Алгебраическая структура с частичными операциями и модель вычислений для арифметики ограниченных целых неотрицательных чисел

Ю. Г. Сметанин[1], М. В. Ульянов[2,3]
[1]*Вычислительный центр им. А.А. Дородницына РАН, Москва, Россия*
[2]*Национальный исследовательский университет-высшая школа экономики, Москва, Россия*
[3]*Московский государственный университет печати им. Ивана Федорова, Россия*
e-mail: smetanin.iury2011@yandex.ru, muljanov@mail.ru

Предложен математический формализм компьютерной целочисленной арифметики в виде оригинальной алгебраической структуры с частичными операциями (ограниченными по множеству допустимых операндов) для арифметики целых неотрицательных чисел. В качестве конструктивной реализации введённой алгебраической структуры предложена модель вычислений с предусловиями выполнения команд. Представленная модель является формальной моделью компьютера с ограниченной целочисленной арифметикой. Приведены алгоритмы, реализующие операции алгебраической структуры в элементарных операциях введённой модели вычислений. Показана применимость предложенной модели к задачам проверки допустимости и устранения особенностей входов арифметических выражений на основе эквивалентных преобразований представляющих их полиномов.

Ключевые слова: модель вычислений, предусловия выполнения команд, частично определённые алгебры, ограниченная арифметика, эквивалентные преобразования многочленов.

Введение

При анализе компьютерных алгоритмов очевидный интерес представляет задача построения такой формальной модели, которая позволила бы проводить теоретические исследования в области проверки допустимости входов, корректности вычислений и анализа ресурсной эффективности алгоритмов. Такая формальная модель называется моделью вычислений [1, 2] и должна, очевидно, учитывать реальные особенности хранения чисел и организации выполнения арифметических операций в реальном компьютере. Модель вычислений M, следуя [2], формально задаётся в виде упорядоченной пары

$$M = \langle M_S, R \rangle. \tag{1}$$

Здесь M_S — формальная модель памяти, включающая информационный носитель со структурой адресации, R — механизм реализации, представимый в виде следующей упорядоченной тройки:

$$R = \langle P, C, \Omega \rangle, \tag{2}$$

где P — абстрактный процессор, C — множество базовых операций модели вычислений, элементарно выполняемых процессором P, Ω — множество управляющих и операционных ячеек механизма реализации, используемых при выполнении операций из множества C над операндами — элементами информационного носителя.

Основные формализмы теории алгоритмов — машина Поста, машина Тьюринга, нормальные алгорифмы Маркова [4 – 5] — могут быть представлены в виде моделей вычислений, имеющих вид (1) и состоящих из указанных выше компонентов. Однако главное несоответствие этих формальных моделей реальному компьютеру состоит в том, что носитель данных моделей не ограничен и формально описывается счётным множеством, имеющим мощность $\aleph_0$. Даже наиболее полное описание алгебраического подхода к формализации компьютерных вычислений, основанное на аппарате "системы алгоритмических алгебр", содержащееся в [6], также оперирует понятием "бесконечного в обе стороны абстрактного двоичного регистра" (с. 115). Подход с бесконечным информационным носителем обеспечивает, очевидно, замкнутость арифметических операций в соответствующих алгебрах без ограничений на операнды, но этот подход противоречит реальным форматам данных и схемным или микропрограммным алгоритмам реализации арифметических операций в компьютере.

Таким образом, представляет интерес построение алгебраической структуры и модели вычислений, отражающих ограниченность представления чисел, равно как и порождаемые этой ограниченностью особенности арифметических операций. Такой аппарат интересен и с точки зрения формализации задачи проверки допустимости входов для произвольных арифметических выражений, т. е. проверки допустимости промежуточных и окончательного результатов ограниченным носителем. Один из возможных вариантов такой формализации для арифметических операций над ограниченным множеством целых неотрицательных чисел и составляет предмет исследования настоящей статьи.

1. Частично определённые алгебры и их применение

В соответствии с [7] алгеброй называется упорядоченная пара, состоящая из множества объектов, называемого носителем алгебры, и множества операций, заданных над элементами этого множества объектов, называемого сигнатурой алгебры. Алгебра, сигнатура которой содержит произвольные n-арные операции, причём для различных операций числа арности могут быть как различными, так и совпадающими, называется универсальной алгеброй [8]. Ван дер Варден в [9] вводит понятие алгебры (линейная алгебра) как кольца, являющегося конечномерным векторным пространством над некоторым полем с обязательным требованием замкнутости операций в носителе. Однако в соответствии с [8] n-арная операция универсальной алгебры называется частичной, если эта операция определена не для всех упорядоченных систем (кортежей) из n элементов носителя. В этом случае и сама алгебра называется частично определённой [8].

В настоящее время аппарат частичных алгебр широко используется для формализации и решения разнообразных задач. Приведём некоторые примеры его применения. Первый их них — задача представления траекторий на графах, формулируемая в терминах частично определённой алгебры. Сама задача относится к группе

задач оптимизации на графах. Подробное изложение результатов применения аппарата частично определённой алгебры к данной задаче содержится в [10], при этом формализация задачи имеет следующий вид.

Пусть $G = (V, E)$ — граф, где $E \subseteq V \times V$ — множество рёбер. Частичная алгебра задаётся носителем $I = V \times V$ и сигнатурой, состоящей из единственной операции L:

$$L : I \times I \to I, \quad \forall\, e_1 = (a, b), \quad e_2 = (c, d),$$

$$L((a, b), (c, d)) = \begin{cases} (a, d) \in I, & \text{если } b = c, \\ \text{Не определено}, & \text{если } b \neq c. \end{cases}$$

В терминах такой частично определённой алгебры замыкание $\widetilde{E}$ всего множества рёбер E по операции L, т. е. минимальное множество, состоящее из результатов применения операции L к объектам из E, замкнутое относительно данной операции, содержательно есть множество всех путей в графе G. При этом для фиксированной пары вершин как элемента носителя I минимальная длина представления данного элемента в $\widetilde{E}$, т. е. минимальное по длине представление в виде суперпозиции операции L, есть длина минимального пути между этой парой вершин [10]. Таким образом, задача поиска кратчайшего пути между парой вершин a и b в графе, классически решаемая алгоритмом Дейкстры, сводится к задаче поиска минимального по длине суперпозиции элемента в множестве $\widetilde{E}$. Если множество $\widetilde{E}$ не содержит (a, b), то это интерпретируется как принадлежность исходных вершин двум несвязным компонентам графа.

