Применение методов конструктивной геометрии для визуального синтеза и анализа компьютерных программ

Волошинов Д.В., к.т.н., СПбГПУ, Санкт-Петербург

Аннотация

В статье излагаются результаты исследований в области автоматизации процессов синтеза компьютерных программ для решения геометрических задач методами конструктивного моделирования.

Ключевые слова. Конструктивная геометрическая модель (КГМ), геометрическая машина, геометрический алгоритм, объектно-ориентированное проектирование (ООП).

1. ВВЕДЕНИЕ

В настоящее время при решении научно-технических задач и реализации программных систем используются преимущественно аналитические методы, что объясняется традиционным сопоставлением вычислительного устройства дискретного принципа действия с понятием числа. Между тем геометрический (синтетический, конструктивный) метод обладает рядом неоспоримых преимуществ, таких как наглядность, интегрированное содержание геометрического образа, что проявляется при отображении результатов вычислений и визуальном управлении вычислительными процессами. Однако вычислительные возможности синтетических методов ассоциируются с традиционными инструментальными ограничениями, влияющими на исполнение предметных моделей; считается что аналитические методы первичны по отношению к геометрическим. Поэтому практически всегда разработки, осуществляемые с помощью конструктивных методов, сопровождаются аналитической интерпретацией, обеспечивающей упрощение реализации полученных результатов на ЭВМ. По мнению автора, такая интерпретация далеко не всегда необходима и оправдана.

Универсальный характер геометрических методов вполне позволяет рассматривать вычислительную машину как устройство геометрического типа. Теоретическое обоснование геометрического моделирования как информационного процесса, базирующегося на принципах инвариантной неопределенности, развивается в работах петербургской геометрической школы. Согласно взглядам представителей этой школы любая геометрическая конструкция, дополненная алгоритмом в виде совокупности геометрических операций, рассматривается как действующая геометрическая машина [2], выполняющая некоторое геометрическое преобразование. Совокупность геометрических машин, образующих описание исходного объекта или процесса, составляет их конструктивную геометрическую модель (КГМ). Если для реализации модели в рамках заданного аксиоматического базиса определены операции геометрического содержания, можно говорить о вычислительной машине как об устройстве геометрического типа. При таком подходе геометрические объекты получают интегрированные значения, а операции с привлечением понятия числа становятся необязательными или несущественными.

До настоящего времени исследования в области синтетической геометрии сдерживались отсутствием концепции организации и развития средств автоматизации как для разработки и реализации алгоритмов, так и для прикладного программирования в области конструктивного геометрического моделирования. Автор данной статьи проводит исследования, направленные на выработку положений такой концепции и создание на ее основе специализированной системы синтеза и анализа КГМ Симплекс [5].

В основе разрабатываемой концепции лежит понимание объектов многомерного пространства, (точка, прямая, плоскость и т.п.) как сущностей, проявляющих свои геометрические свойства в многосвязных отношениях и заданных на регулируемом аксиоматическом базисе [3], [4].Такой подход позволяет представлять КГМ в виде реляционных схем и таким образом осуществлять их автоматизированный синтез, В среде системы Симплекс модели строятся на основе понятия отношения – элементарного звена фактологической структуры геометрической модели и выражаются в предикатной форме. Двойственная природа предикатов отношений (логическое воплощение структурных характеристик модели и их расчетное содержание) позволяет отделить процессы логического анализа и синтеза алгоритма от непосредственного расчета значений объектов. Конструктивный геометрический алгоритм становится виртуальным, не зависящим от конкретизации значений и, что особенно важно, типов объектов, а, следовательно, процесс моделирования может быть представлен абстрактной логической программой.

В основе разрабатываемой концепции и базирующейся на ней архитектуре системы Симплекс лежат следующие свойства КГМ:

- объекты КГМ образованы и связаны друг с другом посредством геометрических операций (построений);
- объекты КГМ экзистенциональны и индивидуальны;
- недопустимы деструктивные рекурсивные связи между объектами в КГМ.

Исполнительные механизмы системы Симплекс обеспечивают:

- технологию визуального проектирования КГМ для синтеза, отладки и исполнения прикладных программ;
- декларативный способ описания моделей;
- объектно-ориентированную методику проектирования объектов производных типов и преобразований;
- логический анализ и синтез геометрических моделей;
- открытость системы для оснащения новыми объектами и функциями;
- интерфейс с другими системами проектирования на основе автоматической ретрансляции КГМ из языка внутреннего представления в общетехнические алгоритмические языки (PASCAL [7], PROLOG [1], [8]) и посредством стандартных форматов графических данных.

2. ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ МЕТОДИКИ ПРОЕКТИРОВАНИЯ КГМ

Пусть имеется два множества объектов $\{\mathbf{x}\}$ и $\{\mathbf{y}\}$. Каждый объект $\mathbf{y}$ однозначно сопоставлен с объектом $\mathbf{x}$, и процедура, осуществляющая различение объектов $\mathbf{y}$, достаточно проста.

Будем называть объекты **y** *значениями* **x**. Из существования **x** следует существование его значения. Если сопоставление {**x**} с {**y**} не установлено, будем считать значения {**x**} *нерассчитанными* (множество {**y**} фиктивно). Обычно в математике роль объектов **y** играют числа или организованные совокупности чисел. В практике конструктивного геометрического моделирования для выражения геометрических значений объектов используются *реперы*.

Пусть между объектами **a** и **b**, **c**, … установлено соответствие **R**: **a**←**f**(**b**, **c**, …), которое в дальнейшем будем называть *отношением* между объектами **a** и **b**, **c**, …. Отношение **R** выражает факт зависимости значений объекта **a** от значений объектов **b**, **c**, …. Способ нахождения значений **a** по значениям **b**, **c**, … задан функцией **f**. Абстрагируясь от конкретных имен объектов **a**, **b**, **c**, … и вида преобразования **f**, отношение **R** можно записать в обобщенном виде (1):

$$\mathbf{Out}\leftarrow\mathbf{F}(\mathbf{In}). \tag{1}$$

Здесь **In** – входные параметры отношения **R**, **Out** – выходные параметры **R**, а **F** – функция, задающая вид отношения. Представим отношение **R** в предикатной форме:

$$\sim\!\boldsymbol{r}\ (\mathbf{In}, \mathbf{Out}, \mathbf{Fm}, \mathbf{F}, \mathbf{A}). \tag{2}$$

где $\sim\!\boldsymbol{r}$ – предикатный символ, используемый для выражения отношения, **Fm** – способ согласования входных параметров, **A** – признак группового соотнесения отношения.

Объекты КГМ, выраженной отношениями (2), будем обозначать символьными термами – носителями геометрических значений. Если **X** имеет значение (определенное или неопределенное), то достижима цель

$$\boldsymbol{value}\ (\mathbf{X}). \tag{3}$$

Пусть задано множество отношений $\{\mathbf{R}\}=\{\mathbf{R_a}, \mathbf{R_b}, \ldots, \mathbf{R_c}\}$, такое что:

1. Все выходные параметры множества {**R**} различны;
2. Все входные параметры множества {**R**} одинаковы с точностью до наименования, порядка и количества;

$$\begin{gathered}\mathbf{R_a}\colon \mathbf{a} \leftarrow \mathbf{f_a}\ (\mathbf{l}, \mathbf{m} \ldots, \mathbf{n}),\\ \mathbf{R_b}\colon \mathbf{b} \leftarrow \mathbf{f_b}\ (\mathbf{l}, \mathbf{m} \ldots, \mathbf{n}),\\ \ldots\\ \mathbf{R_c}\colon \mathbf{c} \leftarrow \mathbf{f_c}\ (\mathbf{l}, \mathbf{m} \ldots, \mathbf{n}),\\ \{\mathbf{a}, \mathbf{b}, \ldots \mathbf{c}\}\cap\{\mathbf{l}, \mathbf{m}, \ldots \mathbf{n}\}=\varnothing.\end{gathered} \tag{4}$$

