ЧТО ТАКОЕ АЛГЕБРАИЧЕСКАЯ АЛГОРИТМИКА?

Цейтлин Г.Е., Мохница А.С.

Международный Соломонов университет, Киев, просп. 40-летия Октября, д. 100, кв. 93

Институт программных систем НАНУ, Киев-136, ул. Маршала Гречко, д.11а, кв. 148

E-mail: tseytlin@vikno.relc.com, alex_s_m@ukr.net

Даний огляд присвячено алгебраїчній алгоритміці (АА) - актуальній області комп'ютерної науки, що інтенсивно розвивається. Наведено обгрунтування напряму досліджень з алгебри алгоритміки в межах АА. Окреслено загальну характеристику та перспективи розвитку алгебри алгоритміки та її інструментальних засобів.

The review is dedicated to the algebraic alghorithmics, an intensively developed area of research of the computer science. The prospects of studies on algebra of algorithmics and its instrumental means in the framework of algebraic algorithmics are described.

1. Введение.

К числу актуальных и интенсивно развивающихся областей компьютерной науки относится алгебраическая алгоритмика (АА) [1]. В Украине исследования в данной области восходят к фундаментальным работам В.М. Глушкова по теории систем алгоритмических алгебр (САА) [2, 3]. Современное состояние исследований по алгебре алгоритмики отражено в [4] и связано с решением ряда важных теоретических и практических задач построения сверхвысокого уровня языков проектирования, а также средств автоматизации процесса синтеза (сборки) программ в объектно-ориентированных (ОО) вычислительных средах [5, 6]. Следует отметить, что множественность указанных языков, определяется концепцией алгоритмического клона, системы образующих (СО) которого соответствуют подходящей парадигме программирования (структурной, неструктурной, ОО и др.).

Материал данной статьи подчинён следующей структуре. В разделе 2 приведено обоснование направления исследований по алгебре алгоритмики в рамках алгебраической алгоритмики. Краткий обзор результатов по теории клонов приведен в разделе 3. Раздел 4 посвящён интегрированным инструментальным средствам автоматизированного синтеза программ в современных ОО-средах. В заключении приводятся результаты внедрения знаний по алгебраической алгоритмике в учебный процесс Международного Соломонова университета (МСУ). Очерчены общая характеристика и перспективы развития алгебры алгоритмики и ее инструментальных средств в рамках алгебраической алгоритмики.

2. Объект и метод алгебраической алгоритмики.

Данный раздел посвящен определению алгебраической алгоритмики (АА) в современной трактовке, принятой, например, в [1]. Приведено обоснование "экологической ниши" в данной области знаний, которая предназначена для алгебры алгоритмики (см. разд. 1).

Под алгебраической алгоритмикой в [1] понимается дисциплина, которая «оседлала математику и информатику» и предназначенная для алгоритмического описания методов обработки математических (алгебраических) объектов.

Следует, однако, отметить, что данное утверждение оказалось бы ещё более оправданным, при условии рассмотрения самих алгоритмов как алгебраических объектов и, в качестве следствия, инвариантного характера их описания относительно программной реализации алгоритмов в ОО-средах. Иными словами, алгоритмические описания, выполненные на алгебраическом языке сверхвысокого уровня, могут быть погружены в произвольно выбранную вычислительную среду, в частности, реализованы в языке АДА [1], также, впрочем, как и на любом другом языке программирования (Delphi, C++, Java и др.).

Вместе с тем, следует подчеркнуть, что, как и алгебраические описания вообще, подобная формализация алгоритмов предполагает их трансформацию, приведение к каноническим формам, оптимизацию, в частности, улучшение по выбранным критериям (быстродействие, объемы памяти и пр.).

Проблеме алгебраических трансформаций алгоритмов (последовательных и параллельных) посвящена многочисленная литература (см., например, [2, 3], [7] и др.). Применение преобразований, а также и других метаправил проектирования алгоритмов символьной обработки (сортировка, поиск, языковые процессоры и др.) продемонстрировано в [5] (главы 7-9, 11).

