К. т. н. В. Е. ТРОФИМОВ

Украина, Одесский национальный политехнический университет
E-mail: kpra@rtf.ospu.odessa.ua

Дата поступления в редакцию
21.01 2004 г.
Оппонент *В. С. ЛОЗОВСКИЙ*
(ИПР НАНУ, г. Одесса)

АВТОМАТИЗАЦИЯ ПРОЦЕССА ДИАГНОСТИКИ РЭА НА ОСНОВЕ МЕТОДА ЭВРИСТИЧЕСКОЙ КЛАССИФИКАЦИИ

На примере поиска неисправностей в телефонных аппаратах показана программная реализация базы знаний с использованием языка CLIPS.

Автоматизация обнаружения источников неисправностей по результатам наблюдений — одна из задач диагностики РЭА. В данной статье предлагается подход, который предусматривает создание базы знаний в предметной области, представляющей симптомы неисправностей и побудившие их причины или гипотезы. Использование для этой цели языка CLIPS [1] в качестве инструментального средства позволяет применить метод и алгоритм, которые для традиционных языков программирования являются, мягко говоря, неудобными.

Как показано в [2], элементом данных является запись или список. Несколько записей (списков) образуют таблицу или файл. Большой файл есть база данных. Как следует из [3], чтобы данные стали знаниями, они должны быть охарактеризованы функционально, т. е. в терминах действия, а не в терминах структурной организации, как это делается в базе данных. База знаний — это база данных, в которой содержатся сведения о том, как эти данные могут быть использованы. Именно такой смысл будет вкладываться в понятия "база данных" и "база знаний" в дальнейших рассуждениях.

Особенность диагностики РЭА состоит в том, что возможные симптомы и гипотезы можно заранее пронумеровать или перечислить и отразить множество симптомов на множество гипотез. Эта особенность относит диагностику РЭА к типичной проблеме классификации, в которой требуется сделать выбор из ряда возможных решений. В устройствах, для которых симптомы и гипотезы имеют явно выраженную иерархическую организацию, сделать такой выбор достаточно просто. В тех же устройствах, для которых иерархическая схема классификации симптомов и гипотез не является четкой и однозначной, верный выбор оказывается затруднительным.

В этом случае решение можно получить, используя метод эвристической классификации [4]. Применительно к теме данной статьи его суть заключается в установлении неиерархических ассоциативных связей между симптомами и гипотезами, которое требует выполнения промежуточных логических заключений.

В общем случае алгоритм выполнения эвристической классификации состоит из трех этапов.

1. *Абстрагирование от симптомов*. При поиске неисправностей зачастую важно не конкретное значение параметра, характеризующего работу устройства, а то, что оно выше/ниже допустимой величины. Важен не только конкретный симптом, а также то, что он проявляется одновременно с какими-то другими событиями или симптомами.

2. *Сопоставление абстрактных категорий симптомов с абстрактными категориями гипотез*. Сразу сопоставить конкретный симптом с окончательной гипотезой — редкая удача. Гораздо проще сопоставить абстрактные симптомы с абстрактными гипотезами. Применительно к РЭА это означает рассмотрение неисправности на уровне укрупненных узлов. Например, повышенная температура может служить индикатором неисправности системы охлаждения, блока питания или какого-либо другого узла, связанного с данным узлом функционально. Такое сопоставление имеет ярко выраженный эвристический характер, т. к. соответствие между симптомами и гипотезами не является однозначным. Оно требует дополнительных уточняющих вопросов и промежуточных логических заключений.

3. *Конкретизация гипотез*. После того, как определена абстрактная гипотеза, пространство решений существенно сужается. В нем определяется конкретная гипотеза путем дальнейших уточняющих вопросов и логических заключений с включением количественных параметров и сбора дополнительной информации.

Рассмотрим применение этого алгоритма для разработки базы знаний, предназначенной для поиска неисправностей в телефонных аппаратах с электронным номеронабирателем.

Сосредоточим основное внимание на особенностях программной реализации базы знаний. Для этого упростим задачу, ограничившись двумя гипотезами, которые объясняют появление четырех симптомов [5]:

