МЕТОДИ І СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ

3 курс, весна 2021

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

Лекція 17

Приклад складної експертної системи та бази знань на CLIPS

Пропонується студентам ознайомитися з приведеними текстами програм реалізації експертної системи на основі бази знань в CLIPS та запустити їх.

Нижче приведена експертна система, яка намагається ідентифікувати тварину на основі її характеристик. База знань у цьому прикладі - це сукупність фактів, що представляють правила зворотного ланцюга. Потім правила прямого ланцюжка CLIPS використовуються для імітації механізму зворотного виведення ланцюгів назад.

```
;;;======================================================
;;;   Animal Identification Expert System
;;;
;;;     A simple expert system which attempts to identify
;;;     an animal based on its characteristics.
;;;     The knowledge base in this example is a
;;;     collection of facts which represent backward
;;;     chaining rules. CLIPS forward chaining rules are
;;;     then used to simulate a backward chaining inference
;;;     engine.
;;;
;;;     CLIPS Version 6.3 Example
;;;
;;;     To execute, merely load, reset, and run.
;;;     Answer questions yes or no.
;;;==========================================================

(defmodule MAIN (export ?ALL))

(defmodule VALIDATE (import MAIN ?ALL))

(defmodule CHAIN (import MAIN ?ALL))

(defmodule ASK (import MAIN ?ALL))

;;;*************************
;;;* DEFGLOBAL DEFINITIONS *
;;;*************************

(defglobal MAIN
   ?*rule-index* = 1
   ?*validate* = TRUE)

;;;***************************
;;;* DEFFUNCTION DEFINITIONS *
;;;***************************
```

```
(deffunction generate-rule-name ()
   (bind ?name (sym-cat rule- ?*rule-index*))
   (bind ?*rule-index* (+ ?*rule-index* 1))
   (return ?name))

;;;***************************
;;;* DEFTEMPLATE DEFINITIONS *
;;;***************************

(deftemplate MAIN::rule
   (slot name (default-dynamic (generate-rule-name)))
   (slot validate (default no))
   (multislot if)
   (multislot then)
   (multislot processed))

(deftemplate MAIN::question
   (multislot valid-answers)
   (slot variable)
   (slot query))
```

```
(deftemplate MAIN::answer
   (slot variable)
   (slot prefix (default ""))
   (slot postfix (default "")))

(deftemplate MAIN::goal
   (slot variable))

(deftemplate MAIN::variable
   (slot name)
   (slot value))

(deftemplate MAIN::activity)

(deftemplate MAIN::legalanswers
   (multislot values))

;;;**************************
;;;* INFERENCE ENGINE RULES *
;;;**************************

(defrule MAIN::startup
   =>
   (if ?*validate*
      then
      (focus VALIDATE CHAIN ASK)
      else
      (focus CHAIN ASK)))

(defrule MAIN::continue
   (declare (salience -10))
   ?f <- (activity)
   =>
   (retract ?f)
   (focus CHAIN ASK))

(defrule MAIN::goal-satified ""
   (goal (variable ?goal))
   (variable (name ?goal) (value ?value))
   (answer (prefix ?prefix) (postfix ?postfix) (variable ?goal))
   =>
   (format t "%s%s%s%n" ?prefix ?value ?postfix))

;;; #################
;;; CHAIN MODULE RULES
;;; #################

(defrule CHAIN::propagate-goal ""
   (logical (goal (variable ?goal))
            (rule (if ?variable $?)
