УДК 681.3

П. В. ГОРШКОВ, Л. В. КОВАЛЬ, А. В. ЛЯЛЕЦКИЙ,
С. П. РОДИМИН
(Киев)

НЕКОТОРЫЕ ПРИНЦИПЫ ЭФФЕКТИВНОГО ПРЕДСТАВЛЕНИЯ ЗНАНИЙ О ПРОБЛЕМНЫХ ОБЛАСТЯХ И ИХ ПРИМЕНЕНИЕ ДЛЯ ИНТЕЛЛЕКТУАЛИЗАЦИИ ИНТЕРФЕЙСА С ПАКЕТАМИ ГРАФИЧЕСКИХ ПРОГРАММ

В процессе постановки и решения прикладных задач при помощи ЭВМ естественным образом возникает несколько уровней представления информации о проблемных областях: 1) языки специалистов проблемных областей (машиностроение, самолетостроение и др.); 2) непрерывные и дискретные математические модели проблемных областей (геометрическая модель, информационная модель и др.); 3) инструментальное (базовое) обеспечение ЭВМ (системы программирования, пакеты прикладных программ (ППП), базы данных); 4) исполняющие устройства (графические устройства, станки с числовым программным управлением (ЧПУ), роботы-манипуляторы и др.).

Первый уровень рассмотрения позволяет учитывать характерные особенности и традиции проектирования в проблемных областях, а также выбирать принципы моделирования, отражающие эти особенности. Фиксация принципов моделирования дает возможность, переходя ко второму уровню, выбирать соответствующий математический аппарат для уточнения постановки задач и разработки алгоритмов их решения. Использование ЭВМ в процессе автоматизированного решения этих задач ведет к необходимости выбора на третьем уровне структур данных и их представления в памяти ЭВМ, а также разработки алгоритмов, осуществляющих преобразование информации в соответствии с выбранными методами. Четвертый уровень рассмотрения связан с планированием действий исполняющих устройств и использует детально разработанную информацию, полученную с предыдущего уровня.

В данной работе излагаются результаты исследования, проведенного на кафедре теоретической кибернетики Киевского государственного университета, по разработке эффективных средств представления, хранения и обработки информации о проблемных областях в интегрированных системах автоматизированного проектирования (САПР). Формулируются принципы, которые могут быть положены в основу разработки набора согласованных языковых средств различного уровня рассмотрения, и показывается применение этих принципов для интеллектуализации интерфейса с пакетами графических программ (ПГП). Под эффективностью представления понимаются удобства, предоставляемые неквалифицированному (относительно области программирования) пользователю — специалисту проблемной области (проблемная ориентация, возможность параметризации уровня объектов проблемной области, расширяемость в процессе эволюции проблемной области и т. п.). Под эффективностью хранения понимается повышение информативности представления знаний о проблемной области по отношению к традиционным средствам базового программного обеспечения. Под эффективностью обработки понимается возможность эффективного отображения (интерпретации) представленных знаний на средства базового программного обеспечения ЭВМ, а также встраивание средств логического вывода ответов на запросы об отношениях, имеющих место в проблемной области.

Традиционные подходы к представлению знаний в интегрированных САПР [1] исходят из жесткой структуры внешнего представления проектируемых объектов, индуцированной их кодированием в памяти ЭВМ. Это ограничивает творческие возможности пользователя до рамок, заложенных при проектировании системы. Настоящее исследование можно рассматривать как попытку взглянуть на эту проблему под несколько другим углом зрения, позволяющим с общих позиций представлять знания о различных проблемных областях, включая программирование. Фактически это приводит к выделению промежуточного уровня рассмотрения между математическими моделями и средствами программного обеспечения ЭВМ. Фиксация этого уровня в виде формальной модели представления знаний, согласованной с базовым программным обеспечением, позволяет систематически преодолевать большое расстояние от формализованных моделей проблемных областей к их реализации средствами базового программного обеспечения ЭВМ. Такой подход к автоматизации программирования в проблемных областях согласуется с основными принципами композиционного программирования [2].