Перепишем (4) в виде

$$\mathbf{R}\colon \mathbf{a}, \mathbf{b}, \ldots, \mathbf{c}\leftarrow \mathbf{f}(\mathbf{l}, \mathbf{m} \ldots, \mathbf{n}), \tag{5}$$

устанавливая связь объектов **a**, **b**, …, **c** с объектами **l**, **m**, …, **n**. Расширяя определение (2), будем называть *отношением* выражение (5). Любой **x: x** ← $\mathbf{f_x}$ (**l**, **m** …, **n**) находится в зависимости от всех входных параметров. Терминальный характер функции $\mathbf{f_x}$ позволяет рассматривать отношение (5) как преобразователь с закрытой внутренней структурой и называть его простым отношением. При отсутствии ограничений на количество входных и выходных параметров, **In** и **Out** представим в виде линейных списочных структур. Тогда в предикате (6) **In** – список входных параметров **R**; **Out** – список выходных параметров **R**; $\mathbf{F_m}$, **F** и **A** – те же, что и в (2).

$$\sim\!\boldsymbol{r}\ (\mathbf{In}, \mathbf{Out}, \mathbf{F_m}, \mathbf{F}, \mathbf{A}). \tag{6}$$

Отношение (7) с пустым списком **In** есть способ записи в виде предиката $\sim\!\boldsymbol{r}$ значений объектов-констант. Конкретизация значений в списке **Out** зависит от вида функции **Fconst**.

$$\sim\!\boldsymbol{r}\ ([\ ], \mathbf{Out}, \mathbf{Fm}, \mathbf{Fconst}, \mathbf{A}). \tag{7}$$

Возникновение нового факта, выраженного предикатами (6) или (7), будем называть *объявлением отношения*. В момент объявления отношения декларируются названные в нем объекты и констатируется зависимость значений выходных параметров от входных (*логическая работа* предиката отношения). Объявление отношения в среде проектирования КГМ служит причиной активизации вычислительной подсистемы для расчета значений объектов модели. Расчетные действия составляют *вычислительную работу* предиката отношения (т.н. побочный нелогический эффект [8]). Единый способ выражения переменных и констант, а также возможность отделения вычислительной работы отношения от логической предопределили выбор предиката отношения в качестве основного средства описания конструктивных КГМ.

Вычислительная работа отношения инициируется системным предикатом ***execute*** (11). Объекты **X**: $\sim\!\boldsymbol{r}$(**In**, **Out**, **Fm**, **F**, **A**), ***member***(**X**, **Out**), получают значения при успешном исполнении цели ***execute***($\sim\!\boldsymbol{r}$(**In**, **Out**, **Fm**, **F**, **A**)). В случае успеха цели генерируется предикат-свойство

$$\boldsymbol{counted}(\mathbf{X}). \tag{8}$$

Свойства (8) различают среди **X** *рассчитанные* (успех ***counted***(**X**)) и *нерассчитанные* объекты.

Поскольку все объекты выходных параметров (5) получают значения одновременно, для отношения в целом вводится обобщенный признак совершения расчета:

$$\boldsymbol{rcounted}(\sim\!\mathbf{r}(\mathbf{In}, \mathbf{Out}, \mathbf{Fm}, \mathbf{F}, \mathbf{A}))\ \boldsymbol{if}\ \boldsymbol{allcounted}(\mathbf{Out}). \tag{9}$$

$$\begin{gathered}\boldsymbol{allcounted}([\ ]).\\ \boldsymbol{allcounted}([\mathbf{X}|\mathbf{Other}])\ \boldsymbol{if}\ \boldsymbol{counted}(\mathbf{X}), \boldsymbol{allcounted}(\mathbf{Other}).\end{gathered} \tag{10}$$

Определение предиката ***execute***, заданное импликацией (11), обеспечивает однократность расчета простого отношения:

$$\boldsymbol{execute}(\sim\!\boldsymbol{r}(\mathbf{In}, \mathbf{Out}, \mathbf{Fm}, \mathbf{F}, \mathbf{A}))\ \boldsymbol{if\ not}\ \boldsymbol{rcounted}(\sim\!\boldsymbol{r}(\mathbf{In}, \mathbf{Out}, \mathbf{Fm}, \mathbf{F}, \mathbf{A})), \boldsymbol{allcounted}(\mathbf{In}). \tag{11}$$

Рассчитанные значения **X**, принадлежащие области допустимых значений **F**, являются определенными:

$$\boldsymbol{definite}(\mathbf{X}). \tag{12}$$

Объекты **X** списка **In** – непосредственные предки объектов **Y** списка **Out**, если истинно утверждение ***parent*** (**X**, **Y**, **A**) (13).

$$\boldsymbol{parent}\ (\mathbf{X}, \mathbf{Y}, \mathbf{A})\ \boldsymbol{if}\ \sim\!\boldsymbol{r}(\mathbf{In}, \mathbf{Out}, \_, \_, \mathbf{A}), \boldsymbol{member}\ (\mathbf{X}, \mathbf{In}), \boldsymbol{member}\ (\mathbf{Y}, \mathbf{Out}). \tag{13}$$

В отношении обнаруживается циклическая зависимость, если истинно ***cycle***(**X**, **A**) (14).

$$\boldsymbol{cycle}(\mathbf{X}, \mathbf{A})\ \boldsymbol{if\ parent}\ (\mathbf{X}, \mathbf{X}, \mathbf{A}). \tag{14}$$

Значения **X** неопределенны, если конкретизация правила (15) – истина.

$$\boldsymbol{indefinite}(\mathbf{X})\ \boldsymbol{if\ parent}(\mathbf{X},\mathbf{Y},\mathbf{A}), \boldsymbol{not\ definite}(\mathbf{X}), \boldsymbol{counted}(\mathbf{Y}). \tag{15}$$

Рассмотрим принципы организации совокупностей отношений. Пусть объявлены два отношения, записанные в виде:

$$\begin{gathered}\mathbf{R1}(\mathbf{F1})\colon \mathbf{A}, \mathbf{B}, \ldots \leftarrow\mathbf{K}, \mathbf{L}\ldots\\ \mathbf{R2}(\mathbf{F2})\colon \mathbf{C}, \mathbf{D}, \ldots \leftarrow\mathbf{M}, \mathbf{N}\ldots\end{gathered} \tag{16}$$

Пара отношений **R1**, **R2** несовместна, если цель ***contrariety*** (**R1**, **R2**) истинна (17):

$$\boldsymbol{contrariety}\ (\sim\!\boldsymbol{r}(\_, \mathbf{Out1}, \_, \_, \mathbf{A}), \sim\!\boldsymbol{r}(\_, \mathbf{Out2}, \_, \_, \mathbf{A}))\ \boldsymbol{if}\ \mathbf{Out1} \neq \mathbf{Out2}, \boldsymbol{member}(\mathbf{X}, \mathbf{Out1}), \boldsymbol{member}(\mathbf{X}, \mathbf{Out2}). \tag{17}$$

В противном случае пара отношений совместна.

