УДК 519.3

А.Е. Дорошенко, Л.М. Захария, Г.Е. Цейтлин

АЛГЕБРАИЧЕСКОЕ ПРОЕКТИРОВАНИЕ ПРОГРАММ: АЛГОРИТМЫ, ОБЪЕКТЫ, ИНСТРУМЕНТЫ

Работа посвящена современным исследованиям средств проектирования алгоритмов и программ. Известный тезис Вирта: “алгоритмы + структуры данных = программы” распространен на современные исследования по методологии и технологии программирования. При этом особое внимание уделено тем из них, которые сопряжены с использованием алгебраического аппарата.

Введение

Одна из перспективных областей знаний, интенсивно развивающихся в настоящее время на Западе, получила название алгебраической алгоритмики (АА) [1, 2]; другая – ментальное программирование (МП, или IP) [3] (далее АА и МП назовём прототипами).

Отнесём АА к методам нисходящего проектирования алгоритмов и программ, тогда как МП – к их восходящему проектированию. Алгебраическая алгоритмика – формализованный подход к описанию методов обработки математических (алгебраических) объектов. В указанном исследовании в качестве предметной области выбраны наиболее известные алгоритмы Теории чисел и алгебры от Евклида до наших дней. Основные алгебраические структуры и концепции реализованы в языке АДА, в связи с чем следует отметить, что рассмотрение самих алгоритмов как алгебраических объектов и разработка методов их инвариантного описания относительно программной реализации в объектно-ориентированных средах дало бы возможность погружения их в произвольную вычислительную среду в любом языке программирования. К сожалению, такое глубокое исследование данной предметной области не выработало обобщенного подхода к разработке формализованных методов проектирования алгоритмов и программ, которое бы базировалось на алгебраическом аппарате, и как следствие предполагало их трансформацию, приведение к каноническим формам, оптимизацию по выбранным критериям.

Ментальное программирование, или порождающее программирование (Generative Programming, GP), открывает перед разработчиками приложений глобальные перспективы. Оно реализует идею перехода от «одноразовых» программных систем к полуавтоматическому производству самых разнообразных продуктов. Самое ценное качество методики порождающего программмирования состоит в том, что она учитывает преимущества автоматизации применительно к разработке программных средств. Эта книга содержит обзор методов и инструментов, обеспечивающих возможность проектирования и реализации «правильных» компонентов семейств систем, а также автоматизации их сборки. Представленные в этой книге методы применимы к любым программным проектам – от «локального» программирования на уровне классов и процедур до масштабных разработок семейств комплексных систем

Суть МП (или IP) характеризует следующая цитата из [3]: "Вариант IP: расширяемая среда программирования - обеспечение адекватной поддержки предметно-ориентированных абстракций – состоит в построении расширяемой среды программирования, какой и является система IP; на смену идее фиксированных языков программирования в ней предлагается принцип настройки специальных программных нотаций, которая становится возможной благодаря комбинированию активных библиотек реализации отдельных характеристик языков (намерений) или их наборов".

Интересным представляется исследование, содержащее сравнительный анализ указанных технологий с возможностями алгебры алгоритмики, подхода, ба-

зирующегося на алгебро-алгоритмических специификациях алгоритмов и программ, восходящим к системам алгоритмических алгебр В.М. Глушкова. Сравнительный анализ алгебраической алгоритмики и алгебры алгоритмики проведен в [2], в то же время анализу МП посвящена данная работа.

1. Сущности алгебраического проектирования

Общность указанных прототипов (МП и алгебраической алгоритмики) состоит в их целевой близости к идее и паРадигме развития алгебры алгоритмики [4, 5] – одного из направлений исследований украинской алгебро-кибернетической школы, восходящего от фундаментальных работ В.М. Глушкова по теории систем алгоритмических алгебр (САА) [6, 7]. В отличие от АА, будем для краткости в дальнейшем обозначать алгебру алгоритмики как <АА>.

Действительно, алгебраическая алгоритмика (АА) по своему замыслу должна была включить в сферу охвата и САА, как алгебру алгоритмов. Это словосочетание соответствует во-первых, алгебраической сущности САА, во-вторых, ориентации на конструирование алгоритмов обработки структур данных, характерных для решаемой задачи. При этом, в частности, отметим, что инструментарий САА [8] может быть использован и для синтеза программ в АДЕ – основной алгоритмической составляющей АА [1].

