Докл. АН СССР. 1976. Т. 227. №1. С. 36–38.
7. *Захаров В. А., Подымов В. В.* Применение алгоритмов проверки эквивалентности для оптимизации программ // Труды ИСП РАН. 2015. Т. 27. Вып. 4. С. 145–174.
8. *Zakharov V. A.* An efficient and unified approach to the decidability of equivalence of propositional programs // LNCS. 1998. V. 1443. P. 247–258.
9. *Захаров В. А.* Быстрые алгоритмы разрешения эквивалентности пропозициональных операторных программ на упорядоченных полугрупповых шкалах // Вестн. Моск. ун-та. Сер. 15. Вычислительная математика и кибернетика. 1999. № 3. С. 29–35.
10. *Захаров В. А.* О проблеме эквивалентности операторных программ на уравновешенных однородных обратимых шкалах // Матем. вопр. кибернетики. 2001. Вып. 10. С. 155–166.
11. *Zakharov V. A.* The equivalence problem for computational models: Decidable and undecidable cases // LNCS. 2001. V. 2055. P. 133–152.
12. *Zakharyaschev I. M. and Zakharov V. A.* On the equivalence-checking problem for polysemantic models of sequential programs // Труды ИСП РАН. 2004. Т. 6. С. 179–198.
13. *Podlovchenko R. I., Rusakov D. M., and Zakharov V. A.* The equivalence problem for programs with mode switching is PSPACE-complete // Труды ИСП РАН. 2006. Т. 11. С. 109–128.
14. *Щербина В. Л., Захаров В. А.* Эффективные алгоритмы проверки эквивалентности программ в моделях, связанных с обработкой прерываний // Вестн. Моск. ун-та. Сер. 15. Вычислительная математика и кибернетика. 2008. №2. С. 33–41.
15. *Подловченко Р. И., Кузюрин Н. Н., Щербина В. Л., Захаров В. А.* Использование алгебраических моделей программ для обнаружения метаморфного вредоносного кода // Фундамент. и прикл. матем. 2009. Т. 15. №5. С. 181–198.
16. *Zakharov V. A.* Program equivalence checking by two-tape automata // Cybernetics and Systems Analysis. 2010. V. 46. No. 4. P. 554–562.
17. *Подымов В. В., Захаров В. А.* О двухленточных машинах, описывающих полугруппы с сокращением // Проблемы теоретической кибернетики. Материалы XVI Междунар. конф. (Нижний Новгород, 20–25 июня 2011 г.). Н. Новгород: Изд-во Нижегор. ун-та, 2011. С. 372–375.
18. *Захаров В. А.* Модели и алгоритмы в задаче проверки эквивалентности программ // Материалы XI Междунар. семинара «Дискретная математика и ее приложения», посвященного 80-летию со дня рождения академика О. Б. Лупанова (Москва, МГУ, 18–23 июня 2012 г.), М.: Изд-во механико-математического факультета МГУ, 2012. С. 53–62.
19. *Подловченко Р. И., Захаров В. А.* О двух методах распознавания эквивалентности в алгебраических моделях программ // Интеллектуальные системы. 2013. Т. 17. №1–4. С. 366–370.
20. *Подымов В. В., Захаров В. А.* Полиномиальный алгоритм проверки эквивалентности в модели программ с перестановочными и подавляемыми операторами // Труды ИСП РАН. 2014. Т. 26. Вып. 3. С. 145–166.
21. *Подымов В. В.* Улучшение алгоритмов проверки эквивалентности операторных программ при помощи анализа весов вершин // Ломоносовские чтения-2021. Секция Вычислительной математики и кибернетики. М.: Изд-во Моск. ун-та, 2021. С. 124–125.
22. *Карпов Ю. Г.* Теория автоматов. СПб.: Питер, 2003.

23. *Zakharov V. A. and Zakharyaschev I. M.* On the equivalence checking problem for a model of programs related with muti-tape automata // LNCS. 2005. V. 3317. P. 293–305.
24. *Пентус А. Е., Пентус М. Р.* Теория формальных языков: учеб. пособие. М.: Изд-во ЦПИ при механико-математическом факультете МГУ, 2004.
25. *Гаврилов Г. П., Сапоженко А. А.* Задачи и упражнения по дискретной математике: учеб. пособие. 3-е изд., перераб. М.: Физматлит, 2005.
