УДК 681.3

О.Н. Булавенко, В.Н. Коваль
Институт кибернетики им. В.М. Глушкова НАН Украины, г. Киев

Интегрированный интеллектуальный интерфейс системы ИРМ

В статье рассматриваются проблемы создания интеллектуального интерфейса, интегрированного в произвольную платформу и предназначенного для усовершенствования взаимодействия пользователя с распределенной многопроцессорной вычислительной средой, содержащей сложные структуры данных. Достоинства разработки – гибкость, возможность оперативного создания баз знаний при взаимодействии на естественных языках, удобная система навигации.

Введение

Интенсивное развитие информационных технологий, в том числе многопроцессорных вычислительных систем с возможностью распределенной обработки информации, выдвигает все новые и сложные требования к программированию задач, ориентированных на такую среду. Поэтому необходимы новые подходы к организации интерфейсов между пользователем и вычислительной средой, облегчающие постановку, отладку задач и эксплуатацию вычислительных систем. В этом направлении уже давно трудятся многие разработчики программного обеспечения, в результате чего созданы такие мощные интерфейсы, как операционные системы Windows XP, Linux, OS/2 и др., а также системы параллельного программирования MPI [1], DVM [2], mpC [3] и др., намного облегчающие программирование и отладку программ, оптимальное распределение и эксплуатацию вычислительных ресурсов для задач преимущественно вычислительного характера.

Решение проблемы подготовки и отладки задач в некоторой степени берут на себя компиляторы и интерпретаторы. Однако основная нагрузка по программированию, отладке и постановке задач для решения в распределенных системах остается за пользователем. Актуальным здесь является вопрос автоматизации такой работы. Она может развиваться в нескольких направлениях. Одним из таких направлений является создание библиотеки расширения и средств взаимодействия с ней, упрощающих программирование задач для распределенных многопроцессорных вычислительных систем [4].

Общение пользователя с аппаратно-программной средой в существующих интерфейсных системах осуществляется на технических языках (общее название всех форм алгоритмических языков – пакетных, диалоговых, декларативных и др.). Актуальным для интеллектуальных систем является введение возможности общения пользователей на национальных (или ограниченных искусственных) языках. Проблема, которая встает на этом пути, – семантическая адекватность понимания текстов и речи. В большинстве систем эта проблема решается путем применения определенного набора директив или фраз с жестким ограничением на их расширение.

В других системах вводится единый цифровой код семантики информации [5] для распознавания смысловой информации. В интегрированном интеллектуальном интерфейсе (ИИ) системы ИРМ [6-8] разработан метод распознавания смысла текста через семантическую базу знаний.

Одной из интеллектуальных задач, решаемых средствами интеллектуального интерфейса системы ИРМ, является задача построения базы знаний (БЗ) в виде семантической сети по информации, содержащейся во вводимом тексте на естественном языке, далее – модификация БЗ, а также взаимодействие с ней в процессе эксплуатации в виде вопросов и ответов с целью уточнения и конкретизации. Решение этой задачи разбивается на два этапа: анализ текста и генерация ответов. Исследования в области машинного анализа текстов на естественном языке показали, что с точки зрения автоматической обработки в тексте всегда можно выделить семантические единицы и связи между ними, которые можно представить в виде семантической сети, отражающей языковую модель вводимого текста. В свою очередь, семантическая сеть может являться разновидностью более общей структуры данных – направленных графов, состоящих из узлов и направленных дуг. Узлами здесь являются понятия – собственно семантические единицы, а дугами – связи между ними, в роли которых могут выступать также семантические единицы, выражающие отношения. При вводе текста основными структурами данных являются слова, предложения, фрагменты текста и определенный контекст их обработки, который определяется структурой словаря, отражающего профиль знаний, например медицинский, технический, экономический и т.д. Поиск и выборка знаний из семантической сети может включать поиск по всей сети. Если искомый факт не сохранен явно, то, вероятно, придется вывести этот факт из другой сохраненной информации. Программы для поиска знаний часто тратят большую часть своего времени на повторение нескольких базисных операций типа логических операций на наборах, сортировку, сопоставление с образцом. Ускорение получения результата может быть достигнуто одновременной инициализацией этих операций для многих узлов сети. В идеале каждое понятие (узел) должно быть распределено отдельному процессору. Для этого требуется большое количество процессорных элементов с программируемыми логическими соединениями – такими, чтобы топология могла быть реконфигурирована для решения проблемы. В ИИ системы ИРМ эта проблема решается библиотечными методами расширения С+Граф и ОС Windows/NT/XP для проведения распределенной параллельной обработки группы узлов графа: (1) на каждом процессоре в режиме одновременного исполнения множества потоков, каждый из которых поддерживает работу одного узла, и (2) для параллельного исполнения групп узлов на массиве используемых процессоров.