Отметим, что в ряде публикаций рассматриваются представления, основанные на аппарате частично определённых алгебр, и для более широкого класса структур, чем графы. Содержательный пример такого представления — применение данного аппарата к задаче алгебраической характеризации преобразований в категориях [11, 12].

2. Алгебраическая структура с частичными операциями

В настоящей статье мы применяем аппарат частичных алгебр к представлению арифметических операций, заданных на ограниченном множестве целых чисел. Далее, во избежание возможной некорректности в терминологии, будет использован термин “алгебраическая структура” [7, 13] и, поскольку требование замкнутости операций в носителе нарушено для арифметики ограниченного подмножества целых чисел, — термин “алгебраическая структура с частичными операциями”.

Предлагаемый формализм строится следующим образом. Обозначим через I_n множество первых неотрицательных целых чисел мощности n (обозначения в соответствии с [14]):

$$I_n \subset N_0, \quad I_n = \{0, 1, \ldots, n - 1\},$$

а через $B(I_n \times I_n)$ — множество всех подмножеств декартова произведения $I_n \times I_n$. Множество I_n представляет собой носитель создаваемой алгебраической структуры. Авторами предлагается ввести ограничения операндов для операций сигнатуры через специальную функцию — функцию ограничения носителя, использующую предикаты ограничения допустимых операндов (элементов носителя) для стандартных операций целочисленной арифметики.

Таким образом, *алгебраическая структура с частичными операциями для арифметики ограниченных целых неотрицательных чисел* — A_Q — вводится в виде упорядоченной четвёрки:

$$A_Q = \langle I_n, S, F, Q \rangle, \tag{3}$$

где I_n — носитель, S — сигнатура алгебраической структуры, F — функция ограничения носителя для операций, Q — множество предикатов ограничения операндов операций.

Сигнатура S структуры A_Q есть множество, включающее следующие пять арифметических операций:

$$S = \{s_i \mid i = \overline{1,5}\} = \{s_1 = ''+'', s_2 = ''-'', s_3 = ''*'', s_4 = ''/'', s_5 = ''//''\},$$

причём все операции — сложения $(+)$, вычитания $(-)$, умножения $(*)$, целочисленного деления $(/)$ и получения остатка от деления $(//)$ — имеют арность (2,1), т. е. получают одно число — элемент носителя I_n — как результат операции для пары операндов из подмножества $I_n \times I_n$. В силу целочисленности носителя операция деления даёт целый результат деления, а операция получения остатка — целочисленный остаток. Вводятся отдельные операции для деления и получения остатка, поскольку введение операции деления как операции с арностью

"(делимое, делитель) $\to$ (частное, остаток)"

противоречит требованию замкнутости результата операции в носителе. Отметим также, что операции деления и получения остатка при корректных операндах $(k, l) \in I_n$, $l \neq 0$, замкнуты в носителе, поскольку $k/l \leq \max(k, l)$, $k//l < \min(k, l)$.

Функция F — функция ограничения носителя — отображает множество операций во множество всех подмножеств $I_n \times I_n$, используя заданные множеством Q ограничения операндов, при этом нумерация предикатов в Q соответствует нумерации операций в S. Значением функции F является множество допустимых операндов операции:

$$F : S \to B(I_n \times I_n), \quad F(s_i) \subset B(I_n \times I_n), \quad F(s_i) = \{(k, l) \mid q_i(k, l) = true\},$$
$$(k, l) \in I_n \times I_n, \quad q_i \in Q. \tag{4}$$

Множество предикатов $Q = \{q_i \mid i = \overline{1,5}\}$ задаёт условия, ограничивающие операнды из I_n для частичных арифметических операций, а именно:

$$\left.\begin{array}{rcrcl} s_1 & = & ''+'': & q_1(k,l) & = & (k + l \leq n - 1), \\ s_2 & = & ''-'': & q_2(k,l) & = & (k \geq l), \\ s_3 & = & ''*'': & q_3(k,l) & = & (k * l \leq n - 1), \\ s_4 & = & ''/'': & q_4(k,l) & = & (l \neq 0), \\ s_5 & = & ''//'': & q_5(k,l) & = & (l \neq 0). \end{array}\right\} \tag{5}$$

Таким образом, каждая операция s_i построенной алгебраической структуры ограничена по операндам (аргументам операции) и задаётся соответствующим отображением

$$s_i : F(s_i) \to I_n, \quad i = \overline{1,5},$$

причём в силу (4) и (5) обеспечена замкнутость результата всех операций в носителе I_n.

3. Модель вычислений с предусловиями выполнения операций

Формальные компоненты модели вычислений (1) необходимо конкретизировать с учетом особенностей предложенной алгебраической структуры (3). С этой целью рассмотрим более подробно свойства алгоритмов [2], определяющие компоненты модели вычислений:

— свойство наличия данных, обрабатываемых алгоритмом, формализуется как существование некоторого множества объектов, над которыми выполняются операции, задаваемые алгоритмом. В данном случае такими объектами являются целые неотрицательные числа — элементы множества I_n. В целях конкретизации будем представлять эти числа в двоичной позиционной системе счисления с цифрами из алфавита $\{0, 1\}$. Фиксируя через k битовую длину записи чисел, получаем значение $n = 2^k$. Таким образом, носитель I_n есть множество целых неотрицательных чисел, двоичное представление которых имеет длину не более k бит;

— свойство наличия памяти для размещения данных приводит к введению формальной модели памяти [2], включающей информационный носитель и сигнатуру, которая представляет собой набор операций, определяющих способ доступа к этим данным;

— свойство априорности операций приводит к постулированию некоторого конечного набора операций, составляющих множество C, в дальнейшем называемых базовыми, которые однозначно определены и известны до формулировки алгоритма. Как правило, эти базовые операции включают в себя и операции доступа к объектам информационного носителя;

— свойство существования механизма реализации предполагает, что существует некоторый абстрактный механизм, который выполняет базовые операции над объектами носителя, включая и операции доступа к данным. Кроме того, предполагается, что механизм реализации распознает запись алгоритма, задающую последовательность базовых операций, и тем самым предполагается наличие ещё одной группы базовых операций, определяющих последовательность выполнения операций над данными в записи алгоритма. Как правило, предполагается, что механизм реализации имеет собственные ячейки, используемые при выполнении операций над операндами носителя. Эти ячейки формально описаны в (2) множеством управляющих и операционных ячеек Ω.