                  (then ?goal ? ?value)))
   =>
   (assert (goal (variable ?variable))))

(defrule CHAIN::modify-rule-match-is ""
   (variable (name ?variable) (value ?value))
   ?f <- (rule (if ?variable is ?value and $?rest)
               (processed $?p))
   =>
   (modify ?f (if ?rest)
              (processed ?p ?variable is ?value and)))

(defrule CHAIN::rule-satisfied-is ""
   (variable (name ?variable) (value ?value))
   ?f <- (rule (if ?variable is ?value)
               (then ?goal ? ?goal-value)
```

```
                (processed $?p))
   =>
   (modify ?f (if)
               (processed ?p ?variable is ?value #)))

(defrule CHAIN::apply-rule ""
   (logical (rule (if)
                  (then ?goal ? ?goal-value)))
   =>
   (assert (variable (name ?goal) (value ?goal-value))))

;;; ################
;;; ASK MODULE RULES
;;; ################

(defrule ASK::ask-question-no-legalvalues ""
   (not (legalanswers))
   ?f1 <- (goal (variable ?variable))
   (question (variable ?variable) (query ?text))
   (not (variable (name ?variable)))
   =>
   (assert (activity))
   (retract ?f1)
   (format t "%s " ?text)
   (assert (variable (name ?variable) (value (read)))))

(defrule ASK::ask-question-legalvalues ""
   (legalanswers (values $?answers))
   ?f1 <- (goal (variable ?variable))
   (question (variable ?variable) (query ?text))
   (not (variable (name ?variable)))
   =>
   (assert (activity))
   (retract ?f1)
   (format t "%s " ?text)
   (printout t ?answers " ")
   (bind ?reply (read))
   (if (lexemep ?reply)
      then
      (bind ?reply (lowcase ?reply)))
   (if (member ?reply ?answers)
     then (assert (variable (name ?variable) (value ?reply)))
     else (assert (goal (variable ?variable)))))
```

```
;;; ####################
;;; VALIDATE MODULE RULES
;;; ####################

(defrule VALIDATE::copy-rule
   (declare (salience 10))
   ?f <- (rule (validate no))
   =>
   (duplicate ?f (validate yes))
   (modify ?f (validate done)))

(defrule VALIDATE::next-condition
   (declare (salience -10))
   ?f <- (rule (name ?name) (validate yes)
               (if ?a ?c ?v and $?rest))
   =>
   (modify ?f (if ?rest)))

(defrule VALIDATE::validation-complete
   (declare (salience -10))
   ?f <- (rule (validate yes) (if ? ? ?))
   =>
   (retract ?f))

;;; *******************
;;; Validation - Syntax
;;; *******************

(defrule VALIDATE::and-connector
   ?f <- (rule (name ?name) (validate yes)
               (if ?a ?c ?v ?connector&~and $?))
   =>
   (retract ?f)
   (printout t "In rule " ?name ", if conditions must be connected
using and:" crlf
               "    " ?a " " ?c " " ?v " *" ?connector "*" crlf))

(defrule VALIDATE::and-requires-additional-condition
   ?f <- (rule (name ?name) (validate yes)
               (if ?a ?c ?v and))
   =>
   (retract ?f)
   (printout t "In rule " ?name ", an additional condition should
follow the final and:" crlf
```

```
                 "    " ?a " " ?c " " ?v " and <missing condition>"
crlf))

(defrule VALIDATE::incorrect-number-of-then-terms
   ?f <- (rule (name ?name) (validate yes)
               (then $?terms&:(<> (length$ ?terms) 3)))
   =>
   (retract ?f)
   (printout t "In rule " ?name ", then portion should be of the
form <variable> is <value>:" crlf
                 "    " (implode$ ?terms) crlf))

(defrule VALIDATE::incorrect-number-of-if-terms
   ?f <- (rule (name ?name) (validate yes)
               (if $?terms&:(< (length$ ?terms) 3)))
   =>
   (retract ?f)
   (printout t "In rule " ?name ", if portion contains an
incomplete condition:" crlf
                 "    " (implode$ ?terms) crlf))

(defrule VALIDATE::incorrect-then-term-syntax
   ?f <- (rule (name ?name) (validate yes)
               (then ?a ?c&~is ?v))
   =>
   (retract ?f)
   (printout t "In rule " ?name ", then portion should be of the
form <variable> is <value>:" crlf
               "   " ?a " " ?c " " ?v " " crlf))

(defrule VALIDATE::incorrect-if-term-syntax
   ?f <- (rule (name ?name) (validate yes)
               (if ?a ?c&~is ?v $?))
   =>
   (retract ?f)
   (printout t "In rule " ?name ", if portion comparator should
be \"is\"" crlf
               "   " ?a " " ?c " " ?v " " crlf))

(defrule VALIDATE::illegal-variable-value
   ?f <- (rule (name ?name) (validate yes)
               (if ?a ?c ?v $?))
