УДК 510.676, 519.7

О НЕВОЗМОЖНОСТИ ЗАДАНИЯ ЛИНЕЙНОГО ПОРЯДКА ПРИ ПОМОЩИ НЕДЕТЕРМИНИРОВАННЫХ ПРОГРАММ В ГРУППОИДАХ СПЕЦИАЛЬНОГО ВИДА

Попов И.В.
Кафедра информатики

Поступила в редакцию 07.04.2008, после переработки 24.09.2008.

В работе рассматривается класс группоидов, предложенный в статье М.А. Тайцлина «Пример полиномиального запроса не распознаваемого в недетерминированной логарифмической памяти». Изучается вопрос выразимости линейного порядка на элементах группоидов из данного класса при помощи недетерминированных программ. Показывается, что вопрос о невозможности задания линейного порядка может быть сведен к вопросу об отсутствии решения системы уравнений в заданной четырехэлементной квазигруппе.

In this paper we consider a special class of groupoids, which was presented in the article by M.A.Taitslin «Пример полиномиального запроса не распознаваемого в недетерминированной логарифмической памяти». In this paper we study a question of unexpressibility of linear order on the elements of such groupoid using non-determined programs. It is shown, that the question of linear order unexpressibility can be reduced to the question of unsolvability of some equations system in four-element quasi-group.

Ключевые слова: недетерминированные программы, линейный порядок, линейные программы, квазигруппы.
Keywords: non-determined programs, linear order, linear programs, quasi-groups.

1. Введение

Вопрос о том, совпадают ли классы сложности **NLogSpace** и **PTime**, был сформулирован в работах Кука [3] и [4]. Одним из наиболее значимых результатов в данной области было доказательство теоремы Кука [4]. В теореме доказывается, что полиномиальный алгоритм может быть реализован в ограниченной памяти с использованием неограниченного стека.

Позднее, в работе М.А.Тайцлина [2] была предложена проблема, принадлежащая классу **PTime**, и была высказана гипотеза о том, что эта проблема не принадлежит классу **NLogSpace**. Доказать эту гипотезу означает доказать, что классы **NLogSpace** и **PTime** различны.

В данной работе изучается вопрос о выразимости линейного порядка на элементах группоидов, предложенных в [2], при помощи недетерминированных программ. Мы рассматриваем недетерминированные программы специального вида

(без использования оператора следования) и показываем, что вопрос о невыразимости порядка на элементах группоида может быть сведен к вопросу об отсутствии решения системы уравнений в заданной четырехэлементной квазигруппе.

2. Определения

2.1 Квазигруппы Q_{4^i}

Рассмотрим четырехэлементную квазигруппу Q_4, описанную в [2]. Q_4 содержит элементы $\{1,2,3,4\}$, операция $\circ$ на которых определена в соответствии с таблицей:

	1	2	3	4
1	1	3	4	2
2	4	2	1	3
3	2	4	3	1
4	3	1	2	4

Строки таблицы соответствуют первому операнду, столбцы – второму. Например, $(2 \circ 1) = 4$, но $(1 \circ 2) = 3$.

Далее мы построим квазигруппы Q_{4^i} для каждого натурального $i > 1$. Процедура построения также заимствована из [2]. Пусть Q_4 есть Q_{4^1}. Допустим, что для некоторого i квазигруппа Q_{4^i} с операцией $\circ_i$ уже построена и содержит 4^i элементов. Для построения $Q_{4^{i+1}}$ используются четыре копии квазигруппы Q_{4^i} с попарно различными элементами. Пусть копии квазигруппы Q_{4^i} содержат элементы a_e^g, где $e \in Q_{4^i}$, а g – это номер копии, $g \in \{1,2,3,4\}$.

Квазигруппа $Q_{4^{i+1}}$ содержит все элементы a_e^g, операция на которых определяется так:

$$(a_{e_1}^{g_1} \circ_{i+1} a_{e_2}^{g_2}) = a_{(e_1 \circ_i e_2)}^{(g_1 \circ g_2)}.$$

Очевидно, что $Q_{4^{i+1}}$ также является квазигруппой, содержащей 4^{i+1} элементов. Используя данную процедуру, легко строить сколь угодно большие квазигруппы на основе заданной Q_4.

Мы часто будем опускать индекс i при написании операции $\circ_i$ в квазигруппе Q_{4^i} и писать просто $\circ$, если ясно, о каком i идёт речь.

