УДК 004.891:004.942

С.Ю. МЕЛЕШЕНКО[1], А.Р. ЕМАД[1], М.Н.Х. ХИЛАЛ[2], О.В. ЯРОВАЯ[1]

[1] Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Украина
[2] Национальный технический университет «ХПИ», Украина

ЯЗЫК ПРЕДСТАВЛЕНИЯ ЗНАНИЙ В ЭКСПЕРТНЫХ СИСТЕМАХ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ В СЛОЖНЫХ ПРОЕКТАХ

В работе представлены основные элементы интегрированной экспертной модели в форме языка представления и манипулирования знаниями для задач принятия решений в сложных проектах. Представлена структура языка описания интегрированной экспертной модели в виде определений и соглашений. Проведено моделирование синтаксиса языка интегрированной экспертной модели, на примерах рассмотрены его особенности

экспертная система, интегрированная экспертная модель, язык представления знаний, синтаксис языка, принятие решений, программирование

Введение

Информационные технологии, базирующиеся на современных методах искусственного интеллекта, получили значительное распространение в мире. Их важность, а, в первую очередь важность экспертных систем, состоит в том что, данные технологии существенно расширяют круг практически значимых задач, которые можно решать на компьютерах, и их решение приносят значительный экономический эффект. В то же время, технология экспертных систем является важнейшим средством для решения глобальных проблем традиционного программирования, а именно, уменьшения длительности и, следовательно, высокой стоимости разработки приложений; высокой стоимости сопровождения сложных систем; повторная используемости программ и т.п. Кроме того, объединение технологий экспертных систем с технологией традиционного программирования добавляет новые качества к коммерческим продуктам за счет обеспечения динамической модификации приложений пользователем, а не программистом, большой «прозрачности» приложений, лучших графических средств, пользовательского интерфейса и взаимодействия [1-3].

Формулирование целей статьи

Выделим две основные причины, которые привлекают интерес к экспертным системам [4]:

- экспертные системы ориентированы на решения широкого круга задач принятия решений в слабо формализованных предметных областях, а так же на приложения, которые считаются мало доступными для вычислительной техники;

- экспертные системы при решении практических задач принятия решений достигают, а иногда и превосходят возможности людей экспертов, не оснащенных экспертными системами.

Причины, приведшие системы искусственного интеллекта к коммерческому успеху, следующие:

1. *Специализация.* Переход от разработки инструментальных средств общего назначения к проблемно специализированным средствам, что обеспечивает сокращение сроков разработки приложений, увеличивает эффективность использования инструментария, упрощает и ускоряет работу эксперта, позволяет повторно использовать информационное и программное обеспечение (объекты, классы, правила, процедуры).

2. *Использование языков традиционного программирования.* Переход от систем, основанных на языках искусственного интеллекта (Lisp, Prolog и

т.п.), к языкам традиционного программирования (C, C++ и т.д.) упростил «интегрированность» и снизил требования приложений к быстродействию и ёмкости памяти.

3. *Интегрированность*. Разработаны инструментальные средства искусственного интеллекта, легко интегрирующиеся с другими информационными технологиями и средствами (с CASE, СУБД).

4. *Открытость и переносимость*. Разработки ведутся с соблюдением стандартов, обеспечивающих данные характеристики.

Перечисленные причины рассматриваются как общие требования к инструментальным средствам создания систем искусственного интеллекта, и в частности, требования к реализации интегрированной экспертной модели в форме языка представления и манипулирования знаниями для задач принятия решений в сложных проектах.

Экспертная система состоит из основных компонентов, представленных на рис. 1 [5]:

1. Решатель (интерпретатор).
2. Рабочая память (БД).
3. База знаний (БЗ).
4. Подсистема преобразования знаний.
5. Подсистема объяснений.
6. Диалоговый интерфейс.

Очевидно, что для применения общих элементов экспертных систем к задачам управления проектами необходимо разработать интегрированную экспертную модель, обеспечивающую способы представления всех компонент экспертной системы.

Решение проблемы

1. *Структура языка описания интегрированной экспертной модели*

Прежде всего, необходимо определить понятие алфавита интегрированной экспертной модели.

Алфавитом называется любое множество символов. Предполагается, что термин «символ» имеет достаточно ясный интуитивный смысл и не нуждается в пояснении. Алфавит не обязан быть конечным и даже счетным, но во всех практических приложениях он будет конечным. Термин *буква* и *знак* будут использоваться как синонимы термина *символ* для обозначения элемента алфавита. Если написать последовательность символов, располагая их один за другим, то получится цепочка символов. Термин

