Лекція 16.

CLIPS.

Приклад створення експертної системи

РАЗРАБОТКА ПРОТОТИПА ЭКСПЕРТНОЙ СИСТЕМЫ AUTO

Рассмотрим пример создания диагностической экспертной системы, которая позволяет установить причину неисправности автомобиля и выдать соответствующую рекомендацию.

Разработку любой экспертной системы следует начинать с выявления **основных сущностей**, имеющих значение при решении конкретной задачи и **законов**, скорее всего эмпирических, действующих над этими сущностями.

В результате работы с экспертом были установлены следующие эмпирические правила:

1. Двигатель обычно находится в одном из 3 состояний: он может работать нормально, работать неудовлетворительно или не заводиться.

2. Если двигатель работает нормально, то это означает, что он нормально вращается, система зажигания и аккумулятор находятся в норме и никакого ремонта не требуется.

3. Если двигатель запускается, но работает ненормально, то это говорит, по крайней мере, о том, что аккумулятор в порядке.

4. Если двигатель не запускается, то нужно узнать, пытается ли он вращаться. Если двигатель вращается, но при этом не заводится, то это может говорить о наличии плохой искры в системе зажигания. Если двигатель даже не пытается заводиться, то это говорит о том, что искры нет в принципе.

5. Если двигатель не заводится, но вращается, нужно проверить наличие топлива. Если топлива нет – то, скорей всего, для ремонта машины нужно просто заправиться.

6. Если двигатель не заводится, нужно также проверить, заряжен ли аккумулятор, если нет, то его следует зарядить.

7. Если двигатель не заводится и существует вероятность плохой искры в системе зажигания, то необходимо проверить контакты. Контакты могут быть в одном из трех состояний – чистые, опаленные и грязные, в случае опаленных контактов их необходимо заменить, в случае если контакты грязные, их достаточно просто почистить.

8. Если двигатель не заводится, искры нет и аккумулятор заряжен, то нужно проверить катушку зажигания на электрическую проводимость. В случае, если ток не проходит через катушку, то ее необходимо заменить. Если катушка зажигания в порядке, значит необходимо заменить распределительные провода.

9. Если двигатель запускается, но при этом ведет себя инертно, не сразу реагирует на подачу топлива, то необходимо прочистить топливную систему.

10. Если двигатель запускается, но происходят перебои с зажиганием, то это говорит о наличии плохой искры в системе зажигания, для устранения данной неисправности необходимо отрегулировать зазоры между контактами.

11. Если двигатель запускается и стучит, то необходимо отрегулировать зажигание.

12. Если двигатель запускается, но не развивает нормальной мощности, то это может говорить об опаленных или загрязненных контактах (см. правило 7).

13. Возможны ситуации, когда состояние двигателя нельзя описать приведенными выше факторами и машине может потребоваться более детальный анализ состояния.

Из приведенных выше правил можно выделить следующие сущности, имеющие значение при решении задачи.

−Во-первых, для решения задачи экспертной системе необходимо знать, в каком состоянии находится машина, диагностика которой производится. Эксперт выделил три возможных состояния: нормальная работа двигателя, двигатель работает неудовлетворительно, не заводится (см. правило 1).

−Во-вторых, большинство приведенных правил, помимо состояния двигателя в целом, используют понятие состояния вращения двигателя. Согласно этим правилам двигатель может находиться в одном из двух состояний, которые определяются в зависимости от того, способен он вращаться (работать) или нет.

−В-третьих, в некоторых правилах (см. правила 4, 7, 8, 10) используется понятие состояния системы зажигания. Система зажигания может быть в одном из трех состояний: нормальное состояние, нерегулярная работа и нерабочее состояние.

−В-четвертых, в правилах 6 и 8 используется понятие состояния аккумулятора. Аккумулятор может быть в одном из двух состояний: заряженным и разряженным.

Таким образом, можно выделить следующие факты, описывающие состояние автомобиля и его узлов:

Группа фактов, описывающая состояние машины:

working-state engine normal – нормальная работа;

working-state engine unsatisfactory – неудовлетворительная работа;

working-state engine does-not-start. – не заводится

Группа фактов, описывающая состояние двигателя:

rotation-state engine rotates – двигатель вращается;

rotation-state engines does-not-rotate – двигатель не вращается.

Группа фактов, описывающая состояние системы зажигания:

spark-state engine normal – зажигание в порядке;

spark-state engine irregular-spark – искра не регулярна;

spark-state engine does-not-spark– искры нет.

