บทที่ 6

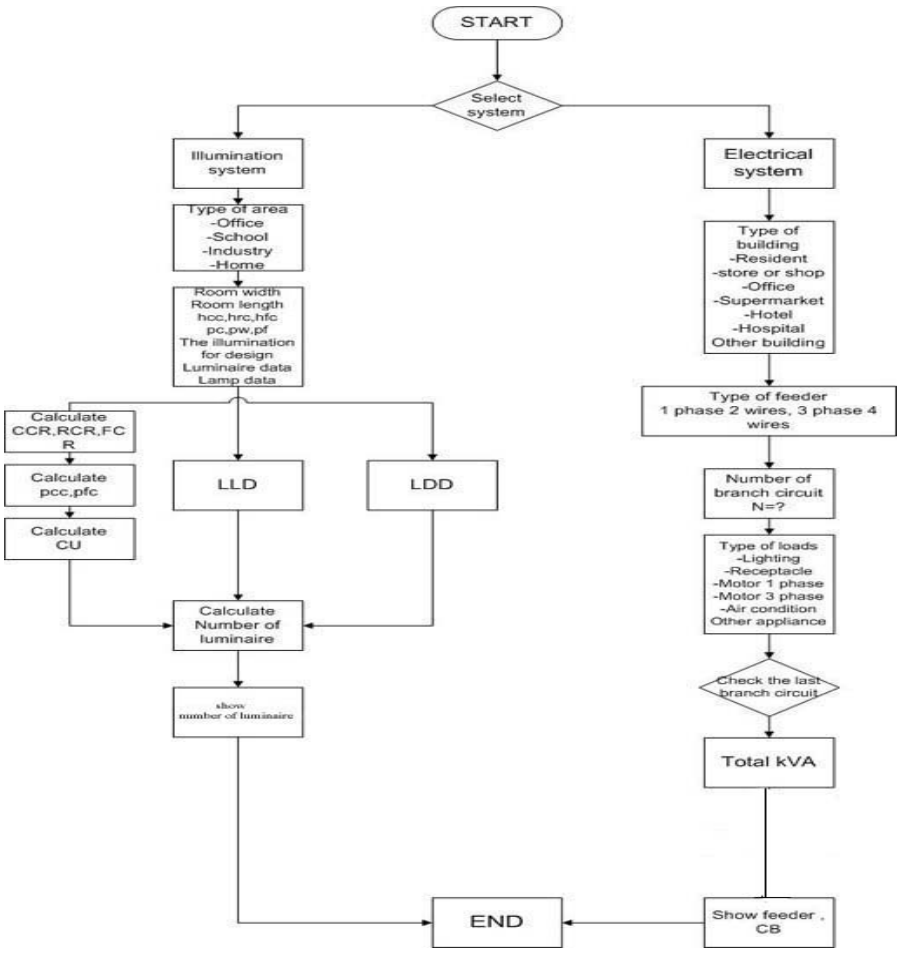
EEIS : ระบบผู้เชี่ยวชาญเพื่อช่วยในการออกแบบระบบไฟฟ้าและแสงสว่างในอาคาร

EEIS เป็นโปรแกรมที่พัฒนามาจากระบบเปลือกผู้เชี่ยวชาญ “GES” เพื่อใช้ในการออกแบบระบบไฟฟ้าและแสงสว่างในอาคาร ที่ใช้งานบนคอมพิวเตอร์ที่เป็นระบบปฏิบัติการลีนุกซ์โดยใช้โปรแกรมภาษา LISP ในการพัฒนาโปรแกรม ในส่วนของโปรแกรมได้แบ่งการออกแบบระบบเป็น 2 ส่วนคือ ระบบไฟฟ้าและระบบแสงสว่าง โดยในส่วนของระบบไฟฟ้านั้น ระบบที่พัฒนาขึ้นสามารถแสดงผลลัพธ์เป็นค่าต่างๆที่จำเป็นในการออกแบบระบบไฟฟ้า เช่น ขนาดสายไฟฟ้า ขนาดเซอร์กิตเบรกเกอร์ หรือว่าขนาดหม้อแปลงไฟฟ้า เป็นต้น ส่วนระบบแสงสว่างระบบจะสามารถคำนวณจำนวนหลอดที่ต้องใช้ต่อพื้นที่ๆต้องการใช้งาน รวมถึงตำแหน่งการติดตั้ง เป็นต้น