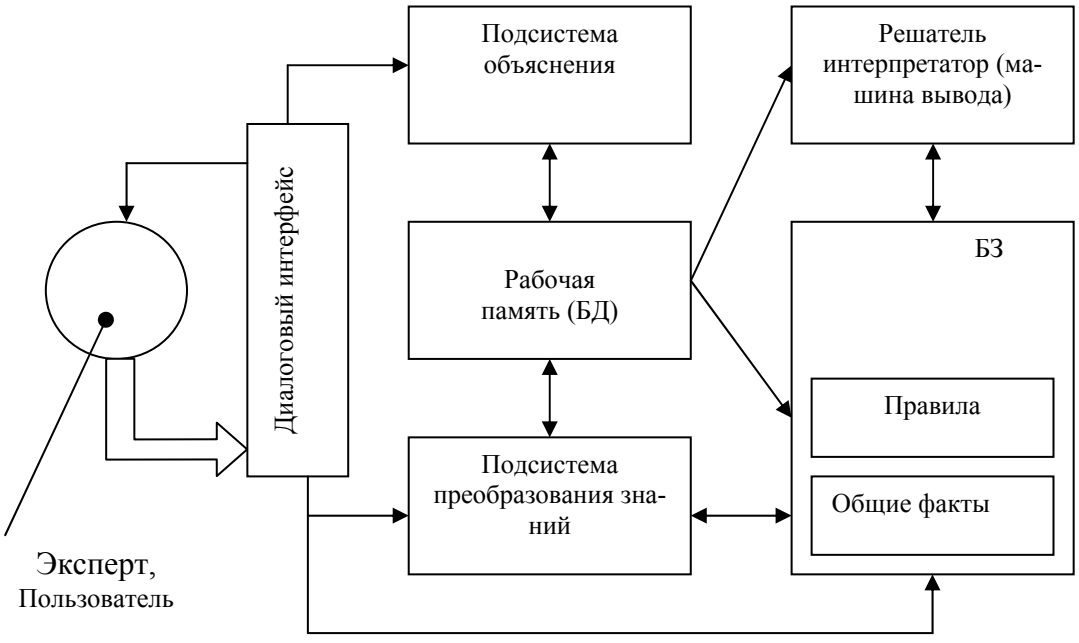


Рис. 1. Схема взаимодействия элементов экспертной системы

слово и *строка* (а иногда, особенно при лингвистических интерпретациях, *предложение*) часто используется как синонимы термина *цепочка.* Пустая цепочка не содержит не одного символа и обозначается *e* [6].

Соглашение 1. Буквы *a,b,c* и *d* обозначает отдельные символы, буквы *t, u, v, w, x, y, z* обозначают цепочки символов. Цепочки, состоящие из *i* символов a, обозначаются a^i. тогда a^0 пустая цепочка e.

Определение 1. *Цепочка в алфавите* Σ определяется следующим образом:

1) *e* - цепочка в Σ;

2) если *x* - цепочка в Σ и $a \in \Sigma$, то *xa* – цепочка в Σ;

3) *y* – цепочка в Σ тогда и только тогда, когда она является таковой в силу 1. или 2.

Определим операции над цепочками.

Если *x* и *y* – цепочки то цепочку *xy* называют *конкатенацией* (или с*цепление*) *x* и *y*.

Обращением цепочки *x* (обозначается x^R) называется цепочка *x*, записанная в обратном порядке, т.е. если $x=a_1...a_n$, где все a_i – символы, то $x^R=a_n...a_1$. Кроме $e^R=e$. Пусть *x, y, z*- произвольные цепочки в некотором алфавите Σ. Назовем *x* префиксом цепочки *xy*, а *y* – суффиксом цепочки *xy*. Цепочку *y* называется подцепочкой цепочки *xyz*. Префикс и суффикс цепочки являются ее подцепочками. Например, *ba* – префикс подцепочки *bac*. Если $x \neq y$ и *x*-префикс (суффикс) цепочка *y*, то *x* называется *собственным* префиксом (суффиксом) цепочки *y* [6-8].

Длина цепочки – это число символов в ней, т.е. если

$$x = a_1...a_n ,$$

где все a_i- символы, то длина цепочки *x* равна *n*. Длина цепочки обозначается $|x|$.

Определение 2. *Языком* в алфавите Σ называется множеством цепочек в Σ. Под это определение подходит почти любое понятие языка в том числе и естественные языки.

Обозначим через Σ^* множество, содержащие все цепочки в алфавите Σ, включая *e*. Например, если Σ - бинарный алфавит $\{0,1\}$, то $\Sigma^* = \{e,0,1,00,01,11,000,001,...\}$. Так как язык – это множество, то операции объединения, пересечения, нахождения разности и дополнения применимы и к языкам. Операции конкатенации можно применить к языкам так же, как к цепочкам [7].

Определение 3. Если язык *L* таков, что никакая цепочка в *L* не является собственным префиксом (суффиксом) никакой другой цепочки в *L*, то говорят, что *L* обладает *префиксным* (суффиксным) свойством [8].

Определение 4. Пусть L_1 - язык в алфавите Σ_1, а L_2 - язык в алфавите Σ_2. Тогда язык L_1L_2, называется *конкатенацией* (также сцеплением или произведением) языков L_1 и L_2 - это язык $\{xy \mid x \in L_1 \text{ и } y \in L_2\}$ [9].

Для многих языков нельзя установить верхнюю границу длины самой длинной цепочки языка. Следовательно, приходится рассматривать языки, содержащие сколь угодно много цепочек. Очевидно, что такие языки нельзя определить исчерпывающим перечислением входящих в них цепочек, и, стало быть, необходимо искать другие способы их описания. Известно несколько методов описания языков. Один из них состоит в использовании порождающей системы, называемой грамматикой. Цепочки языка строятся точно определенным способом с применением правил грамматики. Одно из преимуществ определения языка с помощью грамматики состоит в том, что операции, производимые в ходе синтаксического анализа и перевода, можно сделать проще, если воспользоваться структурой, которую грамматика приписывает цепочкам (“предложениям”) языка [10].

Грамматика – это математическая система, которая придает цепочкам языка полезную структуру [8].

В грамматике, определяющей язык L, используются для конечных непересекающихся множеств символов множество нетерминальных символов, которое часто обозначаются буквой N_1, и множество терминальных символов, обозначаемое Σ. Из терминальных символов образуется слова (цепочки) определяемого языка. Нетерминальные символы служат для порождения слов языка L. Сердцевину грамматики составляет конечное множество P правил образования, которые описывают процесс порождения цепочек языка.