В основе предлагаемого подхода лежит теория, развитая в [3] и позволяющая рассматривать различные модели вычислений как данные, допускающие возможность обработки с учетом их структуры, индуцированной вычислительными аспектами. Применение этого подхода к проблемной области опирается на имеющийся опыт по разработке системы моделирования роботов-манипуляторов с программным управлением [4] и анализ интегрированных систем, содержащих графическую компоненту [5, 6]. Результаты, относящиеся к машинной графике, представлены в [7—11].

Общая точка зрения на проблемные области, включая и программирование, заключается в следующем. В каждой проблемной области представляют интерес не только сами объекты (как правило, разных типов), но также преобразования (операции) и отношения (предикаты) над объектами, имеющие место в данной проблемной области. Поэтому на абстрактном уровне рассмотрения проблемную область удобно задавать в виде тройки, содержащей множества объектов, операций и отношений, т. е. в виде многосортной алгебраической системы [12]. Вводя символические обозначения для объектов, операций и отношений, получаем сигнатуру алгебраической системы. Составляя все возможные выражения (термы) из констант (символических обозначений объектов) при помощи символов операций, образуем множество так называемых основных термов, которые на синтаксическом уровне представляют объекты проблемной области. Множество основных термов (с операциями и отношениями) назовем базовой синтаксической системой и в дальнейшем будем отождествлять проблемную область с базовой синтаксической системой.

Представление объектов проблемной области на уровне базовой синтаксической системы плохо обозримо, не поддается классификации и параметризации, хотя, вообще говоря, позволяет вычислять любой объект, заданный в виде основного терма, если указана требуемая интерпретация символов операций, отношений и констант.

Выбор той или иной базовой синтаксической системы для представления сущностей проблемной области зависит от многих обстоятельств (класса решаемых задач, квалификации пользователя и т. п.). Например, в системе автоматизированного программирования для станков с ЧПУ APT [13] сигнатура базовой синтаксической системы содержит символы для обозначения основных геометрических объектов и их характеристик (POINT, CIRCLE, LINE, PLANE, CENTER, RADIUS и др.), операций движения (FROM, GO, GOBACK, GODOWN и др.) и технологических операций (SPINDL, FERDAT, COOLNT и др.). Примером основных термов этой системы могут служить

POINT/—1.0, —1.0, 0,0; CIRCLE/CENTER, 4.0, 3.0, 0.0, RADIUS, 2; FROM/—1.0,— 1.0, 0.0; COOLNT/OFF.

Существующая практика и теория программирования показывают, что в целях удобства задания вычислений в базовой алгебраической системе желательно расширять выразительные возможности базовой синтаксической системы, что может привести к построению целой иерархии алгоритмических и(или) логических расширений. Введение символов переменных в сигнатуру базовой синтаксической системы позволяет использовать термы с переменными. Например, в языке кодирования параметризованных типовых элементов чертежей системы ГРИС [14] предусмотрено фиксированное конечное множество переменных, относительно которых параметризуются типовые элементы чертежа. Примером терма с переменными относительно базовой синтаксической системы ГРИС может служить следующая запись:

точки $4U = U2 + A1$, $4V = V2 + A2$,

где $4U$ и $4V$ — символическое обозначение абсциссы и ординаты определяемой точки с номером 4; $U2$ и $V2$ — символическое обозначение абсциссы и ординаты ранее заданной точки с номером 2; $A1$ и $A2$ — параметры привязки точки 4 к точке 2, значения которых устанавливаются в момент формирования чертежа. Синтаксические термы могут иметь разметку, т. е. символы, которые не интерпретируются на базовой алгебраической системе, а выполняют роль служебных символов в процессе интерпретации (например, использование меток в ГРИС-программе).

