РАЗДЕЛ XIV. ТЕХНИЧЕСКИЕ НАУКИ

Балтрашевич В.Э.

Анализ комбинации блоков ввода и логического вывода в ЭС на базе списка атрибутов

Санкт-Петербургский Государственный электротехнический университет

(Россия, Санкт-Петербург)

doi: 10.18411/trnio-09-2021-37

Аннотация

Рассматривается разработка программных моделей блоков ввода знаний (БВ) и логического вывода (БЛВ) для инструментальной экспертной системы (ИЭС) на базе списка атрибутов для автономного и сетевого режима работы. Приводится уточнение основных понятий системного анализа многоуровневых ЭС. Разрабатываются базы знаний (БЗ), состоящие из поверхностных знаний (ПЗ) и глубинных знаний (ГЗ).

Ключевые слова: Поверхностные и глубинные знания, экспертные системы, системный анализ, имитационное моделирование.

Abstract

The development of software models of knowledge input (BV) and logical inference (LB) blocks for an instrumental expert system (IES) based on a list of attributes for autonomous and network modes of operation is considered. The clarification of the basic concepts of the system analysis of multilevel ES is given. Knowledge bases (KB) are being developed, consisting of superficial knowledge (LK) and in-depth knowledge (DK).

Keywords: Surface and in-depth knowledge, expert systems, systems analysis, simulation.

На основании опыта проведения системного анализа многоуровневых ЭС, представляется целесообразным упорядочить некоторые понятия.

Основные понятия системного анализа многоуровневых ЭС.

- 1) **Виды ЭС.** Есть два основных вида ЭС: на базе правил продукции (ПП) и на базе списка атрибутов (СА) без правил продукции [1]. Каждый программный блок имеет БЗ, которая управляет его работой, и обрабатывает исходную БЗ. Эти БЗ могут иметь разные структуры. Поэтому при обозначении блоков будем использовать сокращения для внутренней БЗ и затем, сокращение для обрабатываемой БЗ. Так, например, БЛВ_СА_ПП – это программный блок логического вывода, работающий под управление БЗ на списке атрибутов, и обрабатывающий БЗ на правилах продукции. Раньше использовались ЭС_ПП_ПП. В работе также будут проанализированы ЭС_СА_ПП и ЭС_СА_СА. В ЭС_СА отсутствует блок объяснений (в обычном смысле, т.к. нет правил продукции), но они позволяют получить след процесса, который является другим представлением блока объяснений [1].

Системный анализ ЭС требует создания моделей [2]. В качестве формализма используем язык эксперта. Т.е. фактически мы получаем **второй уровень** ЭС на основе соответствующих БЗ.

- 2) **Виды блоков ЭС.** Рассматриваем многоуровневые ЭС, как результат системного анализа ЭС. Нижний уровень будем называть «аппаратным». «Аппаратные» блоки записываются на языке программирования, а программные блоки (результат формализации «аппаратных» блоков) – на языке эксперта. Т.е. рассматриваем **«аппаратные» и программные блоки**. Сокращения для «аппаратных» блоков будут состоять только из сокращения

для типа входных данных. Так, например, «аппаратный» БВ_СА – это блок для ввода БЗ_СА. А «аппаратный» БЛВ_СА – это блок для логического вывода на БЗ_СА. При создании разных структур ЭС будем использовать **разные «аппаратные» блоки**: БВ_СА, БВ_ПП, блок объяснений (БО_ПП), БЛВ_СА, БЛВ_ПП, блок просмотра БЗ_СА либо БЗ_ПП, блоки диалога.