6.1 การออกแบบระบบ

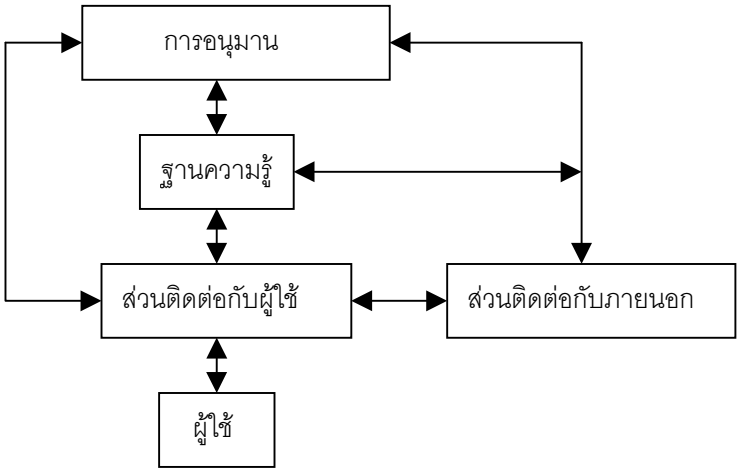
EEIS พัฒนามาจากระบบเปลือกผู้เชี่ยวชาญ “GES” โดยทำการพัฒนา แก้ไขและเพิ่มเติมตัวโปรแกรมเพื่อให้เหมาะสมกับการนำไปประยุกต์ใช้ ซึ่งในที่นี้คือประยุกต์ใช้เพื่อสร้างโปรแกรมการออกแบบระบบไฟฟ้าและแสงสว่างในอาคาร การแก้ไขตัวโปรแกรมเดิมนั้นสิ่งที่ทำการแก้ไขคือโลโก้ต้อนรับตอนใช้งานและส่วนของการอนุมาน(จากระบบเปลือกผู้เชี่ยวชาญ ”GES” ที่สามารถอนุมานได้ทั้ง 3 แบบ แต่ในระบบ EEIS จะให้ใช้เพียงแบบเดียว คือการอนุมานแบบเดินหน้า) ส่วนที่ต้องเพิ่มให้กับระบบเดิม คือ ระบบฐานความรู้ที่เกี่ยวข้องกับการออกแบบระบบไฟฟ้าและแสงสว่าง

ส่วนของโปรแกรมการออกแบบระบบไฟฟ้าและแสงสว่างในอาคารที่พัฒนาขึ้นจะทำขึ้นมาให้ผู้ใช้ ใช้งานในลักษณะของเมนู โดยมีแผนผังการทำงาน ดังภาพประกอบ 6-1



ภาพประกอบ 6-1 ขั้นตอนการทำงานของโปรแกรมการออกแบบระบบไฟฟ้าและแสงสว่างในอาคาร

6.2 **โครงสร้างของระบบผู้เชี่ยวชาญ** “EEIS”



ภาพประกอบ 6-2 โครงสร้างของระบบผู้เชี่ยวชาญ “EEIS”

จากโครงสร้างดังกล่าวจะเห็นว่าในส่วนติดต่อกับผู้ใช้เป็นส่วนที่ติดต่อกับผู้ใช้โดยตรง ซึ่งในส่วนนี้มีความสำคัญมาก จึงจำเป็นต้องมีการออกแบบให้ผู้ใช้สามารถใช้ระบบและเข้าถึงระบบได้สะดวกที่สุด โดยระบบผู้เชี่ยวชาญที่พัฒนาขึ้นเพื่อช่วยในการออกแบบระบบไฟฟ้าและแสงสว่างนี้ จะสร้างส่วนติดต่อกับผู้ใช้ในลักษณะเมนู (menu) เพื่อทำให้ผู้ใช้สามารถใช้ระบบนี้ได้อย่างสะดวกมากขึ้น โดยผู้ใช้ทำการเพียงแค่ป้อนค่าตัวเลขที่ระบบต้องการ