В частности, посредством циклических конструкций и аппарата разметки были представлены в САА/1-языке, улучшены с применением трансформаций и синтезированы на Паскале известные рекурсивные алгоритмы, спроектированные по методу быстрой сортировки Хоара (БСХ) [5].

Так, в качестве иллюстрации представим в алгебраической форме рекурсивный дихотомический алгоритм возведения в степень, который в общем виде ориентирован на вычисление n-й степени элемента и является одним из наиболее часто употребляемых в методах алгебраических вычислений [1].

Пусть необходимо вычислить степень a^n, где n – натуральное число. Данная задача может быть решена посредством элементарного вычислителя ЭЛВЫС накапливающего типа, функционирующего следующим образом. При поступлении на вход ЭЛВЫС одного из элементарных множителей - a^1, или a^2 он умножается на 'ЭЛВЫС - содержимое вычислителя ЭЛВЫС, при этом предполагается, что начальное состояние вычислителя $'ЭЛВЫС_0 = 1$.

Рассмотрим начальную разметку обрабатываемой структуры данных:

m_0: Н $У_1$ a^n К, где:

Н и К - маркеры начала и конца;

$У_1$ - указатель, отделяющий очередной элементарный множитель с его последующим поступлением на вход ЭЛВЫС.

Тогда дихотомический алгоритм ДИХАЛГ представим следующим образом:

ДИХАЛГ ::= {[d($У_1$,К)]

([n\2] $Р(У_1)$ * ЭЛВЫС (a^2) * ОБРАБ ((a^2) $^{n/2}$),

$Р(У_1)$ * ЭЛВЫС (a^1) * ОБРАБ' (a^{n-1}))},

где:

{[u] A} и ([u] A, B) - алгебраические представления цикла и альтернативы в САА [4];

d (У1, К) - предикат, истинный при достижении $У_1$ маркера К;

n\2 - предикат, истинный, если n - чётно;

$Р(У_1)$ - оператор сдвига $У_1$ на множитель вправо;

ОБРАБ ((a^2)$^{n/2}$), ОБРАБ' (a^{n-1}) - операторы отделения очередного элементарного множителя обрабатываемой структуры данных по истинной и ложной ветвям соответственно.

3. Функциональные построения в алгебре алгоритмики.

Данный раздел посвящен теории клонов, которая относится к числу фундаментальных разделов универсальной алгебры, имеющих ряд важных приложений в области дискретной математики и компьютерной науки [8, 4].

В частности, понятие клона (мета-алгебры) применяется для построения различных семейств алгебр, сходных по средствам описания, анализа и синтеза исследуемых объектов. Например, в теории формальных грамматик и связанных с ними автоматных и алгоритмических структур исследуются грамматические и алгоритмические клоны [9, 10]. Указанные клоны описывают семейства равномощных, по изобразительным средствам, алгебр и ассоциированы с известными парадигмами программирования (см. Разд. 1), а также методологией и технологией проектирования и синтеза алгоритмов и программ [11, 12].

Под клоном понимается, обычно одноосновная, универсальная алгебра вида: К = < А; SUPER >, где

А - основа, представляющая собой множество однотипных функций;

SUPER -сигнатура, состоящая из операции суперпозиции функций (подстановки одних функций вместо переменных в другие), а также операций отождествления и переименования переменных функций (более детально различные определения суперпозиции см. [13, 14]).

В частности, исследованные логические, грамматические и алгоритмические клоны имеют не только одну, но и несколько основ, а сигнатуры алгебр из соответствующего семейства, являются функционально полными, или СО данного клона [4].

Следует отметить, что фундаментом для построения различных алгебр из некоторого семейства, служит решение проблемы функциональной полноты сигнатуры операций для клона, определяющего данное семейство. Как известно, решение указанной проблемы сопряжено с построением решеток подалгебр и, в частности, совокупности максимальных подалгебр для соответствующего клона.

Построение решётки подалгебр может быть должным образом упорядоченно благодаря исследованию изолированных подалгебр и ряда их модификаций [3, 9, 15].