В то же время, МП, в частности, предназначено для облегчения и ускорения конструирования кода языков программирования (ЯП) различного уровня и прикладного назначения. Эта цель определяет инструментарий МП, предназначенный для построения прикладных абстракций и развития средств их поддержки в связи с ориентацией на многообразие областей приложений, включая и математизированные. Заметим, что парадигма САА также представляет собой вызов сложности конструирования программ различных языков программирования.

Таким образом, приходим к выводу, что развитие парадигмы САА и создание соответствующего инструментария преследует цели, подобные назначению МП.

Отмечая близость САА к указанным прототипам, следует подчеркнуть концептуальную целостность парадигмы САА, что во многом определяет преимущества этой парадигмы в сравнении с прототипами:

1. **Алгебраическая сущность.** *САА* – это алгоритмические алгебры, которые направлены на формализованное нисходящее проектирование объектов (алгоритмов, программ, абстрактных типов данных (АТД) и памяти (АТП)) в терминах неинтерпретированных и частично интерпретированных схем, называемых стратегиями обработки [5]. Схемное представление объектов формирует знания о средствах их проектирования и синтеза (рис. 1). Наряду с синтезом знаний в рамках <АА> [4] имеются средства прогнозирования и декомпозиции, а также моделирования и проверки эффективности поведения проектируемых моделей.

2. **Синтаксическая и семантическая правильность.** В процессе проектирования необходимо обеспечить правильность проектируемых объектов, которая должна быть поддержана соответствующими инструментальными средствами. Инструментальные средства поддержки <АА> содержат *диалоговые синтаксически-правильные* конструкторы (ДСП) [8, 9] ,синтаксические анализаторы, синтезаторы объектов [10]. Проблемы анализа и синтеза получили своё развитие в классической теории автоматов. Следует отметить необходимость декомпозиции (анализа) объектов для возможно последующей сборки (синтеза) нового знания (шаблоны-схемы, стратегии обработки, интерпретации и пр.).

Отметим, что процесс синтеза, как известно, состоит из двух основных этапов:

- фиксация контейнеров – базовых понятий, ассоциированных с выбранной предметной областью (семантические идентификаторы, интерпретации, их реализации, пр.);