26. *Зорич В. А.* Математический анализ. Ч. I. 4-е изд., испр. М.: МЦНМО, 2002.
27. *Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К.* Алгоритмы: построение и анализ. 3-е изд. М.: ООО «И. Д. Вильямс», 2013.
28. *Лалльеман Ж.* Полугруппы и комбинаторные приложения. М.: Мир, 1985.
29. *Клиффорд А., Престон Г.* Алгебраическая теория полугрупп. Т. 1. М.: Мир, 1972.
30. *Страуструп Б.* Язык программирования C++. Краткий курс. 2-е изд. СПб.: ООО «Диалектика», 2019.
31. *Захаров В. А.* Об эффективной разрешимости проблемы эквивалентности линейных унарных рекурсивных программ // Матем. вопр. кибернетики. 1999. Т. 8. С. 255–273.
32. *Захаров В. А.* Об эквивалентности потоковых программ // Материалы XI Междунар. семинара «Дискретная математика и ее приложения», посвященного 80-летию со дня рождения академика О. Б. Лупанова (Москва, МГУ, 18–23 июня 2012 г.), М.: Изд-во механико-математического факультета МГУ, 2012. С. 119–122.
33. *Подымов В. В.* Алгоритм проверки эквивалентности линейных унарных рекурсивных программ на упорядоченных полугрупповых шкалах // Вестн. Моск. ун-та. Сер. 15. Вычислительная математика и кибернетика. 2012. № 4. С. 37–43.
34. *Захаров В. А.* Об эквивалентности ограниченно недетерминированных автоматов-преобразователей над полугруппами // Проблемы теоретической кибернетики. Материалы XVII Междунар. конф. (Казань, 16–20 июня 2014 г.). Казань: Отечество, 2014. С. 100–102.
35. *Захаров В. А.* Моделирование и анализ поведения последовательных реагирующих программ // Труды ИСП РАН. 2015. Т. 27. № 2. С. 221–250.
36. *Подымов В. В.* Алгоритмы проверки эквивалентности программ с процедурами в прогрессивных полугрупповых перегородчатых моделях // Вестн. Моск. ун-та. Сер. 15. Вычислительная математика и кибернетика. 2019. № 4. С. 37–44.
37. *Подымов В. В.* О сложности проверки эквивалентности линейных унарных рекурсивных программ над уравновешенными полугруппами // Ломоносовские чтения-2024. Секция Вычислительной математики и кибернетики. М.: МАКС Пресс, 2024. С. 53–55.
38. *Котов В. Е., Сабельфельд В. К.* Теория схем программ. М.: Наука, 1991.

REFERENCES

1. *Rice H. G.* Classes of recursively enumerable sets and their decision problems. Trans. AMS, 1953, vol. 74, pp. 358–366.
2. *Kleene S. C.* Introduction to Metamathematics. N.Y., Toronto, Nostrand Company, 1952.
3. *Glushkov V. M. and Letichevskiy A. A.* Teoriya diskretnykh preobrazovateley [Theory of discrete processors]. Izbrannye Voprosy Algebry i Logiki, Novosibirsk, Nauka, 1973, pp. 5–39. (in Russian)
4. *Zakharov V. A.* Bystrye algoritmy razresheniya ekvivalentnosti operatornykh programm na uravnoveshennykh shkalakh [Fast equivalence-checking algorithms for operator programs over balanced frames]. Matematicheskie Voprosy Kibernetiki, 1998, iss. 7, pp. 303–324. (in Russian)
5. *Letichevskiy A. A.* Funktsional'naya ekvivalentnost' diskretnykh preobrazovateley III [Functional equivalence of discrete processors III]. Kibernetika, 1972, no. 1, pp. 1–4. (in Russian)

6. *Letichevskiy A. A. and Smikun L. B.* Ob odnom klasse grupp s razreshimoy problemoy ekvivalentnosti [On a class of groups with solvable problem of automata equivalence]. Doklady AN SSSR, 1976, vol. 227, no. 1, pp. 36–38. (in Russian)
7. *Zakharov V. A. and Podymov V. V.* Primenenie algoritmov proverki ekvivalentnosti dlya optimizatsii programm [On the application of equivalence checking algorithms for program minimization]. Proc. ISP RAS, 2015, vol. 27, iss. 4, pp. 145–174. (in Russian)
8. *Zakharov V. A.* An efficient and unified approach to the decidability of equivalence of propositional programs. LNCS, 1998, vol. 1443, pp. 247–258.
