Fuzzy Clips

- CLIPS contain the capability of handling fuzzy concepts and reasoning is called FuzzyCLIPS.

Defining Fuzzy Variables

- All fuzzy variables must be predefined before use with the **deftemplate** statement.
- Syntax

(deftemplate <name> [“<comments>”]

<from> <to> [<unit>] ; universe of discourse

(

t1

.

. ; list of primary terms

.

tn

)

)

Defining Fuzzy Variables

- Primary Terms

  A primary term ti (i=1,...,n) has the form

  (<name> <description of fuzzy set>)
- E.g.

```
(deftemplate group ;a linguistic variable declaration
    0 9          ;universe of discourse limits (no units)
    ( ;start of primary term declarations
    ; a primary term few described in singleton notation
    (few (1 0) (2 0.3) (3 0.9) (4 1) (5 0.8) (6 0.5) (7 0))
    ) ;end of primary term declarations
)   ;end of fuzzy deftemplate
```

Standard Deftemplate Definitions

- A fuzzy deftemplate describes a fuzzy variable. One may use these deftemplates to describe fuzzy facts in patterns and assert commands.

```
;; assume that the fuzzy deftemplates fz-height and
;; fz-weight have already been defined
(deftemplate person
     (slot name (type SYMBOL))
     (slot height (type FUZZY-VALUE fz-height))
     (slot weight (type FUZZY-VALUE fz-weight))
)

(defrule big-person
     (person (name ?n)
                  (weight heavy)
                  (height tall))
  =>
     (printout t ?n " is a big person" crlf)
)
```

- Syntax

```
<fuzzy-slot> ::= (slot <slotname> (type FUZZY-VALUE <fuzzy-deftemplate-name>)
```

Modifiers (Hedges) and Linguistic Expressions

- Predefined Modifiers

  change (modify) the shape of a fuzzy set in a way that suits the meaning of the word used.

Modifier Name	Modifier Description
not	1-y
very	y**2
somewhat	y**0.333
more-or-less	y**0.5

- User Defined Modifiers

  (add-fuzzy-modifier *modname modfunction*)

  user may also define modifiers exactly the same as the predefined.

- Linguistic Expressions

  primary terms +modifiers+*and* and *or*

  temperature very hot or very cold

  height below tall and above short

Using Fuzzy Variables in LHS Patterns

- A fuzzy LHS pattern is of the form

  (fuzzy-variable-name <linguistic-expr>)

  or

  (fuzzy-variable-name ?)

  or

  (fuzzy-variable-name ?<var-name>)

  or

  (fuzzy-variable-name ?<var-name> & <linguistic-expr>)

  or

  (template-name <slot-description>+)

Example

```
(deftemplate group          ;declaration of fuzzy variable group
  0 20 members
     ((few (3 1) (6 0))     ;primary term few
      (many (4 0) (6 1))    ;primary term many
     )
)
```

```
(defrule more-complex-lhs
  ?f <- (group very few or very many)
 =>
  (printout t "We are at the extreme limits of the number of"
               (get-u-units ?f) " in our club" crlf)
)
```

Using Fuzzy Variables in Deffacts Constructs

- Fuzzy fact specifications in a deffacts construct have the following form:

  (deffacts <deffacts-name> [<comment>]
  <RHS-pattern>*
)

```
(deffacts groupA "some fuzzy facts"
     (my_group (1 0) (5 1) (7 0))        ;singleton description
     (your_group (z 4 8))                ;standard description
     (their_group (s (+ 1 1) 4))
    (person (name ralph) (ht tall))
)
```

Using Fuzzy Variables in Assert Statements

- The assert command (with a single fact asserted in an assert function call) is of the form:

  (assert

  (<crisp fact> | fuzzy-variable-name <*description of fuzzy set*> | template-name <slot-description>+)

  [CF <*certainty factor*> | <*certainty factor expression*>]

)

Example

```
(assert (group few))

(assert (group (1 0) (5 1) (7 0)) )

(assert (group NOT [ very few OR many ] ))

(assert (group (z 4 8)))

(assert (person (name john) (ht extremely tall)))

(assert (person (name dan) (ht (pi 0 5.6)))

(assert (temp (24 0) (25 1) (26 0)))
```

Defuzzification

- A crisp value may be extracted from a fuzzy set using either the centre of gravity or mean of maxima techniques
- syntax

```
(moment-defuzzify ?<fact-var> | integer | <fuzzy-value>)      ; COG algorithm
(maximum-defuzzify ?<fact-var> | integer | <fuzzy-value>)     ; MOM algorithm
```

Certainty Factors of Rules

- The certainty factor of a rule may be declared in a manner similar to declaring the rule salience:
- syntax

```
(defrule some-rule
                (declare (CF <certainty factor>))
                ......
 =>
                ......
)
```

```
(deffacts initial-facts
     (fact1) CF 0.8                ;fact with crisp CF of 0.8
)

(defrule some-other-rule
     (declare (CF 0.7))            ;a rule with CF of 0.7
     (fact1)
  =>
     (printout t "Hello!")
)
```

