CLIPS “Advance”

Junli Tao

jtao076@aucklanduni.ac.nz

Rm 321-723, Tamaki Campus

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

Content

- Template
- Variable (connective constraints, predicate constraints)
- Module
- Sample (dilemma1.clp)

deftemplate

- Used to create objects (nested facts, complex information)
- `(deftemplate [name] [comment]`
  ```
          [list of attributes])
  ```
- Simplifies related facts
- Makes facts more uniform, more structured
- Order of attributes irrelevant

Example

```
(deftemplate person "information about a
  person"
   (slot name)
   (slot gender (allowed-symbols M F N))
   (slot age (type NUMBER))
   (multislot friends)
)


(assert (person (name Daniel) (age 24)
   (gender M) (friends Simon Jane)))
```

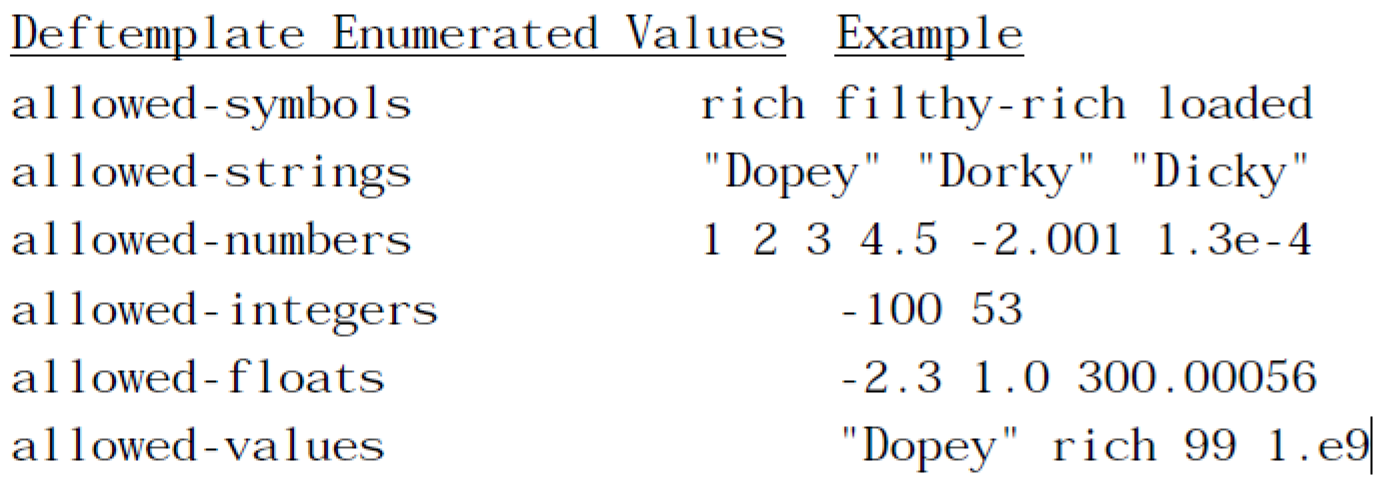
Type of Value

- Specify the type of value for slot/multi-slot
  - Supported types:
    - SYMBOL
    - STRING
    - NUMBER
    - INTEGER
    - FLOAT
    - FACT-ADDRESS
    - …

(type INTEGER FLOAT)   the same as (type NUMBER)