Правило – это просто пара цепочек, или, точнее элемент множества $(N \cup \Sigma)^* N (N \cup \Sigma)^* x (N \cup \Sigma)^*$. Иначе говоря, первой компонентой правила является любая цепочка, содержащая хотя бы один нетерминал, а второй компонентой –любая цепочка.

Соглашение 2. Правило (α, β) записывается в виде $\alpha \to \beta$.

Определение 5. Грамматикой называется четверка

$$G = (N, \Sigma, P, S),$$

где N - конечное множество нетерминальных символов, или нетерминалов (иногда называется вспомогательными символами, синтаксическими переменными или понятиями); Σ - не пересекающееся с N конечное множество терминальных символов, или терминалов; P-конечное подмножество множества $(N \cup \Sigma)^* N (N \cup \Sigma)^* x (N \cup \Sigma)^*$ (элемент (α, β) множества P называется правилом (или продукцией) и записывается в виде $\alpha \to \beta$); S - выделенный символ из N, называемый начальным (или исходным) символом.

Соглашение 3. Для обозначения n правил

$$\alpha \to \beta_1$$
$$\alpha \to \beta_2$$
$$\ldots\ldots\ldots\ldots$$
$$\alpha \to \beta_n$$

удобно пользоваться сокращенной записью $\alpha \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$.

Грамматики можно классифицировать по виду их правил.

Пусть $G = (N, \Sigma, P, S)$ - грамматика. Грамматика G называется контекстно-свободной (КС-грамматика) (или бесконтекстной), если каждое правило из P имеет вид $A \to \alpha$, где $A \in N$, $\alpha \in (N \cup \Sigma)^*$. С помощью КС-грамматики можно определить большую часть синтаксических языков программирования. В ходе самого процесса компиляции синтаксическую структуру, придаваемую входной программе КС-грамматикой, можно использовать при построении перевода этой программы. Синтаксическую структуру входной цепочки можно определить по последовательности правил, применяемых при выводе этой цепочки. Таким образом, на часть компилятора, называемую синтаксическим анализатором [4], можно смотреть, как на устройство, которое пытается выяснить, существует ли в некоторой фиксированной КС-грамматике вывод входной цепочки.

В грамматике может быть несколько выводов, эквивалентных в том смысле, что во всех них применяются один и те же правила в одних и тех же местах, но в различном порядке. Определить понятие эквивалентности двух выводов для грамматик произвольного вида сложно, но в случае КС-грамматик можно внести удобное графическое представление класса эквивалентных выводов, называемое деревом вывода.

Дерево вывода в КС-грамматике $G = (N, \Sigma, P, S)$ – это помеченное упорядоченное дерево, каждая вершина которого помечена симво-

лом из множества $N \cup \Sigma\{e\}$. Если внутренняя вершина помечена символом A, а её прямые потомки - символами $X_1, X_2, ..., X_n$, то $A \to X_1, X_2, ..., X_n$ - правило этой грамматики.

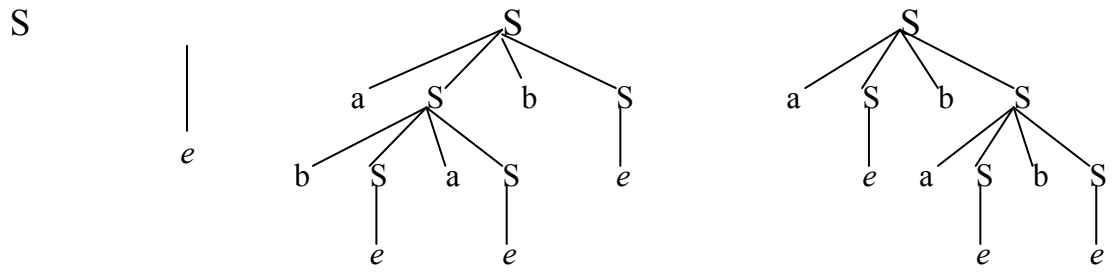
Определение 6. Помеченное упорядоченное дерево D называется *деревом вывода* (или *дерево разбора*) в КС-грамматике $G(A) = (N, \Sigma, P, A)$, если выполняются условия:

1. Корень дерева D помечен A.
2. Если $D_1, ..., D_k$ – поддеревья, над которыми доминируют прямые потомки корней дерева, и корень дерева D_i помечен X_i, то $A \to X_1, ..., X_k$ – правило из множества P. D_i должно быть деревом вывода в грамматике $G(X_i) = (N, \Sigma, P, X_i)$, если X_i - не терминал, и D_i состоит из единственной вершины помеченной X_i, если X_i – терминал.
3. Если корень дерева имеет единственного потомка, помеченного e, то этот потомок образует дерево, состоящее из единственной вершины, и $A \to e$ – правило из множества P. На рис. 2 изображены деревья разбора в грамматике $G = G(S)$ с правилами $S \to aSbS \mid bSaS \mid e$ [11].

Определение 7. Кроной дерева разбора называется цепочка, которая получится, если выписывать слева направо метки листьев. Например, кроны деревьев выводов, показанные на рис 2., таковы: (а) *S*, (б) *e*, (в) *abab*, (г) *abab*.

