Revision of Jess (1)

CS3025, Knowledge-Based Systems
Lecture 20

Yuting Zhao
Yuting.zhao@gmail.com

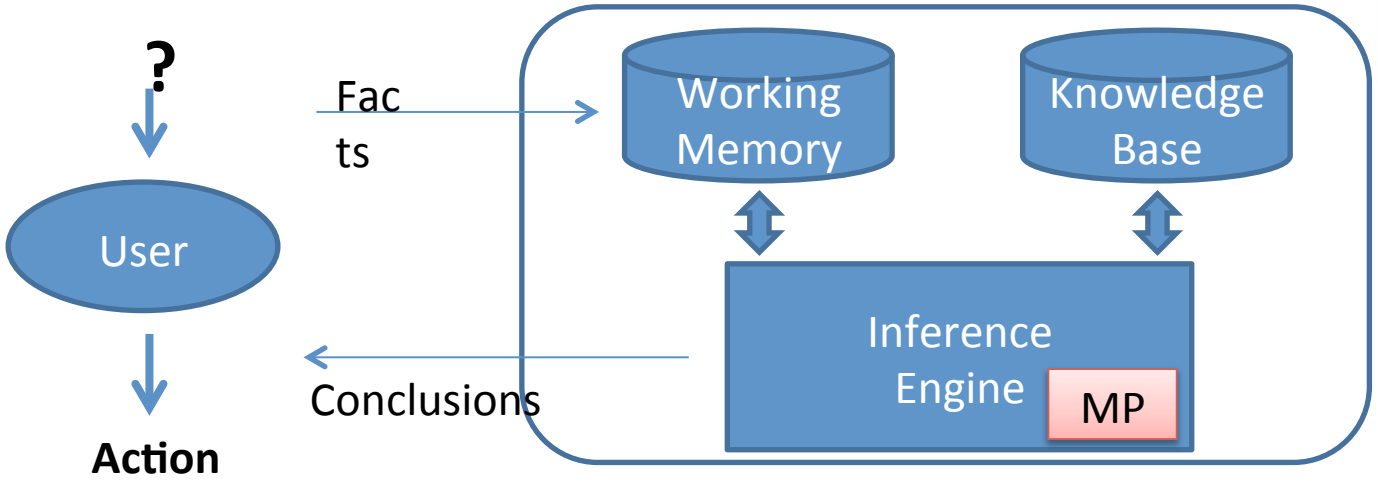
2017-11-23

What have we learned?

- Questions:
  - What is a Knowledge-Based System or Expert System?
  - What are the main components of a Knowledge-based System? What is a Knowledge base?
  - How do rules of a knowledge base influence each other during their execution?
  - How does forward chaining work?

Knowledge-Based Systems

- What is a KB-System:
  - A software system that uses inference to draw conclusions from a symbolic representation of knowledge
  - The knowledge represented is central
- Also called an Expert System



- fact
- rule
- engine (Modus Ponens)

Ordered Facts

- Simple, so-called “ordered” facts in Jess are formulated in the following way:

(<field> <field> <field> …)

  - Is enclosed in round brackets … “(“ “)”
  - The first field must be a symbol
  - Fields must be constants (symbols, strings, integer, float)
  - Ordering of fields is important
- For example:
  - `(creature body-covering is hair)`
  - `(creature eats grass)`
  - `(creature colour is black and white)`
- Two different ordered facts:
  - **(person Michael male 35 Diana Philip Julia)**
  - **(person Michael 35 male Diana Philip Julia)**
- Asserting such a fact into working memory:

```
(deffacts more-creatures
          (body-covering Sh
          (eats Shere-Khan
          (colour-of Shere-
          (marking-of Shere
          (body-covering Pe
          (motion Pengi swi
          (motion Pengi wad
          (colour-of Pengi
```

```
Jess> (assert (person Michael male 35 Diana Philip Julia))
<fact-0>
```

“Unordered” Facts based on Templates

- Jess templates are used to specify structured data that can be used as facts:

```
(deftemplate person
     (slot name)
     (slot age)
     (slot gender)
     (slot partner)
     (multislot children))
```

```
(assert (person
                    (name Michael)
                    (gender male)
                    (age 35)
                    (partner Diana)
                    (children Philip
Julia)))
<fact-0>
```

How many facts defined here??? 5? or 6? or 1?

