การพัฒนาระบบเพื่อตรวจสอบกฎในฐานความรู้

VALIDATION OF RULE – BASED SYSTEM

โดย

นายสรายุทธ กรวิรัตน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

การพัฒนาระบบเพื่อตรวจสอบกฎในฐานความรู้

VALIDATION OF RULE – BASED SYSTEM

โดย

นายสรายุทธ กรวิรัตน์

อาจารย์ที่ปรึกษา

รศ.ดร. เอื้อน ปิ่นเงิน

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

ปริญญานิพนธ์ปีการศึกษา 2549

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาระบบเพื่อตรวจสอบกฎในฐานความรู้

VALIDATION OF RULE – BASED SYSTEM

ผู้จัดทำ

นายสรายุทธ กรวิรัตน์ รหัสประจำตัว 46010811

อาจารย์ที่ปรึกษา

(รศ.ดร. เอื้อน ปิ่นเงิน)

การพัฒนาระบบเพื่อตรวจสอบกฎในฐานความรู้

นายสรายุทธ กรวิรัตน์ 46010811
รศ.ดร. เอื้อน ปิ่นเงิน อาจารย์ที่ปรึกษา
ปีการศึกษา 2549

บทคัดย่อ

ระบบฐานกฎความรู้(Rule-Based Systems)เป็นระบบฐานความรู้(Knowledge-Based Systems)ที่เก็บความรู้เป็นกฎ(Rule) ซึ่งกฎในฐานกฎความรู้อาจมีจำนวนมากและบางครั้งอาจไม่มีความสามารถ(Competence) เนื่องจากการขาดความกลมกลืนของกฎซึ่งกันและกัน(Consistency) เช่น การซ้ำซ้อนของกฎหรือการขัดแย้งกันของกฎ เป็นต้น หรืออาจไม่มีความสามารถเนื่องจากการขาดความสมบูรณ์ของส่วนประกอบของกฎในฐานกฎความรู้(Completeness) เช่น ค่าของแอททริบิวท์(Attribute)ที่ไม่มีการอ้างถึงในฐานกฎความรู้หรือข้อสรุปของกฎไม่สามารถทำให้ค้นหาได้โดยกลไกการอนุมาน(Inference engine) เป็นต้น

จากที่กล่าวมาฐานกฎความรู้ยากต่อการตรวจสอบกฎด้วยมนุษย์เพราะอาจมีกฎเป็นจำนวนมาก ดังนั้นเพื่อทำให้กฎในฐานกฎความรู้มีความสมบูรณ์จึงต้องมีระบบตรวจสอบกฎในฐานกฎความรู้ ซึ่งระบบตรวจสอบกฎในฐานความรู้จะมีหน้าที่แก้ไขความไม่สมบูรณ์นั้นหรือแจ้งต่อผู้ใช้ระบบให้ทราบในกรณีแก้ไขไม่ได้เพื่ออำนวยความสะดวกในการสร้างกฎในฐานกฎความรู้

VALIDATION OF RULE – BASED SYSTEM

Mr. Sarayut Gonwirat 46010811

Assoc. Prof Ouen Pinngern Advisor

Academic Year 2006

ABSTRACT

Due to the fact that Rule-Based Systems is Knowledge-Based Systems which keeps knowledge as a rule. That rule in base of knowledge rule may have a lots and sometimes none of competence because of lacking of consistency such as redundancy of rules or confliction of rules etc. Or may not have ability due to lacking of completeness such as attribute's value which do not refer to rule-base or conclusion of rule that cause to can not fire by inference process etc.

Above-mentioned, rule-base is difficult for validating rule by human because of a lot of rule. For rule in rule-base complete that must have system which can validate and verify rule in rule-base or alarms user when problem unavailable for verifying.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี ด้วยคำแนะนำ และคำปรึกษาจาก รศ.ดร. เอื้อน ปิ่นเงิน ซึ่งเป็นอาจารย์ผู้ควบคุมปริญญานิพนธ์ ที่ให้ความเอื้อเฟื้อในการสอน ให้การทำงานเป็นไปอย่างมีขั้นตอน ทำให้ผู้จัดทำมีความกระตือรือร้นและทำงานอย่างเป็นระบบมากขึ้น ข้าพเจ้ารู้สึกทราบซึ้งในความอนุเคราะห์จากท่านอาจารย์ และขอขอบพระคุณเป็นอย่างสูง

ขอกราบพระคุณคณาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุก ๆ ท่านที่ได้ประสิทธิ์ประสาทวิชาให้กับข้าพเจ้า

ขอขอบคุณพี่ๆในห้อง Lab ReCCIT สถาบันเทคโนโลยี พระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกคนที่ให้คำแนะนำต่างๆ

ขอขอบคุณเพื่อนๆ พี่ๆ น้องๆ ในภาควิชาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยี พระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกคนที่ให้คำแนะนำต่างๆ และคอยให้กำลังใจเสมอมา

สุดท้ายนี้ข้าพเจ้าขอกราบขอบพระคุณ บิดา มารดา และครอบครัวของข้าพเจ้าที่เป็นกำลังใจและให้การสนับสนุนในทุกเรื่องๆ ทำให้ข้าพเจ้าสามารถทำปริญญานิพนธ์ฉบับนี้ลุล่วงด้วยดี

คุณค่าและประโยชน์อันพึงมาจากปริญญานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่าน

นายสรายุทธ กรวิรรัตน์

สารบัญ

หน้าที่

หน้าที่

หน้าที่

สารบัญภาพ

หน้าที่

หน้าที่

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของโครงงาน

การที่กฎ (Rule) ในระบบฐานกฎความรู้ (Rule-Based Systems) เกิดความไม่สมบูรณ์นั้น สาเหตุส่วนหนึ่งเกิดจากการที่ผู้สร้างระบบฐานกฎความรู้นำเข้ากฎที่ทำให้เกิดความไม่สมบูรณ์ขึ้น กฎที่มีความไม่สมบูรณ์นั้นอาจเป็นกฎที่ไม่มีประโยชน์ต่อระบบฐานกฎความรู้หรืออาจจะทำให้ระบบฐานกฎความรู้เกิดความผิดพลาดได้ ซึ่งสาเหตุนี้สามารถแก้ไขได้โดยผู้สร้างระบบฐานกฎ-ความรู้ตรวจสอบกฎในระบบฐานกฎความรู้ให้ดีก่อนการนำเข้า ซึ่งสามารถทำได้ถ้าระบบฐานกฎความรู้มีกฎจำนวนที่สามารถตรวจสอบเองได้ แต่ถ้าระบบฐานกฎความรู้มีกฎจำนวนมากทำให้การตรวจสอบเองเป็นเรื่องที่ยากลำบากและเสียเวลามาก ดังนั้นเพื่อให้การตรวจสอบเป็นเรื่องที่ง่าย จึงได้มีแนวคิดการสร้างเครื่องมือที่สามารถตรวจสอบความไม่สมบรูณ์ของกฎขึ้นมา เพื่อที่สามารถเตือนความไม่สมบูรณ์ของกฎเองโดยอัตโนมัติก่อนที่จะนำเข้า จนถึงสามารถแก้ไขกฎในฐานกฎความรู้ให้มีความถูกต้องเองได้เพื่ออำนวยความสะดวกผู้สร้างฐานกฎความรู้ จากที่ได้กล่าวมาระบบตรวจสอบกฎในฐานกฎความรู้ (Validation of Rule-Based Systems) ทำให้ระบบฐานกฎความรู้มีความสมบูรณ์มากขึ้นเนื่องจากกฎมีความสมบูรณ์มากขึ้น

1.2 วัตถุประสงค์ของโครงงาน

ปริญญานิพนธ์ฉบับนี้มุ่งหวังเพื่อศึกษาหลักการของรูปแบบตรวจสอบกฎในฐานกฎความรู้ และนำหลักการของรูปแบบตรวจสอบกฎในฐานกฎความรู้มาประยุกต์ใช้ออกแบบและพัฒนาสร้างให้เป็นระบบตรวจสอบกฎในฐานกฎความรู้ที่สามารถตรวจสอบกฎในฐานกฎความรู้ได้จริง และเพื่อให้กฎในฐานกฎความรู้เป็นกฎที่มีความสมบูรณ์มากขึ้น

1.3 ขอบเขตของโครงงาน

1.3.1 รูปแบบความไม่สมบูรณ์ของกฎในฐานกฎความรู้ที่เกี่ยวข้องกับโครงงาน มี 2 รูปแบบคือ กฎขาดความกลมกลืน (Inconsistency)และส่วนประกอบของกฎไม่มีความสมบูรณ์ (Incompleteness)

1.3.2 รูปแบบของกฎที่เกี่ยวข้องกับโครงงานเป็นแบบกฎที่แน่ใจ (Certainty rule)

1.3.3 รูปแบบที่ใช้แสดงกฎที่เกี่ยวข้องกับโครงงานเป็นแบบ Production rule ในรูปแบบ OV

1.3.4 รูปแบบกฎส่วนของ Premise Clause เชื่อมด้วย Conjunctive และมีหนึ่ง Conclusion Clause

1.4 วิธีการดำเนินการ

1.4.1 ศึกษาระบบฐานกฎความรู้

1.4.2 ศึกษารูปแบบของกฎในฐานกฎความรู้ต่างๆ

1.4.3 ศึกษารูปแบบของกฎในฐานกฎความรู้ที่ขาดความสมบูรณ์

1.4.4 ศึกษารูปแบบที่ใช้การแสดงกฎเป็นแบบ Production rule ในรูปแบบต่างๆ และรูปแบบการแสดงความรู้(Knowledge Representation)รูปแบบอื่น เช่น Semantic-Network

1.4.5 ออกแบบขั้นตอนการตรวจสอบกฎในฐานกฎความรู้

1.4.6 ออกแบบฟังก์ชั่นการตรวจสอบกฎตามขั้นการตรวจสอบที่ออกแบบมาโดยใช้ Production rule ในรูปแบบ OV ทดลองสร้างโปรแกรมตรวจสอบกฎด้วยกฎที่จำลองขึ้นและทดสอบ

1.4.7 ออกแบบแอพพลิเคชั่นของระบบเพื่อตรวจสอบกฎในฐานกฎความรู้ โดยใช้ Production rule ในรูปแบบ OV ทดลองสร้างแอพพลิเคชั่นกฎด้วยกฎที่จำลองขึ้น และทดสอบ

1.4.8 ศึกษาแนวทางการสร้างให้แอพพลิเคชั่นสามารถขยายให้รับรูปแบบกฎและการแสดงของกฎให้ได้หลายรูปแบบให้ได้สูงสุด

1.4.9 ดำเนินการขยายให้แอพพลิเคชั่นให้ใช้ประโยชน์ได้สูงสุด

1.5 ประโยชน์ที่คาดจะได้รับ

1.5.1 ทำให้ได้ระบบที่สามารถตรวจสอบกฎในฐานกฎความรู้สร้างความสะดวกและประโยชน์แก่ผู้ใช้ระบบตรวจสอบ

1.5.2 ทำให้ได้กฎในฐานกฎความรู้มีความสมบูรณ์

1.5.3 ทำให้ได้ระบบฐานกฎความรู้มีความสมบูรณ์เนื่องจากมีระบบตรวจสอบกฎในฐานกฎความรู้

1.6 ส่วนประกอบของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้ได้แบ่งเนื้อหาออกเป็น 7 บทด้วยกันคือ

- บทที่ 1 กล่าวถึงความสำคัญและที่มาของโครงงาน วัตถุประสงค์ของโครงงาน ขอบเขตของโครงงาน วิธีการดำเนินการ ประโยชน์ที่คาดว่าจะได้รับและส่วนประกอบของปริญญานิพนธ์
- บทที่ 2 กล่าวถึงระบบฐานกฎความรู้ โครงสร้างของระบบผู้เชี่ยวชาญฐานกฎความรู้และ Production rule
- บทที่ 3 กล่าวถึงรูปแบบตรวจสอบเพื่อความกลมกลืนและตรวจสอบเพื่อความสมบูรณ์
- บทที่ 4 กล่าวถึงวัตถุกับฐานกฎความรู้ ความสัมพันธ์ OAV และการกำหนดเซ็ตรูปแบบการตรวจสอบ
- บทที่ 5 กล่าวถึงการออกแบบและการพัฒนาระบบตรวจสอบกฎในฐานความรู้ Use Case Diagram Class Diagram และER Diagram
- บทที่ 6 ผลการทดลอง
- บทที่ 7 สรุปและวิจารณ์

บทที่ 2

ระบบฐานกฎความรู้ (Rule-Based Systems)

2.1 ระบบฐานกฎความรู้

ความรู้ (Knowledge) คือ ค่าของตัวแปรหรือปัจจัยและกฎที่ได้รับการเรียนรู้ (Learned)หรือประเมิน (Estimated) มาจากข้อมูล

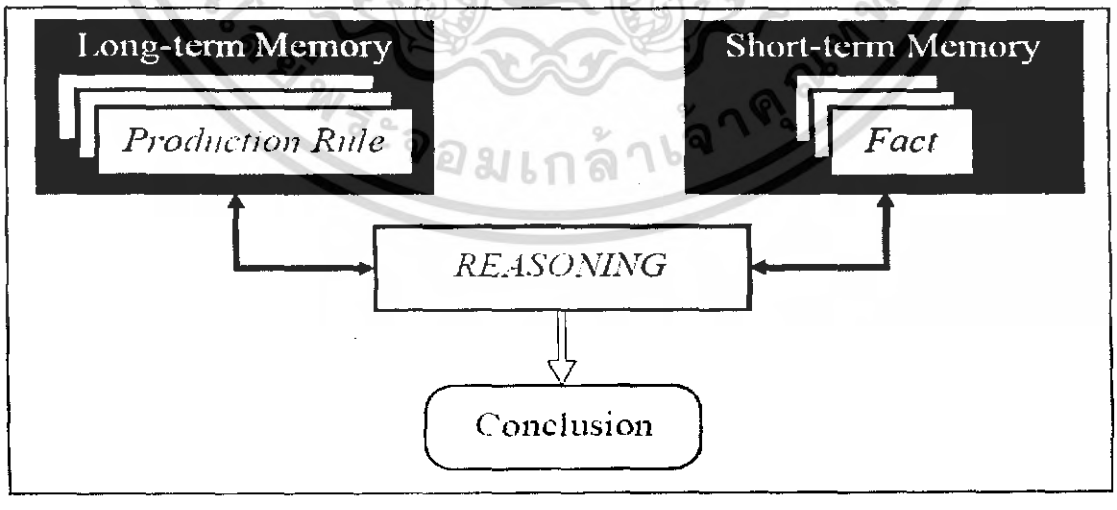
ระบบฐานความรู้ (Knowledge – Based Systems) คือ ระบบที่มีความรู้(Knowledge)ที่ใช้แก้ปัญหาที่เกี่ยวกับปัญหาที่สนใจ

ระบบฐานกฎความรู้ (Rule – Based Systems) คือ ระบบที่มีความรู้(Knowledge)ที่ใช้แก้ปัญหาเกี่ยวกับปัญหาที่สนใจ ซึ่งมีลักษณะการเก็บความรู้เป็นกฎ(Rule) ที่มีการแสดงเป็น IF ... THEN ... ระบบฐานกฎความรู้ได้รับความนิยมมากในสร้างระบบฐานความรู้

2.2 โครงสร้างของระบบผู้เชี่ยวชาญฐานกฎความรู้ (Structure of Rule-Based Expert Systems)

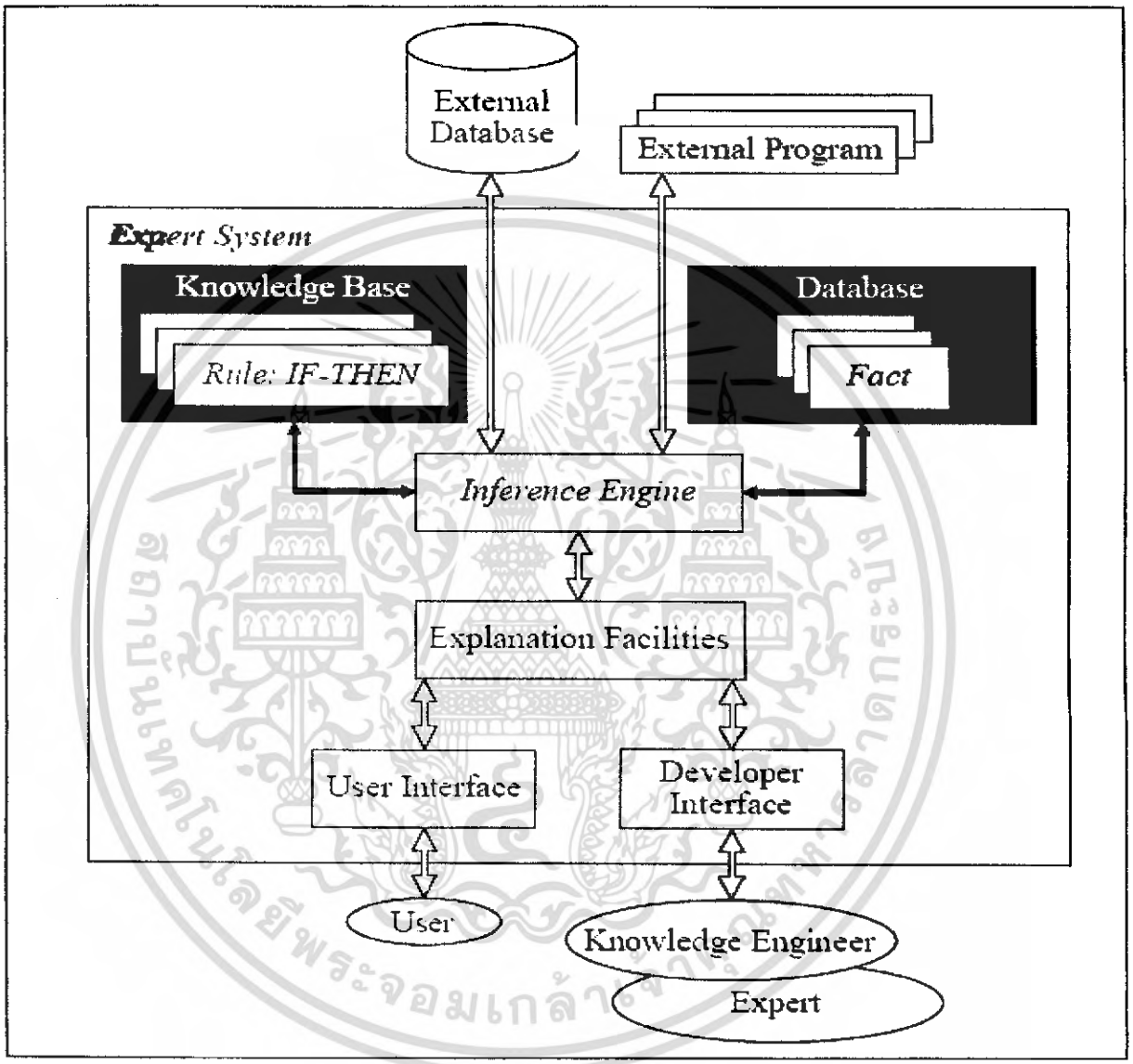
2.2.1 ระบบผู้เชี่ยวชาญ(Expert System) คือ ระบบคอมพิวเตอร์ที่สามารถจำลองความคิดที่มีความรู้ของมนุษย์ผู้ที่เชี่ยวชาญในปัญหาที่สนใจ

2.2.2 Production model คือ การนำเอาความคิดของมนุษย์ที่ใช้ในการแก้ปัญหาที่เราสนใจมาใช้โดยมีนำเอาความรู้มาประกอบ(ความรู้ที่เก็บเป็น Production rule)โดยการดึงเอาความรู้มาใช้ขึ้นอยู่กับเหตุการณ์ที่ได้รับเข้ามา(Facts)



รูปที่ 2.1 แสดงระบบ Production model

จากรูปที่ 2.1 แสดงระบบ Production model ที่ประกอบด้วย Production rule ที่เก็บอยู่ใน Long-term Memory คือ ความจำที่ฝังลึก เช่น ภาษาพูด เป็นต้นและ Facts ที่เก็บอยู่ใน Short-term Memory คือความจำ ที่เกิดขึ้นมาเฉพาะเหตุการณ์ไม่ถาวร เช่น เมื่อขับรถไปเจอไฟแดง เป็นต้น เมื่อรวมกันจะได้เหตุผลที่ใช้ในการตัดสินใจ