- 3) **Виды БЗ.** Пустая ЭС предназначена для решения задач некоторого класса, после заполнения ее знаниями из вводимой БЗ. Т.е. эта БЗ является основной. С другой стороны, в процессе проведения системного анализа ЭС, мы разрабатываем БЗ, описывающие работу исследуемого блока. Т.е. для нас при системном анализе эта база является основной. Чтобы не запутаться с терминологией и отличать две основные БЗ, предлагается исходную БЗ, поступающую на вход ЭС, называть эталонной. А БЗ, описывающую работу исследуемого блока, так и называть. Таким образом, мы делим БЗ на:

- БЗ исследуемого блока;
- эталонные БЗ (исходные данные (БЗ) для исследуемого блока).

Выбор эталонных БЗ. При проведении системного анализа ЭС мы будем разрабатывать БЗ, описывающие работу блоков ИЭС. Но т.к. в процессе разработки возможны ошибки, то нам нужны проверенные исходные данные, т.е. эталонные БЗ проверенные и отлаженные (из любой предметной области). Эти БЗ должны удовлетворять определенным требованиям. Мы должны провести системный анализ структур и процессов в них протекающих. Чтобы проверить структуры, нам надо задать 1) БЗ с ПЗ, 2) БЗ с ПЗ и с коэффициентами доверия (КД), и 3) БЗ с ПЗ и с ГЗ. Это БЗ для ИЭС на базе ПП. В качестве эталонной БЗ для ИЭС на базе списка атрибутов (ИЭС_СА) будем использовать БЗ_СА дихотомии, т.е. БЗ с ПЗ и с ГЗ [1].

БЗ_СА с ПЗ и КД не существует, т.к. нет правил продукции, а КД предназначены для них. БЗ_СА для ПЗ пока нет. Для разработки БЗ_СА для БЗ с ПЗ (без ГЗ – пример ботаническая БЗ) требуется доработка системной части ЭС_СА, так как в основе работы ЭС_СА лежит прямой логический вывод, а для ЭС_ПП – обратный логический вывод.

- 4) **Ввод и стыковка ПЗ и ГЗ одной БЗ.** С помощью «аппаратного» или программного БВ вводим файл с ПЗ. Передаём его в модуль логического вывода (infers). ГЗ с eval_proc записаны в unit PredObl. Этот модуль подключается к unit inferences. Среди ГЗ (для многоуровневых ЭС) есть процедура ввода знаний, которые и стыкуются с ГЗ (с помощью конструкции with BZ do).

- 5) **Виды ГЗ.** ГЗ бывают двух видов:

- простые ГЗ, которые решают исходную задачу под управлением ПЗ (например, дихотомия, классификация растений и др.);
- ГЗ для многоуровневых ЭС. Они служат для стыковки исследуемой БЗ и БЗ, описывающей исходную задачу. Они «висят» (ни к чему не подсоединены – до времени) и потом соединяются с исходной эталонной БЗ.

Рисунок 1. Использование глубинных знаний в многоуровневых ЭС

В модуле PredObl синтаксически описывается заглушка БЗ (пустая БЗ с правильной структурой) с помощью конструкции with BZ do. Как показал эксперимент – левый прямоугольник может «дышать» при «висящих» ГЗ (до использования эталонной БЗ). Если подключить введенную БЗ (исследуемого блока) для примерной проверки перебора вершин графа, то каждая процедура не находит нужных параметров (т.к. нет правильной БЗ), но выдает ответы заданные по умолчанию.

Перебор вершин графа осуществляется с помощью конструкции [1].

```
if (value <> 'nem') then
```

```
    BaseZnD.GetMainGoal(pItem.NextGoal_1,BaseZnD.GoalList);
```

```
if (value = 'nem') then
```

```
    BaseZnD.GetMainGoal(pItem.NextGoal_2,BaseZnD.GoalList);
```

Если на этой конструкции поставить точку останова, то можно просмотреть перебор вершин графа. Напомним, что каждая вершина графа выдает значение параметра value.

Есть поверхностные знания – они образуют процесс (левый прямоугольник). ГЗ – это горизонтальные ответвления. Справа пока ничего нет (будет правый прямоугольник). Если подключить к ГЗ исходную (исходные данные) БЗ, то система будет работать. Но подключить надо к локальной переменной BZ. Если подключить к полной БЗ, то испортим левый прямоугольник.