FuzzyCLIPS Commands and Functions

- Accessing the Universe of Discourse (get-u, get-u-from, get-u-to, get-u-units)
- Accessing the Fuzzy Set (get-fs, get-fs-x, get-fs-y, get-fs-length, get-fs-lv, get-fs-value)
- Accessing the Certainty Factor (get-cf)
- Accessing the Threshold Certainty Factor (threshold, get-threshold)
- Setting the Rule CF Evaluation Behaviour (set-CF-evaluation, set-CF-evaluation)
- Controlling the Fuzzy Set Display Precision (set-fuzzy-display-precision, get-fuzzy-display-precision)

Simple Example

```
CLIPS>(load "simplTst.clp")
Defining deftemplate: speed_error
Defining deftemplate: speed_change
Defining deffacts: my_facts
Defining defrule: speed-too-fast +j
Defining defrule: speed-ok +j
Defining defrule: get-crisp-value-and-print-rslt +j
TRUE
CLIPS> (reset)
==> f-0     (initial-fact) CF 1.00
==> f-1     (speed_error zero) CF 0.90                    [ linguistic description of fuzzy set ]
            ( (0.0 1.0) (0.11 0.0)  )                     [ singletons describe fuzzy set in detail ]

==> Activation 0      speed-ok: f-1
==> Activation 0      speed-too-fast: f-1

CLIPS> (run)
FIRE    1 speed-too-fast: f-1
==> f-2     (speed_change ???) CF 0.63                    [ CF = 0.9 * 0.7 for fuzzy-fuzzy rule ]
            ( (0.1 0.0) (0.1495 0.0991)  )

==> Activation -1     get-crisp-value-and-print-rslt: f-2
FIRE    2 speed-ok: f-1
<== f-2     (speed_change ???) CF 0.63                    [retraction of fuzzy fact and ...]
            ( (0.1 0.0) (0.1495 0.0991)  )

<== Activation -1     get-crisp-value-and-print-rslt: f-2
==> f-3     (speed_change ???) CF 0.63                    [reassertion as fact is modified]
            ( (0.0 1.0) (0.1 0.1) (0.1333 0.06667) (0.1495 0.0991)  )

==> Activation -1     get-crisp-value-and-print-rslt: f-3
FIRE    3 get-crisp-value-and-print-rslt: f-3
Change speed by a factor of:  0.3553202565269306
3 rules fired        Run time is 0.06400000000212458 seconds.
46.87499999844391 rules per second.
3 mean number of facts (3 maximum).
1 mean number of instances (1 maximum).
1 mean number of activations (2 maximum).
CLIPS>
```

Continuous Systems

- **The Run Command**

In Fuzzy-CLIPS the run command is extended to receive any of the following parameters:

n	(a positive integer) FuzzyCLIPS will run until n rules have executed or until the agenda is empty, whichever comes first, e.g., (run 10)
-1	FuzzyCLIPS runs until the agenda is empty, e.g., (run -1)
-2	FuzzyCLIPS runs forever. Control-C interrupts the execution, e.g., (run -2)
-n	(a negative integer less than -2). FuzzyCLIPS runs until n rules have executed, e.g., (run -10)

- **Runstart and Runstop Functions**

CLIPS allows users to call external functions that are executed at the end of each cycle of the inference engine(i.e., after each rule firing). This is done by calling the AddRunFunction routine of CLIPS to include the function in the list of exec functions.

CLIPS Functionality within FuzzyCLIPS

- Modifying and Duplicating Facts

  These functions will always return FALSE when used with deftemplate fuzzy facts, the behaviour for modify is the same as a normal modify, Duplicate doesn't work.
- Load, Save, Bload, Bsave, Load-facts, Save-facts
- Constructs-to-c
- CreateFact, GetFactSlot, PutFactSlot

Matlab

Conclusion

- Fuzzy systems, including fuzzy logic and fuzzy set theory, provide a rich and meaningful addition to standard logic. The mathematics generated by these theories is consistent, and fuzzy logic may be a generalization of classic logic. The applications which may be generated from or adapted to fuzzy logic are wide-ranging, and provide the opportunity for modeling of conditions which are inherently imprecisely defined, despite the concerns of classical logicians. Many systems may be modeled, simulated, and even replicated with the help of fuzzy systems, not the least of which is human reasoning itself.

Conclusion

- Fuzzy systems, including fuzzy logic and fuzzy set theory, provide a rich and meaningful addition to standard logic. The mathematics generated by these theories is consistent, and fuzzy logic may be a generalization of classic logic. The applications which may be generated from or adapted to fuzzy logic are wide-ranging, and provide the opportunity for modeling of conditions which are inherently imprecisely defined, despite the concerns of classical logicians. Many systems may be modeled, simulated, and even replicated with the help of fuzzy systems, not the least of which is human reasoning itself.