Структура интерфейса

Интерфейс («Intelligent Solver of Problem» (ISP) – «интеллектуальный решатель задач» (ИРЗ)) отражает модель поведения «человек – машина» на современном уровне ее развития и предназначен для совершенствования взаимодействия пользователя с вычислительной системой (ВС) при решении прежде всего интеллектуальных задач, а также задач других классов. ИРЗ выполнен в виде пакета библиотечных программ по распределенной обработке информации и является интегрированной системой в том смысле, что объединяет средства и методы взаи-

модействия интерфейса с классами и методами языка С+Граф и стандартной операционной системой типа WindowsNT/XP, реализованными на одном или нескольких процессорах. Для работы с графами ИРЗ предоставляет средства диалогового многооконного интерфейса (рис. 1) в виде директив – запросов и ответов. И библиотеки, и сама многооконная среда запрограммированы на языке Java и его расширении С+Граф [9], что позволяет применять их на всех использующихся платформах благодаря свойствам технологии языка Java [10].

Каждый класс задач использует специализированный интерфейс. ИРЗ работает в среде ОС семейства Windows, что позволяет унифицировать внешний вид приложения и способ запуска и отладки задач.

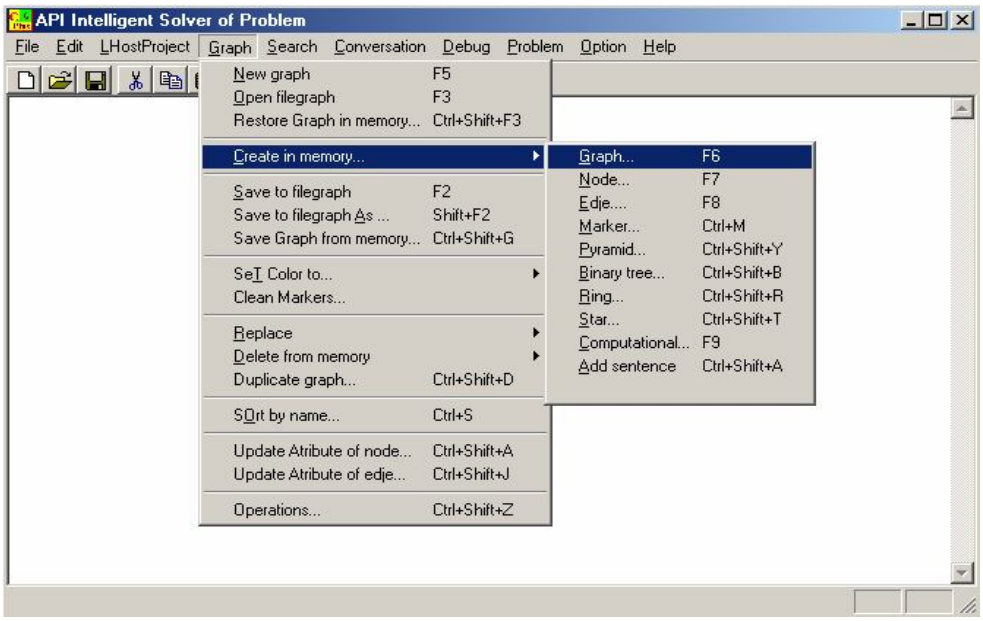

Рисунок 1 – Общий вид оконного интерфейса ИРЗ:
на фоне главного окна, предназначенного для ввода исходного текста
(в том числе голосом через микрофон), открыта панель работы с графами – их
создание, модификация, сохранение, уничтожение и др.