Мы имеем два процесса: первый – это процесс работы блока, задаваемый вводимой БЗ исследуемого блока, второй – процесс, задаваемый БЗ, являющейся исходными данными для первой БЗ. У каждого процесса – свой стек целей и стек фактов, граф работы, свой диалог и средства просмотра БЗ.

- б) **Окно «предметной области»** - это интерфейс второго процесса (рис.1). Оно позволяет отвечать на вопросы системы с помощью кнопки «Ответ» и задавать **автоматический и пошаговый режим** работы программных блоков. Отличия режимов – при ответах пользователя.

Реализация пошагового режима в программном БЛВ потребовала модификации «аппаратного» БЛВ. При автоматическом режиме, при наличии вопроса, печатается текст вопроса и подключается блок диалога. После ответа пользователя логический вывод продолжается. При пошаговом режиме – печатается вопрос и осуществляется останов логического вывода. Пользователь может обратиться к другим окнам, или сделать какие-то действия. Например, задать цель логического вывода, просмотреть содержимое стека фактов и стека целей. Потом пользователь нажимает кнопку «Ответ» в окне «Предметной области» и производится запуск блока диалога. После нажатия на кнопку «Вывод» на окне «аппаратного» БЛВ продолжается логический вывод.

При реализации пошагового режима при ответе на вопрос, соответствующая процедура ГЗ (в ниже приведенном примере – это процедура 12) выдает значение выходного параметра «останов». Получив это сообщение, алгоритм логического вывода «аппаратного» БЛВ останавливает логический вывод.

- 7) **Режим работы программных блоков.**

Под автономным режимом работы будем понимать режим, при котором для запуска исследуемого программного блока требуется дополнительная «аппаратная» ИЭС, в качестве исполняющей системы. Для реализации этого режима в окне «Предметной области» добавляется кнопка «ИЭС». При автономном режиме работы сначала вводится исходная эталонная БЗ, а потом вводится БЗ с описанием работы исследуемого блока. Эта ситуация возникает, когда программный блок подключается к работающей системе и использует ранее введенную эталонную БЗ.

При обычном или сетевом режиме работы сначала вводится БЗ с описанием работы одного или двух программных блоков, а затем вводится исходная эталонная БЗ.

Режим работы определяет **порядок ввода БЗ в многоуровневых ЭС.**

- 8) **Реализация ГЗ.** ГЗ реализуются с помощью процедуры eval_proc(), которая на основании конструкции «параметры» языка эксперта связывает ПЗ и ГЗ.

По номеру процедуры запускается процедура или составной оператор. Реализация ГЗ - в отдельном модуле с заглушкой БЗ. Потом вводимые знания присваиваются БЗ заглушки. Используется case по номерам процедур (пример, ниже).

Используются два разных варианта процедуры eval_proc:

procedure eval_proc(number:integer; par:domen; var par1:domen; var val: string40); -

для реализации проверки и возможной подсказки ответов.

procedure eval_proc3(number:integer; par:domen; var val:string40; var KD: integer); -

только для проверки ответов. В ЭС_СА коэффициенты доверия KD не используются.

Соответственно используются разные виды блока «Диалог».

9) Виды блока «Диалог»:

- простой – получает ответ пользователя;
- проверочный – проверяет правильность ответа пользователя;
- подсказывающий – наряду с проверкой ответа пользователя может выдать правильный ответ.

Описание и примеры использования диалогов приведены в [1].

10) Стыковка «аппаратных» и программных блоков в разных сочетаниях.

Стыковка программных блоков будет продемонстрирована ниже на конкретном примере совместной работы БВ_СА_ПП и БЛВ_СА_ПП.

