УДК 519.3

Г.Е. ЦЕЙТЛИН

АЛГЕБРО-АЛГОРИТМИЧЕСКИЕ АСПЕКТЫ ПОЛНОТЫ: АБСТРАКЦИИ, БИОЛОГИЯ И ЭКОЛОГИЯ

Ключевые слова: *алгебра алгоритмики, операция суперпозиции, канонические формы, функциональная и аксиоматическая полнота, подалгебры (замкнутые классы), грамматики структурного проектирования.*

1. СОСТОЯНИЕ ИССЛЕДОВАНИЙ

Аксиоматическая полнота относится к числу фундаментальных проблем алгебры и логики. Ее значимость определяется известными теоремами Гёделя [1]. Важность этих теорем трудно переоценить особенно в эру компьютерных вычислений. В этой связи необходимо упомянуть, например, работу [2], где моделируется доказательство теоремы Гёделя о неполноте формальной арифметики с программистских позиций.

Настоящая статья посвящена рассмотрению проблематики аксиоматической полноты в аспекте предпринятых на Западе усилий по алгебраизации современного программирования [3, 4]. Следует отметить, что в Украине алгебраизация алгоритмов и программ стала актуальной, начиная с фундаментальных работ В.М. Глушкова [5, 6] и Л.А. Калужнина [7].

На рис. 1 показана связь алгебры алгоритмики $<\text{АА}>$ с алгебраической алгоритмикой (АА): алгебры и языка программирования (ЯП) АДА [3]. Сопоставительный анализ алгебраической алгоритмики (АА) с алгеброй алгоритмики $<\text{АА}>$ был проведен на УкрПрог-2004 [8]. Отметим, что сравнению $<\text{АА}>$ с ментальным программированием (IP) посвящены работы [9, 10].

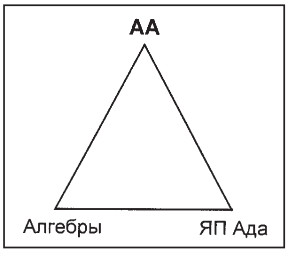

Рис. 1. Концепция алгебраической алгоритмики

Для IP характерна триада: абстракции, биология и экология знаний (рис. 2). Под абстракциями в IP понимаются формулы в различных алгебрах (иными словами, аналитическая форма представления знаний об алгоритмах). В $<\text{АА}>$ наряду с аналитической (в зависимости от методов проектирования) рассматриваются две другие эквивалентные формы — текстовая и графовая. Например, с системой алгоритмических алгебр (САА) [5], получившей название алгебры Глушкова (АГ), связано текстовое проектирование алгоритмов в языке САА-схем [11] и визуализация алгоритмов в терминах эквивалентных граф-схем [7] (более детально см. [12]). Следует отметить, что в $<\text{АА}>$ функциональная эквивалентность рассмотренных форм представления алгоритмов обеспечивается наличием соответствующих инструментальных средств [9, 25].

Рис. 2. Сопоставление алгебры алгоритмики и ментального программирования

К числу «вечных» проблем программирования относится обоснование правильности сконструированных алгоритмов и программ. Эффективным средством обоснования правильности служит преобразование (трансформация) алгоритмов, а также предложенные в < АА > формализованные метаправила: свертка (укрупнение), развертка (детализация), переинтерпретация (свертка–развертка) и трансформация схем алгоритмов.

К экологии программирования относятся инструментальные средства, автоматизирующие применение перечисленных метаправил [9, 10, 25]. Для алгебры в целом и для < АА > в частности особый интерес представляет метаправило трансформации в плане построения алгоритмов, удовлетворяющих необходимым критериям качества [13, 14]. Важным для процесса аксиоматизации любой алгебры является построение ее «законов»:

— свойств элементов, входящих в основу и/или основы (если алгебра многоосновная);

— свойств операций, входящих в сигнатуру рассматриваемой алгебры.

