УДК 681.3.016

ЭКОЛОГИЧЕСКИЕ АСПЕКТЫ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ СРЕДСТВАМИ АЛГЕБРЫ АЛГОРИТМИКИ

Г.Е.Цейтлин, Л.М. Захария, О.В. Захария, Ю.И. Жовнир

Институт программных систем НАН Украины, проспект Академика, Глушкова, 40

zlm.lviv@gmail.com

Работа представляет экологические аспекты порождения алгоритмических знаний предметных областей. Средства алгоритмической формализации и синтеза знаний предметных областей представляют собой триаду – абстракции, биологию и экологию программирования. В качестве абстрактного механизма используется алгебраический аппарат теории клонов. Биологическая компонента отвечает за распространение полученных алгоритмических знаний на другие задачи данной предметной области и близких к ней областей. Экологический компонент предназначен для формирования инструментальных средств поддержки методов абстрактной и биологической составляющих теории клонов. В рамках экологической компоненты предлагаются различные интерпретации алгоритмических операций, средства исследования параллелизма (в частности структурного на примере клеточных автоматов Минского), механизмы вывода алгоритмических знаний.

This paper presents the environmental aspects of generation of algorithmic knowledge domains. Facilities of algorithmic formalization and synthesis of knowledge domains represent a triad - abstraction, biology and ecology of programming. As an abstract mechanism is used an algebraic formalism of the theory of clones. The biological component is responsible for the spread of obtained algorithmic knowledge to other tasks of the subject area and related areas. Environmental component is designed to build tools for support of abstract and biological components of the theory of clones. As part of the environmental components are offered different interpretations of algorithmic operations, facilities of the study of parallelism (including structural), the output engine of algorithmic knowledge, means to support the accumulated knowledge bases of subject areas.

Введение

Алгебраизация программирования – направление, формализующее опыт программирования и предоставляющее алгебраические инструменты для дальнейшего его развития. В Украине такой подход стал развиваться еще с 1965 г., получив свое воплощение в системах алгоритмических алгебр (САА) [1], названных по имени автора САА – Глушкова. САА формализуют процессы проектирования и синтеза (сборки) алгоритмов и программ. Эти объекты проектируются в терминах регулярных схем – алгебраических представлений в САА. Основными методами проектирования регулярных схем признаны интерпретация (развертка), абстрагирование (свертка), переинтерпретация (свертка с последующей разверткой) и трансформация. Развитие аппарата эквивалентных преобразований, который лежат в основе трансформации, обеспечило также возможность оптимизации объектов проектирования по выбранным критериям (например, память, быстродействие и пр.), определило пути перехода от одной предметной области к другой, близкой по алгоритмическим аспектам обработки информации. Развитием САА стала теория клонов [2], которая формализует современные подходы к программированию: представление алгоритмических знаний различных предметных областей (биология программирования [3]) и инструментальная поддержка создания и сопровождения таких алгоритмических баз знаний ПО (т.н. экология программирования).

Изложение материала статьи подчинено следующей структуре:

— биологическая и экологическая компоненты алгебры алгоритмики, функционанальная и аксиоматическая полнота;

— методы проектирования алгоритмических знаний и их инструментальная поддержка – САА- автомат;

— средства параллелизма в алгоритмике – исследование структурного параллелизма, однородные и неоднородные автоматы Минского – алгоритмический аспект;

— инструментарий автоматизированного проектирования компьютерных приложений – назначение и основные компоненты.

1. Основные аспекты теории клонов. Функциональная и аксиоматическая полнота

Представленное в работе исследование определяет формализованные средства проектирования и исследования алгоритмических знаний предметных областей.

В алгебре алгоритмики предложен подход к использованию клонов – как основного инструмента описания различных средств проектирования алгоритмических знаний в предметных областях.

Под клоном будем понимать одно- или многоосновную алгебру

$$C ::= \langle F/q; SUPER \rangle, \quad (1)$$

где F/q – совокупность основ q = 1, 2, ..., n. Каждая основа состоит из множества функций определенного типа (логического, операторного, структур памяти и данных и пр.); SUPER – сигнатура клона, представляющая операцию суперпозиции функций, принадлежащих основам. В одну – главную функцию вместо переменных определенных типов подставляются другие функции соответствующего типа. Каждая основа клона состоит из множества функций одного и того же типа.