Пусть {**R**} – совокупность отношений, представленная в виде списка. {**R**} в целом обладает совместным составом, если успешна цель (18):

$$\begin{gathered}\boldsymbol{relationset}(\mathbf{R})\ \ \boldsymbol{if}\ \mathbf{R}=[\sim\!\boldsymbol{r}(\_, \_, \_, \_, \_)], \mathbf{!}.\\ \boldsymbol{relationset}(\mathbf{R})\ \ \boldsymbol{if\ not}(\boldsymbol{member}(\mathbf{R1}, \mathbf{R}), \boldsymbol{member}(\mathbf{R2}, \mathbf{R}), \boldsymbol{contrariety}(\mathbf{R1}, \mathbf{R2}))).\end{gathered} \tag{18}$$

Выразим причинно-порождающую зависимость объектов **X** и **Y** правилом (19):

ancestor(**X**,**Y**,**A**) ***if parent***(**X**,**Y**,**A**).
ancestor(**X**,**Y**,**A**) ***if parent***(**X**,**Z**,**A**), ***ancestor***(**Z**,**Y**,**A**). (19)

Совокупность отношений {**R**} обладает корректным составом, если достижима цель (20):

bagof (**A**, ***~r***(_, _, _, _, **A**), **R**), ***relationset***(**R**), ***not ancestor***(**X**, **X**, **A**). (20)

Для объектов-констант (7), истинно выражение (21):

objconst(**X**,**A**) ***if*** ***~r***([ ], **Out**, _, _, **A**), ***member***(**X**, **Out**). (21)

Объекты **X**, для которых целевое утверждение (22) истинно, составляют группу вакантных объектов.

objvacant(**X**, **A**) ***if*** ***~r***(**In**, _, _, _, **A**), ***member***(**X**, **In**), ***not parent***(_, **X**, **A**), ***parent***(**X**, _, **A**). (22)

Если объект **X** вакантный, то истинны

objvacant(**X**, **A**), ***not counted***(**X**).
ancestor (**X**, **Y**, **A**), ***not counted***(**X**)., (23)

и, следовательно, все отношения, декларирующие объекты **Y**, являются нерассчитанными.

Статусы совершения расчета (8), определенности (12) и неопределенности (15) значений объектов всецело определяют характер визуального взаимодействия пользователя с КГМ в процессе ее конструирования и применения в среде системы Симплекс. Отсутствие значений у объектов **Y** создает известные неудобства при реализации моделей, поскольку объекты без значений не могут быть явно визуализированы. Поэтому, если для этого имеются условия, имеет смысл присваивать вакантным объектам фиктивные значения и считать такие объекты на определенный период времени фиктивными константами. Тогда любой объект **X** следует соотносить по признаку наличия значения с одной из следующих категорий:

- объект, обладающий реальным значением (определенным или неопределенным) ***realvalue***(**X**);
- объект, не обладающий значением ***nonvalue***(**X**);
- объект с фиктивным значением ***fictivevalue***(**X**).

Запишем следующие теоремы для простых отношений, вытекающие из определения статуса значения объекта:

realvalue(**X**), ***objconst***(**X**, **A**) = ***true***. (24)

~r(**In**, **Out**, _, _, **A**), (***member***(**X**, **In**), ***realvalue***(**X**)), ***member***(**Y**, **Out**), ***realvalue***(**Y**) = ***true***. (25)

~r(**In**, **Out**, _, _, **A**), ***member***(**X**, **In**), ***fictive***(**X**), (***member***(**Y**, **Out**), ***fictive***(**Y**)) = ***true***. (26)

~r(**In**, **Out**, _, _, **A**), ***member***(**X**, **In**), ***nonvalue***(**X**), (***member***(**Y**, **Out**), ***nonvalue***(**Y**)) = ***true***. (27)

Следует также отметить, что для функций **F**, геометрическая природа которых известна, истинно утверждение, выражающее статус определенности реальных значений объектов:

~r(**In**, **Out**, _, **F**, **A**), ***member***(**X**, **In**), ***indefinite***(**X**), (***member***(**Y**, **Out**, ***indefinite***(**Y**))=***true***. (28)

Теоремы (24 – 28) повышают эффективность исполнения предиката ***execute*** (11) за счет избирательной подстановки в него отношений, совершающих вычислительную работу.

Объекты **X**, удовлетворяющие правилу (29), сочетают в себе признаки вакантных объектов и объектов-констант:

orphan(**X**, **A**) ***if objconst***(**X**, **A**).
orphan(**X**, **A**) ***if objvacant***(**X**, **A**). (29)

Если совокупность отношений не пуста, то в ней всегда имеются объекты **X**, обеспечивающие истинность (29).

Введем правило (30), конкретизирующее объекты **X**:

kid(**X**, **A**) ***if*** ***~r***(_, **Out**, _, _, **A**), ***member***(**X**, **Out**), ***not parent***(**X**, _, **A**). (30)

Если совокупность отношений {**R**} не пуста, то в ней всегда имеются объекты, конкретизирующие **X** при истинности (30). С любым из объектов **Z**, для которых утверждение ***kid***(**Z**, **A**) ложно, можно сопоставить объект **Y** с истинным значением предиката ***kid***(**Y**, **A**), если исходную совокупность отношений дополнить отношением

~r([**Z**|[ ]], [**Y**|[ ]], _, ***copy***, **A**)., (31)

в котором функция ***copy*** присваивает значения объекта **Z** значениям объекта **Y**. В соответствии с (24 – 28) статус значения **Y** эквивалентен статусу значения **Z**. Поэтому объект **Y** можно считать выразителем значения **Z** и при необходимости «распространять» через **Y** действие (30) на объект **Z**.

Пусть совокупность отношений {**R**} имеет совместный состав (18). Состав корректен, если для любого **X** истинно (32):

not ancestor(**X**, **X**, _). (32)

Именованную (возможно, пустую) совокупность отношений {**R**}, обладающую корректным составом, будем называть алгоритмом (геометрической машиной). Принадлежность отношения к алгоритму в (6) выражается параметром **A**.

Алгоритм представляет собой объект, инкапсулирующий все отношения своего состава и объявленные в них объекты. Создание алгоритма заключается в сопоставлении с ним пустой совокупности отношений {**R**}=[ ] и присвоении ему уникального имени. Вычислительная работа алгоритма состоит в согласованном выполнении вычислительной работы всех отношений его состава, приводящей к нахождению значений всех декларированных объектов.

Конструирование КГМ в Симплексе – это, прежде всего, модификация состава отношений алгоритма. Определены две основные операции модификации состава: включение отношения в состав алгоритма и исключение отношения из его состава. Модификация состава является причиной активизации предиката ***execute*** (11), для выполнения вычислительной работы всех отношений алгоритма.

Алгоритм как объект сам может быть выражен в форме отношения. Будем формально относить к естественным входным и выходным параметрам алгоритма все его объекты, для которых истинны утверждение (33) и (34) соответственно. Любой из объектов алгоритма можно отнести к естественным выходным параметрам, при условии выполнения (31).

input (**A**, **L**) ***if bagof*** (**X**, ***orphan*** (**X**, **A**), **L**). (33)

output (**A**, **L**) ***if bagof*** (**X**, ***kid*** (**X**, **A**), **L**). (34)

Исключение отношения из совокупности {**R**}, не нарушает корректности его состава. Наоборот, включение отношения-претендента **R*** в совокупность отношений {**R**}, способно нарушить корректность {**R**+**R***}.

Пусть алгоритм **A** определен на непротиворечивой совокупности отношений {**R**}. Совокупность отношений утрачивает свойство совместности, если в нем можно выделить хотя бы одну несовместную пару (17). Обозначим количество выходных параметров отношения **R*** величиной **k**. Так как все выходные параметры **R*** различны (4), то оно способно образовать в {**R**} до **k** несовместных пар. Целевое утверждение ***has_contrariety*** (35), позволяет выявить **m**≤**k** отношений **R1**, противоречащих **R***=***~r***(_, **Out**, _, _, **A**).

has_contrariety(**R**, ***~r***(_, **Out**, _, _, **A**), **R1**) ***if member***(**R1**, **R**), ***contrariety*** (**R1**, ***~r***(_, **Out**, _, _, **A**))). (35)

Отношение-претендент **R***=***~r**(**In**, **Out**, _, _, **A**) не образует противоречивых пар с отношениями из **R**, если оно успешно конкретизирует аргументы правила ***no_contrariety*** (36).

no_contrariety(**R**, ***~r***(_, **Out**, _, _, **A**)) ***if mem-*** (36)

ber(**R1**, **R**), ***not contrariety***(**R1**, **~*r***(_, **Out**, _, _, **A**)).