3.1. Формальная модель памяти

Предлагаемая формальная модель памяти для размещения данных с операциями доступа к ним строится на основе следующих предпосылок. Из возможных способов организации памяти (пространства символов в формализме Поста [3]) и методов адресации мы останавливаемся на структуре памяти с конечным числом ячеек, обладающих символическими именами. Каждая ячейка обладает уникальным символическим именем и хранит целое неотрицательное число. Для построения символических имен зафиксируем некоторый конечный алфавит символов Σ и введём в рассмотрение конечное множество символических имен A_Σ как подмножество множества Σ^+ — транзитивного замыкания алфавита по операции конкатенации Σ^* без пустого слова λ — $\Sigma^+ = \Sigma^* \backslash \lambda$:

$$A_\Sigma = \{a \mid a \in \Sigma^+\}, \quad A_\Omega \subset A_\Sigma, \quad A = A_\Sigma \backslash A_\Omega,$$

где A_Ω — конечное множество имен внутренних ячеек механизма реализации, A — конечное множество имен ячеек информационного носителя.

Введём информационный носитель формальной модели памяти — множество Y — как конечное множество ячеек, каждая из которых обладает символическим именем из A и может хранить объекты носителя — целые неотрицательные числа из I_n:

$$Y = \{y_i \mid i = \overline{1,m}\}, \quad Y \subset A \times I_n, \quad y_i = (a,z), \quad a \in A, \quad z \in I_n,$$

Здесь a — адресный, z — информационный компонент ячейки, при этом символические имена ячеек являются уникальными, т. е. разные элементы из Y не могут иметь одинаковые имена:

$$\forall z \in I_n : (y_i = (a_i, z)) \& (y_j = (a_j, z)) \Rightarrow i = j.$$

На этой основе вводим формальную модель памяти в виде упорядоченной пары

$$M_S = \langle Y, C_I \rangle. \tag{6}$$

где Y — носитель, C_I — множество операций механизма реализации над объектами из Y, состоящее из операции чтения содержимого ячейки и записи в ячейку с использованием дополнительных ячеек механизма реализации. Более подробно вариант такой формальной модели памяти описан в [15]. Мы считаем операции из C_I включёнными во множество базовых операций механизма реализации C и опишем их формат ниже.

3.2. Механизм реализации — абстрактный процессор

В состав механизма реализации в соответствии с (2) включаем его собственные ячейки, составляющие множество Ω. Результат алгоритма всегда формируется в специальной ячейке θ механизма реализации и в случае его допустимости возвращается в носитель. Для идентификации отсутствия результата операции введём в рассмотрение специальный элемент $''undef''$, который помещается в ячейку результата θ, но не возвращается в носитель. Таким образом, все исходные операнды арифметических операций являются корректными. Алгоритмы, реализующие вычисление арифметических выражений, возвращают в носитель только окончательный результат при условии, что он является корректным. При недопустимости результата ячейка θ содержит элемент $''undef''$.

Ячейки множества Ω имеют структуру, аналогичную структурам ячеек носителя информационной алгебры. Их символические имена составляют множество A_Ω, а именно:

$$A_\Omega = \{\alpha, \beta, \gamma, \theta, f, t, r, \mathrm{nul}, u, g\},$$

при этом сами элементы множества Ω определяются следующим образом:

$$\begin{gathered}\Omega = \{y_\alpha = (\alpha, z), y_\beta = (\beta, z), y_\gamma = (\gamma, z), y_\theta = (\theta, x),\\ y_f = (f, 0), y_t = (t, 1), y_r = (r, b), y_{\mathrm{nul}} = (\mathrm{nul}, z = 0),\\ y_u = (u, n-1), y_g = (g, ''undef''')\},\\ z \in I_{n^2}, x \in I_n \cup'' undef'', b \in \{0, 1\}.\end{gathered} \tag{7}$$

Таким образом, ячейки с именами α, β, γ — это операционные ячейки, которые могут хранить целые неотрицательные числа вплоть до значения $n^2 - 1$, обеспечивая тем самым замкнутость всех арифметических операций в механизме реализации над всеми операндами из множества I_n, так как $(n-1)(n-1) = n^2 - 2n + 1 < n^2 - 1$. Только в этих операционных ячейках процессор механизма реализации может выполнять арифметические операции; тем самым мы требуем предварительную загрузку операндов в операционные ячейки до выполнения операции. Ячейка θ хранит или допустимый результат, или значение $''undef''$. Ячейки с именами f, t хранят предопределённые значения 0 и 1, которые трактуются как логические значения, а ячейка с именем r хранит логический результат операции сравнения. Ячейка с именем nul хранит "длинный ноль" длиной $2k$ бит, ячейка с именем u — предопределённое значение максимального числа из I_n, а ячейка с именем g — специальное значение $''undef''$ для идентификации некорректного результата арифметической операции.

Рассмотрим множество C базовых операций модели вычислений, предопределённо выполняемых процессором механизмом реализации:

$$C = C_I \cup C_D \cup C_P \cup \{\text{stop}\},$$

где C_I — операции формальной модели памяти, C_D — операции сравнения и арифметики чисел, C_P — операции управления последовательностью выполнения операций в модели вычислений. Операция stop приводит к останову механизма реализации.

Операции из C_I включают в себя операции чтения и записи содержимого элементов информационного носителя в операционные ячейки механизма реализации. Для нотации будем использовать штрих-операцию [6] как операцию взятия содержимого по указанному символическому адресу. Напомним, что информационный компонент ячейки носителя имеет длину k бит, а соответствующий компонент операционной ячейки — длину $2k$ бит.

