Алгебры различных классов программ

Н. Н. Непейвода*

Аннотация

Algebraic structures for three practically interested classes of program are described

1 Содержательная постановка задачи

Алгебры программ интенсивно рассматриваются в мировой литературе, им посвящены сотни работ (см., например, [2], где даны ссылки на большинство важнейших работ до середины 90-х, а также более новые работы). Практически все эти работы посвящены полным по Тьюрингу системам. Это находится в полном соответствии с доказавшим на многих примерах свою порочность, а порою уже и гибельность, духом современной науки и информатики (чем больше возможностей, тем лучше! Не дай Бог что-то пропустить, сразу некомпетентные «peer reviewers» и просто коллеги, не желающие разбираться в сути работы, закричат: “У Вас нет того-то! Он даже факториал не может запрограммировать! У него нет функции Аккермана!”).

В связи с этим невозможно отделить принципиально важные и общие аспекты от частностей тьюринговской модели вычислений. Более того, как показано, например, в [3], во многих случаях практически важные классы вычислений приводят к заведомо неполным по Тьюрингу системам. Поэтому в данной работе строятся алгебраические модели программ таким образом, чтобы они могли единообразно охватить все многообразие возможных программных систем: от автоматных до полных по Тьюрингу, и далее, до использующих физические устройства и физические датчики и тем самым заведомо выходящих за пределы тьюринговской вычислимости.

Но ограничивая себя, мы всегда многое приобретаем. Чтобы выразить глубокое чувство, мы ограничиваем свой язык и выражаем его в

*Институт программных систем имени А. К. Айламазяна РАН

стихах. Отказавшись от иллюзии всезнания, интуиционисты получили возможность выражать и явно использовать незнание. Здесь мы сразу добавляем к программам операции метапрограммирования, преобразующие программы. Это не соответствует функциональному программированию, где программа используется как целостный модуль для ее применения либо построения на базе ее новой программы, не изменяющей исходную, а использующей ее как параметр.

Поскольку прослеживается, что в ближайшее время будут играть важнейшую роль инъективные и обратимые программы. эти классы программ анализируются отдельно. Кроме того, наш подход позволяет бросить новый взгляд на нильпотентные программы и на автоматное программирование.

Вместо подозрительного термина «множество» мы часто будем использовать понятие «тип элементов». Слово «множество» резервируется для тех случаев, когда построенная совокупность заведомо не может иметь вычислительной интерпретации, либо когда на самом деле она не построена, а доказана грязная теорема существования, использующая, например, аксиому выбора.

Внимание! Для большей естественности соотношения с алгебраическими понятиями и для того, чтобы сразу выбить читателя из привычной численной и бинарной парадигмы, мы пишем функцию после ее аргумента и используем принятую в λ- исчислении и функциональном программировании запись применений функций как списков. $(x\ f)$ — применение функции f к аргументу x

2 Общие алгебраические структуры с точки зрения программиста

В самом общем случае программы, рассматриваемые как алгебраическая структура вычисляемых ими функций, образуют полугруппу, операцией в которой является композиция программ (или их последовательное исполнение, что в данном случае то же самое). Напомним элементарные алгебраические определения.

Определение 1. Сигнатура σ — список символов: функций (бинарные функции могут записываться между их аргументами, как операции), констант и предикатов. Каждой функции и предикату сопоставлена их арность — количество аргументов. Если в сигнатуре присутствует предикат $=$, он обязательно интерпретируется как настоящее равенство.

Для обозначения принадлежности символа сигнатуре будем использовать обычное обозначение $\in$.

Алгебраическая система сигнатуры σ $\mathbf{S}$ — интерпретация этой сигнатуры, то есть тип данных либо множество (носитель системы) S, и функция второго типа, сопоставляющая каждому символу функции функцию $S^{\mathrm{arity}(f)} \to S$, символу константы — элемент носителя, символу предиката — булевозначную функцию $S^{\mathrm{arity}(P)} \to \{\mathrm{false}, \mathrm{true}\}$.