รูปที่ 2.2 แสดงโครงสร้างแบบสมบูรณ์ของระบบผู้เชี่ยวชาญฐานกฎความรู้

2.2.3 ส่วนโครงสร้างแบบสมบูรณ์ของระบบผู้เชี่ยวชาญฐานกฎความรู้ ประกอบด้วย ดังรูปที่ 2.2

2.2.3.1 ฐานความรู้(Knowledge Base) จะเก็บความรู้ที่ใช้ในการหาคำตอบที่เกี่ยวข้องกับเรื่องที่สนใจโดยเก็บเป็นเซ็ตของกฎ (Rule : IF (Condition) THEN (Action)) เมื่อเงื่อนไขในส่วนหลัง IF เป็นจริง ก็จะทำส่วนหลัง THEN

2.2.3.2 ฐานข้อมูล(Database) จะเก็บเซ็ตของข้อเท็จของเหตุการณ์ที่เกี่ยวข้องกับปัญหา(Fact)ที่จะใช้จับคู่ กับส่วนหลัง IF ของกฎในฐานความรู้

2.2.3.3 ส่วนกลไกอนุมาน(Inference engine) คือ สิ่งที่สามารถหาเหตุผล ซึ่งระบบผู้เชี่ยวชาญ ใช้หาคำ ตอบมีทั้งแบบเหตุมาหาผล(Forward Chaining)และผลมาหาเหตุ (Backward Chaining)

2.2.3.4 ส่วนอธิบายความ(Explanation Facilities) เป็นส่วนที่ผู้ใช้สามารถถามระบบผู้เชี่ยวชาญเพื่อหาข้อสรุปที่ต้องการและข้อเท็จจริงตั้งต้นที่ต้องการ โดยที่ระบบผู้เชี่ยวชาญจะต้องสามารถอธิบายเหตุผลและข้อพิสูจน์สิ่งที่ได้แนะนำ พิจารณาหรือข้อสรุปนั้นได้

2.2.3.5 ส่วนติดต่อผู้ใช้ระบบ(User interface) เป็นส่วนที่ไว้สื่อสารระหว่างผู้ใช้กับระบบ

จากส่วนประกอบข้างบนห้าส่วนเป็นส่วนประกอบหลักที่จำเป็นสำหรับระบบผู้เชี่ยวชาญ ฐานกฎความรู้ จะมีเพิ่มจากนี้เล็กน้อย คือส่วน **ฐานข้อมูลภายนอก**(External Database) โปรแกรมภายนอก(External Program)และส่วนติดต่อผู้พัฒนาระบบ(Developer interface) เป็นส่วนที่วิศวกรความรู้(Knowledge Engineer) และผู้เชี่ยวชาญที่เกี่ยวกับปัญหา (Expert) สามารถใช้ปรับปรุงและตรวจสอบระบบได้

2.3 Production rule

Production rule คือ การประกาศที่แสดงในรูปแบบ IF (Premise) THEN (Conclusion) ถ้าเงื่อนไข (Premise) เป็นจริงแล้วข้อสรุป(Conclusion) เป็นจริง

Premise, Condition หรือ Antecedent คือ เงื่อนไขที่อยู่ในส่วน IF ของกฎ

Conclusion, Action หรือ Consequent คือ ข้อสรุปหรือการกระทำ(Action)ในส่วนของ THEN ของกฎ

Production rule เป็นรูปแบบที่ใช้แสดงกฎในฐานกฎความรู้(Rule Base)โดยรูปแบบทั่วไปของกฎจะเป็น IF-THEN อย่างไรก็ตาม บางครั้งก็จะมีกฎรูปแบบ IF-THEN-ELSE ดังแสดงตัวอย่างที่ 2.1

ตัวอย่างที่ 2.1 **Rule** : IF the student's GRE score is 1350 or more

THEN admit the student to the graduate program.

ELSE Then accept into honor society.

ซึ่งเท่ากับกฎรูปแบบ IF-THEN 2 กฎ ดังแสดงตัวอย่างที่ 2.2

ตัวอย่างที่ 2.2 **Rule A** : IF the student's GRE score is 1350 or more
THEN admit the student to the graduate program.
Rule B : IF the student's GRE score is 1350 or more
THEN accept into honor society.

โดยทั่วไปจะเจาะจงเฉพาะรูปแบบ IF-THEN และหลีกเลี่ยงการใช้รูปแบบ IF-THEN-ELSE ด้วยเหตุผลดังนี้

- เครื่องมือที่ใช้พัฒนาระบบผู้เชี่ยวชาญส่วนใหญ่จะใช้รูปแบบ IF-THEN
- การตรวจสอบกฎในรูปแบบ IF-THEN-ELSE จะทำได้ยากกว่าใช้รูปแบบ IF-THEN
- ในกระบวนการกลไกการอนุมาน กฎจะมีแนวโน้มค้นหาข้อสรุป (Conclusion) ทุกครั้ง

เหตุผลที่ใช้กฎในรูปแบบส่วนของ Premise Clause เชื่อมด้วย Conjunctive และมีหนึ่ง Conclusion Clause

- **การปรับเปลี่ยนกฎที่เชื่อมด้วย OR ในส่วนของ Premises โดยการแยกกฎเป็นหลายกฎ (Rule Conversion Disjunctive Clause)**

ตัวอย่างที่ 2.3 Rule 1: *IF* B is true
OR A is true
THEN Y is true

จากตัวอย่างที่ 2.3 แยกออกเป็น 2 กฎได้ดังตัวอย่างที่ 2.4 ข้างล่าง

ตัวอย่างที่ 2.4 Rule 1a: *IF* B is true
THEN Y is true

Rule 1b: *IF* A is true
THEN Y is true

การแยกกฎที่เชื่อมด้วย OR อาจจะไม่มีความจำเป็นกับระบบ Rule – Based System ทั่วไป แต่จะทำให้ง่ายต่อทำกระบวนการ inference เช่น การทำ Rule Checking ที่ทำได้เพียงรูปแบบกฎที่มีการเชื่อมด้วย AND (Conjunctive) ในส่วนของ Premises

- **การหลีกเลี่ยงการมีหลาย Conclusion ในกฎเดียวกัน**

(Conjunctive Conclusion Clauses : Should They Be Avoided)

เนื่องจากการจัดกลุ่มกฎ(Rule Grouping) มีประโยชน์ คือ ทำให้เข้าใจกฎง่าย ง่ายต่อการ-ปรับแก้ การตรวจสอบกฎและการทำให้ง่ายต่อการทำเอกสารประกอบ เพราะฉะนั้นจึงต้องมีการทำให้กฎมีเพียง Conclusion เดียว เพื่อให้ง่ายต่อการจัดกลุ่ม

ตัวอย่างที่ 2.5 Rule 1: *IF* B is true
THEN A is true *AND* Y is true

ตัวอย่างที่ 2 .6 Rule A: *IF* X is true Rule B: *IF* Z is true

THEN A is false *THEN* Y is false

จากกฎตัวอย่างที่ 2.5 ยากที่จะจัดเข้ากลุ่มของกฎ (Rule Grouping) กับกลุ่มกฎในตัวอย่างที่ 2.6 จึงต้องมีการแยกเป็นกฎที่มี Conclusion เดียว ดังตัวอย่างที่ 2.7 เป็นการแยกกฎในตัวอย่างที่ 2.5

ตัวอย่างที่ 2.7 Rule 1a: *IF* B is true Rule 1b: *IF* B is true

THEN A is true *THEN* Y is true

จากกฎตัวอย่างที่ 2.7 อาจจัดกฎ 1a กับกฎ A ในตัวอย่างที่ 2.6 เข้าเป็นกลุ่มเดียวกันและกฎ 1b กับกฎ B ในตัวอย่างที่ 2.6 เข้าเป็นกลุ่มเดียวกัน จะเห็นได้ว่ากฎตัวอย่างที่ 2.7 จะง่ายต่อการจัดกลุ่มเข้ากับกลุ่มกฎในตัวอย่างที่ 2.6

บทที่ 3

การตรวจสอบกฎในฐานกฎความรู้ (Rule-Based Validation)

การตรวจสอบกฎในฐานกฎความรู้เพื่อให้กฎมีความสมบูรณ์ เช่น ไม่ขัดแย้งกัน ซ้ำซ้อนกัน หรือเป็นกฎที่สามารถใช้ประโยชน์ได้ เป็นต้น แบ่งประเภทการตรวจเป็น 2 ประเภท คือ ตรวจสอบเพื่อความกลมกลืน(Check for Consistency)และตรวจสอบเพื่อความสมบูรณ์ (Check for Completeness)

3.1 ตรวจสอบเพื่อความกลมกลืน(Check for Consistency)

ประกอบด้วย 5 รูปแบบ ได้แก่

- กฎที่มีการซ้ำซ้อนกัน (Redundant Rules)
- กฎที่มีการขัดแย้งกัน (Conflicting Rules)
- กฎที่สามารถรวบรวมเป็นในกฎเดียวกัน (Subsumed Rules)
- กฎที่มีอนุประโยคเงื่อนไขที่ไม่จำเป็น (Unnecessary Premise Clause)
- กฎเป็นโซ่วงกลม (Circular Rule Chains)

กำหนดให้

Rule : *IF* Premise Clause *THEN* Conclusion Clause

- {P(k)} คือ เซ็ตของอนุประโยคเงื่อนไข (Premise Clauses) ของ Rule k
- {C(k)} คือ เซ็ตของอนุประโยคข้อสรุป (Conclusion Clauses) ของ Rule k
- p(k) คือ เป็นหนึ่งอนุประโยคเงื่อนไข (Premise Clauses)ในเซ็ตของ {P(k)}
- c(k) คือ เป็นหนึ่งอนุประโยคข้อสรุป (Conclusion Clauses) ในเซ็ตของ {C(k)}
- {S} = {T} คือ เซ็ต S เท่ากับ เซ็ต T

3.1.1 กฎที่มีการซ้ำซ้อนกัน (Redundant Rules)

คือมีกฎที่อนุประโยคเงื่อนไข(Premise Clauses)เหมือนกันทั้งหมดแต่มีอนุประโยคข้อสรุป (Conclusion clauses) ของกฎหนึ่ง เป็นสับเซ็ตของอีกกฎ ดังแสดงตัวอย่างที่ 3.1

ตัวอย่างที่ 3.1

Rule r		Rule k	
IF	A=X	*IF*	B=Y
AND	B=Y	*AND*	A=X
THEN	C=Z	*THEN*	C=Z
			D=W

จากตัวอย่างที่ 3.1 จะเห็นได้ว่า {P(r)} = {P(k)} และ {C(r)} รวมอยู่ใน {C(s)}
ในกรณีจะทำการลบ Rule r

3.1.2 กฎที่มีการขัดแย้งกัน (Conflicting Rules)

คือ มีกฎที่อนุประโยคเงื่อนไข (Premise Clauses) เหมือนกันทั้งหมดแต่มีอนุประโยคข้อสรุป (Conclusion Clauses) ทั้งสองกฎขัดแย้งกัน ดังแสดงตัวอย่างที่ 3.2

ตัวอย่างที่ 3.2 Rule r : *IF* A = X *AND* B=Y *THEN* C=Z

Rule k : *IF* B=Y *AND* A=X *THEN* C=D

จากตัวอย่างที่ 3.2 จะเห็นได้ว่า {P(r)} = {P(k)} และ {C(r)} ขัดแย้ง{C(s)}
ในกรณีวิศวกรความรู้ (Knowledge Engineer) จะทำการพิจารณาทบทวน

3.1.3 กฎที่สามารถรวบรวมเป็นกฎเดียวกัน (Subsumed Rules)

คือ มีกฎที่มีอนุประโยคข้อสรุป (Conclusion Clauses) เหมือนกันทั้งหมด แต่มีกฎที่มีอนุประโยคเงื่อนไข (Premise Clauses) ที่ไม่มีความจำเป็นอยู่ในกฎ ดังแสดงตัวอย่างที่ 3.3

ตัวอย่างที่ 3.3 Rule r : *IF* A=X *AND* B=Y *THEN* C=Z

Rule k : *IF* B=Y *THEN* C=Z

จากตัวอย่างที่ 3.3 จะเห็นได้ว่า {C(r)} = {C (k)} และ {P (k)} เป็นสับเซ็ต {P(r)}
ในกรณีจะทำการลบ Rule r

3.1.4 กฎที่มีอนุประโยคเงื่อนไขที่ไม่จำเป็น (Unnecessary Premise Clause)

จากตัวอย่างข้างล่าง มีกฎที่มีอนุประโยคข้อสรุป(Conclusion Clauses) เหมือนกันทั้งหมด แต่มีบาง อนุประโยคเงื่อนไข (Premise clauses) ที่ไม่จำเป็นอยู่ขัดแย้งกันอยู่ ดังแสดงตัวอย่างที่ 3.4

ตัวอย่างที่ 3.4 Rule r : *IF* A=X *AND* B=Y *THEN* C=D

Rule k : *IF* B= NOT Y *AND* A=X *THEN* C=D

จากตัวอย่างที่ 3.4 จะเห็นได้ว่า {C(r)} = {C (k)} และ
p(r) ขัดแย้ง p(s) และที่เหลือทั้งหมดเหมือนกัน
ในกรณีจะลบกฎทั้งสองทิ้งจะทำได้ กฎใหม่ ดังแสดงตัวอย่างที่ 3.5

ตัวอย่างที่ 3.5 *IF* A=X *THEN* C=D

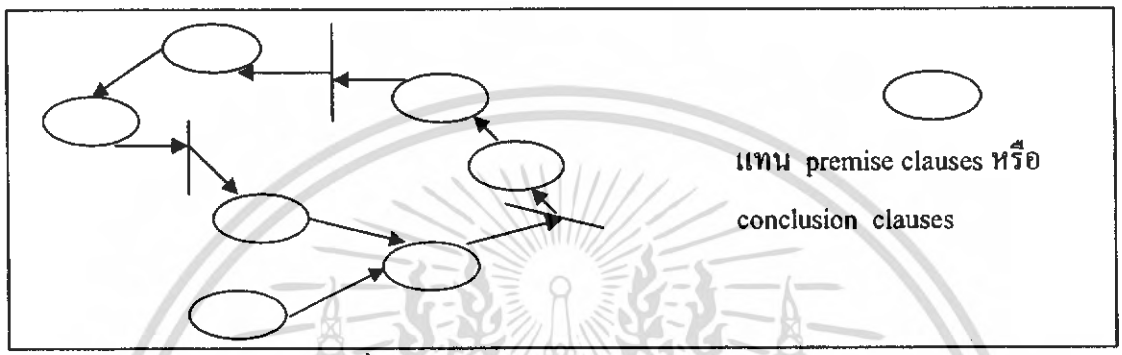
3.1.5 กฎเป็นโซ่วงกลม (Circular Rule Chains)

จากตัวอย่างกฎข้างล่างมีอนุประโยคข้อสรุป (Conclusion Clauses) จะเป็นอนุประโยคเงื่อนไข (Premise Clauses) ของอีกกฎได้ในลักษณะเป็นวงกลม ถ้าใช้ตัวนำกลไกการอนุมานแบบผลหาเหตุ(Backward Chaining) อาจทำให้เกิดลูป ดังรูปที่รูป 3.1 และดังแสดงตัวอย่างที่ 3.6

ตัวอย่างที่ 3.6 Rule r : *IF* A=X *THEN* B=Y

Rule k : *IF* B= Y *AND* Z=C *THEN* A=X

Rule s : *IF* decision = yes *THEN* decision = yes

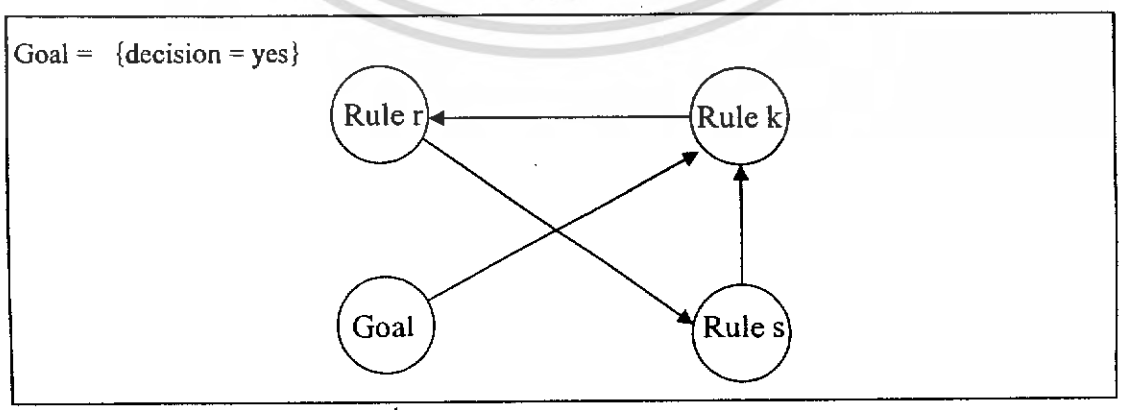


รูปที่ 3.1 แสดงกฎเป็นโซ่วงกลม (Circular rule chains)

การตรวจสอบต้องใช้ Rule Dependency Graph โดย

- กำหนด Node เป้าประสงค์ (Goal) และ Node แต่ละ กฎ
- ลากลูกศรที่มีอนุประโยคเงื่อนไข (Premise clauses) ที่เหมือนกับมีอนุประโยคข้อสรุป (Conclusion clauses) โดยทิศทางชี้ไปที่อนุประโยคข้อสรุป(Conclusion clauses)
- ลากลูกศรที่จาก Node เป้าประสงค์ (Goal) ที่เป้าประสงค์ (Goal) เหมือนกับอนุ-ประโยคข้อสรุป(Conclusion clauses)ของ node นั้น โดยทิศทางชี้ไปที่ ทิศทางจาก Goal ไปกฎ

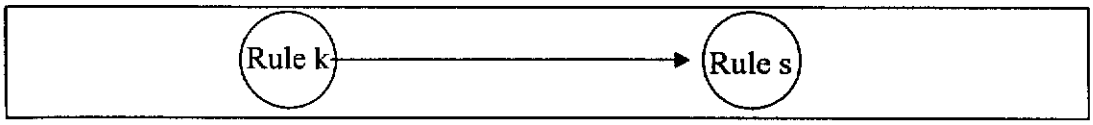
จากตัวอย่างที่ 3.5 ข้างบนจะได้ดังรูปที่ 3.2



รูปที่ 3.2 แสดง Rule Dependency Graph

จากรูปที่ 3.2 จะเห็นได้ว่าจะเกิดเป็นลูกศรวิ่งเป็นวงกลม

ถ้าเราทำการกลับหัวลูกศร จาก Rule k ไป Rule s จะทำไม่เกิดกฎขึ้นมา ดังรูปที่ 3.3



รูปที่ 3.3 แสดง การกลับหัวลูกศร

หมายความว่าอนุประโยคเงื่อนไข(Premise clauses) ของ Rules จะไม่เหมือนอนุประโยคข้อสรุป (Conclusion clauses) ของ Rule k อีก จึงทำการย้าย Z=C ของ Rule k ไปแทน decision = yes ของ Rule s และอนุประโยคเงื่อนไข (Premise clauses) ของ Rule k จะต้องเหมือนอนุประโยคข้อสรุป (Conclusion clauses) ของ Rule s จึงเพิ่ม B=Y ไปในอนุประโยคข้อสรุป (Conclusion clauses) ของ Rule s จะได้ rule ใหม่ที่ไม่เกิดลูปเมื่อกลับหัวลูกศรจะได้กฎดังแสดงตัวอย่างที่ 3.6

ตัวอย่างที่ 3.7

Rule r : *IF* A=X *THEN* B=Y

Rule k : *IF* B= Y *THEN* decision = yes

Rule s : *IF* Z=C *THEN* A=X B= Y

3.2 ตรวจสอบเพื่อความสมบูรณ์ (Check for Completeness)

- ค่าแอททริบิวท์ที่ค่าไม่ได้อ้าง (Unreferenced Attribute Values)
- ค่าแอททริบิวท์ที่ค่าผิด (Illegal Attribute Values)
- ข้อสรุปที่เป็นตัวกลางที่ไม่สามารถสำเร็จได้ (Unachievable Intermediate Conclusions)
- ข้อสรุปที่เป็นสุดท้ายที่ไม่สามารถสำเร็จได้ (Unachievable (final) Conclusions, or Goal)
- เงื่อนไขที่ไม่สามารถสำเร็จได้ (Unachievable Premises)

3.2.1 ค่าแอททริบิวท์ที่ค่าไม่ได้อ้าง (Unreferenced Attribute Values)