Value Range

- Specify the range of field values

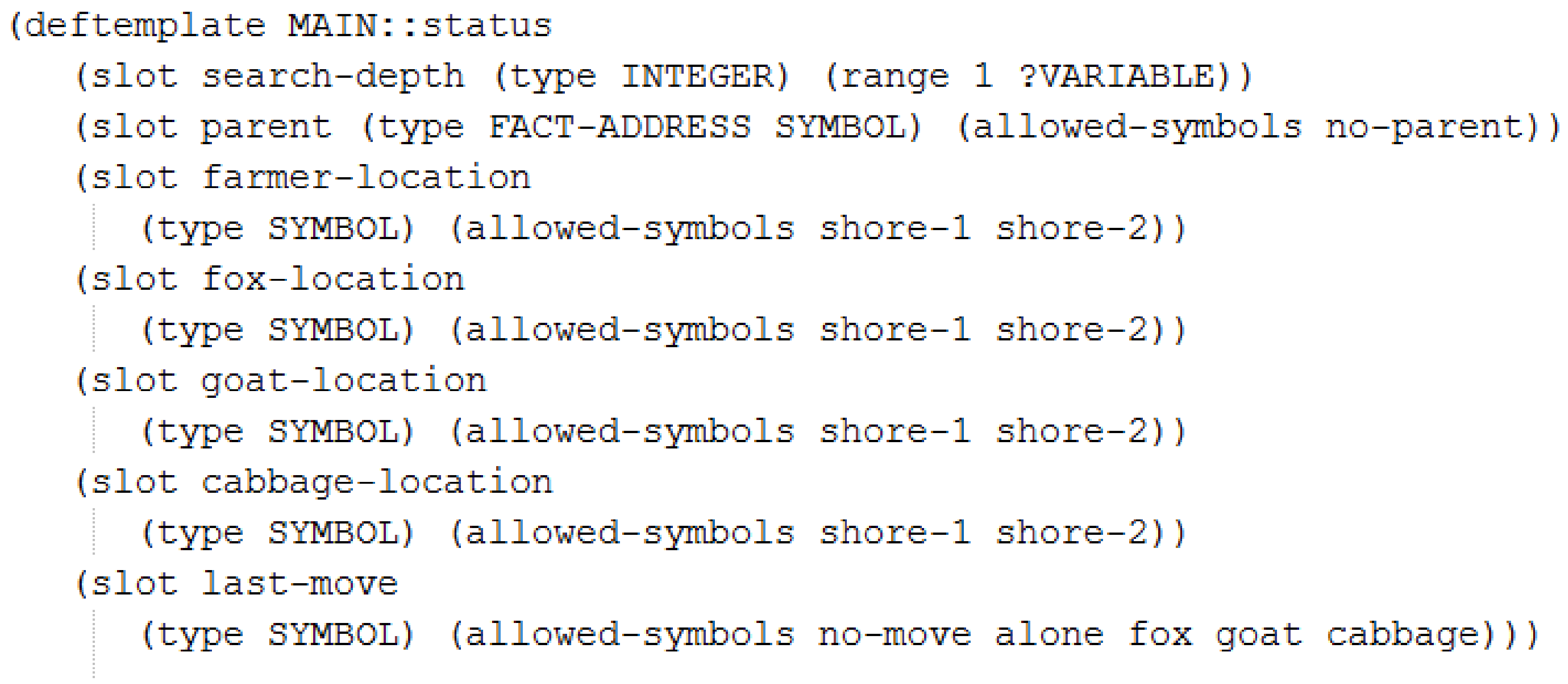


Deftemplate Enumerated Values	Example
allowed-symbols	rich filthy-rich loaded
allowed-strings	"Dopey" "Dorky" "Dicky"
allowed-numbers	1 2 3 4.5 -2.001 1.3e-4
allowed-integers	-100 53
allowed-floats	-2.3 1.0 300.00056
allowed-values	"Dopey" rich 99 1.e9

define a **range** with function **(range 1 100)**

Example

- delima1.clp



```
(deftemplate MAIN::status
   (slot search-depth (type INTEGER) (range 1 ?VARIABLE))
   (slot parent (type FACT-ADDRESS SYMBOL) (allowed-symbols no-parent))
   (slot farmer-location
      (type SYMBOL) (allowed-symbols shore-1 shore-2))
   (slot fox-location
      (type SYMBOL) (allowed-symbols shore-1 shore-2))
   (slot goat-location
      (type SYMBOL) (allowed-symbols shore-1 shore-2))
   (slot cabbage-location
      (type SYMBOL) (allowed-symbols shore-1 shore-2))
   (slot last-move
      (type SYMBOL) (allowed-symbols no-move alone fox goat cabbage)))
```