ซึ่งเมื่อพิจารณาในโครงสร้างของระบบผู้เชี่ยวชาญที่กล่าวมาข้างต้น ซึ่งได้นำมาเป็นหลักในการพัฒนาการออกแบบระบบไฟฟ้าและแสงสว่าง โดยในแต่ละส่วนที่ทำการออกแบบนั้นจะทำให้ระบบมีประสิทธิภาพในการทำงานและเป็นระบบที่สมบูรณ์มากขึ้น ซึ่งสามารถอธิบายในแต่ละส่วนได้ดังนี้

6.2.1 **ส่วนติดต่อกับผู้ใช้**

ส่วนติดต่อกับผู้ใช้เป็นส่วนที่ระบบติดต่อกับผู้ใช้โดยตรง โดยระบบมีการตั้งคำถามเพื่อให้ผู้ใช้ตอบคำถามเหล่านั้น ซึ่งระบบอาจจะให้ผู้ใช้เลือกคำตอบจากตัวเลือกที่แสดงออกมาให้เห็น จากนั้นระบบจะทำการเก็บความรู้ที่ได้จากคำตอบที่ได้จากผู้ใช้เพื่อนำเข้าสู่กระบวนการอนุมานความรู้ ซึ่งนำไปใช้แก้ปัญหาต่อไป ระบบที่ดีนั้นควรเป็นวิธีที่ทำให้ผู้ใช้เกิดความสะดวกในการใช้มากที่สุด ในส่วนการติดต่อกับผู้ใช้นั้น สามารถจำแนกได้คือ วิธีการโต้ตอบระหว่างผู้ใช้กับระบบ และการเก็บความรู้จากคำตอบของผู้ใช้

วิธีการโต้ตอบระหว่างผู้ใช้กับระบบ

ในงานวิจัยนี้ ได้ทำการสร้างการโต้ตอบระหว่างผู้ใช้กับระบบในรูปของเมนู โดยตัวอย่างส่วนของโปรแกรมมีลักษณะดังนี้

```
(defun a3()
(format t"~% $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
             $$          ELECTRICAL SYSTEMS              $$
             $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
             $$     1. Design Electrical systems         $$
             $$     2. Help                              $$
             $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$")
(format t"~%")
(format t"~%   q-Quit from EEIS ")
(format t"~%")
(setq ele_sys(check-word-error " ENTER MENU: " '(1 2 q)))
```

```
(
cond
(
(equal ele_sys '1) (setq x3 'Design)
(format t"~& You are select menu:~a ELECTRICAL SYSTEMS" x3)
)
(
(equal ele_sys '2) (setq x4 'Help)
(format t"~& You are select menu:~a " x4)
)
)
(setq ans(list 'ele_sys ele_sys))
(setq *binding-stream*(cons(cons ans(car *binding-stream*))(cdr
*binding-stream*)))
(fn_set_variable_match(reverse(first *binding-stream*)))
)
```

จากตัวอย่างโปรแกรมที่ยกตัวอย่างมานี้ ถ้าผู้ใช้ทำการเลือกเมนูที่ 1 โดยทำการป้อนค่าคือ เลข 1 ระบบจะเข้าสู่เมนูของการออกแบบระบบไฟฟ้า แต่ถ้าหากผู้ใช้ทำการเลือกเมนูที่ 2 ระบบจะแสดงเมนูช่วยเหลือ(help) ซึ่งเป็น Text file ซึ่งจะเป็นส่วนที่อธิบายแนวทางการออกแบบระบบไฟฟ้า คำนิยามต่างๆที่เกี่ยวข้องกับการออกแบบ

การเก็บความรู้จากคำตอบของผู้ใช้

ความรู้ที่ระบบสามารถนำไปใช้เพื่อการอนุมาน มีรูปแบบดังนี้

((type-of-object object) Attribute Comparator (Type-of –value Value))