Рассмотрим базовые операции механизма реализации более подробно.

1. *Операция чтения из носителя в операционную ячейку*: "содержимое операционной ячейки $\leftarrow$ содержимое ячейки носителя". При этом старшие k бит операционной ячейки заполняются битовыми нулями.

Пример. $'\alpha \leftarrow 'a$. В операционную ячейку с символическим именем α помещается содержимое ячейки носителя с символическим именем a.

2. *Операция записи из операционной ячейки в носитель*: "содержимое ячейки носителя $\leftarrow$ содержимое операционной ячейки". При этом в ячейку носителя пересылаются только младшие k бит операционной ячейки, а в случае ячейки θ этими битами её содержимое исчерпывается.

Пример. $'b \leftarrow '\beta$. В ячейку носителя с символическим именем b помещается содержимое младших k бит операционной ячейки с символическим именем β.

Очевидно, что допустимы операции перемещения чисел внутри операционных ячеек, за исключением случая записи в ячейку с именем θ, при которой происходит запись только младших k бит операционной ячейки в силу определения ячейки θ как ячейки, согласованной с носителем информационной алгебры.

Операции из C_D включают в себя арифметические операции над числами и сравнения в операционных ячейках, причём указываются символические имена операндов и результата. Мы используем общепринятую запись арифметических операций.

3. *Операции арифметики целых чисел*:

“содержимое операционной ячейки $\leftarrow$ содержимое операционной ячейки
“символ операции”
содержимое операционной ячейки”.

Символ операции — элемент из множества $\{+, -, *, /, //\}$.

Примеры: $'\gamma \leftarrow '\beta + '\alpha$, $'\gamma \leftarrow '\beta - '\alpha$, $'\gamma \leftarrow '\beta // '\alpha$.

4. *Операция сравнения.* Для записи операций сравнения содержимого операционных ячеек будем использовать нотацию, предложенную Айверсоном [16]. Например, для двух чисел a, b при сравнении по “меньше” нотация имеет вид

$$[a < b] = \begin{cases} 0, & a \geq b, \\ 1, & a < b. \end{cases}$$

При сравнении результат, имеющий логическое значение, помещается в специальную операционную ячейку с символическим именем r. Сравнения разрешены только над содержимым операционных ячеек. Набор операций сравнения считаем стандартным. Формат операции сравнения: “содержимое ячейки $r \leftarrow$ сравнение в нотации Айверсона”.

Пример. $'r \leftarrow ['\beta < '\alpha]$.

Операции множества C_P управляют порядком выполнения последовательностей базовых операций. Мы вводим предусловие выполнения группы базовых операций, причём оно распространяется на всю группу, заключённую в фигурные скобки. Для ссылок на предусловия вводятся их имена. Предусловие формулируется как сравнение содержимого операционной ячейки r с содержимым одной из ячеек f или t. Постулируем, что базовые операции вне предусловий и внутри группы выполняются последовательно одна за другой.

5. *Операция выполнения группы операций по предусловию: Формат*: “имя предусловия: $['r = 't]$ {группа базовых операций}”.

6. *Операция перехода на предусловие*: “$\Rightarrow$ имя предусловия”.

7. *Операция* “stop”. Останов механизма реализации модели вычислений.

4. Реализация алгебраической структуры с частичными операциями в предложенной модели вычислений

Покажем, как частичные операции алгебраической структуры (3) реализуются в предложенной модели вычислений с предусловиями выполнения операций.

Запишем алгоритм сложения чисел, хранящихся в ячейках носителя с символическими именами a и b, с занесением результата в ячейку с символическим именем c при условии, что этот результат допустим:

$$
\begin{array}{rcll}
'\theta & \leftarrow & 'g & \\
'\alpha & \leftarrow & 'a & \\
'\beta & \leftarrow & 'b & \\
'\gamma & \leftarrow & '\alpha + 'b & \\
'r & \leftarrow & ['\gamma \le 'u] & \\
Q1 & : & ['r = ']t & \{ \\
 & & & '\theta \leftarrow '\gamma \\
 & & & 'c \leftarrow '\theta \\
 & & & \}
\end{array}
$$

stop.

В начале алгоритма предопределённое значение ячейки θ устанавливается в $''undef''$. Мы выполняем арифметическую операцию сложения с двумя числами из носителя I_n, причём, поскольку операционные ячейки механизма реализации позволяют оперировать с числами из I_{n^2}, результат в ячейке с символическим именем γ всегда разрешён. Последующее сравнение с ячейкой u определяет возможность возврата результата в носитель I_n, тем самым реализуется предикат $q_1(k,l) = (k + l \le n - 1)$ алгебраической структуры. Таким образом, в ячейку носителя с символическим именем c результат операции заносится только в случае, если он допустим. В специальной ячейке θ механизма реализации будет размещен или допустимый результат, или специальный элемент $''undef''$, идентифицирующий невозможность выполнения операции в заданном ограниченном подмножестве целых неотрицательных чисел.

Фрагмент для умножения отличается только базовой операцией $'\gamma \leftarrow '\alpha * '\beta$ и соответствующим предикатом $q_3(k,l) = (k*l \le n-1)$, реализация которого выполняется при сравнении результата умножения с ячейкой u, содержащей ограничивающее значение.

Для алгоритма, реализующего операцию вычитания, последовательность базовых операций несколько иная, поскольку множество I_{n^2} не содержит отрицательных чисел. Вначале необходимо проверить возможность выполнения операции вычитания:

$$
\begin{array}{rcll}
'\theta & \leftarrow & 'g & \\
'\alpha & \leftarrow & 'a & \\
'\beta & \leftarrow & 'b & \\
'r & \leftarrow & ['\alpha \ge '\beta] & \\
Q1 & : & ['r = 't] & \{ \\
 & & & '\gamma \leftarrow '\alpha - '\beta \\
 & & & '\theta \leftarrow '\gamma \\
 & & & 'c \leftarrow '\theta \\
 & & & \}
\end{array}
$$

stop.

Вычитание выполняется только при условии, что результат операции допустим по предикату $q_2(k,l) = (k \ge l)$. Поэтому прежде всего выполняется проверка, после которой — группа операций, состоящая из вычитания и сохранения результата.