Specifying Rules

IF [**LHS** ANTECEDENT] **then** [**RHS** CONSEQUENT]

IF "the traffic light shows red"
THEN "wait 1min",
"switch on yellow light"

IF "the traffic light shows red
AND yellow"
THEN "wait 3seconds",
"switch off red light",
"switch off yellow light",
"switch on green light"

LHS?
RHS?

chunk

Rules: General Form

IF $(a_{11} \wedge a_{12} \wedge \ldots \wedge a_{1b}) \vee$

$(a_{21} \wedge a_{22} \wedge \ldots \wedge a_{2c}) \vee$

$\ldots$

$(a_{d1} \wedge a_{d2} \wedge \ldots \wedge a_{de})$

THEN $c_1 \wedge c_2 \wedge \ldots \wedge c_m$

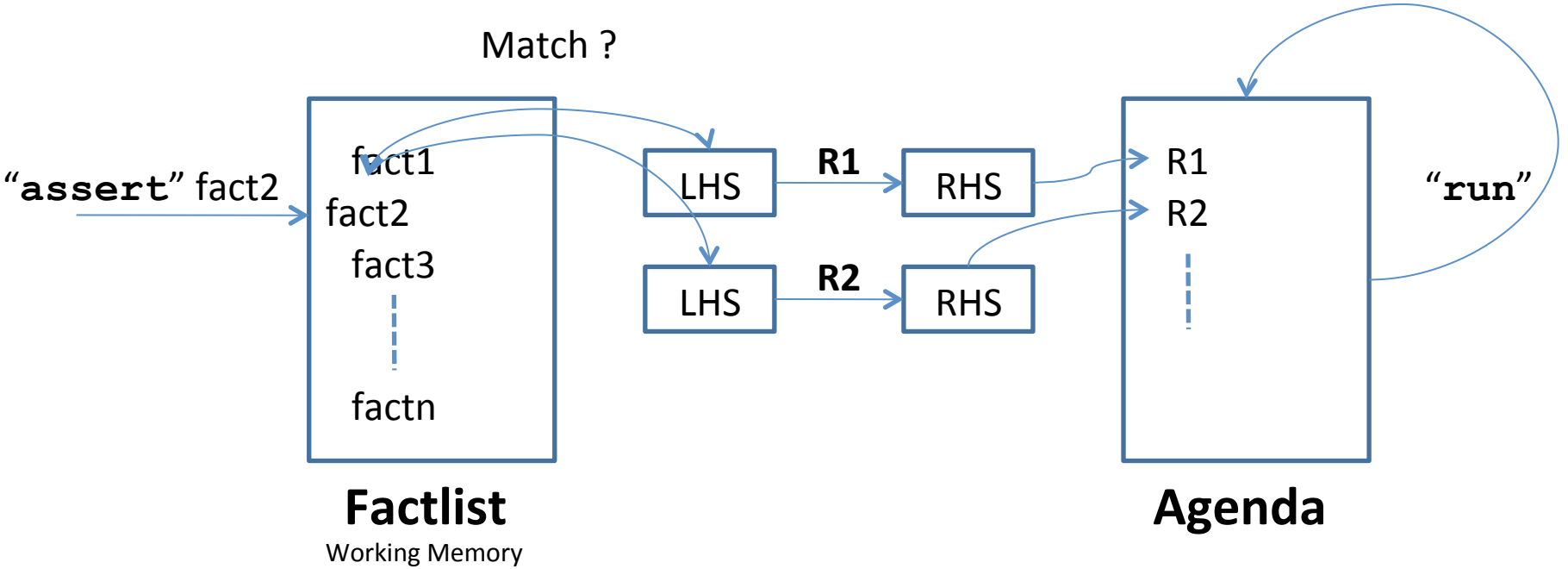
- Constructing the Antecedent
  - use logical operators to combine propositions in antecedent and consequent of a rule
    - and ($\wedge$)
    - or ($\vee$)
    - not ($\neg$)
- Constructing the Consequent
  - Specify a sequence of actions

- We can now write really complicated and difficult to understand rules!
- *Keep each rule as simple as possible.*

Executing Jess Expert Systems

- Questions:
  - Jess maintains a Working Memory and an Agenda. How do they influence and determine the execution of Jess programs?
  - What determines the sequence of actions taking place when Jess is executing a rule base?

Jess Execution Revisited



- How can we influence the ordering of rule activations on the Agenda?

