УДК 681.3.06

В.П. КЛИМЕНКО, А.Л. ЛЯХОВ

ПРИКЛАДНАЯ МАТЕМАТИЧЕСКАЯ ЗАДАЧА КАК ОБЪЕКТ КОМПЬЮТЕРНОЙ АЛГЕБРЫ

Вступление

Существуют представительные классы так называемых «сложных задач компьютерной алгебры», в ходе решения которых функции, состоящие в анализе свойств данных, выборе способов их представления, преобразования и др., вынужденно выполняются человеком [1–3].

Такой режим работы систем компьютерной алгебры (СКА) находится в противоречии с естественным стремлением исследователей переходить к решению все более сложных задач, что стимулируется прогрессом компьютерной техники.

В истории появления и развития СКА как средств автоматизации численно-аналитических методов (ЧАМ) можно выделить этапы, на каждом из которых указанные трудности, постепенно нарастая, приобретают очертания проблемы, способной замедлять темпы развития прикладной науки, задаваемые ее повсеместной математизацией и компьютеризацией [2–5].

Путем анализа этих закономерностей в работах [1–3] из общего процесса автоматизации ЧАМ выделена составляющая – интеллектуализация как деятельность, направленная на частичное или полное исключение человека из процесса численно-аналитического решения путем передачи его интеллектуальных функций специально создаваемым программным средствам.

Необходимость интеллектуализации на каждом этапе обуславливается [3] развитием компьютерной техники и технологий, что воплощается в возможность достичь большей продуктивности решения сложных задач перераспределением интеллектуальных функций между человеком и компьютером.

Достаточность интеллектуализации для повышения продуктивности решения сложных задач определяется таким уровнем развития СКА, который позволяет создавать программное обеспечение (ПО), способное поддерживать новое распределение функций в процессе решения.

Таким образом, указанные выше этапы отражают периоды установившегося баланса между этими условиями и постепенного осознания существования новых классов задач с новым уровнем или фактором сложности, а смена этапов – качественное изменение свойств СКА, устраняющее факторы сложности предыдущего этапа и обеспечивающее большую относительную продуктивность ЧАМ. Такое понимание сложности задач компьютерной алгебры, на наш взгляд, вполне соответствует как объективной, так и субъективной сути общего понятия «сложности» [6, 7], и является его развитием.

В процессе эволюции СКА интеллектуализация программного обеспечения (ПО) осуществлялась последовательным развитием и сочетанием следующих качеств [1–4]:

– специализация данных;

– типизация данных;

– создание интерфейса, усиливающего интеллект человека при интерактивной обработке нетиповых данных;

– абстракция данных.

В результате анализа современных проблем в применении ЧАМ [3,4] установлено нарушение баланса необходимости и достаточности, достигнутого на предыдущем этапе благодаря появлению и развитию интерактивных средств. В настоящее время при усложнении решаемых задач продуктивность интерактивного режима быстро падает и преимущественно за счет увеличения затрат труда специалистов высокой квалифика-

ции – авторов задач, которым приходится непосредственно участвовать в процессе решения, выполняя рутинные интеллектуальные функции в роли оператора, что требует нового перераспределения функций между человеком и компьютером. При этом современный уровень развития компьютерной техники вполне обеспечивает выполнение необходимого условия такого перераспределения, но достаточное условие не выполняется. Трудности при создании автоматического ПО сложных задач имеют принципиальный характер, так как обусловлены неполнотой представлений о задаче как об объекте СКА, а также отсутствием единой точки зрения на способы решения этой проблемы.

Таким образом, представляется актуальной дальнейшая разработка представлений о задаче как об объекте СКА, которые являлись бы теоретической основой для разработки, и реализация входных языков, что в целом и обуславливает выполнение достаточного условия интеллектуализации.

В данной статье исследованы три взаимосвязанных аспекта этой тематики. В первой части разработаны методологические принципы интеллектуализации. Во второй и третьей частях, исходя из этих принципов, разработаны и апробированы представления о сложной задаче как об объекте языка СКА, которые могут быть теоретической и прикладной основой интеллектуализации ПО сложных задач на современном этапе.

Часть I. Методологические принципы интеллектуализации решения сложных задач

I. 1. Исходные положения

Методологические принципы, на наш взгляд, естественно вывести из природы самой задачи, как явления прикладной науки.

Сходство общего метода решения сложных задач компьютерной алгебры и задач искусственного интеллекта, установленное в [4], обосновывает возможность применения к анализу и разработке представлений о задаче теоретико-множественного подхода [8], единого для описания процесса решения и распознавания свойств объектов.

Известным представлениям о научной и прикладной задаче [2, 6, 8–13], а также распространенной практике применения СКА, не противоречат следующие исходные положения:

1. Под задачей в данной работе понимается прикладная математическая задача «на нахождение», которая возникает и решается численно-аналитическим методом при моделировании некоторой научной или инженерной проблемы.

2. Задача объективна и осознается субъектом как данный объект.

3. Для представления данных о задаче в процессе решения используется целесообразная, с точки зрения исследователя предметной области, совокупность Φ выражений разнообразных разделов классической или современной математики.

4. Хотя совокупность выражений, описывающих данные, построена по определенным правилам, их вид и структура детерминированы не только этими правилами, а и так называемыми отношениями φ феноменологической симметрии [14]. Эти отношения являются воплощением условия задачи и более фундаментальных и глубоких закономерностей в исследуемой прикладной области и в природе. Значит, эту совокупность выражений можно считать некоторым данным (естественным) языком, а не порожденной системой.

5. Осознанная и корректно поставленная задача имеет решение.

6. Процесс нахождения решения задачи может быть представлен в виде алфавитного отображения [15]:

$$f : \Phi_0 \to \Phi_N , \tag{1}$$

а каждый шаг решения также является алфавитным отображением:

$$f_i : \Phi_{i-1} \to \Phi_i , \quad (2)$$

где $i = \overline{1, N}$ – конечное число шагов решения; $\Phi = \{\Phi_0, \Phi_1, ..., \Phi_N\}$ – последовательность из множеств выражений на конечном словаре Γ, каждое из которых состоит из выражений

$$\Phi_i = \{\alpha_i\}, \quad (3)$$

связывающих имена начальных, промежуточных и неизвестных атомарных данных; Φ_0, Φ_N – поставленная и решенная задача; $f = \{f_i\}$ – последовательность функций.

7. На каждом шаге i решения множество выражений Φ_{i-1} содержит всю информацию, необходимую для выбора и выполнения преобразований $f_i(\alpha_{i-1}) = \alpha_i$, где выражения $\alpha_{i-1} \in \Phi_{i-1}, \alpha_i \in \Phi_i$.

Из сформулированных положений и известной теоремы [8] следует, что для построения процесса решения путем проверки соотношения

$$\alpha \in s_{i-1}, \quad (4)$$

где s_{i-1} классы из универсума на Γ, выбора и применения соответствующего отображения f_i достаточно иметь описание класса s_{i-1}.

I. 2. Анализ подходов к представлению данных входными языками современных СКА при решении сложных задач

В общем случае объектом входного языка СКА является вся совокупность Φ данных о задаче. Полнота входного языка означает [2] наличие аппарата, дающего возможность пользователю самому описывать в программе данные конкретной задачи в процессе ее решения вместе с операциями над ними. Как и было предвидено в работах [2,16], таким аппаратом в настоящее время становятся абстрактные типы данных (АТД). Это мощная методология, поддерживающая модульность и снижающая общую сложность программ. Им обладают все современные универсальные СКА, однако направления и уровень его развития отличаются.

В общем случае АТД можно представить в виде объекта, инкапсулирующего некоторый тип структуры данных и операций (преобразований) f над данными этого типа, а также процедур, позволяющих создавать такие объекты. В развитии АТД современных СКА выделим два подхода.

I. 2.1. Конструктивный подход

На уровне реализации входного языка СКА определены стандартные типы данных [4]. Каждому такому типу данных во внешнем представлении входного языка соответствует класс выражений, семантика которых очевидна (числа, полиномы, тензоры и др.). Свойства (атрибуты) этих объектов именуются словами из специального класса Γ_S словаря входного языка. Сложный объект является агрегацией стандартных типов. Он наследует свойства и структуру стандартных типов данных, которыми порожден. Его свойства s именуются выражением, которое выводится по определенным грамматическим правилам [17–20] из элементов Γ_S. Это выражение интерактивно создается пользователем или генерируется ядром СКА в процессе автоматической проверки соотношения (4), где теперь s – выражение языка Γ_S, а α_{i-1} – выражение-прообраз для f_i. При этом существует изоморфизм между структурой агрегата и выражением, именующим его свойства. По способу

построения объектов и именования свойств этот подход к созданию АТД можно назвать «конструктивным». Появился он в наиболее осознанном виде, по-видимому, в СКА SCRATCHPAD/1 [21] и в настоящее время поддерживается всеми ведущими СКА, а наибольшее развитие получил в AXIOM [17] – современной версии упомянутой системы.