Определение 2. *Группоид* — тип элементов (множество) с заданной на нем бинарной операцией $\circledast$. Таким образом, сигнатура группоида — $\langle \circledast, = \rangle$ Элемент называется нейтральным и обозначается e, если $x \circledast e = e \circledast x = x$. Элемент называется нулевым и обозначается 0, если $x \circledast 0 = 0 \circledast x = 0$. Группоид называется *полугруппой*, если операция ассоциативна. Группоид коммутативен, если операция коммутативна. Группоид инъективен, или обладает левыми сокращениями, если

$$x \circledast f = y \circledast f \Rightarrow x = y.$$

Соответственно, он обладает правыми сокращениями, если

$$z \circledast x = z \circledast y \Rightarrow x = y.$$

Группоид обладает правыми обратными, если для любого f найдется такое g, что $f \circledast g = e$. Группоид обладает левыми обратными, если для любого f найдется такое g, что $g \circledast f = e$. *Моноид* — полугруппа с единицей. *Группа* — моноид с левыми и правыми обратными. *Бигруппоид* — алгебраическая структура с двумя бинарными операциями, определенными на одном и том же множестве. *Мультигруппоид* — структура с несколькими бинарными операциями.

Напомним, что гомоморфизм алгебраических систем — отображение, сохраняющее все операции и константы и истинность всех предикатов, кроме предиката равенства. Изоморфизм — взаимно-однозначный гомоморфизм, или, что эквивалентно, такой гомоморфизм, что обратное отображение также гомоморфизм. Говорить о гомоморфизмах или изоморфизмах можно лишь для алгебраических систем с одним и тем же набором операций и констант (одной и той же сигнатурой).

Прокомментируем смысл этих понятий с точки зрения информатики. Произвольная неассоциативная операция задает на выражениях структуру упорядоченного бинарного дерева или же списков в смысле Lisp'a. В самом деле, выражение $((a \circledast (b \circledast c)) \circledast (a \circledast d))$ задает список Lisp ((a b c) a d), если интерпретировать операцию $\circledast$ как CONS, и бинарное дерево на рисунке 1. В случае коммутативности дерево становится

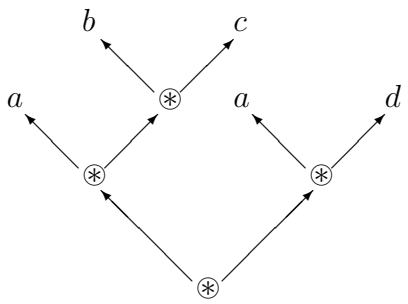

Рис. 1: $((a \circledast (b \circledast c)) \circledast (a \circledast d))$ как дерево

неупорядоченным, а прямые аналогии со списками пропадают.

Если же операция ассоциативна, все скобки можно убрать, список становится линейным. Более того, любая ассоциативная операция допускает интерпретацию как композиция функций. Это следует из классической теоремы.

Теорема 1. (Теорема о представлении для полугрупп) Каждая полугруппа изоморфна полугруппе функций с операцией композиции.

Доказательство. Рассмотрим некоторую полугруппу с типом элементов G и операцией $*$. Рассмотрим тип всех функций $G \to G$ вида $f_a = \lambda x.\, x \star a$, обозначим его F_G. Тогда имеем

$$((a\ f_b)\ f_c) = ((a\ \lambda x.\,(x * b))\ \lambda x.\,(x * c)) =$$
$$((a * b) * c) = (a * (b * c)) = (a\ \lambda x.\,(x * (b * c)) = (a\ f_{b*c})$$

Чтобы различить функции, которые могут совпасть, добавим к области определения каждой функции еще одно значение name и определим $(\text{name}\ f_a) = a$.

$$((\text{name}\ f_a)\ f_b) = (a\ f_b) = a * b = (\text{name}\ f_{a*b}).$$

□

Таким образом, говорить о том, что у нас имеется совокупность функций, которые мы можем применять последовательно, то же самое, что говорить: у нас есть полугруппа. Тем самым мы получаем заодно и критерий, чтобы проверить, можно ли данную совокупность программных действий рассматривать как функции: ассоциативность композиции.