2. *Моделирование синтаксиса языка интегрированной экспертной модели*

В результате анализа функций экспертной системы можно выделить такую входную информацию необходимую для работы экспертной системы: переменные цели, начальные значения переменных,



а) б) в) г)

Рис. 2. Пример дерева разбора

описание лингвистических типов, описание лингвистических переменных, правила, вводимые переменные, действия при завершении работы (нахождения цели) экспертной системы. Согласно этому списку можно разбить входную информацию (входной файл) на такие разделы [12]:

1. Раздел цели (обязательный)
2. Раздел инициализации (обязательный)
3. Раздел описания лингвистических типов
4. Раздел описания лингвистических переменных
5. Раздел правил (обязательный)
6. Раздел ввода
7. Раздел завершения

Раздел цели

Обязательный раздел, который начинается с ключевого слова GOAL (цель) и содержит список целей (переменных) которые надо найти. Синтаксис раздела в виде формул Бэкуса-Науэра имеет вид:

<Цель>::=GOAL: <имя>|<имя1>{,<имя2>..};

Раздел инициализации

Обязательный раздел, который содержит список переменных, используемых для работы экспертных систем. Этот раздел начинается с ключевого слова INITIAL (инициализация). Синтаксис раздела в виде формул Бэкуса-Науэра:

<инициализация>::=INITIAL:
<имя1>=<значение1> [//<коментарий1>]
{<имя2>=<значение2>[//<коментарий2>]};
<значение>::=|true|false|unknown|null|<число>|
<множество>|<строка>;

причем такие слова как *true, false, unknown, null* могут быть в любом регистре.

Раздел описания лингвистических типов

Не обязательный раздел, который содержит нечеткое множество. Это совокупность пар вида $(x, \mu(x))$ - значение и соответственно функция принадлежности в этом значении. Этот раздел начинается с ключевого слова FUZZYTYPE (нечеткий тип). Синтаксис раздела в виде формул Бэкуса-Науэра:

<Лингвистический тип >::=FUZZYTYPE
<имяЛТ1>=<Свойство_Характеристика1> (<пара_1>[,<пара_2>…<пара_n>])
…
<Свойство_Характеристика1>(<пара_1>[,<пара_2>…пара_n>])
…
<имяЛТn>=<Свойство_Характеристика1>(<пара_1>[,<пара_2>..<пара_n>])
…
<Свойство_Характеристика1>(<пара_1>[,<пара_2>..<пара_n>]),
где <имя ЛТ>::=<имя>,
<Свойство_Характеристика >::=<имя>,
<пара>::=<Значение>,<Функция принадлежности>
<Значение>::=<число>,
<Функция принадлежности>::=<число>.

Раздел описания лингвистических переменных

Не обязательный раздел, который содержит названия лингвистических переменных и их типов. Этот раздел начинается с ключевого слова FUZZYVAR (нечеткая переменная). Синтаксис раздела в виде формул Бэкуса-Науэра:

<Лингвистические переменные >::=FUZZYVAR
<имя ЛТ1>:<имя ЛТ>
…
<имя ЛТn>:<имя ЛТ>,
где <имя ЛТ>::=<имя>,
<имя ЛП>::=<имя>.

Раздел правил

Обязательный раздел, который состоит из одного или нескольких последовательных разделов правил. В каждом разделе специфицируется одно правило. Раздел правила начинается с ключевого слова RULE (правило) и содержит несколько предложений, которые могут появляться в любом порядке в пределах раздела правила. Синтаксис раздела в виде формул Бэкуса-Науэра:

<правило>::=RULE: <имя>
PRIORITY:<число>
COST:<число>
READY:<выражение>|<выражение>
{;<выражение>..}
IF:<условие>
THEN: <выражение> | <выражение>
{;<выражение>..}|<INPUT>|<SQL>
REASON:<строка>

PRIORITY – приоритет правила. Если несколько правил определяют одну и ту же неизвестную переменную, то должен быть критерий выбора правил. Диапазон приоритета от 1 до 100, где 100 – наивысший приоритет.

COST – цена правила. Если несколько правил определяют одну и ту же неизвестную переменную, то должен быть критерий, задающий порядок оценивания.

READY – серия команд, которые выполняются до проверки предложения ***IF***. Она используется для инициализации рабочих переменных.

IF (обязательный в разделе правила) – основанием предложения является логическое выражение, выполнятся действие в том случае если условие будет *true*. Зарезервированный раздел правила, анализируемый синтаксическим анализатором но не выполняемый транслятором.

THEN (обязательный в разделе правила) – специфицирует те действия, которые выполняются, если условие правила будет *true*. Каждый оператор последовательности представляет собой любую команду.

REASON – предложение имеет текст рассуждения, который выводится в процессе или после консультации.

Раздел ввода

Состоит из одного или нескольких последовательных разделов ввода. В каждом разделе специфицируется одна переменная. Раздел ввода начинается с ключевого слова **VAR**. Синтаксис этого раздела зависит от типа вводимой переменной и способа ввода (ввод переменной с клавиатуры, выбор из списка, ввод из базы данных):

<Вводимая переменная> :: = VAR:<имя>: <Тип переменной>

<Тип переменной>::=BOOLEAN | NUMBER |STRING | SELECT_NUM | SELECT_ST

BOOLEAN

<SQL>|{<комментарий>|{переменная}| <комментарий>{переменная_1}...{переменна_n} <комментарий>}…{<комментарий>|{переменная}| <комментарий> {переменная_1}... {переменна_n} <комментарий>}

NUMBER

<SQL>|<RANGE_NUM>|{<комментарий>| {переменная}|<комментарий>{переменная_1}... {переменна_n}<комментарий>}...{<комментарий>| {переменная}|<комментарий>{переменная_1}... {переменна_n} <комментарий>}

Вводимое число может быть ограничена с помощью команды RANGE, где первая цифра это левое (*min*) включающее ограничение, а вторая после запятой это правое (*max*) включающее ограничение.

