О разработке инструментальных систем, ориентированных на решение информационно-логических задач

Р. Г. Бухараев, А. И. Еникеев, И. И. Макаров

Практика использования вычислительных машин для автоматизации исследовательских и проектных работ выдвинула на первое место среди множества классов решаемых задач задачи информационно-логического типа. Основные из них — это формульные преобразования, машинные эксперименты по моделированию различных процессов, информационный поиск и т.п. Сложность и многообразие структур данных, а также программных средств их обработки, характерные для упомянутого класса задач, обусловили необходимость эффективного управления процессами использования и создания соответствующих пакетов прикладных программ. Средства, имеющиеся в существующих операционных системах, не обладают достаточной гибкостью и эффективностью для генерации программ в пакетах. Использование только традиционных языков программирования в качестве входных для пакетов программ также не обеспечивает упомянутые выше возможности из-за отсутствия в них средств, позволяющих осуществлять качественный диалог, динамическое планирование решения задач, эффективное структурирование программ и адекватность представления моделей предметных областей конструкциями этих языков.

Все это привело к необходимости создания концепции инструментальных систем программирования, позвольяющих эффективное создание пакетов прикладных программ и систем программирования с проблемной ориентацией. Эта концепция, наиболее полно рассмотренная в работе [1], получила свое развитие в разработках интегрированных систем программирования, представляющих разновидность инструментальных систем с совместимостью видов проблемноориентированных данных в разных пакетах [8], расширяющейся системы автоматизации проектирования вычислительных машин [2] и диалоговых систем для решения информационно-логических задач [4].

В докладе предлагается один из подходов к разработке такого рода систем с диалоговыми возможностями, обеспечивающих: 1) эффективное создание проблемно-ориентированных языков для пакетов прикладных программ; 2) эффективную генерацию новых программ пакета на основании имеющихся; 3) динамическое планирование решения задач на вычислительной машине

в режиме диалога. Этот подход был использован при создании системы МАТИСС [6]. Особенность последней состоит в едином концептуальном подходе к разработке системы, основанном на оптимальном сочетании таких требований, как расширяемость входного языка системы, возможность эффективного и гибкого управления диалоговыми средствами, наличие средств структурирования и обработки данных (в том числе и программ), обеспечивающих, в частности, возможности динамического преобразования программ в режиме диалога, выполнения смешанных вычислений и формульных преобразований. Одним из основных принципов построения системы явилось создание базисного языка и средств расширения, позволяющих последовательно надстраивать над базисным языком множество проблемно-ориентированных подсистем для решения информационно-логических задач.

Базисный язык является входным языком первой очереди реализации системы и позволяет:

— описание синтаксиса и семантики вводимых элементов расширения;

— непосредственное составление программ для решения как вычислительных, так и информационно-логических задач, связанных с обработкой сложных структур данных;

— в режиме диалога динамическое изменение структуры программы на стадии ее выполнения с целью гибкой адаптации алгоритма к условиям, возникающим в процессе решения задачи.

Для обеспечения перечисленных возможностей в базисный язык включены средства:

а) выполнения арифметических, логических и текстовых операций;
в) обработки структур данных и программ;
с) расширения каждого уровня языка;
д) диалогового взаимодействия;
е) формульных преобразований.

К особенностям языка можно отнести следующие:

— допускается возможность смешанного (частичного) вычисления выражений, в которых не для всех переменных к моменту вычислений определены значения;

— допускается возможность динамического описания переменных; в этом случае тип переменной однозначно определяется в текущий момент работы программы;

— в операторах формульных преобразований допускается использование условных соотношений;

— язык позволяет реализацию рекурсивных обращений (реализация рекурсии выполнена аналогично ЛИСП-системам [3]);

— диалоговые средства языка обеспечивают планирование точек выхода на диалог по заданной в программе системе условий;

— синтаксис языка описывается $LL[K]$ грамматикой [5], позволяющей иметь однопроходной синтаксический анализатор.

Последняя особенность языка вызвана необходимостью повышения реактивности диалога, в котором динамическое изменение структуры программы требует компиляции фрагментов изменения.

Средства расширения системы позволяют расширять входной язык путем введения в него новых операций (функций), отношений и операторов. Расширение сводится к введению пары (s, g), где s — синтаксическое, а g — семантическое описание элемента расширения входного языка. Введенная пара запоминается в таблице соответствий между синтаксическими и семантическими описаниями, которая служит исходной информацией для настройки компилятора на соответствующий уровень расширения входного языка системы. Семантические описания, задаваемые в виде исходных программ перед записью в таблицу предварительно компилируются. Для составления семантических программ можно использовать базисный язык, любой уровень расширения входного языка, а также языки, входящие в стандартное математическое обеспечение.

Одним из основных принципов расширения языка является принцип модульности, позволяющий выделение отдельных уровней расширения системы в автономные подсистемы, работающие независимо от всей системы. При реализации средств расширения был использован опыт разработки системы ПРОЕКТ [2].

Важным моментом реализации диалоговой системы является соотношение компиляции и интерпретации, поскольку чистая интерпретация увеличивает время выполнения программ, а обычная компиляция часто бывает неприспособленной к диалогу.