Отношение-претендент **R*=~*r***(**In**, **Out**, _, _, **A**) не образует циклической зависимости объектов в новом составе алгоритма **A**, если достижима цель, выраженная антецедентом импликации (37).

no_cycle (**R**, **~*r***(**In**, **Out**, _, _, **A**)) ***if not*** (***member***(**~*r*(In1**, _, _, _, **A**), **R**), ***member***(**X**, **In1**), ***member***(**X**, **Out**), ***member***(**~*r*(_, Out2**, _, _, **A**), **R**), ***member***(**Y**, **Out2**), ***member***(**X**, **In**)). (37)

При выполнении составной цели (38), являющейся условием обеспечения корректности состава алгоритма,

no_contrariety (**R**, **R***), ***no_cycle*** (**R**, **R***). (38)

отношение-претендент **R*** может быть включено в его состав.

Операция включения отношения в состав алгоритма приводит к ликвидации вакансии объекта **X**, если истинно (39)

~*r*(_, **Out**, _, _, **A**), ***member*** (**X**, **Out**), ***objvacant***(**X**, **A**). (39)

Операции, обусловленные изменением состава отношений алгоритма **A** и приводящие к неуспеху цели (38), будем называть устранением противоречия отношений. Противоречие может быть устранено одним из двух способов:

1) с сохранением прежнего (корректного) состава {**R**} алгоритма **A**, при этом претендент **R*** отвергается;
2) с включением отношения-претендента **R*** в состав {**R**}.

Первый способ тривиален. Во втором случае из состава {**R**} должны быть исключены все отношения **R1**, успешно конкретизирующие (35), а отношение **R*** включено в {**R**}.

Алгоритм как преобразователь может рассматриваться с позиций открытости или закрытости его структуры. На алгоритм с закрытой структурой распространяются свойства простых отношений (4). Любое отношение с открытой структурой и корректным составом {**R**} – сложное. В отличие от простого отношения требование обязательной зависимости всех выходных параметров от любого из входных параметров в сложном отношении утрачивает силу. В данном случае правило (37) и цель (38) не могут быть применены непосредственно. Для поддержания корректности {**R**} необходимо опровергать факт присутствия циклической зависимости всех выходных параметров **A,** учитывая как внутреннюю структуру **A**, так и влияние внешних связанных с **A** отношений**.**

Сложное отношение – основа для группового проектирования

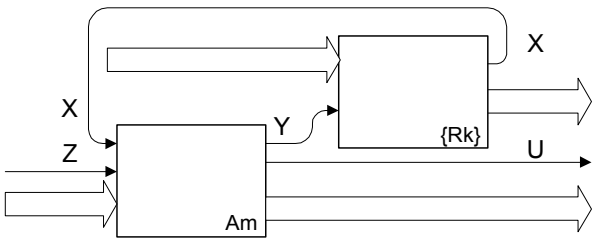

Рис. 1

и классификации КГМ, а также создания открытых функциональных наборов вычислительных процедур. Сложное отношение (алгоритм **Am**) «скрывает» в себе общие функциональные механизмы модели, а управление вычислительными процессами осуществляется за счет организации обратных связей внешними отношениями {$\mathbf{R_k}$} (рис. 1).

Подводя итог, отметим, что КГМ в среде системы Симплекс представлена алгоритмами, логическими правилами и фактами, составляющими единый проект. Симплекс синтезирует корректный состав отношений КГМ на этапах конструирования или применения модели. Динамическую реорганизацию программы во время ее исполнения можно осуществлять с помощью системных предикатов ***addline***(**R**) и ***removeline***(**A**, **X**), подчиненных правилам (35) – (38). В названных предикатах **R –** добавляемое отношение, **A** – алгоритм, **X** – объект, декларированный в удаляемом отношении. Типовая программа, созданная на основе КГМ, состоит из следующих компонентов:

- программно неизменяемой секции DATABASE – совокупности отношений, генерируемых средой Симплекса;
- секции PROGRAM, в которой допускается размещение необязательной логической программы на диалекте языка PROLOG и осуществляется динамический синтез отношений посредством предикатов ***addline*** и ***removeline***.

Вычисления осуществляется посредством исполнения системной цели ***execute*** над множеством отношений всех алгоритмов проекта Симплекса и инициируются любым изменением состава отношений или по запросу логической программы. Логический интерпретатор системы осуществляет решение и иных пользовательских целевых утверждений, составляющих секцию GOALS.

3. ПРИМЕНЕНИЕ МЕТОДИКИ

Рассмотрим несколько примеров анализа и синтеза КГМ, осуществляемых в среде системы Симплекс.

Устранения дублированных отношений в КГМ

Пусть {**R**} такова, что истинно

~*r*(**In**, **Out1**, **Fm**, **F**, **A**), **~*r*** (**In**, **Out2**, **Fm**, **F**, **A**), **Out1≠ Out2**. (40)

Два отношения (40) дублируют друг друга, т.е. выполняют одну и ту же вычислительную работу. Присутствие в алгоритме отношений (40) не нарушает корректность его состава. Однако в целях сокращения временных затрат на вычисления одно из отношений следует изъять, переименовав при этом входные параметры **Y** всех отношений из {**R**} на **X** (41).

nth(**N**, **X**, **Out1**), ***nth***(**N**, **Y**, **Out2**), **X≠Y**. (41)

Введем правило подстановки (42) [8] для синтеза термов

subst(**Old**, **New**, **Old**, **New**).

subst(**Old**, **New**, **T**, **T**) ***if constant***(**T**), **T≠Old**.

subst(**Old**, **New**, **T**, **T1**) ***if compound***(**T**), ***functor***(**T**, **F**, **N**), ***functor***(**T1**, **F**, **N**), ***subst***(**N**, **Old**, **New**, **T**, **T1**).
subst(**N**, **Old**, **New**, **T**, **T1**) ***if*** **N>0**, ***arg***(**N**, **T**, **Arg**), ***subst***(**Old**, **New**, **Arg**, **Arg1**), ***arg***(**N**, **T1**, **Arg1**), **N1=N-1**, ***subst***(**N1**, **Old**, **New**, **T**, **T1**). (42)

subst(**0**, **Old**, **New**, **T**, **T1**).

Упрощение алгоритма достигается за счет выполнения целевого утверждения (43)

simplify(**~*r***(**In**, **Out1**, **Fm**, **F**, **A**), **~*r***(**In**, **Out2**, **Fm**, **F**, **A**), **R**, **RR**, **RNew**) ***if member***(**RR**, **R**), **~*r*** (**In**, **Out2**, **Fm**, **F**, **A≠ RR**, ***member***(**X**, **Out1**), ***nth***(**N**, **X**, **Out1**), ***nth***(**N**, **Y**, **Out2**), ***subst***(**Y**, **X**, **RR**, **RNew**)., (43)

где **RNew** – отношение с измененными входными параметрами, модифицировавшими параметры отношения **RR**. В результате модификации **RR\RNew** возможно появление новых дублирующих пар. Следовательно, следует осуществлять проверку (43) до окончательного неуспеха цели (40).

