МЕТОДИ І СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ

3 курс, весна 2021

- Доц. Баклан І.В.
- Email: iaa@ukr.net
- Web: baklaniv.at.ua

Лекція 14

CLIPS.

Приклад створення експертної системи

Розробка прототипу ЕКСПЕРТНОЇ СИСТЕМИ AUTO

Розглянемо приклад створення діагностичної експертної системи, яка дозволяє встановити причину несправності автомобіля і видати відповідну рекомендацію.

Розробку будь експертної системи слід починати з виявлення **основних сутностей**, що мають значення при вирішенні конкретної задачі і **законів**, швидше за все емпіричних, діючих над цими сутностями.

В результаті роботи з експертом були встановлені наступні емпіричні правила:

1. Двигун зазвичай знаходиться в одному з 3 станів: він може працювати нормально, не завжди працює коректно або не заводиться.

2. Якщо двигун працює нормально, то це означає, що він нормально обертається, система запалювання і акумулятор знаходяться в нормі і ніякого ремонту не потрібно.

3. Якщо двигун запускається, але працює ненормально, то це говорить, по крайней мере, про те, що акумулятор в порядку.

4. Якщо двигун не запускається, то потрібно дізнатися, чи намагається він обертатися. Якщо двигун обертається, але при цьому не заводиться, то це може говорити про наявність поганий іскри в системі запалювання. Якщо двигун навіть не намагається заводитися, то це говорить про те, що іскри немає в принципі.

5. Якщо двигун не заводиться, але обертається, потрібно перевірити наявність палива. Якщо палива немає - то, скоріше за все, для ремонту машини потрібно просто заправитися.

6. Якщо двигун не заводиться, потрібно також перевірити, чи заряджений акумулятор, якщо немає, то його слід зарядити.

7. Якщо двигун не заводиться і існує ймовірність поганий іскри в системі запалювання, то необхідно перевірити контакти. Контакти можуть бути в одному з трьох станів - чисті, обпалені і брудні, в разі обпалених контактів їх необхідно замінити, в разі якщо контакти брудні, їх досить просто почистити.

8. Якщо двигун не заводиться, іскри немає і акумулятор заряджений, то потрібно перевірити котушку запалювання на електричну провідність. У разі, якщо струм не проходить через котушку, то її необхідно замінити. Якщо котушка запалювання в порядку, значить необхідно замінити розподільні дроти.

9. Якщо двигун запускається, але при цьому веде себе інертно, не відразу реагує на подачу палива, то необхідно прочистити паливну систему.

10. Якщо двигун запускається, але відбуваються перебої з запалюванням, то це говорить про наявність поганий іскри в системі запалювання, для усунення даної несправності необхідно відрегулювати зазори між контактами.

11. Якщо двигун запускається і стукає, то необхідно відрегулювати запалювання.

12. Якщо двигун запускається, але не розвиває нормальної потужності, то це може говорити про обпалених або забруднених контактах (див. Правило 7).

13. Можливі ситуації, коли стан двигуна не можна описати наведеними вище факторами і машині може знадобитися більш детальний аналіз стану.

З наведених вище правил можна виділити наступні сутності, що мають значення при вирішенні задачі.

-По-перше, для вирішення завдання експертної системі необхідно знати, в якому стані знаходиться машина, діагностика якої проводиться. Експерт виділив три можливих стани: нормальна робота двигуна, двигун працює незадовільно, не заводиться (див. Правило 1).

-По-друге, більшість наведених правил, крім стану двигуна в цілому, використовують поняття стану обертання двигуна. Згідно з цими правилами двигун може перебувати в одному з двох станів, які визначаються в залежності від того, здатний він обертатися (працювати) чи ні.

-По-третє, в деяких правилах (див. Правила 4, 7, 8, 10) використовується поняття стану системи запалювання. Система запалювання може бути в одному з трьох станів: нормальний стан, нерегулярна робота і неробочий стан.

-В-четвертих, в правилах 6 і 8 використовується поняття стану акумулятора. Акумулятор може бути в одному з двох станів: зарядженим і вирядженим.

Таким чином, можна виділити наступні факти, що описують стан автомобіля і його вузлів:

Група фактів, що описує стан машини:

working-state engine normal – нормальна робота;

working-state engine unsatisfactory – незадовільна робота;

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

Група фактів, що описує стан двигуна:

rotation-state engine rotates – двигун обертається;

rotation-state engines does-not-rotate

– двигун не обертається.