Default Value

- Inserted when no explicit values defined

```
(deftemplate prospect   ;name of deftemplate relation
  "vital information"   ;optional comment in quotes
  (slot name            ;name of field
    (type STRING)       ;type of field
    (default ?DERIVE))  ;default value of field name
  (slot assets          ;name of field
    (type SYMBOL)       ;type of field
    (default rich))     ;default value of field assets
  (slot age             ;name of field
    (type NUMBER)       ;type. NUMBER can be INTEGER or FLOAT
    (default 80)))      ;default value of field age
```

```
(assert (prospect (age 99) (name "Dopey"))))

(prospect (name "Dopey") (assets rich) (age 99))
```

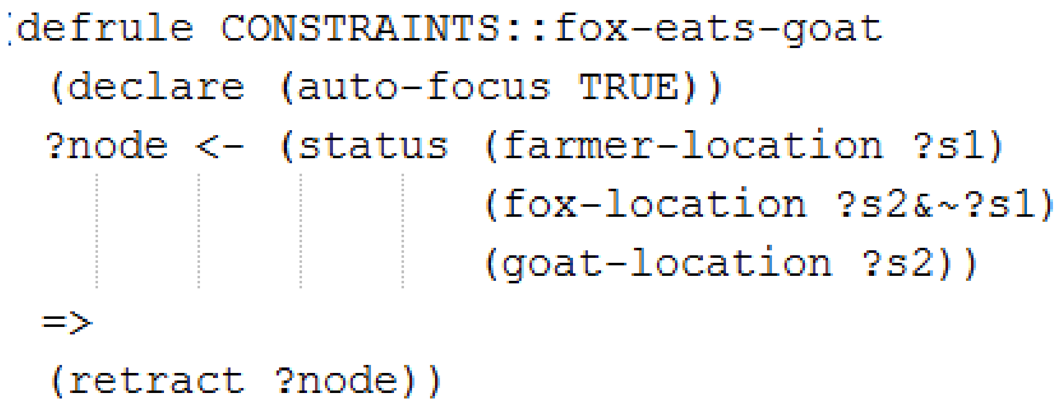
deffacts

```
(deftemplate MAIN::status
   (slot search-depth (type INTEGER) (range 1 ?VARIABLE))
   (slot parent (type FACT-ADDRESS SYMBOL) (allowed-symbols no-parent))
   (slot farmer-location
      (type SYMBOL) (allowed-symbols shore-1 shore-2))
   (slot fox-location
      (type SYMBOL) (allowed-symbols shore-1 shore-2))
   (slot goat-location
      (type SYMBOL) (allowed-symbols shore-1 shore-2))
   (slot cabbage-location
      (type SYMBOL) (allowed-symbols shore-1 shore-2))
   (slot last-move
      (type SYMBOL) (allowed-symbols no-move alone fox goat cabbage)))
```

```
(deffacts MAIN::initial-positions
  (status (search-depth 1)
          (parent no-parent)
          (farmer-location shore-1)
          (fox-location shore-1)
          (goat-location shore-1)
          (cabbage-location shore-1)
          (last-move no-move)))

(deffacts MAIN::opposites
  (opposite-of shore-1 shore-2)
  (opposite-of shore-2 shore-1))
```

Connective Constraints

- Connect individual variables
  - & (and), | (or), and ~ (not)
  - precedence high to low: ~ & |



```
(defrule CONSTRAINTS::fox-eats-goat
  (declare (auto-focus TRUE))
  ?node <- (status (farmer-location ?s1)
                   (fox-location ?s2&~?s1)
                   (goat-location ?s2))
  =>
  (retract ?node))
```

- Different from function (or) (and) (not)