จากตัวอย่างที่ 3.8 ทั้งสองแอททริบิวท์ interest rates มีค่า low และ high แต่เนื่องจากได้กำหนดค่าไว้ก่อนมี 3 ค่าได้แก่ low , high และ medium ซึ่งแอททริบิวท์ interest rates ไม่ปรากฏค่า medium ซึ่งจะทำให้ไม่สามารถหาข้อสรุปของแอททริบิวท์interest ratesมีค่าmediumได้จะทำการค้นหาผิดพลาดได้ เพราะฉะ นั้น การตรวจสอบกฎจะต้องสามารถเตือนวิศวกรความรู้

(Knowledge Engineer) ได้เมื่อทำการตรวจสอบพบว่าไม่มีค่าของแอททริบิวท์ ในเซ็ตของกฎ และทำการแก้ไข

ตัวอย่างที่ 3.8 Rule r : *IF* interest rates are high *THEN* invest in bonds

Rule k : *IF* interest rates are low *THEN* invest in stocks

3.2.2 ค่าแอททริบิวท์ที่ค่าผิด (Illegal Attribute Values)

จากตัวอย่างที่ 3.9 กฎทั้งสองแอททริบิวท์ interest rates มีค่า very high และแอททริบิวท์ inflation มีค่า hi ซึ่งไม่ใช่ค่าของ แอททริบิวท์ที่ถูกต้อง เพราะฉะนั้น การตรวจสอบฐานกฎความรู้ (Rule - Based Validation) จะต้องสามารถบอกได้

ตัวอย่างที่ 3.9 Rule r : *IF* interest rates are very high *THEN* invest in stocks

Rule k : *IF* inflation is hi *THEN* invest in stocks

3.2.3 ข้อสรุปที่เป็นตัวกลางที่ไม่สามารถสำเร็จได้ (Unachievable Intermediate Conclusions)

ข้อสรุปที่เป็นตัวกลางที่ไม่สามารถสำเร็จได้ คือ มีอนุประโยคข้อสรุปที่เป็นตัวกลาง (Intermediate Conclusion) ที่ไม่เกิดในหรือไม่ตรงกับอนุประโยคเงื่อนไข (Premise Clauses) ของกฎใดในเซ็ตกฎ

อนุประโยคข้อสรุปที่เป็นตัวกลาง(Intermediate Conclusion) คือ อนุประโยคข้อสรุป (Conclusion Clauses) ที่จะต้องพบในอนุประโยคเงื่อนไข (Premise clauses) ของกฎใดในเซ็ตกฎที่ ไม่ใช่อนุประโยคข้อสรุปที่เป็นสุดท้าย (Final Conclusion)

จากตัวอย่างที่ 3.10 กฎที่ 5 เป็นข้อสรุปที่เป็นตัวกลางที่ไม่สามารถสำเร็จได้

ตัวอย่างที่ 3.10

Rule 1: *IF* B is true
THEN Y is true

Rule 2: *IF* C is true
THEN B is true

Rule 3: *IF* C is false
THEN B is false

Rule 4: *IF* A is true
THEN Y is true

Rule 5: *IF* C is true
THEN D is true

3.2.4 ข้อสรุปที่เป็นสุดท้ายที่ไม่สามารถสำเร็จได้ (Unachievable (final) Conclusions, or Goal)

ข้อสรุปที่เป็นสุดท้ายที่ไม่สามารถสำเร็จได้ คือ ไม่มีในเงื่อนไขที่สอบถาม(Query Premise) ของเป้าประสงค์ (Goal) และอนุประโยคเงื่อนไข (Premise Clauses) ของกฎของเป้าประสงค์ (Goal) ไม่สามารถหาจากอนุประโยคข้อสรุป (Conclusion Clauses) จากกฎอื่นได้ จากตัวอย่างที่ 3.11 เป็นเซ็ตกฎที่ไม่สิ่งที่สอบถาม(Query Premise)ของแอททริบิวท์ E ซึ่งเป็นกฎที่เป็นเป้าประสงค์ และไม่สามารถหาค่าแอททริบิวท์ E จากกฎอื่นได้ในเซ็ตกฎ

ตัวอย่างที่ 3.11

Rule r : *IF* A=X *THEN* C=W

Rule k : *IF* C= W *THEN* D =X

Rule s : *IF* E=Q *THEN* goal = yes

3.2.5 เงื่อนไขที่ไม่สามารถสำเร็จได้ (Unachievable Premises)

เงื่อนไขที่ไม่สามารถสำเร็จได้ คือ ไม่มีเป้าประสงค์(Goal)และอนุประโยคเงื่อนไข(Premise Clauses) ไม่สามารถหาจาก อนุประโยคข้อสรุป(Conclusion clauses)จากกฎอื่นได้

จากตัวอย่างที่ 3.11 เป็นเซ็ตกฎที่ไม่สิ่งที่สอบถาม(Query Premise)ของแอททริบิวท์ E และไม่สามารถ หาค่าแอททริบิวท์ E จากกฎอื่นได้ในเซ็ตกฎ

3.3 รูปแบบของเซ็ตค่าที่เป็นไปได้ใน Clause ระหว่าง 2 Clause

รูปแบบของเซ็ตค่าที่เป็นไปได้ใน Clause ระหว่าง 2 Clause ที่มี Object เหมือนกันจะมีผลทำให้ 2 Clause นั้น ซ้ำซ้อนกัน(Redundant Clause), ขัดแย้งกัน(Conflict Clause)และซับซัมอีก Clause (Subsumed Clause)

กำหนดให้ $CA_1, CA_2, \ldots CA_n$ คือ Clause ที่มี Object เป็น Object A

$SCA_1, SCA_2, \ldots SCA_n$ คือ เป็นเซ็ตค่าที่เป็นไปได้ใน Clause ของ $CA_1, CA_2, \ldots CA_n$ ตามลำดับ

SVA คือ เป็นเซ็ตค่าทั้งหมดของ Object A

3.3.1 รูปแบบของเซ็ตค่าที่เป็นไปได้ใน Clause ระหว่าง 2 Clauseที่ทำให้ซ้ำซ้อนกัน

2 Clause ที่มี Object เหมือนกันนั้นจะซ้ำซ้อนกันก็ต่อเมื่อมีเซ็ตค่าที่เป็นไปได้ใน Clause ระหว่าง 2 Clause เป็นเซ็ตที่เท่ากัน ดังตัวอย่างที่ 3.12

ตัวอย่างที่ 3.12

กำหนดให้ $SVA = \{a_1, a_2\}$

ถ้า CA_1 => **A is** a_1 แล้ว $SCA_1 = \{a_1\}$

ถ้า CA_2 => **A is not** a_2 แล้ว $SCA_2 = SVA - \{a_2\} = \{a_1\}$

ถ้า CA_3 => **A is not** a_1 แล้ว $SCA_3 = SVA - \{a_1\} = \{a_2\}$

จะได้ว่า CA_2 ซ้ำซ้อนกับ CA_1 เนื่องจาก $SCA_1 = SCA_2$

แต่ CA_3 ไม่ซ้ำซ้อนกับ CA_1 เนื่องจาก $SCA_1 \neq SCA_3$

3.3.2 รูปแบบของเซ็ตค่าที่เป็นไปได้ใน Clause ระหว่าง 2 Clauseที่ทำให้ขัดแย้งกัน

2 Clause ที่มี Object เหมือนกันนั้นจะขัดแย้งกันก็ต่อเมื่อมีเซ็ตค่าที่เป็นไปได้ใน Clause ระหว่าง 2 Clause เป็นเซ็ตที่ตรงกันข้ามกัน (Inverse Set) ดังตัวอย่างที่ 3.13

ตัวอย่างที่ 3.13

กำหนดให้ $SVA = \{a_1, a_2, a_3\}$

ถ้า CA_1 => **A is** a_1 แล้ว $SCA_1 = \{a_1\}$

ถ้า CA_2 => **A is not** a_1 แล้ว $SCA_2 = SVA - \{a_1\} = \{a_2, a_3\}$

ถ้า CA_3 => **A is** a_3 แล้ว $SCA_3 == \{a_3\}$

จะได้ว่า CA_2 ขัดแย้งกับ CA_1 เนื่องจาก $SCA_1 = SVA - SCA_2$

แต่ CA_3 ไม่ขัดแย้งกับ CA_1 เนื่องจาก $SCA_1 \neq SVA - SCA_3$

3.3.3 รูปแบบของเซ็ตค่าที่เป็นไปได้ใน Clause ระหว่าง 2 Clauseที่ทำให้ Clause หนึ่งซับซ้อนกับอีก Clause หนึ่ง

2 Clause ที่มี Object เหมือนกันนั้นจะของอีก Clause ก็ต่อเมื่อมีเซ็ตค่าที่เป็นไปได้ใน Clause ระหว่าง 2 Clause เป็นซับเซ็ตของอีกเซ็ต ดังตัวอย่างที่ 3.14

ตัวอย่างที่ 3.14

กำหนดให้ $SVA = \{a_1, a_2, a_3\}$

ถ้า CA_1 => **A is** a_1 แล้ว $SCA_1 = \{a_1\}$

ถ้า CA_2 => **A is not** a_1 แล้ว $SCA_2 = SVA - \{a_1\} = \{a_2, a_3\}$

ถ้า CA_3 => **A is** a_3 แล้ว $SCA_3 = \{a_3\}$

จะได้ว่า CA_3 ซับซ้อนกับ CA_2 เนื่องจาก $SCA_1 \subset SCA_3$

แต่ CA_1 ไม่ซับซ้อนกับ CA_2 เนื่องจาก $SCA_1 \not\subset SCA_2$

บทที่ 4

วัตถุกับฐานกฎความรู้ (Object and Rule Base)

4.1 วัตถุและความสัมพันธ์ (Object and Relationship)กับฐานกฎความรู้ (Rule base)

วัตถุ (Object) คือ สิ่งที่สามารถจับต้องได้ สิ่งที่สามารถสัมผัสได้หรือสิ่งที่รู้สึกได้ การนำเอาแนวคิดวัตถุ (Object) มาใช้สร้าง (Define) กฎในฐานกฎความรู้ เพื่อทำให้กฎมีความสอดคล้องกับสิ่งที่สนใจหรือสิ่งที่ต้องการ (Domain Problem หรือ Domain Expert) ในระบบผู้เชี่ยวชาญประโยชน์ที่จะได้รับจากการนำเอาแนวคิดนี้มาใช้ จะทำให้ระบบสามารถเข้าใจได้ง่ายและการบำรุงรักษา (Maintenance) ระบบได้ง่ายในอนาคต

ความสัมพันธ์ (Relationship) คือ ความเกี่ยวข้องหรือความสัมพันธ์ระหว่างวัตถุ 2 ตัวขึ้นไป เช่น ความเป็นแม่ลูกกันหรือความเป็นเจ้าของกัน เป็นต้น ซึ่งในระบบของการตรวจสอบกฎจะเฉพาะเจาะจงความสัมพันธ์ที่เป็นแบบ ความสัมพันธ์แบบเป็น (Is-a) และความสัมพันธ์แบบเป็นชิ้นส่วน (Has-a)

4.2 วัตถุ – แอททริบิวท์ – ค่า (Object – Attributes – Values Triplets (OAV))

วัตถุ – แอททริบิวท์ – ค่า (Object – Attributes - Values Triplets (OAV))คือ รูปแบบหนึ่งที่ใช้ในการแสดงข้อเท็จจริงของเหตุการณ์ (Fact) และกฎในฐานความรู้ ที่แสดงเป็นวัตถุที่ประกอบด้วยเซ็ตของแอททริบิวท์และแอททริบิวท์ประกอบด้วยเซ็ตของค่า รูปที่ 4.1 แสดง OAV KB ที่แสดงโดยระบบฐานกฎความรู้ที่ใช้ OAV Triplets

<Rule-name>: IF <premise> Then <conclusion>

<Premise>: <Object> . <Attribute> <Operator> <Value>| AND <Premise>

<Conclusion>: <Object> . <Attribute> <Operator> <Value>

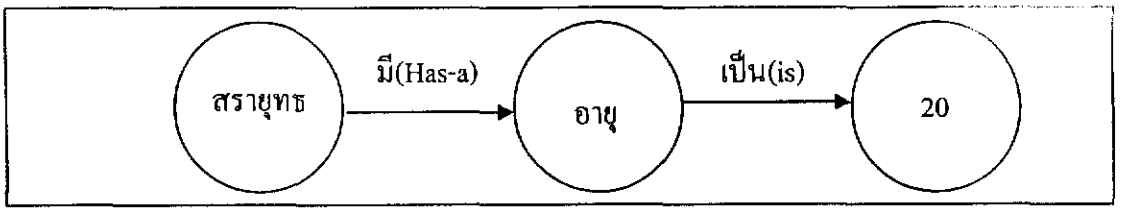
<Object>: string

<Attribute>: string | Boolean | real

<Operator>: < | > | = | ≠

<Value>: real | string | TRUE | ?Attribute | @Attribute

รูปที่ 4.1 แสดงระบบฐานกฎความรู้ที่ใช้ OAV Triplets



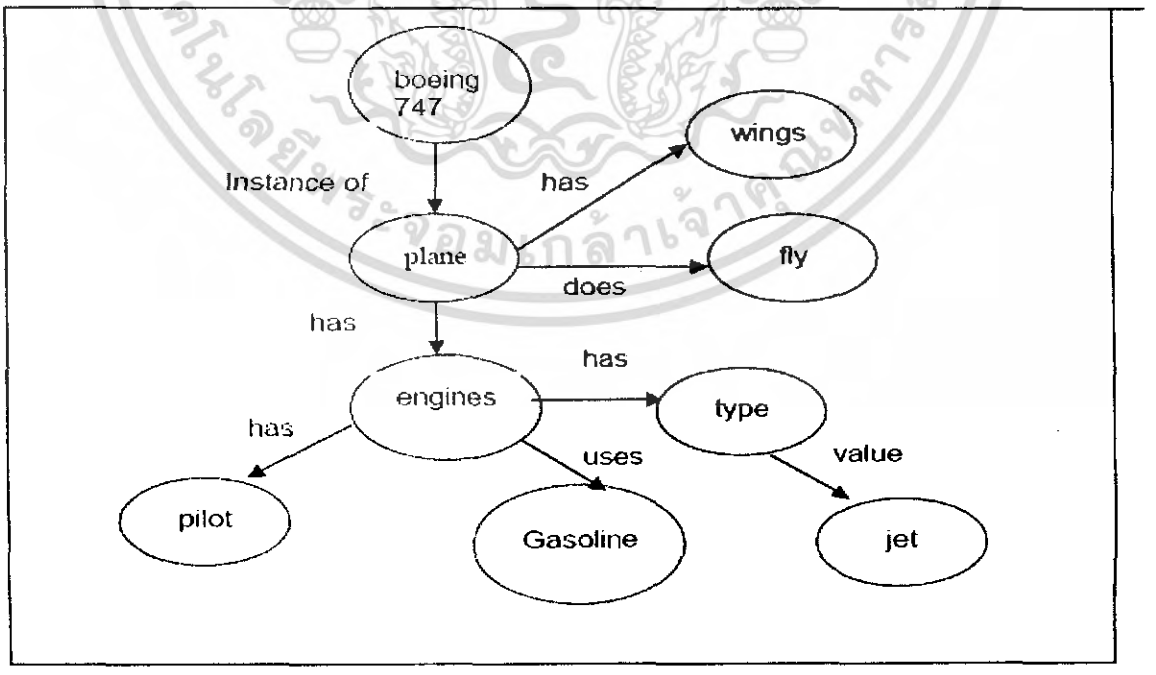
รูปที่ 4.2 แสดงโครงข่าย OAV Triplets (OAV Networks)

จากรูปที่ 4.2 แสดงโครงข่าย OAV triplets โครงข่ายที่แสดงโดยโนด (Node) และเส้นแสดงความสัมพันธ์ระหว่างโนด โดยตัวอย่างแสดง OAV ของวัตถุสรายุทธที่มีแอททริบิวท์อายุมีค่าเท่ากับ 20

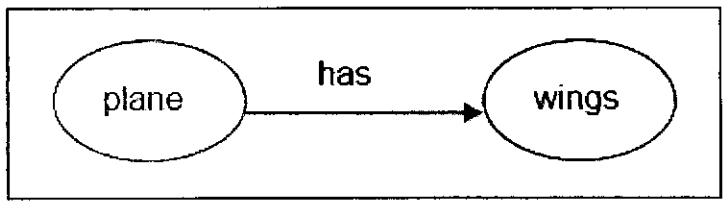
4.3 โครงข่าย semantic (Semantic network)

โครงข่าย Semantic เป็นโครงข่ายที่จำลองมาจากความจำ (Memory) โดยที่แสดงโครงข่ายโดยโนด (Node) และเส้นแสดงความสัมพันธ์ระหว่างโนด (Arc)

โครงข่าย Semanticประกอบด้วยโครงข่าย OAV Triplets หลายๆโครงข่ายรวมกัน โดยความสัมพันธ์ระหว่างโนดที่เป็นความสัมพันธ์แบบ ส่วนประกอบ (has-a) สามารถเปลี่ยนไปสู่ OAV Triplets ดังแสดง รูปที่ 4.3 แสดงโครงข่าย semantic ของเครื่องบิน (plane) ที่มีความสัมพันธ์แบบต่างๆ รูปที่ 4.4 ความสัมพันธ์ระหว่างโนดที่เป็นความสัมพันธ์แบบส่วนประกอบ (has-a) คือโนดเครื่องบิน (plane) กับโนดปีก (wings) และรูปที่ 4.5 แสดงการแปลงความสัมพันธ์แบบส่วนประกอบเป็น OAV Triplets



รูปที่ 4.3 แสดงโครงข่าย semantic ของเครื่องบิน(plane)



รูปที่ 4.4 ความสัมพันธ์ระหว่างโนดที่เป็นความสัมพันธ์แบบส่วนประกอบ(has-a)

Plane. wings = ?wing

รูปที่ 4.5 แสดงการแปลงความสัมพันธ์แบบส่วนประกอบเป็น OAV Triplets

4.4 Production rule ในรูปแบบ OAV

จากรูปแบบกฎเดิมที่ไม่ได้ใช้การจำลองแบบวัตถุให้แปลงให้อยู่ในรูปแบบ วัตถุซึ่งวัตถุที่ประกอบด้วยแอททริบิวท์ต่างๆ และแอททริบิวท์สามารถประกาศเซ็ตของค่าได้ดังแสดงตัวอย่างที่ 4.1

ตัวอย่างที่ 4.1

Rule 1: IF the student's GRE score is 1350 or more
THEN admit the student to the graduate program.
Rule 2: IF grade point average equals or exceeds 3.5
THEN accept into honor society.