Пусть К ::= < А, СУПЕР > - некоторый клон, где А – основа, h = f(f1,f2,...,fm) - суперпозиция произвольно выбранных функций из основы А. Множество $M \subset A$ - изолированно, если любая суперпозиция $h \in M$ только тогда, когда все функции суперпозиции h включаются в М; иными словами, $h \in {}^{\wedge}M = A - M$, если хотя бы одна функция суперпозиции h из ^М (^ - дополнение М до основы А). Тем самым, построение решётки подалгебр клона А сводится к построению решеток подалгебр для М и ^М = А - М, где М - изолированная подалгебра.

В частности, решение проблемы функциональной полноты для алгебры А связано с построением совокупности максимальных подалгебр для М и расширений М до подалгебр максимальных для А.

3.1. Клон Поста. Рассмотрим двухзначную алгебру логики, именуемую далее клоном Поста (КП),

КП ::= <L(2); SUPER>, где

L(2) - множество всех булевых функций (БФ) - основа КП;

SUPER - вышеупомянутая операция суперпозиции.

Сформулируем основную теорему для КП, доказанную Постом [16].

Теорема 1 (о функциональной полноте в КП).

1. Совокупность всех подалгебр (замкнутых классов) - счётна;
2. Каждая подалгебра имеет конечный базис;
3. Построена решётка подалгебр, с указанием для каждой из них конечной совокупности ее максимальных подалгебр (предполных классов).
4. Верхний уровень построенной решётки - совокупность всех предполных классов КП (T0,T1, C,M,L) в терминах которых установлен критерий полноты для произвольно выбранной системы БФ.

Под поверхностью будем понимать верхний фрагмент решетки клона, содержащий все конечнопорожденные подалгебры [15].

Следствие. Построена поверхность КП, которая совпадает с решёткой подалгебр данного клона.

Решение проблемы функциональной полноты является инструментом для построения различных алгебр БФ: алгебры Буля, алгебры Жегалкина и др.[5, 16].

Заметим, что сигнатуры операций данных алгебр служат СО клона Поста, который входит в состав ряда многоосновных клонов, связанных с различными семействами алгебр алгоритмов.

Таким образом, мощность множества подалгебр КП - счётна, а их решётка "лежит" на поверхности.

3.2. Клоны континуального типа и их поверхности. В данном разделе приведены результаты по функциональным построениям в логических, грамматических и алгоритмических клонах [4, 11].

Универсальная алгебра, имеющая континуум подалгебр, называется алгеброй континуального типа [3].

Клон, основа которого содержит хотя бы одну бинарную ассоциативную операцию, будем называть полугрупповым. Из данного определения следует, что ассоциативная операция может принадлежать СО подалгебр данного клона, или быть производной от СО; примеры: клоны, основы которых содержат композицию операторов, или умножение языков [9, 10].

Теорема 2 [4]. Полугрупповые клоны - алгебры континуального типа.

Доказательство данного утверждения связанно с построением подалгебр, имеющих счётный базис.

3.2.1. Модификации клона Поста. Рассмотрим модифицированный клон Поста КП/м, связанный с полугруппой периодически определённых (ПО) преобразований на абстрактных (бесконечных) регистрах [3, 17].

На основании Теоремы 3 в основу КП/м и связанной с ним полугруппы входит композиция ПО преобразований; тем самым справедливо утверждение:

Теорема 3. Для КП/м выполняются следующие результаты:
1. КП/м и полугруппа ПО преобразований, а также ряд их подалгебр являются алгебрами континуального типа;
2. Построена поверхность решетки подалгебр для КП/м и установлены критерии функциональной полноты для её подалгебр с конечным базисом;
3. Исследованы пограничные для КП/м бесконечно порожденные подалгебры.

Подчеркнем, что аппарат КП\m представляет практический интерес в связи с тем, что ПО преобразования на абстрактных регистрах и их неоднородные обобщения реализуют синхронную параллельную обработку данных - содержимых регистров [3].