При этом характерные для логики конструкции (кванторы), их свойства, логические правила вывода расширяются за счет алгоритмической компоненты: метаправила вывода, структуры данных и памяти и т.д. В результате создается аппарат лемм и теорем для обеспечения возможности его приложений в различных актуальных предметных областях.

Таким образом, исследование аксиоматической полноты некоторой алгебры осуществимо с учетом возможных аналогий по отношению к функциональной полноте и ограничений, вытекающих из упомянутой теоремы Гёделя о неполноте формальной арифметики.

Структуру статьи составляют следующие разделы. В разд. 2 рассмотрена алгебра логики, или клон Поста (КП). Разд. 3 посвящен логическим и алгоритмическим клонам, а также клонам формальных языков. В разд. 4 рассмотрен алгебро-грамматический формализм грамматик структурного проектирования (обобщающий язык САА-схем). В разд. 5 интерпретируется биология и экология программирования.

2. АЛГЕБРА ЛОГИКИ И БУЛЕВЫ ФУНКЦИИ

Основной операцией теории клонов [23], классическим примером которой служит алгебра логики $L(2)$, является суперпозиция функций (операций) рассматриваемой алгебры. Для случая $L(2)$ такими операциями служат булевы функции (БФ). Их суперпозиция суть подстановка одной БФ вместо переменной в другую.

В рамках $L(2)$ известны знаменитая теорема Поста о функциональной полноте [6, 9, 10, 12, 13] и базирующийся на ней механизм построения различных алгебр БФ. В основу исследований Поста положено построение решетки подалгебр (замкнутых классов) $L(2)$. Мощность совокупности подалгебр алгебры $L(2)$ счетна, но количество различных типов подалгебр конечно.

Посредством теоремы Поста могут быть построены различные алгебры БФ: алгебра Буля, алгебра Жегалкина и др. Для каждой из этих и подобных алгебр, представляющих интерес для различных приложений, необходимо построить ее аксиоматику. Например, хорошо известна аксиоматика алгебры Буля, обобщением которой служит система аксиом булевых алгебр, имеющая ряд важных приложений в программировании, в частности в связи с разработкой баз данных и знаний. К классу булевых алгебр относится также общеизвестная алгебра множеств [12].

Особенность аксиоматизированной алгебры состоит в том, что ее аналитические представления (формулы) могут быть сведены к каноническому виду.

Так, в алгебре Буля любая формула сводима к СДНФ или СКНФ. Следствием этого факта служит разрешимость проблемы тождеств в этой алгебре. Именно для того, чтобы убедиться в справедливости тождества $f = g$ (f и g — булевы функции), необходимо его левую и правую части свести, например, к СДНФ. Отсюда следует и способ проверки на полноту аксиоматики A некоторой алгебры БФ: необходимо убедиться, что все аксиомы, входящие в A, не принадлежат ни одной из максимальных подалгебр теоремы Поста.

3. ПОЛУГРУППОВЫЕ КЛОНЫ И ПРИЛОЖЕНИЯ (ПАРАДИГМА КЛОНА ГЛУШКОВА)

Рассмотрим полугрупповые клоны, их отличие и связь с КП. Как и в [22],

лингвистическая дедуктивная теория T / L с алфавитом $L = a_1, a_2, \ldots, a_k$ состоит из букв a_i, где $i = 1, 2, \ldots, k$; T / L_1 — элементарная словарная теория с однобуквенным алфавитом $L_1 = a$. Здесь мы назовем словарной полугрупповой алгеброй элементарную теорию T / L_1 с операцией умножения (конкатенации) слов $s_1 * s_2 = s_1 s_2$. Например, если $s_1 = aa$ и $s_2 = aaa$, то $s_1 * s_2 = aaaaa$.

На основе элементарной полугрупповой алгебры с операцией * построим элементарный полугрупповой клон $C(T / L_1) = < T / L_1; \mathrm{SUPER} >$. Рассмотрим множество слов $M / P = S / P$, длины которых суть простые числа. Пусть A — элементарная полугрупповая алгебра с операцией * и $C(T / L_1) =< T / L_1; \mathrm{SUPER} >$ — элементарный полугрупповой клон. Отметим, что множество всех подмножеств счетного множества имеет мощность континуума. Справедливо следующее утверждение.