Построение базы знаний

База знаний представляет собой набор семантических сетей (СС), создаваемых на базе графовых структур. В ИРЗ БЗ может строиться двумя способами: (1) с помощью форматированного ввода конкретной информации об узлах и дугах сети, (2) посредством ввода текста на естественном языке в виде предложений.

Общая схема правил кодирования при форматированном вводе может быть представлена следующим образом:

имя-узла[(функция-узла|значение)]{,[<имя-дуги/значение|nil>]|-имя-узла[(функция-узла|значение)]};

Скобки [ ] означают необязательное вхождение элемента; скобки { } означают возможность повторения элемента кодирования через запятую; вертикальная черта | означает альтернативу.

Например, база знаний «Птицы и другое», описанная ниже и введенная в главное окно под названием Bird (рис. 1),

///*** База данных «Птицы и другое» ***///
Bird-center;
Aeroplane<have>wing;
Aeroplane<ability>flay;
pilot<operate_to>Aeroplane;
Boeing747<is>Aeroplane;
Aeroplane<have_a>motor;
motor<use>benzine;
Bird<ability_to>flay;
Bird<have_the>wing;
Falcon<is_a>Bird;
Bird<consist>empennage;
Falcon<include>bill;
Ostrich<is_one_of>Bird;
Ostrich<cannot_to>flay;//Страус не умеет летать
Ostrich<have_high>growth;//Страус имеет высокий рост
Ostrich<have_long>foots;//Страус имеет длинные ноги
foots<is_very>long;
Bird<is_ordinal>Animal;
Animal<can_to>eat;//Животное может есть
Animal<can_also>breathe;//Животное может также дышать
Animal<can_also_to>move;//Животное может также двигаться
Animal<have_also>skin;//Животное имеет также кожу

представляется в графическом окне в следующем виде (рис. 2).

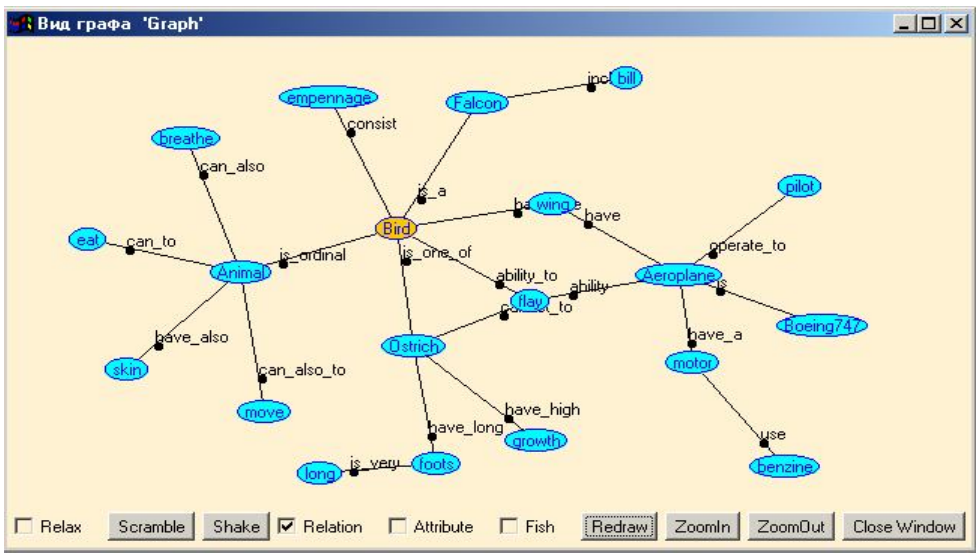

Рисунок 2 – Графическое окно ИРЗ с изображением введенной СС: узлы графа отображаются в виде овалов с вписанными именами, а дуги – в виде линий с точками, обозначающими направление отношений между узлами

Как видно из примера, пользователь вводит имена узлов и дуг с возможным указанием их атрибутов (функция или значение), а также указывает направление следования дуг.

При другом способе ввода информации БЗ представляется в виде совокупности предложений, вводимых в главное окно с клавиатуры или голосом через микрофон. В последнем случае должна одновременно функционировать система распознавания голоса (например, Dragon Naturally Speaking Dictate-5.0/6.0 [11]). Общая схема последовательности обработки представлена на рис. 3.