   (question (variable ?a) (valid-answers))
   (legalanswers (values $?values))
   (test (not (member$ ?v ?values)))
   =>
   (retract ?f)
   (printout t "In rule " ?name ", the value " ?v " is not legal
for variable " ?a ":" crlf
               "    " ?a " " ?c " " ?v crlf))

(defrule VALIDATE::reachable
   (rule (name ?name) (validate yes)
         (if ?a ?c ?v $?))
   (not (question (variable ?a)))
   (not (rule (then ?a $?)))
   =>
   (printout t "In rule " ?name " no question or rule could be
found "
               "that can supply a value for the variable " ?a ":"
crlf
               "    " ?a " " ?c " " ?v crlf))

(defrule VALIDATE::used "TBD lower salience"
   ?f <- (rule (name ?name) (validate yes)
               (then ?a is ?v))
   (not (goal (variable ?a)))
   (not (rule (if ?a ? ?v $?)))
   =>
   (retract ?f)
   (printout t "In rule " ?name " the conclusion for variable " ?a
               " is neither referenced by any rules nor the primary
goal" crlf
               "   " ?a " is " ?v crlf))

(defrule VALIDATE::variable-in-both-if-and-then
   ?f <- (rule (name ?name) (validate yes)
               (if ?a $?)
               (then ?a is ?v))
   =>
   (retract ?f)
   (printout t "In rule " ?name " the variable " ?a
               " is used in both the if and then sections" crlf))

(defrule VALIDATE::question-variable-unreferenced
   (question (variable ?a) (query ?q))
   (not (rule (validate done) (if $? ?a is ?v $?)))
   =>
   (printout t "The question \"" ?q "\", assigns a value to the
variable " ?a
               " which is not referenced by any rules" crlf))

;;;**************************
;;;* DEFFACTS KNOWLEDGE BASE *
;;;**************************

(deffacts MAIN::knowledge-base
   (goal (variable type.animal))
   (legalanswers (values yes no))
   (rule (if backbone is yes)
         (then superphylum is backbone))
   (rule (if backbone is no)
         (then superphylum is jellyback))
   (question (variable backbone)
             (query "Does your animal have a backbone?"))
   (rule (if superphylum is backbone and
          warm.blooded is yes)
         (then phylum is warm))
   (rule (if superphylum is backbone and
          warm.blooded is no)
         (then phylum is cold))
   (question (variable warm.blooded)
             (query "Is the animal warm blooded?"))
   (rule (if superphylum is jellyback and
          live.prime.in.soil is yes)
         (then phylum is soil))
   (rule (if superphylum is jellyback and
          live.prime.in.soil is no)
         (then phylum is elsewhere))
   (question (variable live.prime.in.soil)
             (query "Does your animal live primarily in soil?"))
   (rule (if phylum is warm and
          has.breasts is yes)
         (then class is breasts))
   (rule (if phylum is warm and
          has.breasts is no)
         (then type.animal is bird/penguin))
   (question (variable has.breasts)
             (query "Normally, does the female of your animal nurse
its young with milk?"))
   (rule (if phylum is cold and
          always.in.water is yes)
```

```
      (then class is water))
(rule (if phylum is cold and
       always.in.water is no)
      (then class is dry))
(question (variable always.in.water)
          (query "Is your animal always in water?"))
(rule (if phylum is soil and
       flat.bodied is yes)
      (then type.animal is flatworm))
(rule (if phylum is soil and
       flat.bodied is no)
      (then type.animal is worm/leech))
(question (variable flat.bodied)
          (query "Does your animal have a flat body?"))
(rule (if phylum is elsewhere and
       body.in.segments is yes)
      (then class is segments))
(rule (if phylum is elsewhere and
       body.in.segments is no)
      (then class is unified))
(question (variable body.in.segments)
          (query "Is the animals body in segments?"))
```

```
(rule (if class is breasts and
       can.eat.meat is yes)
      (then order is meat))
(rule (if class is breasts and
       can.eat.meat is no)
      (then order is vegy))
(question (variable can.eat.meat)
          (query "Does your animal eat red meat?"))
(rule (if class is water and
       boney is yes)
      (then type.animal is fish))
(rule (if class is water and
       boney is no)
      (then type.animal is shark/ray))
(question (variable boney)
          (query "Does your animal have a boney skeleton?"))
(rule (if class is dry and
       scaly is yes)
      (then order is scales))
(rule (if class is dry and
       scaly is no)
      (then order is soft))
```

```
   (question (variable scaly)
             (query "Is your animal covered with scaled skin?"))