Перестановкой квазигруппы Q_{4^i}, как обычно, называется биекция $\tau : Q_{4^i} \to Q_{4^i}$. Перестановка называется тривиальной, если для нее выполнено $\tau[a] = a$ для любого $a \in Q_{4^i}$. Мы будем рассматривать операцию $\star$ на перестановках Q_{4^i}. Будем говорить, что $(\tau_1 \star \tau_2) = \tau_3$ тогда и только тогда, когда

$$(\tau_1[a] \circ_i \tau_2[b]) = \tau_3[(a \circ_i b)], \text{ для всех } a, b \in Q_{4^i}.$$

Заметим, что операция $\star$ не является корректно определенной операцией на всех перестановках Q_{4_i}. Например, для перестановок $\tau_1 = \{2,1,4,3\}$ и $\tau_2 = \{3,2,1,4\}$ не существует такой перестановки τ_3, что выполнено $(\tau_1 \star \tau_2) = \tau_3$. Однако в данной работе рассматривается не все множество перестановок Q_{4_i}, а лишь небольшой класс, описанный в разделе 3. Внутри данного класса операция $\star$ корректна, что также показывается в разделе 3.

2.2 Постановка задачи

Проблема для разделения классов **PTime** и **NLogSpace** была сформулирована в [2]. Здесь приводится ее более короткое эквивалентное описание.

Мы будем рассматривать группоид $G_{I,J}$, параметризованный двумя положительными натуральными величинами I и J. Группоид строится следующим образом.

Рассмотрим полное бинарное дерево T_J высотой J. Известно, что T_J имеет $2^J - 1$ элементов и 2^{J-1} листьев. Рассмотрим идемпотентную квазигруппу Q_{4^I}, содержащую 4^I элементов. Элементами группоида $G_{I,J}$ являются пары элементов (q,t), где $q \in Q_{4^I}$, а $t \in T_J$. Таким образом, группоид содержит $4^I(2^J-1)$ элементов.

Операция f на элементах группоида $G_{I,J}$ определяется следующим образом.

- Если элементы t_l, t_r являются левым и правым сыновьями элемента t в дереве T_J, то для любых $q_1, q_2 \in Q_{4^I}$ выполнено:
$$f((q_1,t_l),(q_2,t_r)) = ((q_1 \circ q_2),t).$$
- Если элемент t_1 является i-ым листом и t_2 является $(i+1)$-ым листом дерева T_J (все листы нумеруются подряд слева направо), то для любого $q \in Q_{4^I}$ выполнено:
$$f((q,t_1),(q,t_1)) = (q,t_2).$$
- Во всех остальных случаях, для любых $q_1, q_2 \in Q_{4^I}$ и $t_1, t_2 \in T_J$, операция определяется так:
$$f((q_1,t_1),(q_2,t_2)) = (q_1,t_1).$$

Заметим, что в группоиде $G_{I,J}$ один и только один элемент вида (q,t), где t является корнем дерева T_J, порождается фиксированным элементом вида (q_1,t_1), где t_1 является самым левым листом дерева T_J. Этот факт легко доказывается индукцией по высоте дерева T_J.

Пусть $N_{I,J}$ – начальный отрезок натуральных чисел длины $4^I(2^J-1)$. На $N_{I,J}$ естественным образом определена операция следования $'$, а также две константы: 0 – начало отрезка, $LAST$ – конец отрезка.

Построим произвольное отображение ν группоида $G_{I,J}$ во множество $N_{I,J}$ и рассмотрим алгебраическую систему

$$A = (N_{I,J}, 0, LAST, ', f_\nu, a, b).$$

Задача о разделении классов **PTime** и **NLogSpace** формулируется так. Возможно ли с помощью **NLogSpace**-программы Π выделить те и только те системы, где значение константы b может быть получено с помощью функции f_ν из значения константы a.

В данной работе рассматривается более узкий класс недетерминированных программ, меньший чем **NLogSpace**. Рассматриваемые программы не используют операцию следования $'$. Как и прежде, мы рассматриваем некоторое отображение ν группоида $G_{I,J}$ во множество $N_{I,J}$ и рассматриваем алгебраические системы вида

$$A = (N_{I,J}, 0, LAST, f_\nu, a, b).$$

Факт того, что программа П выполняется на алгебраической системе A мы будем обозначать $A \vdash$ П. Мы будем говорить, что недетерминированная программа П выражает порядок на множестве рассматриваемых алгебраических систем, если для каждого отображения ν множество

$$\{(a,b)|(N_{I,J},0,LAST,f_\nu,a,b) \vdash \Pi\}$$

является отношением линейного порядка на $N_{I,J}$.

Носитель алгебраической системы A мы будем обозначать $U(A)$.

В работе показывается, что доказательство невыразимости порядка или принадлежности заданного элемента подполугруппе, порождённой другим заданным элементом, недетерминированными программами может быть сведено к доказательству существования решения системы уравнений в фиксированной четырехэлементной квазигруппе.