Теорема 1. Множество M / P является системой образующих (СО) клона $C(T / L_1)$.

Представительной назовем элементарную полугрупповую алгебру A (с операцией *), по которой строится клон C / A.

Следствие. Клон $C / A =< A; \mathrm{SUPER} >$ является клоном континуального типа, где A — представительная алгебра.

Приведем построенные в [5] пары (рис. 3): представительная алгебра и соответствующий ей алгоритмический клон. Каждой паре соответствует определенный метод и технология программирования.

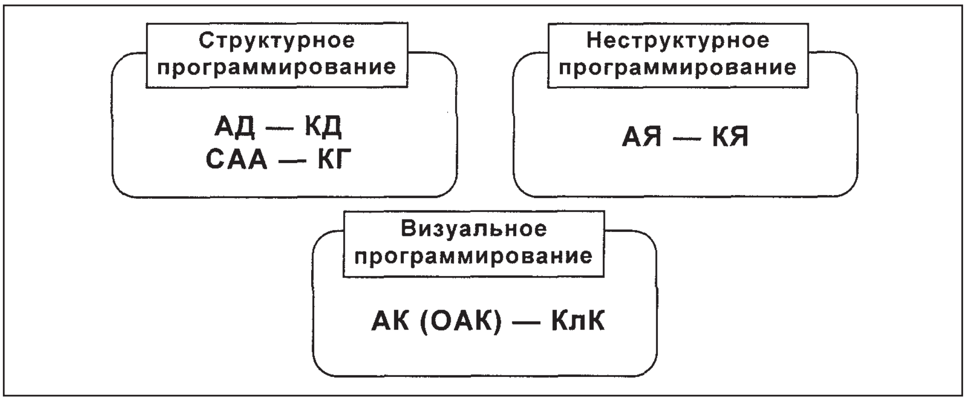

Рис. 3. Классификация клонов

Структурное программирование: алгебра Дейкстры (АД) — клон Дейкстры (КД); системы алгоритмических алгебр (САА) Глушкова — клон Глушкова (КГ).

Неструктурное программирование: алгебра Янова (АЯ) — клон Янова (КЯ).

Визуальное программирование: алгебра граф-схем Калужнина (АК) — клон Калужнина (КлК) (обобщенные АК (ОАК) — КлК).

Прокомментируем приведенные пары. Каждая представительная алгебра является двухосновной. Логическая основа состоит из булевых операций и операции прогнозирования (наличие памяти) в случае САА [6]. Таким образом, КД, КлК и КЯ в качестве логической компоненты имеют КП. Проблема функциональной полноты для перечисленных клонов решена в [6] с учетом наличия операции прогнозирования в КГ, что и объясняет изобразительную мощность САА и КГ в сравнении с другими клонами. Отметим возможность перехода к k-значным логикам в САА, что важно для многозначных средств представления знаний (неопределенность, таблицы решений и пр.). В [5] рассмотрен также полугрупповой клон Клини, относящийся к клонам формальных языков и грамматик.

4. ЗАМКНУТОСТЬ УСЛОВИЙ. СТАНДАРТНЫЕ ПАРАЛЛЕЛЬНЫЕ ПОЛИНОМЫ

В [14] построена аксиоматика для САА, базирующихся на понятии замкнутых условий (адекватной предложенной А.А. Летичевским концепции монотонного оператора [15]). Суть понятия замкнутого условия состоит в том, что, став истинным, оно сохраняется вплоть до окончания вычислительного процесса (независимо от состояния обрабатываемых данных) ввиду применения операторов, возможно, влияющих на такое состояние. В этом смысле замкнутые условия фиксируют некоторые вехи, характеризующие процесс вычислений.