Группа фактов, описывающая состояние системы питания:

charge-state battery charged – аккумулятор заряжен;

charge-state battery dead – аккумулятор разряжен.

Обратите внимание, что факты, входящие в одну группу (содержат одинаковое первое поле), являются взаимоисключающими, т.е. наличие в системе сразу двух фактов из одной группы лишено смысла.

Из постановки задачи следует, что наша экспертная система должна предоставлять пользователю рекомендации, позволяющие устранить найденную неисправность. Из приведенных выше правил можно выделить следующие рекомендации:

- добавить топливо (правило 5);
- зарядить аккумулятор (правило 6);
- заменить или почистить контакты (правило 7 или 12);
- заменить катушку зажигания или распределительные провода (правило 8);
- прочистить топливную систему (правило 9);
- отрегулировать зазоры между контактами (правило 10); отрегулировать зажигание (правило 11).

Необходимо помнить также о
двух крайних случаях: ремонт не требуется в принципе; экспертная система не смогла поставить диагноз.

Таким образом, получатся следующие рекомендации:

- repair "Add gas";
- repair "Charge the battery";
- repair "Replace the points";
- repair "Clean the points";
- repair "Replace the ignition coil";
- repair "Repair the distributor lead wire";
- repair "Clean the fuel line";
- repair "Point gap adjustment";
- repair "Timing adjustment";
- repair "No repair needed";
- repair "Take your car to a mechanic".

Обратите также внимание, что одни и те же рекомендации могут выводиться как правилом 7, так и правилом 12. Однако состояние машины при этой поломке отличается. Для того, чтобы иметь возможность обрабатывать эту ситуацию с помощью одного правила CLIPS, введем еще два дополнительных факта:

symptom engine low-output – низкая мощность;

symptom engine not-low-output нормальная мощность.

–

Как упоминалось выше, для работы нашей системы можно заставить пользователя вручную вводить факты, описывающие проявление возникшей неисправности. Однако такой метод имеет ряд серьезных недостатков: пользователь может забыть о каких-нибудь существенных деталях или, наоборот, указать слишком много информации, что может помешать нормальной работе системы. Кроме того, факты, описывающие проявление неисправности, должны иметь строго определенный формат, и система не сможет их обработать в случае ошибки со стороны пользователя.

В нашей экспертной системе мы реализуем правила диагностики, которые в зависимости от той или иной ситуации будут задавать пользователю необходимые вопросы и получать ответ в строго заданной форме. Дальнейшая диагностика будет производиться с учетом предыдущих ответов на вопросы, заданные пользователю. Эти ответы будут формировать описание текущей ситуации с помощью фактов, приведенных выше.

Для реализации подобной архитектуры будет необходимо реализовать функцию, задающую пользователю произвольный вопрос и получающую ответ из заданного набора корректных ответов. Далее приведена одна из возможных реализаций такой функции.

```
(deffunction ask-question (?question $?allowed-values)
     (printout t ?question)
     (bind ?answer (read))
     (if (lexemep ?answer)
      then (bind ?answer (lowcase ?answer)))
     (while (not (member ?answer ?allowed-values)) do
      (printout t ?question)
      (bind ?answer (read))
      (if (lexemep ?answer)
       then (bind ?answer (lowcase ?answer))))
     ?answer)
```

Функция принимает два аргумента: простую переменную question, которая содержит текст вопроса, и составную переменную allowed-values с набором допустимых ответов. Сразу после своего вызова функция выводит на экран соответствующий вопрос и читает ответ пользователя в переменную answer. Если переменная answer содержит текст, то она будет принудительно приведена к прописному алфавиту. После этого функция проверяет, является ли полученный ответ одним из заданных корректных ответов. Если нет, то процесс повторится до получения корректного ответа, иначе функция вернет ответ, введенный пользователем.

Будет также очень полезно определить функцию, задающую пользователю вопрос и допускающую ответ в виде да/нет, так как это один из самых распространенных типов вопросов.

```
(deffunction yes-or-no-p (?question)
 (bind ?response (ask-question ?question yes no y n))
 (if (or (eq ?response yes) (eq ?response y))
  then TRUE
  else FALSE))
```

Функция yes-or-no-p вызывает функцию ask-question с постоянным набором допустимых ответов: yes, no, y и n. В случае, если пользователь ввел ответ yes или y, функция возвращает значение TRUE, иначе FALSE. Обратите внимание, что поскольку функция yes-or-no-p использует функцию ask-question, то она должна быть определена после нее.