- собственно генерация стратегий и алгоритмов обработки, посредством бази-

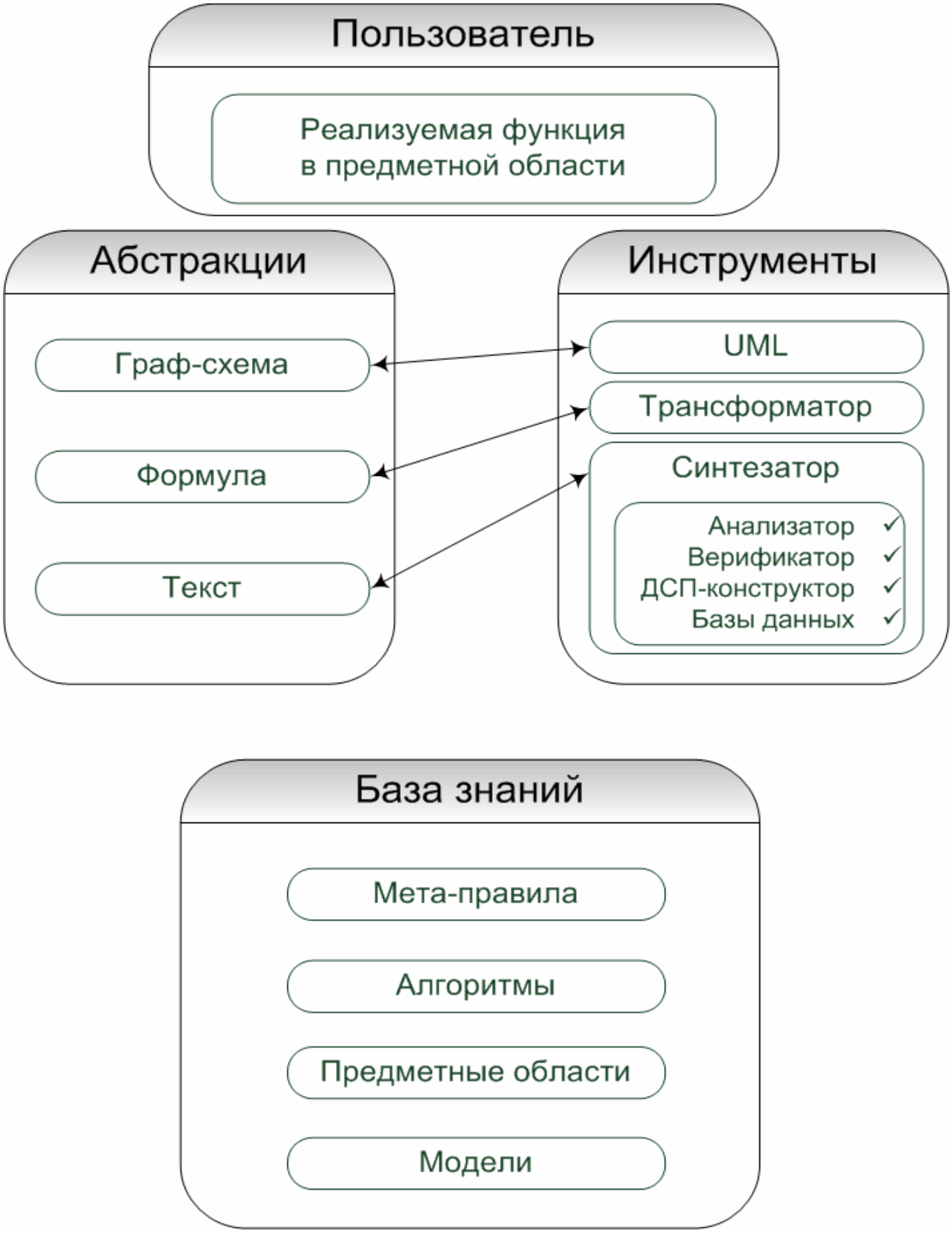

Рис. 1. Средства проектирования и синтеза

рующихся на соответствующих алгебрах алгоритмов, языков проектирования и синтеза программ на целевых языках программмирования [5, 11]. В настоящее время указанные этапы получили названия родового и генерирующего программирования [3].

Данные средства особенно важны для организации мультиобработки, в связи с взаимодействием параллельных ветвей.

Разумеется, эти знания могут быть использованы при распространении на различные (в том числе и близкие) модели вычислений.

3. **Модифицируемость.** Схемы – формы представления проектируемых обьектов базируются на применении определённых метаправил: свёртки (абстрагирование), развёртки (детализация), переинтерпретации (свёртка-развёртка) и трансформации (применение равенств – тождеств, квазитождеств и соотношений, характеризующих свойства операций, входящих в сигнатуру САА [12].

4. **Корректность модификаций.** Следует из функциональной эквивалентности схем, полученных в результате модификаций по перечисленным метаправилам, что как раз и определяет их семантическую правильность.

5. **Инструментарий проектирования.** Созданы и далее развиваются инструментальные средства проектирования объектов, принадлежащих одной и той же, или различным предметным областям [8, 9]. При этом проектируемые объекты имеют общую структуру, заданную трансформационными преобразованиями и интегрированные формы представления: аналитическую, текстовую и граф-схемную [13]. Эти формы отражают различные взаимодополняющие и инструментально поддерживаемые аспекты проектирования объектов.

6. **Последовательное и параллельное функционирование.** В зависимости от сигнатуры операций алгебра, положенная в основу выбранных средств проектирования может быть сориентирована на последовательное или параллельное функциионирование проектируемых объектов. Мультиобработка тесно связана с решением на уровне схем проблемы тупиков (клинчей), овеществлением событий – замкнутых условий прохождения соответствующих контрольных точек. Таким образом поставленная проблема клинчей оказывается связанной с задачей выявления фиктивных итерационных конструкций. Решение обеих задач требует распознавания соответствующего расположения контрольных точек и синхронизаторов. Иными словами, необходимо обеспечить проверку выполнения событий и средств задержки процессов вычислений по взаимосвязанным параллельным ветвям. Организация параллелизма на уровне схем связана также с использованием средств, развитых в теории замкнутых и локально-замкнутых логических условий (или монотонных операторов и их обобщений [6, 14].