При реализации целочисленного деления и получения остатка от деления мы сталкиваемся с необходимостью проверки делителя на ноль. Для этой цели служит специальная ячейка механизма реализации с именем nul:

$$
\begin{array}{lcll}
'\theta & \leftarrow & 'g & \\
'\alpha & \leftarrow & 'a & \\
'\beta & \leftarrow & 'b & \\
'r & \leftarrow & ['\beta \neq '\mathrm{nul}] & \\
Q1 & : & ['r = 't] & \{ \\
 & & & '\gamma \leftarrow '\alpha / '\beta \\
 & & & '\theta \leftarrow '\gamma \\
 & & & 'c \leftarrow '\theta \\
 & & & \}
\end{array}
$$

stop.

Если условие предиката $q_4(k, l) = (l \neq 0)$ выполнено, то производится целочисленное деление и формирование результата.

Приведём также пример алгоритма в элементарных операциях предложенной модели вычислений с предусловиями выполнения команд, который вычисляет арифметическое выражение $c \leftarrow (a + b) * d$:

$$
\begin{array}{lcll}
'\theta & \leftarrow & 'g & \\
'\alpha & \leftarrow & 'a & \\
'\beta & \leftarrow & 'b & \\
'\gamma & \leftarrow & '\alpha + '\beta & \\
'r & \leftarrow & ['\gamma \leq 'u] & \\
Q1 & : & ['r = 't] & \{ \\
 & & & '\alpha \leftarrow 'd \\
 & & & '\beta \leftarrow '\alpha * '\gamma \\
 & & & 'r \leftarrow ['\beta \leq 'u] \\
 & & & Q2 : ['r = 't] \quad \{ \\
 & & & \qquad '\theta \leftarrow '\beta \\
 & & & \qquad 'c \leftarrow '\theta \\
 & & & \qquad \} \\
 & & & \}
\end{array}
$$

stop.

Отметим, что фрагмент, вычисляющий произведение, вложен в предусловие допустимости результата сложения; таким образом, результат арифметического выражения возвращается в носитель только в случае, если допустимы результаты всех промежуточных арифметических операций.

5. Полиномиальное подмножество арифметических выражений и особенности входов

Особенности арифметики в ограниченном носителе приводят к очевидной проблеме выхода окончательного и/или промежуточных результатов вычислений за границы носителя, которая возникает при вычислении значений арифметических выражений для конкретных входов. Эта проблема конкретизируется в виде двух следующих задач:

1 — задачи проверки *допустимости* данного входа, т. е. проверки замкнутости в носителе промежуточных результатов вычисления арифметического выражения и его окончательного значения;

2 — задачи построения такого эквивалентного арифметического выражения, промежуточные результаты которого допустимы в носителе при условии, что окончательное значение этого выражения допустимо на данном входе, или констатации невозможности такого эквивалентного преобразования.

Предлагаемая формализация этих задач и возможный подход к их решению излагаются далее для частного случая представления арифметического выражения полиномом от его операндов. Тем самым рассматриваемый частный случай ограничивается алгебраической структурой (3) и соответствующей моделью вычислений (1)–(2), поддерживающей только три арифметические операции:

$$\tilde{S} = \{s_i \mid i = \overline{1,3}\} = \{s_1 = ''+'', s_2 = ''-'', s_3 = ''*''\}. \tag{8}$$

Будем называть *арифметическим выражением* полиномиальную функцию $g(X)$, где X — m-компонентный кортеж $(x_1, x_2, \ldots, x_m)$. Область определения аргументов ограничена множеством I_n, а область допустимых значений функции g есть множество I_n, объединённое с элементом $''undef''$:

$$g : I_n^m \to I_n \cup ''undef'', \quad g(X) = g(x_1, x_2, \ldots, x_m) = \sum c \prod_{j,k \in 1,\ldots,m} x_j, \cdots, x_k, \tag{9}$$

где c — некоторые константы из I_n. Запись вида (9) будем называть *обобщённой полиномиальной формой* арифметического выражения. Очевидно, что функция g является многочленом от m переменных, который отличается от общепринятой формы записью степеней в виде последовательности умножений. Таким образом, в соответствии с (9) мы отождествляем значение арифметического выражения со значением, получаемым в ячейке θ модели вычислений (7) в результате выполнения алгоритма, реализующего данное арифметическое выражение. Рассматриваемая совокупность многочленов g является, очевидно, подмножеством арифметических выражений, для которого разрешёнными операциями являются умножение, сложение и вычитание (8).

При этом запись полинома g порождена грамматикой арифметических выражений G [17], терминальные символы которой заданы множеством

$$T = \tilde{A} \cup \tilde{S} \cup \mathrm{Const} \cup \{(,)\},$$

где $\tilde{A} = \{x_1, x_2, \ldots, x_m\}$ — множество аргументов функции g (символических имён операндов), $\tilde{S}$ — подмножество арифметических операций (8), $\mathrm{Const} = \{c \mid c \in I_n\}$ — множество констант. Строковая запись функции g определяется порождающей грамматикой G, в терминальные символы которой включены скобки, определяющие порядок вычислений. Обозначим эту строковую запись через $[g]_s$. Порождённая грамматикой G форма арифметического выражения путём раскрытия скобок может быть сведена к обобщённой полиномиальной форме (9):

$$g_1(x_1, x_2) = (4x_1 + 7x_2) * x_2 + 3x_1^2, \quad g_2(x_1, x_2) = 4x_1 * x_2 + 7x_2^2 + 3x_1^2, \quad [g_1]_s \neq [g_2]_s.$$

Дальнейшее изложение предполагает рассмотрение полиномиального арифметического выражения — многочлена g, заданного именно в форме результата порождающей грамматики.