จากกฎในฐานกฎความรู้ข้างบนสามารถแปลงให้อยู่ในกฎรูปแบบ OAV ได้ดังกฎข้างล่าง โดยการมอง นักเรียน (student) เป็นวัตถุที่มีเซ็ตแอททริบิวท์ประกอบสมาชิกเป็นแอททริบิวท์ GREscore GPA Admission statusและaccept ดังแสดงตัวอย่างที่ 4.2

ตัวอย่างที่ 4.2

Rule 1: IF Student . GREscore >= 1350
THEN Student . admissionstatus = yes
Rule 2: IF Student . GPA >= 3.5
THEN Student . accept = into honor society

4.5 การกำหนดเซ็ตของการตรวจสอบฐานกฎความรู้แบบOAV

(Definitions of rule OAV Triplets Validation)

กำหนดให้

RB = ฐานกฎความรู้(Rule Base)

ask(voc) = แอททริบิวท์ที่สามารถถามได้(the askable attributes, query premise)

{P(k)} = เซ็ตของอนุประโยคเงื่อนไข (Premise clauses) ของ rule k

p(k) = เป็นหนึ่งอนุประโยคเงื่อนไข (Premise clauses)ในเซ็ตของ {P(k)}

c(k) = เป็นข้อสรุปของกฎที่มีหนึ่งอนุประโยคข้อสรุป (Conclusion clauses)

{S} = {T} = เซ็ต S เท่ากับ เซ็ต T

{G} = เซ็ตของเป้าประสงค์(Goals of system)

{Val (O / A)}= เซ็ตของทุกค่า "V" ของทุกแอททริบิวต์ "A" ของทุกวัตถุ "O" ที่อยู่ในฐานกฎความรู้ทั้งในส่วนอนุประโยคเงื่อนไข (Premise clauses) และอนุประโยคข้อสรุป(Conclusion clauses)

OA-LST = เซ็ตของทุกค่าของทุกแอททริบิวต์ "A" ของทุกวัตถุ "O" ที่ได้กำหนดขึ้นมาตอนตั้งตน

O.A = V = ค่า "V" ของแอททริบิวต์ "A" ของวัตถุ "O"

4.5.1 การกำหนดเซ็ตของความกลมกลืน (Definitions of Consistency)

4.5.1.1. กฎที่มีการซ้ำซ้อนกัน (Redundant rules)

กฎ r จะซ้ำซ้อนถ้า

$$\{\exists\ s \mid s \in RB,\ \{P(r) = P(s)\},\ c(r) = c(s)\}$$

4.5.1.2. กฎที่มีการขัดแย้งกัน (Conflicting rules)

กฎ r ขัดแย้งกับกฎ s ถ้า

$$\{\{P(r) = \{P(s)\},\ (O.A = V1) \in c(r),\ (O.A = V2) \in c(s),\ V1 = \neg V2\}$$

4.5.1.3. กฎที่สามารถรวมเป็นกฎเดียวกัน (Subsumed rules)

กฎ r สามารถรวมในกฎ s ถ้า

$$\{c(r) = c(s),\ \{P(s) \subset P(r)\}\}$$

4.5.1.4. กฎที่มีอนุประโยคเงื่อนไขที่ไม่จำเป็น (Unnecessary premise clause)

กฎ r และกฎ s มีเงื่อนไขที่ไม่จำเป็น ถ้า

$$\{\exists s \mid s \in RB,\ c(r) = c(s),\ \{\exists\ p_1, p_2 \mid p_1 \in \{P(r)\},\ p_1 = \{O_1.A_1 = V_1), p_2 \in \{P(s)\},$$
$$p_2 = (O_1.A_1 = V_2),\ V_1 = \neg V_2,\ p_1 \neq p_2\},\ \{\forall p_i \mid p_i \neq p_1,\ p_i \in \{P(r)\}, p_i \in \{P(s)\}\}$$

4.5.1.5. กฎเป็นโซ่วงกลม (Circular rule chains)

กฎ $r_1, r_{2,...} r_n$ เป็นโซ่วงกลม ถ้า

$\{c(r_n) \in \{G\}, c(r_{n-1}) \in p(r_n), c(r_{n-2}) \in p(r_{n-1}), \ldots, c(r_1) \in p(r_2), c(r_n) \in p(r_1)\}$

4.5.2 การกำหนดเซ็ตของความสมบูรณ์ (Definitions of Completeness)

4.5.2.1. ค่าแอททริบิวท์ที่ค่าไม่ได้อ้าง (Unreferenced attribute values)

ค่า "V" เป็นค่าแอททริบิวท์ที่ค่าไม่ได้อ้าง ถ้า

$\{ \forall V | V \in OA\text{-}LST, V \notin \{VAL(O/A)\}\}$

4.5.2.2. ค่าแอททริบิวท์ที่ค่าผิด (Illegal attribute values)

ค่า "V" เป็นค่าแอททริบิวท์ที่ค่าผิด ถ้า

$\{ \exists V | V \in \{VAL(O/A)\}, V \notin OA\text{-}LST\}$

4.5.2.3. ข้อสรุปที่เป็นตัวกลางที่ไม่สามารถสำเร็จได้ (Unachievable intermediate conclusions)

อนุประโยคข้อสรุปของกฎ R c(R)เป็นข้อสรุปที่เป็นตัวกลางที่ไม่สามารถสำเร็จได้ ถ้า

$\{c(R) \neq \{G\}, \{\forall S | S \in RB, S \neq R, c(R) \notin \{P(S)\}\}\}$

4.5.2.4. ข้อสรุปที่เป็นสุดท้ายที่ไม่สามารถสำเร็จได้ (Unachievable (final) conclusions ,or goal)

อนุประโยคข้อสรุปของกฎ R c(R)เป็นข้อสรุปที่เป็นสุดท้ายที่ไม่สามารถสำเร็จได้ ถ้า

$\{c(R) = \{G\}, \{\forall S, p(R) | S \in RB, S \neq R, p(R) \in \{P(R)\}, p(R) \notin \{C(S)\}, p(R) \notin ask(voc)\}\}$

4.5.2.5. เงื่อนไขที่ไม่สามารถสำเร็จได้ (Unachievable premises)

อนุประโยคเงื่อนไขของกฎ R p(R)เป็นเงื่อนไขที่ไม่สามารถสำเร็จได้ ถ้า

$\{p(R), \{\forall S, | S \in RB, S \neq R, p(R) \notin \{C(S)\}, p(R) \notin ask(voc)\}\}$

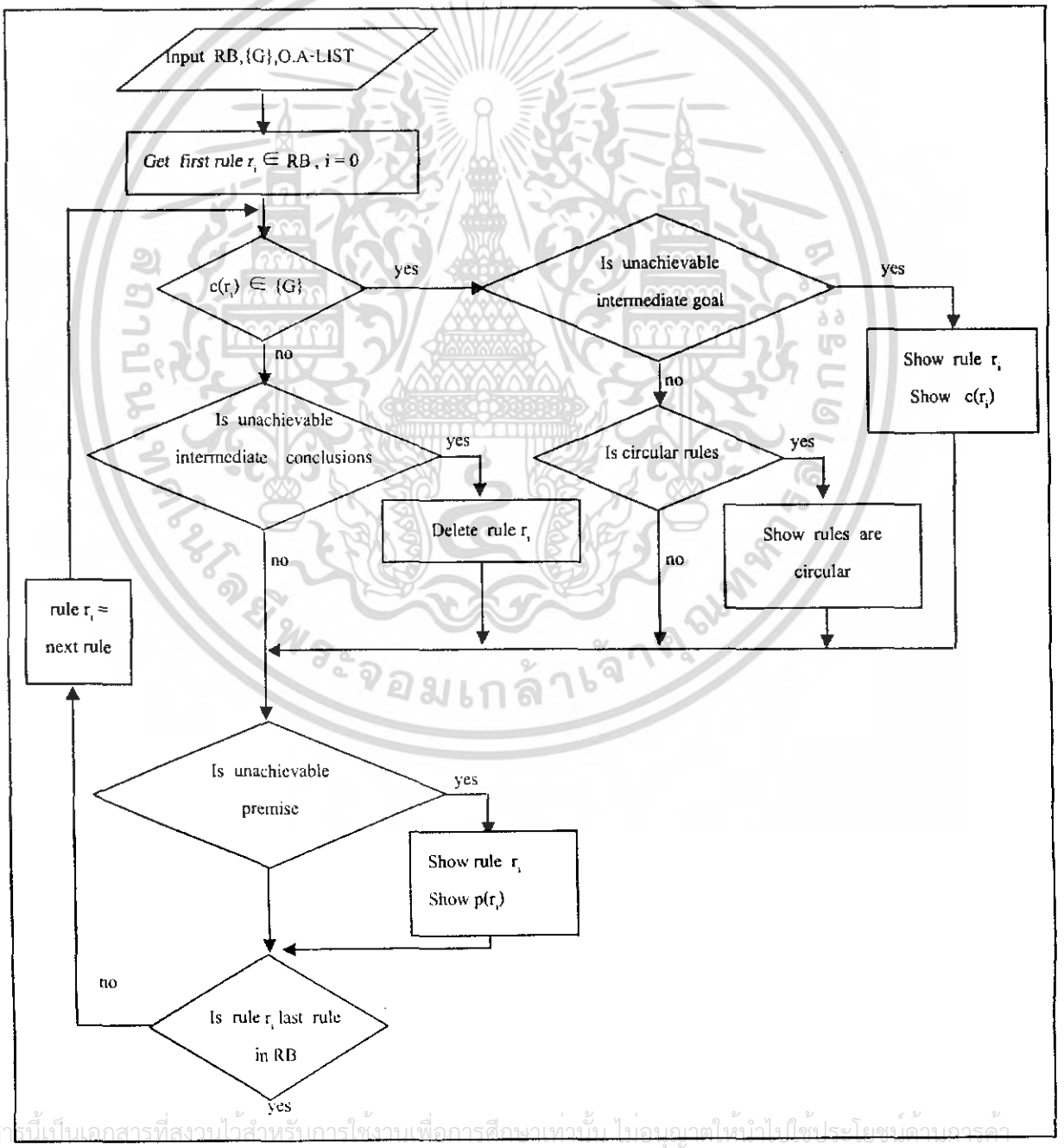
บทที่ 5

การออกแบบและการพัฒนาระบบตรวจสอบกฎในฐานความรู้

5.1 ขั้นตอนการตรวจสอบ

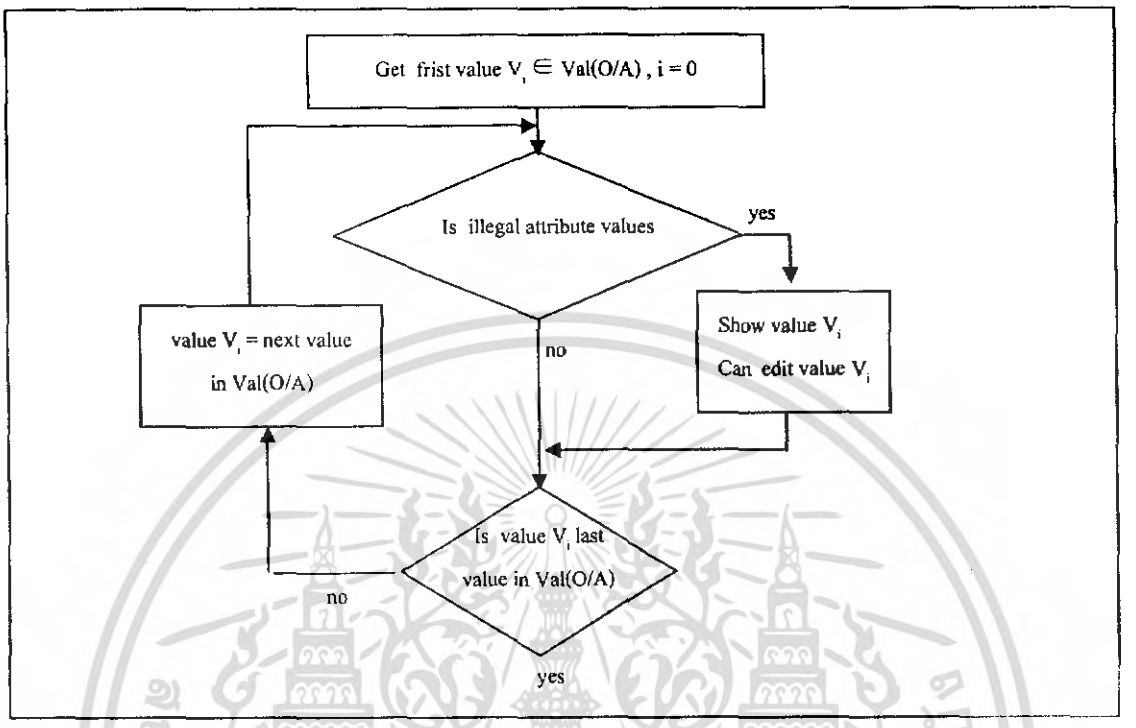
ขั้นตอนการตรวจสอบแบ่งออกเป็น 4 ขั้นตอนหลัก

ขั้นตอนที่ 1 ตรวจสอบกฎโดยใช้กฎเดียวกับกฎในฐานกฎความรู้ทั้งหมดเพื่อตรวจสอบข้อสรุปที่เป็นตัวกลางที่ไม่สามารถสำเร็จได้ ข้อสรุปที่เป็นสุดท้ายที่ไม่สามารถสำเร็จได้ เงื่อนไขที่ไม่สามารถสำเร็จได้หรือกฎเป็นโซ่วงกลม ดังรูปที่ 5.1 แสดง Flowchart ขั้นตอนที่ 1



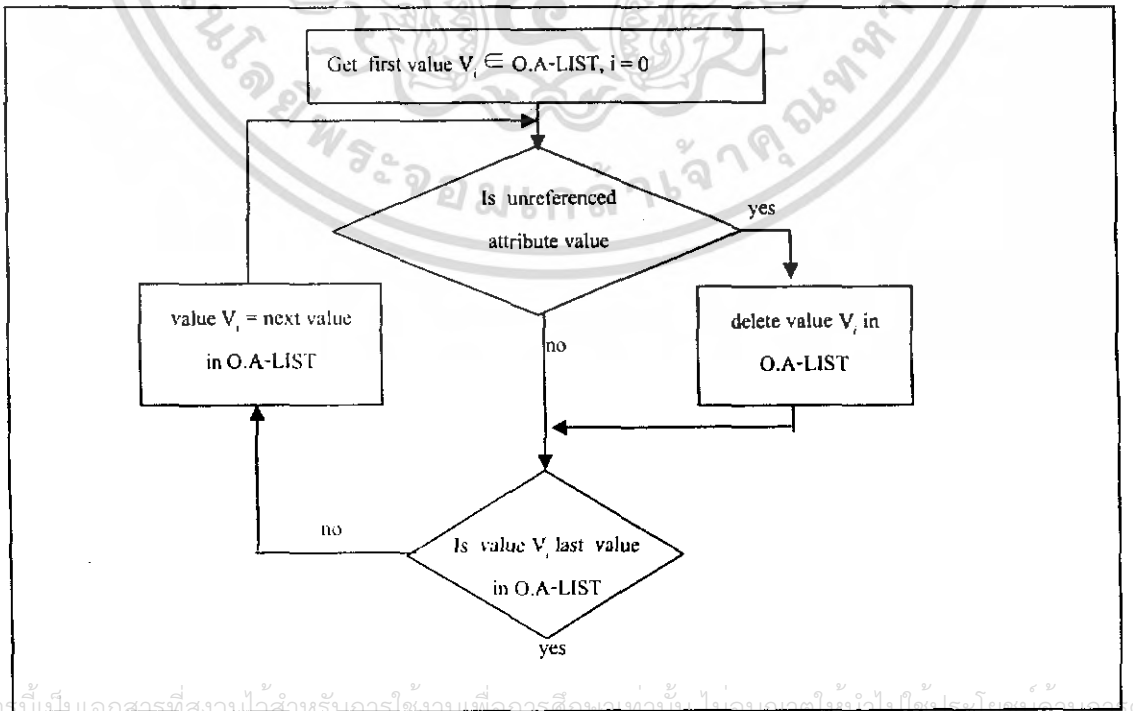
รูปที่ 5.1 แสดง Flowchart ขั้นตอนที่ 1

ขั้นตอนที่ 2 ตรวจสอบค่าโดยใช้ค่าที่อยู่ในฐานกฎความรู้ค่าเดียวกับค่าที่ได้กำหนดค่าขึ้นมาตอนตั้งต้นทั้งหมดเพื่อตรวจสอบค่าแอททริบิวท์ที่ค่าผิดดังรูปที่ 5.2 แสดง Flowchart ขั้นตอนที่ 2



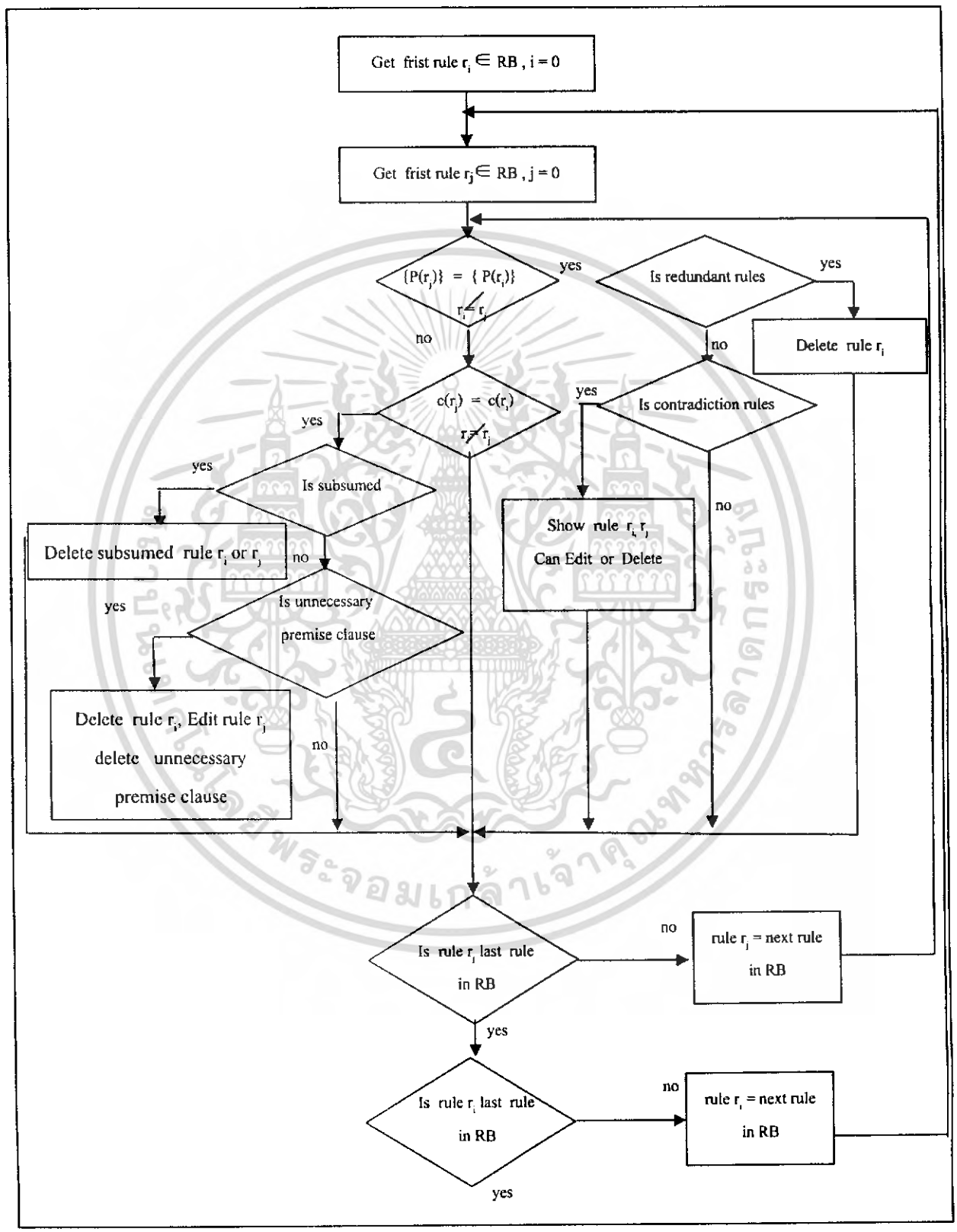
รูปที่ 5.2 แสดง Flowchart ขั้นตอนที่ 2

ขั้นตอนที่ 3 ตรวจสอบค่าโดยใช้ค่าที่ได้กำหนดค่าขึ้นมาตอนตั้งต้นค่าเดียวกับค่าที่อยู่ในฐานกฎความรู้ทั้งหมดเพื่อตรวจสอบค่าแอททริบิวท์ที่ค่าไม่ได้อ้าง ดังรูปที่ 5.3 แสดง Flowchart ขั้นตอนที่ 3



รูปที่ 5.3 แสดง Flowchart ขั้นตอนที่ 3

ขั้นตอนที่ 4 ตรวจสอบกฎโดยใช้สองกฎที่อยู่ฐานกฎความรู้เปรียบเทียบกันเอง เพื่อตรวจสอบความซ้ำซ้อนของกฎ ความขัดแย้งของกฎ รวมในกฎเดียวกันและเงื่อนไขที่ไม่จำเป็น ดังรูปที่ 5.4 แสดง Flowchart ขั้นตอนที่ 4