Например, если речь идет о семействе задач в смеси некоторой операционной системы, то факт решения одной из этих задач можно считать окончательным независимо от состояния других задач в рассматриваемой смеси. Замкнутость весьма важна для организации взаимодействия параллельных ветвей при асинхронной (распределенной) обработке данных. С этой целью в аппарат САА были введены:

— синхронизаторы, обеспечивающие ожидание выполнения замкнутых условий по параллельным ветвям;

— контрольные точки (КТ), в которых осуществляются проверки значений указанных условий;

— фильтры $\Phi(u\ )$, в терминах которых описаны свойства ряда операций, входящих в сигнатуры алгоритмических алгебр.

Предложенная аксиоматика является полной. Таким образом, спроектированный параллельный алгоритм в результате преобразований, базирующихся на разработанной аксиоматике, сводим к канонической форме — стандартному параллельному полиному (СПП).

Таким образом, если имеется гипотеза в виде равенства $f = g$, то, преобразовав левую и правую части данного равенства к СПП (в случае совпадения — с точностью до перестановки параллельных ветвей), получим доказательство справедливости указанной гипотезы.

Естественным обобщением понятия замкнутого условия служит квазизамкнутость, суть которой состоит в допущении возможности сбрасывания значения истины на ложь при повторном вычислении очередного витка цикла. Как известно, основной объем вычислений сосредоточен при выполнении циклов — отсюда и сложность их распараллеливания. С понятием квазизамкнутости связана иерархия вложенных циклов, когда при вычислении тела основного цикла его условие, став однажды истинным, уже не меняет этого значения до очередного витка основного цикла.

Это относится и к циклам, вложенным в основной цикл. При этом некоторые циклические условия связаны импликациями, определяющими иерархию управления вычислений основного цикла [19].

5. ГРАММАТИКИ СТРУКТУРНОГО ПРОЕКТИРОВАНИЯ. МАКРОКОНТЕЙНЕРЫ

В рамках $< АА >$ предложены базирующиеся на аппарате АГ формализованные средства проектирования и синтеза алгоритмов и программ.

Построенный формализм сопряжен с применением подстановок вида

$$СИД :: = \text{спецификация}, \qquad (1)$$

где СИД — имя семантического идентификатора (оператора или предиката), подлежащего спецификации в АГ — правой части подстановки (1).

1. Параллельные вычисления. Матричная мультиобработка. Рассмотрим метод синтеза алгоритмов и программ в $< АА >$, базирующийся на сочетании аппарата $< АА >$ с механизмами управления выводом, которые характерны для теории формальных грамматик [18]. Предлагаемые системы были названы грамматиками структурного проектирования (ГСП). Процесс проектирования в ГСП базируется на применении подстановок вида (1) с целью детализации или укрупнения схем в рамках нисходящей, восходящей или комбинированной стратегии проектирования.

Под ГСП понимается интегрированный алгебро-грамматический аппарат, сочетающий определенную алгебру алгоритмов и соответствующий класс грамматик. Настройка на необходимую предметную область осуществляется посредством контейнеров — семантических идентификаторов, определяющих элементарные предикаты и операторы, на базе которых осуществляется синтез программ.

Следует отметить, что при необходимости могут быть использованы и макроконтейнеры, которые, в свою очередь, могут быть достаточно крупными алгоритмами, входящими в состав проектируемого алгоритмического комплекса. Например, в качестве макроконтейнера может быть использован такой адаптивный алгоритм Дейкстры, как ЧЕЛНОК (или прямые вставки в терминологии Кнута [20]).

ГСП содержит систему обобщенных продукций, каждая из которых суть конечная совокупность подстановок, имеющих, как правило, левую и правую части. Подстановки в обобщенных продукциях могут выполняться последовательно или параллельно. В свою очередь, сами обобщенные продукции могут также применяться в указанных режимах к уже полученным строкам вывода. ГСП представимы в матричной форме, где строки — обобщенные продукции, столбцы — подстановки. В частности, матрицами представимы традиционные формальные языки и грамматики, например в классификации Хомского [18].