Обозначим через $X^{(0)} = (x_1^{(0)}, \ldots, x_m^{(0)}) \in I_n^m$ кортеж из m элементов множества I_n, представляющий собой конкретный вход арифметического выражения. При этом аргумент x_i функции g получает значение $x_i^{(0)}, i = \overline{1,m}$. Определим функцию g как тождественно совпадающую по записи с g: $[g]_s = [\tilde{g}]_s$, но обладающую другой областью значений промежуточных и окончательных результатов — множеством целых чисел Z: $\tilde{g} : I_n^m \to Z$. При вычислении значения $g(X^{(0)})$ могут возникнуть три ситуации, которые авторами предлагается различать и терминологически. Будем называть вход $X^{(0)}$ для заданного функцией g полиномиального арифметического выражения

1 — *допустимым* входом, если $g(X^{(0)}) \in I_n$, т. е. арифметическое выражение вычислимо для $X^{(0)}$;

2 — входом с *потенциально устранимой особенностью*, если $\tilde{g}(X^{(0)}) \in I_n$, $g(X^{(0)}) = \text{= }''undef''$. Содержательно эта ситуация означает, что промежуточные результаты вычисления арифметического выражения лежат вне носителя I_n. Возможно, путём некоторых эквивалентных преобразований такое выражение приводится к виду, для которого вход X_0 является допустимым. В таком случае будем говорить о том, что этот вход является входом с *эквивалентно устранимой особенностью*;

3 — входом с *неустранимой особенностью*, если $\tilde{g}(X^{(0)}) \in Z \backslash I_n$, $g(X^{(0)}) = ''undef''$. Это ситуация, когда окончательное значение арифметического выражения g для входа X_0 лежит вне носителя I_n. Очевидно, что в данной ситуации никакие эквивалентные преобразования g не приведут к допустимости входа.

6. Устранение особенностей входов на основе эквивалентных преобразований

Следующей целью представленного исследования является введение понятия эквивалентных арифметических выражений и описание методики эквивалентных преобразований. Два арифметических выражения, заданных полиномиальными функциями $g(X)$ и $h(X)$, будем называть *эквивалентными*, если $\forall X_0 \in I_n^m$ имеет место одна и только одна из следующих ситуаций:

$$\left.\begin{array}{lll} 1. & g(X^{(0)}) = h(X^{(0)}), & g(X^{(0)}) \in I_n, \quad h(X^{(0)}) \in I_n, \\ 2. & \tilde{g}(X^{(0)}) = \tilde{h}(X^{(0)}), & g(X^{(0)}) \in I_n, \quad h(X^{(0)}) = ''undef'', \\ 3. & \tilde{g}(X^{(0)}) = \tilde{h}(X^{(0)}), & g(X^{(0)}) = h(X^{(0)}) = ''undef''. \end{array}\right\} \qquad (10)$$

Таким образом, для конкретного входа $X^{(0)}$ ситуация 1 содержательно означает совпадение и замкнутость результатов двух арифметических выражений в I_n, ситуация 2 — совпадение результатов в Z с допустимостью только одного результата в I_n, ситуация 3 — совпадение результатов в Z с недопустимостью обеих результатов в I_n. Особый интерес представляет ситуация 2, так как в этом случае возникает возможность некоторого эквивалентного преобразования полиномиального арифметического выражения $h(X)$, приводящего к допустимости (в I_n) его значения на входе $X^{(0)}$. В этом случае, используя введённую выше терминологию, можно говорить,

что вход $X^{(0)}$ является входом с эквивалентно устранимой особенностью для арифметического выражения $h(X)$, причём эквивалентным арифметическим выражением для $h(X)$ является арифметическое выражение $g(X)$.

Заметим, что функция $g(x_1, x_2, \ldots, x_m)$ в общем случае представима в виде композиций функций $s_1(\cdot), s_2(\cdot)$ либо $t_1(\cdot), t_2(\cdot), t_3(\cdot)$, объединённых операциями из $\tilde{S}$, аргументы которых являются подмножествами множества аргументов функции g. Преобразование $L : g_1(X) \to g_2(X)$, задаваемое оператором L, будем называть *эквивалентным преобразованием*, если L есть произвольная суперпозиция операторов L_c, L_a, L_d:

$$\left.\begin{array}{l} L_c : \left\{ \begin{array}{l} s_1(\cdot) \pm s_2(\cdot) \to s_2(\cdot) \pm s_1(\cdot), \\ s_1(\cdot) * s_2(\cdot) \to s_2(\cdot) * s_1(\cdot), \end{array} \right. \\ L_a : \left\{ \begin{array}{l} (t_1(\cdot) \pm t_2(\cdot)) \pm t_3(\cdot) \to t_1(\cdot) \pm (t_2(\cdot) \pm t_3(\cdot)), \\ (t_1(\cdot) * t_2(\cdot)) * t_3(\cdot) \to t_1(\cdot) * (t_2(\cdot) * t_3(\cdot)), \end{array} \right. \\ L_d : (t_1(\cdot) \pm t_2(\cdot)) * t_3(\cdot) \Leftrightarrow t_1(\cdot) * (t_3(\cdot) \pm t_2(\cdot) * t_3(\cdot)). \end{array}\right\} \quad (11)$$

Конкретную суперпозицию операторов L_c, L_a, L_d будем обозначать через $L^{(0)}$. На основании свойств кольца Z полиномиальные арифметические выражения, заданные функциями $g_1(X)$ и $g_2(X) = L(g_1(X))$, эквивалентны в смысле (10).

Пример. $g_1(x_1, x_2, x_3) = x_1 * x_2 - x_1 * x_3$, $L(g_1) = g_2(x_1, x_2, x_3) = x_1 * (x_2 - x_3)$.

Заметим, что в носителе I_{32} для входа $X^{(0)} = (6, 6, 5)$ значение $g_1(X^{(0)}) = \text{''}undef\text{''}$, поскольку $x_1 * x_2 = 36$, но при этом $g_2(X^{(0)}) = 6$ и $6 \in I_{32}$. Эти арифметические выражения эквивалентны в смысле (10), а вход $X^{(0)} = (6, 6, 5)$ является входом с эквивалентно устранимой особенностью для $g_1(X)$. Эквивалентным арифметическим выражением, устраняющим особенность данного входа, является функция $g_2(X)$.