STRING

<SQL>|<RANGE_ST>|{<комментарий>| {<переменная>}|<комментарий>{<переменная_1>} ...{<переменна_n>}<комментарий>}... {<комментарий>|{<переменная>}|<комментарий> {<переменная_1>}...{<переменна_n>} <комментарий>}

Длина строки может быть ограничена с помощью команды RANGE, где цифра после команды обозначает максимальную длину вводимой строки.

SELECT_NUM

&:<число1>|<SQL>&:<число 2>|<SQL> ...

&: <число n>| <SQL>

{<комментарий>|{<переменная>}| <комментарий> {<переменная_1>}...{<переменна_n>} <комментарий>}...{<комментарий>|{<переменная>} |<комментарий>{<переменная_1>} ...{<переменна_n>} <комментарий>}

SELECT_ST

&: <символ>|<символ>{<символ>}|<SQL>

&: <символ>|<символ>{<символ>}|<SQL>...

&: <символ>|<символ>{<символ>}| <SQL>

{<комментарий>|{<переменная>}|<комментарий> {<переменная_1>}...{<переменна_n>} <комментарий>}...

{<комментарий>|{<переменная>}|<комментарий> {<переменная_1>}...{<переменна_n>} <комментарий>}

Раздел завершения

Представляет собой конструкцию переменных условия (IF ... THEN ... ELSE ... ENDIF) с применением операторов вывода на экран (OUTPUT), оператора работы с реляционной базой данных (SQL), оператора метки (LABEL), оператора перехода (GOTO) и оператора выполнения выражения (EXECUTE). Раздел завершения начинается с ключевого слова DO. Синтаксис раздела в виде формул Бэкуса-Науэра:

<завершение>::= DO:
<SQL>{<конструкция условия> | <OUTPUT> | <SQL> | <LABEL> | <GOTO> | <EXECUTE>}|
<конструкция условия>{<конструкция условия>| <OUTPUT> | <SQL>} | <LABEL> | <GOTO>|<EXECUTE>|
<OUTPUT>{<конструкция условия> | <OUTPUT>|<SQL> <LABEL> | <GOTO> |<EXECUTE>}
<конструкция условия>::=IF <условие>
THEN
<SQL> | <OUTPUT> | <LABEL> | <GOTO> |<EXECUTE>| {<конструкция условия> | <OUTPUT> | <SQL>| <LABEL> | <GOTO> |<EXECUTE>}
<конструкция условия> {<конструкция условия>|<OUTPUT>|<SQL>| <LABEL> | <GOTO> |<EXECUTE>}|
<OUTPUT>{<конструкция условия> | <OUTPUT>|<SQL>| <LABEL> | <GOTO> |<EXECUTE>}
ELSE
SQL>{<конструкция условия>|<OUTPUT>|<SQL>| <LABEL> | <GOTO> |<EXECUTE>}|
<конструкция условия> {<конструкция условия> |<OUTPUT>|<SQL>| <LABEL> | <GOTO> |<EXECUTE>}|
<OUTPUT>{<конструкция условия>|<OUTPUT>|<SQL>| <LABEL> | <GOTO> |<EXECUTE>}
ENDIF

Главной особенностью этого раздела является то, что может включать рекурсивные вложенности которые можно представить в качестве примера в виде дерева разбора для конструкции условия (рис. 3) [13].

<выражение>::=<переменная>:=(<переменная>|<число>|<строка>)<оператор> (<переменная>|<число>|<строка>){<оператор> (<переменная> | <число>|<строка>)};
<условие>::=(<переменная>|<число>|<строка>)
<оператор>(<переменная>|<число>|<строка>)
{<оператор>(<переменная> |<число> | <строка>)};
<INPUT>::=INPUT(<переменная>);
<SQL>::=SQL(<Строка запроса на языке SQL (SELECT)>);
<LABEL>::=LABEL(<имя метки>);
<GOTO>::=GOTO(<имя метки>);
<EXECUTE >::=EXECUTE(<выражение>);
<RANGE_NUM>::=RANGE:{<число_min>},{<число_max>}
<RANGE_ST>::=RANGE:<число_max>;
<число>::=<цифра>{,<цифра>..}[.{<цифра>..}];
<строка>::=”<символ>|<символ>{<символ>}”;
<символ>::=<буква>|<цифра>|!|+|-|,|.|;|:|..|\|;
<имя>::=<буква>|<буква>{<буква>|<цифра>.. };
<буква>::=A|B|..|Z|a|b|..|z;
<цифра>::=0|1|2|3|4|5|6|7|8|9;
<оператор>::= + | - | * | / | > | < | = | && | || | ? | + =|-=|>=|<=|&=|;

В результате создания языка можно выделить такие ключевые слова: GOAL, INITIAL, RULE, READY, PRIORITY, COST, IF, THEN, REASON, VAR, NUMBER, STRING, BOOLEAN, SELECT_ST, SELECT_NUM, SQL, RANGE, DO, ELSE, ENDIF, OUTPUT, INPUT, EXECUTE, GOTO, LABEL, FUZZYTYPE, FAZZYVAR. Для создания гибкого языка заполнения базы знаний можно заменять эти ключевые слова, настраивая язык под потребности пользователя. Например, можно заменить эти слова соответственно русскими понятиями как: ЦЕЛЬ (GOAL), ИНИЦИАЛИЗАЦИЯ (INITIAL), ПРАВИЛО (RULE), ДО (READY), ПРИОРИТЕТ (PRIORITY), ЦЕНА (COST), ЕСЛИ (IF), ТО (THEN), ОПИСАНИЕ (REASON), ВВОД (VAR), ЧИСЛО (NUMBER), СТРОКА (STRING), ЛОГИЧЕСКАЯ (BOOLEAN), ВЫБОР_СТРОКИ (SELECT_ST), ВЫБОР_ЧИСЛА (SELECT_NUM), ЗАПРОС (SQL), ДИАПАЗОН (RANGE), ЗАВЕРШЕНИЕ (DO), ИНАЧЕ (ELSE), КОНЕЦ_ЕСЛИ (ENDIF), ВЫВЕС-