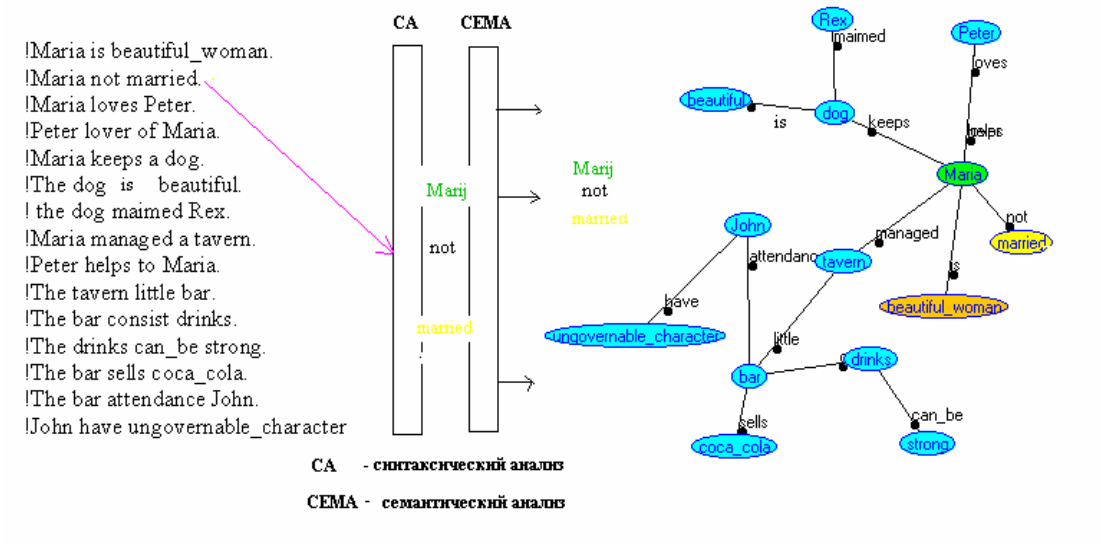

Рисунок 3 – Последовательность действий для отображения текста в СС: при построении графа вводимый текст проходит синтаксический и семантический анализ, по результатам которого строится СС

На этапе синтаксического анализа (СА) английского предложения используется морфологический англо-русский словарь (до 100 тыс. слов), формируемый в памяти машины в виде хэш-таблицы, содержащий словоформы с атрибутами-предикатами: *n* – существительное, *v* – глагол, *pron* – личное местоимение, *a* – прилагательное, *adv* – наречие, *prep* – предлог, *art* – артикль, past – прошедшее время глагола, *relat* – относительное местоимение, *cj* – союз и др. Фрагмент словаря представлен ниже:

…
he / pron > он
him / pron > ему
be / v > быть
boy / n > мальчик
beautiful / a > красивый
very / adv > очень
little / a > маленький
with / prep > с
the / art
…

Блок СА выделяет слова и по словарю определяет тип предиката. Для слов, производных от словарных форм (герундий, глаголы в разных временах, существи-

тельные во множественном числе), сначала осуществляется их опознание во вспомогательных таблицах и шаблонах и представление в словарной форме, а затем их поиск в словаре для сравнения и установления типа предиката. На основе синтаксических шаблонов, определенных в грамматике (в данном случае английской), и с использованием морфолексических характеристик слов из словаря формируются синтаксические структуры – группы подлежащего, группы дополнения, обстоятельства или глагольные группы. Взаимосвязь между синтаксическими группами уточняется на этапе семантического анализа предложения (СЕМА) по положению слов в предложении (основываясь на шаблонах твердого порядка слов и шаблонах сложноподчиненных предложений для английского языка). Этапы СА и СЕМА перекрываются. Отношения между семантическими объектами в СС отражают отношения между членами предложения. Обычно в предложении отношение выражается предлогами или глаголами (или глагольными группами) в разных формах, а семантические объекты – подлежащими или группами подлежащих, дополнениями и т.п. В качестве примера рассмотрим анализ предложения

This little boy was a very beateful child.