В [21] построены матричные ГСП, порождающие классы асинхронных схем алгоритмов символьной мультиобработки: сортировки, поиск, языковое процессирование [10, 12]. Порождение формальных языков посредством грамматических моделей сопряжено с построением полугрупповых клонов (более подробно см. [13, 23]).

2. **Метаправила вывода схем алгоритмов**. Проектирование алгоритмов в ГСП связано с применением метаправил конструирования схем алгоритмов (свертки, развертки, переинтерпретации, трансформации и др. [10, 12]). С помощью метаправил осуществляется переход от одних алгоритмов к другим, а также порождение новых алгоритмических знаний в рассматриваемой предметной области.

Для алгебраического подхода большое значение имеет применение метаправила трансформации — преобразования схем путем применения соотношений. В каждом соотношении левая и правая части отражают свойства операций, входящих в сигнатуру рассматриваемой алгебры. Отметим важность теории клонов, классификация которых была проведена в [23] с использованием понятия представляющей алгебры, восходящей к известной статье А. Успенского [22].

6. ЭКОЛОГИЯ ПРОГРАММИРОВАНИЯ (ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА)

В [10, 23] разработан интегрированный инструментарий проектирования и синтеза (ИПС) алгоритмов и программ. Данный инструментарий восходит к синтезатору МУЛЬТИПРОЦЕССИСТ [6, 11, 12] — одной из первых украинских CASE-систем, ориентированной на генерацию программ. Система также может быть отнесена к прототипу ментального программирования. В ее основу положен метод многоуровневого структурного проектирования программ по их описаниям в языке САА/1.

Важно отметить, что алгебраические средства проектирования алгоритмов и структур данных соответствуют принятому в объектно-ориентированных средах методу конструирования объектов в терминах шаблонов. При этом поуровневое присвоение логическим и операторным переменным их интерпретаций суть заполнение полей шаблонов в рамках объектных сред.

Следует также отметить возможность использования алгебраических преобразований схем на основе свойств алгебраических операций, входящих в формулы, представляющие проектируемые объекты.

Таким образом, к числу важных открытых проблем <АА> относятся следующие.

1. Построение поверхности клонов континуального типа и исследование пограничной зоны, ограничивающей данную поверхность.

2. Создание эффективных алгебро-грамматических систем, ориентированных на структурное проектирование и другие методы описания алгоритмов и программ, включая мультиобработку в современных вычислительных средах.

3. Создание для актуальных предметных областей соответствующих баз знаний и эффективных алгоритмов поиска в указанных базах.

4. Развитие архитектуры инструментария проектирования и синтеза алгоритмов и программ с учетом специфики создаваемых актуальных предметных областей.

5. Адаптация развиваемых теоретических и прикладных средств для важной гуманитарной сферы к группам лиц с различными физическими ограничениями, включая и наиболее уязвимые их категории (в частности, с проблемами зрения).

СПИСОК ЛИТЕРАТУРЫ

1. Новиков П.С. Элементы математической логики. — М.: Наука, 1973. — 400 с.
2. Глушков В.М. Теорема о неполноте формальных теорий с позиций программиста // Кибернетика. — 1979. — № 2. — С. 1–5.
3. Ноден П., Китте К. Алгебраическая алгоритмика. — М.: Мир, 1999. — 720 с.
4. Чарнецки К., Айзенекер У. Порождающее программирование: методы, инструменты, применение. Для профессионалов. — СПб.: Питер, 2005. — 731 с.
5. Глушков В.М. Теория автоматов и формальные преобразования микропрограмм // Кибернетика. — 1965. — № 5. — С. 1–10.
6. Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л. Алгебра. Языки. Программирование. — К.: Наук. думка. — 1-е изд., 1974. — 327 с.; 2-е изд., перераб., 1978. — 318 с.; 3-е изд., перераб. и доп., 1989. — 376 с.
7. Калужнин Л.А. Об алгоритмизации математических задач // Пробл. кибернетики. — 1959. — Вып. 2 — С. 51–69.
8. Цейтлин Г.Е., Мохница А.С. Что такое алгебраическая алгоритмика? // Пробл. программирования. Спецвыпуск по материалам 4-й Междунар. науч.-практ. конф. по программированию УкрПРОГ’2004. — К.: ИПС НАН Украины, 2004. — № 2–3. — С. 52–57.
9. Дорошенко А.Е., Захария Л.М., Цейтлин Г.Е. Алгебраическое проектирование программ: алгоритмы, объекты, инструменты // Проблемы

