тия, дополнения и согласования с иными социально значимыми для населения средствами информирования, административными службами и органами, призванными решать задачи предупреждения в условиях приближающихся кризисных ситуаций, а также при развитии социальных и природных катастроф. Все перечисленные факторы имеют столь же важное значение в корректировке функций и систем человека, как и биологически значимые физические факторы внешней среды.

Искусственные интеллектуальные возможности современных компьютерных систем и разрабатываемые для указанных целей возможности современного понятийного аппарата системологии в режиме самообучающейся структуры позволяют надеяться на совершенствование прогнозов и их возрастающую значимость, надежность и эффективность в рамках предлагаемой системы корректировки функций и состояния организма отдельного человека, что имеет большое значение при управлении социальными структурами в целом.

Список литературы: 1. *Лагутин М.Ф.* Солнечно-земное взаимодействие и квантовая терапия // Фотобиология и фотомедицина. 1998. №1. С. 109-113. **2.** *Лагутин М.Ф., Солодовников Г.К., Рудавина Л.В.* Экологические исследования частоты госпитализации больных с психическими расстройствами. Харьков. 1994. 42 с. **3.** *Шурыгин И.Г., Лагутин В.М. и др.* УФ-лидар для озонозондирования // Оптика атмосферы. 1990. Т.3, №10. С. 1056-1059. **4.** *Лагутин М.Ф., Лагутин В.М.* Информационные технологии в коррекции функций, систем и состояний организма по пейджерному каналу // Труды Международной конференции "Радиоэлектроника в медицинской диагностике". Москва, 1995. С. 83-84. **5.** *Лагутин В.М.* Особенности системы мониторинга и принципы подготовки данных о медико-экологических кризисных ситуациях // Материалы Молодежного форума "Радиоэлектроника и молодежь-2002". Харьков: ХНУРЭ. С. 51-52.

Поступила в редколлегию 25.01.2004

Рецензент: д-р техн. наук, проф. Прошкин Е.Г.

Лагутин Владимир Михайлович, старший научный сотрудник кафедры РЭУ ХНУРЭ. Научные интересы: экологическая безопасность и мониторинг внешней среды. Адрес: Украина, 61070, Харьков, ул. Шосткинская, 5, тел. 40-94-44.

УДК 519.713

РАЗРАБОТКА ЭКСПЕРТНЫХ СИСТЕМ ДЛЯ РЕШЕНИЯ ЗАДАЧ ПЛАНИРОВАНИЯ И ПРОЕКТИРОВАНИЯ

ДЮБКО Г.Ф., ФУДЖУ ХАЛЛИД ИССА

Рассматривается процесс разработки экспертных систем на основе трёхкомпонентной модели знаний. Приводится пример разработки базы знаний для создания программных моделей цифровых устройств, описанных как элемент комбинационной логики.

1. Введение

Методы искусственного интеллекта (ИИ) показали свою дееспособность при решении важных практических задач. Экспертная система (ЭС) – одна из форм систем ИИ, и ее построение является актуальной задачей. Основа ЭС – её база знаний. *Цель* предлагаемой работы – построение прототипа ЭС в области проектирования, задача – построение фрагментов баз знаний в конкретной предметной области.

Экспертные системы – это большие программные комплексы, в которых смоделированы знания специалистов в узких предметных областях. ЭС позволяют решать задачи в заданной предметной области аналогично человеку-специалисту (эксперту). Они создаются путём совместной работы эксперта, инженера по знаниям и программиста. Эксперт формулирует необходимые знания, инженер по знаниям представляет их в некотором формальном виде, а программист реализует систему, работающую на знаниях.

Современные экспертные системы конструируются для решения широкого круга проблем в математике, вычислительной технике, программировании, бизнесе, военном деле, образовании, медицине и т.д. Несмотря на многообразие предметных областей, можно выделить общие классы задач, решаемых экспертными системами [1]:

интерпретация – формирование высокоуровневых выводов из набора строк данных;

проектирование – нахождение конфигурации компонентов системы, которая удовлетворяет целевым условиям и множеству проектных ограничений;

планирование – разработка последовательности действий для достижения множества целей при данных начальных условиях и временных ограничениях;

прогнозирование – определение возможных последствий заданной ситуации;

диагностика – определение причин неисправностей в сложных ситуациях на основе наблюдаемых симптомов;

мониторинг – сравнение наблюдаемого поведения системы с её ожидаемым поведением;

инструктирование – помощь в образовательном процессе;

управление – управление поведением сложной среды.