Дальнейшее расширение происходит за счет введения сокращений для представления громоздких синтаксических конструкций (термов), возможно, с переменными, которые входят в другие термы на правах новой операции. Таким образом осуществляется расширение базовой сигнатуры символами производных операций. В программировании это соответствует макроопределениям. Синтаксический терм, содержащий символы производных операций, понимается как сокращенная запись терма, который может быть получен из исходного подстановкой вместо вхождений символов производных операций соответствующих термов. При этом вхождения символов переменных в макроопределениях заменяются на аргументы производных операций. Например, в системе APT может быть представлено макроопределение технологической операции сверления в заданной точке TX:

DRILL = MACRO/TX	сверление в заданной точке TX;
GOTO/TX	выход инструмента в точку TX;
GODLTA 0.0, 0.0, −5.0	сверление на толщину детали
GODLTA 0.0, 0.0, +5.0	$tl = 5.0$;
TERMAC	конец макроопределения.

Базовую синтаксическую систему, сигнатура которой расширена переменными, метками (неинтерпретируемые символы) и средствами определения производных операций (макроопределения), будем называть проблемно ориентированным языком низкого уровня, или ассемблерным языком.

Выразительные возможности ассемблерного уровня для представления проблемной области бедны. Например, в проблемной области, содержащей точки, отрезки и ломаные, относительно операции добавления звена ломаной нельзя представить понятие n-звенника для произвольного n, хотя для каждого фиксированного $n = 1, 2, 3, \ldots$ существует параметрическое задание 1-звенника, 2-звенника, 3-звенника и т. д. Действительно, класс 3-звенников, например, может быть выражен при помощи производной операции

$$3\text{-звен}(p, q, r) = \langle\rangle * p * q * r = \langle p, q, r\rangle$$

в синтаксической системе $\langle P, L;\ l * p\rangle$, где P — множество точек; L — множество последовательностей точек (ломаные); $l * p$ — операция добавления точки p к последовательности l; $\langle\ \rangle$ — константа, соответствующая пустой последовательности. Расширив базовую синтаксическую систему традиционными программными инструкциями последовательного выпол-

нения, ветвления и цикла, можно выразить произвольный n-звенник, имеющий регулярную структуру, например состоящий из точек целочисленной решетки, обход которых начинается в заданной точке и подчиняется простому закону «вверх и влево». Если этих средств окажется недостаточно для выражения некоторых сущностей проблемной области, например кривых Гильберта [15], можно произвести дальнейшее расширение функциональными (рекурсивными) определениями. Таким образом может быть получен спектр средств расширения, в рамках которого представимы конструктивные понятия проблемной области.

Помимо вычислительного аспекта (конструирование объектов), большой интерес представляет также логический аспект (формирование запросов об их свойствах). С этой целью в расширенную синтаксическую систему можно встроить логические средства путем использования классического языка первого порядка и соответствующей техники поиска логического вывода, например, метода резолюции [16] или машинно-ориентированного исчисления генценовского типа [17].

Разработка языковых средств осуществляется снизу вверх от базовой синтаксической системы к расширенным синтаксическим системам. Цель данной работы — достижение полноты представления объектов, преобразований и отношений проблемной области и удобства специалистов, не являющихся квалифицированными программистами. После того как уровень расширения базовой синтаксической системы зафиксирован, осуществляется реализация языковых средств сверху вниз до ассемблерного уровня, который поддерживается средствами базового программного обеспечения.

Таким образом, разработка систем автоматизации в проблемной области должна включать следующие основные этапы: анализ проблемной области (вырабатывается содержательная постановка задачи); формализация проблемной области (вырабатывается базовая синтаксическая система); построение иерархии расширений базовой синтаксической системы (достижение полноты и удобства выразительных средств); реализация расширенной системы средствами базового программного обеспечения ЭВМ.