9. *Zakharov V. A.* Bystrye algoritmy razresheniya ekvivalentnosti propozitsional'nykh operatornykh programm na uporyadochennykh polugruppovykh shkalakh [Fast equivalence-checking algorithms for operator programs over ordered frames]. Vestnik Moskovskogo Universiteta. Ser. 15. Vychislitel'naya matematika i kibernetika, 1999, no. 3, pp. 29–35. (in Russian)
10. *Zakharov V. A.* O probleme ekvivalentnosti operatornykh programm na uravnoveshennykh odnorodnykh obratimykh shkalakh [On the equivalence problem for operator programs over balanced homogenous invertible frames]. Matematicheskie Voprosy Kibernetiki, 2001, iss. 10, pp. 155–166. (in Russian)
11. *Zakharov V. A.* The equivalence problem for computational models: Decidable and undecidable cases. LNCS, 2001, vol. 2055, pp. 133–152.
12. *Zakharyaschev I. M. and Zakharov V. A.* On the equivalence-checking problem for polysemantic models of sequential programs. Proc. ISP RAS, 2004, vol. 6, pp. 179–198.
13. *Podlovchenko R. I., Rusakov D. M., and Zakharov V. A.* The equivalence problem for programs with mode switching is PSPACE-complete. Proc. ISP RAS, 2006, vol. 11, pp. 109–128.
14. *Shcherbina V. L. and Zakharov V. A.* Effektivnye algoritmy proverki ekvivalentnosti programm v modelyakh, svyazannykh s obrabotkoy preryvaniy [Efficient equivalence-checking algorithms for program models related to interrupt handling]. Vestnik Moskovskogo Universiteta. Ser. 15. Vychislitel'naya matematika i kibernetika, 2008, no. 2, pp. 33–41. (in Russian)
15. *Podlovchenko R. I., Kuzyurin N. N., Shcherbina V. L., and Zakharov V. A.* Using algebraic models of programs for detecting metamorphic malwares. J. Math. Sci., 2011, vol. 172, no. 5, pp. 740–750.
16. *Zakharov V. A.* Program equivalence checking by two-tape automata. Cybernetics and Systems Analysis, 2010, vol. 46, no. 4, pp. 554–562.
17. *Podymov V. V. and Zakharov V. A.* O dvukhlentochnykh mashinakh, opisyvayushchikh polugruppy s sokrashcheniem [On two-tape machines used for description of cancellative semigroups]. Proc. XVI Intern. Conf. "Problemy Teoreticheskoy Kibernetiki" (Nizhny Novgorod, June 20–25, 2011), Nizhniy Novgorod, UNN Publ., 2011, pp. 372–375. (in Russian)
18. *Zakharov V. A.* Modeli i algoritmy v zadache proverki ekvivalentnosti programm [Models and algorithms related to program equivalence checking]. Proc. XI Intern. Conf. "Diskretnaya Matematika i ee Prilozheniya" (Moscow, MSU, June 18–23, 2012), Moscow, MSU Faculty of Mechanics and Mathematics Publ., 2012, pp. 53–62. (in Russian)
19. *Podlovchenko R. I. and Zakharov V. A.* O dvukh metodakh raspoznavaniya ekvivalentnosti v algebraicheskikh modelyakh programm [On two equivalence-checking techniques for algebraic program models]. Intellektual'nye Sistemy, 2013, vol. 17, no. 1–4, pp. 366–370. (in Russian)
20. *Podymov V. V. and Zakharov V. A.* Polinomial'nyy algoritm proverki ekvivalentnosti v modeli programm s perestanovochnymi i podavlyaemymi operatorami [A polynomial algorithm for checking the equivalence in models of programs with commutation and vast operators]. Proc. ISP RAS, 2014, vol. 26, iss. 3, pp. 145–166. (in Russian)
21. *Podymov V. V.* Uluchshenie algoritmov proverki ekvivalentnosti operatornykh programm pri pomoshchi analiza vesov vershin [Improving equivalence-checking algorithms for operator

programs with node weight analysis]. Lomonosovskie Chteniya-2021, Moscow, MSU Publ., 2021, pp. 124–125. (in Russian)

22. *Karpov Yu. G.* Teoriya avtomatov [Automata Theory]. Saint Petersburg, Piter, 2003. (in Russian)
23. *Zakharov V. A. and Zakharyaschev I. M.* On the equivalence checking problem for a model of programs related with muti-tape automata. LNCS, 2005, vol. 3317, pp. 293–305.