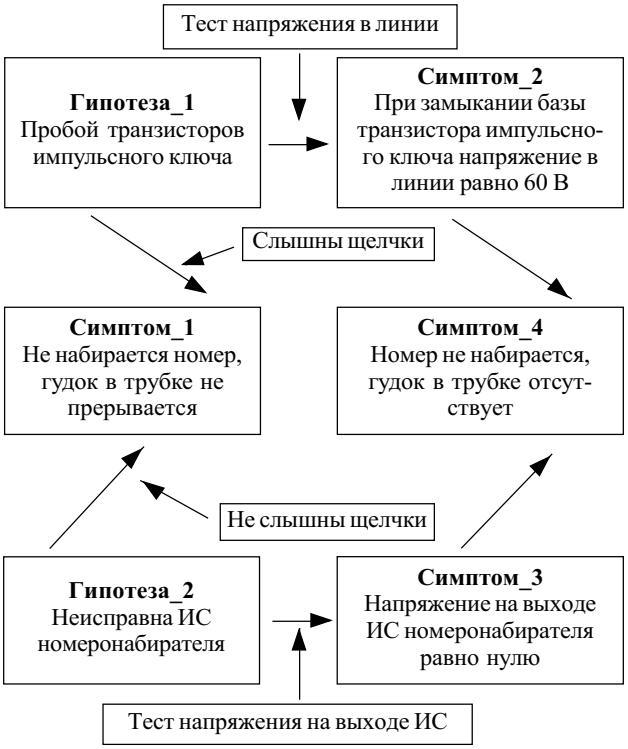
гипотеза_1 — пробой транзисторов импульсного ключа;

гипотеза_2 — неисправна ИС номеронабирателя;

симптом_1 — не набирается номер, гудок в трубке не прерывается;

симптом_2 — при замыкании базы транзистора импульсного ключа на "землю" напряжение в линии равно 60 В;

Модель предметной области



симптом_3 — напряжение на выходе ИС номеронабирателя равно нулю;

симптом_4 – номер не набирается, гудок в трубке отсутствует.

Эти гипотезы и симптомы образуют модель предметной области, которую можно представить графически в виде ассоциативной сети, показанной на **рисунке**.

Как видно из причинно-следственных связей модели, первичными являются **симптом_1** и **симптом_4**. Их появление может быть объяснено как **гипотезой_1**, так и **гипотезой_2**, однако механизм объяснения будет разным.

Связь **симптома_1** с **гипотезой_1** или **гипотезой_2** определяется фоновым условием — событием, наличие или отсутствие которого подтверждает или опровергает эту связь. В данном случае фоновым условием является слышимость щелчков в телефонной трубке при наборе номера.

Связь **симптома_4** с той или иной гипотезой проявляется через промежуточные или вторичные симптомы — **симптом_2** и **симптом_3**. Если такие симптомы не обнаруживаются в процессе поиска неисправности, это является свидетельством против гипотезы. Связь вторичных симптомов с гипотезами параметрирована и требует выполнения тестовых процедур. В данном случае это проверка напряжения в линии и на выходе ИС номеронабирателя. В отличие от **симптома_1**, появление **симптома_4** может быть объяснено **гипотезой_1** и **гипотезой_2** одновременно.

Чтобы автоматизировать процесс поиска неисправности, следует реализовать модель предметной области программно. Для этого очень удобными оказываются средства языка CLIPS — фреймовые структуры и порождающие правила. Первые позволяют объединить симптомы в базу данных и структурировать их вместе с гипотезами в виде, удобном для последующего формирования суждений и выводов. Вторые позволяют дать ответ на вопрос “Что делать с симптомами, которые хранятся в базе данных?”, т. е. сопоставить симптомы с гипотезами, охарактеризовать их функционально и вместе с базой данных получить базу знаний.