ТИ (OUTPUT), ВВЕСТИ (INPUT), ВЫПОЛНИТЬ (EXECUTE), ПЕРЕЙТИ (GOTO), МЕТКА (LABEL), ЛИНГВИСТИЧЕСКИЕ ТИПЫ (FAZZYTYPE), ЛИНГВИСТИЧЕСКИЕ ПЕРЕМЕННЫЕ (FAZZYVAR) В дальнейшем разделы будут упоминаться в английском варианте.

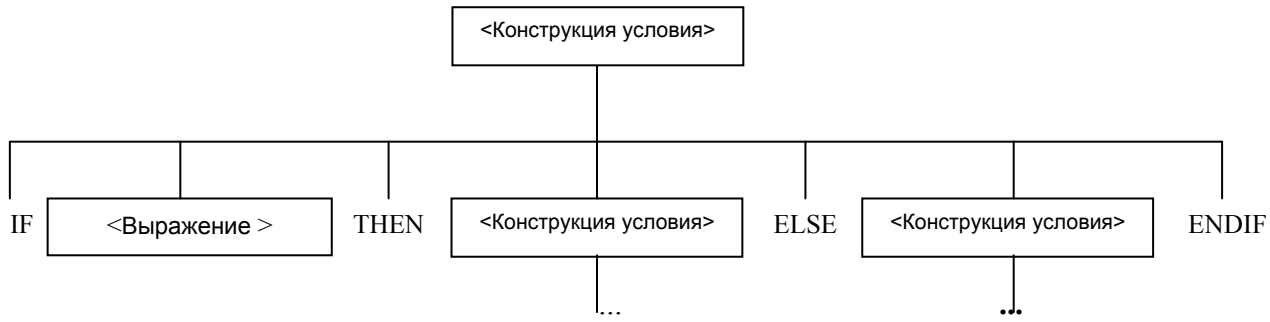


Рис. 3. Пример конструкции условия в виде дерева разбора

Польская запись

Для представления арифметических и логических выражений часто используется польская запись, которая просто и точно указывает порядок выполнения операций. Кроме того, она не требует скобок. Знания в продукционной экспертной системе состоят из арифметических и логических выражений, поэтому благоразумно использовать польскую запись как структуру для хранения знаний в экспертной системе. В этой записи, впервые примененной польским логиком Я. Лукашевичем [3], операторы следуют непосредственно *перед* операндами (или *за* операндами). Поэтому ее иногда называют *префиксной (или постфиксной) записью*. Например, A*B записывается как *BA, A+B+C – как + C*BA, A*(B+C/D) – *+/DCBA (для префиксной записи).

Идентификаторы в префиксной польской записи следуют в обратном порядке относительно инфиксной записи. Операторы в префиксной польской записи следуют по порядку, в котором должны вычисляться (справа налево). Операторы располагаются непосредственно перед операндами [4].

В данном проекте предусмотрены следующие операции:

- логические операции:
  - ‘!’ – отрицание
  - ‘&&’ – логическое И
  - ‘||’ – логическое ИЛИ
  - ‘>’ – больше
  - ‘<’ – меньше
  - ‘>=’ – больше или равно
  - ‘<=’ – меньше или равно
  - ‘=’ – равно
  - ‘!=’ – неравно
- арифметические операции
  - ‘-‘ – унарный минус
  - ‘+’ – сложение
  - ‘-’ – вычитание
  - ‘/’ – деление
  - ‘*’ – умножение
- операции над множеством
  - ‘+=’ – добавление элемента или множества
  - ‘-=’ – удаление элемента или множества
  - ‘?’ – количество элементов в множестве
  - ‘&=’ – вхождение элементов в множество
  - '[ ]' – обращение к элементу множества.

Совокупность этих операторов, переменных и значений, расположенных согласно польской записи

в своей совокупности, составляют базу знаний экспертной системы.

3. *Особенности синтаксиса языка интегрированной экспертной модели на примерах*

Следует заметить, что регистр букв в имени переменной не существенен, т.е. переменная "temp" эквивалента "tEmP"; переменная может содержать также русские символы. Цифры также могут входить в название переменных, кроме первого символа. Все разделы должны быть отделены между собой пустой строкой. Раздел завершения начинается с ключевого слова DO, остальные команды пишутся с новой строки.

Работа со многими целями

Программный продукт позволяет работать с экспертной системой, у которых несколько целей. Поиск каждой цели осуществляется в соответствии с порядком описания переменных в разделе цели. Рассмотрим следующий пример.