Клон Поста (КП) суть одноосновный клон. Его основа – множество всех булевых функций (БФ). КП ориентирован на построение семейства алгебр БФ. Необходимость в построении такого семейства связана с традиционными приложениями двухзначной алгебры логики – конструирование комбинационных схем (компьютерная аппаратура), алгоритмов и программ в языках программирования (ЯП).

Перечисленные далее клоны формализуют проектирование алгоритмической структуры программных приложений.

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

Приведем пары: представительная алгебра и соответствующий ей алгоритмический клон [2]. Подчеркнем, что каждой паре соответствует определенный метод и технология программирования.

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 (КК) (обобщенные АК – КК).

Каждая представительная алгебра является двухосновной: логической и операторной. Логическая основа состоит из булевых операций. В клоне Глушкова логическая компонента дополнена операцией прогнозирования. Клоны Дейкстры, Калужнина и Янова в качестве логической компоненты используют клон Поста. Проблема функциональной полноты для перечисленных клонов решена в [2] с учетом наличия операции прогнозирования в КГ, что и объясняет изобразительную мощь САА и клона КГ в сравнении с другими клонами.

Для клона n-отношений определена трехосновная представительная алгебра. Двумя основами такой алгебры являются известные логическая и операторная компоненты систем алгоритмических алгебр Глушкова. В качестве третьей компоненты выбрана алгебра n-отношений. Исследование клона n- отношений сопряжено с исследованием канонических форм запросов в базах данных.

Доказано, что клон n-отношений изоморфен клону регулярных выражений Клини, тем самым исследование функциональной полноты клона n-отношений приведено к проблеме исследования функциональной полноты клона Клини.

Основной операцией теории клонов [4, 5], классическим примером которой служит алгебра логики L(2), является суперпозиция функций (операций) представительной 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 и базирующийся на ней механизм построения различных алгебр БФ. В основу исследований Поста положено построение решетки 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 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. Например, к классу булевых алгебр относится общеизвестная алгебра множеств.

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 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.

Важным направлением настоящего исследования является построение аксиоматик для вышеперечисленных представительных алгебр с целью исследования проблемы аксиоматической полноты и построения канонических форм выражений в каждой представительной алгебре.

С этой же точки зрения важной проблемой, открытой для исследований является переход к *k-значным логикам* в САА, что важно для представления многозначного знания (например, при работе с неопределенностью).

2. Методы проектирования алгоримических знаний. САА-автомат

Одна из основных алгебраических задач состоит в том, чтобы, построив законодательство данной алгебры – систему основных равенств (аксиом), характеризующих базовые свойства операций, входящих в сигнатуру алгебры, выполнять аналитические преобразования формул – суперпозиций операций из сигнатуры.

В результате применения эквивалентных преобразований осуществимо приведение формул к канонической (единственной) форме, оптимизации, в частности, улучшению формул по тем или иным критериям, решения проблемы тождеств аналитических представлений в данной алгебре.

В алгебрах алгоритмов формулы представляют собой схемы – модели алгоритмов или программ. Тем самым, алгоритмы возможно улучшать по выбранным критериям (память, быстродействие и пр.), а также решать ряд других важных и сложных задач алгоритмики и программирования.

Определены следующие механизмы вывода новых знаний в клонах: свертка, развертка, переинтерпретация, трансформация. Каждая схема является выражением в любой из вышеперечисленных представительных алгебр, а проектирование алгоритмов как раз и связано с применением этих метаправил конструирования.

Свертка – процесс преобразования схемы, в котором все составные объекты (операторы, условия, составные n-отношения) заменяются на алгебраические выражения, конкретизирующие эти объекты. В результате свертки схема приводится к единому аналитическому выражению, содержащему только операции алгебры и элементарные объекты. Элементарные объекты схемы не подлежат дальнейшей детализации средствами алгебры и реализуются непосредственно в языках программирования. Именно такие свернутые схемы используются для автоматизированного синтеза соответствующих им программ.

Развертка – процесс противоположный свертке. Во время развертки схема приводится к множеству алгебраических выражений в соответствующей представительной алгебре, каждое выражение из такого множества использует только одну операцию этой алгебры и представляет отдельный составной объект.