2.3 Программы

Мы рассматриваем *недетерминированные программы* с конечным множеством переменных V и конечным множеством констант C. Мы также всегда предполагаем, что множество C содержит константу 0.

Простая программа является одним из присваиваний следующего вида:

- $\mathbf{x} \leftarrow \mathbf{c}$, для любого $c \in C$ и $x \in V$;
- $\mathbf{x} \leftarrow \mathbf{y}$, для любых $x, y \in V$;
- $\mathbf{x} \leftarrow \mathbf{f}(\mathbf{y}, \mathbf{z})$, для любых $x, y, z \in V$;
- $\mathbf{RAND}(\mathbf{x})$, для любого $x \in V$.

$\mathbf{RAND}(\mathbf{x})$ – это недетерминированный оператор, который присваивает переменной x произвольное значение.

Более сложные программы формируются по правилам:

- $\mathbf{П_1}; \mathbf{П_2}$ - композиция программ, Π_1, Π_2 - программы;
- $\mathbf{while}(\mathbf{T})\{\mathbf{П}\}$ - оператор цикла, П - программа, T - тест;
- $\mathbf{if}(\mathbf{T})\{\mathbf{П_1}\}\mathbf{else}\{\mathbf{П_2}\}$ - условный оператор, Π_1, Π_2 - программы, T - тест.

В качестве тестов используются операторы сравнения двух переменных:

- $\mathbf{x} = \mathbf{y}$, для любых $x, y \in V$; тест истинен, когда значение переменной x совпадает со значением переменной y;
- $\mathbf{x} \neq \mathbf{y}$, для любых $x, y \in V$.

Семантика работы программы стандартная. В начальный момент времени значение каждой переменной есть 0.

Линейной программой Λ называется конечная последовательность операторов, каждый из которых является либо присваиванием, либо тестом. Присваивания могут иметь вид:

- $\mathbf{x} \leftarrow \mathbf{c}$, для любого $c \in C$ и $x \in V$;
- $\mathbf{x} \leftarrow \mathbf{y}$, для любых $x, y \in V$;
- $\mathbf{x} \leftarrow \mathbf{f}(\mathbf{y}, \mathbf{z})$, для любых $x, y, z \in V$.

Тесты могут быть двух видов:

- $(\mathbf{x} = \mathbf{y})$, для любых $x, y \in V$;
- $(\mathbf{x} \neq \mathbf{y})$, для любых $x, y \in V$.

Выполнение линейной программы происходит последовательно оператор за оператором. В начальный момент времени значения всех переменных равны 0. Если очередной оператор – это присваивание, то оно выполняется. Из-за этого могут измениться значения некоторых переменных. Если очередной оператор – это тест, то происходит проверка условия, описанного в тесте. Если условие выполняется, то выполнение программы продолжается, в противном случае выполнение программы останавливается на данном тесте.

Будем говорить, что линейная программа выполняется на некоторой системе $A = (U, f, C)$, если выполняются все ее операторы. Если же выполнение программы заканчивается на каком-либо тесте из-за того, что условие теста ложно, то будем говорить, что данная линейная программа не выполняется на данной системе.

Параметризованной линейной программой Θ называется пара (Λ, P), где P – это конечное множество параметров, а Λ – это линейная программа с множеством констант $C \cup P$.

Выполнение Θ зависит от значений, присвоенных используемым параметрам P. Если значения параметров описываются функцией φ, то процесс выполнения аналогичен выполнению обычной линейной программы, за тем лишь исключением, что при выполнении присваивания переменной некоторого параметра $p \in P$ переменной присваивается значение $\varphi(p)$. Говоря о том, что Θ выполняется в некоторой алгебраической системе A функцией φ, мы будем писать $A \vdash (\Theta, \varphi)$.

3. Идемпотентная квазигруппа перестановок Q_{4^i}

Рассмотрим четыре перестановки $\tau_N, \tau_C, \tau_L, \tau_R$ квазигруппы Q_4. Перестановки определяются так:

- τ_N – перестановка $\{1, 2, 3, 4\}$, тривиальная перестановка Q_4;
- τ_C – перестановка $\{2, 1, 4, 3\}$;
- τ_L – перестановка $\{3, 4, 1, 2\}$;
- τ_R – перестановка $\{4, 3, 2, 1\}$.