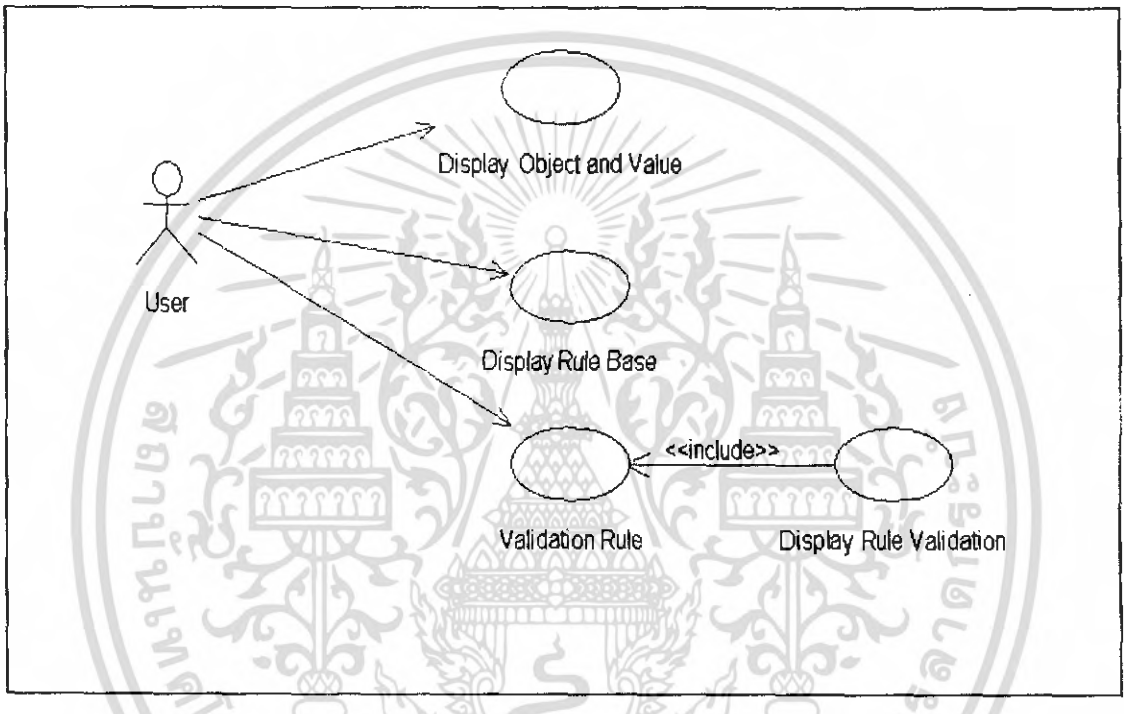
ดังรูปที่ 5.4 แสดง Flowchart ขั้นตอนที่ 4

5.2 การออกแบบระบบตรวจสอบกฎในฐานความรู้

จากที่ได้เสนอรูปแบบการแสดงความรู้ Production Rule ของระบบฐานกฎความรู้ การกำหนดเซ็ตของการตรวจสอบฐานกฎความรู้แบบ OAV รูปแบบและขั้นตอนการตรวจสอบ ซึ่งระบบสามารถใช้แนวคิด Object – Oriented มาออกแบบระบบได้โดยใช้ UML และใช้ Database ในการเก็บข้อมูลของ Rule Base

5.2.1 Use Case Diagram

5.2.1.1 Use Case Diagram ระบบโดยรวม



รูปที่ 5.5 Use Case Diagram ระบบโดยรวม

จากรูปที่ 5.5 แสดง Use Case Diagram ของระบบโดยรวมที่ประกอบด้วย 4 Use case หลักคือ

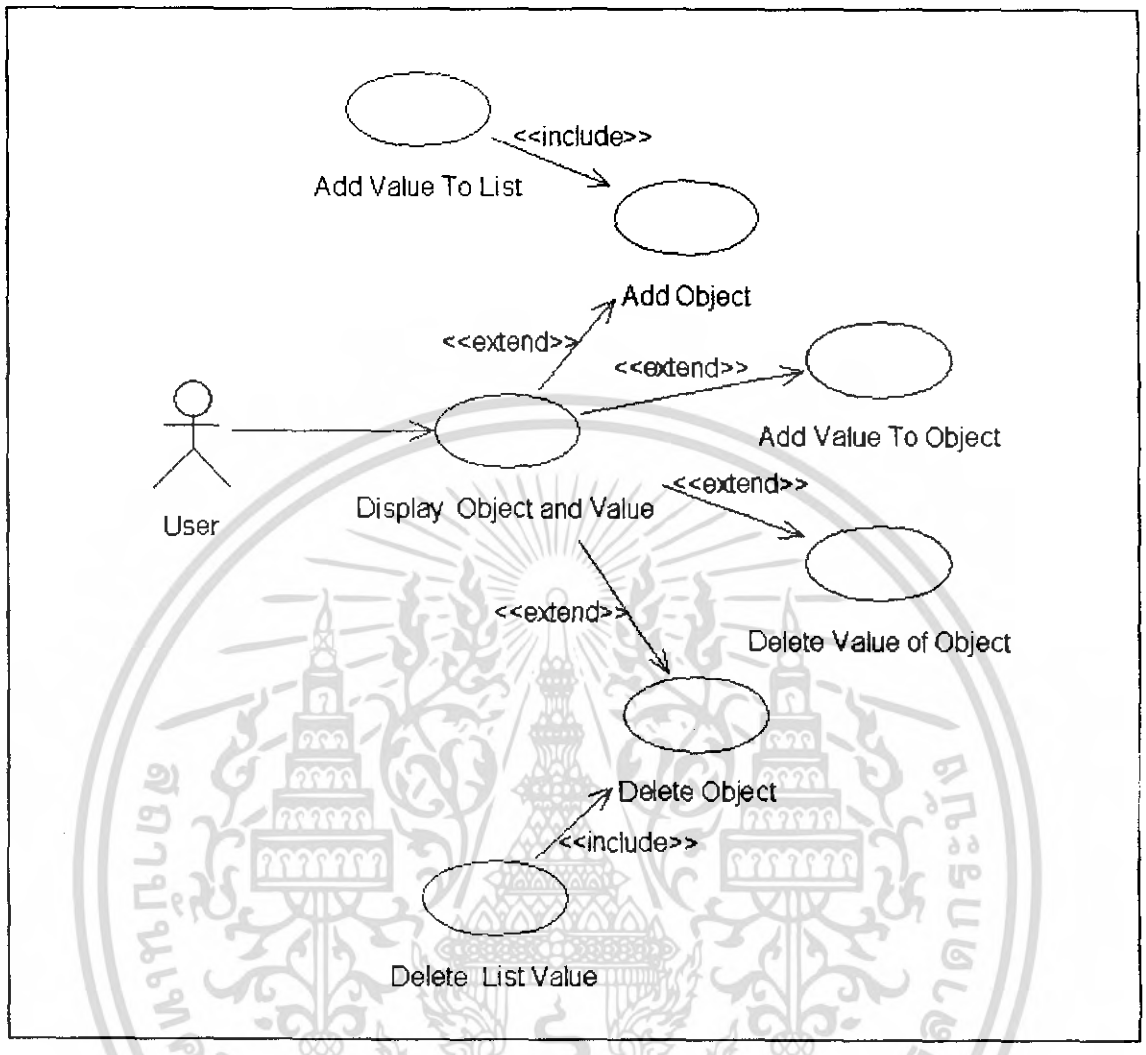
Display Object and Value เป็น Use Case ที่ Actor User ใช้ดู Objects และ List Value ใน Object นั้น เป็น Objects ที่ตรงกับปัญหาที่จะใช้ในการสร้างกฎในฐานกฎความรู้

Display Rule Base เป็น Use Case ที่ Actor User ใช้ดู Rule ทั้งหมดใน Rule Base

Validation Rule เป็น Use Case ที่ Actor User ใช้ดูตรวจสอบ Rule ทั้งหมดใน Rule Base และ เมื่อตรวจสอบจะแสดง Use Case Display Rule Validation พร้อมทุกครั้ง

Display Rule Validation เป็น Use Case ที่ Actor User ใช้ดู Rule ที่พบความไม่กลมกลืนหรือความไม่สมบูรณ์ของส่วนประกอบ

5.2.1.2 Use Case Diagram ในส่วน Display Object and Value



รูปที่ 5.6 Use Case Diagram ในส่วน Display Object and Value

จากรูปที่ 5.6 แสดง Use Case Diagram ในส่วน Display Object and Value ประกอบด้วย 6 Use Case ที่เพิ่มเข้ามา ดังต่อไปนี้

Add Object เป็น Use Case ในบางครั้งที่ต้องการเพิ่ม Object ที่ต้องการให้ระบบ

Delete Object เป็น Use Case ในบางครั้งที่ต้องการลบ Object ที่ไม่จำเป็นกับระบบ

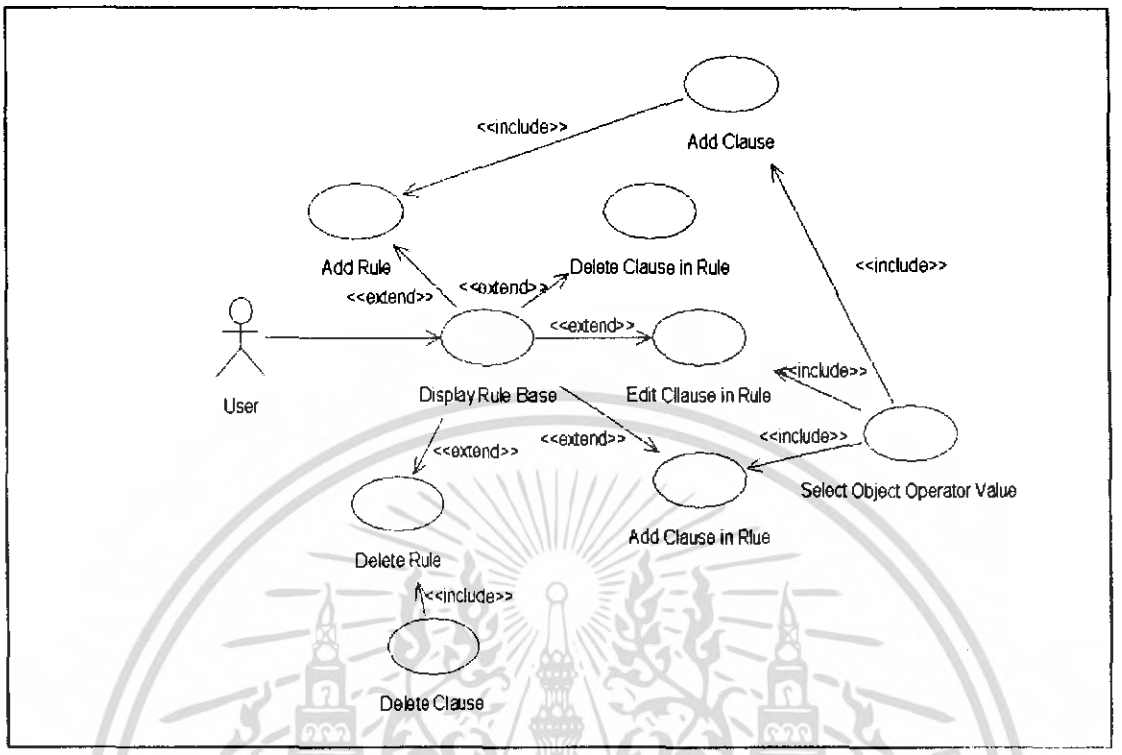
Add Value To Object เป็น Use Case ในบางครั้งที่ต้องการเพิ่ม Value ที่เป็นไปได้ ใน List Value ของ Objcct ที่ต้องการให้ระบบ

Delete Value of Object เป็น Use Case ในบางครั้งที่ต้องการลบ Value ที่ไม่ต้องการ ใน List Value ของ Object โดย Value นั้นไม่จำเป็นกับระบบ

Add Value To List เป็น Use Case ที่ทุกครั้งที่ Add Object จะต้องสร้าง List Value

Delete List Value เป็น Use Case ที่ทุกครั้งที่ Delete Object จะต้องลบ List Value

5.2.1.3 Use Case Diagram ในส่วน Display Rule Base



รูปที่ 5.7 Use Case Diagram ในส่วน Display Rule Base

จากรูปที่ 5.7 แสดง Use Case Diagram ในส่วน Display Rule Base ประกอบด้วย 8 Use - Case ที่เพิ่มเข้ามา ดังต่อไปนี้

Add Rule เป็น Use Case ในบางครั้งที่ต้องการเพิ่ม Rule ที่ต้องการให้ระบบ

Delete Rule เป็น Use Case ในบางครั้งที่ต้องการลบ Rule ที่ไม่จำเป็นกับระบบ

Add Clause in Rule เป็น Use Case ในบางครั้งที่ต้องการเพิ่ม Clause ใน Rule ที่ต้องการให้ระบบ

Delete Clause in Rule เป็น Use Case ในบางครั้งที่ต้องการลบ Clause ใน Rule ที่ที่ไม่จำเป็นกับระบบ

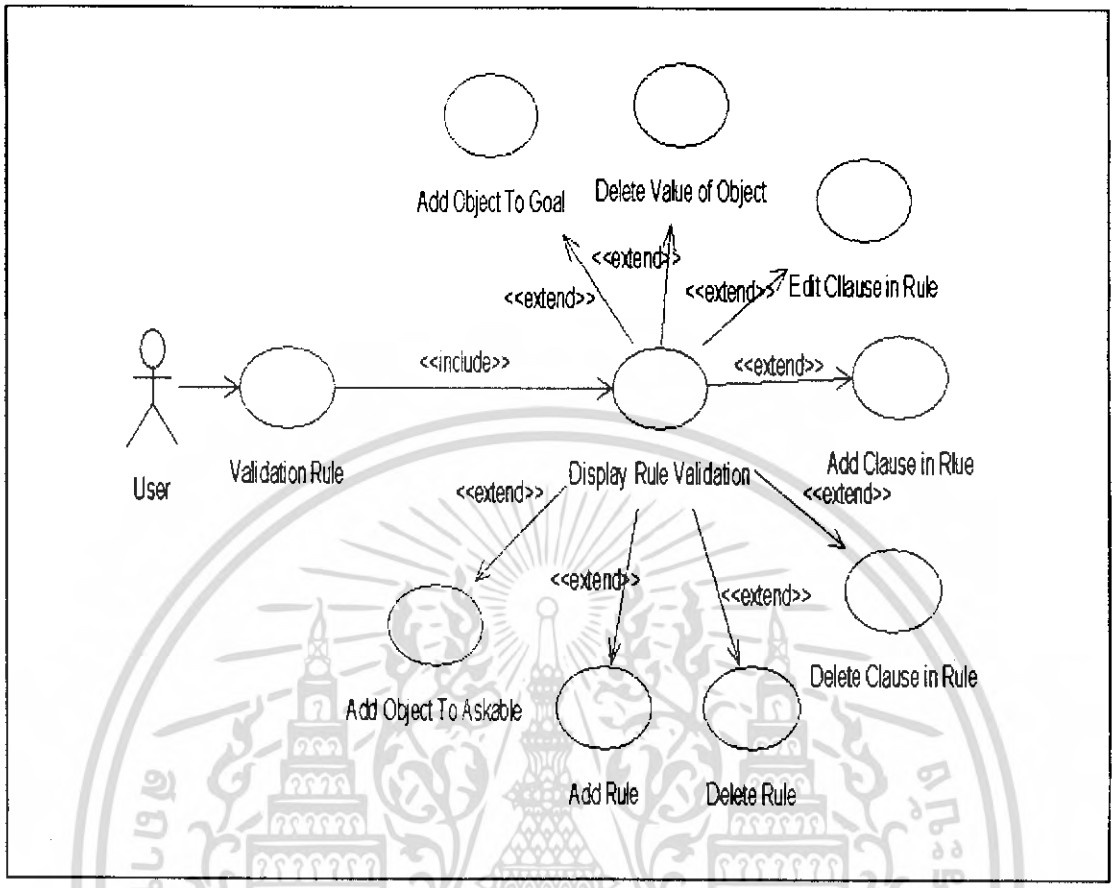
Edit Clause in Rule เป็น Use Case ในบางครั้งที่ต้องการแก้ไข Clause ใน Rule ที่ต้องการปรับเปลี่ยนในระบบ

Add Clause เป็น Use Case ที่ทุกครั้ง Add Rule จะต้องมีการสร้าง Clause ให้กับ Rule

Delete Clause เป็น Use Case ที่ทุกครั้ง Delete Rule จะต้องมีการลบ Clause ที่อยู่ใน Rule นั้น ให้กับระบบด้วย

Select Object Operator Value เป็น Use Case ที่ทุกครั้ง Add Clause, Edit Clause in Rule หรือ Add Clause in Rule จะให้เลือก Object ที่จะใช้, Operator ที่จะใช้และ Value ที่จะใช้

5.2.1.4 Use Case Diagram ในส่วน Validation Rule



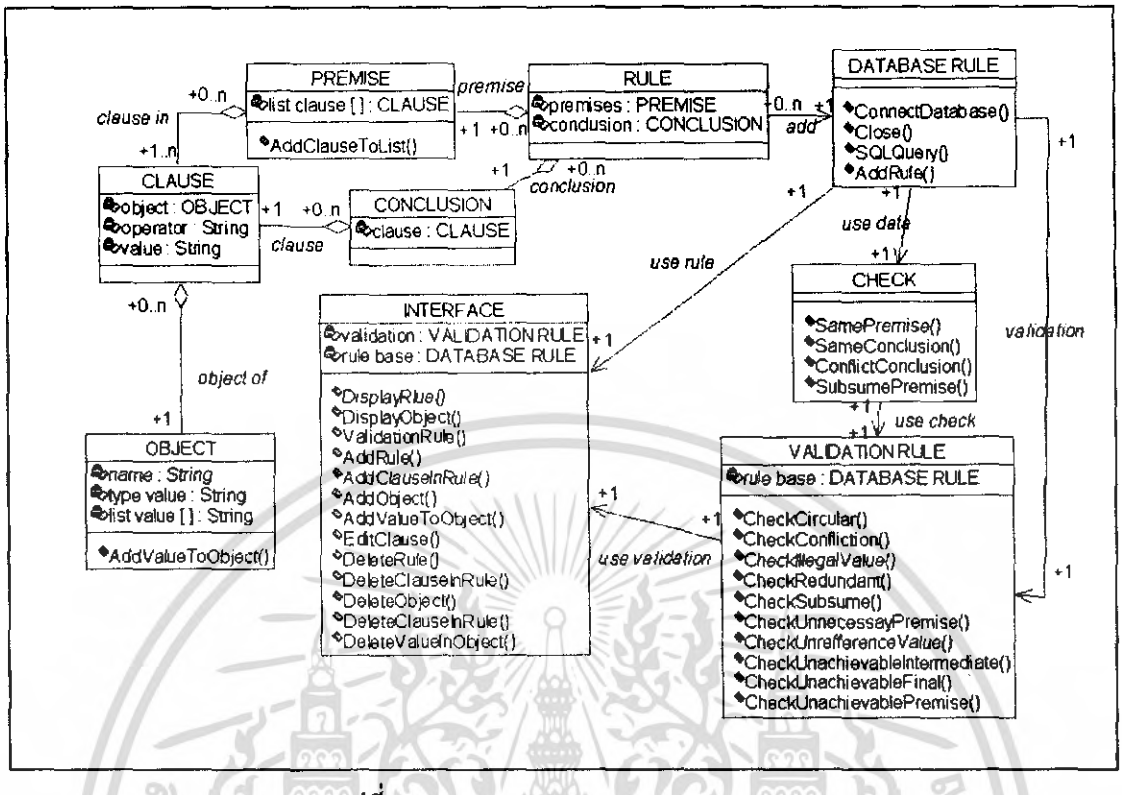
รูปที่ 5.8 Use Case Diagram ในส่วน Validation Rule

จากรูปที่ 5.8 แสดง Use Case Diagram ในส่วน Validation Rule ประกอบด้วย 2 Use - Case ที่เพิ่มเข้ามานอกเหนือจากของเดิมที่เมื่อ Display Rule Validation คือ Delete Value of Object(ใช้ในกรณีที่ Unreferenced Value), Edit Clause in Rule, Add Clause in Rule, Delete Clause in Rule, Add Rule และDelete Rule ที่ใช้ในการปรับแก้ Rule ที่ได้จากการตรวจสอบกฎให้สมบูรณ์ ดังต่อไปนี้

Add Object To Goal เป็น Use Case ที่เมื่อทำการตรวจสอบแล้วได้กฎที่ไม่สมบูรณ์และสามารถแก้ไขปัญหาโดยการให้ Conclusion นั้นเป็น Goal จะพบในปัญหา Unachievable Intermediate Conclusions เมื่อได้ Rule จากการตรวจสอบแล้วพบว่าต้องการให้ Conclusion นั้นเป็น Goal แล้วสามารถแก้ปัญหาได้

Add Object To Askable(Query Premise) เป็น Use Case ที่เมื่อทำการตรวจสอบแล้วได้กฎที่ไม่สมบูรณ์และสามารถแก้ไขปัญหาโดยการให้ Premise นั้นสามารถถามได้ จะพบในปัญหา Unachievable Final(Goal) Conclusions และ Unachievable Premise เมื่อได้ Rule จากการตรวจสอบแล้วพบว่าต้องการให้ Premise นั้นเป็น Query Premise แล้วสามารถแก้ปัญหาได้