При создании ЭС прослеживается тенденция строить их структурно так, чтобы всё многообразие задач решалось программной архитектурой, единой для всех задач, а различие предметных областей отражалось бы в знаниях, сконцентрированных в одной структуре данных. Знания, таким образом, есть ядро экспертной системы, они сосредоточены в базе знаний (БЗ), которая отображает конкретную предметную область. Решения в ЭС достигаются путем вывода на знаниях, процедуры, которая даёт

результат, обрабатывая знания. Содержание операций этой процедуры зависит от модели представления знаний. Типичная ЭС состоит из модулей: пользовательский интерфейс, база знаний, машина вывода, подсистема объяснений, интерфейс инженера по знаниям.

Разработка экспертных систем требует значительных денежных инвестиций и человеческих ресурсов. При этом большая часть работы посвящается вопросам извлечения знаний из эксперта, выбору подходящей модели знаний, её программной реализации и процессу вывода на знаниях. При выборе модели представления знаний широко используются такие формы представления, как продукционные системы, семантические сети, фреймы, концептуальные графы. Реже используется логическая модель представления, в основе которой лежит исчисление предикатов и которая является теоретической базой для обоснования других моделей представления.

Все названные выше формы представления знаний отражают знания в явном виде и подразумевают централизованную БЗ, управляемую некоторой общецелевой схемой вывода. В последнее время появились работы, в которых утверждается, что интеллектуальное решение проблем не требует централизованного запоминания знаний [2]. Это коннекционистский (нейронные сети), эмерджентный (стохастический), агентно-ориентированный и распределённый подходы. Однако эти работы не поколебали позиций экспертных систем, а лишь заставили по-новому взглянуть на разработку основ ЭС, вводя в них распределённые БЗ, агентное управление процессом решения задач.

2. Экспертная система на базе трёхкомпонетной модели знаний

Для разработки ЭС, как и для разработки любой сложной программной системы, используют инструментальные средства и среды. Наиболее употребляемым из средств является оболочка (генератор ЭС), которая предполагает некоторую форму представления знаний и определённую стратегию вывода. Применение оболочек позволяет минимизировать разработку процедурных компонентов ЭС, реализуя их в оболочке и настраивая на конкретное выполнение наполнением базы знаний.

В дальнейшем ограничимся рассмотрением задач проектирования и планирования, в которых имеется исходное описание задачи на языке предметной области. Описание может представлять целевые условия, проектные ограничения, начальные условия, временные ограничения, поведение входящих в систему подсистем. Подобное описание назовём исходной языковой конструкцией (ИЯК).

В экспертных системах для решения задач проектирования и планирования будем использовать трёхкомпонентную модель базы знаний [3], которая является распределенной базой знаний с различными формами представления и различными системами вывода. Нижним уровнем ТМБЗ служит лингвистическая база знаний (ЛБЗ), представленная продукциями атрибутной транслирующей грамматики. Процесс вывода- синтаксический и семантический анализ в грамматике. Результат анализа представляется в виде синтаксического дерева, концептуального графа, семантической сети.

Верхним уровнем ТМБЗ является система принятия решений. Форма представления знаний здесь, по содержанию – продукционная, по структуре – логическая, в виде хорновских дизъюнктов с функциями в качестве термов. Выводом служит метод резолюций, а базой знаний для вывода – множество правил, фактов и целей. Если правила принятия решений находятся непосредственно в предметной базе знаний (ПРБЗ), то факты и цели зависят от текущей ситуации и формируются на некотором промежуточном уровне. Назовём этот уровень базой метазнаний (БМЗ).

