РАЗДЕЛ XIII. ТЕХНИЧЕСКИЕ НАУКИ

Балтрашевич В.Э.

Технология разработки интеллектуальной АОС на базе списка атрибутов

Санкт-Петербургский Государственный электротехнический университет
(Россия, Санкт-Петербург)

doi: 10.18411/sr-10-02-2019-47

idsp: sciencerussia-10-02-2019-47

Аннотация

Рассматривается технология разработки интеллектуальной автоматизированной обучающей системы (АОС) на базе списка атрибутов. Решаемая задача разбивается на блоки, с каждым из которых связывается атрибут. Подробно представлены этапы разработки поверхностных и глубинных знаний. Особое внимание уделяется представлению графа и следа решения задачи предметной области.

Ключевые слова: Поверхностные и глубинные знания эксперта, контроль знаний обучаемого, логический вывод, представление графа

Abstract

The technology of developing an intelligent automated learning system (AOS) based on the list of attributes is considered. The solved task is divided into blocks, with each attribute associated. Details are the stages of the development of superficial and in-depth knowledge. Particular attention is paid to the representation of the graph and trace of the solution to the problem domain.

Keywords: Surface and depth knowledge of the expert, knowledge control of the trainee, logical conclusion, graph representation

Цель данной работы разработка технологии создания инструментальной АОС на базе списка атрибутов. В работе [1], на примере АОС симплекс-методу задачи линейного программирования, обосновывались преимущества создания АОС на базе списка атрибутов, перед использованием АОС, базирующихся на экспертных системах (ЭС), и сделаны первые шаги по разработке технологии разработки АОС. Отмечалось, что изучаемая задача разбивается на ряд блоков, которые связывались в процесс решения разными вариантами установки целей.

Недостатком всех разобранных методов являлось использование конкретных атрибутов (их названий) в системной части АОС при установке текущей цели, что не позволяло использовать ее как инструментальную. Под инструментальностью понимаем, что ехе-файл АОС во всех АОС один и тот же, а меняются файлы, определяющие предметную область и изучаемый метод, т.е. файлы поверхностных и глубинных знаний, и файл исходных данных для изучаемой задачи. Это фактически интеллектуальные АОС.

Модификация исполнительской системы.

Установка цели. При обработке текущего атрибута функция задания цели задает имя следующего атрибута. После обработки текущего атрибута логический вывод осуществляет удаление из стека целей текущей цели и стек останется пустым, что в свою очередь должно привести к окончанию вывода. Таким образом, функция задания цели должна поместить следующий атрибут в стек целей ниже текущего атрибута. Для этого реализована новая функция PutMainGoalNew, которая записывает новую цель в стек целей ниже текущей. Как только текущая цель будет обработана и удалена, на ее место встает новая цель и т.д. до тех пор, пока новый целевой атрибут не будет null, который не записывается в стек целей, и вывод завершается. Связь атрибутов между

собой задается в файле поверхностных знаний конструкцией next [1]. Предлагаемый блок установки новой цели логического вывода выглядит следующим образом

```
if (value <> 'нет') then

BaseZnD.PutMainGoalNew(pItem.NextGoal_1,BaseZnD.GoalList);
    if (value = 'нет') then

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

где value – выходной системный параметр – значение атрибута, полученное в результате диалога. Т.е конкретные значения атрибутов не используются и разработанная АОС становится инструментальной. Инструментальность АОС приводит к необходимости добавления в файл с поверхностными знаниями конструкции <цель>.