Здесь, согласно шаблонам, можно выделить группу подлежащего «*This little boy*», группу дополнения «*a very beautiful child*» и глагол «*was*». Как видно из рис. 4, группа подлежащего формируется узлами *This, little* и *boy* и объединяющими их дугами связи is_an1, is_an2, а группа дополнения – узлами *a*, *very, beautiful* и *child* и объединяющими их дугами связи is_ad2, is_ad3, is_ad4. Направление отношения между этими группами образуется дугой *was*. Маркер узла и визуально цвет узла отражают тип предиката. Таким образом, построенный граф содержит информацию о полной семантической структуре предложения. Благодаря этому можно восстановить (вывести) информацию из БЗ путем вопросов и ответов или соответствующим сканированием СС, используя соответственно панель «Conversation» или «Debug».

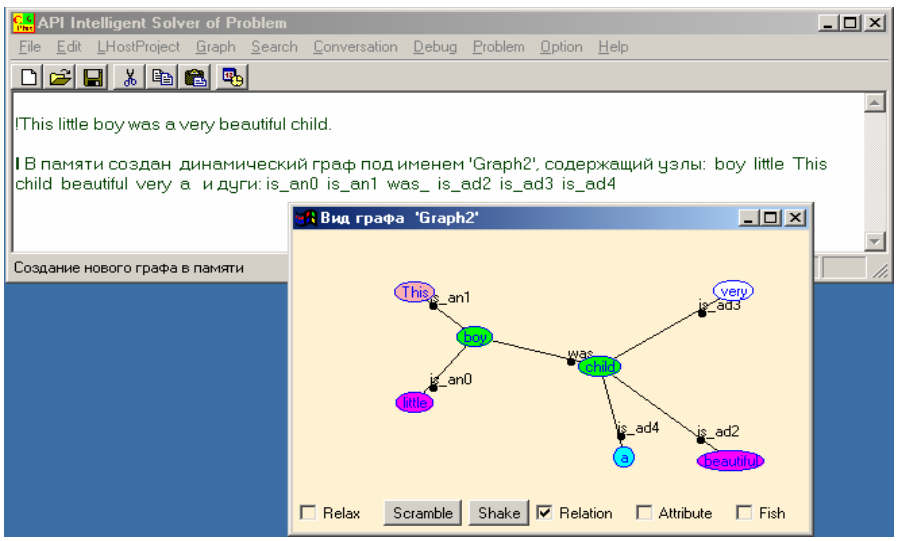

Рисунок 4 – Граф, отражающий семантическую структуру предложения *This little boy was a very beautiful child:* на рисунке видно, как связываются узлы, образующие группы подлежащего и дополнения; в свою очередь, эти группы связываются отношением, представляющим глагол

В процессе ввода и анализа предложений новые семантические объекты «взаимодействуют» с уже находящимися в БЗ. Эти взаимодействия отражаются либо в виде непосредственных связей, либо через понятия (концепты). Например, если вслед введенному выше предложению последует предложение «*He was my brother*», семантический анализатор сошлется на предыдущее предложение, а именно – на базовое существительное в группе подлежащего, т.е. на *boy* (рис. 5).