Лемма 1. *Если разность многочленов $g_1(X)$ и $g_2(X)$, приведенных к обобщённой полиномиальной форме, есть нулевой многочлен (многочлен с нулевыми коэффициентами), то многочлен $g_2(X)$ может быть получен из $g_1(X)$ с помощью преобразования L.*

Доказательство. Приведём многочлен $g_1(X)$ к обобщённой полиномиальной форме $\tilde{g}_1(X)$ с использованием эквивалентного преобразования $L_1^{(0)}$ (преобразование раскрытия скобок); аналогично приведём многочлен $g_2(X)$ к обобщённой полиномиальной форме $\tilde{g}_2(X)$ с использованием эквивалентного преобразования $L_2^{(0)}$. По условию леммы $\tilde{g}_1(X) - \tilde{g}_2(X) \equiv 0$, из чего следует почленное равенство всех коэффициентов у приведённых многочленов. Тогда ${L_2^{(0)}}^{-1} L_1^{(0)}(g_1(X)) = g_2(X)$, а суперпозиция эквивалентных преобразований является эквивалентным преобразованием, что и доказывает утверждение леммы. □

Рассмотрим применение входа $X^{(0)}$ к многочлену $g(X)$. Последовательно подставляя в $g(X)$ значения $x_m^{(0)}, \ldots, x_2^{(0)}$, придём к представлению $g(X)$ в виде многочлена от одной переменной x_1:

$$\hat{g}(x_1) = a_r x_1^r + a_{r-1} x_1^{r-1} + \cdots + a_0, \quad (12)$$

где коэффициенты a_i есть результаты подстановки $x_m^{(0)}, \ldots, x_2^{(0)}$ в $g(X)$. Предполагаем далее, что вход $X^{(0)}$ допустим для $g(X)$ в I_n, и, следовательно, $a_i \in I_n$.

Лемма 2. *Если арифметические выражения, заданные многочленами от x_1: $\hat{g}_1(x_1)$ и $\hat{g}_2(x_1)$ — эквиваленты в смысле (10) и степень каждого из них меньше n, то $g_1(X)$ можно получить из $g_2(X)$ с помощью преобразования L.*

Доказательство. Пусть $\hat{g}_1(x_1)$ и $\hat{g}_2(x_1)$ — многочлены, максимальная степень m которых меньше n по x_1. Рассмотрим многочлен $d(x_1) = \hat{g}_1(x_1) - \hat{g}_2(x_1)$. Если $d(x_1)$ — нулевой многочлен, то утверждение леммы следует из леммы 1. Если $d(x_1)$ не равен тождественно нулю, то он имеет не более m корней в $\Re_1$ и заведомо не более m корней в Z, а значит, и в I_n. Таким образом, $d(x_1)$ не может быть равен нулю для всех элементов I_n; следовательно, $\exists x_1^0 \in I_n : d\left(x_1^{(0)}\right) \neq 0$. Но это означает, что многочлены $\hat{g}_1(x_1)$ и $\hat{g}_2(x_1)$ не эквиваленты в смысле (10), что противоречит условиям леммы. □

Очевидно, что лемма 2 не применима, если степень многочлена $\hat{g}_1(x_1)$ больше n, так как такой многочлен может иметь n корней в I_n, однако справедлива следующая лемма.

Лемма 3. *Для арифметического выражения, заданного многочленом $g(X)$, преобразованная форма (12) которого $\hat{g}_1(x_1)$ имеет степень, большую или равную n, можно построить эквивалентный в Z многочлен меньшей степени.*

Доказательство. Пусть $\hat{g}(x_1) = a_{n+p}x_1^{n+p} + a_{n+p-1}x_1^{n+p-1} + \cdots + a_0$ и $p \geq 0$. Рассмотрим многочлен $h(x_1)$, представляющий собой убывающую факториальную степень $n+p$ по x_1:

$$h(x_1) = x_1^{n+p} = x_1(x_1-1)(x_1-2)\cdots(x_1-(n+p-1)) = \sum_{k=1}^{n+p} \left[\begin{array}{c} n+p \\ k \end{array} \right] (-1)^{n+p-k} x_1^k, \quad (13)$$

где квадратные скобки означают числа Стирлинга первого рода [16].

Заметим, что $h(x_1) \equiv 0$ в I_{n+p} и $\forall x_1 \in I_n$ заведомо обращается в ноль. В силу этого из (13) следует, что $\forall x_1 \in I_n$ имеет место эквивалентное преобразование (преобразование понижения степени) с коэффициентами из Z:

$$x_1^{n+p} = (-1) \sum_{k=1}^{n+p-1} \left[\begin{array}{c} n+p \\ k \end{array} \right] (-1)^{n+p-k} x_1^k. \quad (14)$$

Применяя замену (14) к $\hat{g}(x_1)$, получим многочлен $\hat{g}_1(x_1)$, имеющий степень не большую, чем $n+p-1$. Последовательно, а именно $p+1$ раз, применяя аналогичное (14) преобразование, можно получить многочлен $\hat{g}_p(x_1)$, имеющий степень меньшую n, с коэффициентами в Z. □

Из лемм 1–3 следует теорема.

Теорема. *Если вход $X^{(0)}$ является входом с потенциально устранимой особенностью для арифметического выражения, заданного полиномом $g(X)$, то задача построения эквивалентного арифметического выражения, возможно допускающего вход $X^{(0)}$, требует выполнения только двух видов преобразований — эквивалентного (11) и понижения степени (14).*

Доказательство. Если после преобразования понижения степени (14) все коэффициенты $\hat{g}_p(x_1)$ лежат в I_n, то арифметическое выражение, заданное многочленом $\hat{g}_p(x_1)$, может допускать вход $X^{(0)}$. Если многочлен $\hat{g}_p(x_1)$ не допускает вход $X^{(0)}$ в смысле ситуации 2 из (10), то согласно лемм 1 и 2, возможно, существует эквивалентное преобразование $L(\hat{g}_p(x_1))$ арифметического выражения $g(X)$, допускающего вход $X^{(0)}$. □

Отметим, что задача выбора эквивалентного преобразования $L(\hat{g}_p(x_1))$ построения некоторой суперпозиции операторов L_c, L_a, L_d является алгоритмически разрешимой в силу конечной символьной записи $[\hat{g}_p(x_1)]_s$ и, следовательно, конечного

числа возможных преобразований, допускаемых грамматикой G и эквивалентных в смысле (11). Вполне возможно, что все варианты эквивалентного преобразования не допускают вход $X^{(0)}$. Тогда за конечное время устанавливаем невозможность построения эквивалентно устранимого преобразования арифметического выражения $g(X)$ для входа $X^{(0)}$. Построение эффективного по трудоёмкости алгоритма выбора последовательности эквивалентных преобразований — суперпозиции операторов L_c, L_a, L_d — авторы рассматривают как продолжение настоящего исследования.