Пример 1. Рассмотрим совокупность действий программной системы. Пусть для некоторых действий a определено правое обратное a^{-1}: отмена только что произведенного действия a. Если же предшествующее

действие было не a, то a^{-1} ничего не делает. Тогда $a * a^{-1} = e$, но $a * (e * a^{-1}) = a * e = a$, $(a * e) * a^{-1} = a * a^{-1} = e$.

Аналогичными же рассуждениями можно показать, что общая операция undo (отменить предыдущее действие) также не может рассматриваться как функция.

Таким образом, *абстрагирование программ до функций не всегда является корректным действием.*

Сами понятия гомоморфизма и изоморфизма систем также являются важнейшими для информатики. Изоморфизм — совершенно корректный переход от одного представления данных к другому (например, от алгебраической формулы к дереву), не теряющий и не добавляющий никакой информации, не теряющий и не добавляющий никаких свойств. формулируемых на чисто математическом языке. Поэтому математики обычно высокомерно говорят в случае изоморфных структур: «Выбор структуры не имеет никакого значения». Но, как известно, в практических задачах выбор адекватного представления данных — три четверти успеха. Гомоморфизм — аналогия, переход к новому представлению, упрощающему данные за счет потери части информации, возможно, разрушающему некоторые свойства, но сохраняющему основные преобразования данных. Это более рискованный, но часто исключительно эффективный метод действий.

Поскольку в полугруппах операция всегда может быть интерпретирована как композиция функций, она обычно обозначается $\circ$. Насколько широка область применения полугрупп, можно видеть из следующих примеров. Любая совокупность функций, замкнутая относительно композиции, является полугруппой. Любая совокупность отношений (то есть частичных недетерминированных функций), замкнутая по композиции, является полугруппой. Строки образуют полугруппу. Программы с операцией последовательной композиции (обычно обозначаемой точкой с запятой) образуют полугруппу как синтаксические структуры, даже если их действия не являются функциональными. Каждый конечный автомат может рассматриваться как конечная полугруппа, порожденная действиями входных символов на состояниях автомата. И каждая конечная полугруппа может рассматриваться как конечный автомат.

Полугруппа заодно порождает любопытный вариант функционального программирования. Каждый элемент a полугруппы может рассматриваться и как данное, и как функция $\lambda x.\, x \circ a$. Соответственно, полугруппа изоморфна полугруппе своих естественных гомоморфизмов, и теперь мы можем рассматривать функции как данные. Так что операция умножения может быть интерпретирована и как применение функции, и как

применение функционала к функции, и так далее.

Теперь рассмотрим коммутативные неассоциативные действия. Область их применения намного уже, чем неассоциативных и некоммутативных либо ассоциативных. Структура данных, порождаемая ими — неупорядоченное двоичное дерево.

Ассоциативная коммутативная операция порождает коммутативную полугруппу. С точки зрения функциональной интерпретации, коммутативность полугруппы означает, что наши действия независимы. С точки зрения вычислительной она означает, что они могут исполняться параллельно. С точки зрения логики — что их можно рассматривать как траты некоторого ресурса, автоматически «списываемого» с центрального счета, а еще лучше — с нескольких независимых счетов. Каждая операция не зависит по потраченным ресурсам от других, исполнявшихся ранее или исполняющихся параллельно.

Из-за столь важной роли полугрупп, частные понятия из теории полугрупп имеют фундаментальный смысл в информатике. Единица e означает действие «ничего не делать». Поскольку в реальности такое действие не всегда возможно (например, мы не можем остановить время), не всегда нужно предполагать, что полугруппа является моноидом. В синтаксических полугруппах e означает пустой элемент, добавление которого ничего не меняет. В автомате — это входной символ, который не меняет состояний автомата. 0 в функциональных полугруппах естественно интерпретируется как ошибка. В полугруппах отношений и частичных функций это — нигде не определенная функция и, соответственно, пустое отношение. Если в полугруппе более одного элемента, то 0 и e различны.