Легко проверить, что данное множество перестановок образует четырехэлементную идемпотентную квазигруппу R_4 с операцией $\star$, определенной в разделе 2.1. Значения операции $\star$ в R_4 образуют следующую таблицу:

	τ_N	τ_C	τ_L	τ_R
τ_N	τ_N	τ_L	τ_R	τ_C
τ_C	τ_R	τ_C	τ_N	τ_L
τ_L	τ_C	τ_R	τ_L	τ_N
τ_R	τ_L	τ_N	τ_C	τ_R

(1)

Покажем, что идемпотентную квазигруппу R_4^i перестановок квазигруппы Q_{4^i} можно построить для любого положительного $i > 1$.

Теорема 1. *Для каждого $i > 1$ можно построить идемпотентную квазигруппу R_4^i перестановок Q_{4^i} такую, что*

1. *R_4^i содержит четыре элемента $\tau_N^i, \tau_C^i, \tau_L^i, \tau_R^i$ с определенной на них операцией $\star$, задаваемой таблицей (1);*
2. *перестановка τ_C^i содержит цикл длины два.*

Доказательство. Индукцией по i.

Для $i = 1$ в качестве R_4^1 возьмем построенную выше R_4. Циклом длины 2 для τ_C^1 может быть, например, цикл $(1, 2)$, так как $\tau_C^1[1] = 2$ и $\tau_C^1[2] = 1$.

Пусть квазигруппа R_4^i уже построена. Тогда элементы R_4^{i+1} определяются по следующей формуле:

$$\tau_X^{i+1}[a_e^g] = a^g_{\tau_X^i[e]}, \quad X \in \{N, C, L, R\}, \quad a_e^g \in Q_{4^{i+1}}.$$

Свойство 1. Легко заметить, что операция $\star$ при переходе от i к $i+1$ сохраняется. Действительно, пусть

$$(\tau_X^i \star \tau_Y^i) = \tau_Z^i, \text{ для некоторых } X, Y, Z \in \{N, C, L, R\}.$$

Тогда для любых $a_{e_1}^{g_1}, a_{e_2}^{g_2} \in Q_{4^{i+1}}$

$$(\tau_X^{i+1}[a_{e_1}^{g_1}] \circ \tau_Y^{i+1}[a_{e_2}^{g_2}]) = (a^{g_1}_{\tau_X^i[e_1]} \circ a^{g_2}_{\tau_Y^i[e_2]}) = a^{(g_1 \circ g_2)}_{(\tau_X^i[e_1] \circ \tau_Y^i[e_2])} =$$
$$= a^{(g_1 \circ g_2)}_{\tau_Z^i[(e_1 \circ e_2)]} = \tau_Z^{i+1}[a^{(g_1 \circ g_2)}_{(e_1 \circ e_2)}] = \tau_Z^{i+1}[(a_{e_1}^{g_1} \circ a_{e_2}^{g_2})].$$

Следовательно,

$$(\tau_X^{i+1} \star \tau_Y^{i+1}) = \tau_Z^{i+1}.$$

Свойство 2. Пусть для τ_C^i элементы (e_1, e_2) квазигруппы Q_{4^i} образуют цикл длины 2. Тогда очевидно, что $(a_{e_1}^g, a_{e_2}^g)$ будет циклом длины 2 для τ_C^{i+1} в $Q_{4^{i+1}}$ для любого $g \in \{1, 2, 3, 4\}$. Действительно,

$$\tau_C^{i+1}[a_{e_1}^g] = a^g_{\tau_C^i[e_1]} = a_{e_2}^g;$$

$$\tau_C^{i+1}[a_{e_2}^g] = a^g_{\tau_C^i[e_2]} = a_{e_1}^g.$$

□

4. Построение параметризированных программ

Допустим, что недетерминированная программа Π со множеством переменных V и множеством констант C выполняется на некоторой алгебраической системе A. Тогда по данному процессу выполнения легко построить параметризованную линейную программу $\Theta = (\Lambda, P)$ и отображение φ такие, что $A \vdash (\Theta, \varphi)$.

Построение Θ и φ происходит по следующему алгоритму. В начале построения последовательность Λ пуста и множество параметров P пусто. Считаем также, что множество переменных линейной программы Λ есть V, множество констант – $C \cup P$. Построение происходит на каждом шаге выполнения программы Π по следующим правилам:

- если на текущем шаге в программе Π выполняется детерминированный оператор присваивания, то в Λ заносится данный оператор;
- если на текущем шаге в программе Π выполняется оператор $\mathbf{RAND}(\mathbf{x})$, присваивающий некоторое значение v переменной x, то выбирается некоторый уникальный параметр p (не содержащийся в уже построенном множестве параметров P) и добавляется в P; отображение φ доопределяется на параметре p значением v, иными словами, $\varphi(p) = v$; в Λ заносится оператор $\mathbf{x} \leftarrow \mathbf{p}$;
- если на текущем шаге в программе Π происходит проверка некоторого теста (для циклов и ветвлений), то в случае удачной проверки (тест истинен), он добавляется в Λ, в случае неудачной проверки (тест оказался ложным), в Λ заносится тест, противоположный данному.