7. **Объектно-ориентированная среда.** Процесс проектирования объектов предполагает интеграцию инструментария алгебры алгоритмики с развитыми в настоящее время средствами автоматизации программмирования (UML, Rational Rose, библиотеки объектов и др.). При этом отметим взаимодополнительность интегрируемых инструментальных средств в плане их независимого и совместного использования, в частности, для разработки специальных библиотек, ориентированных на поддержку синтеза объектов, относящихся к тем или иным предметным областям [10, 14].

8. **Клоны, методологический и технологический аспекты.** Со средины 90-х годов, в рамках дальнейшего развития алгебры алгоритмики, на основе известных методов и технологий программирования были построены и исследованы мета-алгебры-клоны [15 – 17], охватывающие семейства подобных алгебр алгоритмов (структурного, неструктурного, визуального типов и пр.). С каждой из алгебр, входящей в тот, или иной клон, могут быть ассоциированы свои инструментальные средства, адекватные выбранным предметной области, методу проектирования и привычной для разработчиков технологии производства объектов.

Следует подчеркнуть также, что присущий алгебрам схемный (аналитический) подход весьма близок шаблонам, принятым в современных объектно-ориентированных средах. При этом, наряду с детализацией (присвоением значений переменным формул, или полям шаблонов) сохраняется возможность модификации, характерной алгебраическому аппарату вообще. Создаваемые при этом объекты оптимизированы в плане улучшения их

качества соответственно выбранным критериям (память, быстродействие и др.).

К числу позитивных моментов развиваемого подхода следует отнести также возможность формализации ПИК-технологии (или многократного применения компонент), что соответствует использованию упомянутых ранее метаправил проектирования объектов.

9. **Предметная ориентация.** Первые предметные области, созданные в рамках развиваемого подхода, были связаны с алгоритмизацией ряда задач символьной обработки (сортировка, поиск, языковое процессирование). Именно, спроектирована (в терминах САА-схем) и синтезирована серия (порядка 50-ти) алгоритмов и программ последовательной сортировки (общий объём полученного программного продукта до 10 тыс. строк на ПАСКАЛе) [11]. Метаправило переинтерпретации обеспечивает возможность распространения полученных проектов на алгоритмы поиска не только последовательные, но и параллельные [18].

10. **Создание базы знаний.** Интегральные схемы, вместе с интерпретациями, образуют базу знаний (БЗ), отражающую сущность выбранной предметной области (рис. 2). В данную БЗ входят также и тождества, квазитождества, соотношения, которые используются в процессе трансформации схем. Соответственно методу нисходящего проектирования переход от неинтерпретированных к частично интерпретированным схемам, а затем и к алгоритмам, может сочетаться с применением восходящего метода. При этом следует отметить возможность использования упомянутых ранее метаправил свертки, развертки, переинтерпретации и трансформации, что обычно связано с построением смежной предметной области (переход от сортировки к поиску). Это означает привлечение новых структур данных, соответствующих адекватной постановке задачи (массовость сортировки и поиска – выполнение не одного запроса на поиск, а их массива).

Таким образом, построение БЗ предполагает возможность её распространения вширь для охвата смежных задач, или предметных областей, наличие не только вертикальных связей (проектирование сверху-вниз, или в противоположном направлении), но и вдоль – образование для различных проектов их "слоенного пирога" с инкапсуляцией структур данных посредством применения необходимых структур памяти и пр. Тем самым, наряду с вертикальным и горизонтальным расслоением БЗ можно говорить и о третьем измерении, – вдоль, в частности, согласно выбранному критерию качества (например, быстродействия) осуществляется переход к распределённой мультиобработке на основе Grid технологий [19].

Отметим, что шаблонное (или схемное) проектирование объектов осуществимо по всем рассмотренным направлениям. При этом шаблонное (или схемное) проектирование весьма эффективно, при обработке, например n-отношений, по горизонталям, вертикалям, диогоналям и т.п. [20].

Способы "нарезания пирога", или стратегии реализации параллельных вычислений должны учитывать особенности среды, условия поставленной задачи, необходимое быстродействие, память и пр. Важно, в частности, при проектировании параллельных вычислений использовать, как "полуфабрикаты" (шаблоны или схемы), хранящиеся в предназначенных для этого разделах БЗ.

