ІНТЕЛЕКТУАЛЬНІ ІНФОРМАЦІЙНІ СИСТЕМИ

Лекція 3.

Проектування простих баз знань

1. Приклад простої бази знань

```
; This is a very simple example of a CLIPS knowledge base,
; just using the pattern matching to create new knowledge.

; To run this example:
; 1) start CLIPSWin from Explorer
; 2) load the KB with (load "emh1.txt")
; 3) initialize with (reset)
; 4) run the KB with (run)
; 5) to view the facts generated, enter (facts)
; The current fact is for a patient with symptoms of measles.
; To change this, edit the deffacts at the end of this file.

; Before we can define rules and facts, we have to define the
; "variables" used with deftemplate, which defines the concept
; name, as well as the names of all attributes which could be
; given values.

; A "Patient" may have values for fever, spots, rash, sore_throat
; and innoculated.

(deftemplate Patient
   (slot fever)
```

```
   (slot spots)
   (slot rash)
   (slot sore_throat)
   (slot innoculated))

; Thes deftemplates represent conclusions which we may assign
; values to as a result of the inference.

(deftemplate Diagnosis
   (slot diagnosis))

(deftemplate Treatment
   (slot treatment))

; In its simplest form, a rule just has a right side which is a
; "template" that the inference engine will try to match to some
; fact. In this case, it matches a Patient with specific values
; for fever, spots, and innoculated (the other values don't matter).

; The left side of the => are actions to take if the RHS matches a
; fact. In this case, we assert a new fact (Diagnosis (diagnosis
measles))
; and printout the string "Measles diagnosed" to the terminal (t),
; followed by a return (crlf).
```

```
(defrule Measles
   (Patient (fever high) (spots yes) (innoculated no))
   =>
   (assert (Diagnosis (diagnosis measles)))
   (printout t "Measles diagnosed" crlf))

; We can also combine template matches, using the standard connectives
; of and, or, not. Note that the syntax of CLIPS is prefix-oriented.

(defrule Allergy1
   (and (Patient (spots yes))
        (or (not (Patient (fever high)))
            (Patient (innoculated yes))))
   =>
   (assert (Diagnosis (diagnosis allergy)))
   (printout t "Allergy diagnosed from spots and lack of measles"
crlf))

(defrule Allergy2
   (Patient (rash yes))
   =>
   (assert (Diagnosis (diagnosis allergy)))
   (printout t "Allergy diagnosed from rash" crlf))
```

```
(defrule Flu
   (Patient (sore_throat yes) (fever mild | high))
   =>
   (assert (Diagnosis (diagnosis flu)))
   (printout t "Flu diagnosed" crlf))

; Rules for recommedaing treatments on the basis of
; Diagnosis facts created.

(defrule Penicillin
   (Diagnosis (diagnosis measles))
   =>
   (assert (Treatment (treatment pennicillin)))
   (printout t "Penicillin prescribed" crlf))

(defrule Allergy_pills
   (Diagnosis (diagnosis allergy))
   =>
   (assert (Treatment (treatment allergy_shot)))
   (printout t "Allergy shot prescribed" crlf))

(defrule Bed_rest
   (Diagnosis (diagnosis flu))
```

```
   =>
   (assert (Treatment (treatment bed_rest)))
   (printout t "Bed rest prescribed" crlf))

; Facts are created with deffacts (the can also
; be directly asserted while in CLIPS). The list
; consists of a name, and a list of facts.

(deffacts Symptoms
   (Patient (fever low)
            (spots yes)
            (rash no)
            (sore_throat no)


            (innoculated no)))
```

2. База знань з використанням властивості salience

```
; This is a very simple example of a CLIPS knowledge base,
; just using the pattern matching to create new knowledge.

; We have modified this to use salience measure to order
; rules (necessary for the spots --> allergy rule).

; A "Patient" may have values for fever, spots, rash, sore_throat
; and innoculated.

(deftemplate Patient
   (slot fever)
   (slot spots)
   (slot rash)
   (slot sore_throat)
   (slot innoculated))

; Thes deftemplates represent conclusions which we may assign
; values to as a result of the inference.