В [3] проводилась проверка работы автономных блоков БВ_СА_ПП и БЛВ_СА_ПП. После ввода знаний с помощью программного БВ_СА_ПП проводилась проверка семантики введенной БЗ с помощью кнопки «Экспресс проверка». С помощью кнопки «ИЭС - вывод» осуществлялся ввод программного БЛВ_СА_ПП, и к нему подключалась введенная БЗ. В чем разница между «Экспресс проверкой» и «ИЭС - вывод»? В первом случае берется «пустой аппаратный» БЛВ и ему подается испытываемая БЗ (ПЗ и ГЗ – если есть). Во втором случае с помощью новой «аппаратной» ИЭС («аппаратные» БВ и БЛВ) вводится файл программного логического вывода и ему передается введенная БЗ. Создается левый прямоугольник (рис.1). ГЗ ждут подключения испытываемой БЗ (ПЗ и ГЗ) и получают ее. Создается правый прямоугольник.

11) Проверка работоспособности программных блоков. Проверка проводится с помощью имитационного моделирования.

12) Накопление знаний (онтология).

Одним из основных достоинств ЭС являются хранящиеся в них знания. Мы рассматриваем разработку знаний об индивидуальной (автономной) и совместной (сетевой) работе блоков ввода (БВ) и логического вывода (БЛВ) разных структур. В [3] разработаны знания работы БВ_СА_ПП и знания работы БЛВ_СА_ПП. Теперь разрабатываем БЗ их совместной работы. Также рассмотрим возможность создания и использования БВ_СА_СА и БЛВ_СА_СА.

Все разработанные и накопленные знания могут быть связаны в общую онтологию, описывающую работу многоуровневых ЭС.

БЗ, описывающая совместную работу БВ_СА_ПП и БЛВ_СА_ПП.

Поверхностные знания (файл 5_trans_inf_SA_PP.txt). Объединенный файл – это аналог соединения БВ_ПП_ПП и БЛВ_ПП_ПП [3].

цель начальная установка

параметры начальная установка

42 42 word no

Замечания.

1. Следующие атрибуты используются в конструкции «параметры» языка эксперта при описании транслятора: *начальная установка, конец_файла, ввод_слова, ввод_слова1, очистка, лексема_число, правило, лексема_вопрос, вопрос, лексема_подсказка, подсказка, лексема_перевод, перевод, лексема_параметры, параметры, лексема_рекомендация, рекомендация.*

2. Для экономии места будем записывать в одной строке несколько предложений языка эксперта. <BK> - возврат каретки.

```
next начальная_установка ввод_слова null
next ввод_слова лексема_число null<BK>next ввод_слова1 лексема_число null
next лексема_число правило_лексема_вопрос<BK>next правило_ввод_слова1 null
next лексема_вопрос вопрос_лексема_подсказка<BK>next вопрос_ввод_слова1 null
next лексема_подсказка подсказка_лексема_перевод
next подсказка_ввод_слова1 null<BK>next перевод_ввод_слова1 null
next лексема_перевод перевод_лексема_параметры
next лексема_параметры параметры_лексема_рекомендация
next лексема_рекомендация рекомендация_конец_файла
next рекомендация_ввод_слова1 null<BK>next конец_файла очистка_ввод_слова1
next параметры_ввод_слова1 null<BK>next очистка_ввод_знаний null
вопрос установка_цели<BK>вы_выбрали_цель_вывода
```