โดยที่ Type-of-object คือ Argument ที่กล่าวถึงชนิดของสิ่งที่อธิบาย

Object คือ Argument ที่เป็นสิ่งที่อธิบาย

Attribute คือ Argument ที่อธิบายถึงสิ่งที่สนใจ

Comparator คือ Argument ที่เปรียบเทียบค่าระหว่าง Object กับ Value

Type-of –value คือ Argument ที่กล่าวถึงชนิดของ Value

Value คือ Argument ที่ค่าของสิ่งที่อธิบาย

ซึ่งจากตัวอย่างโปรแกรมข้างต้น พบว่า

- ถ้าผู้ใช้เลือกเมนูที่1 คือ Design Electrical systems จะทำให้ระบบได้รับความรู้ใหม่หรือความจริงอันใหม่ขึ้น คือ ((select_electrical_systems menu)(1)=(atomic true))

- ถ้าผู้ใช้เลือกเมนูที่ 2 คือ Help จะทำให้ระบบได้รับความรู้ใหม่หรือความจริงอันใหม่ขึ้น8คือ ((select_electrical_systems menu)(2)=(atomic true)) ซึ่งความรู้ใหม่หรือความจริงอันใหม่นี้ระบบจะทำการเก็บเข้าในระบบเพื่อนำไปทำการอนุมานต่อไป

6.2.2 **ส่วนที่ติดต่อกับภายนอก**

ส่วนนี้เป็นส่วนที่ติดต่อกับภายนอกซึ่งจะทำหน้าที่ในการเรียกโปรแกรมต่างๆ ที่เกี่ยวกับการคำนวณค่าต่างๆ หรือโปรแกรมในการแสดงภาพ เข้ามาในระบบ โดยโปรแกรมที่นำเข้ามานั้น เป็นภาษาเดียวกับภาษาที่ใช้เขียนในระบบ ในที่นี้คือภาษา LISP