Если функции взаимно-однозначны, то появляются обратные элементы и полугруппа превращается в группу. Более того, любая полугруппа взаимно-однозначных функций легко пополняется до группы (добавим обратные элементы для всех и замкнем получившийся тип функций относительно операций композиции и обращения). Группа означает, что для любого действия есть полное обратное, которое может и предотвратить его, и аннулировать его результаты после выполнения

$$a \circ a^{-1} = e; \qquad a^{-1} \circ a = e. \tag{1}$$

Область применения групп также широка. Изоморфизмы алгебраической структуры на себя образуют группу. Симметричность объекта означает, что есть группа преобразований, при которой он сохраняется (например, сдвиги и повороты для мозаики, перестановки переменных для фрагмента программы или формулы и так далее...). Кристаллы

описываются группами симметрий. Динамические системы — группами инвариантов и сдвигов. Примеры неисчислимы...

Не удивляйтесь, что мы не переопределили понятия гомоморфизма и изоморфизма для полугрупп и групп. В алгебраических системах гомоморфизмы сохраняют все тождества [1]. Но, естественно, в группе появляется новая операция взятия обратного, которая тоже должна сохраняться при гомоморфизмах (даже это выполнится автоматически, поскольку обратный определяется двумя тождествами (1)).

Пример 2. Рассмотрим гомоморфизм кольца целых чисел в поле остатков по большому простому модулю p: $\lambda x : \mathrm{int}.\, x \ \mathbf{mod}\ p$. Он сохраняет операции сложения, умножения, вычитания и деления для всех натуральных чисел, меньших p (или целых, меньших по модулю $p/2$, если решим использовать большие числа как коды отрицательных, как и сделано в стандартной машинной арифметике). А вот если воспользоваться гомоморфизмом $\lambda x : \mathrm{int}.\, x \ \mathbf{mod}\ 2^n$, то появляются делители нуля и другие гадости. Тем не менее бездумно и автоматически этот порочный гомоморфизм используется во всех современных компьютерах.

3 Общая алгебра программ

Из рассмотрений в предыдущем параграфе видно, что не любая совокупность программ и действий может быть описана их функциями. Таким образом, пространство программ в общем случае не является полугруппой. Но синтаксическая структура программ, которая важна при их преобразованиях, полугруппой является.

Заметим, что любой группоид порождает полугруппу действий.

Определение 3. Действие последовательности элементов группоида $f_1, \ldots, f_n$ — функция

$$\varphi_{f_1;\ldots;f_n} = \lambda x. (\ldots((x \circledast f_1) \circledast f_2)\ldots) \circledast f_n).$$

Предложение 1. Действия любого группоида являются полугруппой относительно операции композиции функций.

Доказательство. Композицией $\varphi_{f_1;\ldots;f_n}$ и $\varphi_{g_1;\ldots;g_k}$ является $\varphi_{f_1;\ldots;f_n;g_1;\ldots;g_k}$. Композиция функций всегда ассоциативна.
□

На данном примере видно, почему мы предпочли несколько необычный порядок записи функции и ее аргументов: в сложных случаях он намного естественнее.

Рассмотрим теперь применение программы к аргументу $x \star f$. Эта операция для большинства программных систем неассоциативна, как было показано выше, она согласована с композицией, но не сводится к ней. Поскольку любой элемент у нас может играть роль и функции, и данного ($f \star x$ столь же законная операция), алгебра применений заодно дает возможность описать преобразования программ.

Мы приходим к алгебре, являющейся бигруппоидом с двумя операциями. Одна из них полугрупповая операция композиции $\circ$, вторая — неассоциативная операция применения программы $\star$. Они связаны следующими тождествами.

$$((x \star f) \star g) = (x \star (f \circ g)) \tag{2}$$

$$(0 \star x) = (x \star 0) = 0 \tag{3}$$

$$(e \star x) = (x \star e) = e \tag{4}$$

Такие алгебры назовем GAPS (Generic algebraic program structure).