5.2.2 Class Diagram



รูปที่ 5.9 แสดง Class Diagram ระบบตรวจสอบ

จากรูปที่ 5.9 แสดง Class Diagram ระบบตรวจสอบที่ประกอบด้วย

Class OBJECT มีความสัมพันธ์เป็นส่วนประกอบหรือเป็นชิ้นส่วนของ Class CLAUSE ประกอบด้วยเซ็ตของ Value ที่สามารถเป็นไปได้

Class CLAUSE ประกอบด้วย Object, Operator และ Value สร้าง Clause และมีความสัมพันธ์เป็นส่วนประกอบหรือเป็นชิ้นส่วนของ Class PREMISE กับ Class CONCLUSION

Class PREMISE ประกอบด้วยเซ็ตของ Clause และมีความสัมพันธ์เป็นส่วนประกอบ Class RULE

Class CONCLUSION ประกอบด้วย Clause และมีความสัมพันธ์เป็นส่วนประกอบ Class RULE

Class RULE ประกอบด้วย PremiseและConclusion และมีความสัมพันธ์เป็น Rule ใน Class DATABASE RULE

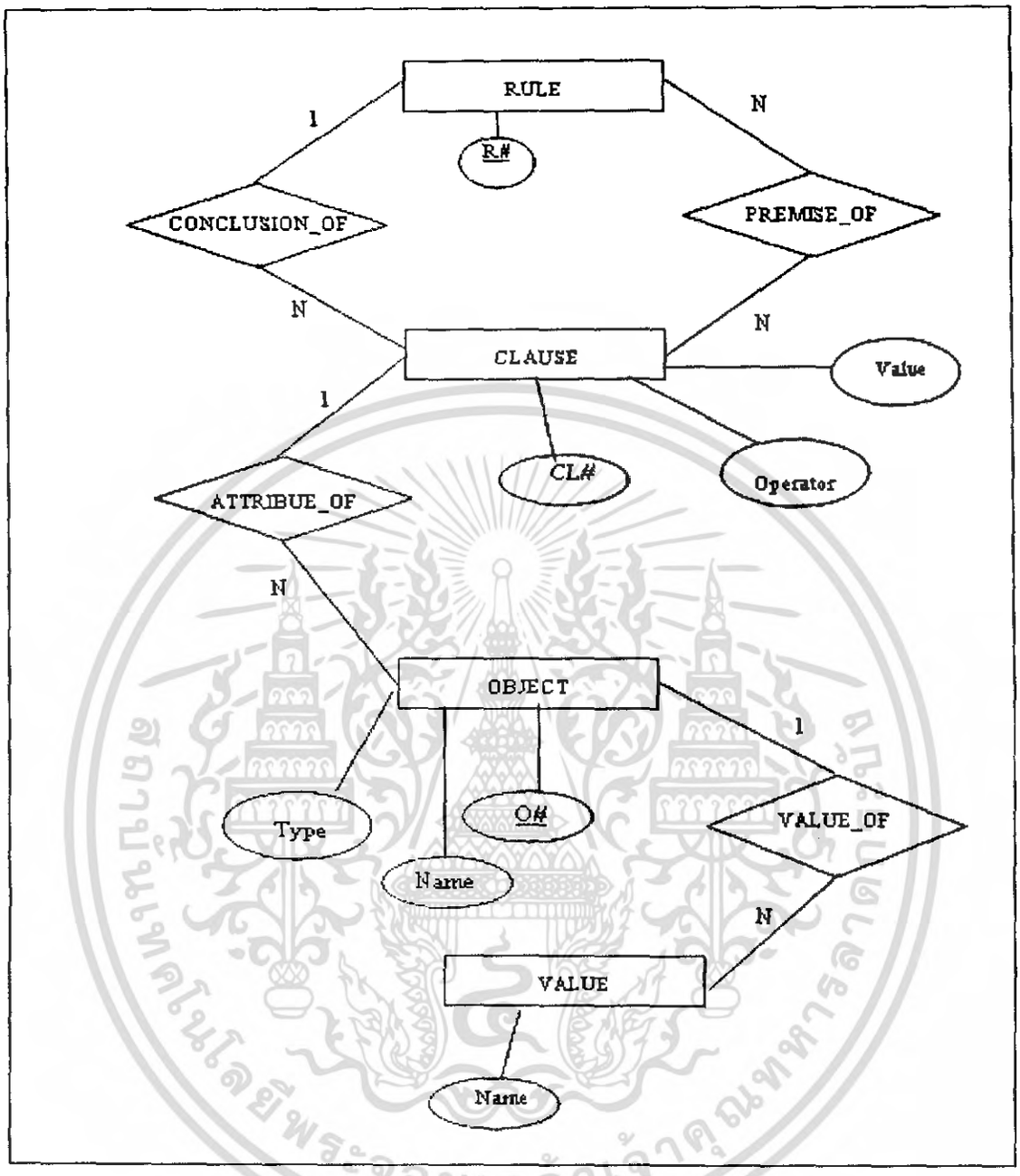
Class DATABASE RULE ประกอบด้วยเซ็ตของ rule

Class CHECK มีความสัมพันธ์เป็นใช้ตรวจสอบพื้นฐานให้กับ Class VALIDATION

Class VALIDATION มีความสัมพันธ์เป็นใช้ตรวจสอบกับ Class DATABASE RULE

Class INTERFACE มีความสัมพันธ์ใช้แสดง Class DATABASE RULE และ Class VALIDATION

5.2.3 ER Diagram



รูปที่ 5.10 ER Diagram

จากรูปที่ 5.10 แสดง ER – Diagram การเก็บข้อมูล Rule ลง Database เพื่อให้ง่ายต่อการจัดการกับข้อมูล ประกอบด้วย **4 Entity Type** คือ

RULE Entity Type ประกอบด้วย Attribute Rule ID ซึ่งเป็น Primary Key

CLAUSE Entity Type ประกอบด้วย Attribute Clause ID ซึ่งเป็น Primary Key, Operator ที่แสดงถึง Clause มีการใช้ Operator อะไรและ Value ที่แสดงถึง Clause ค่าของ Value เท่าไร

OBJECT Entity Type ประกอบด้วย Attributes Object ID ซึ่งเป็น Primary Key, Name ที่แสดงถึงชื่อของ Object นั้นและ Type ที่แสดงถึงชนิดของ Object นั้น

VALUE Entity Type ประกอบด้วย Attributes Value ID ซึ่งเป็น Primary Key, Name ที่แสดงถึงชื่อของ Value นั้น

จากรูปที่ 5.10 แสดง ER – Diagram ประกอบด้วย **4 Relationship** คือ

CONCLUSION_OF แสดงความสัมพันธ์ว่า Rule ประกอบด้วย Conclusion Clause อะไร โดยที่ 1 Rule จะมีได้ 1 Conclusion Clause และ 1 Clause สามารถเป็น Conclusion Clause ได้มากกว่า 1 Rule จึงมีความสัมพันธ์เป็นแบบ 1:M

PREMISE_OF แสดงความสัมพันธ์ว่า Rule ประกอบด้วย Premise Clause อะไร โดยที่ 1 Rule ได้มากกว่า 1 Premise Clause และ 1 Clause สามารถเป็น Premise Clause ได้มากกว่า 1 Rule จึงมีความสัมพันธ์เป็นแบบ M:M

ATTRIBUTE_OF แสดงความสัมพันธ์ว่า Clause ประกอบด้วยแอททริบิวท์ที่เป็น Object อะไร โดยที่ 1 Clause จะมีได้ 1 Object เป็นแอททริบิวท์และ 1 Object สามารถเป็นแอททริบิวท์ได้มากกว่า 1 Clause จึงมีความสัมพันธ์เป็นแบบ 1:M

VALUE_OF แสดงความสัมพันธ์ว่า Object ประกอบด้วย List Value อะไรจึงมีความสัมพันธ์เป็นแบบ 1:M

จาก ER – Diagram รูปที่ 5.10 จะได้ 5 Relations ดังรูปที่ 5.11

RULE

p.k. f.k.

RULE_ID#	CLAUSE_CONCLUSION#

OBJECT

p.k.

OBJECT_ID#	OBJECT_NAME	TYPE

RULE_PREMISES_CLAUSE

p.k. p.k.

RLUE_ID#	CLAUSE_PREMISE#

VALUE

p.k. f.k.

VALUE_ID#	VALUE_NAME	OBJECT_ID#

CLAUSE

p.k. f.k.

CLAUSE_ID#	OBJECT_ID#	OPERATOR	VALUE

รูปที่ 5.11 Relations ของ ER Diagram ในรูปที่ 5.10

การเก็บกฎตัวอย่างที่ 5.1 ลงใน Database ได้ดังรูปที่ 5.12

ตัวอย่างที่ 5.1

Traffic Light = {red, green} Car = {stop, run , run slow } Time Count = {low, high}

IF Traffic Light is red THEN Car is stop

IF Traffic Light is green AND Time Count is high THEN Car is run

IF Traffic Light is green AND Time Count is low THEN Car is run slow

OBJECT

OBJECT_ID	OBJECT_NAME	TYPE
O00001	Traffic Light	Set Value
O00002	Car	Set Value
O00003	Traffic Time Count	Set Value

VALUE

VALUE_ID	VALUE_NAME	OBJECT_ID
V00001	green	O00001
V00002	red	O00001
V00003	run	O00002
V00004	run slow	O00002
V00005	stop	O00002
V00006	high	O00003
V00007	low	O00003

CLAUSE

CLAUSE_ID	OBJECT_ID	OPERATOR	VALUE
CL00001	O00002	is	stop
CL00002	O00001	is	red
CL00003	O00002	is	run
CL00004	O00001	is	green
CL00005	O00002	is	run slow
CL00006	O00003	is	high
CL00007	O00003	is	low

RULE

RULE_ID	CLAUSE_CONCLUSION
R00001	CL00001
R00002	CL00003
R00003	CL00005

RULE_PREMISES_CLAUSE

RULE_ID	CLAUSE_PREMISE
R00001	CL00002
R00002	CL00004
R00002	CL00006
R00003	CL00004
R00003	CL00007

รูปที่ 5.12 การเก็บกฎตัวอย่างที่ 5.1 ลงใน Database

บทที่ 6

การทดลอง

การทดลองเป็นการทดสอบรูปแบบของการตรวจสอบต่าง ๆ ที่ได้เสนอมาในบทที่ 3 และบทที่ 4 โดยใช้โปรแกรมแอพพลิเคชั่นที่ได้พัฒนาขึ้นมาจากการออกแบบในบทที่ 5 โดยใช้ภาษา Python ในการพัฒนา Module wxPython ในการพัฒนาส่วนติดต่อกับผู้ใช้ (User Interface) และใช้ MySQL Server เป็น Database โดยมี Module MySQLdb ติดต่อระหว่าง MySQL กับ Python

6.1 การสร้าง Object ในการทดสอบ

ระบบตรวจสอบกฎในฐานความรู้มีที่การแสดงความรู้เป็นกฎ (Production rule) ในรูปแบบ OV (Object - Value) ดังนั้นก่อนที่จะมีการสร้างกฎจะต้องสร้าง Object และ List Value ที่ Object สามารถเป็นไปได้ แล้วนำ Object ที่สร้างขึ้นมาสร้างเป็นกฎต่างๆ อย่างเช่นตัวอย่างที่ 6.1

ตัวอย่างที่ 6.1

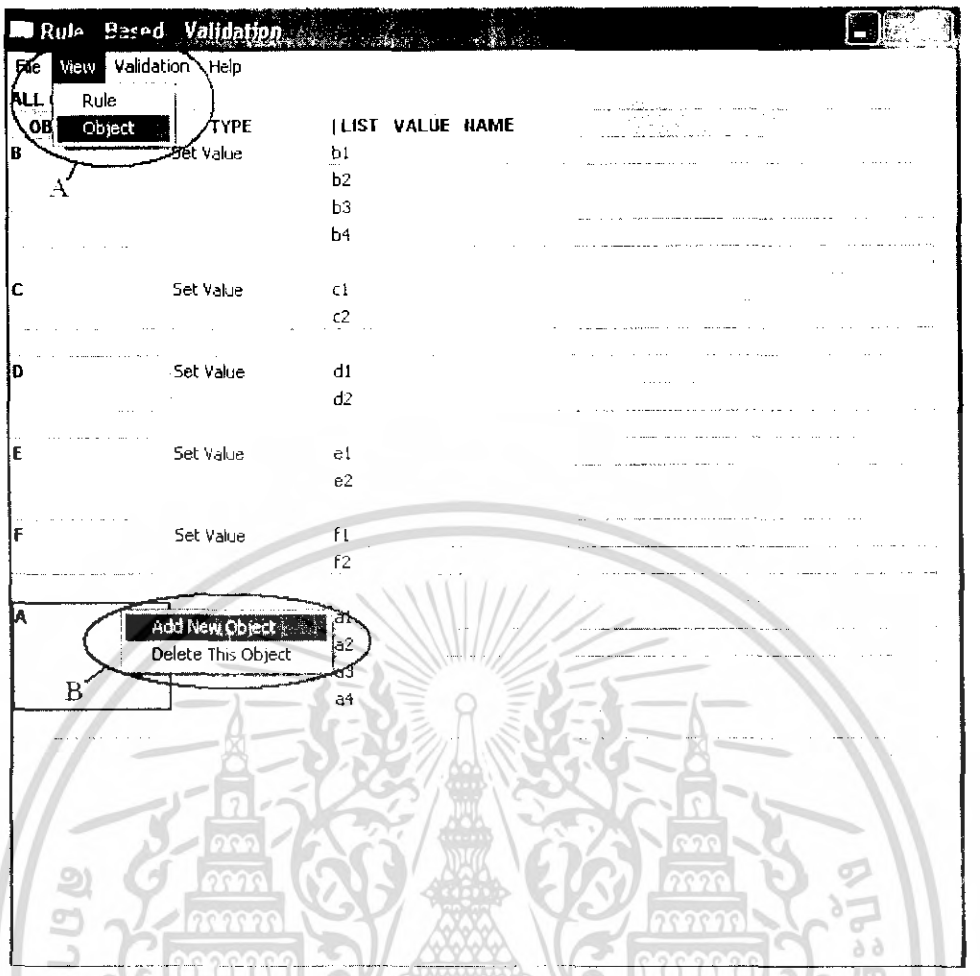
Traffic Light = {red, green} Car = {stop, run, run slow}

Rule A: IF Traffic Light is red THEN Car is stop

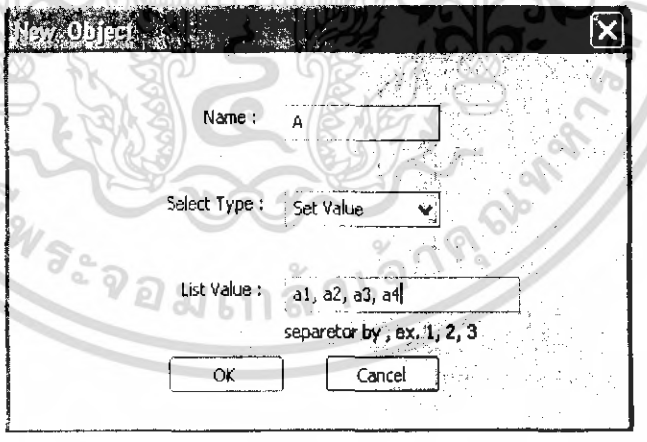
จากตัวอย่างที่ 6.1 แสดงก่อนที่จะสร้างกฎ A จะต้องมีการกำหนด Object Traffic Light ที่มี List Value ที่สามารถเป็นไปได้คือ red และ green และ Object Car ที่มี List Value ที่สามารถเป็นไปได้คือ stop, run และ run slow Object ที่ใช้ในการทดลองและ List Value ประกอบด้วย ดังตารางที่ 6.1

Object Name	Type Value	List Value
A	Set Value	a1, a2 , a3, a4
B	Set Value	b1, b2 , b3, b4
C	Set Value	c1, c2
D	Set Value	d1, d2
E	Set Value	e1, e2
F	Set Value	f1, f2

ตารางที่ 6.1 แสดง Object และ List Value ที่ใช้ใน Rule – Based สำหรับการทดลองนี้



รูปที่ 6.1 แสดง Object ที่ใช้ใน Rule – Based ที่สร้างมาจากตารางที่ 6.1



รปที่ 6.2 แสดงการสร้าง Object A และ List Value ประกอบด้วยค่า a1. a2.a3 และ a4

จากรูปที่ 6.1 วงกลมสีแดง A แสดงการเลือก -> View -> Object เมื่อคลิกจะทำการ Display Object ทั้งหมดของระบบ วงกลมสีแดง B แสดงเมื่อคลิกขวาที่ Object ใด ๆ บน Object View ในที่นี้คลิกขวาที่ Object A ถ้าคลิกที่ Delete This Object จะทำการลบ Object A ทิ้งจาก Object View แต่ถ้าคลิกที่ Add New Object จะเกิด Dialog ดังรูปที่ 6.2 ที่สามารถสร้าง Object ใหม่ได้

6.2 การสร้าง Rule ในการทดสอบ

หัวข้อที่ 6.1 ได้อธิบายการสร้าง Object ที่ใช้ในระบบ เมื่อสามารถสร้าง Object ได้ก็สามารถสร้าง Rule ในรูปแบบ OV ได้ โดยการสร้างส่วนประกอบของกฎที่ประกอบด้วยส่วน Premises และ Conclusion ซึ่งเป็น Clause ที่ประกอบด้วย Object, Operator และ Value ได้ดังตัวอย่างที่ 6.2 เป็นการสร้างกฎ A จาก Object ในตัวอย่างที่ 6.1

ตัวอย่างที่ 6.2

1. Create Object

```
object_traffic_light  =  Object ('Traffic Light', 'Set Value')
object_car            =  Object ('Car', 'Set Value')
```

2. Create List Value

```
object_traffic_light.AddValueToSetOfObject ('red')
object_traffic_light.AddValueToSetOfObject ('green')
object_car.AddValueToSetOfObject ('stop')
object_car.AddValueToSetOfObject ('run')
object_car.AddValueToSetOfObject ('run slow')
```

3. Create Clause

```
premise_clause    = Clause (object_traffic_light, 'is', 'red')
conclusion_clause = Clause (object_car, 'is', 'stop')
```

4. Create Rule

```
rule_A = Rule (premise_clause, conclusion_clause)
```

จากตัวอย่างที่ 6.2 อธิบายขั้นตอนการสร้างกฎจากตัวอย่างที่ 6.1 จาก Code ที่มาจาก Class ที่ได้จากการออกแบบในบทที่ 5 โดยขั้นตอนมีดังนี้ ขั้นตอนแรกสร้าง Object ที่จะใช้ในการสร้างกฎซึ่งในตัวอย่างประกอบด้วย Object Traffic Light และ Car ขั้นตอนที่สองสร้าง List Value แต่ละ Object ขั้นตอนที่สามสร้าง Clause ที่จะใช้ในส่วน Premises และ Conclusion ของ กฎโดยประกอบด้วย Object, Operator และ Value และขั้นตอนสุดท้าย สร้างกฎจาก Clause

ตัวอย่างที่ 6.3 เป็นตัวอย่างกฎที่ใช้ในการทดสอบที่สร้างมาจาก Object จากตารางที่ 6.1

ตัวอย่างที่ 6.3

```
IF  A is a1  AND  B is b1  THEN  F is f1
IF  A is a2  AND  B is b2  THEN  F is f2
```