База метазнаний представляется продукционной системой, состоящей из правил «если <условие>, то <действие>», условиями в которых является наличие некоторых сущностей, их свойств и связей, а действиями – формирование элементарных предикатов (фактов, целей). На уровне БМЗ возможно использование нечетких знаний в виде продукций с вероятностными характеристиками или формированием всех вариантов набора фактов и целей с дальнейшим разрешением неопределённости на уровне принятия решений.

Структура экспертной системы, базирующейся на трёхкомпонентной модели знаний, представлена на рис. 1.

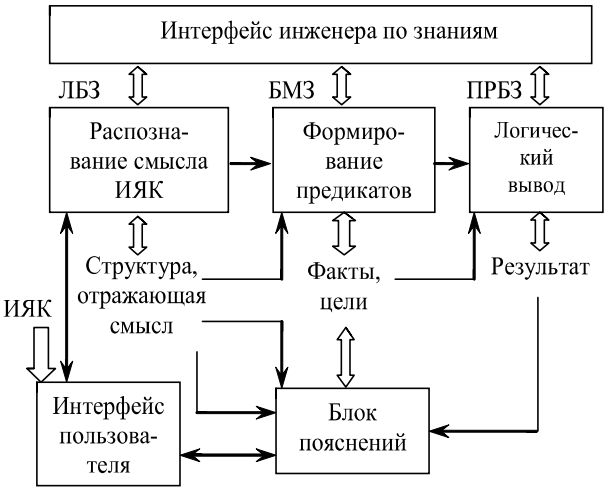


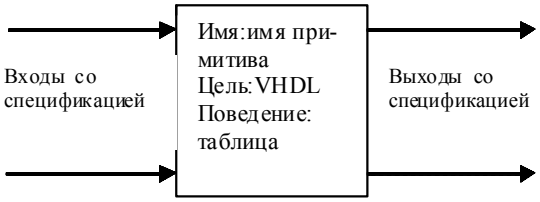
Рис.1. Структура ЭС

Методика построения распределённой трёхкомпонентной базы знаний требует, чтобы сначала были сформулированы правила принятия решений. Литералы хорновских дизъюнктов, представляющие понятия, должны иметь соответствие среди нетерминальных символов грамматики. Поэтому следующим этапом описания знаний является построение грамматики исходных языковых конструкций. Построение базы метазнаний – последний этап в формировании базы знаний. Результатом вывода в БМЗ являются литералы, связывающие лингвистические знания со знаниями принятия решений.

3. Построение базы знаний

Рассмотрим демонстрационный пример конструирования основных элементов экспертной системы, создающей программную модель на языке описания аппаратуры VHDL для компонента (примитива), представляющего элемент комбинационной логики.

Пусть, визуально, примитив представляется как показано на рис. 2. Здесь поведение примитива описывается таблицей истинности, где каждый входной и выходной сигнал представлен битом или битовым вектором. Необходимо разработать средства, с помощью которых визуальное представление на рис. 2 могло бы быть автоматически преобразовано в программную модель на языке VHDL при посредстве оболочки ЭС.



<имяпримитива>:<описание поведения>

Рис. 2. Представление визуальным компонентом примитива и его поведения

Так как методом преобразования является формальный вывод в формальных системах, нужно предложить в качестве средств преобразования конкретные грамматики, базы знаний, алгоритмы. Рассмотрим конструирование этих средств для компонента, поведение которого описывается таблицей истинности.

Первый шаг преобразования состоит в формализации трансформирования графического элемента и текста в графовую продукцию, которая для рассматриваемого случая имеет в левой части имя и интерфейсы компонента, а в правой – поведение в виде таблицы. Графовая продукция является входом блока преобразования заданного компонента в код VHDL и имеет вид:

```
TI_VHDL <имя компонента> (<список соединений интерфейса>): -<список входных имён> (<данные>);<список выходных имен->(<данные>),   (1)
```

где -<список соединений интерфейса> – разделенные двоеточием номер соединения, вход (ключевое слово in) или выход (ключевое слово out), имя соединения, его спецификация; <данные> – матрицы из нулей и единиц; каждое соединение интерфейса отделено друг от друга точкой с запятой.