Очевидно, что построенная таким образом $\Theta = (\Lambda, P)$ будет выполняться отображением φ на A.

Верно также и следующее утверждение.

Теорема 2. *Пусть недетерминированная программа* Π *выполняется на алгебраической системе* A. *Пусть параметризированная линейная программа* $\Theta = (\Lambda, P)$ *и* $\varphi : P \to U(A)$ *построены по ходу этого выполнения процедурой, описанной выше. Тогда, если для алгебраической системы* A' *существует такое отображение* $\varphi' : P \to U(A')$, *что* $A \vdash (\Theta, \varphi')$, *то программа* Π *тоже выполняется на алгебраической системе* A'.

Доказательство. Так как Θ выполняется на A', то и в программе Π будут выполняться и не выполняться точно те же тесты и в тот же момент времени, что и на A, таким образом, программа Π сделает столько же шагов, сколько она сделала на A. Значения, угадываемые оператором **RAND**, возможно, будут другими, но те тесты, что были истинными, останутся истинными, те что были ложными, останутся ложными. Это гарантирует выполнение Θ на A'.

Следовательно, $A' \vdash \Pi$. □

5. Основные результаты

Мы рассматриваем недетерминированные программы сигнатуры $\Omega = (f^{(2)}, 0, LAST, a, b)$. Мы рассматриваем произвольные биекции $\nu : G_{I,J} \to N_{I,J}$ и соответствующие им алгебраические системы сигнатуры Ω:

$$A = (N_{I,J}, f_\nu, 0, LAST, a, b),$$

где f_ν образ функции f группоида, соответствующий ν, $a, b \in N_{I,J}$ – произвольные константы. Мы доказываем следующую теорему.

Теорема 3. *Для каждой недетерминированной программы* Π *можно построить такую систему уравнений* U *для квазигрупп* R_4^I*, что из существования решения данной системы в одной из этих квазигрупп следует, что* Π *не выражает порядок на множестве алгебраических систем* A*.*

Теорема 4. *Для каждой недетерминированной программы* Π *можно построить такую систему уравнений* U *для квазигрупп* R_4^I*, что из существования решения данной системы в одной из этих квазигрупп следует, что* Π *не выражает на множестве алгебраических систем* A*, что* b *принадлежит подгруппоиду, порождённому* 0*.*

Доказательство. Для квазигруппы Q_{4^I} построим соответствующую ей идемпотентную квазигруппу перестановок R_4^I так, как это описано в разделе 3. Пусть элементы $q_a, q_b \in Q_{4^I}$ задают цикл длиной 2 в $\tau_C^I \in R_4^I$.

Построим отображение $\nu : G_{I,J} \to N_{I,J}$ следующим образом. Выберем такой элемент $(q_a, t) \in G_{I,J}$, что t является самым левым листом в дереве T_J. Определим $\nu((q_a, t)) = 0$. Затем построим элемент $(q_a, t') = f((q_a, t), (q_a, t))$, здесь t' – это следующий (в порядке слева направо) лист дерева T_J. Определим $\nu((q_a, t')) = LAST$. Все остальные элементы $G_{I,J}$ определим произвольным образом. Такое определение ν позволяет сделать так, чтобы константы $0, LAST$ давали возможность программе попасть только на листья дерева T_J.

Зафиксируем построенное отображение ν и рассмотрим множество A из $|N_{I,J}|^2$ алгебраических систем:

$$A = \{(N_{I,J}, 0, LAST, f_\nu, a, b) | a, b \in N_{I,J}\}.$$

Рассмотрим два различных элемента группоида $G_{I,J}$: $g_a = (q_a, r)$ и $g_b = (q_b, r)$, здесь r – корень дерева T_J.

Эти элемента различны и поэтому, если программа Π выражает порядок, то она должна выполняться ровно на одной из алгебраических систем

$$A_{ab} = (N_{I,J}, 0, LAST, f_\nu, g_a, g_b) \text{ или } A_{ba} = (N_{I,J}, 0, LAST, f_\nu, g_b, g_a).$$

Допустим, она выполняется на системе A_{ab}.

По процессу выполнения программы Π на системе A_{ab} построим параметризированную линейную программу $\Theta = (\Lambda, P)$ и отображение $\varphi : P \to N_{I,J}$, выполняющее ее на A_{ab}. Пусть линейная программа Λ состоит из s операторов $(\Lambda = \lambda_1, \lambda_2, \ldots, \lambda_s)$. Она так же, как и исходная программа Π, использует n переменных.