   (rule (if class is segments and
          shell is yes)
         (then order is shell))
   (rule (if class is segments and
          shell is no)
         (then type.animal is centipede/millipede/insect))
   (question (variable shell)
             (query "Does your animal have a shell?"))
   (rule (if class is unified and
          digest.cells is yes)
         (then order is cells))
   (rule (if class is unified and
          digest.cells is no)
         (then order is stomach))
   (question (variable digest.cells)
             (query "Does your animal use many cells to digest its
food instead of a stomach?"))
   (rule (if order is meat and
          fly is yes)
         (then type.animal is bat))
```

```
(rule (if order is meat and
       fly is no)
      (then family is nowings))
(question (variable fly)
          (query "Can your animal fly?"))
(rule (if order is vegy and
       hooves is yes)
      (then family is hooves))
(rule (if order is vegy and
       hooves is no)
      (then family is feet))
(question (variable hooves)
          (query "Does your animal have hooves?"))
(rule (if order is scales and
       rounded.shell is yes)
      (then type.animal is turtle))
(rule (if order is scales and
       rounded.shell is no)
      (then family is noshell))
(question (variable rounded.shell)
          (query "Does the animal have a rounded shell?"))
(rule (if order is soft and
      jump is yes)
     (then type.animal is frog))
(rule (if order is soft and
      jump is no)
     (then type.animal is salamander))
(question (variable jump)
          (query "Does your animal jump?"))
(rule (if order is shell and
      tail is yes)
     (then type.animal is lobster))
(rule (if order is shell and
      tail is no)
     (then type.animal is crab))
(question (variable tail)
          (query "Does your animal have a tail?"))
(rule (if order is cells and
      stationary is yes)
     (then family is stationary))
(rule (if order is cells and
      stationary is no)
     (then type.animal is jellyfish))
(question (variable stationary)
             (query "Is your animal attached permanently to an
object?"))
   (rule (if order is stomach and
          multicelled is yes)
         (then family is multicelled))
   (rule (if order is stomach and
          multicelled is no)
         (then type.animal is protozoa))
   (question (variable multicelled)
             (query "Is your animal made up of more than one
cell?"))
   (rule (if family is nowings and
          opposing.thumb is yes)
         (then genus is thumb))
   (rule (if family is nowings and
          opposing.thumb is no)
         (then genus is nothumb))
   (question (variable opposing.thumb)
             (query "Does your animal have an opposing thumb?"))
   (rule (if family is hooves and
          two.toes is yes)
         (then genus is twotoes))
   (rule (if family is hooves and
          two.toes is no)
         (then genus is onetoe))
   (question (variable two.toes)
             (query "Does your animal stand on two toes/hooves per
foot?"))
   (rule (if family is feet and
          live.in.water is yes)
         (then genus is water))
   (rule (if family is feet and
          live.in.water is no)
         (then genus is dry))
   (question (variable live.in.water)
             (query "Does your animal live in water?"))
   (rule (if family is noshell and
          limbs is yes)
         (then type.animal is crocodile/alligator))
   (rule (if family is noshell and
          limbs is no)
         (then type.animal is snake))
   (question (variable limbs)
             (query "Does your animal have limbs?"))
   (rule (if family is stationary and
          spikes is yes)
         (then type.animal is sea.anemone))
   (rule (if family is stationary and
          spikes is no)
         (then type.animal is coral/sponge))
   (question (variable spikes)
             (query "Does your animal normally have spikes
radiating from its body?"))
   (rule (if family is multicelled and
          spiral.shell is yes)
         (then type.animal is snail))
   (rule (if family is multicelled and
          spiral.shell is no)
         (then genus is noshell))
   (question (variable spiral.shell)
             (query "Does your animal have a spiral-shaped
shell?"))
   (rule (if genus is thumb and
          prehensile.tail is yes)
         (then type.animal is monkey))
   (rule (if genus is thumb and
         prehensile.tail is no)
        (then species is notail))
   (question (variable prehensile.tail)
             (query "Does your animal have a prehensile tail?"))
   (rule (if genus is nothumb and
         over.400 is yes)
        (then species is 400))
   (rule (if genus is nothumb and
         over.400 is no)
        (then species is under400))
   (question (variable over.400)
             (query "Does an adult normally weigh over 400
pounds?"))