В работе [22] показано, что множество всех языков над конечным словарем Γ неперечислимо, в то же время языки на основе порождающей грамматики над этим же словарем пересчитать можно. Если исходить из сформулированных выше положений, то для языка Φ, представляющего задачу в процессе решения (1) – (4), существование такой грамматики в общем случае является проблематичным (обычно этот вопрос даже не ставится).

Проблематичной в общем случае является и возможность проверки соотношения (4). Даже если предположить, что свойство $S \subset \Phi$ любого данного объекта распознаваемо в языке, порожденном некоторой грамматикой на Γ, то, как известно, оно не распознаваемо в классе всех грамматик на этом же словаре и, следовательно, может быть не распознаваемо в языке СКА, выбранной для решения задачи.

Таким образом, имеет место

Утверждение 1: В общем случае представление решения сложной задачи в виде (1) – (4) языком на основе порождающей грамматики содержит «относительную алгоритмическую проблему» [3].

Существенным для расширения области применения ЧАМ является и то, что область применения АТД, создаваемых на конструктивной основе, определяется мощностью специального словаря Γ_S входного языка. Следовательно, расширение области применения таких АТД требует соответствующего расширения словаря и модификации ядра СКА. Это связано с вполне определенными трудностями [1–3] как при разработке СКА, так и при их использовании (в ведущих СКА [17–20] число стандартных типов исчисляется сотнями).

I. 2.2. Аналитический подход

Альтернативным в разработке АТД является подход, при котором выражением, именующим свойство S, может быть любое выражение универсального языка на Γ, обладающее таким свойством. Свойства и правила преобразования объектов абстрагируются пользователем непосредственно при анализе данных, исходя из отношений, связывающих эти данные в процессе решения, что и определило предлагаемое для подхода название. Таким образом, областью определения АТД, которые создаются на основе аналитического подхода, является весь универсум. Реально область определения ограничивается уровнем развития процедуры, осуществляющей проверку соотношения (4).

Аналитический подход в той или иной мере поддерживается всеми ведущими СКА [4]. К настоящему времени наибольшее развитие он получил в языках семейства АНАЛИТИК, для которых является доминантой.

Вместе с тем на основе анализа входных языков СКА [4] можно сделать вывод, что к настоящему времени наиболее полное развитие получил аппарат АТД, объектом которого является отделенное выражение, что не позволяет [4, 23] решать проблемы, связанные со взрывным ростом объема данных. Возможности обобщения этого подхода только исследуются [3, 4, 23 –25]. В целом же ему также свойственны стихийность и недостаточность обоснования [4].

Таким образом, представляют интерес исследования, направленные на дальнейшее развитие и применение аналитического подхода для разработки представлений о свойствах входных языков универсальных СКА, предназначенных для решения сложных задач.

I. 3. Принципы интеллектуализации ПО сложных задач

Термин «абстрактные типы данных» использован выше для установления связи аналитического подхода с

общей тенденцией в реализации входных языков СКА, отмеченной в работе [2, 4] и направленной на развитие общности языков представления данных при численно-аналитическом решении задач.

С точки зрения структуры языка представления, входной язык СКА при конструктивном подходе поставляет средства, совокупность которых соответствует известным определениям языка на основе порождающей грамматики, например, [26]:

$$(\Gamma, \mathrm{P}, I, f), \tag{5}$$

где $\mathrm{P} = \{\Gamma, \Gamma \setminus \Gamma_s\}$ – разбиение словаря; I – начальный символ; f – правила вывода.

Язык (5) более высокого уровня, чем входной язык СКА, т.е. по отношению к нему является гиперязыком. Таким образом, из утверждения 1 следует, что входной язык, предназначенный для решения сложных задач, должен также содержать набор средств для надстройки гиперязыка на неконструктивной основе.

Неконструктивные теоретико-множественные подходы [26] характерны для структурной, в частности, дескриптивной лингвистики, однако исходный принцип вполне соответствует положениям, изложенным в п.2. При подходе (5) исходным пунктом является грамматика, порождающая язык, а объектом исследования – порожденный ею язык. Теоретико-множественный подход обратный. Тут исходным материалом служит некоторая совокупность выражений. Цель исследования состоит в том, чтобы путем анализа выявить характер отношений между выражениями, а потом порождаемую ими структуру и составляющие элементы.

Для обоснования, дальнейшего развития и обобщения аналитического подхода предлагаются следующие принципы, суммирующие изложенное выше, а также результаты, полученные в [23]:

1. Задача объективна. Выражение в (1)– (4) является записью правила вычисления своего значения. Однако его структура, а также структура всей совокупности Φ, является отражением некоторых феноменологических отношений φ между данными. Анализ отношений между объектами сложной задачи в процессе ее решения необходимо производить на неконструктивной (канторовской) теоретико-множественной основе.

2. Входной язык универсальной СКА, предназначенной для решения сложных задач, должен быть открытым для пользователя и содержать подмножество средств для надстройки гиперязыка для решения конкретной задачи.

3. Язык представления данных о сложной задаче необходимо создавать на основе аналитической грамматики, что соответствует принятому теоретико-множественному подходу. Сложная задача в процессе (1) – (2) рассматривается как некоторое множество Φ выражений универсального языка на словаре Γ, которые содержат имена начальных, промежуточных и неизвестных объектов. Интеллект наделяет множество Φ структурой языка путем анализа отношений между выражениями и их составными частями.

4. Гиперязык имеет следующую структурную четверку:

$$(\Gamma, \mathrm{P}, \Phi, \Re), \tag{6}$$

где Γ – словарь языка; P – разбиение словаря; Φ – множество выражений задачи на всех этапах процесса решения; $\Re$ – категория языка вида $(s(\varphi), F)$; $s(\varphi)$ – класс объектов; F – соответствующий класс морфизмов; φ – феноменологическое отношение, образующее категорию.

5. Множества выражений в (1) – (4) рассматриваются как выражения некоторого языка (6) на аналитической основе над входным языком СКА.

6. Отношения φ абстрагируются пользователем при анализе множества Φ как языка.

7. Структура языка (6) динамическая и может обогащаться за счет добавления новых категорий

$$\Re = ((s_1, F_1), (s_2, F_2), \ldots, (s_n, F_n)), \tag{7}$$

которые формируются, исходя из свойств, абстрагированных при анализе конкретной сложной задачи.

8. Входной язык СКА является метаязыком по отношению к языку на аналитической основе. Естественным для аналитического подхода является следующее: четверка (6) определяет состав базисных процедур как компоненту сигнатуры входного языка СКА.

9. Базисными средствами для создания гиперязыка является алфавит входного языка СКА, словарь Γ и его разбиение P, а также процедуры для расширения или изменения Γ и P. Особое значение имеют базисные процедуры для представления категорий: для именования классов объектов, для проверки (4) принадлежности некоторого объекта заданному классу s и для осуществления алфавитных отображений (2).

10. Реализация базисных процедур должна обеспечивать выполнение аксиом для основных понятий – класс, категория, множество и т.д., – фундаментальных для понятия «язык на аналитической основе».

11. Базисные средства входного языка СКА должны быть ориентированы на сохранение компактности представления данных в процессе решения задачи.

Для обоснования полученного обобщения аналитического подхода к представлению данных необходимо доказать существование языка на основе аналитической грамматики, множеством выражений которого является совокупность Φ в (1) – (2), и исследовать ее структуру в соответствии с принятыми принципами.

Часть II. Теоретико-множественная модель задачи

II. 1. Общие положения

Теоретико-множественную модель построим как обобщение отношений между данными на протяжении всего процесса решения задачи на нахождение.

Пусть Φ – множество выражений из (1) – (2). Данные находятся в некоторых феноменологических отношениях. При разработке обобщенных представлений задачи как объекта языка универсальных СКА будем исходить не из некоторых конкретных типов таких отношений, свойственных объектам той или иной задачи, а из предположения, что они существуют и в наиболее обобщенном виде. На основе анализа понятия “задача” [6, 8–11] считаем, что ее сути в наибольшей мере соответствуют отношения зависимости между данными [27]. Это отношение определяется условием задачи, существует на любом шаге (2) процесса решения и, таким образом, является отношением феноменологической симметрии. Отметим, что функциональная зависимость является частным случаем таких отношений.