Для упрощения реализации нашей экспертной системы введем следующее ограничение: за один запуск система может предоставить пользователю одну рекомендацию по исправлению неисправности. В случае, если в машине несколько только неисправностей, то систему нужно будет последовательно вызывать несколько раз, удаляя обнаруженную на каждом новом шаге неисправность. Таким образом, одним из образцов всех диагностических правил будет (not (repair ?)), гарантирующий, что диагноз еще не поставлен.

Первым реализуем правило, определяющее общее состояние двигателя (см. правило 1).

```
(defrule determine-engine-state ""
     (not (working-state engine ?))
     (not (repair ?))
     =>
     (if (yes-or-no-p "Does the engine start (yes/no)? ")
      then
      (if (yes-or-no-p "Does the engine run normally
(yes/no)? ")
       then (assert (working-state engine normal))
       else (assert (working-state engine unsatisfactory)))
      else
      (assert (working-state engine does-not-start))))
```

Условный элемент (not (working-state engine ?)) гарантирует, что общее состояние двигателя еще не определено. Если это так, то пользователю задаются соответствующие вопросы и в систему добавляется факт, описывающий текущее общее состояние двигателя.

Теперь реализуем правило, определяющее, пытается ли двигатель вращаться в случае, если он не заводится.

```
(defrule determine-rotation-state ""
     (working-state engine does-not-start)
     (not (rotation-state engine ?))
     (not (repair ?))
     =>
     (if (yes-or-no-p "Does the engine rotate (yes/no)? ")
      then
      (assert (rotation-state engine rotates))
      (assert (spark-state engine irregular-spark))
      else
      (assert (rotation-state engine does-not-rotate))
      (assert (spark-state engine does-not-spark))))
```

Это правило выполняется в случае, если общее состояние двигателя определено и известно, что он не заводится. Кроме того, условный элемент (not (rotation-state engine ?)) гарантирует, что это правило еще не вызывалось. В зависимости от того или иного ответа пользователя правило добавляет соответствующий набор фактов (см. правило 4).

Далее реализуем довольно простые правила 5 и 6. Выполняемые ими действия вы поймете без дополнительных комментариев.

```
(defrule determine-gas-level ""
     (working-state engine does-not-start)
     (rotation-state engine rotates)
     (not (repair ?))
     =>
     (if (not (yes-or-no-p "Does the tank have any gas in it
(yes/no)? "))
      then
      (assert (repair "Add gas."))))
    (defrule determine-battery-state ""
     (rotation-state engine does-not-rotate)
     (not (charge-state battery ?))
     (not (repair ?))
     =>
     (if (yes-or-no-p "Is the battery charged (yes/no)? ")
      then
      (assert (charge-state battery charged))
      else
      (assert (repair "Charge the battery."))
      (assert (charge-state battery dead))))
```

Обратите внимание, что правило determine-battery-state, помимо определения возможной неисправности, также применяется для добавления в систему факта, описывающего текущее состояние аккумулятора, который может быть использован другими правилами.

При реализации правила 7 необходимо обратить внимание на то, что рекомендации, предоставляемые этим правилом, подходят для двух в корне отличающихся ситуаций. Во-первых, в случае, если двигатель не заводится и существует вероятность плохой искры в системе зажигания (правило 7). Во-вторых, в случае, если двигатель запускается, но не развивает нормальной мощности (правило 12).

Поэтому выполним реализацию этих правил так, как представлено ниже.

```
    (defrule determine-low-output ""
     (working-state engine unsatisfactory)
     (not (symptom engine low-output | not-low-output))
     (not (repair ?))
     =>
     (if (yes-or-no-p "Is the output of the engine low
(yes/no)? ")
      then
      (assert (symptom engine low-output))
      else
      (assert (symptom engine not-low-output))))
     (defrule determine-point-surface-state ""
     (or (and (working-state engine does-not-start)
       (spark-state engine irregular-spark))
      (symptom engine low-output))
     (not (repair ?))
```

```
     =>
     (bind ?response
      (ask-question "What is the surface state of the
points (normal/burned/contaminated)? "
         normal burned contaminated))
     (if (eq ?response burned)
      then
      (assert (repair "Replace the points."))
      else (if (eq ?response contaminated)
        then (assert (repair "Clean the points."))))))
```

Правило determine-low-output определяет, имеет ли место низкая мощность двигателя или нет. Правило determine-point-surface-state адекватно реагирует на условия, заданные в правилах 7 и 12. Обратите внимание на использование условных элементов or и end, которые обеспечивают одинаковое поведение правила в двух абсолютно разных ситуациях. Кроме того, правило determine-point-surface-state отличается от приведенных ранее тем, что непосредственно использует функцию ask-question вместо yes-or-no-p, так как в данный момент пользователю задается вопрос, подразумевающий три варианта ответа.