โปรแกรมที่ส่วนติดต่อกับภายนอกนำเข้าในระบบมี 2 ประเภท คือ

1. โปรแกรมภายใน (Internal Program) ซึ่งเป็นโปรแกรมที่เขียนด้วยภาษา LISP แต่อยู่ภายนอกระบบ การเขียนโปรแกรมลักษณะนี้จะมีข้อดี คือ เวลาในการที่ระบบเรียกโปรแกรมย่อยเหล่านี้จะใช้เวลาค่อนข้างเร็ว ในงานวิจัยนี้จะมีโปรแกรมภายในจำนวนมากซึ่งเป็นโปรแกรมการคำนวณต่างๆ ทั้งการคำนวณหาขนาดสาย ขนาดเซอร์กิตเบรกเกอร์ หรือว่าการคำนวณจำนวนหลอดไฟ เป็นต้น โดยมีตัวอย่างโปรแกรมภายในที่ใช้ในการคำนวณค่าต่างๆ ดังนี้

```
(defun a6()
(format t"~%
        $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
        $$                  Feeder 1 phase 2 wires                    $$
        $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$")
(format t"~%")
(format t"~% Maximum number of branch circuit = 16 circuits")
(format t"~% PRECAUTION! You must not have 3 phase load  ")
(format t"~% Enter number of branch circuit(1-16): ")
(setq I_total 0)
(setq branch (read))
(dotimes (count  branch)
(setq count (+ count 1))
(format t"~& CIRCUIT NO.:~a" count)
```

```
(format t"~%  $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
               $$                    Type of loads                    $$
               $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
               $$                     1. lighting                     $$
               $$                     2. receptacle                   $$
               $$                     3. air condition                $$
               $$                     4. shower heater                $$
               $$                     5. motor                        $$
               $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$")
(format t"~& Enter type of load (1-5):")
(setq load (read))
(
cond
(
(equal load '1) (setq x7 'lighting)
(format t"~& You are select:~a LOAD" x7)
)
(
(equal load '2) (setq x8 'receptacle)
(format t"~& You are select:~a LOAD" x8)
)
(
(equal load '3) (setq x9 'air)
(format t"~& You are select:~a CONDITION LOAD" x9)
)
(
(equal load '4) (setq x10 'shower)
(format t"~& You are select:~a HEATER LOAD" x10)
)
(
```

```
(equal load '5) (setq x11 'motor)
(format t"~& You are select:~a LOAD" x11)
)
)
(
cond
(
(equal load '1)
(format t"~%

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
 $$                              lamps (watt)                              $$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
 $  Low volt aluminum reflector lamp                        20 50            $
 $  Low volt dicronic reflector halogen lamp                20 35 50         $
 $  Par 30 halogen                                          75               $
 $  Par 38 halogen reflector                                75 100           $
 $  Par 38 reflector                                        80 120           $
 $  spot R50                                                25 40            $
 $  Spot R60                                                60               $
 $  Spot R80                                                40 60 75 100     $
 $  Par 56                                                  300              $
$  High pressure mercury                                    50 80 125        $
 $  GLS                                                     40 60 75 100 150 $
$  Halogen single end clear                                 75 100 150       $
$  Halogen single end prosted                               75 100 150       $
$  Halogen double end                                       150 200 300      $
$  Compact fluo. single tube                                7 8 11           $
$  Compact fluo. double tube                                10 13 18 26      $
$  White son                                                30 50 100        $
```

```
$  HPS                                                    50 70              $
$  Metal hal. single end                                  70 150             $
$  Metal hal. double end                                  70 150             $
 $  LPS                                                 18 35 55 90 135 180       $
 $  HPS                                                150 250 400 700 1000      $
  $  MH                                                 100 125 250 300 400      $
  $                                                        700 1000               $
  $  HPM                                               50 60 125 175 250 300     $
  $                                                     400   700 1000 2000       $
  $  Fluorescent                                                18 20 36 40    $
  $  Fluor.circline                                              22 32 40       $
  $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$")
  (format t "~%")
  (format t " Enter watt of lamp: ")
  (setq watt (read))
  (format t " Enter units of lamp:")
  (setq units (read))
  (setq total (* watt units))
  (setq I1 ( * (/ total 220)  1.25))
  (cond ((and (> I1 0) (<= I1 9))
(setq d_gc  '(2x0.5)))
((and (> I1 9) (<= I1 14))
(setq d_gc  '(2x1)))
((and (> I1 14) (<= I1 17))
(setq d_gc  '(2x1.5)))
((and (> I1 17) (<= I1 23))
(setq d_gc  '(2x2.5)))
((and (> I1 23) (<= I1 31))
(setq d_gc  '(2x4)))
((and (> I1 31) (<= I1 42))
(setq d_gc  '(2x6)))
```

```
((and (> I1 42) (<= I1 60))
(setq d_gc  '(2x10)))
((and (> I1 60) (<= I1 81))
(setq d_gc  '(2x16)))
((and (> I1 81) (<= I1 111))
(setq d_gc  '(2x25)))
((and (> I1 111) (<= I1 137))
(setq d_gc  '(2x35)))
((and (> I1 137) (<= I1 169))
(setq d_gc  '(2x50)))
((and (> I1 169) (<= I1 217))
(setq d_gc  '(2x70)))
((and (> I1 217) (<= I1 271))
(setq d_gc  '(2x95)))
((and (> I1 271) (<= I1 316))
(setq d_gc  '(2x120)))
((and (> I1 316) (<= I1 364))
(setq d_gc  '(2x150)))
((and (> I1 364) (<= I1 424))
(setq d_gc  '(2x185))))
(setq ans (list 'd_gc d_gc))
(format t "~& size of conductor: ~a sq.mm." d_gc)
(setq I_cb I1)
(if (<= I_cb 3) (setq cb 3)
(if (<= I_cb 5) (setq cb 5)
(if (<= I_cb 10) (setq cb 10)
(if (<= I_cb 15) (setq cb 15)
(if (<= I_cb 20) (setq cb 20)
(if (<= I_cb 30) (setq cb 30)
(if (<= I_cb 40) (setq cb 40)
(if (<= I_cb 50) (setq cb 50)
```

```
(if (<= l_cb 60) (setq cb 60)
(if (<= l_cb 70) (setq cb 70)
(if (<= l_cb 90) (setq cb 90)
(if (<= l_cb 100) (setq cb 100)
(if (<= l_cb 125) (setq cb 125)
(if (<= l_cb 150) (setq cb 150)
(if (<= l_cb 175) (setq cb 175)
(if (<= l_cb 200) (setq cb 200)
(if (<= l_cb 225) (setq cb 225)
(if (<= l_cb 250) (setq cb 250)
(if (<= l_cb 300) (setq cb 300)
(if (<= l_cb 350) (setq cb 350)
(if (<= l_cb 400) (setq cb 400)
(if (<= l_cb 450) (setq cb 450)
(if (<= l_cb 500) (setq cb 500)
(if (<= l_cb 600) (setq cb 600)
(if (<= l_cb 700) (setq cb 700)
(if (<= l_cb 800) (setq cb 800)
(if (<= l_cb 1000) (setq cb 1000)
(if (<= l_cb 1250) (setq cb 1250)
(if (<= l_cb 1400) (setq cb 1400)
(if (<= l_cb 1600) (setq cb 1600)
(if (<= l_cb 1800) (setq cb 1800)
(if (<= l_cb 2000) (setq cb 2000)
(if (<= l_cb 2500) (setq cb 2500)
(if (<= l_cb 2800) (setq cb 2800)
(if (<= l_cb 3000) (setq cb 3000)
(if (<= l_cb 3200) (setq cb 3200))))))))))))))))))))))))))))))))))))))))
(format t "~% size of CB: ~a AT" cb)
(setq l_total (+ l_total l1)))
```