Интерпретированная схема – это схема, для которой заданы семантические значения элементарных объектов (например, в виде процедур и функций, шаблонов или классов в языках программирования). Схема, в которой не определены все элементарные объекты называется неинтерпретированной и задает стратегию выполнения схемы в терминах алгебраических операций. Частично интерпретированной схемой является схема, в которой присутствуют интерпретированные и неинтерпретированные элементарные объекты.

Переинтерпретация – процесс, заменяющий одни семантические значения элементарных объектов на другие, не меняя при этом стратегии вычисления.

Трансформация – это такое преобразование схемы с применением эквивалентного соотношения вида f=g (f, g – эквивалентные схемы в соответствующей схеме алгебре), при котором фрагмент этой схемы, имеющий аналогичную алгебраическую структуру с точностью до составных объектов с одной из частей эквивалентного соотношения, может быть заменен на другую часть этого соотношения.

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 САА-схем в результате применения соотношений. Каждое соотношение состоит из левой и правой частей, отражающих свойства операций, входящих в сигнатуру рассматриваемой алгебры.

Сферами использования указанных преобразований являются:

- синтез программ по свернутым схемам и набору реализаций семантических базовых элементарных объектов;
- исследование сопряженных предметных областей на базе переинтерпретации;
- оптимизация схем по определенным критериям, задающих свойства операций алгебры или предметной области в виде трансформационных эквивалентных соотношений;
- исследование аксиоматической полноты клонов и приведение схем различных типов к каноническому виду.

В качестве механизма автоматизированного применения указанных метаправил предлагается САА-автомат. Принцип функционирования автомата базируется на разверточной стратегии синтаксического анализа по LL(k) грамматикам, причем k – имеет переменное значение. На вход автомата подается схема, LL(k) грамматика, которая порождает эту схему и является программой автомата. На выходе автомат в зависимости от режима своей работы выдает свернутую схему, синтезированную по схеме программу в целевом языке программирования, переинтерпретированную схему или преобразованную на основании эквивалентного соотношения новую схему, эквивалентную исходной.

САА-автомат представляет собой рекурсивную машину. Рекурсия здесь реализуется через указатели, которые передвигаются по входной цепочке автомата и фиксируют место применения очередной его команды.

Автомат работает под управлением следующих команд:

B,W,O – команды обработки составных объектов (B – составной оператор, W – составное условие, O – составное отношение).

Команда обработки альтернатив составных объектов: предполагается, что каждый составной объект состоит из 2 и более упорядоченных альтернатив. Автомат проверяет условия входа в альтернативу и выполняет именно ту ветвь, для которой условие входа первым оказалось истинным. Рекомендуется последнюю альтернативу составного объекта предварять тождественно истинным условием с обработкой исключительной ситуации его аномального выполнения.

C – команда циклического выполнения

X; X& – команды завершения альтернативной ветви или сложного логического условия соответственно.

Команды логических дизъюнкции, конъюнкции, отрицания.

Команды выполнения семантических елементарных объектов (процедур, шаблонов).

Программа автомата представляется в списковой форме. Она состоит из последовательности команд автомата и таблиц указателей, которые фиксируют начала детализации каждого объекта программы, как составного, так и элементарного.

Просматривается прямая аналогия между нисходящим синтаксическим анализом и интерпретацией САА-схемы. Составной оператор схемы играет роль нетерминала грамматики. Схема состоит из набора составных объектов, каждый из которых реализован в виде списка альтернатив с условиями входа, тогда как LL(k) грамматика состоит из набора альтернатив нетерминала, а условием входа в альтернативу является совпадение K символов входной цепочки с терминальной цепочкой данного правила.

В случае, когда САА-автомат работает в режиме синтаксического анализа и трансляции, его программа расширена стандартными командами работы с терминалами – сравнения символов входной строки с терминальными символами грамматики и считывания их в случае совпадения.

Именно за счет такой близкой аналогии между схемами и представлением грамматики описанный универсальный автомат может обеспечивать выполнение всех механизмов вывода алгоритмических знаний, а также осуществлять интерпретацию и отладку схем в терминах схемных объектов, а не объектов синтезированной по схеме программы.

Особенно важной представляется роль такого автомата в процессе трансформации, поскольку при определении фрагмента для замены, согласно эквивалентному соотношению, используется метод совпадения синтаксической конструкции этого фрагмента с левой или правой частью соотношения, а не метод их тождественного символьного совпадения.

