УДК 004.421

АЛГЕБРОАЛГОРИТМИЧЕСКИЕ АСПЕКТЫ ПОЛНОТЫ: АБСТРАКЦИИ, БИОЛОГИЯ И ЭКОЛОГИЯ

Г.Е. Цейтлин, Л.М. Захария

Институт программных систем НАН Украины,
03680, Киев, проспект Академика Глушкова, 40.
Тел.: 526 1538, e-mail: tseytlin@vikno.relc.com

Проведен сопоставительный анализ алгебры алгоритмики и аналогичных направлений – алгебраической алгоритмики и порождающего (ментального) программирования. Отражены общность и специфика проблематики функциональной и аксиоматической полноты. С позиций алгебры алгоритмики трактуются понятия: абстракции, биологии и экологии программистских знаний.

The comparative analysis of algebra of algorithmics and similar directions – algebraic algorithmics and generative (intensional) programming is conducted. Common and specific problems in functional and axiomatic completeness are represented. Algebraic-algorithmic treatment of concepts of abstraction, biology and ecology of programming knowledge is proposed.

Введение

Аксиоматическая полнота относится к числу фундаментальных проблем алгебры и логики. Её значимость определяется известными теоремами Гёделя [1]. Важность данных теорем трудно переоценить, особенно в эру компьютерных вычислений. При этом необходимо упомянуть, например, работу [2] по моделированию доказательства теоремы Гёделя о неполноте формальной арифметики с программистских позиций.

Работа посвящена рассмотрению проблематики аксиоматической полноты в аспекте предпринятых на Западе усилий по алгебраизации современного программирования [3, 4].

В целом необходимо подчеркнуть, что алгебраизация алгоритмов и программ в Украине стала актуальной начиная с фундаментальных работ В.М. Глушкова [5, 6, 7] и Л.А. Калужнина [8].

Сопоставительный анализ алгебраической алгоритмики (АА, рис. 1) с алгеброй алгоритмики <АА> проведен на УкрПрог-2004 [9], при этом сравнению <АА> с IP посвящены [10–12].

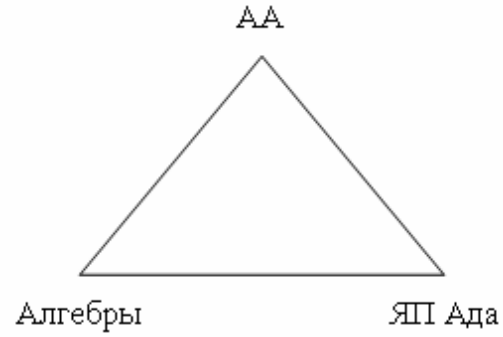

Рис. 1. Алгебраическая алгоритмика

Отметим, что для IP (рис. 2) характерна триада: абстракции, биология и экология знаний. Под абстракциями в IP понимаются формулы в различных алгебрах (иными словами, аналитическая форма представления знаний об алгоритмах). В <АА>, наряду с аналитической, (в зависимости от методов проектирования) рассматриваются и 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 [13] и визуализация алгоритмов в терминах эквивалентных граф-схем [8], более детально рассматриваемые в работе [14].

Следует отметить, что в <АА> функциональная эквивалентность рассмотренных форм представления алгоритмов обеспечивается наличием соответствующих инструментальных средств [10, 13].

К числу “вечных” проблем программирования относится обоснование правильности сконструированных алгоритмов и программ. Мощным средством обоснования правильности служит преобразование (трансформация) алгоритмов, а также предложенные в <АА> формализованные метаправила: свёртка (укрупнение), развёртка (детализация), переинтерпретация (свёртка-развёртка) и трансформация схем алгоритмов.

Рис. 2. Алгебра алгоритмики и ментальное программирование

В состав инструментария входят также инструменты автоматизирующие применение перечисленных метаправил [10, 11, 13]. Для алгебры вообще, и для <АА> в частности, особый интерес представляет метаправило трансформации в плане построения алгоритмов, удовлетворяющих необходимым критериям качества [14, 15].

Важная компонента аксиоматизации любой алгебры состоит в построении её "законодательства" - свойств:

– элементов, входящих в основу и/или основы (если алгебра многоосновна);

– операций, входящих в сигнатуру рассматриваемой алгебры.