   (rule (if genus is twotoes and
         horns is yes)
        (then species is horns))
   (rule (if genus is twotoes and
         horns is no)
        (then species is nohorns))
   (question (variable horns)
             (query "Does your animal have horns?"))
   (rule (if genus is onetoe and
          plating is yes)
         (then type.animal is rhinoceros))
   (rule (if genus is onetoe and
          plating is no)
         (then type.animal is horse/zebra))
   (question (variable plating)
             (query "Is your animal covered with a protective
plating?"))
   (rule (if genus is water and
          hunted is yes)
         (then type.animal is whale))
   (rule (if genus is water and
          hunted is no)
         (then type.animal is dolphin/porpoise))
   (question (variable hunted)
             (query "Is your animal, unfortunately, commercially
hunted?"))
   (rule (if genus is dry and
          front.teeth is yes)
         (then species is teeth))
   (rule (if genus is dry and
          front.teeth is no)
```

```
      (then species is noteeth))
   (question (variable front.teeth)
          (query "Does your animal have large front teeth?"))
   (rule (if genus is noshell and
          bivalve is yes)
         (then type.animal is clam/oyster))
   (rule (if genus is noshell and
          bivalve is no)
         (then type.animal is squid/octopus))
   (question (variable bivalve)
          (query "Is your animal protected by two half-
shells?"))
   (rule (if species is notail and
          nearly.hairless is yes)
         (then type.animal is man))
   (rule (if species is notail and
          nearly.hairless is no)
         (then subspecies is hair))
   (question (variable nearly.hairless)
          (query "Is your animal nearly hairless?"))
   (rule (if species is 400 and
          land.based is yes)
         (then type.animal is bear/tiger/lion))
   (rule (if species is 400 and
          land.based is no)
         (then type.animal is walrus))
   (question (variable land.based)
             (query "Is your animal land based?"))
   (rule (if species is under400 and
          thintail is yes)
         (then type.animal is cat))
   (rule (if species is under400 and
          thintail is no)
         (then type.animal is coyote/wolf/fox/dog))
   (question (variable thintail)
             (query "Does your animal have a thin tail?"))
   (rule (if species is nohorns and
          lives.in.desert is yes)
         (then type.animal is camel))
   (rule (if species is nohorns and
          lives.in.desert is no and
          semi.aquatic is no)
         (then type.animal is giraffe))
   (rule (if species is nohorns and
          lives.in.desert is no and
          semi.aquatic is yes)
         (then type.animal is hippopotamus))
   (question (variable lives.in.desert)
             (query "Does your animal normally live in the
desert?"))
   (question (variable semi.aquatic)
             (query "Is your animal semi-aquatic?"))
   (rule (if species is teeth and
          large.ears is yes)
         (then type.animal is rabbit))
   (rule (if species is teeth and
          large.ears is no)
         (then type.animal is rat/mouse/squirrel/beaver/porcupine))
   (question (variable large.ears)
             (query "Does your animal have large ears?"))
   (rule (if species is noteeth and
          pouch is yes)
         (then type.animal is "kangaroo/koala bear"))
   (rule (if species is noteeth and
          pouch is no)
         (then type.animal is mole/shrew/elephant))
```

```
(question (variable pouch)
          (query "Does your animal have a pouch?"))
(rule (if subspecies is hair and
       long.powerful.arms is yes)
      (then type.animal is orangutan/gorilla/chimpanzie))
(rule (if subspecies is hair and
       long.powerful.arms is no)
      (then type.animal is baboon))
(question (variable long.powerful.arms)
          (query "Does your animal have long, powerful arms?"))
(rule (if species is horns and
       fleece is yes)
      (then type.animal is sheep/goat))
(rule (if species is horns and
       fleece is no)
      (then subsubspecies is nofleece))
(question (variable fleece)
          (query "Does your animal have fleece?"))
(rule (if subsubspecies is nofleece and
       domesticated is yes)
      (then type.animal is cow))
(rule (if subsubspecies is nofleece and
             domesticated is no)
            (then type.animal is deer/moose/antelope))
    (question (variable domesticated)
                (query "Is your animal domesticated?"))
    (answer (prefix "I think your animal is a ") (variable
type.animal) (postfix "."))) 
```