Для продолжения доказательства нам потребуется несколько определений.

Определение 1. *Временным интервалом или просто интервалом от i до k ($i \leqslant k$) называется промежуток во времени выполнения программы Λ, в который выполняются операторы $\lambda_i, \ldots, \lambda_k$. Как правило, каждый временной интервал ассоциирован с некоторой вершиной дерева T_J. Интервал, ассоциированный с вершиной $t \in T_J$ от i до k мы будем обозначать $t[i,k]$.*

Определение 2. *Мы будем говорить, что момент времени j принадлежит интервалу $t[i,k]$, если $i \leqslant j \leqslant k$. Данный факт обозначается так: $j \in t[i,k]$.*

Определение 3. *Мы будем говорить, что интервалы $t[i,k]$ и $t'[i',k']$ пересекаются, если существует такой момент времени j, что $j \in t[i,k]$ и $j \in t'[i',k']$.*

По процессу выполнения $A_{ab} \vdash (\Theta, \varphi)$ построим множество интервалов

$$\Upsilon = \{t[i,k] | t \in T_J, 0 \leqslant i \leqslant k \leqslant s\},$$

удовлетворяющее следующим свойствам:

a) из того, что в момент j выполнения программы Λ на системе A_{ab} значение некоторой переменной x равно значению элемента (q,t) группоида $G_{I,J}$, следует, что существует интервал $t[i,k] \in \Upsilon$, такой, что $j \in t[i,k]$;

b) для каждого $t \in T_J$ и для любых двух интервалов $t[i_1,k_1], t[i_2,k_2] \in \Upsilon$ выполнено либо $i_1 \leqslant k_1 < i_2 \leqslant k_2$, либо $i_2 \leqslant k_2 < i_1 \leqslant k_1$.

Построение множества Υ происходит в два этапа.

Первый этап – заполнение. Для каждого момента времени $0 \leqslant i \leqslant s$ и для каждой переменной $x \in V$ рассмотрим значение, которое принимала данная переменная при выполнении $A_{ab} \vdash (\Theta, \varphi)$. Допустим, это значение равно некоторому элементу (q,t) группоида $G_{I,J}$. Тогда добавим интервал $t[i,i]$ во множество Υ. Добавляем такие интервалы для каждого момента времени i и для каждой переменной x.

Второй этап – нормализация. Повторяем следующие преобразования до тех пор пока хотя бы одно из них может быть применено ко множеству Υ.

1. Если для некоторого $t \in T_J$ интервал $t[i_1,k_1] \in \Upsilon$ пересекается с интервалом $t[i_2,k_2] \in \Upsilon$, то оба эти интервала удаляются из множества Υ, а их объединение – интервал $t[min(i_1,i_2), max(k_1,k_2)]$ добавляется в Υ.

2. Если для некоторого $t \in T_J$ существуют интервалы $t[i_1,k_1], t[i_2,k_2] \in \Upsilon$, такие, что $k_1 + 1 = i_2$, то оба эти интервала удаляются из множества Υ, а вместо них добавляется интервал $t[i_1,k_2]$.

Легко заметить, что каждое преобразование множества Υ уменьшает количество интервалов в нем на 1, а так как общее число интервалов конечно, то эта процедура обязательно остановится и в момент остановки к полученному множеству Υ нельзя будет применить ни одно из преобразований.

Таким образом, после двух этапов построения мы получим множество Υ, удовлетворяющие необходимым нам свойствам. Действительно, выполнение свойства a) гарантирует первый этап, когда каждое значение каждой переменной на протяжении всего времени работы программы заносится в Υ. Второй же этап только

укрупняет интервалы, не нарушая свойства a), и за счет укрупнения достигает выполнения свойства b).

В построенном множестве Υ каждый интервал $t[i,k]$ говорит о том, что начиная с момента i и до момента k (при выполнении операторов $\lambda_i, \ldots, \lambda_k$), существует хотя бы одна переменная, значение которой равно значению элемента (q,t) для некоторого $q \in Q_{4^I}$. При этом в моменты времени $i-1$ и $k+1$ ни одной переменной с таким значением нет.

Далее нам потребуется еще одно определение.