Реструктуризация КГМ для параллельных вычислений

Под «уровнем» объекта в алгоритме будем понимать число **N**, определенное правилом ***level*** (44)

level (**A**, **X**, 0) ***if orphan*** (**X**, **A**), **!** . (44)

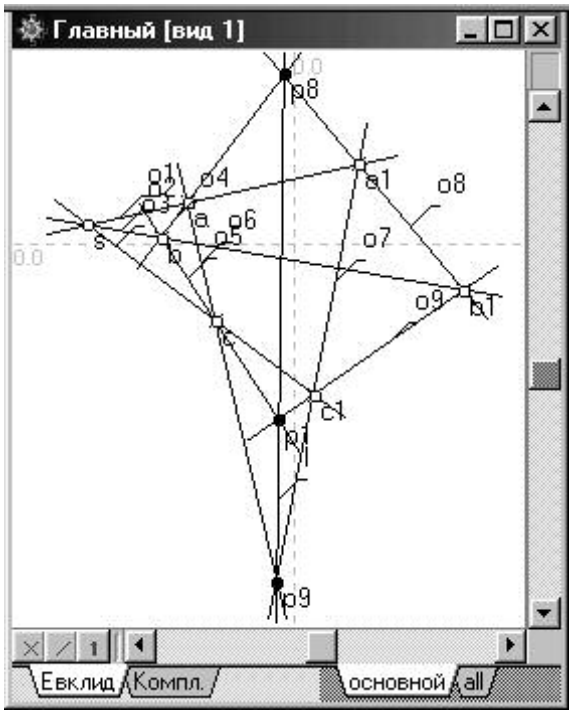

Рис. 2

level (**A**, **X**, **N**) ***if findall*** (**N1**, ***parentlevel*** (**A**, **X**, **N1**), **L**), ***maximum2*** (**L**, **N2**), **N**=**N2**+1.

parentlevel (**A**, **X**, **N**) ***if parent*** (**A**, **Y**, **X**), ***level*** (**A**, **Y**, **N**).

maximum2 ([**X**|**Xs**], **M**) ***if max*** (**Xs**, **X**, **M**).

max ([**X**|**Xs**], **Y**, **M**) ***if*** **X**<=**Y**, ***max*** (**Xs**, **Y**, **M**).

max ([**X**|**Xs**], **Y**, **M**) ***if*** **X**>**Y**, ***max*** (**Xs**, **X**, **M**).

max ([ ], **M**, **M**).

Независимость объектов **X** с одинаковым значением уровня **N** допускает одновременность (параллельность) выполнения вычислительной работы декларирующих их отношений за **m+1** тактов, где **m** – максимальное значение уровня в алгоритме. Данный факт может быть использован для проектирования и аппаратной реализации специализированных вычислительных устройств.

Пример встроенной в систему экспертной функции

Приведенная программа составляет часть экспертной подсистемы Симплекса, выявляющей изоморфизм моделей, и предназначена для доказательства (и учета этого факта при моделировании) инцидентности объектов в конфигурации Дезарга (рис. 2).

PROGRAM

inc(**A**,**B**) ***if*** ***~r***([[**A**], [_]], [**B**], _, 'Прямая точкой и углом', _).
inc(**A**,**B**) ***if*** ***~r***([[**A**], [_]], [**B**], _, 'Прямая двумя точками', _).
inc(**A**,**B**) ***if*** ***~r***([[_], [**A**]], [**B**], _, 'Прямая двумя точками', _).
inc(**A**,**B**) ***if*** ***~r***([[**A**], [_]], [**B**], _, 'Точка пересечения прямых', _).
inc(**A**,**B**) ***if*** ***~r***([[_], [**A**]], [**B**], _, 'Точка пересечения прямых', _).
inc(**A**,**B**) ***if*** ***~r***([[**A**], [_]], [**B**], _, 'Точка инцид. с объектом', _).
inc(**B**,**A**) ***if*** ***~r***([[**A**], [_]], [**B**], _, 'Прямая точкой и углом', _).
inc(**B**,**A**) ***if*** ***~r***([[**A**], [_]], [**B**], _, 'Прямая двумя точками', _).
inc(**B**,**A**) ***if*** ***~r***([[_], [**A**]], [**B**], _, 'Прямая двумя точками', _').
inc(**B**,**A**) ***if*** ***~r***([[**A**], [_]], [**B**], _, 'Точка пересечения прямых', _).
inc(**B**,**A**) ***if*** ***~r***([[_], [**A**]], [**B**], _, 'Точка пересечения прямых', _).
inc(**B**,**A**) ***if*** ***~r***([[**A**], [_]], [**B**], _, 'Точка инцид. с объектом', _).

incid(**A**, **B**) ***if inc***(**A**, **B**).
incid(**P8**, **L**) ***if triplet***(**O4**, **P8**, **A**, **B**), ***triplet***(**O8**, **P8**, **A1**, **B1**), **O4≠O8**, **A1>A**, **B1>B**, **B>A**, **B1>A1**, ***triplet***(**O1**, **S**, **A**, **A1**), ***triplet***(**O2**, **S**, **B**, **B1**), **O1≠O2**, ***triplet***(**O3**, **S**, **C**, **C1**), **C1>C**, **O2≠O3**, **O1≠O3**, ***triplet***(**O6**, **A**, **C**, **P9**), ***triplet***(**O5**, **B**, **C**, **P1**), ***triplet***(**O7**, **A1**, **C1**, **P9**), ***triplet***(**O9**, **B1**, **C1**, **P1**), ***inc***(**L**, **P9**), ***inc***(**L**, **P1**).

line(**A**) ***if*** ***~r*** (_, [**A**], _, 'Прямая точкой и углом', _).
line(**A**) ***if*** ***~r*** (_, [**A**], _, 'Прямая двумя точками', _).

point(**A**) ***if*** ***~r***(_, [**A**], _, 'Точка пересечения прямых', _).
point(**A**) ***if*** ***~r***(_, [**A**], _, 'Точка принадлежит объекту', _).
point(**A**) ***if*** ***~r***(_, [**A**], _, 'Точка задана координатами', _).

triplet(**O**,**A**,**B**,**C**) ***if inc***(**O**,**A**),***inc***(**O**,**B**),***inc***(**O**,**C**),**A≠B**,**B≠C**,**A≠C**.

DATABASE

~r ([[-293],[26.5]], [**s**], '0', 'Точка задана координатами', 'D').
~r ([[**s**],[12.6]], [**o1**], '0', 'Прямая точкой и углом', 'D').
~r ([[**s**],[350.2]], [**o2**], '0', 'Прямая точкой и углом', 'D').
~r ([[**s**],[323.1]], [**o3**], '0', 'Прямая точкой и углом', 'D').
~r ([[**o1**],[1.448]], [**a**], '0', 'Точка инцид. с объектом', 'D').
~r ([[**o2**],[1.054]], [**b**], '0', 'Точка инцид. с объектом', 'D').
~r ([[**o3**],[2.261]], [**c**], '0', 'Точка инцид. с объектом', 'D').
~r ([[**o1**],[3.924]], [**a1**], '0', 'Точка инцид. с объектом', 'D').
~r ([[**o2**],[5.387]], [**b1**], '0', 'Точка инцид. с объектом', 'D').
~r ([[**o3**],[3.987]], [**c1**], '0', 'Точка инцид. с объектом', 'D').
~r ([[**a**],[**b**]], [**o4**], '0', 'Прямая двумя точками', 'D').
~r ([[**b**],[**c**]], [**o5**], '0', 'Прямая двумя точками', 'D').
~r ([[**a**],[**c**]], [**o6**], '0', 'Прямая двумя точками', 'D').
~r ([[**a1**],[**c1**]], [**o7**], '0', 'Прямая двумя точками', 'D').
~r ([[**a1**],[**b1**]], [**o8**], '0', 'Прямая двумя точками', 'D').
~r ([[**b1**],[**c1**]], [**o9**], '0', 'Прямая двумя точками', 'D').
~r ([[**o7**],[**o6**]], [**p9**], '0', 'Точка пересечения прямых', 'D').
~r ([[**o5**],[**o9**]], [**p1**], '0', 'Точка пересечения прямых', 'D').
~r ([[**p9**],[**p1**]], [**l**], '0', 'Прямая двумя точками', 'D').
~r [[**o4**],[**o8**]], [**p8**], '0', 'Точка пересечения прямых', 'D').

GOALS

incid(**p8**,**X**).

SOLUTION

X=o4; **X=o8**; **X=l**.