Predicate Constraints

- Constraints depend on the return value of predicate function

```
(defrule CONSTRAINTS::circular-path
  (declare (auto-focus TRUE))
  (status (search-depth ?sd1)
          (farmer-location ?fs)
          (fox-location ?xs)
          (goat-location ?gs)
          (cabbage-location ?cs))
  ?node <- (status (search-depth ?sd2&:(< ?sd1 ?sd2))
                   (farmer-location ?fs)
                   (fox-location ?xs)
                   (goat-location ?gs)
                   (cabbage-location ?cs))
  =>
  (retract ?node))
```

Module

- Support for the modular development
- Allow a set of constructs to be grouped together

- Define by
```
(defmodule <module-name> [<comment>]
   <port-spec>*)
```

e.g.
```
(defmodule MAIN
   (export deftemplate status))
```

Specify Module for Constructs

- Constructs (deffacts, deftemplate, defrule, deffunction,…)

```
(deffacts MAIN::initial-positions
  (status (search-depth 1)
          (parent no-parent)
```

```
(defrule CONSTRAINTS::circular-path
  (declare (auto-focus TRUE))
  (status (search-depth ?sd1)
```

```
(deftemplate SOLUTION::moves
   (slot id (type FACT-ADDRESS SYMBOL) (allowed-symbols no-parent))
   (multislot moves-list
      (type SYMBOL) (allowed-symbols no-move alone fox goat cabbage)))
```

Export & Import

- Unless specifically exported and imported, the constructs of one module may not be used by another module

```
(defmodule MAIN
  (export deftemplate status))
```

```
(defmodule CONSTRAINTS
  (import MAIN deftemplate status))
```

- ?ALL – export/import all constructs

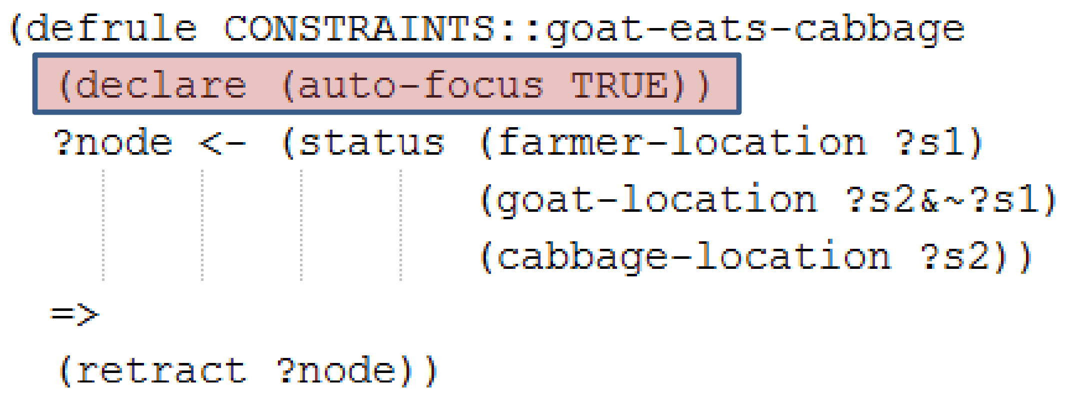
```
(defmodule A (export ?ALL))

(defmodule A (import D ?ALL))
```

Focus

- The agenda of the module with current focus is executed
- Current focus can be changed by focus command
  - (reset) and (clear) commands automatically set the current focus to the MAIN module
  - e.g. (focus CONSTRAINTS)

Auto-focus

- A rule's module is automatically focused upon when that rule, being declared auto-focus, is activated.



```
(defrule CONSTRAINTS::goat-eats-cabbage
  (declare (auto-focus TRUE))
  ?node <- (status (farmer-location ?s1)
                   (goat-location ?s2&~?s1)
                   (cabbage-location ?s2))
  =>
  (retract ?node))
```

Sources

- Template http://home.agh.edu.pl/~ligeza/wiki/clips:deftem
- Variable (connective constraints, predicate constraints) (bpg.pdf http://www.cs.auckland.ac.nz/courses/compsci367s2c/resources/clips/documentation/)
- Modules (auto-focus) (http://www.csie.ntu.edu.tw/~sylee/courses/clips/module.htm)
- Sample (dilemma1.clp)