การพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ

นางสาวกาญจนา อินทรชาธร

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2555

DEVELOPMENT OF EXPERT SYSTEM SHELL WITH CONSTRAINT LOGIC PROGRAMMING

Kanjana Intharachatorn

A Thesis Submitted in Partial Fulfillment of the Requirements for the

Degree of Master of Engineering in Computer Engineering

Suranaree University of Technology

Academic Year 2012

การพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยวิธีการเขียนโปรแกรมเชิงตรรกะ ด้วยเงื่อนไขบังคับ

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

คณะกรรมการสอบวิทยานิพนธ์

____

(รศ. ดร.กิตติศักดิ์ เกิดประสพ)
ประธานกรรมการ

____

(รศ. ดร.นิตยา เกิดประสพ)
กรรมการ (อาจารย์ที่ปรึกษาวิทยานิพนธ์)

____

(อ. ดร.ธรา อั่งสกุล)
กรรมการ

____

(ศ. ดร.ชูกิจ ลิมปิจำนงค์)
รองอธิการบดีฝ่ายวิชาการ

____

(รศ. ร.อ. ดร.กนต์ธร ชำนิประศาสน์)
คณบดีสำนักวิชาวิศวกรรมศาสตร์

กาญจนา อินทรชาธร : การพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ (DEVELOPMENT OF EXPERT SYSTEM SHELL WITH CONSTRAINT LOGIC PROGRAMMING) อาจารย์ที่ปรึกษา: รองศาสตราจารย์ ดร.นิตยา เกิดประสพ, 126 หน้า.

งานวิจัยนี้มีวัตถุประสงค์เพื่อพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับสำหรับฐานความรู้ทั่วไปที่สามารถเขียนอยู่ในรูปแบบของเงื่อนไขบังคับ และมีชนิดข้อมูลเป็นประเภทตัวอักษร ข้อความ หรือสัญลักษณ์ โดยระบบสามารถโต้ตอบเพื่อรับข้อมูลปัญหาจากผู้ใช้ระบบ ให้คำตอบของปัญหา และมีการอธิบายถึงที่มาของคำตอบที่ได้อย่างสมเหตุสมผล เครื่องมือที่ใช้ในการพัฒนาระบบ คือ โปรแกรม ECLiPSe 6.0 เนื่องจากมีภาษาโปรล็อกเป็นภาษาพื้นฐาน และเป็นโปรแกรมสำหรับพัฒนาระบบเชิงตรรกะด้วยเงื่อนไขบังคับโดยเฉพาะ ในการวัดผลการวิจัยจะกระทำโดยการเปรียบเทียบประสิทธิภาพกับเปลือกระบบผู้เชี่ยวชาญที่เขียนด้วยภาษาโปรล็อก โดยเปรียบเทียบในด้านเวลา และปริมาณหน่วยความจำที่ใช้ภายในขอบเขตการทำงานที่คล้ายคลึงกันของระบบ และใช้ฐานความรู้ที่ได้จากแหล่งเดียวกันในการทดสอบระบบ แล้วทำการสังเกตและบันทึกค่าผลการทำงานเพื่อสรุปผลการวิจัย

ผลที่ได้จากงานวิจัยนี้สรุปได้ว่า การพัฒนาด้วยวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับนั้น สามารถลดขนาดของไฟล์ฐานความรู้ให้เล็กลงได้ ช่วยลดปริมาณการใช้หน่วยความจำทั้งในขณะประมวลผลและการจัดเก็บไฟล์ข้อมูล และช่วยลดเวลาที่ใช้ในการแสดงผลของระบบโดยรวมได้อย่างมีประสิทธิภาพ

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ลายมือชื่อนักศึกษา____________________

ปีการศึกษา 2555 ลายมือชื่ออาจารย์ที่ปรึกษา______________

KANJANA INTHARACHATORN : DEVELOPMENT OF EXPERT SYSTEM SHELL WITH CONSTRAINT LOGIC PROGRAMMING. THESIS ADVISOR : ASSOC. PROF. NITTAYA KERDPRASOP, Ph.D., 126 PP.

CONSTRAINT LOGIC PROGRAMMING/ EXPERT SYSTEM SHELL/ KNOWLEDGE BASE/ SYMBOLIC DOMAIN

This research aims to develop expert system shell with constraint logic programming for any knowledge base that can be encoded with constraints in symbolic domain. The system can interact with users to obtain queries and it can provide answers with reasonable explanation. The system uses ECLiPSe 6.0 as a compiler because it is based on Prolog language and suitable for the development of constraint logic programs. For evaluation of this research, the system's performance has been compared against the expert system shell developed with Prolog. Evaluation criteria are execution time, memory usage, and lines of code, performed under similar procedures on knowledge base that originate from the same source. Then observe and record the results to summarize the research.