Пример 1:

```
GOAL: x1, x2
INITIAL:
a=unknown
b=unknow
c=unknown

RULE: R1
IF: a!=0
THEN: x1:=(2*a+5*b+c)/a;
RULE: R2
IF: a!=0
THEN: x2:=(a+b+c)/a;

VAR: a: number
input a
VAR: b: number
input b
VAR: c: number
input c
DO:
IF x1!= unknown THEN output (x1={x1})
endif
IF x2!= unknown THEN output (x2={x2})
endif
```

Работа с множествами

Переменная в интегрированной экспертной системе - это множество любого типа, т.е. допустимы операции:

```
TEMP:=10;
TEMP+=20;
TEMP+= “HI”;
TEMP-=20;
TEMP+=TRUE;
```

В результате TEMP будет равно множеств {10,"HI",TRUE}. Для получения значения из множества используется индексный вызов: TEMP[1]=10, TEMP[2]="HI", TEMP[3]=TRUE, TEMP[4]=UNKNOWN. Для определения количества значений в множестве приемлема такая запись: TEMP=3.

Пример 2:

```
GOAL: a
INITIAL:
b={1, true, false, “HI”, 45}
c={false, “ПРИВЕТ”, 55, true}

RULE: R1
IF: b!=unknown
THEN: a:=b<операция>c;

DO:
output({a})
```

Пример 3. Пусть заданы два множества следующего вида: b={1, TRUE, FALSE, “HI”, 45}, c={FALSE, “ПРИВЕТ”, 55, TRUE}.

Приведем некоторые теоретико-множественные операции над множествами и представим их в виде табл. 1:

Таблица 1

Примеры некоторых теоретико-множественных операций над множествами

b+c	{1, 1, 56, 2, 1, 1, 56, 2, 0, 0, 55, 1, 0, HI, ПРИВЕТ, 55, 1, 45, 45, 100, 46}
b-c	{1, 1, -54, 0, 1, 1, -54, 0, 0, 0, -55, -1, 0, 0, 55, 1, 45, 45, -10, 44}
b*c	{0, 1, 55, 1, 0, 1, 55, 1, 0, 0, 0, 0, 0, 0, 55, 1, 0, 45, 2475, 45}
b/c	ОШИБКА: ДЕЛЕНИЕ НА НОЛЬ
b+=c	{1, TRUE, FALSE, HI, 45, ПРИВЕТ, 55}
b-=c	{1, HI, 45}
?b+=c	{5, FALSE, ПРИВЕТ, 55, TRUE}
?(b+=c)	{7}
b&&c	{FALSE, TRUE, TRUE, TRUE, FALSE, TRUE, TRUE, TRUE, FALSE, TRUE, FALSE, FALSE, TRUE, TRUE, TRUE, TRUE, TRUE, FALSE, TRUE, TRUE, TRUE}
b||c	{TRUE,TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, FALSE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE}
!b	{FALSE, FALSE, TRUE, TRUE, TRUE}
"ПРИ-ВЕТ"&=c	{TRUE}

Работа с оператором INPUT

Ввод переменных осуществляется с помощью оператора INPUT в разделе правил. В разделе завершения этот оператор игнорируется, например:

GOAL: g
INITIAL:
a=unknown
b=unknow
c=unknown
d=unknow
f=unknown
RULE: R1
IF: c>=5
THEN: input (a);
input (b);
g:= a+b;
input (vopros);
RULE: R2
IF: c>5
THEN: input (f);

input (d);
g:= f+d;
input (vopros);
VAR: a: number
input a
VAR: b: number
input b
VAR: d: number
input d
VAR: f: number
input f
VAR: c: number
Range: 1,10
input c
VAR: vopros: SELECT_ST
&: да
&: нет
Хотите увидеть результат на экране?
DO:
IF vopros= “да” THEN
output ({g})
endif

Использование циклов и оператора EXECUTE в разделе завершение

Организацию циклов в программном продукте можно осуществить с помощью условных переходов (LABEL, GOTO). Оператор EXECUTE аналогичен одному выражению в THEN в разделе правил:

GOAL: A
INITIAL:

```
K=unknown

RULE: R1
IF: ?(K+={4,8})>1
THEN: A+= “QQQ1”
RULE: R2
IF: K=1
THEN: A+= “QQQ2”
RULE: R2
IF: K=1
THEN: A+= “QQQ3”
VAR: K: number
Input K
DO:
EXECUTE (i:=1);
LABEL (asd);
EXECUTE (S:=A[1]);
output (Элемент №{i}={S});
EXECUTE (I:=i+1);
IF i<=?A THEN GOTO (asd);
endif
output ({A})
```

В результате если для переменной *K* значение 1, то на экране результатов появятся следующие значения:

элемент № 1=QQQ1
элемент № 2=QQQ2
элемент № 3=QQQ3
QQQ1 QQQ2 QQQ3

Работа с базой данных. Ввод переменных с базы данных

Использование баз данных в экспертной системе приобретает все большое распространение. В интегрированной экспертной системе работа с базами данных осуществляется в разделе правил, в разделе ввода и завершения с помощью оператора SQL. Рассмотрим пример извлечения значений из базы данных, обработки и сохранения в базе данных (таблица SHEV_TEMP с двух полей (NUM, FIO)). Если в SQL запросе необходимо использовать внутреннюю переменную экспертной системы, причем она должна начинаться с двоеточия ":".

```
GOAL: A
INITIAL:
b=unknown

RULE: R1
IF: FIO1>””
THEN:A:=B

VAR: b: number
SQL (SELECT NUM from SHEV_TEMP where FIO1:=FIO1)
присвоение переменной
VAR:= FIO1: SELECT_ST
SQL (SELECT FIO from SHEV_TEMP)
введите имя
DO:
output (цель{a})
IF (FIO!= unknown)&&(a!=unknown) THEN
EXECUT (a:=a+1)
EXECUT (FIO1:=FIO1+”1”);
output (цель{a});
SQL (insert into SHEV_TEMP values (:A,:FIO1));
endif
```

Система будет работать в такой последовательности: для определения цели необходима переменная FIO (она описана в разделе ввода). Возникнет окно ввода переменной FIO, в котором будут представлены варианты значений, взятых из базы данных. После выбора переменной правило активизируется и для определения переменой цели запросится переменная *b* из раздела ввода. Значение переменой *b* также определится из базы данных. В разделе завершения значение цели занесется в базу данных.