Для анализа строк (1) можно предложить множество грамматик с различными нетерминальными символами. Однако нам нужна такая грамматика, которая бы позволяла по результатам синтаксического анализа сформировать предикаты-факты и предикаты-цели для формирования фрагмента базы знаний, отвечающей за создание кода на VHDL для случая таблицы. В качестве фрагмента ПРБЗ можно использовать правило в виде хорновского дизъюнкта

```
Код_VHDL_ti (algorithm_VHDL_ti (IK, LI_in, LIS_in, LI_out, LIS_out, MATR_VX, MATR_VYX)) :-VHDL_ti (IK), VHDL_in (LI_in, LIS_in), VHDL_out (LI_out, LIS_out),
VHDL_VX(MATR_VX),
VHDL_VYX (MATR_VYX),   (2)
```

где код_VHDL_ti – предикат, требующий создания кода на VHDL из таблицы истинности с помощью процедуры «алгоритм_VHDL_ti»; VHDL_ti – предикат, определяющий имя компонента; VHDL_in – предикат, определяющий имена входных переменных и их спецификацию; VHDL_out – предикат, определяющий имена выходных переменных и их спецификацию; VHDL_VX – предикат, определяющий входную матрицу; VHDL_VYX – предикат, определяющий выходную матрицу.

Переменные в (2) имеют следующий смысл:

IK – имя компонента; LI_in – список входных сигналов; LIS_in – спецификация входных сигналов; LI_out – список выходных сигналов; LIS_out – спецификация выходных сигналов; MATR_VX – матрица битовых входов; MATR_VYX – матрица битовых выходов.

Здесь же имеет смысл определить процедуру algorithm_VHDL_ti. Все предлагаемые ниже алгоритмы будут записываться в псевдокоде [4]:

```
algorithm_VHDL_ti(IK,LI_in,LIS_in,LI_out,
LIS_out, MATR_VX, MATR_VYX)
1. form_interf (IK, LI_in, LIS_in, LI_out,
 LIS_out)
2. form_zag_arch (IK, LI-in)
3. algorithm_arch_telo (MATR_VX, MATR_VYX, LI_out)
4. form_end ()
```

Алгоритмом «algorithm_VHDL_ti» формируются различные части программы на VHDL в соответствии с описанием языка VHDL [5]. Так, form_interf формирует интерфейс в виде:

```
entity IK' is
 port (LI_in':in LIS_in', LI_out': out LIS_out');
end IK' ,
```

где штрихи у переменных обозначают их содержимое.

«form_zag_arch» формирует заголовок архитектурного тела в виде трех строк:

```
architecture TELO of IK' is
begin
 process (LI_in') ,
```

а «form-end» завершает программу строками

```
end process;
end TELO;
```

Непосредственное поведение компонента определяется алгоритмом «algorithm_arch_telo», который представлен ниже:

```
algorithm_arch_telo(MATR_VX,
MATR_VYX, LI_out)
1. n ← lenght [strok [MATR_VX]]
2. l ← lenght [stolb [MATR_VYX]]+1
3. for i → 0 to n
4.    do if MATR_VYX [i,l]=0
5.           thenTEK ← vector
                 (MATR_VYX, i)
6.                MATR_VYX [i, l] ← 1
7.                k ← 0
8.                PROM[k] ← vector
                  [MATR_VX, i]
9.                k ← k+1
10.               for j ← i+1 to n
11.               do if MATR_VYX [j,l]=0
                       and TEK =vector
                      (MATR_VYX,
                            j)
12.               then PROM [k] ← vector
                        (MATR_VX, j)
13.                    MATR_VYX [j, l]
                                 ← 1
14.                           k ← k+1
15.        form_telo (TEK, PROM, LI_out)
```

Алгоритм «algorithm_arch_telo» создает для одинаковых выходов в одном массиве PROM набор битовых векторов, по которым процедура form_telo формирует текущий фрагмент архитектурного тела. Архитектурное тело использует логические операции “not”, “and” и “or”. При этом операнды операции “and” определяются по входному вектору так: если текущий бит вектора есть 1, то операнд является сигналом соответствующего бита, иначе операндом является сигнал с отрицанием (“not”).