Замечание. Следующие атрибуты используются в конструкции «параметры» языка эксперта при описании логического вывода: ввод_знаний, установка_цели, процедура_найдена, процедура_обработана, подцель_выбрана, правило_найдено, значение_правила, подцель_записана, номер_правила, правило_истинно, правило_ложно, правило_неопределено, факт, запрет_правила, поиск_вопроса, ответ, печать_сообщения, печать_значения_цели

```
next ввод_знаний установка_цели null<BK>next установка_цели подцель_выбрана null
next подцель_выбрана правило_найдено печать_значения_цели
next поиск_вопроса ответ_процедура_найдена<BK>next ответ_подцель_выбрана null
next правило_найдено значение_правила поиск_вопроса
next процедура_найдена процедура_обработана поиск_вопроса
next процедура_обработана подцель_выбрана null
next номер_правила значение_правила null
next значение_правила правило_неопределено null
next подцель_записана подцель_выбрана null
next правило_неопределено подцель_записана правило_ложно
next правило_ложно запрет_правила правило_истинно
next запрет_правила подцель_выбрана null
next номер_запрещенного_правила подцель_выбрана null
next правило_истинно факт null<BK>next факт_подцель_выбрана null
next печать_значения_цели null null
```

Фраза *вопрос установка_цели* используется для задания вопроса о значении цели логического вывода. В результате чего логический вывод останавливается, и пользователь может выполнить ряд нужных ему действий и задать цель логического вывода. После этого пользователь запускает логический вывод на продолжение либо в пошаговом режиме, либо в автоматическом.

Глубинные знания (фрагмент – не приводятся описания процедур трансляции):

```
{описание используемых процедур}
var BZ:TbaseZnD;
begin (*eval_proc*) FormPredObl.Show;
  val:='da'; KD:=100;
  case number of
3:{цель} begin BZ.MainGoal:=FormPredObl.ComboBox1.Text;
  WITH BZ DO begin GetMainGoal(MainGoal,GoalList);end; val:='da'; end;
4:{процедура найдена} with BZ do begin
  pGoal:=TItem3(GoalList.Items[GoalList.Count-1]);
  k:=FindAttributeInAttributes(pGoal.Attribute);
  if k=-1 then begin done2:=True; val:='nem';end else
```

```

begin pItem:=TItem6(Attributes.Items[k]);
if AnsiCompareText(pItem.NameProc,'нем')<>0 then val:='да' else val:='нем';
end; end;
5:{процедура_обработана} with BZ do begin val:='обработана';
  pItem:=TItem6(Attributes.Items[k]);
  GetAnswer(Eval_Proc3,pItem,FactList,GoalList,'ответ', Dialog_Confidence);
end;
6:{правило_найденно} with BZ do begin
  pGoal:=TItem3(GoalList.Items[GoalList.Count-1]);
  if FoundRule(Rules,pGoal.Attribute,k) then
    begin val:='да'; end else val:='нем'; end;
7:{значение_правила} begin with BZ do
  case RuleValue(k,Rules,FactList,Reason,ky) of
    t:begin rule_value:='истина'; end;
    f:begin rule_value:='ложно'; end;
    u:begin rule_value:='неопределено'; end;
  end; val:='да'; end;
71:{правило_истинно}
  begin if (rule_value='истина') then val:='да' else val:='нем'; end;
72:{правило_ложно}
  begin if (rule_value='ложно') then val:='да' else val:='нем'; end;
73:{правило_неопределено}
  begin if (rule_value='неопределено') then val:='да' else val:='нем'; end;
8:{факт} with BZdo begin val:='записан'
  TrueRule(k,FactList,GoalList,ky,done2); end;
9:{запрет_правила} with BZ do begin val:='да';
  TItem5(Rules.Items[k]).Activ:=False; end;
10:{подцель_записана}
  with BZ do begin val:='да'; PutOnGoal(k,Reason,Goallist); end;
11:{поиск_вопроса} with BZ do begin val:='да';
  pGoal:=TItem3(GoalList.Items[GoalList.Count-1]);
  k:=FindAttributeInAttributes(pGoal.Attribute);
  if k=-1 then begin done2:=True; val:='не_найдена';end else
  begin pItem:=TItem6(Attributes.Items[k]);
  if Length(pItem.Quest)>0 then val:='да' else begin done2:=True; val:='нем';end;
  end; end;
12:{ответ} with BZ do begin val:='да';
  pGoal:=TItem3(GoalList.Items[GoalList.Count-1]);
  k:=FindAttributeInAttributes(pGoal.Attribute);
  if k=-1 then done2:=True else
  begin pItem:=TItem6(Attributes.Items[k]);
  if Length(pItem.Quest)>0 then begin
  if not FormPredObl.CheckBox2.Checked then // AUTO
  begin MessageDlg('**ЖДУ ОТВЕТА ' +pItem.Quest,mtInformation,[mbOk],0);
  val:='останов'; exit; end;
  If not Fuzzi then begin FormAnswer1:= TFormAnswer1.Create(Application);
  FormAnswer1.ShowModal; FormAnswer1.Free;
  end elsebegin
  FormAnswerFuzzi:= TFormAnswerFuzzi.Create(Application);
  FormAnswerFuzzi.ShowModal; FormAnswerFuzzi.Free;
  end; end; end; end;
13:{печать_сообщения}