2.โปรแกรมภายนอก (External Program) ซึ่งเป็นโปรแกรมที่เขียนด้วยภาษาอื่นๆ ที่ไม่ใช่ภาษา LISP

ในการนำเอาโปรแกรมภายนอกเข้ามาในระบบ จะใช้คำสั่ง system เช่น

(system "kwrite /bs/industrial.txt") เป็นการเรียกไฟล์ชื่อ industrial ซึ่งเป็น text file หรือ

(system "kview /bs/001.jpg") เป็นการเรียกไฟล์รูปภาพชื่อ 001 เป็นต้น

6.2.3 ฐานความรู้

ในโปรแกรมการออกแบบระบบไฟฟ้าและแสงสว่างภายในอาคารนั้น จะต้องทำการสร้างฐานความรู้ซึ่งอาจได้มาจากผู้เชี่ยวชาญหรือตำราต่างๆที่เกี่ยวข้องกับการออกแบบระบบไฟฟ้าและแสงสว่างในอาคาร เพื่อที่จะนำความรู้เหล่านี้เข้าสู่ระบบเพื่อนำไปทำการประมวลผล และทำการอนุมานต่อไป โดยการแสดงความรู้นั้นจะแสดงในรูปของกฎและความจริง (แสดงในภาคผนวก ข) โดยโครงสร้างของความจริงในงานวิจัยนี้ จะมีลักษณะดังนี้

Fact n

((type-of-object object) Attribute Comparator (Type-of –value Value))

โดยที่ Type-of-object คือ Argument ที่กล่าวถึงชนิดของสิ่งที่อธิบาย

Object คือ Argument ที่เป็นสิ่งที่อธิบาย

Attribute คือ Argument ที่อธิบายถึงสิ่งที่สนใจ

Comparator คือ Argument ที่เปรียบเทียบค่าระหว่าง Object กับ Value

Type-of –value คือ Argument ที่กล่าวถึงชนิดของ Value

Value คือ Argument ที่ค่าของสิ่งที่อธิบาย

เช่น fact 1 ((show menu) (picture) = (atomic true))

ส่วนโครงสร้างของกฎในงานวิจัยนี้ จะมีลักษณะดังนี้

Identify n

name_rule

If

(condition 1) (condition 2) …… (condition N)

Then

(conclusion 1) (conclusion 2) …….. (conclusion M)

End-of-rule

โดยที่ condition N และ conclusion M เป็นรูปแบบ (expression) ที่มีรูปแบบ 2 แบบ คือ

- รูปแบบทั่วไป (Normal Expression) มีรูปแบบไวยากรณ์ดังนี้

((type-of-object object) Attribute Comparator (Type-of –value Value))

- รูปแบบการเรียก action (Action Expression) มีรูปแบบไวยากรณ์ดังนี้

(@ name-action)

เมื่อ name-action คือ ชื่อของ action ซึ่งมีทั้งที่เป็น Internal action และ external action ดังที่กล่าวมาข้างต้นในบทที่ 4