Определение 4. *Мы будем говорить, что три вершины t_1, t_2, t_3 дерева T_J являются соседними, если вершина t_1 является левым сыном вершины t_2 в дереве T_J, а вершина t_3 правым. Данный факт мы будем обозначать так (t_1, t_2, t_3), при этом важен порядок вершин в записи. В центре записывается родитель, слева – левый сын, справа – правый.*

Построение системы уравнений U по множеству интервалов Υ происходит следующим образом. Каждому интервалу $t[i,k]$ во множестве Υ сопоставляется переменная $v_{t[i,k]}$. Уравнения имеют следующий вид:

1. Для даждой переменной $v_{r[i,k]}$, где r – корень дерева T_J, в U добавляется уравнение
$$v_{r[i,k]} = \tau_C^I.$$

2. Для любых соседних вершин (t_l, t_c, t_r) дерева T_J и момента времени j, если существуют интервалы $t_l[i_l, k_l], t_r[i_r, k_r]$ и $t_c[i_c, k_c]$, содержащие момент времени j, то в U добавляется уравнение
$$(v_{t_l[i_l,k_l]} \star v_{t_r[i_r,k_r]}) = v_{t_c[i_c,k_c]}.$$

3. Для каждой переменной $v_{t[i,k]}$, где t – лист дерева T_J, в U добавляется уравнение
$$v_{t[i,k]} = \tau_N^I.$$

Решение системы U приписывает каждой переменной $v_{t[i,k]}$ некоторое значение из R_4^I, поэтому можно считать, что решение задает отображение $\rho : \Upsilon \to R_4^I$. Имея данное отображение, легко показать, что программа Π выполняется на системе A_{ba} и, следовательно, не выражает порядок и не выражает принадлежность заданного элемента подполугруппе, порождённой другим заданным элементом.

Для этого, по отображениям ρ и φ мы построим такое новое отображение φ', что $A_{ba} \vdash (\Theta, \varphi')$, где $\Theta = (\Lambda, P)$. Для каждого параметра $p \in P$ найдем момент времени i_p такой, что оператор линейной программы Λ есть $\mathbf{x} \leftarrow \mathbf{p}$. По построению линейной программы Λ такой оператор в ней только один. Пусть $\varphi(p)$ есть значение элемента (q,t) группоида $G_{I,J}$. Тогда, по построению множества интервалов Υ, существует интервал $t[i,k]$, содержащий i_p. Пусть $\rho(t[i,k]) = \tau$. Значение отображения φ' на p определяется как значение элемента $(\tau[q], t)$ в рассматриваемой нумерации группоида $G_{I,J}$.

Осталось показать, что $A_{ba} \vdash (\Theta, \varphi')$. Для доказательства данного факта, покажем что справедливо следующее утверждение. Рассмотрим процесс выполнения

программы (Θ, φ) на системе A_{ab}, если в некоторый момент i_0 в некоторой переменной x находится значение элемента (q, t), то, по построению множества интервалов, найдется интервал $i_0 \in t[i, k]$ и тогда в процессе выполнения программы (Θ, φ') на системе A_{ba} в данный момент времени переменная x будет содержать значение $(\rho(t[i, k])[q], t)$.

Доказываем данное утверждение индукцией по высоте узла t в дереве T_J.

1. Если t является листом, то, по построению U, любой интервал для t отображается ρ в τ_N^I, иными словами изменения значений в листах не происходит. Значение (q, t) в переменную x в программе (Θ, φ) может попасть тремя способами:

   (a) Присваивание x одной из констант $0, LAST$. В системах A_{ab} и A_{ba} значения этих констант совпадают, поэтому в этот момент в программе (Θ, φ') переменной x будет присвоено тоже значение, что удовлетворяет условию.

   (b) Присваиванием $\mathbf{x} \leftarrow \mathbf{f}(\mathbf{y}, \mathbf{y})$, где y – лист левее рассматриваемого листа. Так как значения y в программах (Θ, φ) и (Θ, φ') совпадают, то и значение x будет тоже совпадать.

   (c) Присваиванием x параметра p. По построению отображения φ' получим, что $\varphi'(x) = (\tau_N^I[q], t) = (q, t) = \varphi(p)$. Таким образом, значения x в обеих программах тоже совпадают.