Языковые средства представления знаний о проблемной области строятся по принципу расширяемой синтаксической системы, настраиваемой на базовое программное обеспечение проблемной области. Сигнатура базовой алгебраической системы является параметром языка. В качестве символических обозначений констант, базовых операций и отношений используется словарь проблемной области. Конструкции расширенной системы представляют собой, по существу, параметризованные схемы задания вычислений в базовой алгебраической системе и при фиксированных значениях параметров порождают основные термы.

Базовые синтаксические конструкции, использующие словарь проблемной области, фиксируются в виде проблемно ориентированного языка низкого уровня. Конструкции расширенной системы представляют проблемно ориентированный язык высокого уровня. Трансляция из языка высокого в язык низкого уровня осуществляется языковым процессором (транслятором), настроенным на словарь проблемной области. Таким образом, процесс трансляции в значительной степени инвариантен относительно проблемных областей. Настройка языка низкого уровня на базовое программное обеспечение сводится каждый раз к разработке постпроцессора, который интерпретирует базовые конструкции синтаксической системы на заданном наборе средств программного обеспечения.

В основе базовой синтаксической системы лежит алгебра символических выражений языка Лисп [18], которая получается следующим образом. Фиксируется множество атомарных объектов: числа, строки, логические значения и символы словаря проблемной области. Атом — символическое выражение (S-выражение). Если A и B суть символические выражения, то пара $(A.B)$ является символическим выражением. В множестве всех символических выражений выделяется класс списков, каждый эле-

мент которого имеет вид

$$(a1.(a2.(\ldots.(an.\text{nil})\ldots))),$$

где nil — константа «пустой список». Для списков принято сокращенное обозначение $\langle a1, a2, \ldots, an\rangle$, пустой список обозначают еще $\langle\rangle$. Заметим, что каждый элемент списка, в свою очередь, может являться списком. В словаре проблемной области выделяются множества имен типов объектов E, имен характеристик (атрибутов) объектов H и специальных символов G. Вводится тип e-объектов, который образует все S-выражения вида

$$(e.\langle(h1.d1), \ldots,(hm.dm)\rangle),$$

где $e \in E$; $h1, \ldots, hm \in H$; $d1, \ldots, dm$ — либо атомарные объекты, либо e-объекты, либо последовательности (списки) из атомарных объектов, e-объектов или последовательностей. Тип e-отношений есть множество последовательностей e-объектов. Имеются следующие базовые операции: конструкторы e-объектов вида

$$q = e(d1, \ldots, dm);$$

селекторы характеристик e-объектов

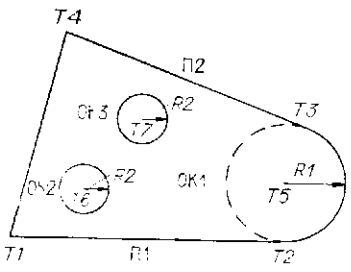
$$q.h1 = d1,\ q.h2 = d2,\ \ldots,\ q.hm = dm$$

(считается, что с каждым именем $e \in E$ связан фиксированный набор имен характеристик $h1, \ldots, hm \in H$); операции над последовательностями: начало(k), хвост(k), $\langle\rangle$, k добавить d; операции базовых типов чисел, строк и логических значений и др. Элементарные условия (предикаты) включают отношения сравнения на базовых типах, предикат пусто(k) — быть пустой последовательностью и др. Для именования конструируемых объектов используются типизированные переменные. Есть средства изменения состояния объектов. Параметризованные схемы базовой алгебры могут задаваться с помощью композиций процедурного, функционального или логического типов. Для этого используется принцип модульности.

Изложенное выше продемонстрируем на примере первой версии системы программирования, основанной на языке Контур [19]. В качестве эксперимента произведена настройка системы программирования Контур на проблемную область вычерчивания контуров плоских машиностроительных деталей [11]. Под контуром понимается последовательность образующих (e-объектов), в качестве которых используются основные геометрические объекты: точка, отрезок, окружность и др.