В СО логических клонов может быть включена бинарная операция: u = O . u', где u, u' – пред- и пост-условия, O - объект различной природы.

Одной из возможных интерпретаций данной операции является левое умножение условия на оператор, именуемое часто операцией прогнозирования [2, 3].

Обозначим КП/п логический клон, порожденный булевыми операциями и прогнозированием.

Для КП/п [10] , справедливы следующие результаты:
1. КП/п и некоторые его подалгебры являются алгебрами континуального типа;
2. Для КП/п установлен критерий функциональной полноты.

Отметим, в частности, что клон Поста и его модификации имеют важные приложения, связанные с описанием логических условий при конструировании алгоритмов и программ, а также обрабатываемых ими объектов в рамках современных языков программирования и вычислительных сред [4, 11].

3.2.2. Грамматические и алгоритмические клоны. Сформулируем следующие основные результаты, относящиеся к полугрупповым грамматическим и алгоритмическим клонам, которые соответствуют известным семействам алгебр алгоритмов [9, 11].

Теорема 4.
1. Указанные клоны - алгебры континуального типа;
2. Каждый клон включает бесконечно порожденные подалгебры (с бесконечным базисом и без базиса);
3. Для клона Клини и его обобщений построены семейства максимальных подалгебр и решена проблема функциональной полноты;

4. Построена поверхность КД и его обобщений ОКД(q), имеющих q выходов из цикла (q = 1,2,...);
5. Построена совокупность максимальных подалгебр клона КЯ, канонический представитель которого - алгебра Янова;
6. КЯ = $\lim_{k=1}^{\infty}$ ОКД(q) - теоретико-множественный предел обобщённых КД;
7. Построена совокупность максимальных подалгебр для клона КГ и его логической компоненты КП/п (здесь КД, КЯ, КГ - клоны Дейкстры, Янова и Глушкова, соответственно).

Важность распространения перечисленных результатов на алгоритмические клоны граф-схем и их обобщения, определяется приложениями теории граф-схем при визуализации объектно-ориентированного проектирования алгоритмов и программ [18, 10, 12].

Отметим взаимодополнительность полученных результатов по отношению к исследованиям по программологии и, в частности, по проблеме полноты для примитивных программных алгебр [19].

Следует заметить также, что рассмотренные результаты могут быть распространены на многосортные клоны, ассоциированные с абстрактными типами данных [5].

3.3. Алгебра алгоритмики и символьная обработка. Алгебра алгоритмики - двухуровневая система:

- верхний уровень, ориентирован на проектирование неинтерпретированных схем, здесь используется аппарат теории клонов в связи с построением подходящей алгебры алгоритмов и формированием её алгебраических основ;
- на втором уровне осуществляется конкретизация неинтерпретированных схем и построение прикладных алгебр алгоритмов, ориентированных на выбранные предметные области.

Получены ряд результатов, относящихся к задачам символьной обработки (сортировка, поиск и языковое процессирование) [5].

На всех уровнях алгебры алгоритмики применяются мета-правила проектирования схем: свёртка (укрупнение), развёртка (детализация), переинтерпретация (сочетание свёртки и развёртки) и трансформация, состоящая в совершенствовании схем на основе применения аппарата тождеств, квазитождеств и соотношений, характеризующих свойства операций алгебры алгоритмов и выбранной предметной области.

К числу актуальных задач алгебры алгоритмики относятся рекурсия и параллелизм, а также формализация концепции АТД. Решение этих задач тесно связано с важными приложениями в современных объектно-ориентированных вычислительных средах. В частности, следует упомянуть концепцию разметки и реализацию рекурсии, а также распределённой мультиобработки при проектировании и эффективизации задач символьной обработки [5, 20].

Отметим также ряд гуманитарных приложений в сфере университетского образования и социальной реабилитации лиц с ограниченными физическими возможностями [21].

4. Интегрированные инструментальные средства алгебры алгоритмики

Данный раздел посвящен интегрированным средствам конструирования синтаксически правильных последовательных и параллельных алгоритмов и программ, созданным в рамках исследований по автоматизации синтеза в объектно-ориентированных средах.