При этом принимаем, что:

1. Выражения из Φ в (1) – (2) образованы конечными последовательностями на словаре Γ с разбиением

$$\Gamma\,' = \Gamma \setminus \Gamma\,'', \tag{8}$$

где $\Gamma\,'$ – имена операций; $\Gamma\,''$ – имена данных (начальных $\Gamma_0\,''$, промежуточных и неизвестных $\Gamma\,''\setminus\Gamma_0\,''$).

В выражении существует главная (последняя) с именем I и единичная (тождественная) операция.

2. Каждой операции можно поставить в однозначное соответствие множество ее операндов, каждому выражению $\alpha_i \in \Phi_i$ – систему O_{α_i} таких множеств, множеству Φ_i – систему O_i, а Φ – соответствует система

$$O = \{O_i\} \subset 2^{\Gamma\,''}, \tag{9}$$

где $2^{\Gamma\,''}$ – булеан множества $\Gamma\,''$.

3. Решение Φ_N представляется системой O_N, в которой элементы имеют вид

$$O_N = \{o_g \cup g\}, \tag{10}$$

где $g \in \Gamma''\backslash\Gamma_0''$ – имена атомарных данных неизвестных или промежуточных; $\{o_g\} \subset 2^{\Gamma_0''}$.

4. Каждая система O_i покрывает $\Gamma''\backslash\Gamma_0''$. Система O_g покрывает Γ_0''.

II. 2. Исследование структуры данных, обусловленной отношением зависимости

Отождествим в полученной системе O равные между собой множества (одноэлементные тоже) и, таким образом, абстрагируемся от структуры на Φ, связанной с порядком выполнения операций при вычислении значений выражений.

Согласно п.4, в системе O выполняется

$$\begin{cases} \forall\alpha \mid \boldsymbol{O}_\alpha \subset \mathbf{2}^{\Gamma''} \neq \varnothing; \\ \forall\alpha\exists\beta \neq \alpha \mid \boldsymbol{O}_\alpha \bigcap \boldsymbol{O}_\beta \neq \varnothing. \end{cases} \tag{11}$$

Добавим к системе O множество имен $g \in \Gamma''\backslash\Gamma_0''$. Множество Γ_0'' считаем независимым. Имеет место

Утверждение 2: Множество Γ_0'' порождает на Γ'' отношение зависимости.

Доказательство: Поскольку $O_N \subset O$, то в соответствии с п.3, имена из $\Gamma''\backslash\Gamma_0''$ зависят от Γ_0'', а, в соответствии с п.п.2 и 4, принадлежат любому выражению из Φ. Таким образом, в соответствии с [27], отношение абстрактной зависимости на Γ'' задается всей системой множеств O, которая соответствует множеству Φ выражений задачи, и порождено Γ_0'', поскольку выполняется

$$\forall\alpha \in \Phi \mid O_\alpha \bigcap O_N \neq \varnothing. \tag{12}$$

В этом смысле совокупность данных Φ следует считать целостным объектом, причем в процессе (2) решения зависимая система $O_{\alpha_{i-1}}$ отображается на зависимую O_{α_i}.

Характер структуры множества Φ, обусловленной отношением зависимости, определяет

Лемма 1: Схема отношений в множестве Φ, обусловленная абстрактным отношением зависимости между именами данных, изоморфна абстрактному комплексу K размерности m, где $m+1$ – количество разных имен в $\Gamma''\backslash\Gamma_0''$.

Доказательство: Если одноэлементные пересечения в (11), принадлежащие $\Gamma''\backslash\Gamma_0''$, считать полем вершин, а элементы O_{α_i} системы O и их не пустые пересечения из (11) – симплексами, то полученный объект соответствует определению абстрактного комплекса [28]. Его размерность определяется соответственно. Понятно, что совокупности Φ_i в (2) на каждом шаге локальных алфавитных отображений f_i изоморфны симплексам K_i комплекса K.

II. 3. Доказательство существования языка представления данных на основе аналитической грамматики

Поскольку существование порождающей грамматики на Φ не предполагается, то из существования решения задачи следует только то, что на каждом шаге локальных алфавитных преобразований в (2) существуют прообразы и образы отображений f_i. Существование языка (6) требует доказательства.

Теорема: Если на каждом шаге в (2) существуют множества выражений Φ_i с отношением зависимости (10) – (11), то существует язык на аналитической основе для представления данных в процессе (2) решения задачи.

Доказательство: Пусть есть разбиение (6) словаря Г, а из существования решения следует существование цепи алфавитных преобразований f_i и цепи из множеств Φ_i. Для существования языка (6) осталось доказать существование категорий, образованных феноменологическим отношением φ.

Из леммы 1 следует, что цепи $f = \{f_i\}$ алфавитных преобразований соответствует изоморфная цепь отображений комплекса K_{i-1} в комплекс K_i.

Если отображение f_i осуществляется с сохранением отношения зависимости и выполняется п. 4, то, согласно (10) и (11), имеет место

$$\forall g \in \Gamma'' \setminus \Gamma_0'' \quad \exists O_{\alpha_0} \cap .... \cap O_{\alpha_{i-1}} \cap ... O_{\alpha_N} \cap S = g \,. \tag{13}$$

То есть между остовами существует такое соответствие, что каждое f_i оказывается отображением всего комплекса K_{i-1} на весь комплекс K_i. Тогда множество комплексов K_i и отображений f_i образуют, соответственно, класс объектов и класс морфизмов, а пара (K, f) – категорию [27].

Существование языка доказано.

II. 4. Пространственные отношения на множестве выражений задачи в процессе решения

Представления о задаче как о целостном пространственном объекте, изложенные в работах [16, 24] и лежащие в основе развития АТД СКА АНАЛИТИК-93, -2000 [29, 30], имеют интуитивную и эмпирическую основу. Реализация этих представлений в виде АТД входных языков СКА нового поколения, в частности, новых версий семьи АНАЛИТИК, предназначенных для интеллектуализации программного обеспечения, требуют их последовательного теоретического обоснования и уточнения.

Обоснование следует из леммы 1 и теоремы в виде

Утверждения 3: На множестве Φ выражений задачи существуют пространственные метрические отношения.

Доказательство: Согласно известной теореме о реализации [31], каждому абстрактному симплексу размерности m в метрическом пространстве размерности $2m+1$ однозначно соответствует геометрический комплекс той же размерности.

Утверждение 4: На множестве выражений задачи существуют пространственные топологические отношения.

Доказательство: Каждому симплициальному комплексу [28] соответствует дискретное топологическое пространство. Морфизмы (2), построенные при доказательстве теоремы, являются симплициальными отображениями

$$f_i : K_{i-1} \to K_i$$

и, как следствие, непрерывными отображениями соответствующих топологических пространств одного в другое. Поскольку эти комплексы являются симплексами комплекса K, изоморфного Φ, отображение (2) также является симплициальным. Таким образом, процесс решения задачи можно представить как непрерывное отображение связанного с ней топологического пространства в себя.

Доказанные утверждения являются теоретическим обоснованием парадигмы, на которой в языках АНАЛИТИК-93,-2000 создаются АТД, отсутствующие в настоящее время в других СКА. На множестве выражений задачи в процессе решения действительно существуют пространственные отношения между данными, однако устанавливают их топологическую, а не метрическую природу.

II. 5. Необходимое условие реализации базисных процедур для представления решения сложных задач языком на основе аналитической грамматики

Решение задачи является непрерывным процессом (утверждение 4). Таким образом, входной язык СКА должен иметь соответствующую семантику для описания такого процесса и направления автоматических преобразований к решению. Возможный способ установления такой семантики следует из п. II.1, доказанной теоремы и утверждения 4.

Следствие: Представление решения сложных задач языком на аналитической основе предполагает такую реализацию языка, при которой базисной процедурой входного языка СКА осуществляют алфавитные отображения f_i в (2) с сохранением отношения зависимости.

Семантическое правило можно сформулировать в виде следующего необходимого условия.

Условие: Выражение входного языка СКА, которое описывает заголовок базисной процедуры, предназначенной для проверки (4) принадлежности выражения универсального языка классу объектов, должен содержать список имен зависимых данных.

При невыполнении этого условия входной язык СКА не приспособлен в общем случае для описания непрерывных автоматических алфавитных отображений (2) на классах объектов. Представление автоматических преобразований может стать неоднозначным [32], а выбранный программой элемент морфизма может нарушить отношение зависимости и вывести образ за пределы класса объектов даже в тривиальных случаях.