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

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

Структуру работы составляют разделы. В разделе 1 рассмотрена алгебра логики, или клон Поста (КП). Раздел 2 посвящён логическим и алгоритмическим клонам, а также клонам формальных языков. В разделе 3 рассмотрен алгебро-грамматический формализм грамматик структурного проектирования (обобщающий язык САА-схем). В разделе 4 интерпретируются биология и экология программирования.

1. Алгебра логики и булевы функции

Основной операцией теории клонов [18, 19], 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 $L(2)$ известна знаменитая теорема Поста о функциональной полноте [6, 10, 11, 14, 18] и базирующийся на ней механизм построения различных алгебр БФ. В основу исследований Поста положено построение решётки подалгебр (замкнутых классов) $L(2)$. Мощность совокупности подалгебр $L(2)$ счетна, но количество различных типов подалгебр конечно.

Посредством теоремы Поста могут быть построены различные алгебры БФ: Буля, Жегалкина и др. Для каждой из данных и подобных алгебр, представляющих интерес для различных приложений, необходимо построить её 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. Например, к классу булевых алгебр относится общеизвестная алгебра множеств [14].

Особенность аксиоматизированной алгебры состоит в том, что её аналитические представления (формулы) могут быть сведены к каноническому виду. Так, в алгебре Буля любая формула сводима к СДНФ или СКНФ.

Следствием данного факта служит разрешимость проблемы тождеств в данной алгебре, именно, для того чтобы убедиться в справедливости тождества $f = g$, необходимо его левую и правую части свести, например к СДНФ. Отсюда следует способ проверки на полноту аксиоматики А некоторой алгебры БФ - необходимо убедиться в том, что все аксиомы, входящие в А не принадлежат ни одной из максимальных подалгебр теоремы Поста.

2. Полугрупповые клоны и приложения (парадигма клона Глушкова)

В данном разделе рассматриваются полугрупповые клоны, их отличие и связь с КП.

Аналогично Успенскому [20], рассмотрим лингвистическую дедуктивную теорию T/L с алфавитом $L = a_1, a_2, ..., a_k$, состоящим из букв a_i, где $i = 1, 2, ..., k$.

Пусть T/L_1 - элементарная словарная теория с однобуквенным алфавитом $L_1 = a$. Назовём словарной полугрупповой алгеброй элементарную теорию T/L_1 с операцией умножения (конкатенации) слов $s_1 * s_2 = s_1 s_2$. Например, если $s_1 = aa$, $s_2 = aaa$, то $s_1 * s_2 = aaaaa$.

Построим на основе элементарной полугрупповой алгебры с операцией * элементарный полугрупповой клон $C(T/L_1) = <T/L_1; SUPER>$. Рассмотрим множество слов $M/P = S/P$, состоящее из всех слов, длины которых суть простые числа.

Пусть A - элементарная полугрупповая алгебра с операцией * и $C(T/L_1) = <T/L_1; SUPER>$ - элементарный полугрупповой клон. Напоминаем, что множество всех подмножеств счётного множества имеет мощность континуума. Справедливо следующее утверждение.

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

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

Следствие. Клон $C/A = <A; SUPER>$ является клоном континуального типа, где A - представительная алгебра.

Приведём построенные в работе [18] пары (рис. 3): представительная алгебра и соответствующий ей алгоритмический клон. Подчеркнём, что каждой паре соответствует определённый метод и технология программирования.