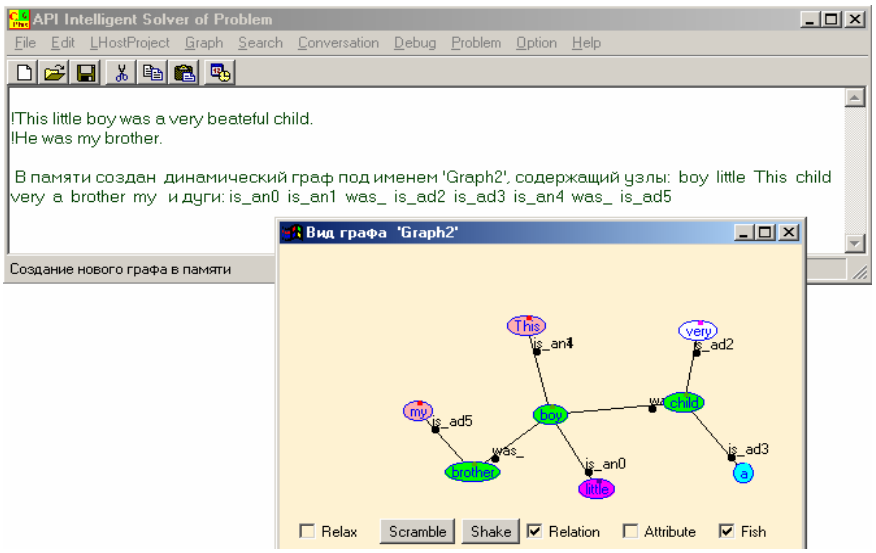

Рисунок 5 – Замена местоимения на введенное ранее в БЗ существительное *boy* в группе подлежащего *This little boy*: из рисунка видно как с базовым существительным *boy* связываются лексемы из другого предложения, образуя единый граф СС

Взаимодействие с базой знаний

Взаимодействие с БЗ производится путем так называемого распространения маркеров – операции поиска и сравнения с отметкой найденных узлов или дуг, удовлетворяющих указанному критерию поиска. Для ускорения операции поиска узлам или дугам созданного (уже имеющего имена узлов и дуг) графа присваивается имя маркера, которое хранится в хэш-таблице в виде соответствия «имя маркера узла – имя узла» или «имя маркера дуги – имя дуги». Маркеры могут быть расцвечены, что облегчает визуальную навигацию. Операция распространения маркеров выполняется последовательно-параллельно средствами библиотеки С+Граф (табл. 1).

Таблица 1 – Основные методы и выполняемые ими функции библиотеки С+Граф

Название метода и аргументы	Функция
addNode(Graph g, String name)	Добавляет объект типа graph в объект типа Node (узел)
clearGraph(Graph g, String nameG)	Удаляет все элементы (узлы и дуги) графа nameG.
createGraph(Graph g, String nameG)	Создается новый пустой граф g типа Graph
deleteEdge(Graph g, String nameG, String name)	Удаляет из графа g дугу с именем name

Продолжение табл. 1

deleteNode(Graph g, String del)	Удаляет из графа g узел с именем del
EmptyGraph(Graph g)	Удаляет все элементы (узлы и дуги) графа g
findEdge(Graph g, String name)	Поиск дуги с именем name в графе g
findNode(Graph g, String name)	Поиск по направлению дуг узла с именем name в графе g
insertEdge(Graph g, String from, String to, String name)	Вставляет новый объект типа Edje (дуга) в граф g
ShowGraph (Graph g)	Создает графическое отображение узлов и дуг графа g
UpdateNode (Graph g, String name, String fun, boolean fish)	Обновляет атрибуты в графе g узла с именем name
UpdateEdge (Graph g, String name, String fun, double val)	Обновляет атрибуты в графе g дуги с именем name
markCreate(graph g, String marker, String s_rel, String e_node)	Создание маркера для узла или дуги в графе g
MarkSearch (graph g,String node, String marker, int val, boolean nodeOredje)	Искать узел или дугу по указанному имени, передать маркеру значение, отмаркировав его (её)
PropagateInGraph (graph g, int indNode, int indMarker, boolean typeFun)	Распространение маркера по графу согласно правилу маркера и заданной функции

Диалог с БЗ осуществляется через диалоговое окно «Conversation» (рис. 6). Рассмотрим процесс диалога с созданной на рис. 5 БЗ.