Описываемые средства базируются на алгебро-алгоритмических спецификациях и их диалоговой трансформации в плане улучшения качественных характеристик (быстродействие, память и т.п.), позволяя строить синтаксически правильные последовательные и параллельные САА-схемы алгоритмов и выполнять синтез соответствующих объектно-ориентированных программ (в частности, на языке Java) [22].

Необходимо также отметить, что инструментарий включает поддержку современных средств визуализированного проектирования объектно-ориентированных программ (UML, Rational Rose и др.) [23–25].

Особенность интегрированного инструментария (в отличие от системы МУЛЬТИПРОЦЕССИСТ [26]) состоит во взаимосвязанном применении всех трех представлений алгоритма [5], что дает более полную картину о конструируемом алгоритме, чем каждое из них в отдельности. При этом, изменение любого из этих представлений в инструментарии автоматически приведет к соответствующему преобразованию остальных.

Разработанный инструментарий был апробирован на задачах символьной мультиобработки (параллельные алгоритмы сортировки и поиска) [6]; он также может трактоваться как средство формализованной технологии повторного использования компонент (ПИК-технологии [23]).

4.1. Архитектура инструментария. Данные инструментальные средства состоят из компонентов, представленных на рис. 1 [27].

Первый компонент инструментария — ДСП-конструктор. В основу его функционирования положен диалоговый режим с использованием меню подстановок и FIFO (память типа очередь). Меню состоит из операторных и логических конструкций, суперпозиция которых позволяет создавать алгоритмы в упомянутых выше взаимосвязанных представлениях. Операторные и логические переменные запоминаются в очереди с дальнейшей их детализацией. В зависимости от типа переменной, считанной из очереди, система предлагает соответствующую компоненту меню или открывает для пользователя архив базисных понятий из базы знаний.

Отметим поуровневый стиль конструирования алгоритма, а также возможность переходов на различные уровни с продолжением процесса диалогового конструирования, причем подобный переход сопровождается соответствующим изменением состояния FIFO.

Процесс конструирования алгоритмов отображается графически в виде дерева. Вначале пользователь выбирает в меню одну из операторных структур, определив таким образом главную конструкцию схемы. Данной конструкцией помечается корень дерева. Затем добавляются его дочерние вершины, которые содержат имена переменных, входящих в данную конструкцию. Далее пользователь может выбирать операции, конкретизирующие соответствующие переменные или базисные конструкции (операторы, предикаты). САА-схема, соответствующая дереву конструирования алгоритма, отображается в отдельном текстовом окне в аналитической или естественно-лингвистической форме, в зависимости от установленной опции.