(deftemplate Diagnosis
   (slot diagnosis))
```

```
(deftemplate Treatment
   (slot treatment))

; Rules for determining diagnosis on the basis of patient symptoms

; Salience added to give this rule priority

(defrule Measles
   (declare (salience 100))
   (Patient (fever high) (spots yes) (innoculated no))
   =>
   (assert (Diagnosis (diagnosis measles)))
   (printout t "Measles diagnosed" crlf))

; Modified to only fire if no measles

(defrule Allergy1
   (and (Patient (spots yes))
        (not (Diagnosis (diagnosis measles))))
   =>
   (assert (Diagnosis (diagnosis allergy)))
   (printout t "Allergy diagnosed from spots and lack of measles"
crlf))
```

```
(defrule Allergy2
   (Patient (rash yes))
   =>
   (assert (Diagnosis (diagnosis allergy)))
   (printout t "Allergy diagnosed from rash" crlf))

(defrule Flu
   (Patient (sore_throat yes) (fever mild | high))
   =>
   (assert (Diagnosis (diagnosis flu)))
   (printout t "Flu diagnosed" crlf))

; Rules for recommedaing treatments on the basis of
; Diagnosis facts created.

(defrule Penicillin
   (Diagnosis (diagnosis measles))
   =>
   (assert (Treatment (treatment pennicillin)))
   (printout t "Penicillin prescribed" crlf))

(defrule Allergy_pills
   (Diagnosis (diagnosis allergy))
```

```
   =>
   (assert (Treatment (treatment allergy_shot)))
   (printout t "Allergy shot prescribed" crlf))

(defrule Bed_rest
   (Diagnosis (diagnosis flu))
   =>
   (assert (Treatment (treatment bed_rest)))
   (printout t "Bed rest prescribed" crlf))

; Facts are created with deffacts (the can also
; be directly asserted while in CLIPS). The list
; consists of a name, and a list of facts.

(deffacts Symptoms
   (Patient (fever high)
            (spots yes)
            (rash no)
            (sore_throat no)
            (innoculated no)))
```

3. Приклад бази знань з використанням порівняння за зразком

```
; This is a very simple example of a CLIPS knowledge base,
; just using the pattern matching to create new knowledge.

; We have modified it to bind facts to patients with a given
; name (a new property of patient).

(deftemplate Patient
   (slot name)
   (slot fever)
   (slot spots)
   (slot rash)
   (slot sore_throat)
   (slot innoculated))

; Thes deftemplates represent conclusions which we may assign
; values to as a result of the inference.

(deftemplate Diagnosis
   (slot name)
   (slot diagnosis))

(deftemplate Treatment
```

```
   (slot name)
   (slot treatment))

; Rules for determining diagnosis on the basis of patient symptoms

; In the following, ?n is a variable, which gets "bound" to the value
; corresponding to the attribute (in this case the "name" attribute)
; in the matching fact. That value then gets used on the rest of the
; rule, including the assert and the printout.

(defrule Measles
   (declare (salience 100))
   (Patient (name ?n) (fever high) (spots yes) (innoculated no))
   =>
   (assert (Diagnosis (name ?n) (diagnosis measles)))
   (printout t "Measles diagnosed for " ?n crlf))

; Modified to only fire if no measles

(defrule Allergy1
   (and (Patient (name ?n) (spots yes))
        (not (Diagnosis (name ?n) (diagnosis measles))))
   =>
   (assert (Diagnosis (name ?n) (diagnosis allergy)))
```

```
   (printout t "Allergy diagnosed for " ?n " from spots and lack of
measles" crlf))

(defrule Allergy2
   (Patient (name ?n) (rash yes))
   =>
   (assert (Diagnosis (name ?n) (diagnosis allergy)))
   (printout t "Allergy diagnosed from rash for" ?n  crlf))

(defrule Flu
   (Patient (name ?n) (sore_throat yes) (fever mild | high))
   =>
   (assert (Diagnosis (name ?n) (diagnosis flu)))
   (printout t "Flu diagnosed for " ?n crlf))

; Rules for recommedaing treatments on the basis of
; Diagnosis facts created.

