COMPSCI 367 Tutorial 2

Overview

- Debugging

- Deffunctions

- Prefix form

- Bind, test

- Error handling example

- Assignment Hints

Debugging

```
(watch facts)
(watch instances)           ; used with objects
(watch slots)               ; used with objects
(watch rules)
(watch activations)
(watch messages)            ; used with objects
(watch message-handlers) ; used with objects
(watch generic-functions)
(watch methods)             ; used with objects
(watch deffunctions)
(watch compilations)        ; on by default
(watch statistics)
(watch globals)
(watch focus)
(watch all)                 ; watch everything
```

Deffunction

```
(deffunction <function-name> [optional comment]
    (?arg1 ?arg2 ...?argM [$?argN])    ;argument list. Last one may
    (<action1>                          ;be optional multifield arg.
    <action2>                           ;action1 to
      ...                               ;action(K-1) do not
    <action(K-1)>                       ;return a value
    <actionK>)                          ;only last action returned
```

Returned

Bind

- Normally used on RHS of rule
- Create & assign value to variable on-the-fly
- Sometimes used with create$

```
(defrule list-stuff
=>
(bind ?list(create$ joe moe hoe))
(printout t ?list crlf))
```

test

- Compare numbers, variables and strings on the LHS of rule

- Rule will only be activated if test evaluates to true

```
(defrule is-greater-than

    (num1 ?num1)

    (num2 ?num2)

    (test (> ?num1 ?num2))

=>

    (printout t ?num1 " is greater than " ?num2
crlf))
```

Incorrect Input Example

- How to deal with garbage input?

Assignment Hints

- Many ways to do assignment

- E.g. poker pre-flop decision tree

- E.g. clips structures (deftemplate, deffacts, defrule)

Don't attempt to use this model at the poker table – you will lose money :)

Do you have a premium hand?
- n → Do you have a suited connectors?
  - n → Do you have a pair?
    - n → Fold
    - y → Call
  - y → Are they high?
    - n → Fold
    - y → Raise
- y → Has another player raised?
  - n → Raise
  - y → Go All-in