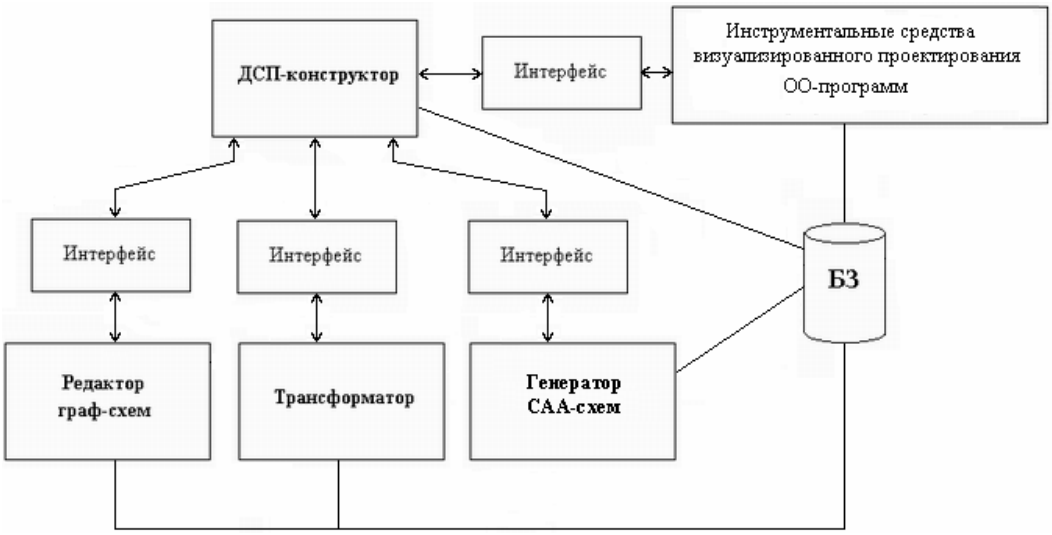

Рис. 1. Архитектура интегрированного инструментария.

С целью ориентации инструментария на построение алгоритмов параллельной обработки в число языковых структур, которые предоставляются пользователю в меню для выбора, были включены средства модифицированных САА (САА-М): асинхронная дизъюнкция, оператор контрольной точки, синхронизатор [6].

Блок трансформатора ориентирован на преобразование схем алгоритмов с использованием соотношений и тождеств, содержащихся в одном из разделов базы знаний и характеризующих свойства конструкций — операций соответствующей алгебры алгоритмов [5]. В диалоговом режиме осуществляется выбор тождества или соотношения, необходимого для использования в процессе преобразования.

Предназначение данного компонента — оптимизация схемы соответственно избранным критериям.

Генератор САА-схем [28] ориентирован на параметрически управляемый синтез формализованных спецификаций параллельных и последовательных алгоритмов и программ. Генерация выполняется посредством схем более высокого уровня, называемых регулярными гиперсхемам (РГС). РГС применяются, в частности, для представления алгоритмов управления выводом в грамматиках структурного проектирования (ГСП) [26]. Проектирование гиперсхем, как и САА-схем, выполняется в диалоговом режиме.

Одна из главных составных частей инструментария – редактор граф-схем – ориентирована на редактирование внешнего вида граф-схем алгоритмов и программ, которые создаются интегрированным инструментарием в процессе ДСП-конструирования и, возможно, модифицируются в блоке диалоговой трансформации. При этом изменения, внесенные в процессе редактирования, соответствующим образом отображаются на остальных представлениях (аналитическом, естественно-лингвистическом).

Редактор предоставляет пользователю возможность изменить вид любой компоненты граф-схемного представления, используя: перемещение узлов граф-схемы (преобразователей и распознавателей) с растягиванием дуг, изменение надписей в узлах и их цвета, размера, толщины и стиля линий и стрелок. Имеется возможность добавить на граф-схему ее заголовок и другие надписи.

Подчеркнем также возможность повторного использования граф-схем, уже созданных и сохраненных в базе алгоритмических знаний.

База алгоритмических знаний инструментария включает следующие разделы:

- разработанные алгоритмы из разных предметных областей (в рамках ПИК-технологии);
- стратегии обработки (схемы, которые описывают классы алгоритмов и подлежат дальнейшей детализации);
- метаправила свертки, развертки и трансформации (обеспечивают абстрагирование, детализацию и переинтерпретацию схем, а также содержат тождества и соотношения для преобразования схем);

- базисные понятия и их программные реализации (ориентированные на проектирование алгоритмов и синтез программ в данной предметной области на избранном целевом языке);
- графические элементы, используемые для представления граф-схем алгоритмов (различные виды стрелок, блоков и т.п.).

4.2. Интеграция с современными системами объектно-ориентированного проектирования. По полученным в процессе конструирования алгоритма дереву, реализациям элементарных операторов и условий, а также другим фрагментам программ на целевом объектно-ориентированном языке программирования, ДСП-конструктор инструментария осуществляет синтез программы.

Для синтеза программы в ДСП-конструкторе управляющие конструкции схемы отображаются в соответствующие операторы языка программирования, а вместо базисных операторов и предикатов подставляются их реализации на этом же языке. На вход синтезатора программ поступает также файл, содержащий каркасное описание основного класса приложения (без реализаций методов), в который выполняется подстановка синтезированного кода. Описать структуру классов и выполнить генерацию каркасного программного кода можно с использованием системы Rational Rose, базирующейся на языке диаграмм UML.