Иными словами, эффективность работы с БЗ тесно связана с организацией в ней удобных средств поиска и синтеза нового знания.

Рассмотренные ранее трехмерные методы построения БЗ могут быть связаны с использованием соответствующих теоретико-автоматных средств. Подчеркнём, что они могут оказаться необходимыми по всем, рассмотренным ранее, направлениям создания и пополнения БЗ.

2. Создание абстракций программирования

К числу основных понятий IP относится предметно-ориентированная абстракция намерений. Аналогом данного понятия может служить алгебраическая схема – суперпозиция операций в различных алге-

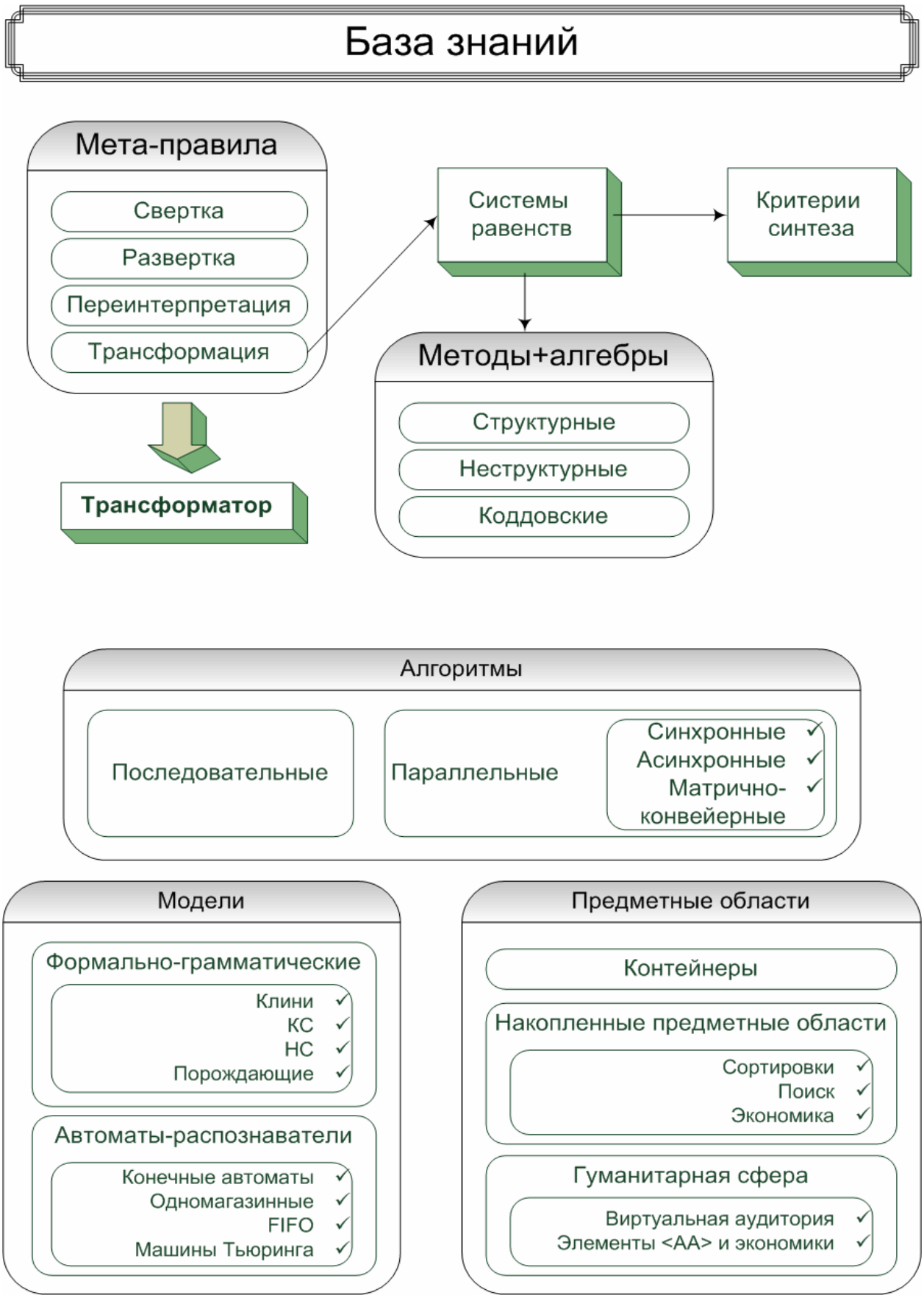

Рис. 2. База знаний предметной области

брах, связанных с той, или иной методологией или технологией программирования: автоматной, алгоритмической, АТД, АТП, расширениями ЯП и пр.