```
<предложение1>: :=<предложение>|<цель>
<цель>:= цель<атрибут>
```

Первоначальная цель должна записываться в стек целей обычной функцией PutMainGoal, т.е. в вершину пустого стека целей.

Чередование автоматического и пошагового режима. АОС может работать в автоматическом или пошаговом режиме. Опыт показал, что целесообразно в окне «Диалог» ввести кнопку «Не знаю» (пока не знаю), которая остановит автоматический режим работы. Когда пользователь будет готов к продолжению работы, он должен иметь возможность продолжить вывод, т.е. в блоке «Логического вывода» он должен нажать кнопку «Ответ» и возможно установить флаг автоматического режима. Пользователь может использовать другие программные средства для получения ответа (например, калькулятор для пересчета элементов матрицы по разным формулам в зависимости от расположения элемента в матрице).

Работа системы без ответов пользователя. Может возникнуть ситуация, при которой пользователь не сможет дать правильный ответ, но не захочет прерывать работу. Если вспомнить, что у нас система проверяет ответы пользователя и соответственно она знает правильный ответ, можно организовать выдачу этого правильного ответа и продолжить работу. Для этого в проверяющей процедуре добавляется выходной параметр с правильным ответом **нужного** типа. В диалоговом окне запроса ответа введена кнопка «Сдаюсь», при нажатии на которую пользователь сообщает системе о невозможности правильного ответа в данный момент. При этом должен быть зафиксирован факт незнания с повышенным штрафом. В этом случае возможны два варианта продолжения работы. Во-первых, система может в виде сообщения показать пользователю значение правильного ответа нужного типа и предоставить ему возможность самому ввести правильный ответ и продолжить работу. Во-вторых, система может не показывать значение ответа, а сразу передать его блоку логического вывода и продолжить работу, а пользователь в дальнейшем может устранить пробел в своих знаниях. Для реализации этого режима в диалоговом окне получения ответа кроме кнопки «Сдаюсь» добавлен флаг «Сам», в зависимости от установки которого система либо показывает, либо не показывает ответ системы. Кроме того, правильный ответ можно поместить в стек фактов, содержимое которого пользователь сможет просмотреть.

След решения задачи. В стеке фактов хранятся значения атрибутов с «признаком записи ответа» yes, раньше в экспертной системе они использовались для ветвления при организации логического вывода (т.е. при обходе графа предметной области (ПрО)). Теперь они тоже нужны для ветвления, но как показал опыт использования сохранение их в стеке фактов, позволяет проследить ход (след) процесса решения задачи ПрО. Поэтому представляет интерес хранить в стеке фактов и значения других атрибутов, которые выдавал пользователь, и проверяла АОС. Поэтому для этих параметров **эксперт должен** поставить признак «записи ответа» yes. При этом

разнотипные ответы надо преобразовать в строковый тип и сохранить в стеке фактов. Для получения файла со следом на форме «Логический вывод» добавлена кнопка «След».

В стек фактов записывается номер проверяющей процедуры. При печати следа в файл можно использовать переводы для атрибутов. След – это разновидность объяснений.

Отметим, что АОС сама может решить задачу (при условии, что пользователь сдастся при ответе на каждый вопрос) и выдать след (протокол) решения задачи.

Теперь рассмотрим технологию разработки поверхностных и глубинных знаний.

Как мы предположили, имеется программная реализация изучаемого метода (например, симплекс-метода ЗЛП).

Шаг 1. Выявление изучаемой процедуры.

Изучаемая задача (симплекс-метод) описывается следующей процедурой (фрагмент):

```
procedure      TSimplexForm.ToolButton3Click(Sender:
TObject);
label 1;
(*Описания используемых структур данных*)
begin
1: if flag3(a, m, n)
then
begin ShowMessage('Допустимой крайней точки не
существует!');exit; end
else if not flag1(a, m, n) then
begin
ichoice(a, m, n, ii);          schoicel(a, n, ii,
s);      rchoice(a, m, n, s, r);
workmat(a, m, n, r, s, nullin, nulstl);
```

Она состоит из вспомогательных процедур:

Шаг 2. Выбираем используемые процедуры (фрагмент).

```
Function flag1(a : matrix; m : index1; n :
index2):Boolean;
Function flag3(a : matrix; m : index1; n :
index2):Boolean;
procedure workmat(var a:matrix; m:index1; n:index2;
r:index1; s:index2;
var nullin : lin; var nulstl :
stl);
procedure ichoice(a:matrix; m:index1; n:index2; var
i:index1);
procedure schoicel(a:matrix; n:index2; i:index1;
var s:index2);
procedure rchoice(a:matrix; m:index1; n, s:index2;
var r:index1);
```

Шаг 3. Каждой процедуре ставится в соответствие некий атрибут.

Напомним конструкцию для параметров языка поверхностных знаний [1]:

```
<параметры> ::= параметры <атрибут> <конец строки>
<номер процедуры> <название процедуры> <признак
типа ответа> <признак записи ответа> <конец строки>
<признак типа ответа> ::= <word> | <integer> |
<real>
<признак записи ответа> ::= <yes> | <no>
```

Пример описания атрибутов на языке эксперта (фрагмент) приведен ниже:

```
параметры крайняя
15 flag1_проверка_ответа_пользователя word yes
параметры кр_не_сущ
3 flag3_проверка_ответа_пользователя word yes
```

```

параметры i
6 ichoice_проверка_ответа_пользователя integer no
параметры s1
7 schoicel_проверка_ответа_пользователя integer no

```

Шаг 4. Для каждого атрибута (в случае необходимости) задаются параметры: вопрос, подсказка, перевод.

Структура файла поверхностных знаний описана в [1]. Для удобства его формирования разработана интерфейсная форма и соответствующая программная поддержка.

Пример описания поверхностных знаний на языке эксперта (фрагмент) приведен ниже:

```

вопрос крайняя
крайняя точка найдена (?)
вопрос кр_не_сущ
крайняя точка не существует (?)
вопрос s1
номер возможного столбца s для разрешающего
элемента=
подсказка крайняя
если все b[i]>0
подсказка кр_не_сущ
если есть строка, все элементы которой <=0
подсказка s1
значение элемента a[i,s] должно быть >0

```

Запускаться в АОС процедуры должны по номерам. Необходимо проверить работоспособность этой структуры.

Шаг 5 (вспомогательный). Вводим связующую процедуру eval_proc. (Этап подготовки и проверки глубинных знаний).

Используем только номер процедуры number и выходной параметр val.

```

(*Описания используемых структур данных*)
procedure eval_proc(number:integer; var
val:string40);
(*Описания используемых процедур*)
begin
val:='да';
case number of
15: begin if flag1(a, m, n) then val:='да'
else val:='нет'; end;
3: begin if flag3(a, m, n) then val:='да'
else val:='нет'; end;
6 : ichoice(a, m, n, ii);
7 : schoicel(a, n, ii, s);

```

Шаг 6 (вспомогательный). Составление управляющей процедуры для проверки eval_proc (временная часть исполняющей системы).

```

begin
eval_proc(20, val); // ввод
eval_proc(11, val); // matprint
33: eval_proc(3, val); // кр_не_сущ
if val='да' then begin eval_proc(16, val); exit
end // print1
else
begin
eval_proc(15, val); // flag1
if val='нет' then
begin
eval_proc(6, val); // i
eval_proc(7, val); // s1
eval_proc(8, val); // r

```

Шаг 7. Вспомогательные процедуры заменяются проверяющими процедурами с тремя дополнительными параметрами (par – проверяемый, par1 – правильный, val – результат проверки).

```
procedure eval_proc(number:integer; par:domen; var
par1:domen; var val:string40); stdcall;
procedure flag1(inp : String40;var inp1 : String40;
a : matrix; m : index1; n : index2; var Val :
String40);
procedure flag3(inp : String40;var inp1 : String40;
a : matrix; m : index1; n : index2; var Val :
String40);
procedure workmat(var a : matrix; m : index1; n :
index2; r : index1; s : index2;
var nullin : lin; var
nulstl : stl);
procedure icoice(inp : Integer; var inp1 :
Integer; a : matrix; m : index1; n : index2;
var i : index1; var Val :
String40);
```

Заметим, что в качестве примера приведены некоторые процедуры.

Шаг 8. Окончательные глубинные знания (фрагмент).