Реализация оставшихся диагностических правил (8 – 11) также не должна вызвать у вас затруднений.

```
    (defrule determine-conductivity-test ""
     (working-state engine does-not-start)
     (spark-state engine does-not-spark)
     (charge-state battery charged)
     (not (repair ?))
     =>
     (if (yes-or-no-p "Is the conductivity test for the
ignition coil positive (yes/no)? ")
      then
      (assert (repair "Repair the distributor lead wire."))
      else
      (assert (repair "Replace the ignition coil."))))
    (defrule determine-sluggishness ""
     (working-state engine unsatisfactory)
     (not (repair ?))
     =>
```

```
(if (yes-or-no-p "Is the engine sluggish (yes/no)? ")
 then (assert (repair "Clean the fuel line."))))
(defrule determine-misfiring ""
(working-state engine unsatisfactory)
(not (repair ?))
=>
(if (yes-or-no-p "Does the engine misfire (yes/no)? ")
 then
 (assert (repair "Point gap adjustment."))
 (assert (spark-state engine irregular-spark))))
(defrule determine-knocking ""
(working-state engine unsatisfactory)
(not (repair ?))
=>
(if (yes-or-no-p "Does the engine knock (yes/no)? ")
 then
 (assert (repair "Timing adjustment."))))
```

Внимательно взглянув на список правил, мы увидим, что некоторые правила (2, 3 и 13) остались до сих пор нереализованными.

Реализация правила 13 будет следующая.

```
(defrule no-repairs ""
 (declare (salience –10))
 (not (repair ?))
 =>
 (assert (repair "Take your car to a mechanic.")))
```

Обратите внимание на использование приоритета при определении этого правила. Все правила, приведенные в предыдущем разделе, определялись с приоритетом, по умолчанию равным нулю. Использование для правила no-repairs приоритета, равного –10, гарантирует, что правило не будет выполнено, пока в плане решения задачи находится, по крайней мере, одно из диагностических правил. Если все активированные диагностические правила опрошены и ни одно из них не смогло подобрать подходящую рекомендацию по устранению неисправности, то система запустит данное правило, рекомендующее пользователю обратиться к механику.

Реализация правил 2 и 3 приведена ниже.

```
(defrule normal-engine-state-conclusions ""
 (declare (salience 10))
 (working-state engine normal)
 =>
 (assert (repair "No repair needed."))
 (assert (spark-state engine normal))
 (assert (charge-state battery charged))
 (assert (rotation-state engine rotates)))
(defrule unsatisfactory-engine-state-conclusions ""
 (declare (salience 10))
 (working-state engine unsatisfactory)
 =>
 (assert (charge-state battery charged))
 (assert (rotation-state engine rotates)))
```

В этих правилах, наоборот, используется более высокий приоритет, что гарантирует их выполнение до запуска любого диагностирующего правила (в случае выполнения соответствующих условий). Это избавит нашу систему от лишних проверок, а пользователя от лишних вопросов.

Экспертная система фактически готова к работе. Единственное, чего ей не хватает – это метода вывода итоговой информации и правила, сообщающего пользователю о начале работы системы. Ниже приведена реализация этих правил.

```
(defrule system-banner ""
      (declare (salience 10))
      =>
      (printout t crlf crlf)
      (printout t "The Engine Diagnosis Expert System")
      (printout t crlf crlf))
     (defrule print-repair ""
      (declare (salience 10))
      (repair ?item)
      =>
      (printout t crlf crlf)
      (printout t "Suggested Repair:")
      (printout t crlf crlf)
      (format t " %s%n%n%n" ?item))
```

Теперь для того, чтобы запустить экспертную систему, достаточно выполнить команду reset, которая добавит факт initial-fact, необходимый для правила system-banner, и команду run. После этого вы сразу увидите сообщение "The Engine Diag-nosis Expert System", которое означает, что система начала работать, и получите серию вопросов, ответы на которые помогут **экспертной системе оценить состояние вашей машины и подобрать соответствующую рекомендацию по ремонту.**