```

```

begin ShowMessage('***HET ЗНАЧЕНИЯ ЦЕЛИ '+#13+ AnsiUpperCase
(BZ.MainGoal) +#13+'ВЫВОД ОКОНЧЕН');
FormPredObl.Free; FormPredObl:=Nil; BZ.Free; BZ:=Nil; end;
14:{печать_значения_цели} with BZ do begin
if not FUZZI then
ShowMessage('***ЗНАЧЕНИЕ ЦЕЛИ'+#13
+AnsiUpperCase(BZ.MainGoal{PrObl})+' '
+AnsiUpperCase(TItem4(FactList.Items[FactList.Count-1]).Value)+
#13+'ВЫВОД ОКОНЧЕН')
else begin k:=-1; Str1:=' РЕЗУЛЬТАТЫ: ';
repeat k:=FindMainGoalInRepository(k+1,BZ.MainGoal{PrObl},FactList);
if k>=0 then begin
pFact:= TItem4(FactList.Items[k]);
Str1:=Str1+(#13+pFact.Attribute+' '+pFact.Value+' '
+ IntToStr(pFact.Confidence_Conclusion)
+' '+IntToStr(pFact.Rule_Number)); end;
until k=-1; ShowMessage(Str1); end; end;
151:{знания2}begin val:='да';ShowMessage('Начинает работать лог. вывод');
FormPredObl.Show; end;
16:{подцель_выбрана}
with BZ do begin if FlagFinal then begin val:='нем'; exit; end;
pGoal:=TItem3(GoalList.Items[GoalList.Count-1]);
if GoalList.Count <> 0 then val:='да' else val:='нем'; end;
{трансляция}
40:{конец_файла} begin if eof(inp) then val:='да' else val:='нем'; end;
42:{начальная установка}begin BaseZnPrObl1:=TBaseZnD.Create;
if MessageDlg('Будете использовать факторы уверенности?',
mtConfirmation,[mbYes,mbNo],0)=IDYES
then BaseZnPrObl1.Fuzzi:=True else BaseZnPrObl1.Fuzzi:=False;
BZ:= BaseZnPrObl1;//Подключение БЗ
if FormPredObl.OpenDialog1.Execute
then NameFile:=FormPredObl.OpenDialog1.FileName;
Assign(inp,NameFile); Reset(inp);error := False;end;
43:{ввод_слова} begin readword(inp, ch); end;
44:{число} begin if ch[1] in ['0'..'9'] then
begin val:='да'; word_integer(ch, nomber); end else val:='нем'; end;
45:{правило}rule(ch, error);
46:{лексема_вопрос}: if ch = 'вопрос' then val:='да' else val:='нем';
47:{вопрос}quest(error);
48:{лексема_подсказка} if ch = 'подсказка' then val:='да' else val:='нем';
49:{подсказка}answer(error);
50:{лексема_перевод} if ch = 'перевод' then val:='да' else val:='нем';
51:{перевод}transfer(error);
52:{лексема_параметры} if ch = 'параметры' then val:='да' else val:='нем';
53:{параметры}parametr(error);
54:{лексема_рекомендация} if ch = 'рекомендация' then val:='да' else val:='нем';
55:{ рекомендация }rec(error);
60:{очистка}begin CloseFile(inp);FormPredObl.ComboBox1.Clear;
FormPredObl.ComboBox1.Text:=BZ.Target[0];
for i:=0 to BZ.Target.Count-1 do
FormPredObl.ComboBox1.Items.Add(BZ.Target[i]);
end; end; end(*eval_proc*);