Група фактів, що описує стан системи запалювання:

spark-state engine normal – запалювання в порядку;

spark-state engine irregular-spark – іскра нерегулярна;

spark-state engine does-not-spark – іскри немає.

Група фактів, що описує стан системи харчування:

charge-state battery charged – акумулятор заряджений;

charge-state battery dead – акумулятор розряжений.

Зверніть увагу, що факти, що входять в одну групу (містять однакову перше поле), є взаємовиключними, тобто наявність в системі відразу двох фактів з однієї групи позбавлене сенсу.

З постановки задачі випливає, що наша експертна система повинна надавати користувачеві рекомендації, що дозволяють усунути знайдену несправність. З наведених вище правил можна виділити наступні рекомендації:додати паливо (правило 5);

- зарядити акумулятор (правило 6);
- замінити або почистити контакти (правило 7 або 12);
- замінити котушку запалювання або розподільні дроти (правило 8);
- прочистити паливну систему (правило 9);

- відрегулювати зазори між контактами (правило 10);
- відрегулювати запалювання (правило 11).

Необхідно пам'ятати також про двох крайніх випадках: ремонт не потрібно в принципі; експертна система не змогла поставити діагноз.

Таким чином, вийдуть наступні рекомендації:

- 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".

Зверніть також увагу, що одні й ті ж рекомендації можуть виводитися як правилом 7, так і правилом 12. Однак стан машини при цій поломці відрізняється. Для того, щоб мати можливість обробляти цю ситуацію за допомогою одного правила CLIPS, введемо ще два додаткових факту:

symptom engine low-output – низька потужність;

symptom engine not-low-output нормальна потужність.

Як згадувалося вище, для роботи нашої системи можна змусити користувача вручну вводити факти, що описують прояв виниклої несправності. Однак такий метод має ряд серйозних недоліків: користувач може забути про будь-які суттєві деталі або, навпаки, вказати занадто багато інформації, що може перешкодити нормальній роботі системи. Крім того, факти, що описують прояв несправності, повинні мати строго певний формат, і система не зможе їх обробити в разі помилки з боку користувача.

В нашій експертній системі ми реалізуємо правила діагностики, які в залежності від тієї чи іншої ситуації будуть задавати користувачеві необхідні питання і отримувати відповідь в строго заданій формі. Подальша діагностика буде проводитися з урахуванням попередніх відповідей на питання, задані користувачеві. Ці відповіді будуть формувати опис поточної ситуації за допомогою фактів, наведених вище.

Для реалізації подібної архітектури буде необхідно реалізувати функцію, задану користувачеві довільний питання і одержує відповідь із заданого набору коректних відповідей. Далі наведена одна з можливих реалізацій такої функції.

```
(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)
```

Функція приймає два аргументи: просту змінну **`question`**, яка містить текст питання, і складову змінну **`allowed-values`** з набором допустимих відповідей. Відразу після свого виклику функція виводить на екран відповідне питання і читає відповідь користувача в змінну **`answer`**. Якщо змінна **`answer`** містить текст, то вона буде примусово приведена до прописного алфавіту.

Після цього функція перевіряє, чи є отриманий відповідь одним із заданих коректних відповідей. Якщо немає, то процес повториться до отримання коректної відповіді, інакше функція поверне відповідь, яку Ви самі ввели.

Буде також дуже корисно визначити функцію, задану користувачеві питання і допускає відповідь у вигляді так / ні, так як це один з найпоширеніших типів питань.

```
(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))
```

Функція **`yes-or-no-p`** викликає функцію **`ask-question`** з постійним набором допустимих відповідей: **`yes`**, **`no`**, **`y`** і **`n`**. У разі, якщо користувач ввів відповідь **`yes`** або **`y`**, функція повертає значення **`TRUE`**, інакше **`FALSE`**. Зверніть увагу, що оскільки функція **`yes-or-no-p`** використовує функцію **`ask-question`**, то вона повинна бути визначена після неї.

Для спрощення реалізації нашої експертної системи введемо наступне обмеження: за один запуск система може надати користувачеві одну рекомендацію щодо виправлення несправності. У разі, якщо в машині кілька тільки несправностей, то систему потрібно буде послідовно викликати кілька разів, видаляючи виявлену на кожному новому кроці несправність. Таким чином, одним із зразків всіх діагностичних правил буде `(not (repair?))`, який гарантує те, що діагноз ще не поставлений.

