АЛГОРИТМЫ И АЛГЕБРА

Ф.Л.Бауэр

Мюнхен, ФРГ

Современные тенденции развития теоретической информатики выдвигают на передний план алгебраические методы. Мы упомянем только два примера: использование теории решеток в семантике программ (Скотт, 1970) и универсальной алгебры в семантике абстрактных типов данных (Гуттаг, 1975). Цель этой заметки - обосновать функциональное содержание самого понятия алгоритма с помощью понятий алгебры, ознаменовав тем самым вдвойне связь Ургенческого симпозиума с аль-Хорезми.

Неформальное введение. Пусть .o. - бинарная операция на множестве M объектов - скажем, полугруппа натуральных чисел относительно сложения или умножения. Рассмотрим совокупность $W_{(M,.o.)}$ выражений, или термов, получаемых посредством этой операции. Эта совокупность включает, в частности, термы со свободной переменной

((((((a o a) o a) o a) o a) o a) o a)
(((a o a) o (a o a)) o (a o a)) o (a o a)
((a o a) o (a o a)) o ((a o a) o (a o a))

и термы с двумя свободными переменными x, y

x o y
y o x

Каждый терм с очевидностью задает последовательность вычислительных действий. Замена свободных переменных $x_1,\ldots,x_n$ в терме t элементами $e_1,\ldots,e_n$ из M называется конкретизацией; результирующий терм называется конкретизированным термом. Если конкретизированный терм не содержит больше свободных переменных, то мы говорим о полной, а в противном случае - о частичной конкретизации. Термы без свободных переменных ("замкнутые" термы) называются также вычислительными термами; (пошаговое) нахождение элемента, задаваемого такими термами, обычно называется вычислением.

При выполнении определенных законов некоторые термы оказываются функционально эквивалентными, т.е. они определяют одну и ту же функцию, хотя и с помощью разных вычислений. При законе ассоциативности верхние три терма оказываются функционально эквивалентными, при законе коммутативности – нижние два. Таким образом, при соблюдении законов на множестве $W_{(M,.o.)}$ возникают классы функционально эквивалентных термов и вычислений.

Явное выписывание термов может быть очень громоздко, а иногда и вовсе невозможно практически. Поэтому мы нуждаемся в механизмах для краткого описания больших вычислительных термов. Следуя обычному математическому языку, мы говорим о "построении" терма. Использование пресловутых троеточий в

$$pow(a,m) = \underbrace{(a\ o\ (a\ o\ (a\ o\ \ldots\ (a\ o\ a)\ \ldots)))}_{m},$$

возможно, уже не считается приемлемым, однако запись

$$pow(a,m) = \begin{cases} a & \text{если } m = 1; \\ a\ o\ pow(a,m-1) & \text{если } m > 1 \end{cases}$$

считается уже вполне стандартной. Алгоритмы в самом широком смысле задают именно такого рода описания, и их можно понимать как средства для порождения вычислительных термов и, стало быть, самих вычислений.

Сигнатуры и термы. Теперь мы можем перейти к общему определению алгоритма. При этом мы будем следовать определению абстрактных типов данных.

Определение. Сигнатура $\Sigma = (M_1, M_2, \ldots, M_m, O_1, O_2, \ldots, O_n)$ – это конечное семейство носителей M_μ вместе с конечным семейством операций O_ν, где каждый носитель M_μ есть символ для множества (малого множества в смысле Маклейна, 1971) и каждая операция O_ν есть символ для отображения некоторой фиксированной местности

$$O_\nu : M_{\mu_1} \times M_{\mu_2} \times \ldots \times M_{\mu_r} \to M_{\mu_o}.$$

Пример 1. $\Sigma = (M, .o.n, e)$,

где $.o. : M \times M \to M$, $n : \to M$, $e : \to M$.