Первое тождество выполняется всегда, оно является определением последовательного исполнения. Два последних выполнены лишь в том случае, если синтаксическая полугруппа содержит 0 или e. Если хоть одна из этих констант присутствует, она сохраняет свой смысл и для операции $\star$.

Напомним понятие обогащения алгебраической системы. Пусть дана сигнатура σ_1, расширяющая сигнатуру σ. Алгебра $\mathbb{A}_1$ сигнатуры σ_1 является обогащением алгебры $\mathbb{A}$ сигнатуры σ, если носители, константы и операции из σ остаются без изменения.

Пусть имеется некоторый гомоморфизм $\alpha : G \to (G \to G)$ нашей полугруппы в полугруппу функций над ее носителем, такой, что $(e\ \alpha) = \lambda x.\, x$, $(0\ \alpha) = \lambda x.\, 0$. С точки зрения программиста, задаем интерпретатор или транслятор наших программ, перерабатывающий запись программы в исполняемый модуль. Синтаксически различные программы могут порождать один и тот же исполняемый модуль, так что считать это отображение изоморфизмом нельзя.

Предложение 2. Каждая полугруппа $\mathbb{G}$ может быть обогащена до GAPS таким образом, что $x \star f = (x\ (f\ \alpha))$.

Доказательство. Проверим тождества для GAPS. Второе и третье выполнены по построению. Убедимся в том, что и первое выполнено.

$$((x \star f) \star g) = ((x\ (f\ \alpha))\ (g\ \alpha)) = (x\ ((f\ \alpha) \circ (g\ \alpha))) =$$
$$(x\ (f \circ g\ \alpha)) = (x \star (f \circ g))$$

Имеется еще одна маленькая тонкость: всегда ли такой гомоморфизм существует? Достаточно рассмотреть тривиальный пример: определим $(x\ \alpha) = \lambda x.\, 0$ для всех делителей нуля, то есть таких x, для которых имеется $y \neq 0$, такое, что $x \circ y = 0$, а для всех остальных: $(x\ \alpha) = \lambda x.\, x$. Второй тривиальный вариант — просто определить $x \star y = x \circ y$.
□

Предложение 3. В любой GAPS для любого действия группоида φ имеется элемент α, такой, что $(x \star \alpha) = (x\ \varphi)$.

Доказательство. Рассмотрим произвольное действие $f_1,\ \ldots,\ f_n$. Применяя (2) и ассоциативность умножения в полугруппе, получаем:

$$(\ldots((x \circledast f_1) \circledast f_2)\ldots) \circledast f_n) = (x \circledast f_1 \circ f_2 \cdots \circ f_n).$$

Таким образом, в GAPS любая последовательность действий программ выразима действием одной программы.
□

Теперь рассмотрим важный вопрос: когда можно некоторую систему преобразований программ соединить с данной синтаксической полугруппой программ, тем самым образовав единый язык метапрограммирования? Эта задача естественно разбивается на три подпроблемы: соединение языка с системой преобразований без изменения языка (то есть полугруппы); сохранение некоторого подъязыка (подполугруппы) и превращение ранее неинтерпретированных или не нужных нам более синтаксических конструкций в действия преобразователей; пополнение существующего языка до метаязыка путем расширения полугруппы без изменения исходной.

Здесь начинает действовать еще одно важное алгебраическое понятие: полугруппа, порожденная совокупностью функций (или элементов полугруппы, или отношений, что с точки зрения математика в данном случае одно и то же). Неявно оно уже было использовано в определении действий группоида.