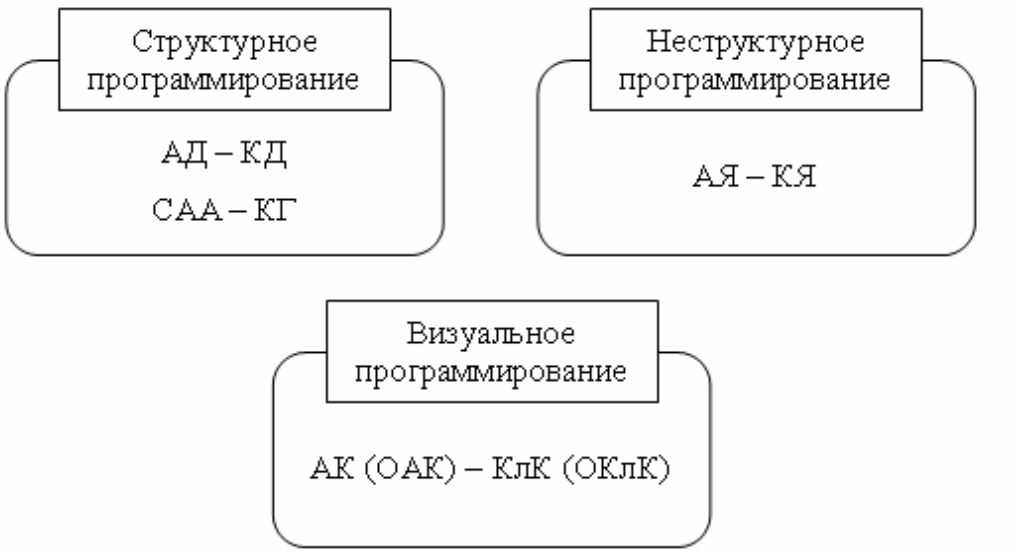

Рис. 3. Классификация клонов

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) (обобщённые АК – КлК).

Прокомментируем приведенные пары. Каждая представительная алгебра является двухосновной. Логическая основа состоит из булевых операций и операции прогнозирования (наличие памяти) в случае САА [6]. Тем самым, КД, КлК и КЯ в качестве логической компоненты имеют КП. Проблема функциональной полноты для перечисленных клонов решена в работе [6] с учётом наличия операции прогнозирования в КГ, что и объясняет изобразительную мощь САА и клона КГ в сравнении с другими клонами. Отметим возможность перехода к k-значным логикам в САА, что важно для представления многозначного знания (неопределённость, таблицы решений и пр.). В работе [18] рассмотрен также полугрупповой клон Клини, относящийся к клонам формальных языков и грамматик.

3. Замкнутость условий, стандартные параллельные полиномы

В работе [16] построена аксиоматика для САА, базирующихся на понятии замкнутых условий (адекватной предложенной А.А. Летичевским концепции монотонного оператора [21]). Суть понятия замкнутого условия состоит в том, что став однажды истинным, оно сохраняет данное значение вплоть до конца вычислительного

процесса, независимо от состояния данных, в результате функционирования операторов, возможно влияющих на данное состояние. При этом замкнутые условия фиксируют некоторые вехи, характеризующие процесс вычислений.

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

– синхронизаторы, реализующие ожидание выполнения замкнутых условий по параллельным ветвям;

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

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

Предложенная аксиоматика - полна так, что спроектированный параллельный алгоритм в результате базирующихся на разработанной аксиоматике преобразований, сводим к канонической форме - стандартному параллельному полиному (СПП).

Таким образом, если имеем гипотезу 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, основная масса вычислений сосредоточенна при выполнении циклов, отсюда и сложность их распараллеливания. С понятием квазизамкнутости связана иерархия вложенных циклов так, что при вычислении тела основного цикла, его условие став однажды истинным уже не меняет данного значения до очередного витка основного цикла.

Это обстоятельство выполняется и для циклов, вложенных в основной цикл. При этом некоторые циклические условия связаны импликациями, определяющими иерархию управления вычислений основного цикла [22].

4. Грамматики структурного проектирования, макро-контейнеры

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

Построенный формализм сопряжён с применением подстановок вида

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

где СИД – имя семантического идентификатора (оператора или предиката), подлежащего спецификации в АГ - правая часть подстановки (1).

4.1. Параллельные вычисления, матричная мультиобработка. Рассмотрим еще один метод синтеза алгоритмов и программ в <АА>, базирующийся на сочетании аппарата <АА> с механизмами управления выводом, характерными для теории формальных грамматик [23]. Предлагаемый формализм был назван 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 стратегий проектирования.

Под ГСП понимается интегрированный алгебро-грамматический аппарат, сочетающий ту, или иную 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 подчеркнуть, что при необходимости могут быть использованы и макро-контейнеры, которые, в свою очередь, могут быть достаточно крупными алгоритмами, входящими в состав проектируемого алгоритмического комплекса. Например, в качестве макро-контейнера может быть использован такой адаптивный алгоритм Дейкстры, как ЧЕЛНОК (или прямые вставки в терминологии Кнута [24]).

ГСП состоит из системы обобщённых продукций, каждая из которых является конечной совокупностью подстановок, имеющих, как обычно, левую и правую части. Подстановки в продукциях могут применяться последовательно или параллельно. В свою очередь, сами обобщённые продукции могут также применяться в указанных режимах к уже полученным строкам вывода.