Першим реалізуємо правило, що визначає загальний стан двигуна (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))))
```

Умовний елемент `(not (working-state engine?))` гарантує, що загальний стан двигуна ще не визначено. Якщо це так, то користувачеві задаються відповідні питання і в систему додається факт, що описує поточний загальний стан двигуна.

Тепер реалізуємо правило, що визначає, чи намагається двигун обертатися в разі, якщо він не заводиться.

```
(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))))
```

Це правило виконується в разі, якщо загальний стан двигуна визначено і відомо, що він не заводиться. Крім того, умовний елемент `(not (rotation-state engine?))` гарантує, що це правило ще не викликалося. Залежно від того чи іншого відповіді користувача правило додає відповідний набір фактів (див. Правило 4).

Далі реалізуємо досить прості правила 5 і 6. Про виконувані ними дії ви зрозумієте без додаткових коментарів.

```
(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))))
```

Зверніть увагу, що правило `determine-battery-state`, крім визначення можливої несправності, також застосовується для додавання в систему факту, що описує поточний стан акумулятора, який може бути використаний іншими правилами.

При реалізації правила 7 необхідно звернути увагу на те, що рекомендації, надані цим правилом, підходять для двох в корені відрізняються ситуацій. По-перше, в разі, якщо двигун не заводиться і існує ймовірність поганий іскри в системі запалювання (правило 7). По-друге, в разі, якщо двигун запускається, але не розвиває нормальної потужності (правило 12).

Тому виконаємо реалізацію цих правил так, як представлено нижче.

```
(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`** визначає, чи має місце низька потужність двигуна чи ні. Правило **`determine-point-surface-state`** адекватно реагує на умови, задані в правилах 7 і 12. Зверніть увагу на використання умовних елементів **`OR`** і **`end`**, які забезпечують однакову поведінку правила в двох абсолютно різних ситуаціях.

Крім того, правило **`determine-point-surface-state`** відрізняється від наведених раніше тим, що безпосередньо використовує функцію **`ask-question`** замість **`yes-or-no-p`**, так як в даний момент користувачеві задається питання, що має на увазі три варіанти відповіді.

Реалізація решти діагностичних правил (8 - 11) також не повинна викликати у вас труднощів.

```
(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."))))
```

Уважно глянувши на список правил, ми побачимо, що деякі правила (2, 3 і 13) залишилися досі нереалізованими.

Реалізація правила 13 буде наступна.

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

Зверніть увагу на використання пріоритету при визначенні цього правила. Всі правила, наведені в попередньому розділі, визначалися з пріоритетом, за замовчуванням рівним нулю. Використання для правила `no-repairs` пріоритету, рівного -10, гарантує, що правило не буде виконано, поки в плані вирішення завдання знаходиться, по крайней мере, одне з діагностичних правил.

Якщо все активовані діагностичні правила опитано і жодне з них не змогло підібрати підходячу рекомендацію щодо усунення несправності, то система запустить дане правило, що рекомендує користувачеві звернутися до механіка.

Реалізація правил 2 і 3 наведена нижче.

```
    (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)))
```

У цих правилах, навпаки, використовується більш високий пріоритет, що гарантує їх виконання до запуску будь-якого діагностуючого правила (в разі виконання відповідних умов). Це позбавить нашу систему від зайвих перевірок, а користувача від зайвих питань.

Експертна система фактично готова до роботи. Єдине, чого їй не вистачає - це методу виведення підсумкової інформації і правила, який повідомляє користувачеві про початок роботи системи. Нижче приведена реалізація цих 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))
```

Тепер для того, щоб запустити експертну систему, досить виконати команду **`reset`**, яка додасть факт **`initial-fact`**, необхідний для правила **`system-banner`**, і команду **`run`**. Після цього ви відразу побачите повідомлення **`"The Engine Diag-nosis Expert System"`**, яке означає, що система почала працювати, і отримаєте серію питань, відповіді на які допоможуть експертній системі оцінити стан вашої машини і підібрати відповідну рекомендацію щодо ремонту.

У наступній лекції ми ознайомимся з прикладами розв’язання логічних задач за допомогою CLIPS.