Заключение

Предложенный в работе формализм компьютерной целочисленной арифметики отражает ограниченность представления целых чисел в реальном компьютере и порождаемые этой ограниченностью особенности арифметических операций. В качестве конструктивной реализации введённой алгебраической структуры предложена специальная модель вычислений с предусловиями выполнения команд, представляющая собой формальную модель компьютера с ограниченной целочисленной арифметикой. Для частного случая представления арифметического выражения полиномом от его операндов показана возможность применения предложенного аппарата для решения задачи проверки входов арифметических выражений на допустимость результата в ограниченном носителе. Кроме того, предложен подход к построению эквивалентных арифметических выражений с целью обеспечения замкнутости в носителе задаваемых ими промежуточных результатов для конкретного входа.

Доказанная в статье теорема об эквивалентных преобразованиях полиноминально заданных арифметических выражений для входов с потенциально устранимой особенностью остается в силе и для носителя, содержащего отрицательные числа. По сути, полученный результат относится к ограниченности диапазона представления чисел, но не к конкретным границам этого диапазона. Имея исходный диапазон, можно, очевидно, путём сдвига значений отобразить его в сегмент $I_n = \{0, 1, \ldots, n-1\}$, а затем возвратить полученные результаты в исходный диапазон.

Очевидный интерес представляет и обобщение полученных результатов на алгебраическую структуру, включающую операции целочисленного деления и вычисления остатка от деления. Включение деления приводит к необходимости рассмотрения более общих полиномиальных структур, которые могут содержать отрицательные степени переменных и для которых доказанные леммы требуют дополнительных обобщений. Авторы рассматривают это обобщение как одно из продолжений данного исследования.

Предложенные формализм и модель вычислений могут быть использованы при теоретическом анализе компьютерных алгоритмов для решения задач проверки допустимости входов арифметических выражений и выполнения их эквивалентных преобразований.

Список литературы

[1] Алексеев В. Е., Таланов В. А. Графы и алгоритмы. Структуры данных. Модели вычислений: Учебник. М.: БИНОМ. Лаборатория знаний, 2006. 320 с.

[2] Ульянов М. В. Ресурсно-эффективные компьютерные алгоритмы. Разработка и анализ. М.: ФИЗМАТЛИТ, 2008. 304 с.

[3] Post E. L. Finite combinatory process — formulation 1 // J. Symbolic Logic. 1936. No. 1. P. 103–105.

[4] Turing A. M. On computable numbers, with an application to the Entsheidungsproblem // Proc. London Math. Soc. 1937. Vol. 42, ser. 2. P. 230–235.

[5] Фалевич Б. Я. Теория алгоритмов: Учеб. пособие. М.: Машиностроение, 2004. 160 с.

[6] Глушков В. М., Цейтлин Г. Е., Ющенко Е. Л. Алгебра. Языки. Программирование. Киев: Наук. думка, 1978. 318 с.

[7] Горбатов В. А. Основы дискретной математики. М.: Высш. шк., 1986. 311 с.

[8] Курош А. Г. Лекции по общей алгебре. М.: Наука, 1973. 400 с.

[9] Ван дер Варден Б. Л. Алгебра. М.: Наука, 1979. 624 с.

[10] Zhang G.-Q. Closures in binary partial algebras // Electr. Notes Theor. Comput. Sci. 2009. Vol. 257. P. 3–18.

[11] Burmeister P., Rossellό F., Torrens F., Valiente G. Algebraic transformation of unary partial algebras I: Double-pushout approach // Theor. Comput. Sci. 1997. Vol. 184, No. 1-2. P. 145–193.

[12] Burmeister P., Monserrat M., Rossello F., Valiente G. Algebraic transformation of unary partial algebras II: Single-pushout approach // Ibid. 1999. Vol. 216, No. 1-2. P. 311–362.

[13] Мальцев А. И. Алгебраические системы. М.: Наука, 1970.

[14] Белоусов А. И., Ткачёв С. Б. Дискретная математика: Учеб. для вузов / Под ред. В. С. Зарубина, А. П. Крищенко. М.: Изд-во МГТУ, 2001. 744 с.

[15] Михайлов Б. М., Головешкин В. А., Ульянов М. В. Модель вычислений с информационной алгеброй и предусловием выполнения элементарных операций // Вестник МГАПИ. Технические и естественные науки. 2006. Т. 3. С. 114–123.

[16] Грэхем Р., Кнут Д., Паташник О. Конкретная математика. Основание информатики. М.: Мир, 1998. 703 с.

[17] Грис Д. Конструирование компиляторов для цифровых вычислительных машин. М.: Мир, 1975. 544 с.

Поступила в редакцию 21 марта 2011 г.,
с доработки — 6 мая 2013 г.

ЖВТ. 2013. Т. 18, № 4. С. 48–63.

Algebraic Structure with Partial Operations and Computational Model for the Arithmetic of Bounded Nonnegative Numbers

Smetanin Yu.G., Uljyanov M.V.

A mathematical framework for computer integer arithmetic is proposed. The framework is based on the original algebraic structure with partial operations (that is, operations with bounded set of feasible operands). For the constructive implementation of the proposed algebraic structure, a model of computation with preconditions of the execution of the instructions is proposed. The proposed model is a formal model of a computer with bounded integer arithmetic. Algorithms that implement the operations of the algebraic structure are presented using elementary operations of the proposed computation model. The model is shown to be applicable for the problems of testing the feasibility and eliminating singularities of input arithmetic clauses using equivalent transformations of their representing polynomials.

Keywords: model of computation, preconditions of instructions, partial algebras, bounded arithmetic, equivalent transformations of polynomials.