```

Рисунок 2. Скриншот этапа имитационного моделирования совместной работы БВ_СА_ПП и БЛВ_СА_ПП

В окне «Логический вывод» получаем след вывода, содержащий результаты работы БВ_СА_ПП и БЛВ_СА_ПП вместе.

Если фразу *next очистка ввод_знаний null* в конце описания графа транслятора заменить на *next очистка null null*, то граф логического вывода запускаться не будет, и мы получим БЗ описания только программного БВ_СА_ПП.

Если фразу *цель начальная_установка* заменить на *цель ввод_знаний*, то мы получим БЗ БЛВ_СА_ПП.

Теперь рассмотрим вариант БВ_СА_СА и БЛВ_СА_СА.

Транслятор БВ_СА_СА предлагается разработать самостоятельно, с учетом реализации транслятора БВ_СА_ПП и описания «аппаратного» транслятора БВ_СА. Пусть, также как в трансляторе БВ_СА_ПП, в разрабатываемом трансляторе степень детализации доходит до конструкции «предложения» языка эксперта, а не до лексемы.

Фрагмент транслятора «аппаратного» БВ_СА:

*begin (*предложение*)*

```
if ch = 'вопрос' then quest(error) else if ch = 'подсказка' then answer(error)
else if ch = 'перевод' then transfer(error) else if ch = 'параметры' then parametr(error)
else if ch = 'рекомендация' then rec(error) else if ch = 'next' then following(error)
else if ch = 'список_значений' then ListValue(error)
else if ch = 'help' then Assist(error) else if ch = 'цель' then goal(error)
else begin
```

MessageDlg('Неверное первое слово предложения '+ch, mtError, [mbOk], 0);

*error := True; end; end (*предложение*);*

В качестве исходных данных для БВ_СА_СА надо использовать эталонную БЗ_СА. Возьмем БЗ_СА дихотомии, приведенную в [1].

Проверку семантики программного БВ_СА_СА можно провести с помощью кнопки «Экспресс проверки» окна «Предметной области».

Окно «предметной области» БВ для стыковки почти совпадает с окном, приведенном на рис. 2. Отличие состоит в новых кнопках «Экспресс проверка» и «ИЭС - вывод».

При нажатии на кнопку «Просмотр БЗ» мы увидим внутреннее представление БЗ дихотомии, полученное после трансляции. Для проверки семантики введенных знаний нажимаем на кнопку «Экспресс проверка».

Проводятся следующие действия:

```
FormInfer1:= TFormInfer1.Create(Application); FormInfer1.BZ:=BaseZnPrObl1;  
FormInfer1.ShowModal; FormInfer1.Free;
```

То есть, создается еще один объект класса «аппаратного» БЛВ_СА. Ему передается БЗ_СА дихотомии и проводится проверка работоспособности БЗ дихотомии, т.е. фактически проводится проверка работы транслятора.

Для проверки работоспособности автономного программного БЛВ_СА_СА нажимаем кнопку «ИЭС - вывод». Для запуска программного блока нужна исполняющая система, т.е. ИЭС.