Обладая правилом (2) ПРБЗ и алгоритмами формирования кода VHDL, можно приступить к созданию грамматики, описывающей язык представления компонента, представленного продукцией типа (1). Грамматика должна обладать такими нетерминальными символами, которые бы позволили алгоритму синтаксического анализа сформировать выходную структуру – синтаксическое дерево, обладающую информацией, достаточной для формирования предикатов-фактов, предиката-цели и необходимых структур данных.

Назовем эту грамматику G_{TI_VHDL}. Заметим, что в грамматике G_{TI_VHDL} нетерминальные символы заключены в треугольные скобки <>”, а терминальные представлены в виде некоторого слова без скобок. Ниже приведена транслирующая атрибутная грамматика G_{TI_VHDL}:

```
1. <программа TI_VHDL> → TI_VHDL
     <описание>:-<поведение>
2.<описание> → <имя компонента>  (<вход-
выход>)
3.  <имя компонента>>идентификатор_j
{ZAP_IK(l)}
                                  l ← i
4.<вход-выход> → <список входов>_r1,r2,r3;
<список выходов>_r4,r5,r6 {ZAP_LI in
(l1,l2)} { ZAP_LI S_in(l3)} { ZAP_LI
_out(l4,l5)}
{ ZAP_LI S out(l6)}
l1 ← r1, l2 ← r2, l3 ← r3, l4 ← r4, l5 ← r5,
l6 ← r6
      5.1 <список входов>_r1,r2,r3 → целое_c:
      идентификатор_i: in: type_t
         r1 ← c, r2 ← i, r3 ← t
      5.2  <список входов>_r1,r2,r3;
      → <список номеров>_n:<список
      идентификаторов>_i: in<список
      спецификаций>_t
         r1 ← c, r2 ← i, r3 ← t
      5.3 <список входов>_r1,r2,r3;
      → <список входов>_l1,l2,l3; целое_c;
      идентификатор_i: in: type_t
      r1 ← list(l1,c), r2 ← list(l2,i),
            r3 ← list(l3,t)
      6.1 <список номеров>_n → <список
      номеров>_n1, целое_c
      n ← list(n1,c)
      6.2 <список номеров>_n → ε
      n ← nul
      7.1 <список идентификаторов>_i →
      <список идентификаторов>_l ,
      идентификатор_k
      i ← list(i1,k)
      7.2 <список идентификаторов>_i → ε
      i ← nul
      8.1 <список спецификаций>_t → <список
      спецификаций>_t1, type_t2
      t ← list(t1,t2)
      8.2 <список спецификаций>_t →  ε
                                  t ← nul
      9.1 <список выходов>_r1,r2,r3 → целое_c:
      идентификатор_i:out: type_t
           r1 ← c, r2 ← i, r3 ← t
      9.2 <список выходов>_r1,r2,r3 → <спи-
          сок номеров>_n: <список
      идентификаторов>_I :o u t  :<список
           спецификаций>_t
```

r1←n, r2←i, r3←t

9.3 <список выходов>$_{r1,r2,r3}$→<список выходов>$_{l1,l2,l3}$; целое$_c$: идентификатор$_i$: out: type$_t$

r1←list(l1,c), r2← list(l2,i), r3← list(l3,t)

10.<Поведение>→<список входных имен>$_{r1}$(<входная таблица>$_{r2}$);

<список выходных имен>$_{r3}$(<выходная таблица>$_{r4}$)

{Form_MATR_VX(l1,l2})

{Form_MATR_VYX(l3,l4)}

l1←r1, l2←r2, l3←r3, l4←r4

11.1<список входных имен>$_r$→ идентификатор$_i$ [целое$_{c1}$, целое$_{c2}$]

r←list(i,c1,c2)

11.2 <список входных имен>$_r$→ <список входных имен>$_{r1}$,идентификатор$_i$

r←list(r1,i)

11.3 <список входных имен>$_r$→ ε

r←nul

12.1 <список выходных имен>$_r$→идентификатор$_i$ [целое$_{c1}$, целое$_{c2}$]

r←list(i, c1, c2)

12.2 <список выходных имен>$_r$→ <список входных имен>$_{r1}$,идентификатор$_i$

r←list(r1,i)