Каждая новая точка унифицирует переменную **P** в цели ***incid***(**P**, **L**). **L** – прямая, претендующая на установление признака инцидентности с **P**.

Моделирование поверхностей и управление их формой

Пусть задан алгоритм **A** с двумя входными и тремя выходными параметрами. Структура **A** закрыта, следовательно, характер функции **r=r(u,v)**, определяющей в трехмерном пространстве поверхность α, неизвестен. Зададимся начальными и конечными значениями параметров **u** и **v** – $u_н$, $u_к$ и $v_н$, $v_к$ и разобьем интервалы [$u_н$, $u_к$] и [$v_н$, $v_к$] на **m** и **n** частей соответственно. Т.о. получено дискретное множество из **k**=(**m**+**1**)(**n**+**1**) параметрических узлов. Для моделирования поверхности дискретным множеством точек, инцидентных с α, необходимо рассчитать все **p** отношений алгоритма **k** раз.

$$k_A = kp \qquad (45)$$

k_A – суммарные временные затраты, необходимые на однократный расчет отношений алгоритма **A**. При открытой структуре **A** количество вычислений может быть сокращено на формальной основе.

Введем несколько правил, оптимизирующих расчет по временному критерию. Два объекта независимы, если истинно (46).

independent (**X**, **Y**, **A**) ***if not ancestor*** (**A**, **X**, **Y**), ***not ancestor*** (**A**,**Y**, **X**). (46)

Для соблюдения условия независимости тройки выходных объектов **x**, **y** и **z** необходимо успешное выполнение цели (47)

independent (**X**,**Y**,**A**), ***independent*** (**X**,**Z**,**A**), ***independent*** (**Y**,**Z**,**A**). (47)

при конкретизации **X=x**, **Y=y**, **Z=z**. Аналогичные ограничения накладываются на параметры **u** и **v**

independent (**U**,**V**,**A**). (48)

при конкретизации **U=u**, **V=v**. Для выражения **r=r(u,v)** необходимо, чтобы каждая из координат **x**, **y** и **z** была зависимой от обоих входных параметров **u** и **v**.

ancestor (**A**,**U**,**X**), ***ancestor*** (**A**,**V**,**X**), ***ancestor*** (**A**,**U**,**Y**), ***ancestor*** (**A**,**V**,**Y**), ***ancestor*** (**A**,**U**,**Z**), ***ancestor*** (**A**,**V**,**Z**). (49)

Отношения, используемые для расчета значений **x**, **y** и **z**, составляют подмножество $\{\mathbf{R}\}_{\mathbf{xyz}}$ алгоритма **A** и выявляются целью ***inXYZ*** посредством трехкратного применения (50) с конкретизациями **X=x**; **X=y**; **X=z**.

inXYZ(**R**, **X**, **A**) ***if ancestor*** (**A**, **V**, **X**), **R=*relation***(_, **Out**, _, **A**), ***member***(**X**, **Out**). (50)

Не вошедшие в $\{\mathbf{R}\}_{\mathbf{xyz}}$ отношения алгоритма **A** составляют подмножество $\{\mathbf{R}\}\backslash\{\mathbf{R}\}_{\mathbf{xyz}}$, в процессе расчета координат точек не участвуют и, следовательно, могут быть исключены из

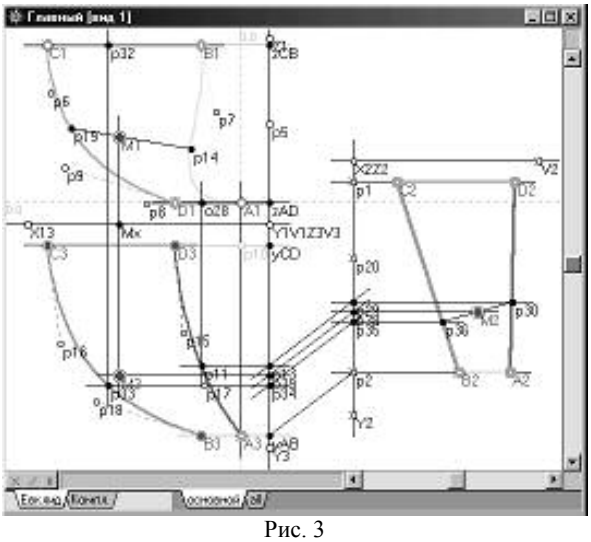

Рис. 3

рассмотрения без ущерба для решения поставленной задачи.

Эффективность расчета можно повысить, если осуществить выбор лучшего из вариантов последовательности $\{\mathbf{R}\}_{\mathbf{xyz}}$, состав которой регулируется порядком следования $\mathbf{u_i}$ и $\mathbf{v_j}$.

группа отношений	кол-во отношений	**u**	**v**
$\{\mathbf{R}\}_{\mathbf{0}}$	$\mathbf{k_0}$	не зависят	не зависят
$\{\mathbf{R}\}_{\mathbf{u}}$	$\mathbf{k_u}$	зависят	не зависят
$\{\mathbf{R}\}_{\mathbf{v}}$	$\mathbf{k_v}$	не зависят	зависят
$\{\mathbf{R}\}_{\mathbf{uv}}$	$\mathbf{k_{uv}}$	зависят	зависят

Пусть на момент расчета точек α состав алгоритма **A** не изменяется и соблюдены условия (47) – (50). В $\{\mathbf{R}\}_{\mathbf{xyz}}$ выделяются четыре группы отношений по признаку зависимости от **u** и **v**. Обозначим через $\mathbf{k_0}$, $\mathbf{k_u}$, $\mathbf{k_v}$ и $\mathbf{k_{uv}}$ суммарные временные затраты, необходимые для однократного расчета отношений по группам $\{\mathbf{R}\}_{\mathbf{0}}$,$\{\mathbf{R}\}_{\mathbf{u}}$,$\{\mathbf{R}\}_{\mathbf{v}}$ и $\{\mathbf{R}\}_{\mathbf{uv}}$ соответственно.

$$\mathbf{k_0+k_u+k_v+k_{uv}=k\{R\}_{xyz}}, \quad (51)$$

где $\mathbf{k}\{\mathbf{R}\}_{\mathbf{xyz}}$ – суммарные временные затраты, необходимые для однократного расчета отношений совокупности $\{\mathbf{R}\}_{\mathbf{xyz}}$. Из сравнения с (45) видно, что $\mathbf{k}\{\mathbf{R}\}_{\mathbf{xyz}} \le \mathbf{k_A}$.

Значения объектов, декларированных отношениями группы $\{\mathbf{R}\}_{\mathbf{0}}$, остаются неизменными на протяжении всего расчета, и поэтому их вычисление однократно. Для обеспечения выбора всевозможных пар значений $\mathbf{u_i}$ и $\mathbf{v_j}$, **i=1…m**, **j=1…n** организуется двойной вложенный цикл, в котором в качестве внешнего параметра выбирается параметр **u**, а в качестве внутреннего – **v**. Тогда группы отношений $\{\mathbf{R}\}_{\mathbf{0}}$,$\{\mathbf{R}\}_{\mathbf{u}}$,$\{\mathbf{R}\}_{\mathbf{v}}$ и $\{\mathbf{R}\}_{\mathbf{uv}}$ распределяются относительно тела цикла в соответствии с признаком зависимости от **u** и **v** следующим образом:

1. Отношения $\{\mathbf{R}\}_{\mathbf{0}}$ выносятся за пределы двойного цикла.
2. Отношения группы $\{\mathbf{R}\}_{\mathbf{u}}$ находятся в области действия внешнего параметра цикла **u**.
3. Отношения групп $\{\mathbf{R}\}_{\mathbf{v}}$ и $\{\mathbf{R}\}_{\mathbf{uv}}$ находятся в общей области действия параметров **u** и **v**.