With the obtained results, we can conclude that the size of knowledge base get smaller when it was developed with constraint logic programming. The constraint-based expert system shell consumes less amount of memory and works faster than the shell that was developed with Prolog language.

School of Computer Engineering	Student's Signature__________________
Academic Year 2012	Advisor's Signature__________________

กิตติกรรมประกาศ

วิทยานิพนธ์นี้สำเร็จลุล่วงด้วยดี ผู้วิจัยขอกราบขอบพระคุณ บุคคล และกลุ่มบุคคลต่าง ๆ ที่ได้กรุณาให้คำปรึกษา แนะนำ ช่วยเหลืออย่างดียิ่ง ทั้งในด้านวิชาการ และในด้านการดำเนินงานวิจัย ดังต่อไปนี้

รองศาสตราจารย์ ดร. นิตยา เกิดประสพ อาจารย์ที่ปรึกษาวิทยานิพนธ์

รองศาสตราจารย์ ดร. กิตติศักดิ์ เกิดประสพ หัวหน้าสาขาวิชาวิศวกรรมคอมพิวเตอร์ ผู้ช่วยศาสตราจารย์ ดร. คะชา ชาญศิลป์ ผู้ช่วยศาสตราจารย์ ดร. ปรเมศวร์ ห่อแก้ว ผู้ช่วยศาสตราจารย์ ดร. พิชโยทัย มหัทธนาภิวัฒน์ และ ผู้ช่วยศาสตราจารย์ สมพันธุ์ ชาญศิลป์ อาจารย์ประจำสาขาวิชาวิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

อาจารย์ ดร. ธรา อั่งสกุล อาจารย์ประจำสาขาวิชาเทคโนโลยีสารสนเทศ สำนักวิชาเทคโนโลยีสังคม มหาวิทยาลัยเทคโนโลยีสุรนารี ให้คำแนะนำระหว่างการสอบโครงร่างวิทยานิพนธ์

ผู้ช่วยศาสตราจารย์ วัชรชัย วิริยะสุทธิวงศ์ อาจารย์ประจำคณะวิศวกรรมศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ ที่ให้คำแนะนำระหว่างการนำเสนอบทความวิจัยในงานประชุมวิชาการ

คุณกัลญา พับโพธิ์ เลขานุการสาขาวิชาวิศวกรรมคอมพิวเตอร์ ที่ให้ความช่วยเหลือในการประสานงานด้านเอกสารต่าง ๆ ระหว่างศึกษา

นักศึกษาบัณฑิตศึกษา สาขาวิชาวิศวกรรมคอมพิวเตอร์ ทุกท่านที่ให้คำปรึกษาและช่วยเหลือด้วยดีมาโดยตลอด

นอกจากนี้ ขอขอบคุณครู อาจารย์ทั้งในอดีตและปัจจุบันที่ให้ความรู้แก่ผู้วิจัยจนประสบความสำเร็จในชีวิต

สุดท้ายนี้ ขอกราบขอบพระคุณ บิดา มารดา ที่ให้กำเนิด อบรม เลี้ยงดูด้วยรัก และส่งเสริมการศึกษาเป็นอย่างดีมาโดยตลอด ทำให้ผู้วิจัยมีความรู้ความสามารถ มีจิตใจที่เข้มแข็ง รวมทั้งเป็นกำลังใจที่ยิ่งใหญ่ให้แก่ผู้วิจัย จนทำให้ผู้วิจัยประสบความสำเร็จในชีวิตเรื่อยมา

กาญจนา อินทรชาธร

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของปัญหาการวิจัย

ระบบผู้เชี่ยวชาญ เป็นสาขาหนึ่งของศาสตร์ด้านปัญญาประดิษฐ์ เน้นการศึกษาเรื่องการสร้างระบบความรู้เพื่อแก้ปัญหาเฉพาะด้าน เช่น ทางการแพทย์ เป็นต้น โดยมีจุดประสงค์หลักในการพัฒนาระบบให้เสมือนมีผู้เชี่ยวชาญที่เป็นมนุษย์คอยให้คำปรึกษา คำแนะนำ และตอบคำถามเกี่ยวกับปัญหาต่าง ๆ (วิกิพีเดีย สารานุกรมเสรี, 2554ข)