Определение 4. Полугруппа $\mathbb{F}$, порожденная совокупностью функций $mathbf F$ — минимальный тип функций, включающий $mathbf F$ и замкнутый относительно композиций, то есть

$$\forall f, g\, (f \in \mathbb{F} \&) g \in \mathbb{F} \supset f \circ g \in \mathbb{F})$$

Содержательно это означает, что $\mathbb{F}$ состоит из всех функций, представимых как $f_1 \circ f_2 \cdots \circ f_n$, где $f_1, f_2, \ldots, f_n$ — элементы (не обязательно различные) $mathbf F$.

Определение 5. Пусть задана некоторая совокупность действий $\mathbb{A}$ как функции на носителе полугруппы $\mathbb{G}$. Она *консервативна*, если имеется GAPS $\mathbb{AG}$, которая является обогащением $\mathbb{G}$, такая, что каждое $\varphi \in \mathbb{A}$ представляется в ней как действие группоида: $(x \star \alpha) = (x\ \varphi)$ для некоторого α. Она *консервативна над подполугруппой* $\mathbb{G}_0$, если она консервативна и в $\mathbb{AG}$ $a \star b = a \circ b$ для всех элементов $\mathbb{G}_0$. Она *слабо допустима*, если имеется полугруппа $\mathbb{G}_1$, подполугруппой которой является $\mathbb{G}$, и функции из $\mathbb{A}$ могут быть продолжены на $\mathbb{G}_1$ таким образом, что полученная $\mathbb{A}_1$ консервативна над $\mathbb{G}$. Она *допустима*, если она слабо допустима и в $\mathbb{A}_1$ остаются истинными все факты вида $f_1 \circ \ \cdots \ \circ f_n = g_1 \circ \ \cdots \ \circ g_k$, истинные в $\mathbb{A}$.

Очевидно, что самое сильное из этих понятий — консервативность над полугруппой. Далее идут консервативность, допустимость, слабая допустимость. Рассмотрим простейшие условия возможности всех четырех обогащений и расширений.

Предложение 4. Совокупность действий $\mathbb{A}$ консервативна тогда и только тогда, когда ее замыкание изоморфно подполугруппе $\mathbb{G}$.

Доказательство. По предложению 3, если обогащение удалось осуществить, то любой элемент полугруппы, порожденной $\mathbb{A}$, представляется как действие некоторого элемента $\mathbb{G}$. Значит, мы изоморфно вложили замыкание $\mathbb{A}$ в $\mathbb{G}$.

Обратно, если замыкание $\mathbb{A}$ вкладывается в $\mathbb{G}$, то просто сопоставим каждому действию из $\mathbb{A}$ соответствующий ему при изоморфизме элемент $\mathbb{G}$.

$\square$

Заметим, что важнейшим фактором является вложение структуры полугруппы действий, а не самих действий.

Пример 3. Если у нас имеется всего одно действие сильной инверсии программ $\mathbf{M}$, обладающее свойством $((a\ \mathbf{M})\ \mathbf{M}) = a$, то обогатить полугруппу программ этим действием удается тогда и только тогда, когда у нас имеется элемент порядка 2, то есть такая программа, что $f \neq e \& f \circ f = e$. Как эта f действует на данные, нам все равно. Важно, что мы можем доопределить ее как операцию метапрограммирования без разрушения общей структуры вычислений.

Предложение 5. Совокупность действий $\mathbb{A}$ консервативна над $\mathbb{G}_{\not\nu}$ тогда и только тогда, когда имеется такое изоморфное вложение ψ ее замыкания в $\mathbb{G}$, что для каждого действия f, такого, что $(f\ \psi) \in \mathbb{G}_{\not\nu}$, выполнено $(a\ f) = (a \circ (f\ \psi)$.

Доказательство. Если такое обогащение удалось осуществить, то действие, попавшее в $\mathbb{G}_{\nvdash}$, обязано по условию сохранения подполугруппы, удовлетворять $(a\ f) = (a \circ (f\ \psi))$.

Обратно, если такой изоморфизм имеется, то просто сопоставим каждому действию из $\mathbb{A}$ соответствующий ему при изоморфизме элемент $\mathbb{G}$. $\square$

Пример 4. Например, пользуясь этим критерием, можно проверить возможность обогащения языка до языка метавычислений, рассматривая строки как программы, а подъязык, работающий с числами, оставляя неизменным.