Rule Activation and Execution

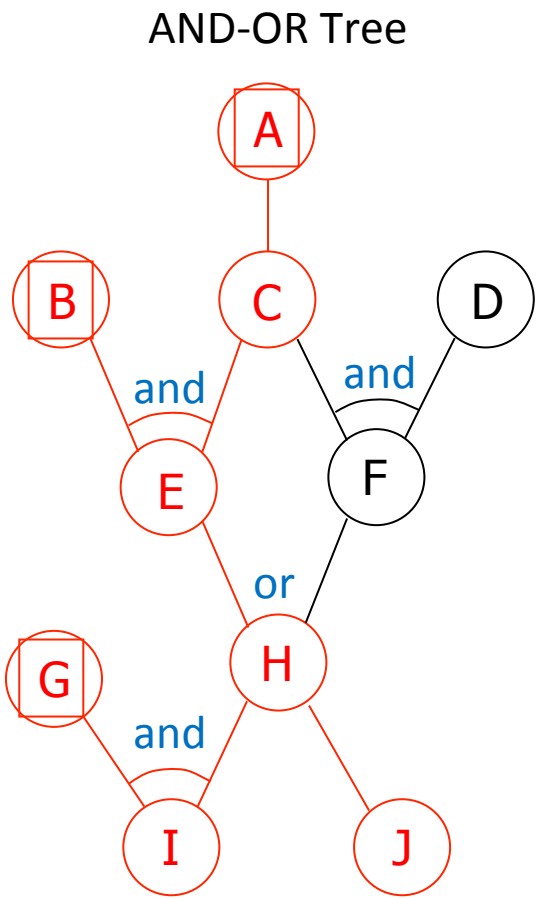
- Rule Activation
  - The Agenda is populated with "activated" rules due to assertion of facts into Working Memory
  - Rules are deactivated and removed from the agenda, if the matching facts are removed from Working Memory
- Rule Execution (Firing):
  - A rule fires, if Jess executes its RHS actions
  - Sequence of actions determined by Agenda
  - An action can be: new assertions of facts, removal / modification of facts, etc.
- Observation:
  - The content of the Agenda can be compared to a "program" that is executed (but this "program" can change during execution)

Forward Chaining

- R1: If A then assert C
- R2: If B and C then assert E
- R3: If C and D then assert F
- R4: If E then assert H
- R5: If F then assert H
- R6: If G and H then assert I
- R7: If H then assert J

Known Facts (Working Memory):

- A, B, G
- C, E, H, I, J



Result: we know now more about the world – that facts C, E, H, I, J are true as well

Retract Facts

- Bind variables to fact addresses:
  - We have the following fact in WM
    - `f-4  (MAIN::goal C on B clear)`
  - We use the following LHS of a rule to match it and also record its Jess-internal address:

```
(defrule check-goals
        ?factid   <-   (goal ?x on ?y clear)
        =>
        (retract ?factid ) )
```

  - `?factid = 4`
  - `?x = C, ?y = B`
- (retract ?factid) removes fact 4 from working memory

f-1 (MAIN:: Black Horse G 3)
Jess> (retract (fact-id 1))
remove (Black Horse G 3) !!!

Pattern Matching and Variable Binding at the LHS of a Rule

- Why do we need variables?
  - To match many patterns
  - Get a value (for future usage)

```
(deftemplate Book (slot hasName)(slot hasAuthor))

(defrule test-books
       (Book (hasName ?name1)(hasAuthor ?author1))
       (Book (hasName ?name2)(hasAuthor ?author2))
       (test (neq ?name1 ?name2))
       (test (eq ?author1 ?author2)) ?????
=>
...
)
```

- We want to test whether two books have the same author

Pattern Matching

- Problems
  - Pattern Matching with Wildcards:
    - Given the following fact in WM:
      - **(data red blue green yellow pink)**
    - Let's assume that a rule has the following pattern (conditional element) as its LHS:
      - **(data $? ?x $?)**
    - Describe whether there is any match with the fact in WM and what values are bound to the variable ?x:
      - This pattern matches the fact 5 times with the following bindings
        - 1. **?x = red**
        - 2. **?x = blue**
        - 3. **?x = green**
        - 4. **?x = yellow**
        - 5. **?x = pink**

Connective Constraints

- We can connect constraints to variables or to each other:
  - **&** : represents "and", satisfied if both adjoining constraints are satisfied
  - **|** : represents "or": satisfied if one of the adjoining constraints is satisfied
  - **~** : represents "not": satisfied if the following constraint is not satisfied
  - Precedence: **~ & |**
- Example patterns:

```
(is-a toy ?name ?color & ~blue &~ red)

(person (name ?name
        (occupation ?myjob & academic | accountant )
```

?myjob can have two possible bindings:
- either "academic" or "accountant"