Пример. Необходимо решить квадратное уравнение

$$2y^2 + 3yxz - 3x^2 + z^2 = 0\,. \qquad (14)$$

При аналитическом подходе элементы класса объектов именуются выражениями

$$\Phi_1 : at^2 + bt + c = 0\,; \quad \Phi_2 : t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}\,.$$

Однако без указания на зависимую переменную в (14) в конечном выражении соответствие

$$\Phi_1 \overset{f_2}{\rightarrow} \Phi_2 \qquad (2')$$

неоднозначно. Автоматическая программа перейдет в режим диалога или будет действовать в соответствии с принятым в языке лексикографическим упорядочиванием.

Часть III. Структуры данных на сцепленном множестве выражений

На современном этапе сложность задач проявляется в дуализме двух факторов [2–4]. Первый – «относительная алгоритмическая проблема» – рассмотрен выше. Второй состоит в превышении объема данных о задаче

некоторого предела, за которым продуктивность выполнения интеллектуальных функций путем визуального анализа данных человеком резко уменьшается в силу естественных причин.

В обзоре [4] показано, что объектом входного языка широко используемых систем компьютерной алгебры (СКА) является выражение: отделенная семантическая единица описания совокупности данных о задаче в процессе ее решения, и что на основе таких представлений о структуре данных устранить этот фактор сложности нельзя. Отношения, отражающие суть условия задачи и более глубокие закономерности исследуемой предметной области, при автоматическом решении становятся доступными для анализа и обработки только после выполнения всех подстановок.

Для ослабления его влияния современные СКА оснащены аппаратом управления режимами выполнения операций, в частности, процессом подстановок. Во входных языках хорошо известных СКА MAPLE V, MATHEMATICA, AXIOM [17–19] и др. это так называемые инертные формы математических функций, специальные атрибуты объектов и набор операторов присвоения для локального или глобального управления процессом преобразований. Наиболее полно аппарат управления реализован в последних версиях языков АНАЛИТИК [29,30].

Подробный обзор средств управления имеется в [4]. Среди них нет процедур для автоматической выработки критериев к такому управлению. Таким образом, в настоящее время управление преобразованиями возможно, но с помощью эвристических критериев или интерактивно, что снова приводит к необходимости визуального анализа свойств громоздких объектов в процессе решения.

Проблему повышения продуктивности ЧАМ для решения сложных задач можно рассматривать в следующих аспектах:

1. Развитие представлений о структуре данных.

2. Развитие представлений об объекте языка СКА и о наборе процедур для его обработки.

3. Разработка на основе п.п. 1– 2 ЧАМ, ориентированных на автоматическое решение сложных задач.

4. Развитие искусственного интеллекта СКА и, в частности, интерфейса «пользователь ↔ СКА».

5. Разработка критериев оценки сложности объектов и процедур, поставляющих в процессе решения задачи информацию, нужную для управления преобразованиями.

6. Разработка методов и стратегий интерактивного и автоматического управления преобразованиями с помощью имеющегося аппарата и вновь созданных средств.

В данной статье разрабатываются вопросы, относящиеся к п.п. 1 – 3.

III. 1. Структура данных о задаче

Класс математических задач, возникающих при моделировании разнообразных прикладных проблем, достаточно полно описан в [6, 9 – 11], применительно к задачам искусственного интеллекта – в [8, 33–35], а к компьютерной алгебре – в [3, 36–39] и в части I этой статьи. Как правило, такие задачи имеют решение и состоят в нахождении неизвестных.

В общем случае на каждом шаге процесса решения данные Φ_i о задаче можно записать в виде конечной совокупности

$$\Phi_i = \{\alpha_i\}. \qquad (3')$$

Выражения в (3') могут быть предикатами. Такая форма записи данных определена в MAPLE V:

> **s:=R^2 = x^2+y^2+z^2: type(s, relation); s1:= -1=-1: s:=s+s1;**

true

$$R^2 - 1 = x^2 + y^2 + z^2 - 1.$$

В языках АНАЛИТИК, MATHEMATICA и др. такая запись возможна, однако выражения, содержащие

символы =, <, >, ≠ и др., отнесены только к типу «логический», что является существенным семантическим ограничением, в частности, при дальнейших обобщениях понятия «объект языка СКА». Для АНАЛИТИК-93, -2000 (MATHEMATICA – аналогично), к примеру, имеем

s:= R**2 = x**2 + y**2 + z**2;

sl:= s ? АНАЛ → 0 (false)

sl:= s ? ЛОГ → 1 (true)

s1: -1=-1; s: : s + s1 → (-1 = -1) + R**2 = x** + y**2 + z**2.

Выражение в языках всех современных СКА рассматривается как иерархическая структура. Процедуры для обработки таких объектов у них практически одинаковые [4]. Далее подобные подструктуры в (3') не рассматриваются, т.к. целью этой статьи в большей мере являются отношения между выражениями.

III. 1.1. Объект языков MAPLE V, MATHEMATICA, AXIOM

С точки зрения представлений, лежащих в основе реализации широко известных СКА [17–20], данные (3') – это множество отделенных выражений различных типов, на котором заданы подстановки и которое упорядочено очередностью их выполнения. В силу принципиальной важности уточним условие отделимости выражений: все имена, входящие в выражения, считаются разными, даже если они имеют одинаковое написание.

III. 1.2. Объект языка АНАЛИТИК

Дальнейшим развитием этих представлений является объект языков АНАЛИТИК-91, -93, -2000 [16, 29, 30].

III. 1.2.1. Поскольку подстановка является операцией, то структура данных над множеством всех операций в (3') , при условии отделимости (п. III.1.1), может быть представлена ориентированным деревом [4, 39]. Программа получает доступ к заданному операнду в совокупности (3') без предварительного выполнения подстановок – ссылкой на его координаты (с учетом структуры выражений) относительно корня дерева этой структуры.

III. 1.2.2. Последние версии языка АНАЛИТИК [29, 30] содержат элементы реализации более сложных представлений [4, 16, 23, 24, 39] о данных – без условия отделимости выражений. Объектом языка является древовидная ориентированная структура над Φ со «склеенными» вершинами, которым соответствуют одинаковые операнды различных выражений или различных частей выражений.

Объект определен в [24] как «многомерная» сеть. Такое название неудачное, поскольку не отражает интуитивно вложенной сути. Сеть всегда может быть уложена в трехмерное пространство [40]. Описанный объект имеет иную связность и будет определен ниже.

III. 1.3. Структуры данных на сцепленном множестве выражений

Непрерывный процесс решения задачи, от постановки и до получения результата, принято делить на этапы [6, 9–11]. Изложенные представления соответствуют структуре, приобретаемой данными на последнем этапе, синтеза неизвестных, на котором обычно начинается применение СКА, и во многом имеют эмпирическую основу.

На наш взгляд, не лишены смысла исследования, целью которых является изучение структур данных в (3') дедуктивным методом. Такой подход содержит возможность развития и дальнейших обобщений понятия «объект языка СКА», а также распространения методов компьютерной алгебры на новые классы сложных за-

дач и на более ранние этапы решения, в частности, на этап анализа данных.

Будем исходить только из того, что совокупность (3') соответствует некоторому шагу решения и содержит достаточно информации для нахождения неизвестных путем преобразований выражений α_i.

Поставим в однозначное соответствие каждой операции, выполняемой при вычислении значения выражения (в том числе и символу отношения), множество ее операндов, а каждому выражению $\alpha_i \in \Phi_i$ в (3') – совокупность O_{α_i} таких множеств. Тогда всей совокупности данных Φ_i соответствует

$$O_i = \{O_{\alpha_i}\} \subset 2^{\Gamma''}, \tag{15}$$

где $2^{\Gamma''}$ – булеан этого множества.

Общая схема отношений в O_i отражает и ассоциативные связи в Φ_i, которые объект не образуют [41]. Объект образует [14, 41, 42] некоторое заданное феноменологическое отношение или же свойство, которое отличает Φ_i как совокупность данных некоторой задачи от произвольной совокупности выражений подобного вида. Эти способы построения объекта являются двойственными и дополняющими друг друга.

III. 1.3.1. В качестве «образующего» отношения в первой части статьи использовано отношение абстрактной зависимости на Γ'' и показано (лемма 1), что структуру данных задачи, обусловленную этим отношением, можно представить в виде полного абстрактного комплекса над полем вершин из имен промежуточных и неизвестных данных. Такой подход открывает определенные перспективы для исследования задачи как объекта, закономерностей процесса ее решения, а также свойств языков для программирования автоматических ЧАМ, но не обладает полнотой, поскольку требует абстракции от структуры, определенной порядком выполнения операций.

III. 1.3.2. Обобщенными свойствами данных Φ_i о задаче является сцепленность и замкнутость системы множеств O_i относительно операции «пересечение». Эти свойства – проявление полноты данных и их семантики как записи процесса преобразований в Φ_i.