При таком распределении отношений в цикле суммарные временные затраты $\mathbf{t_{uv}}$ для расчета всех точек поверхности рассчитываются по формуле:

$$\mathbf{t_{uv}=k_0+mk_u+mnk_v+mnk_{uv}}. \quad (52)$$

При обратном порядке параметров **u** и **v** в цикле

$$\mathbf{t_{vu}=k_0+nk_v+mnk_u+mnk_{uv}}. \quad (53)$$

Коэффициент **f** (54), рассчитываемый с использованием формул (52) и (53), выражает отношение временных затрат с различным порядком выбора параметров **u** и **v**. Значение **f** служит критерием выбора стратегии расчета до его реального осуществления.

$$\mathbf{f=m\,(k_u+nk_v)\,/\,n\,/\,(\,k_v+mk_u\,)} \quad (54)$$

Рис. 4

В качестве примера приведем одну из конструктивных схем, разработанных автором, реализующую ключевой метод моделирования поверхности. Основные принципы ключевых и гиперключевых методов моделирования поверхностей изложены в [9] и [10].

Точки **p5** и **p20** находятся в непосредственной зависимости от параметров **u** и **v**, изменяющихся в пределах [0…1] каж-

дый. Геометрическое построение, приведенное на рис. 3, устанавливает связь точек **p5** и **p20** с выходными параметрами данного алгоритма – величинами **xm**, **ym** и **zm**. **xm** есть приращение по X между точками **Y1V1Z3V3** и **Mx**; **ym** есть приращение по Y между точками **Mx** и **M3**; **zm** есть приращение по Y между точками **M1** и **Mx**. Управление формой поверхности осуществляется интерактивным воздействием на свободные геометрические объекты (29) представленной модели. Лоскут синтезированной поверхности изображен на рис. 4.

Решение позиционных задач на моделях многомерных пространств

Система Симплекс обеспечивает технологию объектно-ориентированного визуального проектирования (ООП) КГМ. Производный класс объектов определяется как структура, состоящая из объектов как непроизводных (встроенных в систему), так и производных классов. Использование значений экземпляров таких объектов осуществляется через обращение к специальным алгоритмам – методам классов.

Описание класса начинается с его именования и указания признака наследования (если он есть). Для совершения вычислительной работы над объектами класса необходимо спроектировать или использовать готовые методы, а также определить их интерфейс – входные и выходные параметры (аналогично (33) и (34)). Вызов метода – это ввод в проект обычного отношения. Описание технологии декларативно-визуального проектирования методов выходит за рамки данной статьи.

Для объектов производных классов допускается создание метода с пустым списком выходных параметров. Такой метод носит название метода глобальных переменных класса. Он представляет собой общую и единственную для всех методов совокупность отношений, которая исполняется в первую очередь при обращении к любому методу как во время исполнения вычислительной работы, так и на этапе проектирования КГМ. Метод **M** глобальных переменных класса может согласовываться с построениями задачи через входные параметры.

$$\sim r\ (\mathbf{In},\ [\ ],\ \mathbf{Fm},\ \mathbf{Fconst},\ \mathbf{M}).,\qquad (55)$$

Технология ООП находит применение в проектировании КГМ с дискретно-непрерывными структурами, в частности при моделировании пространств высших размерностей.

Приведем пример проектирования метода, предназначенного для построения точки пересечения трехмерного пространства и прямой в четырехмерном пространстве на модели $\mathbf{G^4_{22}}$ [3]. (Курсивом выделены вспомогательные отношения, необходимые для визуализации построений при проектировании).

Ядро модели [3], [4] выражено двумя исключенными прямыми **u1** и **u2**, инкапсулированными в классе **g_422**. В классе определен метод глобальных переменных, «распространяющий» значения **u1** и **u2** во все потомки класса **g_422**. Линейные объекты четырехмерного пространства – точка, прямая, плоскость и трехмерное пространство выражены классами **point**, **line**, **plane** и **space** соответственно и являются наследниками класса **g_422**.

Ядро модели $\mathbf{G^4_{22}}$: g_422[*u1*, *u2*] (**u1**, **u2**)
Прямая <**u1**> координатами <**-459**>, <**268**>, <**481**>, <**133**>.
Прямая <**u2**> координатами <**-409**>, <**-136**>, <**485**>, <**-99**>.

На модели $\mathbf{G^4_{22}}$ объект класса **point** представлен двумя точками плоскости.

Точка задана точками полей point[*t1*, *t2*] (**point**; **t1**, **t2**)
Точка <**t1**> задана координатами <**-146**> и <**97**>.
Точка <**t2**> задана координатами <**-67**> и <**-56**>.

Прямая на модели $\mathbf{G^4_{22}}$ (класс **line**) выражена двумя прямолинейными рядами точек, задающими проективитет **pr** в точечных рядах. Прямая задана объектами **point1** и **point2**. Ядро оказывает влияние на точечные ряды через точки **p5** и **p6**.

Прямая задана двумя точками: line[*l1*, *l2*, *pr*] (**line**; **point1**, **point2**)
*Точка <**p1**> задана координатами <**-232**> и <**147**>.*
*Точка <**p2**> задана координатами <**-273**> и <**-78**>.*
*Точка <**p3**> задана координатами <**-22**> и <**169**>.*
*Точка <**p4**> задана координатами <**53**> и <**-91**>.*
*Точка <**point1**> образуется методом "Точка задана точками полей" класса "point": Точка первого поля <**p1**>, Точка второго поля <**p2**>*
*Точка <**point2**> образуется методом "Точка задана точками полей" класса "point": Точка первого поля <**p3**>, Точка второго поля <**p4**>*
Прямая <**l1**> задана точками <**point1**.**t1**> и <**point2**.**t1**>.
Прямая <**l2**> задана точками <**point1**.**t2**> и <**point2**.**t2**>.
Точка <**p5**> есть пересечение прямых <**l1**> и <**u1**>.
Точка <**p6**> есть пересечение прямых <**l2**> и <**u2**>.
Проективитет <**pr**> с носителями <**l1**>-<**l2**> и парами точек <**point1**.**t1**>-<**point1**.**t2**>, <**point2**.**t1**>-<**point2**.**t2**>, <**p5**>-<**p6**>.

Трехмерное пространство на модели $\mathbf{G^4_{22}}$ (класс **space**) выражено двумя пучками прямых, задающими проективитет **pr** в линейных пучках. Пространство задано парой центров пучков (инцидентных с **u1** и **u2** соответственно) и двумя парами лучей этих пучков. Третья пара прямых определена ядром (прямые **u1** и **u2**).

Трехмерное пространство задано пучками: space[*pr*] (**space**; **p1**, **o1**, **o2**, **p2**, **o3**, **o4**) :
*Точка <**p1**> принадл. объекту <**u1**> с параметром <0.2>.*
*Точка <**p2**> принадл. объекту <**u2**> с параметром <0>.*
*Прямая <**o1**> задана точкой <**p1**> и углом <**30**> к оси OX.*
*Прямая <**o2**> задана точкой <**p1**> и углом <**50**> к оси OX.*
*Прямая <**o3**> задана точкой <**p2**> и углом <**20**> к оси OX.*
*Прямая <**o4**> задана точкой <**p2**> и углом <**80**> к оси OX.*
Проективитет <**pr**> с центрами <**p1**> - <**p2**> и парами прямых <**u1**> - <**u2**> , <**o2**> - <**o3**>, <**o1**> - <**o4**> .

Задача о нахождении точки пересечения трехмерного пространства с прямой линией в четырехмерном пространстве сводится к поиску двойных точек проективитета (рис. 5).