Проектирование и синтез объектов посредством алгебраических схем способствует решению ряда проблем, возникающих и при МП:

- формирование специализированных библиотек базовых понятий выбранной предметной области адекватно концепции родового программирования, обеспечивающего интерпретацию переменных, входящих в спроектированные и синтезируемые схемы;

- предметно-ориентированные расширения суть производные языковые конструкции, обеспечивающие ментальную фиксацию предметно-ориентированных абстракций, можно трактовать как создание соответствующих библиотек-разделов БЗ, состоящих из часто используемых в данной предметной области схемных проектов (суперпозиций сигнатурных операций заданной алгебры схем) – аналоги библиотек расширений в IP;

- указанные библиотеки расширений включают, в частности, эффективные для выбранной предметной области: оптимизирующие преобразования, методы тестирования и отладки, редактирования и пр.

- синтаксический анализ схем обеспечивается, во-первых, за счёт наличия в соответствующих языковых процессорах проектирования и синтеза анализаторов объектов, а во-вторых, за счёт диалоговых конструкторов синтаксически правильных (ДСП) схем, или деревьев их грамматического разбора. Следует отметить, что обеспечение синтаксической правильности распространяется на все взаимосвязанные формы представления объектов: аналитическую, текстовую и граф-схемную;

- создание синтезаторов объектов, библиотек расширений (абстракций) представляет самостоятельную ценность и ориентированно на применения в различных предметных областях. Библиотеки схем (абстракций) концентрируют взаимодополняемые модели и существенно расширяют возможности языковых процессоров, облегчают осмысление (распутывание) программного кода. Кроме того, алгебраический (схемный) подход формализует методы, технологии (в частности, ПИК-технологию, или многократное использование компонент), служит инструментом интеграции сред, способствует созданию и накоплению различных библиотек, входящих в состав БЗ.

Таким образом, алгебраический подход к проектированию и синтезу объектов адекватен основным достоинствам IP и обладает рядом существенных преимуществ:

а) алгебра – не только удобный, ясный, точный и компактный язык описания объектов, она ориентирована на формализованные преобразования объектов с целью их качественного улучшения по выбранным критериям (память, быстродействие, аппаратные ресурсы и пр.);

б) алгебраический формализм и базированные на нём инструменты ориентированны на множественное конструирование объектов (включая не только схемы, но языки входные, целевые и разнообразие различных предметных областей;

в) разработана алгебраическая теория клонов, в рамках которой соответственно известным методам проектирования объектов построены семейства родственных алгебр различных типов и предметной ориентации, базисных для языков спецификаций сверхвысокого уровня. В перспективе это означает, что прикладные программисты получат возможность создавать собственные удобные и адекватные решению поставленной задачи, а также среде реализации, языки проектирования и инструменты синтеза объектов. Тем самым для развиваемого алгебраического подхода характерны, в частности, концепции экологии намерений и эволюции биологических систем.

Заключение

Прокомментируем с позиций алгебраического проектирования объектов часто задаваемые вопросы в дискуссиях по IP.

Вопрос 1. Универсальные языки программирования изучены вдоль и поперек. Любую новую предметно-ориентированную нотацию, напротив, придется

тщательно разбирать. Не станет ли это непреодолимым препятствием к их распространению?

Процесс проникновения в сущность предметно-ориентированных нотаций может быть существенно облегчён, если оформление подобных нотаций сопровождать присущими алгебре алгоримики формами: формула, текст, граф-схема.

Подчеркнём, что наличие инструментария <АА> обеспечивает возможность эквивалентных преобразований указанных форм, а также синтеза предметно-ориентированных нотаций на соответствующих специализированных ЯП. При этом возможно их погружение в ОО-среды с применением и интеграции <АА> с UML и Rational Rose для погружения нотаций в ОО-среды, базирующиеся как на универсальных, так и специализированных ЯП. Тем самым, контейнеры в контексте родового и генерирующего концепций предполагает разработку разделов БЗ, содержащих библиотечные классы [20].