Присвоим каждому выражению α_i из (3') имя e_i из некоторого поля вершин[1]. Тогда структуру данных, порожденную этими свойствами над множеством операций в Φ_i, можно представить [4, 39] в виде полного абстрактного комплекса над полем вершин $\{e_i\}$ – «нерва» системы O_i. Выборки $\{e_{i_1}, e_{i_2}, ..., e_{i_n}\} \subset \{e_i\}$ именуют остовы «нерва» (возможно, кратные), для которых

$$O_{\alpha_{i_1}} \cap O_{\alpha_{i_2}} \cap ... \cap O_{\alpha_{i_n}} = O_{\alpha_{i_1 i_2 ... i_n}} \in O_i, \tag{16}$$

где $O_{\alpha_{i_1 i_2 ... i_n}}$ – грани нерва размерности $n-1$, которым соответствуют операнды, общие для различных операций, выполняемых при вычислении выражений из (3').

Построенный таким образом «нерв» для (3') не единственный. «Нервы» с более «тонкой» структурой можно получить следующим образом: в поле вершин следует последовательно включать не только выраже-

[1] Полем вершин могут быть элементы любой природы, в том числе и сами выражения. Однако тут не исключается случай, когда одно и то же выражение может фигурировать под разными именами, что может быть полезным в дальнейшем.

ния (3'), а и операнды $O_{\alpha_{i_1 i_2 \ldots i_n}}$, имеющие между собой непустые пересечения, подобно (16). Таким образом, приходим к выводу, что множество операций в (3') имеет иерархическую структуру, поскольку при дополнении поля вершин указанным способом каждый предыдущий нерв будет собственной гранью предыдущего. Однако, эта иерархия более сложная, чем допускает «сеть» (п.III.1.2), поскольку пространство реализации «нерва» может иметь размерность $n-1 \geq 3$. В частности, для приведенного в работе [24] примера размерность «нерва» равна четырем и для его изображения потребовалось бы, соответственно, пять трехмерных проекций.

В качестве более простого для изображения примера используем математическую модель кинематической схемы реальной инженерной конструкции [43]. «Нерв», отражающий все связи на множестве операций для этой модели, достаточно сложен. В данной работе используем симплекс (рис. 1) этого «нерва», представляющий неявную зависимость $R(a)$:

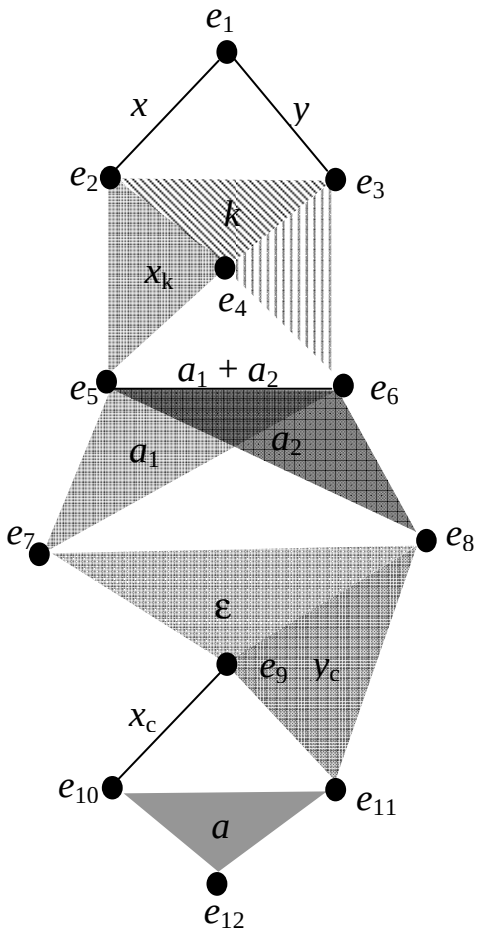

Рис. 1. Симплекс "нерва" множества выражений, описывающих кинематическую схему конструкции

$$e_1 : (b_1 - x)^2 + y^2 = R^2;\ e_2 : \frac{x - x_k}{x_k + b_6} = \frac{b_8}{k};$$

$$e_3 : \frac{y_k - y}{y_k} = \frac{b_8}{k};\ e_4 : (b_6 - x_k)^2 + y_k^2 = k^2;$$

$$e_5 : \frac{x_k}{b_4} = \sin(a_1 + a_2);$$

$$e_6 : \frac{b_1 - y_k}{b_4} = \cos(a_1 + a_2); \qquad (17)$$

$$e_7 : \frac{b_3^2 + \varepsilon^2 - b_7^2}{2 b_3 \varepsilon} = \cos a_1;$$

$$e_8 : \frac{b_1 - y_c}{\varepsilon} = \cos a_2;\ e_9 : x_c^2 + (b_1 - y_c)^2 = \varepsilon^2;$$

$$e_{10} : \frac{x_c - b_2}{r} = \cos(a);\ e_{11} : \frac{b_8 - y_c}{r} = \sin(a);$$

$$e_{12} : a\,.$$

Тоновой штриховкой показаны двумерные грани. Размерность «нерва» данных (17) равна двум.

«Нерв» как абстрактный комплекс является частично-упорядоченным множеством своих остовов. Таким образом, характер связности «нерва» принципиально допускает возможность автоматического [44] выполнения операций на сцепленном множестве выражений (3') без рекурсивных подстановок следующими способами:

1. Преобразование отдельных выражений.

2. Преобразование операнда, соответствующего некоторой грани «нерва», например, вида (16), с подстановкой полученного значения только в остовы этой грани. Частным случаем подобных преобразований является оптимизация символьных и числовых вычислений значения одного выражения, широко используемая в СКА (процедура ОПТИМИЗАЦИЯ языков АНАЛИТИК и процедуры других СКА, применяемые по умолча-

нию), а также в оптимизирующих компиляторах различных систем числового программирования.

3. Менее очевидной выглядит возможность вычисления значения всего множества выражений (3'), исходя из установленных связей. Однако, во-первых, «нерв» является схемой отношений на этом целостном множестве и, следовательно, можно говорить о значении [45] множества выражений. Во-вторых, эта возможность принципиально существует, поскольку «нерв» является частично упорядоченным множеством своих остовов, каждому из которых соответствует операция в (3'), а частичная упорядоченность множества, как известно [46], всегда может быть продолжена до линейного порядка на этом же множестве. Таким образом, вопрос сводится к нахождению частных случаев, когда такое продолжение возможно на практике и целесообразно.

4. Последовательное выполнение операций вдоль цепей из смежных остовов, которые могут принадлежать краям различных граней (например, e_1, e_2, e_5, e_8 или e_7, e_5, e_8 на рис. 1). Современные СКА имеют некоторые возможности для преобразований над множеством выражений. Это, например, одноименные процедуры *map* систем MAPLE V, MATHEMATICA, AXIOM, MuPAD и др., которые выполняют заданную операцию над заданным списком выражений. Однако такой список формируется ассоциативно.

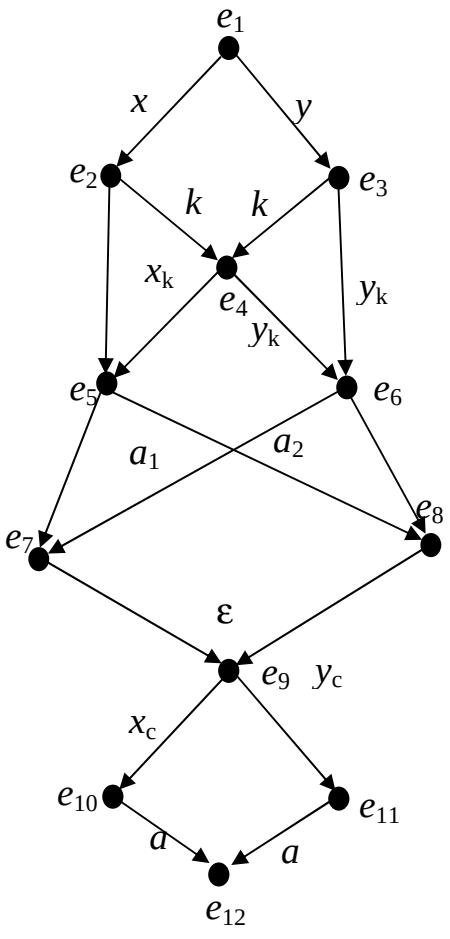

Рис. 2. "Нерв" множества выражений, описывающих кинематическую схему конструкции с учетом отношения зависимости

Цепь выражений (или операндов различных выражений) в «нерве» детерминирована структурой данных о задаче. Однако в полученном виде цепи «нерва» отражают возможные «направления» преобразований, в том числе и ассоциативные, которые можно отнести к различным «идеям решения». Таким образом, полученные в этом пункте представления также не полны, что свойственно, как отмечено выше, каждой в отдельности составляющей двойственного подхода.