Теорема 2. Любая совокупность действий $\mathbb{A}$ допустима над любой $\mathbb{G}$.

Для построений необходимы понятия свободной алгебраической системы, фактов, матрицы системы, тождеств и факторизации (приведенные, в частности, в книге Мальцева [1]).

Определение 6. *Свободная алгебраическая система* сигнатуры σ над совокупностью констант $\mathbb{C}$ — система всех выражений, которые могут быть построены из констант σ и $\mathbb{C}$ при помощи операций σ. Выражения в свободной системе считаются равными, если они совпали текстуально.

Конгруэнтность — отношение эквивалентности, такое, что функции и все предикаты, кроме равенства, сохраняют его, то есть

$$\forall x_1, \ldots, x_n, y_1, \ldots, y_n\, (x_1 \sim y_1 \& \cdots \& x_n \sim y_n \supset (x_1\ \ldots\ x_n\ f) \sim (y_1\ \ldots\ y_n\ f))$$
$$\forall x_1, \ldots, x_n, y_1, \ldots, y_n\, (x_1 \sim y_1 \& \cdots \& x_n \sim y_n \supset (P(x_1, \ldots, x_n) \equiv P(y_1, \ldots, y_n)))$$

Фактор-система $\Sigma/\!\sim$ системы Σ по конгруэнтности $\sim$ — система, носителем которой является множество классов эквивалентности, то есть множеств $\hat{a} = \{x \mid a \sim x\}$. Функции и предикаты переносятся на фактор-систему следующим образом:

$$(\hat{x_1}\ \ldots\ \hat{x_n}\ f) = \{z \mid \exists y_1, \ldots, y_n\, (x_1 \sim y_1 \& \cdots \& x_n \sim y_n \& z = (y_1\ y_n\ f))\};$$
$$P(x_1, \ldots, x_n) \equiv \exists y_1, \ldots, y_n\, (x_1 \sim y_1 \& \cdots \& x_n \sim y_n \& P(y_1, \ldots, y_n))\}$$

Факт — выражение вида $P(t_1, \ldots, t_n)$ либо $\neg P(t_1, \ldots, t_n)$, не содержащее переменных, в частности, $t = u$ или $t \neq u$. Факт *позитивный*, если он не начинается с отрицания. *Матрица системы* — совокупность всех истинных в ней фактов, при условии, что сигнатура пополнена константами для всех элементов носителя системы.

Тождество — формула вида $\forall x_k, \ldots, x_k\, P(t_1, \ldots, t_n)$, где x_i — все свободные переменные элементарной формулы.

Понятие свободной системы близко каждому программисту. Фактор-система образуется, когда у одного и того же объекта много представлений. Один из важнейших для информатики видов фактор-систем — системы, образующиеся в результате доказательства тождеств и фактов.

Предложение 6. Если Th — некоторая теория, то все выводимые в ней факты типа $t = u$ задают отношение конгруэнтности на свободной алгебре, построенной на константах произвольной модели данной теории.

Очевидно. $\square$

Систему, образующуюся в результате факторизации по такому отношению конгруэнтности, назовем Σ/Th.

Доказательство теоремы 2. Построим соответствующее расширение и обогащение. Возьмем свободную алгебру **GA** на базе наших двух операций, всех элементов полугруппы $\mathbb{G}$ и действий $\mathbb{A}$. Зададим теорию $\mathrm{Th}\{\mathbb{GA}\}$ со следующими аксиомами:

1. Матрица полугруппы $\mathbb{G}$.
2. Все заданные факты про $\mathbb{A}$, переведенные в форму тождеств
$$\forall x\,(x \star t) = (x \star u).$$
3. $\forall x, y, z \quad x \circ (y \circ z) = (x \circ y) \circ z.$
4. $\forall x, f, g \quad ((x \star f) \star g) = (x \star (f \circ g)).$
5. $\forall x\, x{\star}0 = 0$, $\forall x\, 0{\star}x = 0$, $\forall x\, x{\circ}0 = 0$, $\forall x\, 0{\circ}x = 0$, если 0 присутствует в нашей полугруппе.
6. $\forall x\, x{\star}e = x$, $\forall x\, e{\star}x = x$, $\forall x\, x{\circ}e = x$, $\forall x\, e{\circ}x = x$, если e присутствует в нашей полугруппе.