Точка пересечения пространства и прямой (**point3**; **line1**, **space1**):
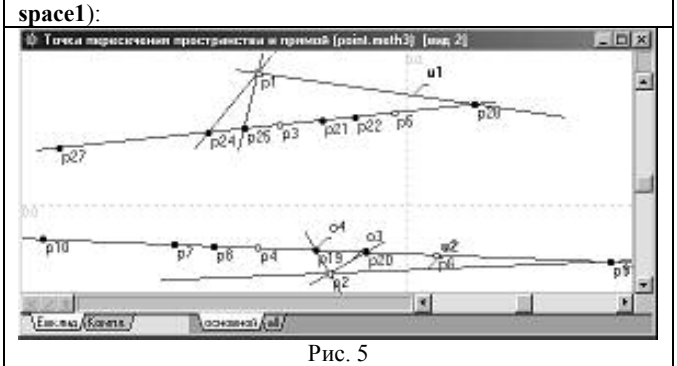
Рис. 5
*Точка <**p1**> принадл. объекту <**u1**> с параметром <**0.2**>.*
*Точка <**p2**> принадл. объекту <**u2**> с параметром <**0.3**>.*
*Прямая <**o1**> задана точкой <**p1**> и углом <**50**> к оси OX.*
*Прямая <**o2**> задана точкой <**p1**> и углом <**75**> к оси OX.*
*Прямая <**o3**> задана точкой <**p2**> и углом <**30**> к оси OX.*
*Прямая <**o4**> задана точкой <**p2**> и углом <**120**> к оси OX.*
*space <**space1**> образуется методом "Пространство зада-*

*но пучками" класса "space": Центр 1 <**p1**>, Прямая 1 1 <**o2**>, Прямая 1 2 <**o1**>, Центр 2 <**p2**>, Прямая 2 1 <**o3**>, Прямая 2 2 <**o4**>*

*Точка <**p3**> задана координатами <**-232**> и <**147**>.*

*Точка <**p4**> задана координатами <**-273**> и <**-78**>.*

*Точка <**p5**> задана координатами <**-22**> и <**169**>.*

*Точка <**p6**> задана координатами <**53**> и <**-91**>.*

*Точка <**point1**> образуется методом "Точка задана точками полей" класса "point": Точка первого поля <**p3**>, Точка второго поля <**p4**>*

*Точка <**point2**> образуется методом "Точка задана точками полей" класса "point": Точка первого поля <**p5**>, Точка второго поля <**p6**>*

*Прямая <**line1**> образуется методом "Прямая двумя точками" класса "line": Точка 1 <**point1**>, Точка 2 <**point2**>*

Точка <**p19**> есть пересечение прямых <**o4**> и <**line1**.**l2**>.

Точка <**p20**> есть пересечение прямых <**o3**> и <**line1**.**l2**>.

Образ <**p21**> точки <**p19**> в проективитете <**line1**.**pr**> в первом поле.

Образ <**p22**> точки <**p20**> в проективитете <**line1**.**pr**> в первом поле.

Точка <**p23**> есть пересечение прямых <**line1**.**l1**> и <**u1**>.

Точка <**p24**> есть пересечение прямых <**line1**.**l1**> и <**o1**>.

Точка <**p25**> есть пересечение прямых <**o2**> и <**line1**.**l1**>.

Проективитет <**pr5**> с носителями <**line1**.**l1**> - <**line1**.**l1**> и парами точек <**p23**> - <**p23**> , <**p22**> - <**p25**>, <**p21**> - <**p24**> .

Точки <**p26**> и <**p27**> - дв. точки проективитета <**pr5**>.

Образ <**p7**> точки <**p24**> в проективитете <**line1**.**pr**> во втором поле.

Образ <**p8**> точки <**p25**> в проективитете <**line1**.**pr**> во втором поле.

Точка <**p9**> есть пересечение прямых <**line1**.**l2**> и <**u2**>.

Проективитет <**pr1**> с носителями <**line1**.**l2**> - <**line1**.**l2**> и парами точек <**p9**> - <**p9**> , <**p8**> - <**p20**>, <**p7**> - <**p19**> .

Точки <**p10**> и <**p11**> - дв. точки проективитета <**pr1**>.

Точка <**point3**> образуется методом "Точка задана точками полей" класса "point": Точка первого поля <**p10**>, Точка второго поля <**p11**>

Виртуализация понятия ядра в сочетании с механизмом наследования открывает новые возможности для классификации моделей многомерных пространств. Переход между моделями эквивалентных классов может быть осуществлен формальной заменой ядер и их согласованием с текущими параметрами среды задачи.

4. ЗАКЛЮЧЕНИЕ

Разработанные методики автоматизированного синтеза и анализа КГМ с использованием системы Симплекс позволили:

- снять инструментальные ограничения, сопутствовавшие геометро-синтетическому методу решения задач
- повысить практическую значимость и подтвердить целесообразность применения теоретических разработок, выполненных методами конструктивной геометрии;
- разработать средства автоматизации научного труда для ведения исследований путем синтеза и реализации новых геометрических теорий и моделей.

С применением системы Симплекс решен ряд практических задач. Среди них:

- представление геометрических объектов и решение задач на евклидовой и проективной плоскостях;
- представление объектов линейных многомерных пространств на моделях с дискретно-непрерывной структурой;
- реализация конструктивных методов проектирования поверхностей для задач САПР;
- оптимизация и автоматизированное преобразование конструктивных геометрических алгоритмов по критериям минимизации их сложности и сокращения вычислительных затрат синтезируемых программ;
- логический анализ геометрических структур в задачах распознавания и классификации;
- синтез изображений с использованием конструктивных методов в целях автоматизации операций графического дизайна;
- автоматизация деятельности преподавателя и учащегося при обучении начертательной геометрии;

В настоящее время система Симплекс обеспечивает автоматический перевод разработанных в ее среде геометрических моделей в программы на алгоритмических языках Паскаль и Пролог, а также осуществляет информационный обмен с иными системами геометрического моделирования посредством форматов DXF, AI, CMX.

ЛИТЕРАТУРА

[1] Братко И. Программирование на языке ПРОЛОГ для искусственного интеллекта. – М.: Мир, 1990.

[2] Вальков К.И. Линейные преобразования многомерного пространства как средство геометрического моделирования в науке и технике: Дисс. …докт. техн. наук. – Л., 1964. – 388с.

[3] Волошинов В.А. Исследование некоторых геометрических алгоритмов: Дисс…канд. техн. наук.– Л., 1971. – 180 с.

[4] Волошинов В.А., Волошинов Д.В. О понятийном аппарате, символизации и базовых процедурах в проекционном моделировании / Геометрическое моделирование и компьютерная графика. Сб. науч. трудов. СПбГТУ – 1995 – № 454.

[5] Волошинов Д.В. Система программирования задач прикладной геометрии «Симплекс» / Фундаментальные исследования в технических университетах. Материалы н/м. конференции СПбГТУ. – СПб: СПбГТУ, 1997.

[6] Волошинов Д.В., Гвоздев М.А. Использование ключевых методов конструирования поверхностей при автоматизированном проектировании оболочек / Сб. трудов Первой электронной международной научно-технической конференции «Автоматизация и информатизация в машиностроении» – Тула: ТулГУ, 2000 – с. 84.

[7] Дарахвелидзе П. Г., Марков Е. П. Delphi — среда визуального программирования: — СПб.: BHV — Санкт-Петербург, 1996. — 352 с.

[8] Стерлинг Л., Шапиро Э. Искусство программирования на языке Пролог: – М.: Мир, 1990. – 235 с.

[9] Болотов В.П. Геометрический и программный комплекс интерактивного расчетно-графического программирования в САПР: Дис. … докт.техн.наук.– М., 1993 – 270 с.

[10] Котов И.И. Геометрические основы ключевых способов построения поверхностей,/- Труды ВЗЭИ. – 1959. – вып. 10.

Об авторе

***Волошинов Денис Вячеславович*,** к.т.н., доцент каф. прикладной геометрии и дизайна СПбГПУ.

тел. (812) 552-75-14; e-mail: kpgd@ phtf.stu.neva.ru