ตัวอย่างกฎ

```
identify
1
rule1
if
((show menu) (picture) = (atomic true))
then
(@ a1)
((select menu) (electrical systems or illumination systems) = (atomic true))
end-of-rule
identify 2
rule2
if
((select menu) (electrical systems or illumination systems) = (atomic true))
then
(@ a2)
((select menu)  ((? num_menu)) = (atomic true))
end-of-rule
```

จากตัวอย่างกฎที่ยกมา 2 กฎ โดยมีชื่อกฎ คือ 1 และ 2 ตามลำดับ โดยทั้ง 2 กฎ จะมีรูปแบบ ทั้ง 2 แบบที่กล่าวมาข้างต้น คือ มีทั้งรูปแบบทั่วไป เช่น ((select menu) (electrical systems or illumination systems) = (atomic true)) และมีทั้งรูปแบบการเรียก action เช่น (@ a2)

6.2.4 การอนุมาน

ในระบบผู้เชี่ยวชาญที่พัฒนาขึ้นในงานวิจัยนี้ จะใช้การอนุมานแบบเดินหน้าอย่างเดียว โดยในการอนุมานนั้นระบบจะเริ่มทำการอนุมานตั้งแต่กฎที่ 1 จากนั้นก็จะไปสู่กฎอื่นต่อไป โดยหลักการอนุมานกล่าวได้ว่า

“ ถ้าเหตุของกฎเป็นจริง จะทำให้กฎของผลนั้นเป็นจริงด้วย “ โปรแกรมจะทำการอนุมานโดยเริ่มจากฐานข้อมูล ซึ่งจะมีการเรียกฐานข้อมูลเหล่านั้นมาทำการพิจารณาเทียบกับกฎว่ากฎไหนมีเหตุทุกเหตุเป็นจริง จะทำให้ผลของกฎนั้นเป็นจริงด้วย ซึ่งผลจากการอนุมานจะทำให้ได้ความจริงใหม่เก็บในฐานความจริง ซึงจะเห็นได้ว่าระบบจะได้ความจริงเพิ่มในฐานความจริงตลอดเวลา สำหรับการเปรียบเทียบหาว่ากฎที่ทีเหตุทุกเหตุเป็นจริงนั้น จะทำโดยการนำเหตุแต่ละตัวของกฎไปเปรียบเทียบกับความจริงว่าตรงกับค่าความจริงหรือไม่ ถ้าตรงแสดงว่าเหตุดังกล่าวเป็นจริงนั่นเอง

เมื่อทำการอนุมานไปเรื่อยๆ แล้วพบไวยากรณ์ของ @ name-action ระบบจะรับรู้ว่าเป็นการเรียกโปรแกรมภายนอก ซึ่งระบบจะทำการประมวลผลโปรแกรมนั้น เมื่อสิ้นสุดการทำงานของโปรแกรมภายนอกแล้วนั้น ระบบก็จะทำการอนุมานต่อไปเรื่อยๆจนสิ้นสุดการทำงาน

ความจริงใหม่ที่เกิดขึ้นในฐานความรู้นั้น ในแต่ละครั้งนั้นอาจจะไม่เท่ากัน ทั้งนี้เนื่องจากการตัดสินใจในการเลือกตัวเลือกของผู้ใช้ เพราะตัวเลือกแต่ละตัวเลือกจะให้ความจริงที่แตกต่างกัน กล่าวโดยสรุปแล้วความจริงที่เกิดขึ้นจะมีอยู่ 3 ประเภท คือ

1. ความจริงเริ่มต้น (Initial Fact) เป็นความจริงที่เราใส่ไว้เริ่มต้นให้กับระบบ (แสดงในภาคผนวก ข) จะไม่เปลี่ยนแปลงเมื่อระบบถูกใช้เพื่อการอนุมานในแต่ละครั้ง

2. ความจริงในขณะนั้น (Real – time Fact) เป็นความจริงที่เกิดจากการเลือกตัวเลือกของผู้ใช้ในขณะใดขณะหนึ่งเพื่อตอบคำถามที่ระบบได้ถามผู้ใช้

3. ความจริงที่เกิดจากการอนุมาน (Inference Fact) เป็นความจริงที่เกิดจากผลของการอนุมาน