Assignment 1 & CLIPS

Junli Tao

jtao076@aucklanduni.ac.nz

Rm 321-723, Tamaki Campus

(Wed. 1:00pm-2:00pm)

Assignment 1

- Modeling knowledge by using decision tree
  - Find a SIMPLE problem

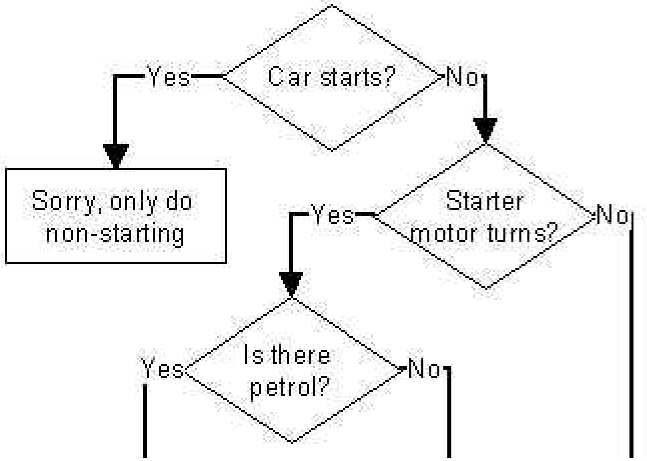
    (e.g. “I can’t connect to Internet” or “choosing a mobile phone”)

- Implementing the modeled knowledge (decision tree) in CLIPS
  - It is OK to reuse or modify an example CLIP code, but remember to cite the source code.

    (e.g. “adapted from auto.clp” or “ modified from example.clp”)

Assignment 1

- Decision tree
  - decision node : a symptom needed to be diagnosed (root node, parent/child node)
  - consequence node : a suggestion outcome (leaf node)