Программа, в которой реализованы рассмотренные соображения, может быть, например, такой:

```
; 1 ========================================

; шаблон фрейма для первичных симптомов
(deftemplate symptom
   (slot datum (type SYMBOL))
   (slot condition (type NUMBER)(default 0))
   (slot status (type SYMBOL)(default NIL)))

; шаблон фрейма для вторичных симптомов
   (deftemplate testing
   (slot datum (type SYMBOL))
   (slot status (type NUMBER)(default 0)))

; шаблон фрейма для гипотез
   (deftemplate hypothesis
   (slot which (type SYMBOL))
   (slot status (type SYMBOL)(default NIL)))

; 2 ========================================

; база данных
(deffacts data_base
   (symptom (datum symptom_1))
   (testing (datum symptom_2))
   (testing (datum symptom_3))
   (symptom (datum symptom_4)))

; 3 ========================================

; выбор первичного симптома
(defrule what_happened
   ?event_1 <- (symptom (datum symptom_1)(status
NIL))
   ?event_4 <- (symptom (datum symptom_4)(status
NIL))
   =>
   (printout t crlf "Назовите симптом: "crlf
      "1. симптом_1 (ответ - 1);" crlf
      "2. симптом_4 (ответ - 4)." crlf
      "Ваш ответ: ")
   (bind ?what (read))
   (if (eq ?what 1) then
      (modify ?event_1 (status done)))
      (if (eq ?what 4) then
      (modify ?event_4 (status done))))

; учет фонового условия для симптома_1
(defrule condition
   ?event <- (symptom (datum symptom_1)(condition
0)(status done))
   =>
```

```
   (printout t crlf "Слышны ли щелчки при на-
боре номера?" crlf
      "1. Слышны (ответ - 1);" crlf
      "2. Не слышны (ответ - 2)." crlf
      "Ваш ответ: ")
      (modify ?event (condition (read))))

; правило, принимающее во внимание вторичные
симптомы для симптома_4
(defrule testing_for_symptom_4
   (symptom (datum symptom_4)(status done))
   ?event <- (testing (datum ?symptom)(status
0))
   =>
   (printout t crlf "Проявляется ли симптом "
?symptom "?" crlf
      "1. Да (ответ - 1);" crlf
      "2. Нет (ответ - 2)." crlf
      "Ваш ответ: ")
      (modify ?event (status (read))))

; 4 =======================================

; правило, свидетельствующее в пользу гипоте-
; зы_1 для симптома_1
(defrule hypothesis_1_for_symptom_1
   (symptom (datum symptom_1)(condition
1)(status done))
   =>
   (assert (hypothesis (which
hypothesis_1)(status done))))

; правило, свидетельствующее в пользу гипоте-
; зы_2 для симптома_1
(defrule hypothesis_2_for_symptom_1
   (symptom (datum symptom_1)(condition
2)(status done))
   =>
   (assert (hypothesis (which
hypothesis_2)(status done))))

; правило, свидетельствующее в пользу гипоте-
; зы_1 для симптома_4
(defrule hypothesis_1_for_symptom_4
   (testing (datum symptom_2)(status 1))
   =>
   (assert (hypothesis (which
hypothesis_1)(status done))))

; правило, свидетельствующее в пользу гипоте-
; зы_2 для симптома_4
(defrule hypothesis_2_for_symptom_4
   (testing (datum symptom_3)(status 1))
   =>
   (assert (hypothesis (which
hypothesis_2)(status done))))

; 5 =======================================

; вывод вероятной гипотезы
(defrule result
   (declare (salience -1))
   (hypothesis (which ?hypo)(status done))
   =>
   (printout t crlf "Гипотеза " ?hypo " явля-
ется вероятной." crlf crlf))

; нет объяснений для симптома_4
(defrule nothing
   (testing (datum symptom_2)(status 2))
   (testing (datum symptom_3)(status 2))
   =>
   (printout t crlf "Симптом_4 не может быть
объяснен" crlf
      "ни гипотезой_1, ни гипотезой_2." crlf
crlf))
```

Как видно из приведенного текста, программа состоит из пяти фрагментов. Рассмотрим их подробно.

В *первом* фрагменте конструируются фреймы с именами `symptom`, `testing` и `hypothesis` для представления в модели предметной области соответственно первичных симптомов, вторичных симптомов и гипотез. Слот `datum` фреймов `symptom` и `testing` и слот `which` фрейма `hypothesis` предназначены для хранения названия симптомов и гипотез. Поля этих слотов могут принимать данные символьного типа. Слот `condition` фрейма `symptom` учитывает фоновое условие. Его поле хранит признак проявления фонового условия в виде числа. По умолчанию содержимое слота `condition` равно нулю. Слот `status` во всех трех фреймах предназначен для представления текущего состояния симптомов и гипотез. Для первичных симптомов и гипотез его поле хранит данные символьного типа и по умолчанию является пустым. Для вторичных симптомов содержимым слота является число. По умолчанию оно равно нулю. В данной задаче выбор типа данных, хранящихся в слотах `condition` и `status`, не является существенным и может быть произвольным.

Во *втором* фрагменте программы представлена база данных с именем `data_base`. Она состоит из последовательности записей в виде списков, в каждом из которых содержится, по крайней мере, два поля. В первом поле хранится имя фрейма, а во втором имя слота и название симптома. Содержимое других слотов здесь можно не указывать, т. к. после загрузки программы в интерпретатор языка CLIPS последний увидит их при анализе конструкции фреймов.

Третий фрагмент программы состоит из трех правил с именами `what_happened`, `condition` и `testing_for_symptom_4`. Правило `what_happened` осуществляет выбор первичного симптома. Оно требует от пользователя назвать симптом и помещает в слот `status` соответствующего фрейма значение `done`, изменяя таким образом текущее состояние названного симптома на активное. Правило `condition` учитывает фоновое условие для случая, когда активным является **симптом_1**. Оно требует от пользователя охарактеризовать фоновое условие и в зависимости от ответа помещает в слот `condition` фрейма для **симптома_1** число 1 (при наборе номера слышны щелчки) или 2 (при наборе номера щелчки не слышны).