III.1.3.3. Отношение зависимости, на наш взгляд, играет главную роль в разработке представлений о задаче в процессе решения как об объекте. Это отношение феноменологической симметрии [14] на Γ'', поскольку данные вида (3') на любом шаге решения подчинены одному и тому же отношению зависимости. Если дополнить построенный «нерв» отношением зависимости на Γ'' (п. III.1.3.1) так, как это сделано в [39], из всех возможных связей в «нерве» останутся только те, которые удовлетворяют этому отношению.

Отношение зависимости можно рассматривать и как частичную упорядоченность на Γ'', поскольку при формировании выражений (3') обычно известно какие элементы именуют начальные данные, какие промежуточные, а какие неизвестные. При удалении связей между независимыми структурными частями «нерв» становится частично упорядоченным комплексом ориентированных остовов. Для

выражений (17) «нерв» приобретает вид, показанный на рис. 2, где зависимость e_{i_1} от e_{i_2} показана как $e_{i_1} \to e_{i_2}$.

III. 1.3.4. Некоторые базисные процедуры для работы со структурой данных “нерв”

Организация преобразований на выделенной в (3’) структуре данных «нерв» требует разработки соответствующих процедур и, в первую очередь, процедур именования.

Из соотношения (16), а также п. III.1.3.2а-г следует необходимость базисной процедуры, которую, используя язык АНАЛИТИК, естественно назвать Ассоциативный Поиск (АСП), поскольку она должна возвращать все цепи, связывающие заданные операнды. Однако, в силу равноправия остовов «нерва» ее реализация должна существенно отличаться от имеющейся.

Из п. III.1.3.3 следует необходимость процедуры, назовем ее Поиск, близкой к существу процедуры АСП языка АНАЛИТИК, которая должна возвращать цепи остовов «нерва», связывающие заданные операнды, находящиеся в отношении зависимости, или пустое множество, если они независимы.

Из соотношения (16) также следует необходимость некоторых процедур, отсутствующих в языках АНАЛИТИК. Это процедура ГРАНЬ, определяющая относительно заданного остова координаты вершин, образующих заданную грань. Также необходима процедура ЦЕПЬ, возвращающая координаты вершин «нерва», находящихся в транзитивном отношении зависимости. Полнота набора процедур, предложенного для организации преобразований на “нерве”, требует исследования.

III. 2. Соответствие между структурой данных «нерв» и объектом языка АНАЛИТИК

III. 2.1. Соответствие между “нервом” и объектом языка АНАЛИТИК

Дедуктивный метод предполагает установление соответствия с известными случаями, которые рассматриваются как частные. Такое соответствие между полученной структурой и объектом последних версий языков АНАЛИТИК можно получить из факта, что отношение зависимости проявляет себя на множестве данных как отношение частичной упорядоченности.

Если выражения из множества Φ разрешены относительно неизвестных или промежуточных данных и упорядочены очередностью выполнения подстановок (а именно такому множеству соответствует объект языка АНАЛИТИК [24]), то это фактически означает существование на (3’) транзитивного отношения зависимости [27], продолженного до линейного порядка на (3’). Данное утверждение и устанавливает необходимое соответствие:

– множество выражений (3’) приводится путем преобразований к виду, соответствующему п. III.1.2. Данными являются правые части этих выражений;

– поле вершин – идентификаторы промежуточных и неизвестных данных, входящие и в другие выражения;

– объект языка АНАЛИТИК – “нерв” такой совокупности выражений.

При установленном соответствии две базисные процедуры именования АСП и ПОИСК естественным образом вырождаются в одну процедуру языка АНАЛИТИК – АСП, поскольку каждая ориентированная цепь (или траектория в терминологии [24]) является последовательностью остовов, связанных отношением зависимости (рис. 2).

III. 2.2. Процедура ЦЕПЬ

Из установленного соответствия следуют возможности перенесения на объект языка АНАЛИТИК способов организации преобразований, описанных в п.п. III.1.3.2а-г. Однако из этого же соответствия следует и неполнота базиса для реализации этих возможностей. В данной статье разработана процедура ЦЕПЬ, дополняющая язык АНАЛИТИК. Язык реализации этой процедуры – АНАЛИТИК-2000.

При условии отделимости объект языка АНАЛИТИК изоморфен ориентированному дереву [4, 39] и его можно представить объединением ориентированных цепей, каждая из которых образована вершинами, находящимися в транзитивном отношении зависимости. Координаты каждой вершины на такой цепи определяются выражением [29]

имя [*кортеж* 1] [*кортеж* 2]…[*кортеж n*], (18)

где *имя* – имя корневой вершины; *кортеж* – последовательность чисел.

Таким образом, последовательность

имя,
имя [*кортеж* 1],
имя [*кортеж* 1] [*кортеж* 2],
… (19)
имя [*кортеж* 1] [*кортеж* 2]…[*кортеж N-1*],
имя [*кортеж* 1] [*кортеж* 2]…[*кортеж N-1*] [*кортеж N*]

является отражением зависимости между вершинами цепи.

До данных исследований выражение (18) являлось словом языка АНАЛИТИК. Таким образом, назначение процедуры ЦЕПЬ – представить выражение (18) в виде последовательности (19).

Опыт работы с языком АНАЛИТИК показывает, что текст программы очень часто оказывается лаконичнее описания алгоритма, а мнемоника языка делает этот текст вполне понятным.

Текст процедуры ЦЕПЬ:

```
ЦЕПЬ(Вх): (
ЕСЛИ(Вх?ПЕРЕМ=1, (Вых:=Вх; НА (ВЫХОД)));  ЕСЛИ(Вх?КООРД=0, (ВОЗВ(АВОСТ); КОНЕЦ));
ВхТекст := ПРЕВ(Вх);  ПОЗИЦИЯ:= StringPosition( ВхТ, ‘ ] ’); Колич:= КО(ПОЗИЦИЯ)
ПОЗИЦИЯ1:= StringPosition( ВхТ, ‘ [ ’);
Вых:= agr( i, 1, Колич,  ); Вых[1]:= ВхТекст[1:(ПОЗИЦИЯ1-1)];
ЦИКЛ( i, 2, Колич+1, Вых[i]:= ВхТекст[1: ПОЗИЦИЯ[i-1]]); ЦИКЛ( i, 1, Колич+1, Вых[i]:= ПРЕВ(Вых[i]));
ВЫХОД. ВОЗВ(Вых));
```

Для процедуры ЦЕПЬ были разработаны процедура StringPosition и вызываемая ею процедура Insert. Эти процедуры, соответственно, возвращают список чисел – позиций заданной литеры в заданном тексте и вставляют в список дополнительную компоненту в указанное место. Функция процедуры StringPosition может быть выполнена непосредственно базисной процедурой АСП. Однако опыт показывает, что форма представления результата последней затрудняет его дальнейшее использование. Процедуры, аналогичные StringPositions и Insert, есть в языке MATHEMATICA и др. Поэтому при разработке использованы оригинальные названия, а тексты не представляют интереса для публикации.

III. 3. Апробация

Представление о данных как о целостном объекте являются новыми для компьютерной алгебры, а их реализация в СКА АНАЛИТИК-93, -2000 – уникальной. В настоящее время совершенно недостаточно исследований, посвященных методологии разработки ЧАМ на основе этих представлений. Более того, за исключением отдельных примеров, моделирующих возможность применений [4, 25], совершенно недостаточно исследований их практической значимости, в частности, для решения сложных задач, что послужило бы обоснованием необ-

ходимости дальнейших теоретических разработок и реализаций.

III. 3.1. Задача

Для апробации выбрана распространенная в прикладном математическом моделировании задача о нахождении полной производной неявной функции, заданной множеством громоздких выражений. Практическая необходимость подобных исследований очевидна. С теоретической точки зрения дифференцирование неявной функции – типичный пример операции над сцепленным множеством выражений.

В данной работе рассмотрен частный случай, соответствующий объекту языка АНАЛИТИК. Считается, что путем преобразований неявная зависимость (3') переменных $y_0 \to y_k$ представлена сложной функцией, заданной множеством выражений с иерархической структурой. Для простоты рассматривается функция одной переменной $y_0 = f(y_k)$, что несущественно, поскольку результат легко обобщается:

$$\begin{gathered} y_0 = f_0(y_1, y_2, y_3, ..., y_k); \\ y_1 = f_1(y_2, y_3, ..., y_k); \\ \ldots \\ y_{k-1} = f_{k-1}(y_k). \end{gathered} \tag{20}$$

Задача состоит в нахождении формульного выражения $\frac{dy_0}{dy_k}$.