(defrule Penicillin
   (Diagnosis (name ?n) (diagnosis measles))
   =>
   (assert (Treatment (name ?n) (treatment penicillin)))
   (printout t "Penicillin prescribed for " ?n crlf))
```

```
(defrule Allergy_pills
   (Diagnosis (name ?n) (diagnosis allergy))
   =>
   (assert (Treatment (name ?n) (treatment allergy_shot)))
   (printout t "Allergy shot prescribed for " ?n crlf))

(defrule Bed_rest
   (Diagnosis (name ?n) (diagnosis flu))
   =>
   (assert (Treatment (name ?n) (treatment bed_rest)))
   (printout t "Bed rest prescribed for " ?n crlf))

; Facts are created with deffacts (the can also
; be directly asserted while in CLIPS). The list
; consists of a name, and a list of facts.

(deffacts Symptoms
   (Patient (name "Fred")
            (fever high)
            (spots yes)
            (rash no)
            (sore_throat no)
            (innoculated no))
   (Patient (name "Barney")
```

```
          (fever mild)
          (spots yes)
          (rash no)
          (sore_throat no)
          (innoculated no))
(Patient (name "Wilma")
          (fever high)
          (spots no)
          (rash no)
          (sore_throat yes)
          (innoculated no)))
```

4. Приклад бази знань з використанням чисельних виразів

```
; This is a very simple example of a CLIPS knowledge base,
; just using the pattern matching to create new knowledge.

; We have modified this to use binding and arithmetic operations
; to conclude the level of fever from temperature.

; A "Patient" may have values for temperature, spots, rash,
sore_throat
; and innoculated.

(deftemplate Patient
   (slot temperature)
   (slot spots)
   (slot rash)
   (slot sore_throat)
   (slot innoculated))
```

```
; Thes deftemplates represent conclusions which we may assign
; values to as a result of the inference.

(deftemplate Diagnosis
   (slot diagnosis))

(deftemplate Treatment
   (slot treatment))

; We add another template for fever, which is now a conclusion
; instead of a patient property

(deftemplate Fever
   (slot level))

; Rules for concluding fever from temperature.
```

```
; Note that these rules find the patient temperature, and then bind
; it to ?t. The next part uses the test keyword to evaluate the
; conditional expression as true or false.

(defrule Fever1
   (Patient (temperature ?t))
   (test (>= ?t 101))
   =>
   (assert (Fever (level high)))
   (printout t "High fever diagnosed" crlf))

(defrule Fever2
   (Patient (temperature ?t))
   (test (and (< ?t 101) (> ?t 98.6)))
   =>
   (assert (Fever (level mild)))
   (printout t "Mild fever diagnosed" crlf))
```

```
; Rules for determining diagnosis on the basis of patient symptoms
; Salience added to give this rule priority

(defrule Measles
   (declare (salience 100))
   (Patient (spots yes) (innoculated no))
   (Fever (level high))
   =>
   (assert (Diagnosis (diagnosis measles)))
   (printout t "Measles diagnosed" crlf))

; Modified to only fire if no measles

(defrule Allergy1
   (and (Patient (spots yes))
        (not (Diagnosis (diagnosis measles))))
   =>
   (assert (Diagnosis (diagnosis allergy)))
   (printout t "Allergy diagnosed from spots and lack of measles"
crlf))

(defrule Allergy2
   (Patient (rash yes))
   =>
```

```
   (assert (Diagnosis (diagnosis allergy)))
   (printout t "Allergy diagnosed from rash" crlf))

(defrule Flu
   (Patient (sore_throat yes))
   (Fever (level mild|high))
   =>
   (assert (Diagnosis (diagnosis flu)))
   (printout t "Flu diagnosed" crlf))

; Rules for recommedaing treatments on the basis of
; Diagnosis facts created.