Вопрос 2. "Чем проще язык, тем легче им пользоваться и тем четче на нем получаются программы. Система IP пропагандирует использование многофункциональных языков." Не усложнят ли такие языки процесс и результат программирования?

Оценивая в плане сформулированного вопроса, алгебраические средства спецификации и синтеза объектов, можно утверждать, что подобные средства относятся к числу наиболее простых и многофункциональных.

Как и в случае IP цель состоит в том, чтобы иметь простые (отнюдь не примитивные) сверхвысокого уровня языки спецификации объектов, упрощающие процессы написания и синтеза объектов-компонент, необходимых для составления эффективных программ.

"Взять язык ассемблера – он прост, в нем не так уж много средств, однако процесс написания и сопровождения на нем сложных программ иногда оборачивается сплошной головоломкой. (Как сказал однажды Альберт Эйнштейн: нам нужна максимальная простота, и ничего проще!.)" Успешный опыт использования алгебраических средств спецификации и синтеза объектов представлены в [14] в связи с применением контейнерного подхода и инструментария синтеза последовательных и параллельных алгоритмов и программ символьной мультиобработки в ОО-средах (C++, Java и пр.).

Расширяемые среды программирования (наподобие IP и рассматриваемых алгебраических средств) лишь упрощают их применение, также предполагают отбор определенного набора абстракций, предназначенных для решения конкретной задачи. Они лишь упрощают этот процесс за счет наличия проблемно-ориентированных контейнеров, которые в универсальных ЯП, как правило, отсутствуют.

Вопрос 3. Если каждый программист получит возможность расширять язык, не наступит ли нотационный хаос?

Разработчики приложений, (работая в рамках IP и рассматриваемых алгебраических средств) , получают возможность одновременно задействовать те универсальные и предметно-ориентированные библиотеки абстракций, которые оптимальным образом обеспечивают выполнение поставленной задачи. Процесс составления абстракций языков (библиотек абстракций) существенно отличается от прикладного программирования. Он предполагает применение интерфейсов прикладного программирования абстракций, работу в рамках специальных протоколов IP, требует наличия определенных навыков, связанных с конструированием языков и реализацией.

Разработка библиотек абстракций должна стать прерогативой поставщиков библиотек, исключающей участие в этом процессе прикладных программистов.

Разработка предметно-ориентированных абстракций, по-видимому, должна происходить точно таким же образом, каким сегодня осуществляються конструирование стандартных, традиционных библиотек. (Во многих областях предметно-ориентированные абстракции уже существуют; задачей IP в этом контексте должна стать их реализация в виде допускающих совместное использование, встраиваемых наборов программных абстракций.) С по-

явлением рынка намерений разработчики получат в свое распоряжение высококачественные, многофункциональные предметно-ориентированные намерения, наличие которых, очевидно, исключит потребность в самостоятельной разработке индивидуальных, узкоспециализированных решений. Формирование рынка предметно-ориентированных абстракций приведет к строгой специализации, благодаря которой конструировать системы более высокого качества и сложности станет значительно проще.

Впрочем, наличие универсальной платформы для создания расширений языков поможет снять распространенную на сегодняшний день проблему – изолированных островков предметно-ориентированных и прикладных языков.

Вопрос 4. А как насчет способности библиотек абстракций к взаимодействию? Не приведет ли проблема взаимосвязи между расширениями (при которой каждое новое расширение способно нарушить структуру языка) к тому, что расширять языки программирования станет нерационально?

Используя возможность клонирования алгебраических и предметно-ориентированных ЯП получаем эффективный инструмент решения поставленной проблемы.