ГСП представимы в матричной форме, где строки – обобщённые продукции, столбцы – подстановки. В частности, матрицами представимы традиционные формальные языки и грамматики, например в классификации Хомского [23].

В работе [25] построены матричные ГСП, порождающие классы асинхронных схем алгоритмов символьной мультиобработки: сортировка, поиск, языковое процессирование [11, 14, 26]. Порождение формальных языков посредством грамматических моделей сопряжено с построением полугрупповых клонов (более подробно рассматриваемых в работах [18, 19]).

4.2. Метаправила вывода схем алгоритмов. Проектирование алгоритмов в ГСП связано с применением метаправил конструирования схем алгоритмов (свертки, развертки, переинтерпретации, трансформации и др.

[11, 14]). 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 - преобразование схем в результате применения соотношений. Каждое соотношение состоит из левой и правой частей, отражающих свойства операций, входящих в сигнатуру рассматриваемой алгебры. Отметим важность теории клонов, классификация которых проведена в работе [18] с использованием понятия представляющей алгебры, восходящей к известной работе Успенского [20].

5. Экология программирования (инструментальные средства)

В работах [11, 15] разработан интегрированный инструментарий проектирования и синтеза алгоритмов и программ (ИПС). Данный инструментарий восходит к синтезатору МУЛЬТИПРОЦЕССИСТ [6, 13, 14] – одной из первых украинских CASE-систем, ориентированной на генерацию программ. Система также может быть отнесена к прототипу ментального программирования. В её основу положен метод многоуровневого структурного проектирования программ (МСПП) по их описаниям в языке САА/1.

Важно отметить также, 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 является заполнением полей шаблонов в рамках объектных сред.

Здесь следует отметить возможность использования алгебраических преобразований схем на базе свойств алгебраических операций, входящих в формулы, представляющие проектируемые объекты.

Заключение

К числу важных открытых проблем <АА> относятся.

1. Построение поверхности клонов континуального типа и исследование пограничной зоны, ограничивающей данную поверхность, служащую естественным обобщением диаграммы Поста для двухзначной алгебры логики. Подалгебры, входящие в диаграмму имеют конечный базис, а вся диаграмма состоит из конечной совокупности таких подалгебр. Это означает, что диаграмма Поста лежит на поверхности алгебры логики. Обобщение состоит в том, что рассматриваются поверхности клонов континуального типа. Алгебры, входящие в пограничную зону таких клонов содержат бесконечно порождённые подалгебры (со счётным базисом и не имеющие базиса). Для клонов континуального типа возникает весьма нетривиальная проблема исследования пограничной зоны, окаймляющей поверхность и включающей бесконечно порождённые подалгебры. Необходимо исследовать структуру указанной пограничной зоны.

2. Разработка развитых алгебро-грамматических средств, ориентированных на проектирование (не только структурное), но и другие методы разработки алгоритмов и программ (включая и параллельные) в современных вычислительных средах. Подобные исследования особенно важны для различных Web-приложений, формализации их семантики, организации взаимодействия параллельных ветвей при синхронной, асинхронной и распределённой мультиобработке, в частности, в циклах (где, как известно, сосредоточена основная масса вычислений).

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

Подобные базы могут содержать общую часть, в которую входят не интерпретированные и частично интерпретированные стратегии (схемы) обработки данных и знаний, относящиеся к различным предметным областям, а также метаправила, ориентированные на интеграцию рассматриваемых областей. В состав развиваемых баз входят средства работы с данными базами (эффективные методы поиска необходимых знаний, контейнеры для генерации (синтеза) конструируемых алгоритмов и программ и пр.

4. Продолжение разработки архитектуры инструментария проектирования и синтеза алгоритмов и программ с учётом специфики создаваемых актуальных предметных областей.

Наряду с инструментами синтеза конструируемых объектов, в состав инструментария должны входить модели, обобщающие диалоговые средства сборки правильных объектов, их трансформации (преобразования), средства построения простейших реляционных баз данных и знаний, а также подключения к известным (более сложным) корпоративным базам.