Рассмотрим процесс настройки системы программирования Контур на проблемную область на примере вычерчивания плоской детали (рисунок). Для описания контура этой детали потребуются следующие геометрические объекты: точка (задана абсциссой и ординатой), отрезок (точками начала и конца), окружность (центром и радиусом) и дуга окружности (точками начала и конца и направлением обхода). В язык Контур введем эти понятия через аппарат образующих:

точка (вещ абсц, ордн); отрезок (точка начало, конец);
окружность (точка центр; вещ радиус);



Эскиз плоской детали:

$T1$, $T4$ — заданные точки; ОК1, ОК2, ОК3 — заданные окружности; $T2$ — точка касания прямой П1, проходящей через точку $T1$ к окружности ОК1; $T3$ — точка касания прямой П2, проходящей через точку $T4$ к окружности ОК1

дуга_окр (окружность окр; точка начало, конец; симв признак), где вещ — обозначение типа вещественные числа, симв — типа символьные строки, а в качестве признака используются символьные строки «по_час» и «против_час». Кроме того, необходимо определить точки $T2$ и $T3$ (см. рисунок), которые получаются как точки касания окружности ОК1 и прямых, проведенных из точек $T1$ и $T4$ соответственно. Так как из заданной точки к окружности можно провести две касательные прямые, то необходимо произвести выбор одной из точек касания по некоторому признаку. В качестве такого признака может выступать информация о том, что точка считается левой или правой при взгляде из исходной точки на заданную окружность. В язык Контур понятие такой точки касания введем как конструктор вида

точка точка_касан (окружность ОК; точка T; симв ПРИЗНАК)., где точка — тип объекта, получаемого в результате выполнения данного конструктора; точка_касан — имя конструктора, а в скобках перечислены параметры конструктора: заданная окружность ОК; точка T, из которой проводятся к этой окружности касательные прямые, и признак выбора. В качестве значений признака выступают символьные строки — «правая» и «левая». Отметим также, что данный конструктор не имеет выполнимой части и будет интерпретирован постпроцессором. В терминах введенных образующих и конструкторов опишем геометрию контура детали в виде следующей схемы:

деталь (точка $T1$, $T2$; окружность ОК1, ОК2, ОК3)
это ‹ОК2, ОК3, отрезок ($T1$, Т2),
дуга_окр (ОК1, $T2$, $T3$, против_час),
отрезок ($T3$, $T4$), отрезок ($T4$, $T1$)›,
где $T2$ = точка_касан (ОК1, $T1$, правая);
$T3$ = точка_касан (ОК1, $T4$, левая).

Результатом интерпретации такой схемы при заданных значениях параметров в системе программирования Контур будет объект типа контур.

Для того чтобы по описанию некоторого объекта задать вычерчивание его чертежа, введем в язык Контур образующие для формирования чертежа:

размер_поля (вещ длина, ширина),
характеристики (размер_поля размеры; вещ масштаб; точка база);
чертеж (характеристики характ; контур изображение).

После того как сформирован словарь терминов, задается отображение этих терминов в виде кодирующей таблицы на среду ПНП, например ПГП ГРАФОР [20]. На основании полученной таблицы генерируется постпроцессор. Полученная в результате система программирования была названа КОНТУР/ГРАФОР [10]. Запрос на выполнение чертежа детали в этой системе может иметь следующий вид:

чертеж (характеристики (размер_поля (18.0, 18.0), 1.0,
точка (4.0, 6.0)), деталь ($T1$, $T4$, ОК1, ОК2, ОК3)),
где $T1$ = точка (0.0, 0.0); $T2$ = точка (2.0, 6.0);
ОК1 = окружность (точка (7.0, 0.0), 3.0);
ОК2 = окружность (точка (1.0, 1.0), 0.5);
ОК3 = окружность (точка (3.0, 2,0), 0,5).

Сформировать такой запрос в системе автоматизированного проектирования можно в диалоговом режиме.