Важно отметить еще одну сферу применения САА-автомата – для работы с иерархическими структурами. В частности, с его помощью удалось создать гипертекстовый редактор, который позволяет работать с текстами как со списковыми формами, упрощая таким образом их редактирование и восприятие.

3. Средства структурного параллелизма.

В алгоритмическом клоне Глушкова определены операции синхронного и асинхронного параллелизма. Здесь нашли свое отображение и периодически определенные преобразования. Для указанных операций параллелизма сформированы соотношения приведения к каноническим формам, представлены соотношения для выявления тупиков и клинчей параллельного выполнения.

В рамках продолжения исследования в данном направлении интересной представляется формализация средствами алгебры алгоритмики автоматов Минского. Однородные автоматы Минского являются хорошей моделью для исследования структурного синхронного параллелизма с различной топологией.

Однородные автоматы Минского с линейной моделью взаимодействия соседних объектов аналогичны периодически определенным преобразованиям.

В рамках реализации средствами алгебры алгоритмики автомата Минского с топологией двухмерного пространства были исследованы задачи «популяция» и «жизнь», описывающие сложные системы синхронного взаимодействия объектов.

Для реализации такого автомата выбрана двухмерная матрица, в ячейках которой размещены «хищники» или «жертвы». Введено 2 указателя X и У, фиксирующие размещение взаимозависимых, влияющих на состояние друг друга объектов. На первом такте указатель X связан с 1-й строкой и 1-м столбцом, указатель У – с теми элементами 2 строки и 2 столбца, которые не задействованы указателем X. Для них синхронным за один такт определяется новое состояние указанных элементов, в зависимости от состояния соседних объектов.

Новые состояния объектов, связанных с указателем X заменяют предыдущие значения, новые значения по указателю У запоминаются, после чего оба указателя сдвигаются на 2 позиции по главной диагонали матрицы на 3 и 4 строки, соответственно (фрагменты 3 и 4 строк и столбцов с пересечением на диагональном элементе – см. рис. 1). Для них синхронно вычисляются новые состояния элементов, при этом по месту указателя У, на первом такте работы алгоритма происходит замена текущего состояния на вычисленное на первом такте и сохраненное. При этом запоминается новое состояние элементов по указателю X второго такта. Алгоритм заканчивает работу по достижении указателем У последнего диагонального элемента матрицы.

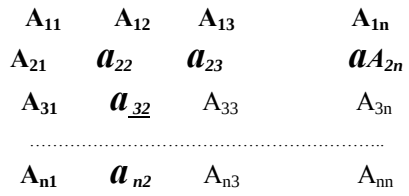

Рис. 1. Схема окрестностей синхронного преобразования состояния автомата Минского для случая пространственной топологии

Такая интерпретация алгоритма позволяет представить окрестности синхронного, периодически определенного преобразования для двухмерного автомата Минского, и реализовать алгоритм изменения состояния значений элементов для таких окрестностей. Для каждого элемента окрестности выполняется одна и та же процедура, которая задается правилами функционирования конкретного автомата Минского.

Неоднородные автоматы Минского используются при реализации нейронных сетей. Учитывая это в настоящее время ведутся исследования по представлению различных типов нейронных сетей средствами алгебры алгоритмики, накопления соответствующей алгоритмической базы знаний по нейронным сетях, и что самое главное, ведется исследование операций параллелизма для таких классов алгоритмов.

4. Инструментальные средства поддержки теории клонов

Аппарат алгебры алгоритмики предлагает высокоуровневые средства проектирования алгоритмов и структур данных специалистам предметных областей, не владеющих в совершенстве техникой объектно-ориентированного программирования. Эти средства составляют основу восходящего от синтезатора МУЛЬТИПРОЦЕССИСТ [6–8] интегрированного инструментария проектирования и синтеза алгоритмов и программ для погружения в объектно-ориентированные среды с подключением развитых в них инструментальных средств.