При этом разработкой и кодированием реализаций методов классов приходится заниматься вручную. Средством автоматизации данного процесса служит интегрированный инструментарий.

Необходимо отметить, что сочетание диаграммы UML с аппаратом граф-схем осуществляется в русле современной тенденции к визуализации средств объектно-ориентированного проектирования. Благодаря наглядности граф-схем и многоуровневости представления алгоритмов достигается независимость созданного таким образом проекта от кодирования на конкретном языке программирования.

Следует отметить возможность использования средств алгебры алгоритмики (включая граф-схемы и интегрированный инструментарий) на начальных этапах проектирования с дальнейшим использованием диаграмм UML и изложенного выше перехода к объектно-ориентированным программам.

Заметим, что необходимость привлечения аппарата граф-схем к средствам UML, нашла отражения в новейших версиях этого языка [25].

Таким образом, сочетание данного инструментария, использующего аппарат граф-схем, и средств визуализированного проектирования объектно-ориентированных программ (UML, Rational Rose) предоставляет пользователю комплект, позволяющий сопровождать разрабатываемый проект на всех стадиях жизненного цикла программной системы.

5. Заключение

Подводя итог исследованию, выполненному в данной статье, следует упомянуть о внедрении знаний по алгебраической алгоритмике при выполнении бакалаврских и магистерских работ на факультете компьютерных наук МСУ. Тематика упомянутых работ выбрана в рамках задач, относящихся к объекту и методу алгебраической алгоритмики. В частности, отметим следующие студенческие работы бакалавров: «Алгоритмы нахождения НОД и НОК по методу Евклида» Ильи Акопника, «Дихотомический алгоритм возведения в степень. Распараллеливание» Владислава Шевеленко, «Параллельный алгоритм нахождения определителя матриц порядка “n”» Александра Бавыкина, «Параллельный алгоритм распределенной сортировки» Владимира Иванова и магистров: «Теория клонов и языки параллельного программирования» Алексея Харченко, «Средства проектирования рекурсивных алгоритмов символьной обработки данных» Игоря Корха.

Таким образом, в обзоре проведен детальный анализ современного состояния исследований в области алгебраической алгоритмики. Установлено и обосновано важное место алгебры алгоритмики в рассмотренной области знаний. Дана характеристика основных результатов, полученных по теории клонов и инструментарию алгебры алгоритмики.

Перспективы теоретических и прикладных исследований по алгебре алгоритмики состоят:

- в дальнейшем внедрении полученных результатов в область алгебраической алгоритмики;
- в привлечении к данным исследованиям специалистов в области алгебры и информатики, в частности, научной молодёжи;
- в распространении полученных результатов за рубежом с целью участия в международном сотрудничестве по развитию рассмотренной области знаний.

Литература

1. *Ноден П., Китте К. Алгебраическая алгоритмика (с упражнениями и решениями). - М.: Мир, 1999. - 720с.*
2. *Глушков В.М. Теория автоматов и формальное преобразование микропрограмм // Кибернетика - 1965. - №5.- С.1-10*
3. *Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л. Алгебра. Языки. Программирование. 3-е изд., перераб. и доп. - Киев: Наук. думка, 1989. – 376 с; 1-е изд. - 1974 г; 2-е изд.-1978 г.; немецкое издание: Gluschkow W. M., Zeitlin G. E., Justchenko J. L. - Algebra. Sprachen. Programmierung. - Akademie-Verlag, Berlin 1980. - 340 p.3.*
4. *Цейтлин Г.Е. Алгебры Глушкова и теория клонов // Кибернетика и системный анализ. – 2003. - №4. - С. 48-58.*
5. *Цейтлин Г. Е. Введение в алгоритмику - К.: Сфера, 1999. - 720 с.*
6. *Цейтлин Г.Е., Яценко Е.А. Элементы алгебраической алгоритмики и объектно-ориентированный синтез параллельных программ // Математические машины и системы. – 2003. – № 2. – С. 64-76.*
7. *Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л. Методы символьной мультиобработки. - К.: Наукова думка, 1980. - 252 с.*
8. *Кон П. Универсальная алгебра. - М.: Мир, 1968. - 348 с.*