СПИСОК ЛИТЕРАТУРЫ

1. **Краузе Ф. Л., Вассилакопулос В.** Подход к созданию автоматизированных систем проектирования и производства // Системы автоматизированного проектирования.— М.: Наука, 1985.
2. **Редько В. Н.** Композиции программ и композиционное программирование // Программирование.— 1978.— № 5.

3. **Горшков П. В.** Рациональные структуры данных и их приложения // Кибернетика.— 1989.— № 6.
4. **Горшков П. В., Лялецкий А. В., Фролов В. Д., Юрчишин В. В.** Моделирование роботов-манипуляторов с программным управлением // УСиМ.— 1989.— № 1.
5. **Коваль Л. В., Родимин С. П.** Языковые средства представления геометрии деталей в системах автоматизированного проектирования (обзор).—Киев, 1987.— Деп. в УкрНИИНТИ 12.09.87, № 2459-Ук87.
6. **Родимин С. П., Коваль Л. В.** Применение метода формализованных технических заданий для разработки языковых средств комплексных САПР // Тез. докл. респ. науч.-техн. конф. «Методологические проблемы автоматизированного проектирования и исследования систем».— Севастополь: СПИ, 1987.
7. **Горшков П. В., Коваль Л. В., Лялецкий А. В., Родимин С. П.** Инструментальные средства эффективного представления, ввода и хранения геометрической информации // Тез. докл. всесоюз. конф. «Методы и средства обработки сложной графической информации».— Горький: ГГУ, 1988.
8. **Родимин С. П., Коваль Л. В.** Язык КОНТУР — инструментальное средство повышения интеллектуального уровня интерфейса с ППП // Тез. докл. шк.-семинара «Методы автоматизированного проектирования электронно-вычислительной аппаратуры и СБИС (САПР СБИС-88)».— Киев: О-во «Знание» УССР, 1989.
9. **Горшков П. В., Лялецкий А. В., Родимин С. П., Коваль Л. В.** Некоторые принципы эффективного представления, хранения и обработки графической информации // Тез. докл. V Всесоюз. конф. по машинной графике «Машинная графика-89».— Новосибирск: НФ ИТМиВТ, 1989.
10. **Родимин С. П., Коваль Л. В.** Комплексная система КОНТУР/ГРАФОР // Там же.
11. **Коваль Л. В.** Опыт использования системы КОНТУР/ГРАФОР в прикладных областях // Там же.
12. **Мальцев А. И.** Алгебраические системы.— М.: Наука, 1970.
13. **Грувер М., Зиммерс Э.** САПР и автоматизация производства.— М.: Мир, 1987.
14. **Система** автоматизации проектирования обустройства нефтяных месторождений. Система программ машинной графики ГРИС-СМ: Описание применения (руководство системного программиста, руководство пользователя-оператора).— Тюмень: Гипротюменнефтегаз, 1988.
15. **Вирт Н.** Алгоритмы и структуры данных.— М.: Мир, 1989.
16. **Чень Ч., Ли Р.** Математическая логика и автоматизированное доказательство теорем.— М.: Наука, 1983.
17. **Дегтярев А. И., Лялецкий А. В.** Логический вывод в системе автоматизации доказательств // Математические основы систем искусственного интеллекта.— Киев: ИК АН УССР 1984.
18. **McCarthy J., Abrahams P. W., Edwards D. J. e. a.** Lisp 1.5 programmer's manual.— Cambridge, Mass.: MIT Press, 1963.
19. **Коваль Л. В., Родимин С. П.** Язык высокого уровня КОНТУР // Тез. докл. всесоюз. конф. «Методы и средства обработки сложной графической информации».— Горький: ГГУ, 1988.— Ч. 1.
20. **Баяковский Ю. М., Галактионов В. А., Михайлова Т. П.** Графор. Графическое расширение Фортрана.— М.: Наука, 1985.

Поступила в редакцию 16 января 1990 г.