12.3 <список выходных имен>$_r$→ ε

r←nul

13.1 <входная таблица>$_r$→бит$_в$

r←list(в)

13.2 <входная таблица>$_r$→<входная таблица >$_{r1}$,бит$_в$

r←list(r1,в)

14.1 <выходная таблица>$_r$→бит$_в$

r←list(в)

14.2 <выходная таблица>$_r$→<выходная таблица >$_{r1}$,бит$_в$

r←list(r1,в)

В грамматике $G^{TA}_{TI\_VHDL}$ аксиомой является нетерминал <программа TI_VHDL>. В продукциях грамматики имеются символы действий, они заключены в фигурные скобки «{ }», и атрибуты. Символы действий представляют собой процедуры, формальные параметры которых получают значение от атрибутов. Формальные параметры также можно считать атрибутами символов действий. Свои значения атрибуты получают в соответствии с правилами, приписанными к атрибутной продукции. Символы действий присваивают значения переменным, конкретизирующим элементарные предикаты. ZAP_IK, ZAP_LI_in, ZAP_LIS_in, ZAP_LI_out, ZAP_LIS_out обозначают присваивание значений переменным IK, LI, LIS для входа и выхода соответственно; FORM_MATR_VX, FORM_MATR_VYX создают входную и выходную матрицы.

Для присваивания значений атрибутам по синтаксическому дереву необходима информация о типе атрибута. Все атрибуты терминального символа считаются заданными, они получают свое значение при вводе компонента и лексическом анализе. Ниже приводятся типы атрибутов всех нетерминальных символов грамматики G_{TI_VHDL}:

<список входов>r1$_{r1,r2,r3}$: атрибуты r1, r2, r3 – синтезируемые.

<список выходов>$_{r1,r2,r3}$: атрибуты r1, r2, r3 – синтезируемые.

<список номеров>$_n$: атрибут n – синтезируемый.

<список идентификаторов>$_i$: атрибут i – синтезируемый.

<список спецификаций>$_t$: атрибут t – синтезируемый.

<список входных имен>$_r$: атрибут r – синтезируемый.

<список выходных имен>$_r$: атрибут r – синтезируемый.

<входная таблица>$_r$: атрибут r – синтезируемый.

<выходная таблица>$_r$: атрибут r – синтезируемый.

Все атрибуты символов действий – наследуемые.

Результатом работы процедуры синтаксического и семантического анализа является синтаксическое дерево с корнем <программа TI_VHDL> и значения, содержащиеся в структурах данных IK, LI_in, LIS_in, LI_out, LIS_out, MATR_VX, MATR_VYX и полученные в результате обхода синтаксического дерева, присваивания значений атрибутам и выполнения символов действий.

База метазнаний по результатам синтаксического и семантического анализа должна сформировать предикаты-факты и предикат-цель, которые составляют вместе с правилом (2) базы знаний рабочую базу знаний, где выполняется логический вывод. Правило БМЗ, которое выполняет это преобразование, в продукционной форме имеет вид:

«если корнем синтаксического дерева является нетерминал <программа TI_VHDL>, то формировать предикаты: не код_VHDL_ti (X), VHDL_ti (IK'), VHDL_in (LI_in', LIS_in'), VHDL_out (LI_out', LIS_out'), VHDL_VX (MATR_VX),VHDL_VYX(MATR_VYX)»

В рассматриваемом нами случае поведения, описанного таблицей истинности, для формирования элементарных предикатов вполне достаточно информации о корне синтаксического дерева и значениях стандартных переменных, являющихся термами в этих предикатах.

4. Пример функционирования

Рассмотрим теперь пример функционирования разработанных подсистем интеллектуальной системы преобразования. Пусть рассматривается компонент AND_2, являющийся стандартным блоком «и» на два входа и один выход. Визуальное представление компонента имеет вид:

AND_2: Таблица:A1,A2(0,0;0,1;1,0;1,1);B (0,0, 0,1).