1. *Ноден П., Ноден К.* Алгебраическая алгоритмика, Издательство «Мир», 1999. – 720 с.
2. *Цейтлин Г.Е., Мохница А.С.* Что такое алгебраическая алгоритмика? // Проблемы программирования. – 2004, № 2-3. – С. 52–57.
3. *Чарнецки К., Айзнекер У.* Порождающее программирование: методы средства и приложения . – Питер, 2005. – 736 с.
4. *Цейтлин Г.Е.* Алгебраическая алгоритмика: теория и приложения // Кибернетика и системный анализ. – 2003. – № 1. – С. 8 – 18.
5. *Цейтлин Г.Е.* Введение в алгоритмику. – К.: Сфера, 1998. – 310 с.
6. *Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л.* Алгебра. Языки. Программирование. – Киев: 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. *Gluschkow W.M., Zeitlin G.E., Justchenko J.L.* Algebra. Sprachen. Programmierung. - Berlin: Akademie-Verlag, 1980. – 340 p.
8. *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.
9. *Яценко Е.А., Мохница А.С.* Инструментальные средства конструирования синтаксически правильных параллельных алгоритмов и программ // Проблемы программирования. Спец. выпуск по материалам 4-й Междунар. научн.-практич. конф. по программированию УкрПРОГ'2004. – К.: ИПС НАН Украины, 2004. – № 2-3. – С. 444 – 450.
10. *Яценко Е.А.* Алгебры гиперсхем и интегрированный инструментарий синтеза программ в современных объектно-ориентированных средах // Кибернетика и системный анализ. – 2004. – №1. – С. 47 – 52.
11. *Цейтлин Г.Е., Терзян Т.К., Захария Л.М.* Инструментарий конструирования экспертных систем символьной обработки // Математические машины и системы. – 1997. – № 1. – С. 14 – 25.
12. *Цейтлин Г.Е., Ющенко Е.Л.* Формализованные спецификации и трансформационный синтез программ // Кибернетика и системный анализ. – 1993. – № 1. – С. 127-152.
13. *Цейтлин Г.Е., Ющенко Е.Л.* Алгебра алгоритмов и граф-схемы Калужнина // Кибернетика и системный анализ. – 1994. – № 2. С. 3 – 16.
14. *Цейтлин Г.Е., Яценко Е.А.* Элементы алгебраической алгоритмики и объектно-ориентированный синтез параллельных программ // Математические машины и системы. – 2003. – № 2. – С. 64 - 76.
15. *Цейтлин Г. Е.* Проблема функциональной полноты в итеративных метаалгебрах // Кибернетика и системный анализ. – 1998. – № 2. – С. 28 – 45.
16. *Цейтлин Г.Е.* "Алгебры Глушкова и теория клонов" // Кибернетика и системный анализ. – 2003. – № 4. – С. 48 – 58.
17. *Post E.L.* The two-valued iterative systems of mathematical logic // Ann. Math. Studies. – 5- 1941. – P. 147.
18. *Бондарчук В.Г., Калужнин Л.А., Котов В.Н., Ромов Б.А.* Теория Галуа для алгебр Поста. – Ч.1, 2 // Кибернетика. – 1969. – № 3. – С. 1 – 10; № 5. – С. 1 – 9.

19. *Яценко Е.А.* Конструирование параллельных объектно-ориентированных программ // Проблемы программирования. Спец. выпуск по материалам 3-й Междунар. научн.-практич. конфер. по программированию УкрПРОГ'2002. – Киев: ИПС НАН Украины, 2002. – №1-2. – С. 188 – 197.
20. *Дорошенко А.Е., Алистратов О.В., Тырчак Ю.М., Розенблат А.П., Рухлис К.А.* Системы Grid – вычислений – перспектива для научных исследований // Проблемы программирования. – 2005. – № 1. – С. 14 – 38.

Получено 19.04.2007

Об авторах:

Дорошенко Анатолий Ефимович,
доктор физико-математических наук,
руководитель отдела,

Захария Любовь Михайловна,
кандидат физико-математических наук,
докторант,

Цейтлин Георгий Евсеевич, доктор физико-математических наук,
ведущий сотрудник.

Место работы авторов:

Институт программных систем
НАН Украины,
03187, Киев 187, Украина,
проспект Академика Глушкова, 40.
тел. 526 1538,
e-mail: dor@isofts.kiev.ua

Институт программных систем
НАН Украины,
Львов, ул. Кульпарковская 128/76,
тел. 8 032 2920203,
e-mail : zlm.lviv@gmail.com

Институт программных систем
НАН Украины, 100,
тел. 8 044 2574374,
e-mail: tseytlin@vikno.relc.com