программирования. — 2007. — № 2. — С. 5–14.

10. Андон Ф.И., Дорошенко А.Е., Цейтлин Г.Е., Яценко Е.А. Алгеброалгоритмические модели и методы параллельного программирования. — К.: Академпериодика, 2007. — 634 с.

11. Грицай В.П., Цейтлин Г.Е. Некоторые вопросы автоматизации структурного параллельного программирования // Кибернетика. — 1979. — № 1. — С. 106–111.

12. Цейтлин Г.Е. Введение в алгоритмику — К.: Сфера, 1999. — 720 с.

13. Szendrei A. Clones in universal algebra, seminaire de mathematiques superieures. Vol. 99. — Montreal: Les Presses de l'Universite de Montreal, 1986.

14. Цейтлин Г.Е. Проблема тождественных преобразований схем структурированных программ с замкнутыми логическими условиями. Ч. 1–3 // Кибернетика. — 1978. — № 3. — С. 50–57; № 4. — С. 10–18; 1979. — № 5. — С. 44–51.

15. Летичевский А.А. Об ускорении итераций монотонных операторов // Кибернетика. — 1976. — № 4. — С. 1–7.

16. Цейтлин Г.Е. Формальная трансформация структурированных алгоритмов сортировки // Программирование. — 1985. — № 2. — С. 79–91.

17. Цейтлин Г.Е. Проектирование алгоритмов параллельной сортировки // Там же. — 1989. — № 6. — С. 4–19.

18. Chomsky N. Formal properties of grammars // In «Handbook of Mathematical Psychology». — 1963. — **2**. — New York: Wiley. — P. 323–418.

19. Цейтлин Г.Е. Трансформационная сводимость и синтез алгоритмов и программ символьной обработки // Кибернетика и системный анализ. — 2006. — № 5. — С. 165–174.

20. Кнут Д. Искусство программирования для ЭВМ, т.3. — М.: Мир, 1978. — 843 с.

21. Цейтлин Г.Е. Алгоритмы символьной обработки: объектная ориентация, трансформация, синтез // Смешанные вычисления и преобразование программ. — Новосибирск: ВЦ СО АН СССР, 1991. — С. 182–199.

22. Uspensky A. Gödel's incompleteness theorem / Theoretical Computer Science. — 1994.— **130**, N 2. — P. 239–319.

23. Цейтлин Г.Е. Алгебра Глушкова и теория клонов // Кибернетика и системный анализ. — 2003. — № 4 — С. 48–58.

24. Doroshenko A., Tseytlin G., Yatsenko O., Zachariya L. Intentional aspects of algebra of algorithmics. — Proceedings of International Workshop «Concurrency, Specification and Programming» (CS&P'2007), 27–29 September 2007, Lagow (Poland). — 2007.

25. Яценко Е.А., Мохница А.С. Инструментальные средства конструирования синтаксически правильных параллельных алгоритмов и программ // Пробл. программирования. Спецвыпуск по материалам 4-й Междунар. науч.-практ. конф. по программированию УкрПРОГ'2004. — 2004. — № 2–3. — С. 444–450.

26. Капитонова Ю.В., Летичевский А.А. Парадигмы и идеи академика В.М. Глушкова. — К.: Наук. думка, 2003. — 330 с.

Поступила 14.01.2008