Если эта теория непротиворечива, то положим $\mathbb{GA} \triangleq \mathbf{GA}/\mathrm{Th}\{\mathbb{GA}\}$. Эта алгебра будет искомой.

Теперь покажем, что $\mathrm{Th}\{\mathbb{GA}\}$ всегда непротиворечива. Поскольку заданные нами факты для $\mathbb{A}$ являются переводом на язык нашей алгебры свойств полугруппы функций, они совместимы с аксиомой ассоциативности операции $\star$. Тем самым они совместимы с аксиомой (2) и с исходной полугруппой и не задают никаких новых равенств на ней.
$\square$

Следовательно, понятие слабой допустимости является излишним. Оно введено, чтобы лучше структурировать определения. Но наше доказательство на самом деле позволяет дать полный общий теоретический

критерий пополнимости некоторой системы операций над программами с заданными свойствами операций до GAPS над уже имеющейся у нас программной системой.

Заменим в построении теории $\mathrm{Th}\{\mathbb{GA}\}$ из доказательства 2 пункт 2 на произвольную совокупность аксиом Th_0 в сигнатуре $\mathrm{GAPS}\cup\{\mathbb{G},\mathbb{A}\}$, пополненной константами для всех элементов полугруппы и всех заданных действий.

Теорема 3. Совокупность действий $\mathbb{A}$, описываемая теорией Th_0, допустима над $\mathbb{G}$ в том и только в том случае, когда теория $\mathrm{Th}\{\mathbb{GA}\}$ непротиворечива.

Доказательство копирует первую часть доказательства предыдущей теоремы: строим алгебру $\mathbf{GA}/\mathrm{Th}\{\mathbb{GA}\}$. Но теперь получившаяся теория не всегда непротиворечива.
□

Принципиально данное решение полное. Но на практике оно, конечно же, редко применимо. Однако есть один практический критерий, который следует из данной теоремы.

Практическое следствие. *Если система преобразований программ разрушает свойства каких-то уже имеющихся программ или заставляет отождествить разные программы, она принципиально некорректна.*

А теперь пара замечаний, показывающих некоторое ехидство нашего аппарата.

Если в полугруппе есть единица, то универсальная функция, описываемая в нашей алгебре тождеством

$$(x \star (f \star \mathbf{U})) = (x \star f),$$

тривиальна: $\mathbf{U} = e$. Зато специализатор программ, описываемый

$$(f \star (x \star \mathbf{PE})) = (x \star f),$$

уже нетривиален почти всегда. И на этом тождестве видно, что специализатор является одним из видов универсальной функции.

Неподвижная точка программ, определяемая

$$((f \star \mathbf{Y}) \star f) = (f \star \mathbf{Y}),$$

тривиальна, если в полугруппе есть 0: она просто тождественно равна 0.

А теперь еще об одном. В принципе GAPS дают общее понятие автомата высших типов, значительно более абстрактное и более общее, чем то, которое дано, например, в [4]. Но непосредственно реализуемы как автоматы здесь могут быть лишь конечные GAPS, так же, как автоматы соответствуют конечным полугруппам.

Список литературы

[1] Алгебраические системы.

[2] Теория языков программирования.

[3] Axelsen H. G., Glück R. What do reversible programs compute? FOCSSACS 2011, LNCS 6604, pp. 42–56, 2011

[4] Непейвода Н.Н.: Уроки конструктивизма. Geidelberg: Lambert Academic Publishing, 98 pp. (2011)