Графовая продукция для этого компонента имеет вид

TI_VHDL AND_2 (1,3: A1, A2: in: BIT; 2: B: out: BIT): - A1, A2 (0,0,0,1,1,0,1,1); B (0,0,0,1).
Результатом синтаксического анализа и символов действий при обходе синтаксического дерева для AND_2 будут следующие присваивания:

```
IK¬'AND_2', LI_in¬'A1,A2',LIS_in¬'BIT',LI_out
¬'B',LIS_out¬'BIT',MATR_VX¬'0,0,0,1,1,0,1,1',
MATR_VYX¬'0,0,0,1'.
```

Обработка синтаксического дерева с помощью БМЗ приведет к формированию предикатов:

```
не код_VHDL_ti (X), VHDL_ti (AND_2), VHDL_in
([A1,A2], BIT), VHDL_out (B, BIT), VHDL_VX
(0,0,0,1,1,0,1,1), VHDL_VYX (0,0,0,1).
```

Логический вывод в рабочей базе знаний и выполнение «algoritm VHDL_ti» дает программу на VHDL:

```
entity AND_2 is
port (A1, A2: in BIT; B: out BIT)
end AND_2;
architecture TELO of AND_2 is
process (A1, A2)
begin
if (not A1 and not A2) or (not A1 and A2) or(A1
and not A2) then BЬ0;
if A1 and not A2 then BЬ1
end process;
end TELO;
```

5. Выводы

Предложены формализмы для представления лингвистических знаний, знаний принятия решений, метазнаний преобразования форм представления знаний при автоматическом построении программной модели, исходя из описания проекта на языке разработчика. Аналоги предлагают универсальный язык описания проекта, что опять же сводится к некоторому алгоритмическому языку, неестественному для разработчика.

Применение трехкомпонентной базы знаний при разработке ЭС для решения задач проектирования и планирования, в частности, для построения программной модели на языке VHDL, представляющей некоторое цифровое устройство, обеспечивает ряд преимуществ. Распределение базы знаний в трёх компонентах позволяет повысить эффективность работы со знаниями, обеспечивает выразительность языка представления знаний, даёт простой механизм работы с нечёткими знаниями, повышает эффективность алгоритмов вывода(не менее 20%). Применение же ЭС для построения программных моделей цифровых устройств позволит экономить время разработчика до 30%.

Разработка базы знаний для создания программных моделей цифровых устройств продемонстрировала работоспособность метода и отработала методику составления базы знаний в виде комбинации грамматики, логического вывода и вывода в продукционной базе знаний.

Литература: 1. *Waterman D.A.* A Guide to Expert Systems Reading, MA: Addis on Wesey, 1986. **2.** *Люгер, Джордж Ф.* Искусственный интеллект: стратегии и методы решения сложных проблем, 4-е издание.: Пер. с англ. М.: Издательский дом «Вильямс», 2003. 864 с. **3.** *Гребенник В.А.* Трехкомпонентная модель представления знаний для проектирования интеллектуальных агентов и экспертных систем. Диссертация на соискание учёной степени кандидата технических наук. Харьковский национальный университет радиоэлектроники. Харьков, 2002. 145с. **4.** *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. М.: МЦНМО, 1999. 960с.**5.** *Армстронг Дж. Г.* Моделирование цифровых систем на языке VHDL: Пер. с англ. М.: Мир, 1992. 175с. **6.** *Хантер Р.* Основные концепции компиляции: Пер. с англ. М.: Издательский дом «Вильямс», 2002. 256с.

Поступила в редколлегию 17.10.2003

Рецензент: д-р техн. наук, проф. Хаханов В.И.

Дюбко Геннадий Фёдорович, канд. техн. наук, профессор кафедры ПО ЭВМ ХНУРЭ. Научные интересы: технология создания программного обеспечения ЭВМ. Адрес: Украина, 61140, Харьков, пр. Гагарина, 38, кв.70, тел. дом. 27-13-26, служ. 702-14-46.

Фуджу Халлид Исса, аспирант кафедры ПО ЭВМ ХНУРЭ. Научные интересы: экспертные системы. Адрес: Украина, 61166, Харьков, ул. Коломенская, 27, кв. 60, тел. дом. 702-07-66, служ. 702-14-46.