(defrule Penicillin
   (Diagnosis (diagnosis measles))
   =>
   (assert (Treatment (treatment pennicillin)))
   (printout t "Penicillin prescribed" crlf))

(defrule Allergy_pills
   (Diagnosis (diagnosis allergy))
   =>
   (assert (Treatment (treatment allergy_shot)))
   (printout t "Allergy shot prescribed" crlf))
```

```
(defrule Bed_rest
   (Diagnosis (diagnosis flu))
   =>
   (assert (Treatment (treatment bed_rest)))
   (printout t "Bed rest prescribed" crlf))

; Facts are created with deffacts (the can also
; be directly asserted while in CLIPS). The list
; consists of a name, and a list of facts.

(deffacts Symptoms
   (Patient (temperature 99)
            (spots no)
            (rash no)
            (sore_throat yes)
            (innoculated no)))
```

5. Приклад бази знань із введенням повідомлень

```
; This is a very simple example of a CLIPS knowledge base,
; just using the pattern matching to create new knowledge.

; We have modified this to use binding to prompt users for input.

; To simplify this, we will no longer use a single "Patient"
; object, but instead separate objects for each symptom (note
; that we will only perform inference on one patient at a time).

(deftemplate Temperature (slot temperature))
(deftemplate Spots (slot spots))
(deftemplate Rash (slot rash))
(deftemplate SoreThroat (slot sore_throat))
(deftemplate Innoculated (slot innoculated))
```

```
; Thes deftemplates represent conclusions which we may assign
; values to as a result of the inference.

(deftemplate Diagnosis
   (slot diagnosis))

(deftemplate Treatment
   (slot treatment))

; We add another template for fever, which is now a conclusion
; instead of a patient property

(deftemplate Fever
   (slot level))
```

```
; Our first rules will be used to gather symptoms from the user.
; Note that there are no conditions, which meand that they will
; always fire. The action is to print a prompt, bind the (read)
; to a variable, and then assert a new fact using that value.

(defrule GetTemperature
   =>
   (printout t "Enter patient temperature: ")
   (bind ?response (read))
   (assert (Temperature (temperature ?response))))

(defrule GetSpots
   =>
   (printout t "Does the patient have spots (yes or no): ")
   (bind ?response (read))
   (assert (Spots (spots ?response))))

(defrule GetRash
   =>
   (printout t "Does the patient have a rash (yes or no): ")
   (bind ?response (read))
   (assert (Rash (rash ?response))))

(defrule GetSoreThroat
```

```
   =>
   (printout t "Does the patient have a sore throat (yes or no): ")
   (bind ?response (read))
   (assert (SoreThroat (sore_throat ?response))))

; We can also ask for certain information only if necessary. For
example,
; it doesn't make sense to ask whether the patient has been
innoculated
; unless there is a possiblity of measles.

(defrule GetInnoculated
   (Fever (level high))
   (Spots (spots yes))
   =>
   (printout t "Has the patient been innoculated for measles (yes or
no): ")
   (bind ?response (read))
   (assert (Innoculated (innoculated ?response))))

; Rules for concluding fever from temperature.
```

```
; Note that these rules find the patient temperature, and then bind
; it to ?t. The next part uses the test keyword to evaluate the
; conditional expression as true or false.

(defrule Fever1
   (Temperature (temperature ?t))
   (test (>= ?t 101))
   =>
   (assert (Fever (level high)))
   (printout t "High fever diagnosed" crlf))

(defrule Fever2
   (Temperature (temperature ?t))
   (test (and (< ?t 101) (> ?t 98.6)))
   =>
   (assert (Fever (level mild)))
   (printout t "Mild fever diagnosed" crlf))
```

```
; Rules for determining diagnosis on the basis of patient symptoms
; Salience added to give this rule priority

(defrule Measles
   (declare (salience 100))
   (Spots (spots yes))
   (Innoculated (innoculated no))
   (Fever (level high))
   =>
   (assert (Diagnosis (diagnosis measles)))
   (printout t "Measles diagnosed" crlf))

; Modified to only fire if no measles

(defrule Allergy1
   (declare (salience -100))
   (and (Spots (spots yes))
        (not (Diagnosis (diagnosis measles))))
   =>
   (assert (Diagnosis (diagnosis allergy)))
   (printout t "Allergy diagnosed from spots and lack of measles"
crlf))

(defrule Allergy2
```

```
   (Rash (rash yes))
   =>
   (assert (Diagnosis (diagnosis allergy)))
   (printout t "Allergy diagnosed from rash" crlf))

(defrule Flu
   (SoreThroat (sore_throat yes))
   (Fever (level mild|high))
   =>
   (assert (Diagnosis (diagnosis flu)))
   (printout t "Flu diagnosed" crlf))
```

```
; Rules for recommedaing treatments on the basis of
; Diagnosis facts created.