Правило testing_for_symptom_4 принимает во внимание вторичные симптомы для случая, когда активным является **симптом_4**. Логика работы этого правила аналогична правилу what_happened — с той лишь разницей, что в слоте status соответствующего фрейма текущее состояние вторичного симптома характеризуется числом 1 (симптом проявляется, т. е. активен) или 2 (симптом не проявляется). Существенным отличием правила testing_for_symptom_4 является то, что оно активизируется столько раз, сколько вторичных симптомов присутствует в базе данных. В рассматриваемой задаче таких симптомов два. Поэтому пользователю будет предложено последовательно ответить на два вопроса — проявляется ли **симптом_2** и проявляется ли **симптом_3**. Из результатов работы правил третьего фрагмента программы видно, что здесь выполняется первый этап алгоритма метода эвристической классификации.

Четвертый фрагмент программы состоит из четырех правил, которые выполняют сопоставление симптомов и гипотез. Правило hypothesis_1-_for_symptom_1 свидетельствует в пользу **гипотезы_1** для **симптома_1**, правило hypothesis_2_for_symptom_1 свидетельствует в пользу **гипотезы_2** для **симптома_1**, правило hypothesis_1_for_symptom_4 свидетельствует в пользу **гипотезы_1** для **симптома_4** и правило hypothesis_2_for_symptom_4 свидетельствует в пользу **гипотезы_2** для **симптома_4**. Логика работы правил одинакова и может быть представлена следующим высказыванием:

```
ЕСЛИ симптом_i является активным,
ТО причиной этого является гипотеза_j
```

Например, если активным является **симптом_1**, и эта активность сопровождается фоновым условием “слышны щелчки при наборе номера”, то вступает в действие правило hypothesis_1_for_symptom_1, которое помещает в рабочую память интерпретатора CLIPS фрейм, представляющий **гипотезу_1** как активную.

Таким образом эти правила выполняют второй и третий этапы алгоритма метода эвристической классификации и вместе с правилами what_happened, condition и testing_for_symptom_4 формируют модель предметной области в виде системы фреймов. Вот как выглядит, например, модель предметной области в рабочей памяти интерпретатора CLIPS для случая, когда активными являются **симптом_4** и **симптом_2**:

```
   (symptom (datum symptom_1)(condition
0)(status NIL))
   (testing (datum symptom_2)(status 1))
   (testing (datum symptom_3)(status 2))
   (symptom (datum symptom_4)(condition
0)(status done))
   (hypothesis (which
hypothesis_1)(status done))
```

Из такой структуры легко получить вывод о том, какая гипотеза является вероятной для активного первичного симптома. Это делают правила с именами result и nothing из *пятого* фрагмента программы. Правило result сообщает название вероятной гипотезы и выполняется столько раз, сколько фреймов активных гипотез присутствует в модели предметной области. Работа правила позволяет моделировать случай, когда **симптом_4** активен вместе с **симптомом_2** и **симптомом_3** и объясняется одновременно **гипотезой_1** и **гипотезой_2**. Если же **симптом_4** активен, но при этом не проявляются ни **симптом_2**, ни **симптом_3**, в действие вступает правило nothing, которое сообщает, что **симптом_4** не может быть объяснен ни **гипотезой_1**, ни **гипотезой_2**.

Как следует из описания работы программы, она содержит базу данных и сведения (правила) о том, как эти данные могут быть использованы. Вместе база данных и правила образуют базу знаний, которая позволяет для обнаруженных симптомов определить вероятную гипотезу их появления.

Интеллект такой базы знаний можно повысить добавлением новых симптомов в базу данных и новых правил для сопоставления симптомов с гипотезами. В рассмотренной задаче база данных не является объемной и находится вместе с правилами в одном программном модуле. Однако CLIPS позволяет представлять эти составляющие базы знаний в виде отдельных программных модулей и сохранять в отдельных файлах. Такое разделение обеспечивает удобство структурирования и редактирования базы данных большого размера.

***

Таким образом, подход, предусматривающий разработку базы знаний на основе метода эвристической классификации и использование такого инструментального средства как язык CLIPS, может стать одним из эффективных при решении задач автоматизации обнаружения источников неисправностей в РЭА.

ИСПОЛЬЗОВАННЫЕ ИСТОЧНИКИ

1. Трофимов В. База данных+CLIPS=база знаний // Компьютеры+программы.— 2003.— № 10.— С. 56—61.

2. Кнут Э. Д. Искусство программирования. Т. 3. Сортировка и поиск.— М.: Изд. дом “Вильямс”, 2001.

3. Джексон П. Введение в экспертные системы.— М.: Изд. дом “Вильямс”, 2001.

4. Clancey W. J. Heuristic classification // Artificial Intelligence.— 1985.— Vol. 27.— P. 289—350.

5. Кизлюк А. И. Справочник по устройству и ремонту телефонных аппаратов зарубежного и отечественного производства.— М.: БИБЛИОН, 1997.