9. *Цейтлин Г. Е. Проблема функциональной полноты для мета-алгебр регулярных событий // Кибернетика и системный анализ. - 2000. - №6. - С. 14-27.*
10. *Цейтлин Г. Е. Проблема функциональной полноты в итеративных мета-алгебрах // Кибернетика и системный анализ. - 1998. - №2. - С. 28-45.*
11. *Цейтлин Г. Е. Алгебраическая алгоритмика: теория и приложения // Кибернетика и системный анализ. - 2003. - №1. - С. 8-18.*
12. *Цейтлин Г.Е., Теленик С.Ф., Амонс А.А. Алгебро-логическая формализация в объектно-ориентированных технологиях // Проблемы программирования. - 2002. - №1-2. - С. 136 - 146.*
13. *Мальцев А.И. Итеративные алгебры и многообразия Поста // Избр. тр. А.И. Мальцева. - 1976. - Т.2 - С. 316-330.*
14. *Яблонский С.В. Функциональные построения в k-значной логике // Труды МИАН СССР. - 1958. - Т.51. - С. 5-142.*
15. *Цейтлин Г.Е. Построение решетки подалгебр алгебры Дейкстры // Кибернетика и системный анализ. - 1997. - №1. - С. 27-45.*
16. *Post E.L. The two-valued iterative systems of mathematical logic // Ann. Math. Studies. - 5. - 1941. - P. 147*
17. *Цейтлин Г.Е. О бесконечно порожденных подалгебрах модифицированной алгебры Поста // Кибернетика. -1971. -№2. - С. 43-56.*
18. *Цейтлин Г.Е. Ющенко Е.Л. алгебра алгоритмов и граф-схемы Калужнина // Кибернетика и системный анализ. - 1994. - №2. - С. 3-17.*
19. *Буй Д.Б., Редько В.Н. Примитивные программные алгебры вычислимых функций // Кибернетика. - 1987. - №3. - С. 68-74.*
20. *Дорошенко А.Е., Цейтлин Г.Е. Алгеброавтоматные спецификации параллельных программ над общей и распределенной памятью // Проблемы программирования. - 2003. - №3. - С. 5 - 21.*
21. *Цейтлин Г.Е., Юсим Г.Е. Развитие алгебраических основ алгоритмики и перспективные информационные технологии // Проблемы управления и информатики. - 1997. - №6. - С. 89-95.*
22. *Яценко Е.А. Конструирование параллельных объектно-ориентированных программ // Проблемы программирования. – 2002. – №1–2. – С. 188–197.*
23. *Бабенко Л.П., Лаврищева К.М. Основи програмної інженерії: Навч. посіб. – К:. Т-во "Знання", КОО. – 2001. – 269 с.*
24. *У. Боггс, М. Боггс. UML и Rational Rose. М.: ЛОРИ. – 2000. – 582 с.*
25. *UML 2.0 Specification. – http://www.omg.org/technology/uml/index.htm.*
26. *Ющенко Е.Л., Цейтлин Г.Е., Грицай В.П., Терзян Т.К. Многоуровневое структурное проектирование программ: Теоретические основы, инструментарий. – М.: Финансы и статистика, 1989. – 208 с.*
27. *Цейтлин Г.Е., Амонс А.А., Головин О.В., Зубцов А.Ю. Интегрированный инструментарий проектирования и синтеза классов алгоритмов и программ // Кибернетика и системный анализ. – 2000. – №3. – С. 165–170.*
28. *Яценко Е.А. Алгебры гиперсхем и интегрированный инструментарий синтеза программ в современных объектно-ориентированных средах. // Кибернетика и системный анализ. – 2004. - №1. – С. 47-52.*