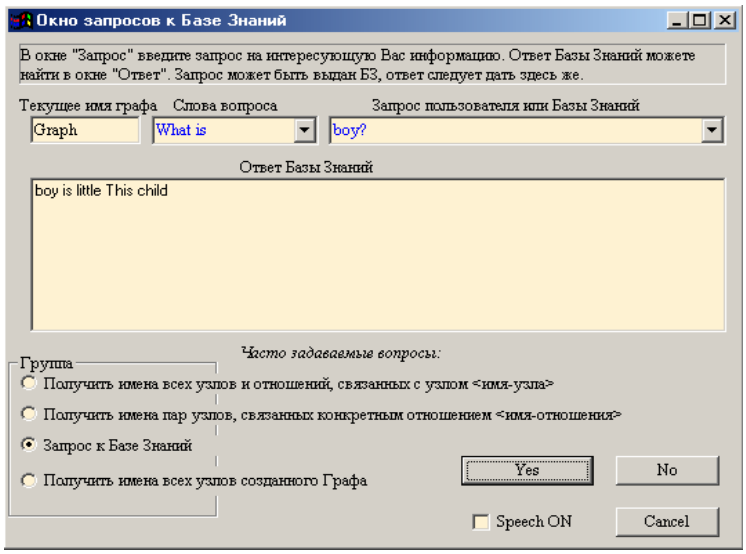

Рисунок 6 – Диалоговое окно взаимодействия с БЗ, созданной в виде СС: выбором опций возможен «ручной» анализ связей узлов в БЗ, либо через опцию «Запрос к Базе Знаний» возможно вывести более понятный для человека ответ

В верхнем поле панели вводится текст вопроса либо с клавиатуры, либо голосом. Ответ выводится в поле ответов. Он может быть озвучен программой «Govorilka» [10] с установкой опции Speech ON. Процесс формирования ответа на-

чинается с поиска в графе БЗ узла с базовым именем, указанным в поле запросов, затем ищутся узлы, ассоциированные с базовым узлом. Текст ответа формируется на основе той структуры, куда входит базовый узел.

Заключение

При разработке интеллектуального интегрированного интерфейса системы ИРМ были реализованы и проверены методы оперативного анализа исходных текстов и методы построения и поиска семантической сети. К моменту написания статьи созданы словари и блоки синтаксического и семантического анализа для ограниченного английского языка, готовятся соответствующие компоненты для русского и украинского языков.

Было показано решение задачи одного из классов задач, решаемых средствами интеллектуального интерфейса. Мы не рассматривали класс вычислительных задач. Постановка, отладка и решение их заслуживают отдельного рассмотрения.

Литература

1. http://www.mpiforum.org
2. http://www.keldish.ru/dvm/
3. http://www.ispras.ru/~mpC
4. Булавенко О.Н., Коваль В.Н, Рабинович З.Л. Параллельное программирование в Интеллектуальных решающих машинах // Друга міжнар. наук.-практ. конф. з програмування «УкрПРОГ'2000». – С. 123-131.
5. Чипашвили Ш.Ш. Некоторые вопросы создания единого кода семантики информаци (Проект «Интерсемантика») // Искусственный интеллект. – 2000. – № 3. – С. 578-583.
6. Пат. №2042193 РФ, Кл. д-06F15/16. Вычислительная система 95/ О.Н. Булавенко, В.Ф. Любарский, В.М. Мушка и др. – Бюл. от 20.08.
7. Пат. № 19875 України, Кл. д-06F15/16. Обчислювальна система / О.Н. Булавенко, В.Н. Коваль, В.Ф. Любарский, В.М. Мушка и др. – Бюл. № 6 от 25.12.1997.
8. Пат. № 56139 Украины. Обчислювальна система / О.Н. Булавенко, В.Н. Коваль, В.Ф. Любарский и др. – Бюл. № 5 від 15.05.2003.
9. Коваль В.Н., Булавенко О.Н., Рабинович З.Л. Развитие знаниеориентированных систем мультипроцессорной обработки информации на основе технологии интеллектуальных решающих машин // Искусственный интеллект. – 2002. – № 3. – С. 242-257.
10. http://java.sun/com/products/jdc1.2/docs/rmi/index.html
11. http://dragonsystems.com
12. Anton Ryazanov, 2001 – 2002, vsoft@vector-ski.ru

In article is considered problems of creation of the intellectual interface integrated in an any platform and intended for improvement of interaction of the user with distributed multiprocessor computing environment, containing complex structure of the data. Advantage of development – flexibility, opportunity of operative creation of bases of knowledge at interaction in natural languages convenient system of navigation.

У статті розглядаються проблеми створення інтелектуального інтерфейсу, інтегрованого до довільної платформи та призначеного для вдосконалення взаємодії користувача з розподіленим багатопроцесорним обчислювальним середовищем, що містить складні структури даних. Цінність розробки – гнучкість, можливість оперативного створення баз знань при взаємодії природними мовами, зручна система навігації.

Статья поступила в редакцию 16.07.03.