Примером такой функции являются выражения (17), преобразованные к виду

$$\begin{gathered} R : \sqrt{(b_1 - x)^2 + y^2}; \\ x : \frac{b_6 b_8}{k} + x_k\left(1 - \frac{b_8}{k}\right);\ y : y_k\left(1 - \frac{b_8}{k}\right); \\ k : \sqrt{(b_6 - x_k)^2 + y_k^2}; \\ x_k : b_4 \sin(a_1 + a_2); \qquad y_k : b_1 - b_4 \cos(a_1 + a_2); \\ a_1 : \arccos\left(\frac{b_3^2 + \varepsilon^2 - b_7^2}{2 b_3 \varepsilon}\right); \quad a_2 : \arccos\left(\frac{b_1 - y_c}{\varepsilon}\right); \\ \varepsilon : \sqrt{x_c^2 + (b_1 - y_c)^2}; \\ x_c : b_2 - r\cos(a);\ y_c : b_8 - r\sin(a); \\ a : \ . \end{gathered} \tag{21}$$

"Нерв" этих выражений отличается от изображенного на рис. 2 только именами вершин. Прикладная модель [43] существенно использует формульное выражение для $\frac{dR}{da}$.

Проблематичность визуального анализа характера зависимости $R(a)$ в явном виде иллюстрирует рис. 3. Кривая 1 показывает рост длины выражения (в символах) для $R(a)$ при последовательном выполнении подстановок m. После выполнения всех подстановок зависимость описывается выражением длиною в 2155 символов.

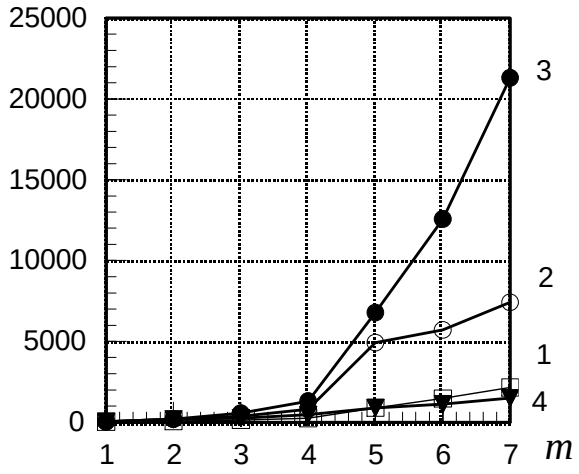

Рис. 3. Увеличение длины выражений в процессе преобразований на сцепленном множестве выражений

III. 3.2. Нахождение производной

Возможности вычисления значений функций вида (20) без рекурсивного выполнения подстановок рассмотрены в [25]. Производная может быть найдена следующим образом.

Условие отделимости (п. III.1.1) означает, что каждой кратной вершине «нерва» совокупности (20) соответствует равное кратности количество различных вершин. “Нерв” распадается на множество цепей вида

$$
\begin{array}{l}
y_0 \to y_1 \to y_2 \to \ldots\ldots\ldots\ldots \to y_{k-1} \to y_k \\
y_0 \to y_1 \to y_2 \to \ldots \to y_{k-2} \to y_k \\
\ldots \\
y_0 \to y_1 \to y_2 \to y_k \\
\ldots \\
y_0 \to y_1 \to y_{k-1} \to y_k \\
y_0 \to y_1 \to y_k
\end{array}
\qquad (22)
$$

$$
\begin{array}{l}
y_0 \to y_2 \to y_3 \to \ldots\ldots\ldots\ldots \to y_{k-1} \to y_k \\
y_0 \to y_2 \to y_3 \to \ldots \to y_{k-2} \to y_k \\
\ldots \\
y_0 \to y_2 \to y_{k-1} \to y_k \\
y_0 \to y_2 \to y_k \\
\ldots \\
y_0 \to y_{k-1} \to y_k \\
y_0 \to y_k,
\end{array}
$$

каждой из которых соответствует частная зависимость сложной функции (20) от «независимых» переменных y_k. Таким образом, полная производная каждой из таких частных зависимостей будет равна произведению производных от каждой переменной в цепи по последующей. Из аддитивности дифференциала также следует, что искомая производная будет равна сумме производных, найденных по всем цепям.

III. 3.3. Процедура для нахождения производной

Легко видеть, что каждая цепь в (22) изоморфна структуре слова вида (19). Алгоритм нахождения производной вполне понятен из текста процедуры:

```
 /* Полное дифференцирование y0 по yk */
Wdif( yk, y0): (
as   := АСП(y0, yk);                          /*Формирование всех цепей вида (22)*/
kc  := КО (das);                              /* Количество цепей, соединяющих y0 и yk  */
dz:=0;                                        /*Начальное значение полной производной*/
ЦИКЛ( m,1, kc, (dz1:=1;                       /*Начальное значение полной производной в цепи*/
                  das:=ЦЕПЬ(as[m]);           /*Формирование координат вершин (19) в цепи из (22)*/
                  kv := КО (das[j])           /* Количество вершин в j – й цепи */
          ЦИКЛ(i,1, kv-1, (
                f1:=ФОРМ(2,dasp[j,i]);   f2:=ФОРМ(2,dasp[j,i+1]);
               ddz:=dif(f2, f1| 6);          dz1:=ФОРМ(1,dz1*ddz | 2)));
    dz:= ФОРМ(1, dz+dz1| 2 )) );
ВОЗВ(dz)
 );
```

Тут ФОРМ – процедура, являющаяся элементом аппарата управления языков АНАЛИТИК. Она формирует выражения с подстановкой на заданное количество шагов (первый операнд), что и обеспечивает возможность вычислений без рекурсивного выполнения подстановок вдоль цепи.

Процедура Wdif апробирована при вычислении производной сложной функции (21). Увеличение длины выражения в процессе нахождения производной dz иллюстрирует кривая 2 на рис. 3.

III. 3.4. Анализ результатов

Производная может быть найдена обычным путем с помощью стандартной процедуры, которая осуществляет аналитическое дифференцирование с вынужденными рекурсивными подстановками. Увеличение длины выражения в таком процессе иллюстрирует кривая 3. Легко увидеть, что длина выражения полной производной dR/da, полученного с помощью структуры «нерв» (кривая 2), составляет лишь 34% длины выражения, полученного обычным путем (кривая 3). Соотношение начального (для $R(a)$) и конечного (для dR/da) объема данных выглядит соответственно как 1:200 и 1: 576. Если отменить условие отделимости и рассматривать объект со «склеенными» вершинами (п. III.1.2.2), что в данном случае тождественно использованию в «ручных» преобразованиях дистрибутивности умножения, то такая оптимизация алгоритма позволила уменьшить длину выражения для полной производной $\frac{dR}{da}$ еще почти в 5 раз (кривая 4). Соотношение длины выражения для производных, полученных с помощью структуры данных «нерв» и обычным путем, при этом выглядят уже как 7%, а к начальному объему данных – как 1:40.

Выводы

На основании исследований, проведенных в данной статье, можно сделать следующие выводы:

1. Современные входные языки СКА содержат совокупности средств, позволяющие надстраивать языки более высокого уровня для представления данных с использованием двух качественно отличающихся подходов – на конструктивной и аналитической основе.
2. Дальнейшее расширение области применения ЧАМ на задачи с большим объемом и сложностью данных

требует соответствующего развития аппарата представления данных о задаче и, в первую очередь, на аналитической основе.

3. Полученные в работе результаты являются обобщением и дальнейшим развитием представлений об аппарате АТД и, в частности, аналитического подхода, являющегося парадигмой входных языков нового поколения семейства АНАЛИТИК.
4. Использование структуры данных «нерв» совместно с аппаратом управления преобразованиями позволило существенно уменьшить длину формульного представления результата по сравнению с существующими методами.
5. Алгоритм, разработанный на структуре «нерв», является «самооптимизирующимся». На практике списки аргументов выражений в (20) очень часто бывают неполными, что соответствует уменьшению кратности вершин «нерва» и, таким образом, уменьшению числа цепей вида (22). Процедура Wdif не является механической реализацией известных формул дифференцирования сложных функций. Она использует реальное количество цепей в (22), поставляемое АСП, что и обосновывает сделанное утверждение. В приведенном примере (21) зависимость описывается одиннадцатью аргументами. Полнота списков вида (22) в этом случае означает существование 1024 частных зависимостей (по числу цепей). При использовании общей формулы дифференцирования сложной функции пришлось бы проводить вычисления вдоль каждой из них. При использовании процедуры Wdif дифференцирование множества (21) автоматически осуществляется по 56 цепям. Многие из этих цепей имеют общие участки (рис. 2). Использованная выше (рис. 3, кривая 4) «вторичная» оптимизация состоит в устранении повторов. Таким образом, есть основание утверждать и то, что программирование символьных преобразований с использованием структуры данных «нерв» и аппарата управления преобразований позволяет разрабатывать процедуры, эффективность которых будет расти вместе с увеличением сложности объектов. Это соответствует результатам, полученным в работе [3].
6. Представляет интерес обобщение этого алгоритма на случай неявной функциональной зависимости, описываемой множеством выражений вида (3'), не являющихся иерархической структурой относительно подстановок.
7. Функциональная зависимость является частным случаем абстрактного отношения зависимости. В проведенных исследованиях, кроме п.4, семантика операций в (3') не играла никакой роли. Таким образом, полученные выше структуры данных обладают общностью. Представляет интерес исследование возможности распространения полученных результатов на различные классы задач, свойства которых удовлетворяют принятым исходным положениям.