2. Пусть t – внутренний узел дерева T_J. Предполагаем, что для низлежащих узов данное условие уже выполнено. Пусть t_l – левый сын узла t, а t_r – правый. Значение (q, t) в переменную x в программе (Θ, φ) может попасть двумя способами:

   (a) Присваиванием $\mathbf{x} \leftarrow \mathbf{f}(\mathbf{y}, \mathbf{z})$, где y, z – некоторые переменные. В данный момент в переменной y находится некоторое значение (q_l, t_l), а в переменной z – значение (q_r, t_r). Значения удовлетворяют условию $(q_l \circ q_r) = q$. По построению множества интервалов, в данный момент времени существуют интервалы $t_l[i_l, k_l]$ и $t_r[i_r, k_r]$. Пусть $\rho(t_l[i_l, k_l]) = \tau_{t_l}$ и $\rho(t_r[i_r, k_r]) = \tau_{t_r}$. По предположению индукции, в программе (Θ, φ') переменные y и z равны соответственно $(\tau_{t_l}[q_l], t_l)$ и $(\tau_{t_r}[q_r], t_r)$. Пусть $\rho(t[i, k]) = \tau_t$. Тогда, по построению системы U, должно выполняться соотношение $(\tau_{t_l}[q_l] \circ \tau_{t_r}[q_r]) = \tau_t[(q_l \circ q_r)] = \tau_t[q]$. Следовательно, в программе (Θ, φ') переменная x будет равна $f((\tau_{t_l}[q_l], t_l), (\tau_{t_l}[q_l], t_l)) = ((\tau_{t_l}[q_l] \circ \tau_{t_r}[q_r]), t) = (\tau_t[q], t)$.

   (b) Присваиванием x параметра p. В этом случае утверждение верно по построению отображения φ'.

3. Если же t является корнем, то дополнительно к предыдущему случаю, значение x может быть получено присваиванием значений констант a и b. По построению системы U, все интервалы корневого элемента дерева t отображаются в τ_C^I. При этом значение константы a в системе A_{ab} есть $g_a = (q_a, t)$, а в системе A_{ba} это $g_b = (q_b, t)$ при этом $q_b = \tau_C^I[q_a]$. Таким образом, если в A_{ab} переменной x присваивается значение $g_a = (q_a, t)$, то в системе A_{ba} будет

присвоено значение $g_b = (q_b, t) = (\tau_C^I[q_a], t)$, что удовлетворяет необходимому условию. Аналогичные рассуждения можно привести и для константы b.

Для доказательства $A_{ba} \vdash (\Theta, \varphi')$ осталось проверить, что все тесты в линейной программе Λ будут выполняться. Тесты могут быть двух видов.

Положительный тест. Тест вида $\mathbf{x} = \mathbf{y}$. Пусть в программе (Θ, φ) на A_{ab} значения переменных есть (q, t), тогда в программе (Θ, φ') значения x и y будут равны $(\tau[q], t)$ и тоже будут равны. Здесь τ есть перестановка, в которую был отображен интервал, соответствующий значениям x и y.

Отрицательный тест. Тест вида $\mathbf{x} \neq \mathbf{y}$. Пусть в пограмме (Θ, φ) на A_{ab} значения переменных есть $x = (q_1, t_1)$ и $y = (q_2, t_2)$ соответственно. Пусть τ_1 – перестановка интервала для значения x, а τ_2 – перестановка интервала для значения y. Тогда в программе (Θ, φ') значения x и y будут равны соответственно $(\tau_1[q_1], t_1)$ и $(\tau_2[q_2], t_2)$. Если $t_1 \neq t_2$, то очевидно, что тест будет истинен в (Θ, φ'), если же $t_1 = t_2$, то переменные x и y принадлежат одному интервалу и, следовательно, $\tau_1 = \tau_2$. В данном случае выполнение теста следует из того, что любая перестановка, и в частности, τ_1 является биекцией, поэтому выполнено условие $q_1 = q_2 \leftrightarrow \tau_1[q_1] = \tau_1[q_2]$. Поэтому тест в этом случае тоже истинен.

Так как все тесты программы (Θ, φ') выполняются на A_{ba} то, следовательно, $A_{ba} \vdash (\Theta, \varphi')$. Из чего следует, что исходная программа Π выполняется как на A_{ab}, так и на A_{ba}, и, следовательно, программа Π не может выражать порядок или принадлежность заданного элемента подполугруппе, порождённой другим заданным элементом. □

Список литературы

[1] Мусикаев И.Х., Тайцлин М.А. О динамических теориях свободных алгебр // Математический сборник, 180 (3), 1989. Стр.307–321.

[2] Тайцлин М.А. Пример полиномиального запроса не распознаваемого в недетерминированной логарифмической памяти // Вестник Тверского государственного университета, серия Прикладная математика, № 6(12), 2005, стр.5–22.

[3] Cook S.A. Variations on pushdown machines // In Proc. 1st ACM Symp. on Theory of Computing, pages 229–232, 1969.

[4] Cook S.A. Characterizations of push-down machines in terms of time-bounded computers // Journal of the ACM, 18(1):4–18, 1971.

[5] Cook S.A., Sethi R. Storage requirements for deterministic polynomial time recognizable languages // Journal of Computer and System Sciences, 13(1):25–37, 1976.