5. Адаптация развиваемых теоретических и прикладных средств для важной гуманитарной сферы – групп лиц с различными физическими ограничениями, включая и наиболее уязвимые их категории (в частности, с проблемами зрения). Здесь имеются в виду средства озвученного и тактильного восприятия информации, а также интегрированные системы восприятия, например, скрин-ридеры (подобные системе JAWS), синтезаторы речи, прежде всего, для отечественных языков. Решением проблемы адаптации могут быть методы конструирования специальных страниц в Интернете, обработки уже имеющихся страниц для их удобства лиц с определёнными физическими ограничениями, создания специальных методов обучения различных категорий инвалидов и пр.

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. – Киев: Наук. думка. – 1974. – 1-е изд. – 327 с.; 1978. – 2-е изд., перераб. – 318 с.; 1989. – 3-е изд., перераб. и доп. – 376 с.
7. *Капитонова Ю.В., Летичевский А.А.* Парадигмы и идеи академика В.М. Глушкова. – Киев: Наук. думка, 2003. – 330 с.
8. *Калужнин Л.А.* Об алгоритмизации математических задач // Проблемы кибернетики. – 1959. – Вып. 2. – С. 51–69.
9. *Цейтлин Г.Е., Мохница А.С.* Что такое алгебраическая алгоритмика? // Проблемы программирования. Спец. вып. по материалам 4-й Междунар. научн.-практ. конф. по программированию УкрПРОГ'2004. – Киев: Институт программных систем НАН Украины, 2004. – № 2–3. – С. 52–57.
10. *Дорошенко А.Е., Захария Л.М., Цейтлин Г.Е.* Алгебраическое проектирование программ: алгоритмы, объекты, инструменты // Проблемы программирования. – 2007. – № 2. – С. 5–14.
11. *Андон Ф.И., Дорошенко А.Е., Цейтлин Г.Е., Яценко Е.А.* Алгеброалгоритмические модели и методы параллельного программирования. – Киев: Академпериодика, 2007. – 631 с.
12. *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.
13. *Грицай В.П., Цейтлин Г.Е.* Некоторые вопросы автоматизации структурного параллельного программирования // Кибернетика. – 1979. – № 1. – С. 106–111.
14. *Цейтлин Г.Е.* Введение в алгоритмику. – Киев: Сфера, 1999. – 720 с.
15. *Яценко Е.А., Мохница А.С.* Инструментальные средства конструирования синтаксически правильных параллельных алгоритмов и программ // Проблемы программирования. Спец. вып. по материалам 4-й Междунар. научн.-практ. конф. по программированию УкрПРОГ'2004. – Киев: Институт программных систем НАН Украины, 2004. – № 2–3. – С. 444–450.
16. *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.
17. *Цейтлин Г.Е.* Формальная трансформация структурированных алгоритмов сортировки // Программирование. – 1985. – № 2. – С. 79–91.
18. *Цейтлин Г.Е.* Алгебра Глушкова и теория клонов // Кибернетика и системный анализ. – 2003. – № 4 – С. 48–58.
19. *Szendrei A.* Clones In Universal Algebra, Seminaire de Mathematiques Superieures, Les Presses de l'Universite de Montreal. – Montreal, 1986. – Vol. 99.
20. *Uspensky A.* Gödel's Incompleteness Theorem. Theoretical Computer Science, 1994, Vol. 130. – N 2. – P. 239–319.
21. *Летичевский А.А.* Об ускорении итераций монотонных операторов // Кибернетика. – 1976. – № 4. – С. 1–7.
22. *Цейтлин Г.Е.* Трансформационная сводимость и синтез алгоритмов и программ символьной обработки // Кибернетика и системный анализ. – 2006. – № 5. – С. 165–174.
23. *Chomsky, N.* Formal Properties of Grammars. In "Handbook of Mathematical Psychology". – Wiley: New York, 1963. – Vol. 2. – P. 323–418.
24. *Кнут Д.* Искусство программирования для ЭВМ. – М.: Мир, 1978. – Т. 3. – 843 с.
25. *Цейтлин Г.Е.* Алгоритмы символьной обработки: объектная ориентация, трансформация, синтез // Смешанные вычисления и преобразование программ. – Новосибирск: ВЦ СО АН СССР, 1991. – С. 182–199.
26. *Цейтлин Г.Е.* Проектирование алгоритмов параллельной сортировки // Программирование. – 1989. – № 6. – С. 4–19.