СПИСОК ЛИТЕРАТУРЫ

1. Клименко В.П. Основные принципы построения систем интерпретации языков, проблемно ориентированных на научные и инженерные задачи // Кибернетика. – 1990. – № 1. – С. 49 – 56.
2. Абрамов С.А., Зима Е.В., Ростовцев В.А. Компьютерная алгебра // Программирование. – 1992. – № 5.– С. 4 – 25.
3. Ляхов О.Л. Деякі сучасні проблеми застосування чисельно-аналітичних методів // Математичні машини і системи. – 2003. – № 2. – С.54 – 63.
4. Клименко В.П., Ляхов А.Л., Фишман Ю.С. Основные тенденции развития языков систем компьютерной алгебры // Математические машины и системы.– 2002. – № 2. – С. 29 – 64.
5. Ефимов Г. Из истории развития и применения компьютерной алгебры в Институте прикладной математики имени М.В. Келдыша // Математичні машини і системи. – 2003. – № 2. – С. 96 –105.
6. Человек и вычислительная техника / Глушков В.М., Брановицкий В.И., Довгялло А.М., Рабинович З.Л., Стогний А.А. / Под ред. В.М. Глушкова. – К.: Наукова думка, 1971. – 294 с.
7. Колмогоров А.Н. Сложность задания и сложность построения математических объектов // УМН. – 1972. – Т. XXVII. – Вып. 2 (164). – С. 159.
8. Бенерджи Р. Теория решения задач. – М.: Мир, 1972. – 224 с.
9. Блехман И.И., Мышкис А.Д., Пановко Я.Г. Механика и прикладная математика: Логика и особенности приложений математики. – М.: Наука, 1983. – 328 с.
10. Тихонов А.Н., Костомаров Д.П. Вводные лекции по прикладной математике. – М.: Наука, 1984. – 192 с.
11. Пойа Дж. Математическое открытие. – М.: Наука, 1976. – 448 с.

12. Дьяконов В.П. Математическая система MAPLE v R3/ R4/ R5. – М.: Солон, 1998. – 400 с.
13. Jenks R.D., Sutor R.S. AXIOM. The Scientific Computation System. – Oxford: Springer-Verlag, 1992. – 742 p.
14. Кулаков Ю.И. О новом виде симметрии, лежащей в основании физических теорий феноменологического типа // ДАН СССР. – 1971. – Т. 201, № 3. – С. 570 – 572.
15. Глушков В.М. Теория алгоритмов. – К.: КВИРТУ, 1961. – 168 с.
16. Морозов А.А., Клименко В.П., Фишман Ю.С. Принципы построения системы программирования АНАЛИТИК-91 (ориентированной на более совершенную технологию программирования научных и прикладных задач)// Упр. системы и машины. – 1992. – № 3. – С. 60 – 69.
17. http://www.nag.com/symbolic/AX.html .
18. http://www.mapleapps.com/ .
19. http://www.wolfram.com/ .
20. http://www.sciface.com/.
21. Criesmer J.H., Jenks R.D. SCRATCHPAD/1, An interactive Fasility for Symbolic Mathematics // Proc. of the 2nd Symp. on Symbolic and Algebraic Manipulation, ACM Headquarters. – N.Y. – 1971. – P. 42 –58.
22. Хомский Н. Три модели описания языка // Кибернетический сборник. – 1961. – Вып. 2. – С. 237 –266.
23. Ляхов А.Л. Интеллектуализация численно-аналитического решения научных и прикладных задач // Современные проблемы информатизации в технике и технологиях (по итогам VIII Междунар. научной конференции): Сб. трудов. – Воронеж: Центрально-Черноземное книжное издательство, 2003. – Вып. 8 – С. 117 – 118.
24. Фишман Ю.С. Основные особенности создания и применения средств реализации численно-аналитических методов на ЭВМ // Математические машины и системы. – 1998. – № 2. – С. 9 –17.
25. Программирование решения инженерных задач в среде системы АНАЛИТИК / Клименко В.П., Фишман Ю.С., Горик А.В., Ляхов А.Л., Пискунов В.Г. // Перша міжнар. науково-практична конф. з програмування (УкрПРОГ'98). – Україна, Київ: Кібернетичний центр НАН України. Праці. – 1998. – 2 – 4 вересня. – С. 553 – 562.
26. Bloomfield L. A set of postulates for science of language // Languages. – 1926. – N 2. – P. 26 –31.
27. Кон П. Универсальная алгебра. – М.: Мир, 1968. – 352 с.
28. Александров П.Я. Комбинаторная топология. – М.: ОГИЗ – Гостехиздат, 1947. – 660 с.
29. АНАЛИТИК-93/ Морозов А.А., Клименко В.П., Фишман Ю.С., Бублик Б.А., Горовой В.Д., Калина Е.А. // Кибернетика и системный анализ. – 1995. – С. 127 – 157.
30. АНАЛИТИК-2000/ Морозов А.А., Клименко В.П., Фишман Ю.С., Ляхов А.Л., Кондрашов С.В., Швалюк Т.Н. // Математические машины и системы. – 2001. – № 1–2.– С. 66 – 99.
31. Понтрягин Л.С. Основы комбинаторной топологии. – М.: ОГИЗ – Гостехиздат, 1947. – 144 с.
32. Ляхов О.Л. Методи комп'ютерної алгебри в задачах згину брусів кусково-однорідної структури // Машинознавство. – 1999. – № 7. – С. 8 – 15.
33. Вычислительные машины и мышление / Под ред. Э. Фейгенбаума и Дж. Фельдмана. – М.: Мир, 1967. – 552 с.
34. Капитонова Ю.В., Скурихин В.И. О некоторых тенденциях развития и проблемах искусственного интеллекта // Кибернетика и системный анализ. – 1999. – № 1. – С. 43 – 50.
35. Сергиенко И.В. Об основных направлениях развития информатики. 5. Искусственный интеллект // Кибернетика и системный анализ. – 1997. – № 6. – С. 29 – 34.
36. Попов Б.О., Монцібович Б.Р. Розв`язування задач на машинах для інженерних розрахунках. – К.: Наукова думка, 1978. – 346 с.
37. Математический пакет MAPLE V Release 4: Руководство пользователя / Г.В. Прохоров, В.В. Колбеев, К.И. Желнов, М.А. Леденев. – Калуга: Облиздат, 1998. – 200 с.
38. Капустина Т.В. Компьютерная система MATHEMATICA 3.0 – М.:СОЛОН – Р, 1999. – 240 с.
39. Ляхов О.Л. Складні інженерні задачі як об'єкт комп'ютерної алгебри // Зб. наук. пр. Полтавський національний технічний університет імені Юрія Кондратюка. – 2003. – Вип. 11. – С.87 – 97.
40. Свами М., Тхуласираман К. Графы, сети и алгоритмы. – М.: Мир, 1984. – 455 с.
41. Уемов А.И. Вещи, свойства и отношения. – М.: Изд-во АН СССР, 1963. – 184 с.
42. Богданович В.І. Про поняття множина з системної точки зору // Філософські проблеми сучасного природознавства. – 1971. – Вип. 22. – С. 37 – 49.
43. Определение теоретической подачи вертикального дифференциального растворонасоса с качающейся насосной колонкой / А.В. Головкин, А.Г. Онищенко, Д.Г. Тищенко, В.Б. Надобко // Проблеми теорії і практики залізобетону: Зб. Наук. Пр. – Полтава: ПДТУ імені Юрія Кондратюка, 1997. – С. 98 – 101.
44. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы. Построение и анализ. – М.: МЦНМО, 2001. – 960с.
45. Бурбаки Н. Начала математики. Теория множеств. – М.: Мир, 1965. – 456 с.
46. Shpilrein A. Fundamental Mathematic. – 1930. – Vol. 16. – P. 386 – 389.