IF A is a3 AND C is c1 THEN E is e1

IF A is a4 AND C is c2 THEN E is e2

IF A is a1 AND B is b1 THEN F is f1

IF A is a2 AND B is not b1 THEN F is f2

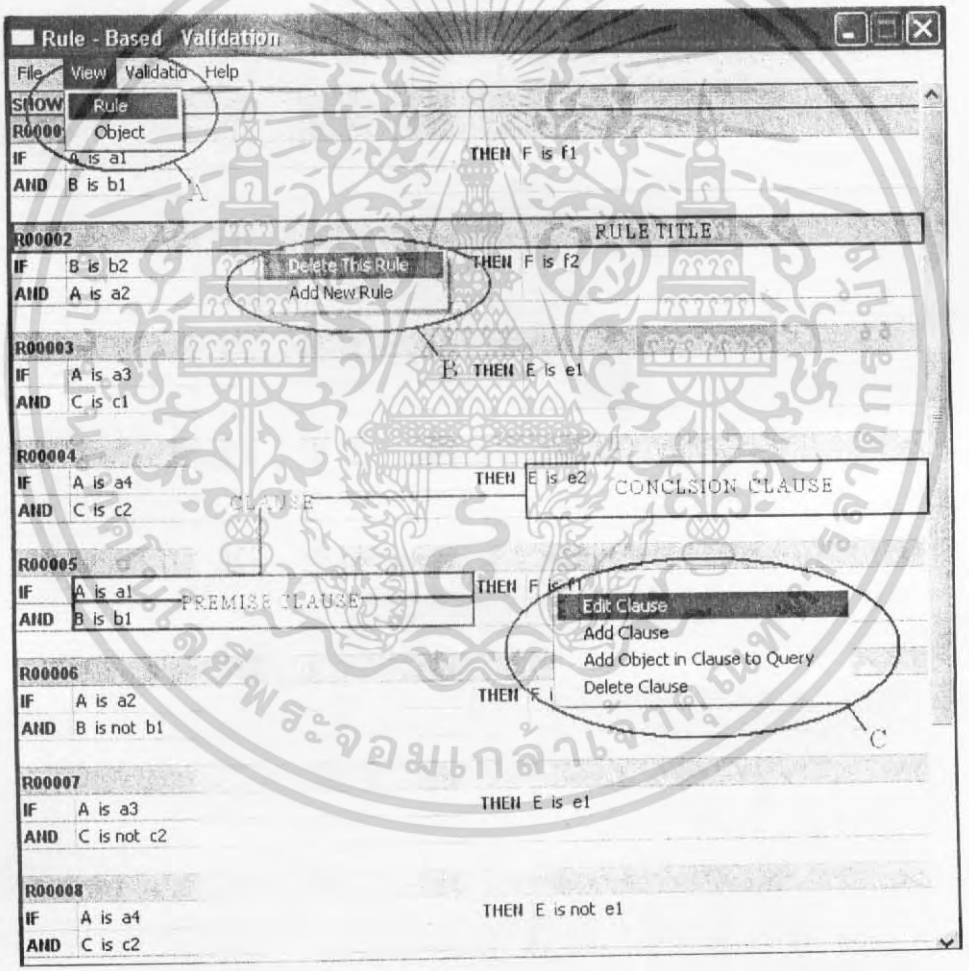
IF A is a3 AND C is not c2 THEN E is e1

IF A is a4 AND C is not c2 THEN E is not e1

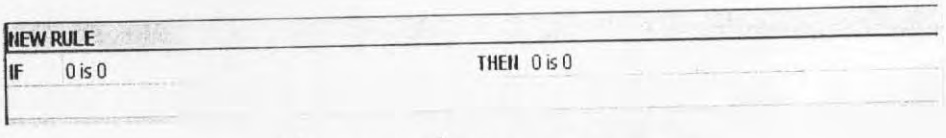
IF D is d1 THEN F is f1

IF D is d2 THEN F is f2

IF D is not d1 THEN F is not f1

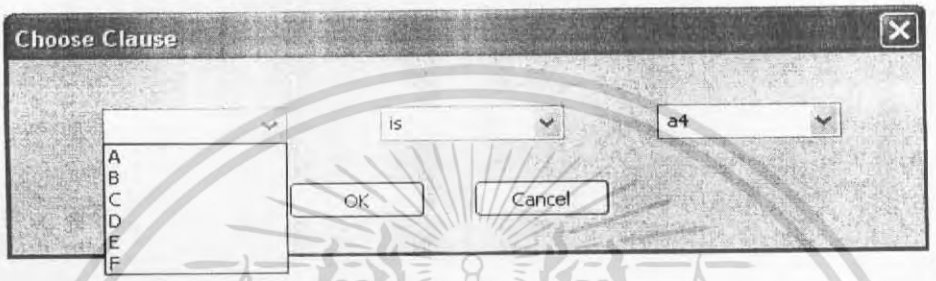


รูปที่ 6.3 แสดง Rule ในตัวอย่างที่ 6.3 และ อธิบายส่วนประกอบ Rule View



รูปที่ 6.4 แสดงมื่อคลิก Add New Rule

จากรูปที่ 6.3 วงกลมสีแดง A แสดงการเลือก -> View -> Rule เมื่อคลิกจะทำการ Display Rule ทั้งหมดของระบบ วงกลมสีแดง B แสดงเมื่อคลิกขวาที่ Rule title ใด ๆ บน Rule View ในที่นี้คลิกขวาที่ Rule R00002 ถ้าคลิกที่ Delete This Rule จะทำการลบ Rule R00002 ทิ้งจาก Rule View แต่ถ้าคลิกที่ Add New Rule จะเกิด Rule ใหม่ดังรูปที่ 6.4 แทรกเข้าไปที่ Rule View วงกลมสีแดง C แสดงเมื่อคลิกขวาที่ Clause ทั้งในส่วน Premise และ Conclusion ซึ่งสามารถที่จะ Edit Clause, Add Clause, Delete Clause และ Add Object in Clause to Query



รูปที่ 6.5 แสดง Dialog เมื่อคลิกที่ Edit Clauseหรือ Add Clause

จากรูปที่ 6.5 แสดง Dialog เมื่อคลิกที่ Edit Clause หรือAdd Clause ในรูปที่ 6.3 วงกลมสีแดง C จะเกิด Dialog ที่ให้ทำการเลือกส่วนประกอบของ Clause ต้องการ Edit หรือ Add

6.3 การทดสอบการซ้ำซ้อนของกฎ

Rule ที่ใช้ในการทดลองประกอบด้วยดังตัวอย่างที่ 6.3

หลังจากการตรวจสอบจะมีการลบกฎที่ซ้ำซ้อนเองโดยแอพพลิเคชั่น ทำให้เหลือกฎตามข้างล่าง

IF D is d1 THEN F is f1

IF D is d2 THEN F is f2

IF A is a1 AND B is b1 THEN F is f1

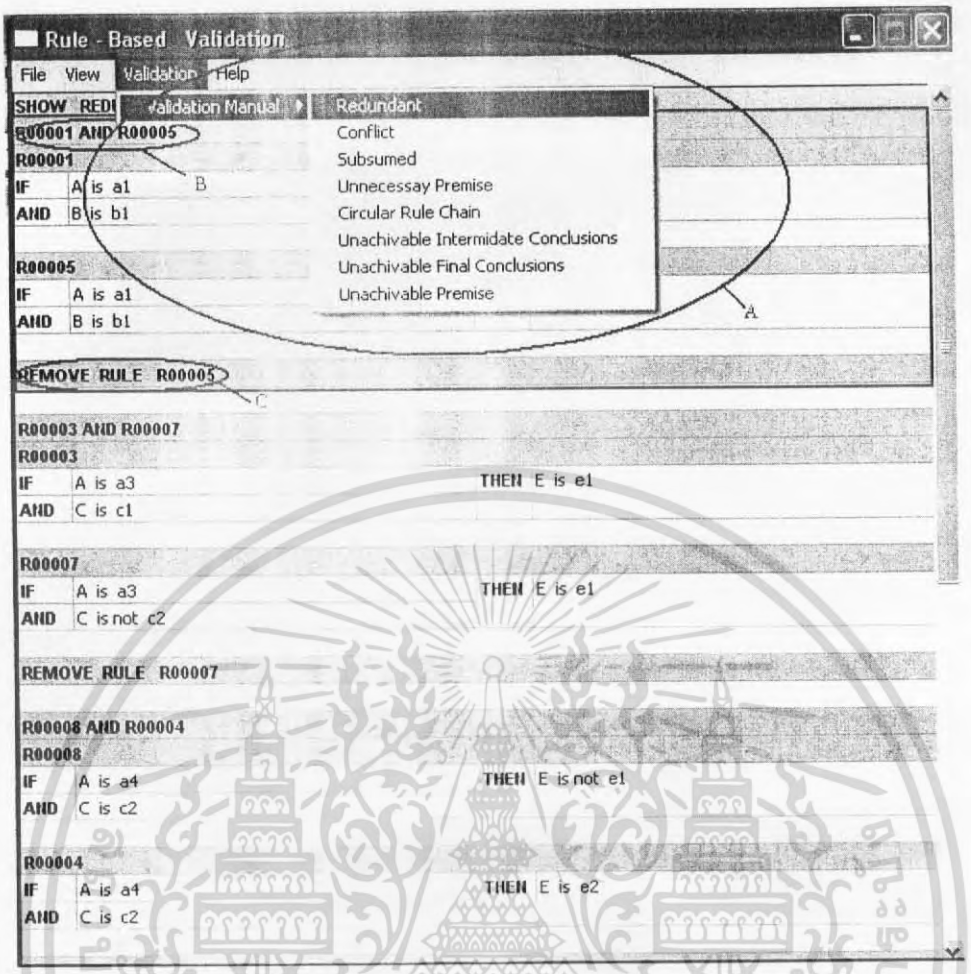
IF A is a2 AND B is b2 THEN F is f2

IF A is a3 AND C is c1 THEN E is e1

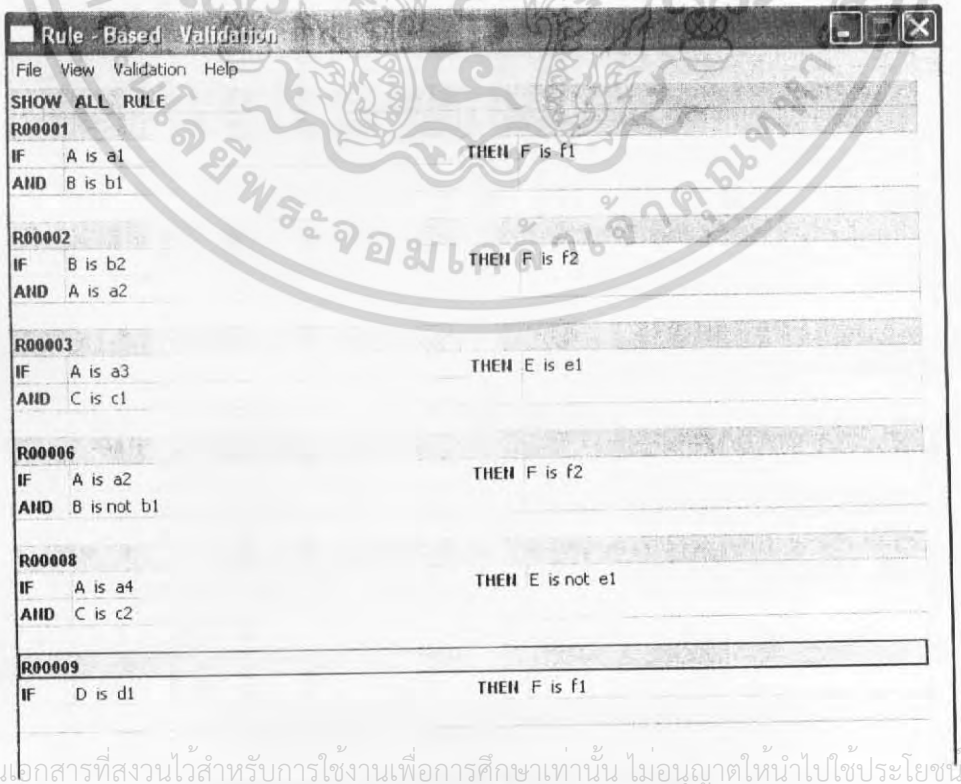
IF A is a2 AND B is not b1 THEN F is f2

IF A is a4 AND C is not c2 THEN E is not e1

จากรูปที่ 6.6 วงกลมสีแดง A แสดงการเลือก -> Validation -> Validation Manual -> Redundant เมื่อคลิกจะทำการ Display Rule Validation (Redundant) ทั้งหมดของระบบ วงกลมสีแดง B แสดงชื่อ Rule ที่ขัดแย้งกันและวงกลมสีแดง C แสดงชื่อ Rule ที่ลบโดยแอพพลิเคชั่น



รูปที่ 6.6 แสดงเมื่อตรวจสอบ Redundant Rule จาก Rule ตัวอย่างที่ 6.3



รูปที่ 6.7 แสดง Rule ตัวอย่างที่ 6.3 หลังจากการตรวจสอบ

6.4 การทดสอบการขัดแย้งของกฎ

Rule ที่ใช้ในการทดลองประกอบด้วยดังตัวอย่างที่ 6.4

ตัวอย่างที่ 6.4

IF D is d1 THEN F is f1

IF D is d2 THEN F is f2

IF D is d2 THEN F is f1

IF A is a1 AND B is b1 THEN F is f1

IF A is a2 AND B is b2 THEN F is f2

IF A is a3 AND B is b2 AND C is c1 THEN E is e1

IF A is a2 AND B is not b1 THEN F is f2

IF A is a3 AND B is b2 AND C is c1 THEN E is not e1

IF A is a2 AND B is not b1 THEN F is f1

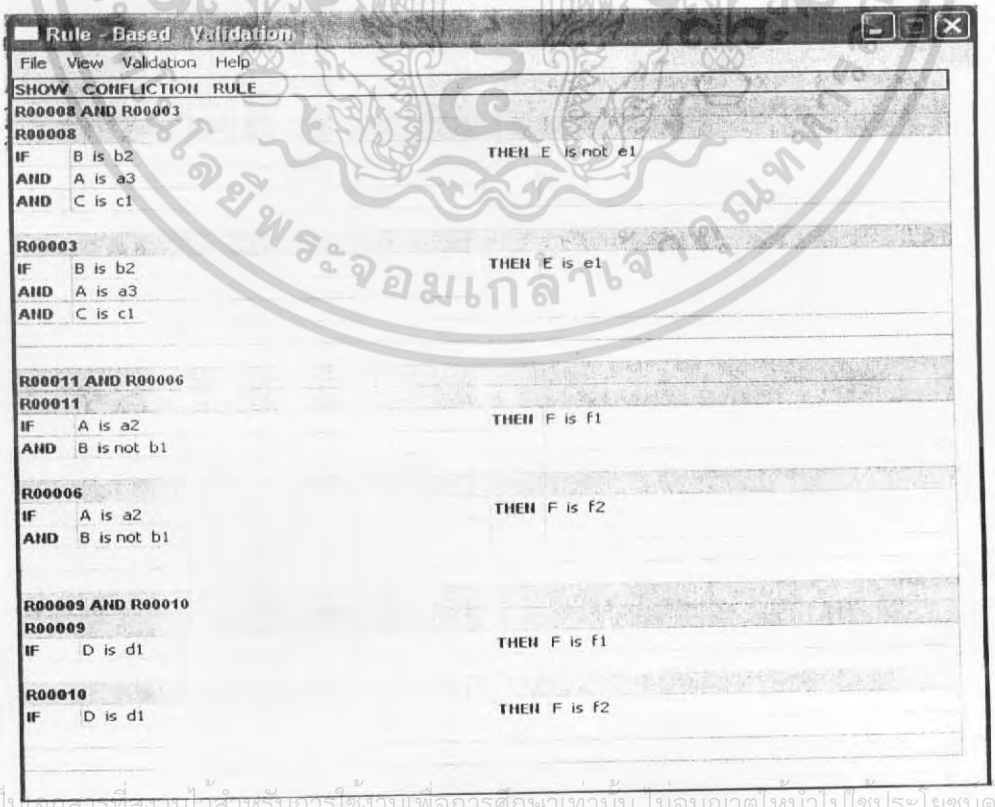
หลังจากการตรวจสอบพบกฎที่ขัดแย้งกัน ดังนี้

IF A is a3 AND B is b2 AND C is c1 THEN E is e1 กับ

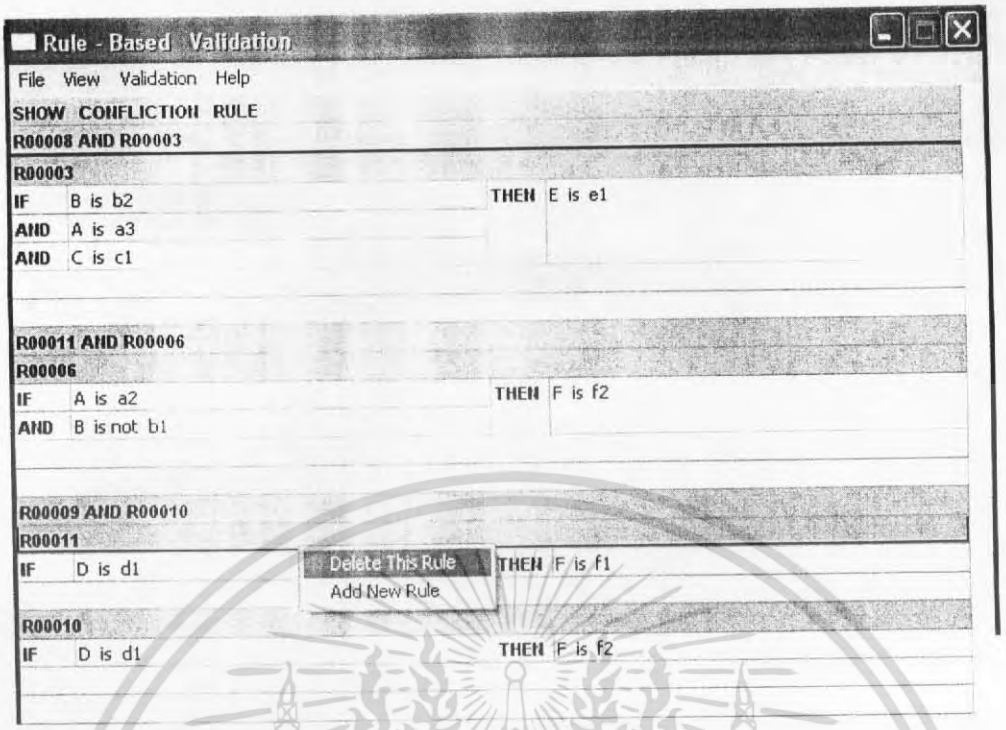
IF A is a3 AND B is b2 AND C is c1 THEN E is not e1

IF A is a2 AND B is not b1 THEN F is f1 กับ IF A is a2 AND B is not b1 THEN F is f2

IF D is d2 THEN F is f1 กับ IF D is d2 THEN F is f2



รูปที่ 6.8 แสดงเมื่อตรวจสอบ Conflict Rule จากกฎตัวอย่างที่ 6.4



รูปที่ 6.9 แสดงการแก้ไข Conflict Rule โดยให้ User ทำการ Delete Rule

6.5 การทดสอบการซับซัมกันของกฎ (Subsumed Rules)