Определение. Терм t в сигнатуре Σ – это правильное выражение, образованное свободными переменными и символами операций из Σ. Если операция, соответствующая заключительному сим-

волу операции в терме, есть отображение в носитель M_μ, то мы говорим, что терм ведет в M_μ.

$W_\Sigma(x)$ означает множество всех термов в сигнатуре Σ со свободными переменными из семейства переменных $X=(X_1, X_2, \ldots, X_m)$, каждая из которых может быть конкретизирована элементами из $(M_1, M_2, \ldots, M_m)$ соответственно. W_Σ – это сокращение для $W_\Sigma(\emptyset, \emptyset, \ldots, \emptyset)$.

Каждая гетерогенная алгебра имеет сигнатуру. Обычно одной сигнатуре соответствует много алгебр. Для сигнатуры примера I алгебра A_1 будет

$M = \{a\}$, $aoa = a$, $n = a$, $e = a$ (тривиальная алгебра);

примерами алгебр с другим кардиналом являются

A_2 : $M = \{n, e\}$, $n \circ n = n$,
$n \circ e = e$,
$e \circ e = n$,
$e \circ n = e$;

A_ω : $M=N$ (натуральный ряд) $a \circ b = a + b$, $n = 0$, $e = 1$;

A_K : $M= R$ (вещественные числа) $a \circ b = a + b$, $n = 0$, $e = 1$.

Не все алгебры интересны для нас. Во многих случаях мы рассматриваем иерархические сигнатуры, где определенное подмножество $C_{прим}$ носителей и операций полагается примитивным, а остальные носители оказываются определимыми. Мы говорим, что алгебра с иерархической сигнатурой Σ порождается множеством примитивных носителей $C_{прим}$, если все элементы других, определимых носителей $C_{опр}$ могут быть получены из термов над $C_{прим}$ конкретизациями. Алгебра называется вполне порожденной, если $C_{прим}$ пуст, так что все элементы получаются с помощью операций (включая "стартовые" нульместные операции). Алгебры A_1, A_2 и A_ω – вполне порожденные, а A_K таковой быть не может: никакое перечислимое множество термов не может породить R.

Мы назовем это ограничение принципом породимости и будем соблюдать его по всей статье. Мы также предполагаем, что примитивные носители являются порожденными некоторой более примитивной сигнатурой или (окончательно) вполне порожденными.

Одна из конкретных алгебр в сигнатуре Σ – это алгебры термов над $C_{прим}$: элементами определимых носителей $C_{опр}$ в этой

модели являются термы со свободными переменными и операциями, ведущими в определимые носители, а сами операции выполняются формально. Алгебра термов над $C_{прим}$ тривиально порождается из $C_{прим}$. Именно поэтому термы играют такую важную роль.

Алгебраическое определение алгоритмов

Определение. Вычислительный терм в произвольной сигнатуре Σ – это терм из $W_\Sigma = W_\Sigma(\emptyset)$.

Вычислительный терм может получаться из терма $t \in W_\Sigma(X)$ конкретизацией элементами из примитивного и/или определимых носителей.

Определение. Алгоритм над алгеброй в сигнатуре Σ – это правило (написанное конечным способом) для порождения множества вычислительных термов, в котором свободные переменные, фигурирующие в правиле, конкретизируются аргументами. Множество всех рассматриваемых конкретизаций образует область алгоритма.

Самые простые алгоритмы описываются (незамкнутыми) термами типа

$$a \circ (a \circ (a \circ (a \circ a))) .$$

Их можно назвать прямыми алгоритмами; для каждой конкретизации вычислительный терм немедленно выдает результат.

Другие алгоритмы могут приводить к прямым алгоритмам после частичной конкретизации (аналогичной "смешанным вычислениям" по Ершову, 1977), например, алгоритм pow (a, 4) выражается термом $a \circ (a \circ (a \circ a))$, причем все вычислительные термы алгоритма pow(a,m) могут быть описаны сначала конкретизацией m в pow(a, m) , а затем конкретизацией · в результирующих термах, которые уже суть прямые алгоритмы. Такие параметры (как a в pow) называются фиксированными параметрами (фиксированными по отношению к рекурсии).