24. *Pentus A. E. and Pentus M. R.* Teoriya formal'nykh yazykov [Formal Language Theory]. Moscow, MSU Faculty of Mechanics and Mathematics Publ., 2004. (in Russian)
25. *Gavrilov G. P. and Sapozhenko A. A.* Zadachi i uprazhneniya po diskretnoy matematike [Discrete Mathematics Problems and Exercises]. Moscow, Fizmatlit Publ., 2005. (in Russian)
26. *Zorich V. A.* Matematicheskiy analiz [Mathematical Analysis]. Part I. Moscow, MCCME Publ., 2002. (in Russian)
27. *Cormen T. H., Leiserson C. E., Rivest R. L., and Stein C.* Introduction to Algorithms. 3rd ed. Cambridge, Massachusetts, London, England, MIT Press, 2009.
28. *Lalleman G.* Semigroups and Combinatorial Applications. N.Y., Chichester, Brisbane, Toronto, John Wiley & Sons, 1979.
29. *Clifford A. H. and Preston G. B.* The Algebraic Theory of Semigroups, vol. I. Providence, Rhode Island, AMS, 1964.
30. *Stroustrup B.* A Tour of C++. 2nd ed. Boston, MA, USA, Addison-Wesley, 2018.
31. *Zakharov V. A.* Ob effektivnoy razreshimosti problemy ekvivalentnosti lineynykh unarnykh rekursivnykh programm [On efficient decidability of an equivalence problem for linear monadic recursive programs]. Matematicheskie Vopposy Kibepnetiki, 1999, vol. 8, pp. 255–273. (in Russian)
32. *Zakharov V. A.* Ob ekvivalentnosti potokovykh programm [On equivalence of streaming programs]. Proc. XI Intern. Conf. "Diskretnaya Matematika i ee Prilozheniya" (Moscow, June 18–23, 2012), Moscow, MSU Faculty of Mechanics and Mathematics Publ., 2012, pp. 119–122. (in Russian)
33. *Podymov V. V.* Algoritm proverki ekvivalentnosti lineynykh unarnykh rekursivnykh programm na uporyadochennykh polugruppovykh shkalakh [An equivalence-checking algorithm for linear monadic recursive programs over ordered semigroup frames]. Vestnik Moskovskogo universiteta. Ser. 15. Vychislitel'naya Matematika i Kibernetika, 2012, no. 4, pp. 37–43. (in Russian)
34. *Zakharov V. A.* Ob ekvivalentnosti ogranichenno nedeterminirovannykh avtomatov-preobrazovateley nad polugruppami [On equivalence of finite-valued transducers over semigroups]. Proc. XVII Intern. Conf. "Problemy Teoreticheskoy Kibernetiki" (Kazan, June 16–20, 2014), Kazan, Otechestvo Publ., 2014, pp. 100–102. (in Russian)
35. *Zakharov V. A.* Modelirovanie i analiz povedeniya posledovatel'nykh reagiruyushchikh programm [Modeling and analysis of the behavior of successive reactive programs]. Proc. ISP RAS, 2015, vol. 27, no. 2, pp. 221–250. (in Russian)
36. *Podymov V. V.* Efficient equivalence-checking algorithms for procedural programs in progressive semigroup gateway models. Moscow Univ. Comput. Math. Cybern., 2019, vol. 43, no. 4, pp. 181–187.
37. *Podymov V. V.* O slozhnosti proverki ekvivalentnosti lineynykh unarnykh rekursivnykh programm nad uravnoveshennymi polugruppami [On complexity of equivalence checking for linear monadic recursive programs over balanced semigroups]. Lomonosovskie Chteniya-2024, Moscow, MAKS Press, 2024, pp. 53–55. (in Russian)
38. *Kotov V. E. and Sabelfeld V. K.* Teoriya skhem programm [Theory of Program Schemes]. Moscow, Nauka, 1991. (in Russian)