(defrule Penicillin
   (Diagnosis (diagnosis measles))
   =>
   (assert (Treatment (treatment pennicillin)))
   (printout t "Penicillin prescribed" crlf))

(defrule Allergy_pills
   (Diagnosis (diagnosis allergy))
   =>
   (assert (Treatment (treatment allergy_shot)))
   (printout t "Allergy shot prescribed" crlf))

(defrule Bed_rest
   (Diagnosis (diagnosis flu))
   =>
   (assert (Treatment (treatment bed_rest)))
   (printout t "Bed rest prescribed" crlf))

; Facts are created with deffacts (the can also
; be directly asserted while in CLIPS). The list
; consists of a name, and a list of facts.
```

6. Приклад бази знань з простими фактами

```
; This is a very simple example of a CLIPS knowledge base,
; just using the pattern matching to create new knowledge.

; We have modified this to use binding to prompt users for input.

; To simplify this, we will no longer use a single "Patient"
; object, but instead separate objects for each symptom (note
; that we will only perform inference on one patient at a time).

; TO simplify this even further, since there are no longer any
; entities, we can just assert facts of the form (attribute value),
; which means that we will no longer need the deftemplates.


; Our first rules will be used to gather symptoms from the user.
; Note that there are no conditions, which meand that they will
; always fire. The action is to print a prompt, bind the (read)
; to a variable, and then assert a new fact using that value.
```

```
(defrule GetTemperature
   =>
   (printout t "Enter patient temperature: ")
   (bind ?response (read))
   (assert (temperature ?response)))

(defrule GetSpots
   =>
   (printout t "Does the patient have spots (yes or no): ")
   (bind ?response (read))
   (assert (spots ?response)))

(defrule GetRash
   =>
   (printout t "Does the patient have a rash (yes or no): ")
   (bind ?response (read))
   (assert (rash ?response)))

(defrule GetSoreThroat
   =>
   (printout t "Does the patient have a sore throat (yes or no): ")
   (bind ?response (read))
   (assert (sore_throat ?response)))
```

```
; We can also ask for certain information only if necessary. For
example,
; it doesn't make sense to ask whether the patient has been
innoculated
; unless there is a possiblity of measles.

(defrule GetInnoculated
   (fever high)
   (spots yes)
   =>
   (printout t "Has the patient been innoculated for measles (yes or
no): ")
   (bind ?response (read))
   (assert (innoculated ?response)))

; Rules for concluding fever from temperature.

; Note that these rules find the patient temperature, and then bind
; it to ?t. The next part uses the test keyword to evaluate the
; conditional expression as true or false.

(defrule Fever1
   (temperature ?t)
   (test (>= ?t 101))
```

```
   =>
   (assert (fever high))
   (printout t "High fever diagnosed" crlf))

(defrule Fever2
   (temperature ?t)
   (test (and (< ?t 101) (> ?t 98.6)))
   =>
   (assert (fever mild))
   (printout t "Mild fever diagnosed" crlf))

; Rules for determining diagnosis on the basis of patient symptoms
; Salience added to give this rule priority

(defrule Measles
   (declare (salience 100))
   (spots yes)
   (innoculated no)
   (fever high)
   =>
   (assert (diagnosis measles))
   (printout t "Measles diagnosed" crlf))

; Modified to only fire if no measles
```

```
(defrule Allergy1
   (declare (salience -100))
   (and (spots yes)
        (not (diagnosis measles)))
   =>
   (assert (diagnosis allergy))
   (printout t "Allergy diagnosed from spots and lack of measles"
crlf))

(defrule Allergy2
   (rash yes)
   =>
   (assert (diagnosis allergy))
   (printout t "Allergy diagnosed from rash" crlf))

(defrule Flu
   (sore_throat yes)
```