```
(*Описания используемых структур данных*)
procedure eval_proc(number:integer; par:domen; var
par1:domen; var val:string40); stdcall;
(*Описания используемых процедур*)
begin (*eval_proc*)
  val:='да';
  case number of
    15 : flag1(par.dw,par1.dw, a, m, n, Val);
    18 : flag2(par.dw,par1.dw, a, m, n, Val);
    3 : begin
      flag3(par.dw,par1.dw, a, m, n, Val);
      if par1.dw = 'да' then
        begin ShowMessage('Допустимой
крайней точки не существует!');exit; end;
      end;
```

Заметим, что в качестве примера приведены некоторые процедуры.

Подчеркнем, что тип domen – это записи с вариантом для Delphi, или тип Variant для Qt.

Отметим, что для режима «Сдаюсь» при возможности выбора из нескольких значений проверяющая процедура выдает первый возможный вариант (при отсутствии ответа пользователя).

Шаг 9. Представление графа. (Поверхностные знания).

Граф представляется специфическим списком инцидентности, т.е. в виде трех вершин (текущей и двух следующих). При ответе обслуживающей процедуры «нет» управление передается второй вершине (третьему атрибуту). При ответе обслуживающей процедуры «не нет» управление передается первой вершине (второму атрибуту). Ответ «не нет» объясняется тем, что не все процедуры выдают «да», а при установке «признака записи ответа» в yes в файле поверхностных знаний процедуры выдают значение числового параметра, которое для записи в стек фактов преобразуются к строковому типу.

Ниже приводится описание графа решения задачи симплекс-метода (фрагмент файла поверхностных знаний).

```
next ввод matprint null
next matprint кр_не_сущ null
next кр_не_сущ печать1 крайняя
next печать1 null null
```

Заметим, что порядок записи этих инструкций произволен. В исполнительной системе АОС фактически осуществляется поиск в глубину. Конструкция *next* фактически задает упрощенные правила productions.

Шаг 10. Задание целевого (исходного) атрибута.

Для задания цели вводится конструкция «цель атрибут», которая добавляется в файл внешнего представления знаний: *цель ввод*.

Тем самым завершена разработка поверхностных и глубинных знаний. После чего система может быть запущена, проведена работа и получен протокол работы.

Пример фрагмента следа вывода (получен ленивым пользователем, который сдавался при ответе на любой вопрос):

```

След вывода
ввод(ввод исходных данных)=да число ошибок=0 №=20
matprint(печать матрицы)=да число ошибок=0 №=11
кр_не_сущ(допустимая крайняя точка не
существует)=нет число ошибок=15 №=3
крайняя(допустимая крайняя точка найдена)=нет число
ошибок=9 №=15
i(номер возможной строки i для разрешающего
элемента)=1 число ошибок=9 №=6
s1(номер возможного столбца для разрешающего
элемента)=1 число ошибок=6 №=7

```

Таким образом, при описании поверхностных знаний эксперт должен.

1. Использовать форму создания атрибутов.
2. Представить граф списком инцидентности из трех вершин.
3. Задать цель (целевой атрибут).
4. Задать значение «признака записи ответа» нужным атрибутам для организации следа решения задачи предметной области.

Разработанный программный продукт обеспечивает простую разработку АОС. Технология легко переводится на различные языки программирования.

1. Балтрашевич В.Э. Варианты разработки АОС на базе списка атрибутов// Актуальные проблемы науки XXI века: сборник статей международной исследовательской организации «Cognitio» по материалам XXIV международной научно-практической конференции г. Москва, М.: Международная исследовательская организация «Cognitio», 2017. с. 40 - 47.

Босенко А.С., Катаргин В.Н.

**Управление производственными расходами в кузовном ремонте
легковых автомобилей**

*ФГАОУ ВПО «Сибирский федеральный университет»
(Россия, Красноярск)*

doi: 10.18411/sr-10-02-2019-48

idsp: sciencerrussia-10-02-2019-48

В России количество ДТП остается стабильно высоким долгие годы. Сейчас после ДТП владельцы автомобилей получают выплаты по автостраховке. Выплат не всегда достаточно на полное восстановление автомобиля. Кроме того, автовладельцы стремятся сэкономить, ограничиваясь самыми необходимыми ремонтными работами. Особенно сильно после начала кризиса снизилась загрузка малярно-кузовных цехов, поскольку малярно-кузовной ремонт многие автовладельцы производят лишь тогда, когда эксплуатация автомобиля становится невозможной.

Основными факторами для автовладельца при выборе автосервиса для ремонта являются стоимость работ, доверие при ремонте, его скорость и гарантия на выполненные работы. Поэтому для того, чтобы выиграть борьбу за клиента и загрузить