В работе [9] разработан Интегрированный Инструментарий Проектирования и Синтеза алгоритмов и программ (ИПС). Данный инструментарий восходит к синтезатору Мультипроцессист – одной из первых украинских CASE-систем, ориентированной на генерацию программ. В основу данной системы был положен метод многоуровневого структурного проектирования программ (МСПП) по их описаниям в языке САА/1. В отличие от системы Мультипроцессист, ориентированной на синтаксический анализ САА-схемы, в ИПС схемы проектируются на основе метода диалогового конструирования синтаксически правильных программ (ДСП-метода), обеспечивающего синтаксическую правильность схемы. Особенностью ИПС является также интеграция всех трех форм представления алгоритмов: аналитического, естественно-лингвистического и графового при их проектировании.

В состав Интегрированного Инструментария входят следующие компоненты (рис. 2):

— ДСП-конструктор, предназначенный для диалогового проектирования синтаксически правильных схем последовательных и параллельных алгоритмов и генерации программ, в том числе программ для объектно-ориентированных сред (в частности, C++ и Java);
— ДСП – конструктор граф-схем алгоритмов;
— ДСП – конструктор табличных структур данных (n-отношений);
— редакторы САА-схем, граф-схем, таблиц;
— трансформатор, предназначенный для интерактивной трансформации схем алгоритмов и программ, с целью их улучшения по различным критериям (используемая память, время выполнения и др.);
— генератор САА-схем по схемам более высокого уровня – гиперсхемам, представляющих собой обобщение грамматик структурного проектирования;
— среда конструирования алгебро алгоритмических описаний (СКАО), содержащая:

- описание конструкций САА, операций алгебры n-отношений в терминах целевого языка программирования проектируемого приложения;
- базисные понятия и их программные реализации;
- метаправила конструирования алгоритмов: свертка, развертка, трансформация, переориентация;
- схемы алгоритмов, стратегии обработки. Стратегии – это схемы, в которых базисные понятия заменены абстрактными переменными соответственного типа, стратегии обработки фиксируют алгоритмическую структуру обработки. Гиперсхемы – параметризированные схемы, описывающие классы алгоритмов с общей алгоритмической структурой.

Основная идея ДСП-конструктора состоит в поуровневом конструировании схем сверху-вниз путем детализации языковых конструкций САА. Процесс проектирования представлен в виде дерева конструирования алгоритма. На каждом шаге конструирования система в диалоге с пользователем предоставляет на выбор лишь те конструкции, подстановка которых в формируемый текст не нарушает синтаксическую правильность схемы. По полученному в результате конструирования дереву алгоритма может быть осуществлен синтез программы на основе программных реализаций из СКАО.

ДСП – конструктор табличных структур данных, разрешает визуальными средствами формировать таблицы баз данных и на их основе конструировать новые, как результат выполнений операций клона n-отношений над ними.