В общем случае, однако, множество вычислительных термов, порождаемых алгоритмом, не может быть описано просто так, что сначала термы порождаются, а лишь затем конкретизируются; правила порождений могут быть весьма сложными. Заметим, что "вычисление" до сих пор не было определено.

Многие конкретные определения алгоритма подпадают под это определение; прежде всего, Марковские алгоритмы и машины Тьюринга, а также и рекурсивные функции, где вычислительные тер-

мы порождаются механизмом Эрбрана-Клини преобразования текста. Наше определение, однако, гораздо шире и только дополнительные условия вернут нас к классическим постановкам.

Прежде всего, термы, о которых мы говорим, не обязательно конечны; мы можем рассматривать термы с бесконечным числом вхождений символов операций и свободных переменных (Нива, 1975). Если раньше такие термы не вызывали интереса в теоретической информатике, поскольку их конкретизация ведет к бесконечному вычислению, то сейчас они приобретают значение (Бауэр, 1978), описывая определенные "бесконечные объекты", оказавшиеся подходящими для трактовки машинно-ориентированных реализаций указателей благодаря наблюдению Хендерсона и Морриса, 1976 о том, что "ленивое вычисление" (Фридман, Уайз, 1976) приводит к конечным термам, даже если обстановка образована бесконечными объектами.

Определение. Алгоритм называется совершенным, если он порождает только конечные вычислительные термы.

Для простоты мы ограничим наше рассмотрение только совершенными алгоритмами. Кроме этого, мы не будем уточнять, сколько вычислительных термов относятся к одной конкретизации.

Определение. Если для некоторой конкретизации не существует вычислительного терма, то мы скажем, что алгоритм не определен для этой конкретизации.

Ограничивая должным образом область алгоритма, мы всегда можем сделать его вполне определенным. Хотя работа с не вполне определенными алгоритмами имеет определенные преимущества (Манна, 1974), мы в последующем ограничим себя только вполне определенными алгоритмами.

Одна конкретизация может приводить к нескольким вычислительным термам.

Определение. Алгоритм называется детерминированным, если для каждой конкретизации порождается хотя бы один вычислительный терм, и регулярным, если для каждой конкретизации порождается по меньшей мере один вычислительный терм.

Классические алгоритмы детерминированы и регулярны.

Недетерминизм и типы. Недетерминированные алгоритмы, впервые рассмотренные Маккарти, 1962, имеют огромное значение. Чтобы это стало ясным, мы вспомним, что до сих пор алгоритм

определялся только в сигнатуре без учета конкретных свойств алгебры. Мы скорее определили схему алгоритма, которая должна, прежде чем подвергаться конкретизации, быть проинтерпретированной заданием определенной алгебры.

Свойства алгебры могут привести к тому, что одна и та же конкретизация разных термов приводит к одному и тому же результату, т.е. разные вычисления становятся функционально эквивалентными, приводя к одинаковым значениям. Может даже оказаться, что два разных терма дают одинаковые результаты для целого класса интерпретаций; тогда мы скажем, что термы функционально эквивалентны (по отношению к этому классу), и выразим это, уравняв эти термы, как, например,

$$a \circ (a \circ (a \circ a)) = (a \circ a) \circ (a \circ a),$$

что справедливо для любой интерпретации сигнатуры $\Sigma = (M, .o.)$, в которой имеет место закон ассоциативности. В общем случае, однако, равенства термов недостаточны для подходящего описания свойств алгебр и, по крайней мере, должны использоваться логические высказывания, конструируемые из таких равенств (например, потребовав справедливости высказывания $(n = e)$, мы можем в примере I исключить случай тривиальной алгебры). Логические высказывания считаются справедливыми для любой конкретизации своих свободных переменных.