Rule ที่ใช้ในการทดลองประกอบด้วยดังตัวอย่างที่ 6.5

ตัวอย่างที่ 6.5

IF A is a2 THEN F is f2

IF A is a1 AND B is b1 THEN F is f1

IF A is a2 AND B is b2 THEN F is f2

IF A is a3 AND B is b3 THEN F is f1

IF A is a4 AND B is b4 THEN F is f2

IF A is a2 AND B is not b1 THEN F is f2

IF A is a2 AND B is b2 AND C is c1 THEN E is e1

IF A is a3 AND B is b2 AND C is c1 THEN E is e1

IF A is a3 AND B is b2 AND C is c1 AND D is d1 THEN E is e1

IF A is a4 AND B is b4 AND C is c1 AND D is d1 THEN F is not f1

หลังจากการตรวจสอบพบกฎที่ซับซัมกันของกฎ ดังนี้

IF A is a2 AND B is b2 THEN F is f2

IF A is a2 AND B is not b1 THEN F is f2

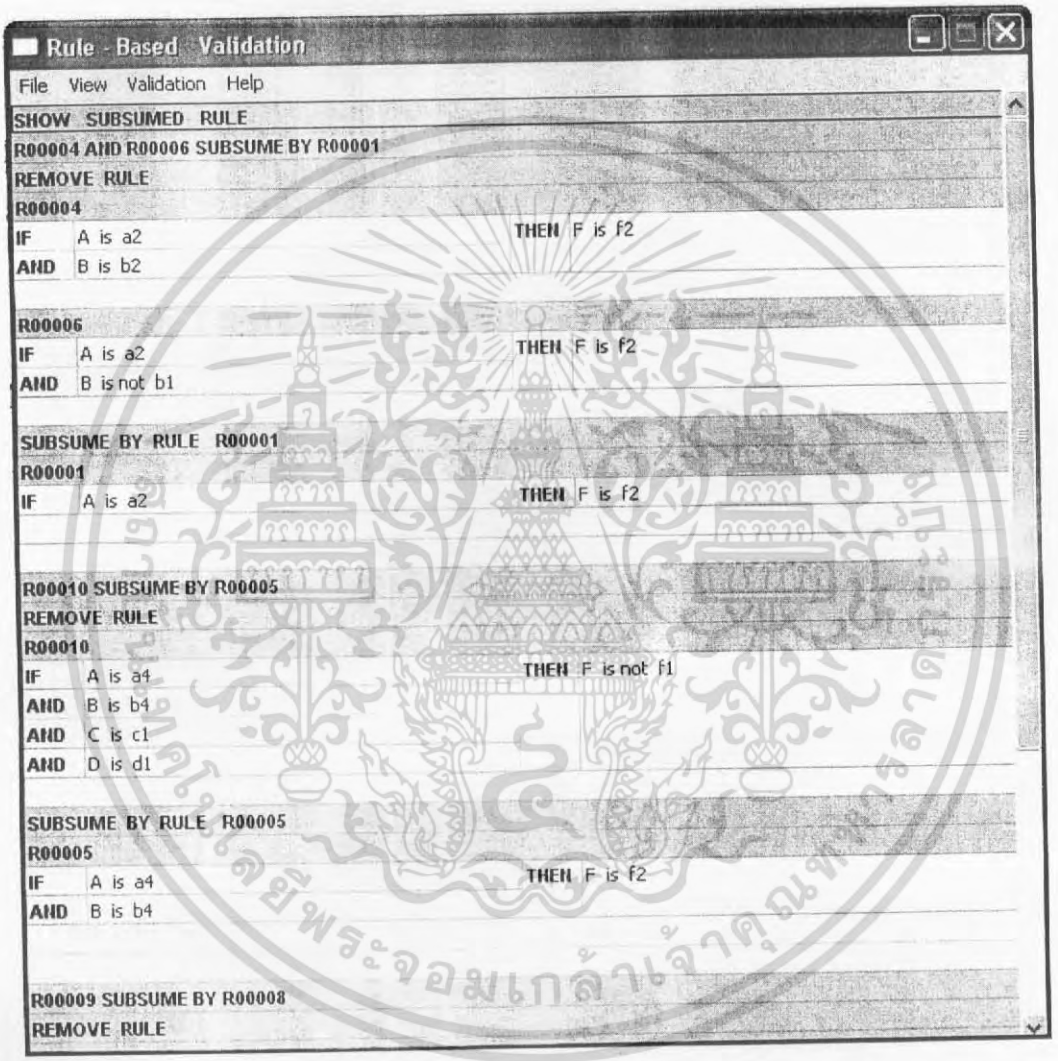
ซับซัมกับ IF A is a2 THEN F is f2

IF A is a4 AND B is b4 AND C is c1 AND D is d1 THEN F is not f1

ซับซัมกับ IF A is a4 AND B is b4 THEN F is f2

IF A is a3 AND B is b2 AND C is c1 AND D is d1 THEN E is e1

ซับซัมกับ IF A is a3 AND B is b2 AND C is c1 THEN E is e1



รูปที่ 6.10 แสดงการตรวจสอบ Subsumed Rules ของตัวอย่างที่ 6.5

จะทำการลบ Rule เองโดยอัตโนมัต จะเหลือ Rule ในตัวอย่างที่ 3 ดังนี้

IF A is a2 THEN F is f2

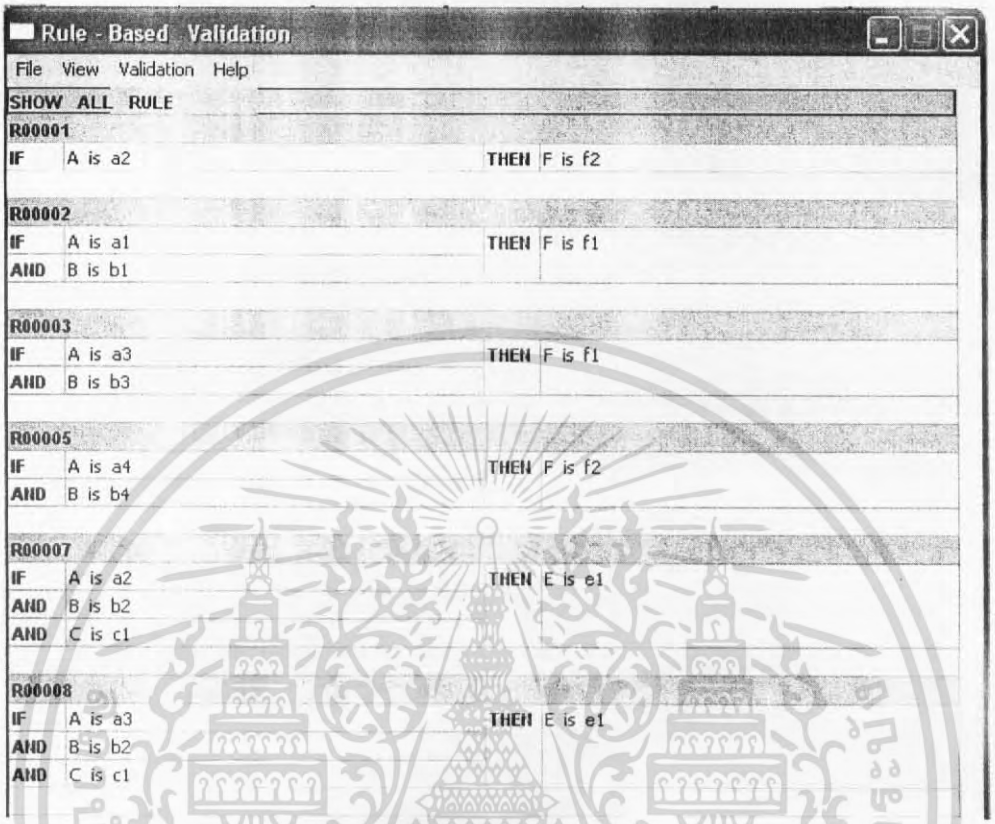
IF A is a1 AND B is b1 THEN F is f1

IF A is a3 AND B is b3 THEN F is f1

IF A is a4 AND B is b4 THEN F is f2

IF A is a2 AND B is b2 AND C is c1 THEN E is e1

IF A is a3 AND B is b2 AND C is c1 THEN E is e1



รูปที่ 6.11 แสดงผลหลังการการตรวจสอบ Subsumed Rules ของตัวอย่างที่ 6.7

6.6 การทดสอบ Premise Clause ที่ไม่จำเป็นของกฎ

Rule ที่ใช้ในการทดลองประกอบด้วยดังตัวอย่างที่ 6.6

ตัวอย่างที่ 6.6

IF A is a2 THEN F is f2

IF A is a1 AND B is b1 THEN F is f1

IF A is a3 AND B is b3 THEN F is f1

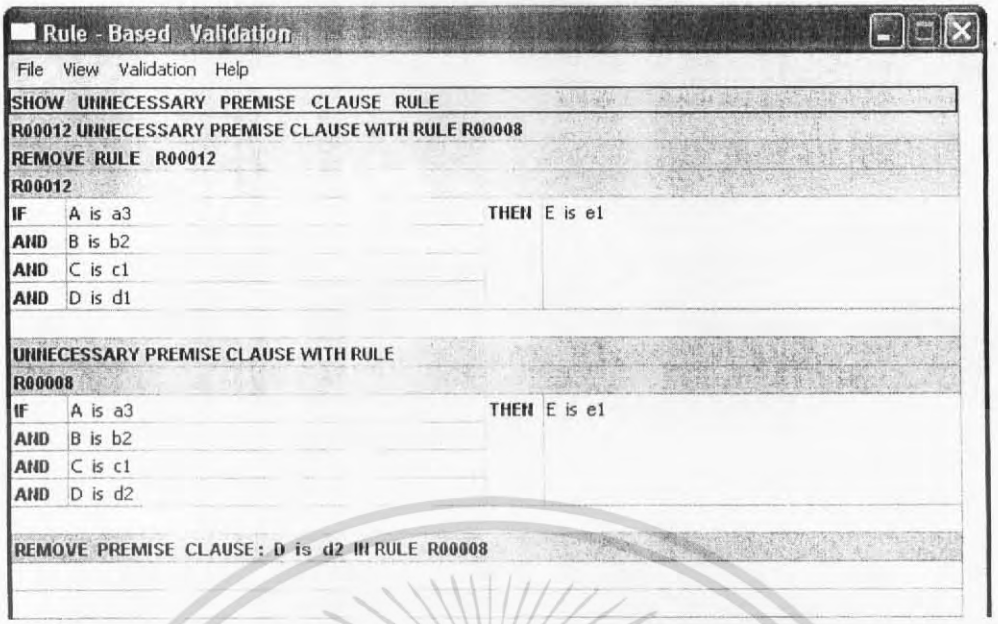
IF A is a4 AND B is b4 THEN F is f2

IF A is a3 AND B is b2 AND C is c1 THEN E is e1

IF A is a3 AND B is b2 AND C is c1 AND D is d1 THEN E is e1

IF A is a3 AND B is b3 AND C is c1 AND D is d1 THEN E is e1

IF A is a3 AND B is b2 AND C is c1 AND D is d2 THEN E is e1



รูปที่ 6.12 แสดงผลการตรวจสอบ Unnecessary Premise Rules ของตัวอย่างที่ 6.6

6.7 การทดสอบ Circular Rule Chain ของกฎ

Rule ที่ใช้ในการทดลองประกอบด้วยดังตัวอย่างที่ 6.7

ตัวอย่างที่ 6.7

Goal : F is f2

IF A is a2 THEN F is f2

IF D is d1 AND B is b1 THEN E is e1

IF D is d2 AND B is b3 THEN E is e2

IF A is a3 AND B is b2 AND C is c1 THEN E is e1

IF F is f2 THEN D is d1

IF E is e2 THEN A is a2

IF E is e1 THEN A is a1

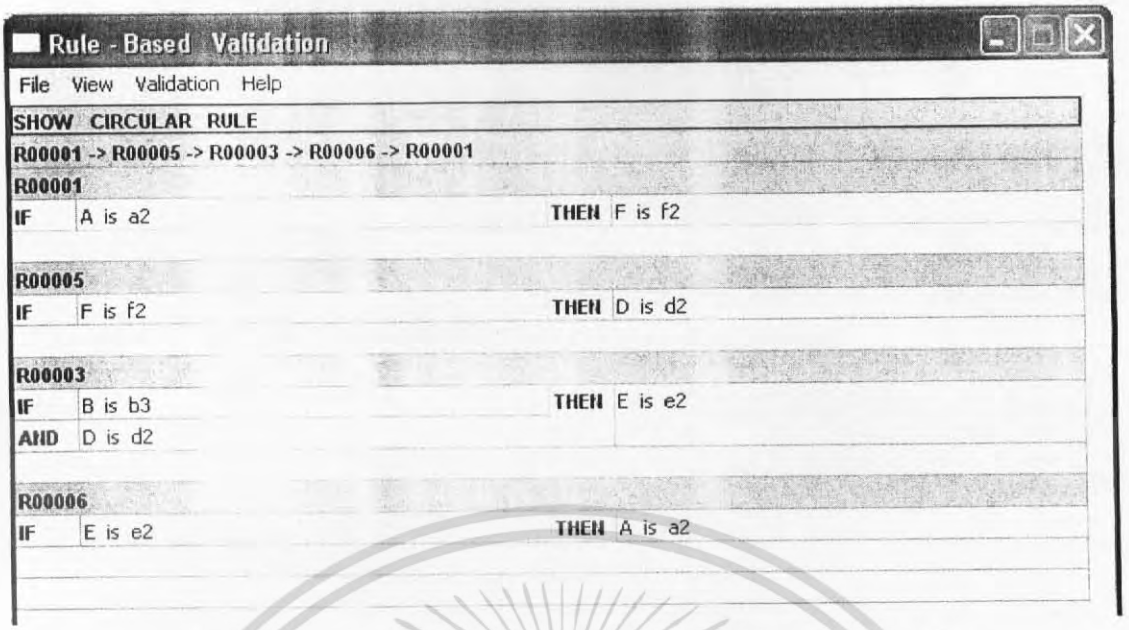
จะได้ Circular Rule

IF A is a2 THEN F is f2

IF F is f2 THEN D is d2

IF D is d2 AND B is b3 THEN E is e2

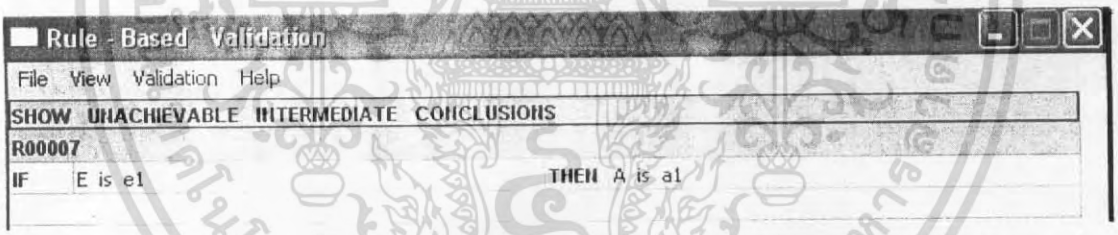
IF E is e2 THEN A is a2



รูปที่ 6.13 แสดงผลการตรวจสอบ Circular Rule Chain ของตัวอย่างที่ 6.7

6.8 การทดสอบ Unachievable Intermediate Conclusions

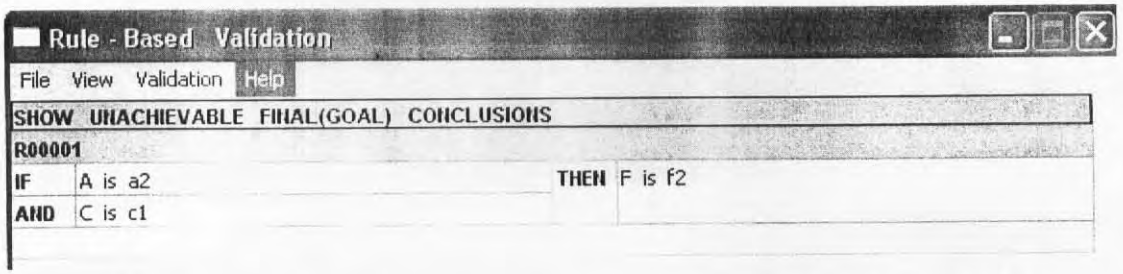
Rule ที่ใช้ในการทดลองประกอบด้วยดังตัวอย่างที่ 6.7



รูปที่ 6.14 แสดงผลการตรวจสอบ Unachievable Intermediate Conclusions ของตัวอย่างที่ 6.7

6.9 การทดสอบ Unachievable Final Conclusions

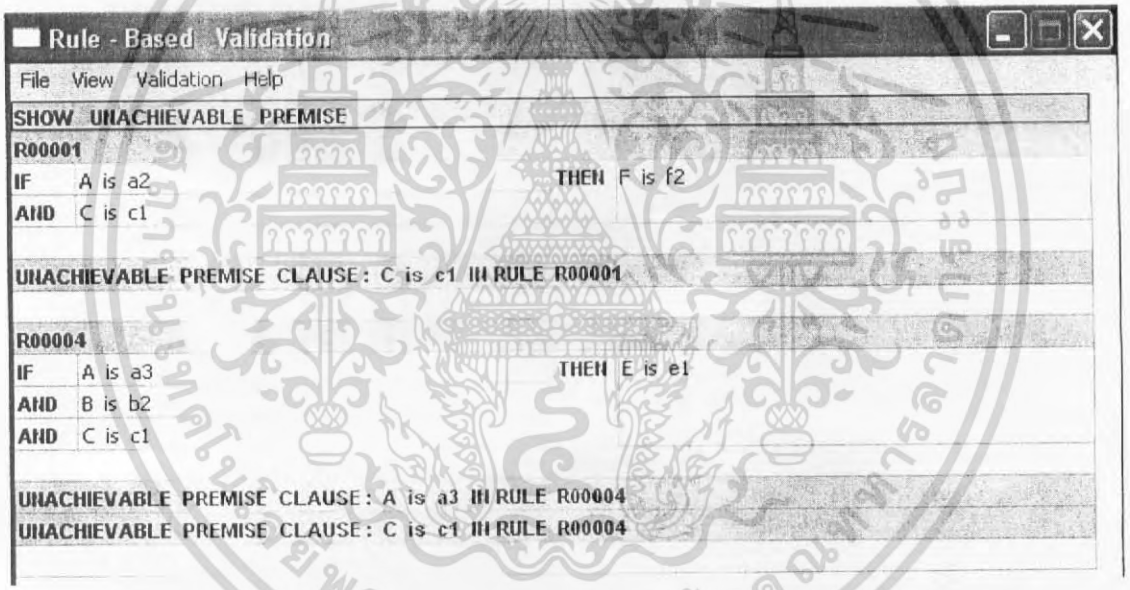
Rule ที่ใช้ในการทดลองประกอบด้วยดังตัวอย่างที่ 6.7 และ ถ้ามี Query Premise Object ={D, B} แต่ไม่มี Object A , C และเพิ่ม Clause C is c เข้าไปใน Rule IF A is a2 THEN F is f2 จะไม่สามารถสำเร็จได้เนื่องจาก ไม่สามารถค้นหา Fact C is c



รูปที่ 6.15 แสดงผลการตรวจสอบ Unachievable Final Conclusions ของตัวอย่างที่ 6.7

6.10 การทดสอบ Unachievable Premise

Rule ที่ใช้ในการทดลองประกอบด้วยดังตัวอย่างที่ 6.7 และ ถ้ามี Query Premise Object ={D, B} แต่ไม่มี Object A , C และเพิ่ม Clause C is c เข้าไปใน Rule IF A is a2 THEN F is f2 จะไม่สามารถสำเร็จได้เนื่องจาก ไม่สามารถค้นหา Fact C is c



รูปที่ 6.16 แสดงผลการตรวจสอบ Unachievable Premise ของตัวอย่างที่ 6.7

บทที่ 7
สรุปและวิจารณ์

7.1 สรุปผลการทดลอง

จากผลการทดลองสามารถสรุปได้ดังนี้

- ระบบสามารถสร้างกฎจากระบบ โดยการสร้าง Object และ List Value แล้วนำ Object เหล่านั้นมาสร้างกฎได้
- ระบบสามารถทำการแก้ไขกฎได้ตามที่ต้องการในขอบเขตรูปแบบที่กำหนด
- ระบบสามารถลดกฎได้เองในกรณีที่แก้ไขเองได้ เช่น Redundant และ Subsumed Rule
- ระบบสามารถแจ้งต่อผู้ใช้ได้ในกรณีที่แก้ไขเองไม่ได้ เช่น Conflict Rule
- การทดสอบตรวจสอบกฎเป็นไปตามที่มีการออกแบบ ได้กฎที่มีจำนวนลดลง ซึ่งกฎที่มีจำนวนลดลงไม่มีกฎที่ขัดแย้ง ซ้ำซ้อนกัน

7.2 ข้อจำกัดของการใช้งานระบบ

- การตรวจสอบยังได้รูปแบบความไม่สมบูรณ์ของกฎในฐานกฎความรู้ที่เกี่ยวข้องกับโครงงาน มี 2 รูปแบบคือ กฎขาดความกลมกลืน (Inconsistency)และส่วนประกอบของกฎไม่มีความสมบูรณ์ (Incompleteness)
- ระบบยังไม่สามารถรับอินพุทของกฎรูปแบบที่เชื่อมด้วย OR
- ระบบยังเป็นกฎที่แสดงความรู้ในรูปแบบ OV ซึ่งแสดงความรู้ไม่ชัดเจนเท่ากับรูปแบบ OAV
- ชนิด Data Type ของ Value ยังเป็น String อยู่ซึ่งยังไม่หลากหลาย

7.3 แนวทางในการพัฒนาต่อ

- สามารถพัฒนาต่อเป็น Expert System Shell ที่สามารถตรวจสอบกฎได้
- สามารถพัฒนาต่อเป็นการแสดงความรู้เป็นแบบ OAV
- สามารถพัฒนาต่อให้รับกฎแบบที่เชื่อมด้วย OR
- สามารถพัฒนาต่อให้จัดการตรวจสอบความรู้แบบ Uncertainty ได้

เอกสารอ้างอิง

Michael Negnevisky. 2005. **Artificial Intelligence: A Guide to Intelligent Systems.** 2nd ed. Harlow:Addison-Wesley.

James P. Ignizio. 1991.**An introduction to expert systems :the development and implementation of rule-based expert systems.** New York:McGraw-Hill

C. Dadkhah and A. Ahmad, "**OAV-VVT Expert, An active system for Verification and Validation of KB Based onEx-OAV KB**", ICSE & INCOSE Joint 2004 Conference, Las Vegas, USA, 2004.