Pattern Matching with Constraints

- Problems
  - Using constraints over variables
    - Given the following fact in WM:
      - **`(data red blue green yellow pink)`**
    - Let's assume that a rule has the following pattern (conditional element) as its LHS:
      - **`(data $? ?x & :(neq ?x yellow) )`**
    - Describe whether there is any match with the fact in WM and what values are bound to the variable ?x:
      - This pattern matches the fact 1 times with the following bindings
        - » 1. **`?x = pink`**

Manipulating Lists

- Problem
  - Manipulating Lists, deffunction
    - You have the following list manipulation functions
      - (length$ <item list>)
      - (create$ <item list>)
      - (nth$ <number> <item list>)
      - (first$ <item list>)
      - (rest$ <item list)
    - Let's assume, we have the following fact representing a list of items:
      - (input red blue green yellow orange pink)
    - Create a deffunction that takes the input list and produces an output list that contains these elements in reverse order

Manipulating Lists

- Problem
  - Manipulating Lists, using deffunction - Solution

```
(deffunction reverse ($?input)
    (bind ?output (create$ output))
    (bind ?i (- (length$ ?input) 1))
    (while (> ?i 0)
        (bind ?output (create$ ?output (nth$ ?i ?input)))
        (bind ?i (- ?i 1))
    )
    return ?output
)
```

Writing more complex RHS of Rules

- The RHS of a rule can consist of **a sequence of actions** that are executed during a Jess run:
  - assert, retract, modify
  - Functions, such as printout
- But also:
  - bind
  - if . . . then . . . Else
  - while
  - foreach
  - …

Note: whatever you can write at the RHS of a rule, you can directly type at the Jess command prompt

Control Statements on the RHS

IF – Then- Else

```
(defrule check-grocery-list
    (grocery-list $?x)
    =>
    (if (member$ eggs ?x)
     then
        (printout t "I need eggs" crlf)
     else
        (printout t "I don't need eggs" crlf))
)
```

- Can also be used at the command prompt:

```
Jess> (bind ?grocery-list (create$ eggs bread milk))
(eggs bread milk)
Jess> (if (member$ eggs ?grocery-list) then
(printout t "I need eggs" crlf)
else
(printout t "no eggs, thanks" crlf))
I need eggs
Jess>
```

Control Statements on the RHS

For-Each, While

- **for each**:

```
(defrule check-grocery-list
    (grocery-list $?grocery-list)
    =>
    (bind ?i 1)
    (foreach ?e ?grocery-list
        (printout t "Item " ?i ": " ?e crlf)
        (bind ?i (+ ?i 1))
    )
)
```

- **while**:

```
(defrule check-grocery-list
    (grocery-list $?grocery-list)
    =>
    (bind ?i 1)
    (while (<= ?i (length$ ?grocery-list)) do
        (bind ?e (nth$ ?i ?grocery-list))
        (printout t "Item " ?i ": " ?e crlf)
        (bind ?i (+ ?i 1))
    )
)
```

- How to assign value to a variable
- operators: + - * /
- brackets

Important Jess Language Constructs

- Templates
  - (deftemplate …)
- Rules
  - (defrule …)
- Facts
  - (deffacts …)
- Jess Functions
  - (deffunction …)
  - http://www.jessrules.com/jess/docs/71/deffunctions.html

Controlling Execution of Jess

- How can we influence the ordering of rule activations on the agenda?
- How does salience order activations on the agenda
- How does Depth order activations on the agenda
- How does Breadth order activations on the agenda

Salience

- We can define a "**salience**" for each rule
  - Is a number between -10000 and +10000
  - The default salience of a rule is 0
  - Introduces a ranking between rules:
    - A rule with higher salience will always fire before a rule with lower salience
    - Salience influences the ordering of rules on the Agenda
- Syntax:
  - (declare (salience <some-number>))

```
(defrule my-rule
      (declare (salience 10))
      . . . )
```

Conflict Resolution Strategies (**CRS**)

- In general:
  - If two rules on the agenda have the **same salience** (which is mostly the case), which one fires first?
- Jess uses a *conflict resolution strategy (CRS)* to determine their order on the agenda
  - Jess has two built-in strategies that can be used
    - **Depth** (Last In First Out)
    - **Breadth** (First In First Out)
  - The default CRS is the ***depth*** strategy

The Jess Language

- Web sites
  - http://www.jessrules.com/
- Tutorials
  - http://www.cs.rochester.edu/~brown/242/docs/JessTutorial.html
  - http://www.ida.liu.se/~her/JessTab/tutorial07/JessTabTutorial.pdf