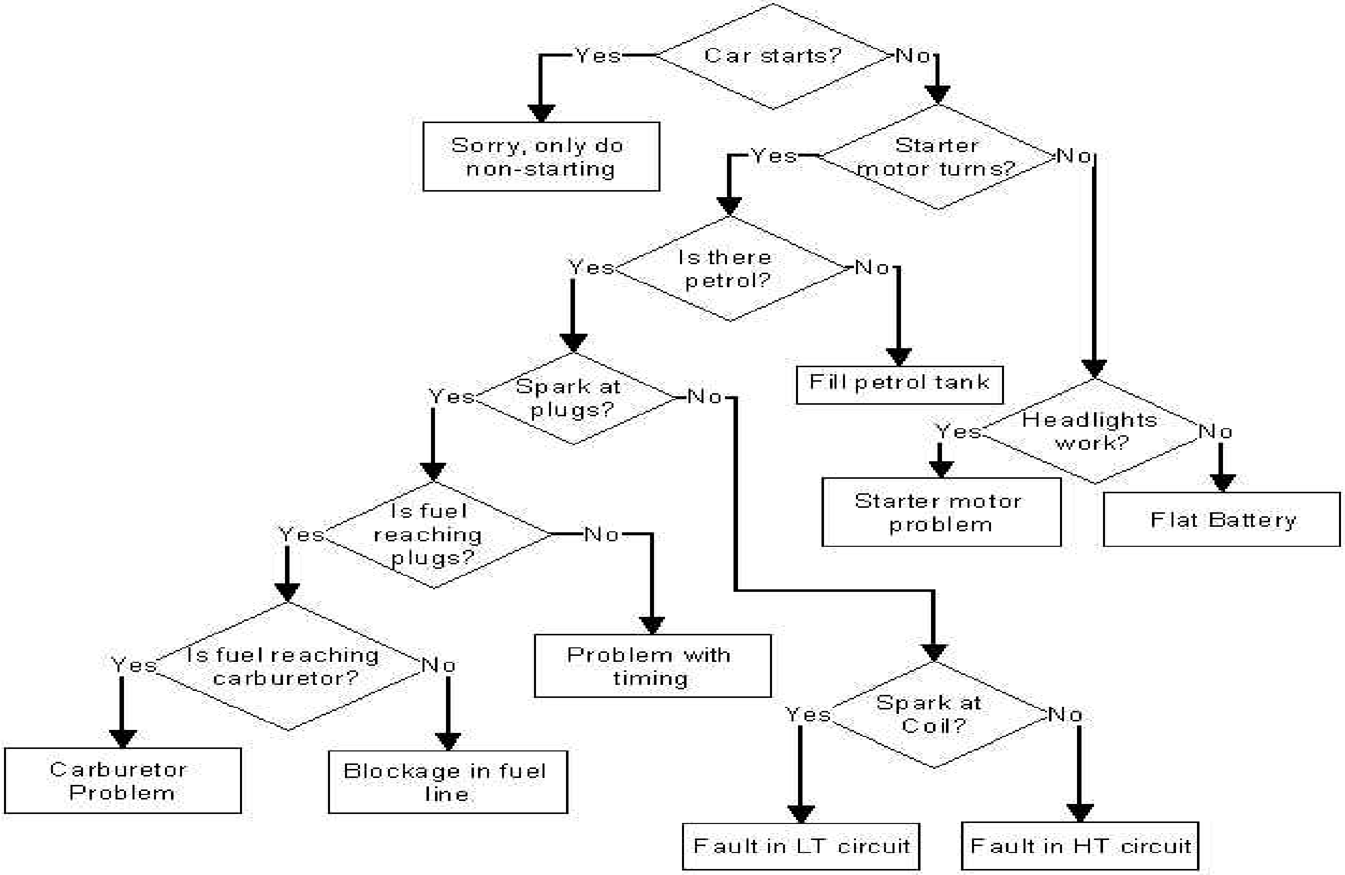


Assignment 1

- **Procedure building decision tree:**
  - Decide the perspective from which the diagnosis should be conducted.
  - What is the fundamental symptom (root node) causing the problem from the decided perspective.
  - Define its constraints (e.g. yes/no) and analysis its all possible children nodes
  - Add *children nodes of the fundamental symptom into decision tree.*
    - *- probable symptoms (decision nodes)*
    - *- conclusions (consequence nodes)*
  - If any of children node is decision node, define its constraints and add its children nodes into decision tree.
  - Repeat step 5 until all leaf nodes are consequence nodes.

Assignment 1

- Decision tree example



CLIPS

- Obtain CLIPS
  - Download CLIPS from

    http://clipsrules.sourceforge.net/

  - For Windows users,
    - download “windows_executables_624.zip”
    - decompose the zip file, click “CLIPWin.exe”

CLIPS

- **Three basic components**
  - **Fact-list**: the data on which inferences are derived
  - **Knowledge base**: all the rules
  - **Inference engine**: control overall execution of rules

CLIPS

- **Facts**
  - add facts to fact-list with **assert**

    (assert (Brian duck)), (assert (duck Brian)),

    (assert (a) (b) (c)),

    (assert (hunter-game duck Brian))
  - see facts in fact-list with **facts**

    (facts 1), (facts 0 1)
  - retract facts from fact-list with **retract**

    *(retract 2), (retract *)*

CLIPS

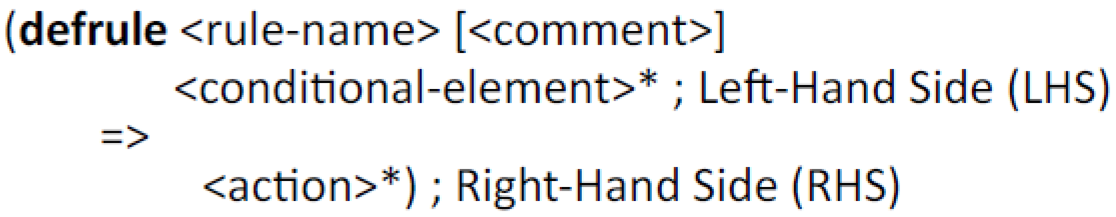
- Types of atoms
  - Symbols: *duck, duck1, d!#^*
  - String: *"duck soup is good!!!“*
  - Integer: *(assert (number 1))*
  - Float: *(assert (distance 3.5e5))*
  - *…*

(assert (The duck says "Quack."))

(assert (The-duck-says "Quack."))

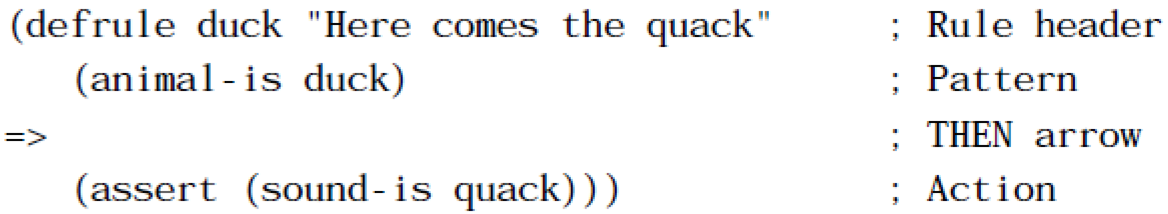
CLIPS

- Define rules (knowledge)



```
(defrule <rule-name> [<comment>]
            <conditional-element>* ; Left-Hand Side (LHS)
      =>
              <action>*) ; Right-Hand Side (RHS)
```

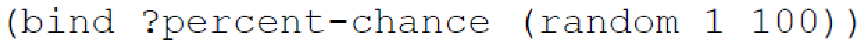
e.g



```
(defrule duck "Here comes the quack"        ; Rule header
   (animal-is duck)                         ; Pattern
=>                                          ; THEN arrow
   (assert (sound-is quack)))               ; Action
```