Это обусловило выбор способа компиляции, предусматривающего получение объектной программы в специальном структурированном виде, удобном для интерпретации и динамического изменения программы в режиме диалога. В отличие от реализации ЛИСП-подобных систем [3] здесь предлагается способ внутреннего представления программы, основанный на обратной польской записи выражений, сокращающий время интерпретации за счёт ликвидации лишних проходов по структуре программы и позволяющий эффективную реализацию режимов коллективного доступа.

Программы, реализующие функции и операторы базисного языка, представляются в виде машинных модулей. Функциям и операторам расширенного языка, семантические программы которых были составлены на входном языке системы, соответствуют структурированные представления программ.

Подобное представление программ достаточно удобно реализуется интерпретатором с использованием стэка. В процессе прохода по структуре программы интерпретатор заносит в стэк данные и выполняет встречающиеся модули. Входные параметры для своей работы каждый такой модуль соответственно выбирает из стэка. Результат работы модуля также заносится в стэк.

Такой подход к внутреннему представлению и интерпретации программ естественным образом позволяет применить методику разработки систем коллективного доступа, предложенную в работе [7] для реализации режима разделения времени, основанного на принципе квантования по модулям соответствующих представлений программ. В отличие от традиционного квантования по времени, упомянутая выше реализация достигается за счёт разрешения прерываний для перехода на обслуживание другого процесса только после полного завершения работы очередного модуля и позволяет: а) упрос-

тить структуру управления процессами; б) минимизировать количество запоминаемой при прерывании информации.

Однако такой принцип реализации режима разделения времени является целесообразным, если продолжительность работы соответствующих модулей сравнительно невелика и может быть оценена на стадии их разработки. Поэтому на разработку системных программ были наложены следующие требования:

1) программная конструкция вида α_1; CALL, β; α_2, где α_1 начальный участок программы до оператора вызова подпрограммы β, а α_2 конечный участок, должна быть заменена цепочкой программ α_1, β, α_2, выполняемых в перечисленной последовательности (если время работы программы β сравнительно невелико);

2) ехли время работы какой-либо из программ увеличивается в зависимости от величины обрабатываемых данных, то конструкция такой программы должна обеспечивать выход после обработки определенной порции информации на управляющую программу, которая после перехода на обслуживание данного процесса может запустить ту же самую программу для обработки следуюшей порции данных.

Следует отметить, что указанный выше подход к интерпретации программ позволил также естественным образом реализовать возможность динамического планирования точек перехода в программах в зависимости от задаваемой совокупности условий (аналог планирования программных прерываний в операционной системе). Последняя возможность обеспечивает: а) эффективное тестирование программ; б) планирование точек выхода на диалог.

Дополнительно к рассмотренному выше представлению программ в систему включены средства получения и обработки полной структурированной формы представления выражений, основанной на прямой польской записи (аналогично представлению выражений в ЛИСПе). Последняя форма представления является эффективной в формульных преобразованиях и смешанных вычислениях выражений.

В состав программного обеспечения системы входят: 1) управляющая программа, обеспечивающая режимы разделения времени и мультипрограммирования; 2) макропроцессор, обеспечивающий процессы расширения и компиляции входного языка; 3) интерпретатор, реализующий программы в специальном внутреннем представлении и управляющий диалоговыми средствами; 4) обрабатывающие программы, непосредственно реализующие операторы и функции входного языка; 5) служебные программы, устанавливающие связь с операционной системой.

Первая очередь системы реализована в рамках дисковой операционной системы ДОС ЕС.

КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
КАФЕДРА ТЕОРЕТИЧЕСКОЙ КИБЕРНЕТИКИ
УЛ. ЛЕНИНА 18
420008 КАЗАНЬ, СССР

Литература

[1] Тамм, Б. Г., Э. Х. Тыугу, О создании проблемно-ориентированного программного обеспечения, *Кибернетика*, № 4, К., 1975.
[2] Глушков, В. М., Ю. В. Капитонова, А. А. Летичевский, Автоматизация проектирования вычислительных машин, *Наукова думка*, Киев, 1975.
[3] Лавров, С. С., Г. С. Силагадзе, Автоматическая обработка данных, Язык ЛИСП и его реализация, *Наука*, Москва, 1978.
[4] Брябрин, В. М. Г. В. Сенин, Руководство к системе ЛИЛОС-БЭСМ—6, ВЦ АН СССР, Москва, 1977.
[5] Ахо, А. Дж. Ульман, Теория синтаксического анализа, перевода и компиляции, Т. I., *Мир*, Москва, 1978.
[6] Бухараев, Р. Г., А. И. Еникеев, И. И. Макаров, Вопросы эффективного управления пакетами прикладных программ в режиме диалога, *Труды Советско-финского симпозиума по интерактивным системам*, Тбилиси, 1979.
[7] Замов, Н. К., Н. И. Звягина, Р. К. Самитов, Автоматизированная обучающая система Гамма, Тезисы докл, 1 Всесоюзной конференции *Человеко-машинное обучение системы*, (г. Телави), Москва, 1979.
[8] Lopez, L. A., POLO-problem oriented language organizer, *Comput. & Structures*, v. 2, 1972.

(Поступило 3-ого декабря 1981 г.)