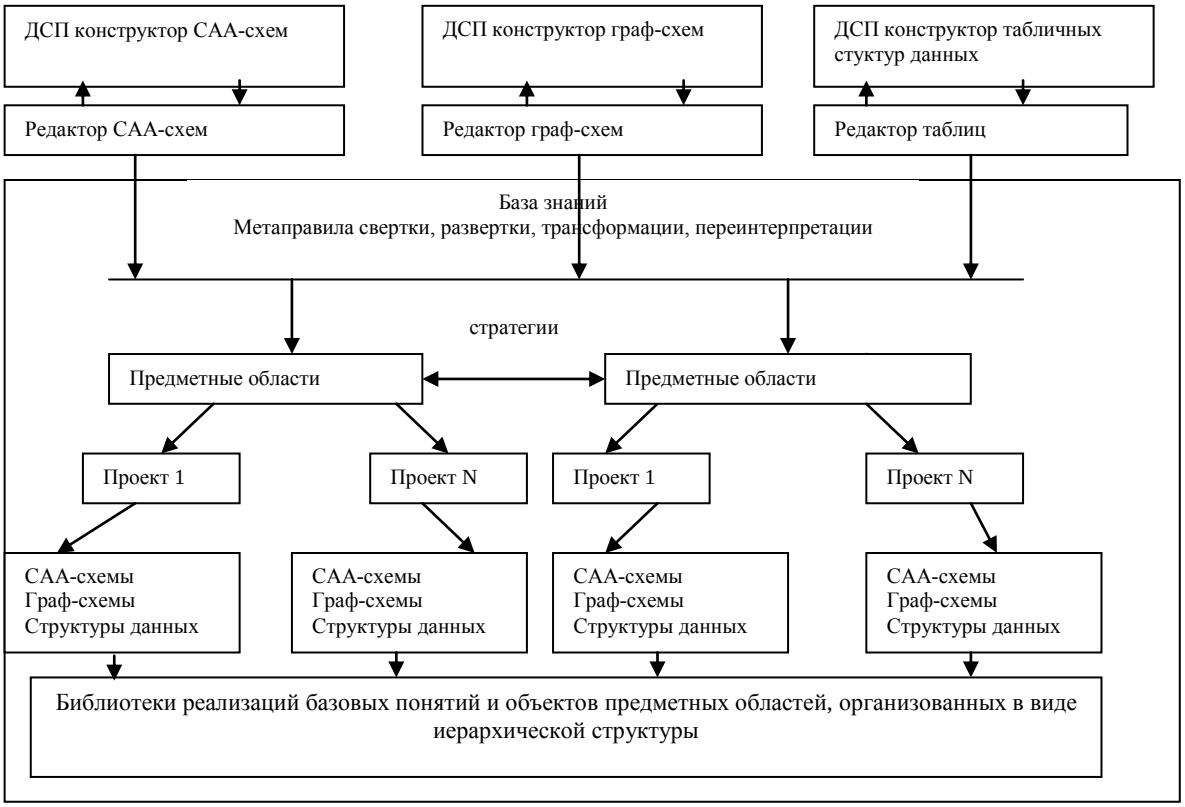

Рис. 2. Архитектура инструментария ИПС

ДСП конструктор граф-схем строит граф-схему алгоритма из базовых конструкций операций клона Калужнина на тех же принципах, что и ДСП-конструктор САА-схем.

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.

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

Наряду с алгебраической формой представления объектов, их адекватных текстовых и графовых описаний, способствует анализу взаимодополняющих аспектов проектируемых объектов и соответствующих средств автоматизированного синтеза. Так, граф-схемная форма проектирования адекватна средствам визуального представления знаний в UML с использованием системы автоматизации объектно-ориентированного программирования Rational Rose.

Для поддержки в базе знаний проектов из различных предметных областей необходимо иметь возможность устанавливать между всеми ее компонентами иерархические связи различной природы (подчиненность, взаимозаменяемость, наследование и т. п.) для поддержки такой иерархической структуры в инструментарии САА-автомат. САА-машина не только отвечает за работу иерархической базы знаний, но и является механизмом синтаксического анализа САА-схем, созданных без использования ДСП конструктора, механизмом синтеза и сборки программ по их САА-схемам и интепретациям базовых понятий, т. е. такая САА-машина представляет собой универсальный механизм работы с иерархическими структурами.

1. *Многоуровневое* структурное проектирование программ: Теоретические основы, инструментарий / Е.Л. Ющенко, Г.Е. Цейтлин, В.П. Грицай, Т.К. Терзян. – М.: Финансы и статистика, 1989. – 208 с**.**
2. *Цейтлин Г.Е.* Введение в алгоритмику. – К.: Сфера, 1998. – 310 с.
3. *Czarnecki K., Eisenecker U.* Generative Programming: Methods, Tools and Applications. 1st edn. Addison-Wesley Professional (2000). 864 p.
4. *Кон П.* Универсальная алгебра. – М.: Мир, 1968. – 348 с.
5. *Курош А.Г.* Лекции по общей алгебре. – М.: Физматгиз. – 1962. – 396 с.
6. *Codd E.F.* A Relational Model of Data for Large Shared Data Banks // Comm. ACM. – 1970. – Vol. 13. – N 6. – P. 377–387.
7. *Грицай В.П., Захарія Л.М.* Мультипроцесист – автоматизована система багаторівневого проектування програм. 0.3 // Тези доп. 9-ї Всесоюз. наради з питань управління, Єреван. – 1983.
8. *Захарія В.П., Захарія Л.М.* Інструментарій синтезу програм для міні- і мікро ЕОМ, ,1.5 // Кібернетика. – № 6. – 1989.
9. *Яценко Е.А., Мохница А.С.* Инструментальные средства конструирования синтаксически правильных параллельных алгоритмов и программ // Проблеми програмування. 4-я Междунар. науч.-практ. конф. по программированию УкрПРОГ'2004. – 2004. – № 2–3.. – С. 444–450.