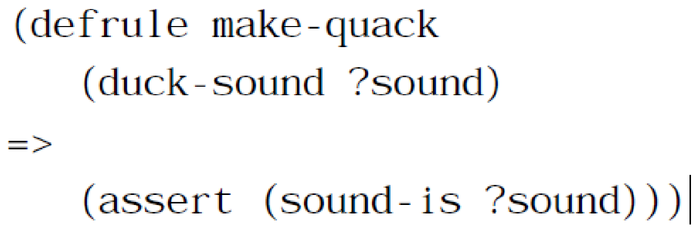
CLIPS

- Variables
  - general format: *?<variable-name>*
    - Explicit binding



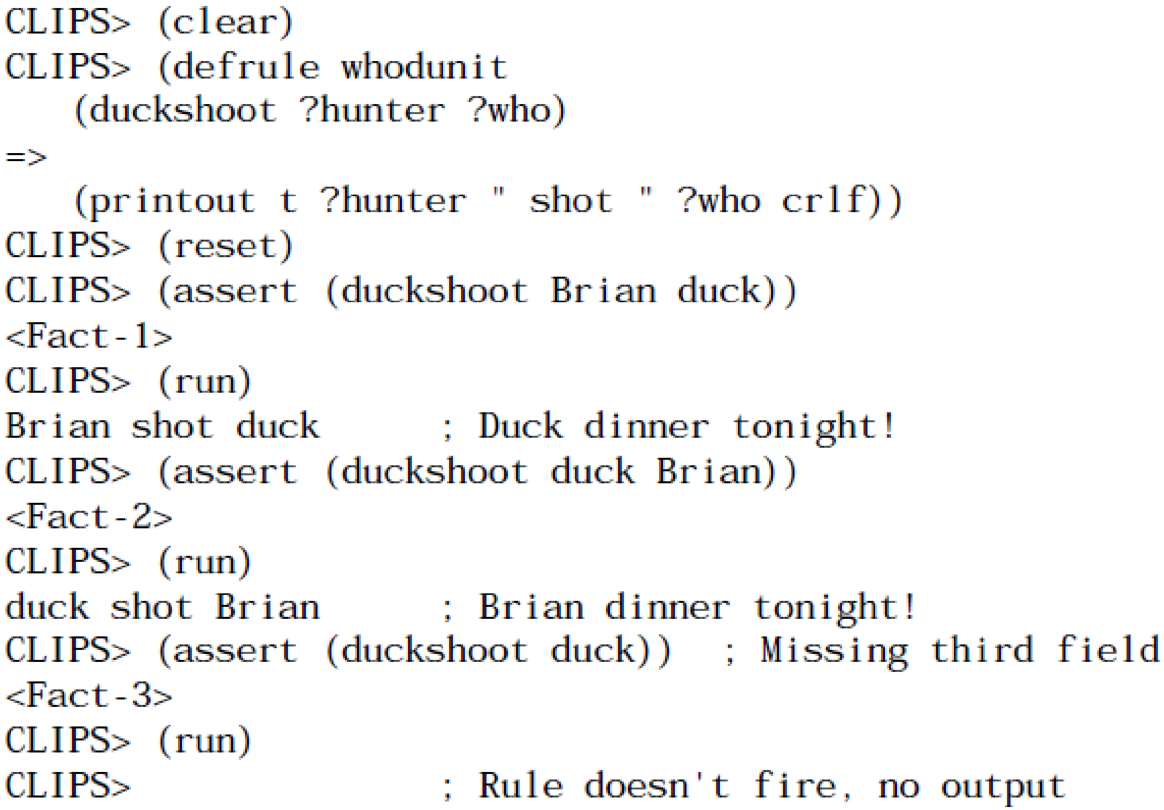
```
(bind ?percent-chance (random 1 100))
```

    - Implicit binding



```
(defrule make-quack
   (duck-sound ?sound)
=>
   (assert (sound-is ?sound)))
```

CLIPS



```
CLIPS> (clear)
CLIPS> (defrule whodunit
   (duckshoot ?hunter ?who)
=>
   (printout t ?hunter " shot " ?who crlf))
CLIPS> (reset)
CLIPS> (assert (duckshoot Brian duck))
<Fact-1>
CLIPS> (run)
Brian shot duck     ; Duck dinner tonight!
CLIPS> (assert (duckshoot duck Brian))
<Fact-2>
CLIPS> (run)
duck shot Brian     ; Brian dinner tonight!
CLIPS> (assert (duckshoot duck))  ; Missing third field
<Fact-3>
CLIPS> (run)
CLIPS>              ; Rule doesn't fire, no output
```