Проводятся следующие действия:

```
procedure TFormPredObl.Button7Click(Sender: TObject);var BZ3: TBaseZnD;  
begin FormEdit1:=TFormEdit1.Create(Application); FormEdit1.ShowModal;  
BZ3:= BaseZnPrObl2; FormEdit2.Free;  
FormInfer1:= TFormInfer1.Create(Application); FormInfer1.BZ:= BZ3;  
FormInfer1.ShowModal; FormInfer1.Free; BZ3.Free;  
ShowMessage('Вывод закончился!'); exit;end;
```

То есть создается еще один «аппаратный» БВ_СА. Вводим текстовый файл с описанием БЛВ_СА_СА. Создается БЗ BaseZnPrObl2. Она передается вновь создаваемому «аппаратному» БЛВ_СА и начинает выполняться файл с описанием БЛВ_СА_СА. Ввод знаний в этом случае осуществлять не надо, т.к. они уже введены и находятся в BaseZnPrObl1 и **их надо передать БЛВ_СА_СА для использования**. Это реализовано следующим образом. В модуле PREDobl2 вводится вспомогательная переменная var BZ:TbaseZnD. В процедуре eval_proc для реализации каждого действия используется конструкция with BZ do.

В процедуре (составном операторе) 151 выполняем **ключевое** действие

```
151:{знания2}  
begin val:='da'; //т.к. знания уже введены, то забираем их, а не вводим.  
BZ:= BaseZnPrObl1;  
end;
```

Далее производится логический вывод для БЗ_СА дихотомии и проверяется правильность БЗ логического вывода. Правильность БЗ_СА дихотомии уже проверена.

В качестве БЛВ_СА_СА можно использовать БЛВ_СА_ПП, в котором не будут использоваться процедуры, связанные с обработкой правил.

В чистом виде БЛВ_СА_СА получается, если в файле IAOC_12.txt [3] убрать атрибут значение_подцели (нужный для АОС), то получим чистый БЛВ_СА_СА. При имитационном моделировании БВ_СА_СА и БЛВ_СА_СА используем БЗ_СА дихотомии.

Выводы.

Проведено уточнение понятий, облегчающее понимание структур и процессов используемых при системном анализе многоуровневых ЭС.

В результате проведенного анализа: сформулированы знания (на языке эксперта) о работе БВ и БЛВ разных структур (ПП и СА) для автономного и сетевого варианта; реализованы варианты стыковки БЗ и блоков; проведено имитационное моделирование разработанных программных блоков в автономном и сетевом режиме.

Ряд проведенных разработок подтвердил и эффективно продемонстрировал полезность системного анализа для исследования многоуровневых ИЭС. Были разработаны программные модели (БЗ) отдельных блоков ИЭС, с помощью которых удалось досконально изучить их работу и ввести улучшающие изменения в описание блоков и в их программную и «аппаратную» реализацию. В качестве формализма при создании моделей использовался язык эксперта.

Системный анализ структур позволил: убрать правила продукций, но сохранить объяснения; предложить новую форму диалога с подсказкой ответа; провести анализ систем, использующих БЗ с разными структурами.

Системный анализ процессов позволил провести имитационное моделирование отдельных блоков и всей ИЭС в целом.

С научной точки зрения, открывается широкая перспектива экспериментов с разработкой различных структур ЭС и их имитационного моделирования. С практической точки зрения, разработанные БЗ могут использоваться для разработки новых ЭС и АОС, решающих разные задачи из разных предметных областей.

1. Балтрашевич В.Э. Интеллектуальная АОС на базе списка атрибутов / В.Э. Балтрашевич. – Beau Bassin: LAP Lambert Academic Publishing, 2019. – 132 с. ISBN 978-620-0-00382-9 (переведена на 8 европейских языков в 2021 г.).
2. Перегудов Ф.И. Введение в системный анализ / Ф.И. Перегудов, Ф.П. Тарасенко. – М.: Высшая школа, 1989.
3. Балтрашевич В.Э. Анализ комбинации блоков ввода и логического вывода ЭС. Наука России: Цели и задачи. Сборник научных трудов по материалам XXVIII международной научной конференции. Август 2021 г. Изд. НИЦ «Л-Журнал». 2021. (в печати)