Пример 2. Следующий алгоритм, описываемый рекурсивно в алгебре $\Sigma = (M, .o., N)$ посредством совместных охраняемых команд (Дейкстра, 1975), недетерминирован:

$$\text{pow}(a, m) = \begin{cases} a & \text{если } m = 1; \\ \text{pow}(a, u) \circ \text{pow}(\text{pow}(a,v)) & \text{если } u,v > 0,\ m = u + v; \\ \text{pow}(\text{pow}(a, p),\ q) & \text{если } p,q > 1,\ m = p \times q. \end{cases}$$

Например, при частичной конкретизации, pow(a, 4) генерирует следующие термы:

$a \circ (a \circ (a \circ a))$, $(a \circ a) \circ (a \circ a)$, $((a \circ a) \circ a) \circ a$.

Важность этого недетерминированного алгоритма в том, что для каждой конкретизации он генерирует множество всех вычислительных термов, эквивалентных при законе ассоциативности в сигнатуре $(M, .o.)$.

Определение. Тип (Σ, E) - это сигнатура Σ вместе с множеством E свойств термов t над Σ, где свойства - это высказывательные формулы над равенствами термов, справедливыми для всех

конкретизаций свободных переменных.

Определение. Алгебра типа (Σ, E) – это алгебра в сигнатуре Σ, удовлетворяющая свойствам E.

Определение. Вычислительная структура типа (Σ, E) – это алгебра типа (Σ, E), которая может быть порождена (некоторым множеством примитивных носителей $C_{прим}$).

Множество W_Σ всех термов в сигнатуре Σ – это конкретная алгебра типа $(\Sigma, \emptyset)$ сигнатуры Σ с пустым множеством свойств. Пара $(\Sigma, \emptyset)$ называется также абсолютно свободным типом. W_Σ-алгебра этого типа.

Определение. Σ- гомоморфизм между двумя алгебрами в сигнатуре Σ – это отображение, совместимое с операциями из Σ.

Определение. Интерпретация множества W_Σ в алгебре A типа (Σ, E) – это отображение, сопоставляющее каждому терму из A элемент из A, который получается "вычислением" терма в алгебре A, т.е. путем некоторого пошагового выполнения операций.

Любая интерпретация – это Σ гомоморфизм из W_Σ в A. Вычислительные структуры суть те алгебры, для которых интерпретация суръективна.

Теорема. Каждая вычислительная структура типа (Σ, E) есть Σ-эпиморфный образ множества W_Σ. Множество свойств E индуцирует отношение эквивалентности на вычислениях над Σ, совместимых с операциями Σ (отношение конгруентности). Каждая вычислительная структура типа (Σ, E) изоморфна остаточным классам по модулю E.

Определение. Алгоритм над Σ однозначен по отношению к типу (Σ, E), если любая конкретизация в алгоритме порождает вычислительные термы, эквивалентные при условиях E.

Детерминированный алгоритм, естественно, однозначен.

Пример. Алгоритм pow из примера 2, являясь недетерминированным, все же однозначен для любого типа (Σ, E), включающего ассоциативную бинарную операцию.

Из недетерминированных алгоритмов можно выводить их упрощения (Маккарти, 1962), в которых выбор варианта рекурсии более узок или даже однозначен. Упрощая вторую строку примера 2 случаем $u = 1$, $v = m-1$ при $m > 1$ и опуская третью строку, получим классическую рекурсию, показанную во введении. Другие, более эффективные детерминированные алгоритмы, также могут

быть легко получены.

Для данного типа могут существовать неизоморфные вычислительные структуры. Тип, в котором нет неизоморфных вычислительных структур, называется мономорфным.

Поскольку сам алгоритм задан только сигнатурой Σ, а свойства E влияют лишь на однозначность, мы можем рассматривать алгоритм не только для неизоморфных алгебр данного типа, но и для разных типов (Σ, E_1) и (Σ, E_2) с одинаковой сигнатурой, при условии, что каждая из E_1 и E_2 достаточно сильны, чтобы обеспечить однозначность.