Заключение

В работе представлены основные элементы интегрированной экспертной модели в форме языка представления и манипулирования знаниями для задач принятия решений в сложных проектах. Дан-

ный язык достаточно удобен для представления продукционных отношений и позволяет расширить использование нечетких множеств и отношений для задач управления сложными проектами.

В дальнейших исследованиях управления проектами предполагается применение языка для моделирования процесса принятия управленческих решений в имитационных системах моделирования.

Литература

1. Шеридан Т.Б. , Феррелл У.Р. Системы человек-машина: модели обработки информации, управления и принятия решений человеком-оператором: Пер. с англ. – М.: Машиностроение, 1980. – 400 с.

2. Интеллектуальные системы принятия проектных решений / А.В. Алексеев, А.Н. Борисов, Э.Р. Вилюмс, Н.Н. Слядзь, С.А. Фомин. – Рига: Зинатне, 1997. – 320 с.

3. Искусственный интеллект: Справочник: В 3 кн. / Под ред. Д.А. Поспелова. – М.: Радио и связь, 1990. – Кн.2: Модели и методы. – 304 с.

4. Dutta A., Basu A. An Artificial Intelligence Approach to Model Management in Decision Support Systems // IEEE Computer.— 1984.— N 9.— P. 24—31.

5. Экспертные системы. Принципы работы и примеры / Под ред. Р.Форсайта. - М.: Радио и связь, 1987. - 224 с.

6. Ларичев О.И., Мошкович Е.М. Качественные методы принятия решений. Вербальный анализ решений. – М.: Наука. Физматлит, 1996. – 208 с.

7. Язенин А. В. Гибридная экспертная система для планирования // Изв. АН СССР. Техн. кибернетика. -1989. - N5. - С. 162-167:

8. Шапиро Д.Н. Принятие решений в системах организованного управления: использование расплывчатых категорий. - М.: Энергоатомиздат, 1983. – 185 с.

9. Экспертные системы для персональных компьютеров: Методы, средства, реализация: Справ. пособие / В.С. Крисевич, Л.А. Кузьмич, А.М. Шиф и др. – М.: Высшая школа, 1990. – 197 с.

10. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. – Спб.: Питер, 2000. –384 с.

11. Осуга С. Обработка знаний: Пер. с япон.— М.: Мир, 1989.— 293 с.

12. Компьютерная программа «Оболочка продукционной экспертной системы “ExpertSYS”» / Соколов А.Ю., Касьян О.В., Мелешенко С.Ю., Сафронов М.Я., Яровая О.В. / Свид. гос. регистр. ПА № 4938. – Зарег. в гос. департ. интеллект. собств. Мин. образования и науки Укр., рег. 24.10.2001., выдан 14.12.2001. – Опубл. Каталог гос. рег. Вып. 5. – 2001. – С. 88.

13. Элти Дж., Кумбс М. Экспертные системы: концепции и примеры: Пер. с англ.— М.: Финансы и статистика, 1987.— 191 с.

Поступила в редакцию 11.10.03

Рецензент: д-р техн. наук, доцент А.Ю. Соколов, Национальный аэрокосмический университет им. Н.Е. Жуковського «ХАИ», г. Харьков

УДК 004.891:004.942

Мова представлення знань в експертних системах підтримки прийняття рішень в складних проектах /*С.Ю. Мелешенко, А.Р. Емад, М.Н. Хилал, О.В. Ярова* // Радіоелектронні і комп'ютерні системи. 2003. № 4. С. .

У статті представлені основні елементи інтегрованої експертної моделі в формі мови представлення й маніпулювання знаннями для задач прийняття рішень у складних проектах. Представлено структуру мови опису інтегрованої експертної моделі у вигляді визначень та угод. Проведено моделювання синтаксису мови інтегрованої експертної моделі, на прикладах розглянуті його специфічності.

Табл. 1. Іл. 3. Бібліогр.: 13 назв.

UDC 004.891:004.942

Knowledge representation language of decision-making expert systems in complex projects / S. Meleshenko, A. R. Emad, M.N. Hilal, O. Yarovaya // Radioelectronic and computer systems. 2003. № 4. PP. .

In this work the basic elements of integrated expert system have been represented. These elements have form of representation and manipulating language for decision-making tasks in complex projects. The structure of definition language for integrated expert models had been represented. This structure is represented as definitions and agreements. Modeling language syntax for integrated expert model are considered on examples.

Tabl. 1. Fig. 3. Ref.: 13 items.

Мелешенко Светлана Юрьевна ассистент кафедры производства радиоэлектронных систем летательных аппаратов Национального аэрокосмического университета «ХАИ»

Емад А.Р. соискатель кафедры производства радиоэлектронных систем летательных аппаратов Национального аэрокосмического университета «ХАИ»

Хилал М.Н. соискатель кафедры информатики Национального аэрокосмического университета «ХАИ»

Яровая Ольга Владимировна старший преподаватель кафедры информатики Национального аэрокосмического университета «ХАИ»