Вычисление. Вычисление по алгоритму при заданной конкретизации определяется как порождение одного, нескольких или всех (если они существуют) вычислительных термов, генерируемых данной конкретизацией, и нахождение его (их) значения. Реальное выполнение такого вычисления не определяется. Остается большая свобода в определении подходящих машин, в том числе весьма абстрактных, с самыми разными правилами вычислений (см. например, Манна, 1974), в том числе и с параллельным выполнением (Брой, 1980).

Заключение. Алгебраическое определение алгоритма, данное выше, шире классического. Это абстрактное определение, которое базируется только на сигнатуре и интерпретируется любой вычислительной структурой этой сигнатуры. Даже задание набора свойств не обязательно задает интерпретацию полностью. Эта свобода дает преимущество, позволяя подвести под одну крышу целое семейство родственных алгоритмов. Более того, даже для данного мономорфного типа алгоритмы могут быть недетерминированными. Эта свобода позволяет переходить к специальным упрощениям ради повышения эффективности, включая детерминизм.

Обе степени свободы могут быть полезно использованы в процессе построения программы ради избежания преждевременных решений.

В частности, такое абстрактное определение алгоритмов (в сочетании с абстрактным определением типов и отношений конгруентности в алгебре термов, индуцирующих представления вычислительных структур данного типа) - это шаг к освобождению от давления нотации, заслоняющего некоторые важные сущности программирования.

Признательности. Идеи, очерченные в этой заметке, были стимулированы дискуссиями с Х.Вёсснером в 1977 г. во время совместной работы над книгой "Алгоритмические языки и разработка программ" (выходящей в издательстве "Шпрингер"). Благодарю также М.Броя, Б.Мёллера и М.Вирзинга за критические замечания.

ЛИТЕРАТУРА

Бауэр, 1978
Bauer F.L. Detailization and lazy evaluation, infinite objects and pointer representation. -In: Program Construction, Lecture Notes in Comp. Sci., 69, Springer, Berlin 1979.

Брой, 1980
Broy M. Transformation parallel ablaufender Programme. Dissertation, Technische Universität München, 1980.

Гуттаг, 1975
Guttag J.V. The specification and application to programming of abstract data types. TR CSRG-59, September 1975, University of Toronto.

Дейкстра, 1975
Dijkstra E.W. Guarded commands, nondeterminacy and formal derivation of programs. Comm. ACM, 18, p. 453-457 (1975).

Ершов, 1977
Ershov A.P. On the essence of compilation. Proc. IFIP Working Conf. on the Formal Description of Programming Concepts, North-Holland, Amsterdam 1978.

Маккарти, 1962
McCarthy J. Towards a mathematical science of computation. -In: Information Processing 1962, North-Holland, Amsterdam 1963, p. 21-28.

Маклейн, 1971
Mac Lane S. Categories for the working mathematician. Springer, New York 1971.

Манна, 1974
Manna Z. Mathematical theory of computation. McGraw-Hill, New York 1974.

Нива, 1975
Nivat M. On the interpretation of recursive program schemes. Symposia Mathematica, vol. XV, Istituto Nationale di Alta Matematica, 1975.

Скотт, 1970
Scott D. Outline of a mathematical theory of computation. Proc. 4th Princeton Conference on Information Sciences and Systems, 1970.

Фридман, Уайз, 1976
Friedman D.P., Wise D.S. CONS should not evaluate its arguments. -In: Automata, Languages and Programming, Proceedings 1976. Edinburg University Press, 1976, p. 257-284.

Хендерсон, Моррис, 1976
Henderson P. and Morris J.H. A lazy evaluator. Proc. 3rd ACM Symp. on Principles of Programming Languages (January 1976), Atlanta, p. 95-103.