ระบบผู้เชี่ยวชาญมีการพัฒนามาอย่างต่อเนื่องตั้งแต่ปีค.ศ. 1965 โดยทีมวิจัยของมหาวิทยาลัยสแตนฟอร์ด ผู้ริเริ่มโครงการสร้างระบบที่สามารถให้เหตุผลทางวิทยาศาสตร์ได้ ชื่อว่าโปรแกรม DENDRAL ซึ่งได้กลายเป็นต้นแบบของระบบผู้เชี่ยวชาญในรุ่นต่อ ๆ มา (Lindsay, Buchanan, Feigenbaum, and Lederberg, 1980; Feigenbaum, 1992) พัฒนาการต่อมาของระบบผู้เชี่ยวชาญ คือ โปรแกรมผู้เชี่ยวชาญที่ชื่อว่า MYCIN เป็นต้นแบบของการโต้ตอบกันระหว่างมนุษย์และระบบ ต่อมาได้มีการพัฒนาต่อยอดจนได้เปลือกระบบผู้เชี่ยวชาญที่เรียกว่า EMYCIN และกลายเป็นต้นแบบของเปลือกระบบผู้เชี่ยวชาญในเชิงพาณิชย์ (Buchanan and Shortliffe, 1984; Melle, Shortliffe, and Buchanan, 1984; Feigenbaum, 1992)

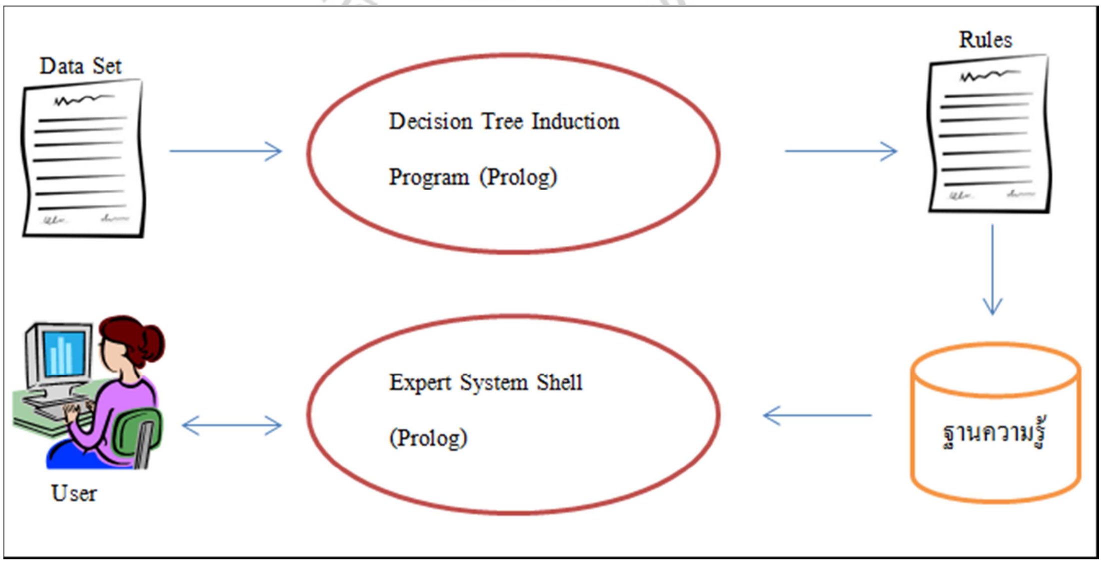
ตัวอย่างโดยทั่วไปของระบบผู้เชี่ยวชาญจะเขียนด้วยภาษา Prolog, Common Lisp, C/C++, COBOL, CLIP, Java และ CLP เป็นต้น ขอบเขตของงานในระบบผู้เชี่ยวชาญในปัจจุบันมุ่งเน้นไปที่การสร้างระบบให้คำปรึกษาในการตัดสินใจบางอย่างที่เป็นปัญหาในขณะนั้นของผู้ใช้ระบบโดยตั้งอยู่บนพื้นฐานของความเป็นเหตุเป็นผล และด้วยความสามารถทางด้านการแทนความรู้และนิรนัย (Knowledge Representation and Deduction) ที่รวมอยู่ในกระบวนการพัฒนาระบบผู้เชี่ยวชาญ ทำให้ระบบนี้มีความสำคัญต่องานหลาย ๆ ด้านที่จำเป็นจะต้องพึ่งความเชี่ยวชาญเฉพาะด้านของผู้เชี่ยวชาญ และนอกจากนี้แล้วความต้องการในการใช้งานระบบผู้เชี่ยวชาญที่มากขึ้น ทั้งในด้านอุตสาหกรรม ส่วนงานภาครัฐ หรือด้านการศึกษา (Salim, Villavicencio, and Timmerman, 2002) ทำให้เปลือกระบบผู้เชี่ยวชาญมีบทบาทสำคัญขึ้นมาในการพัฒนาระบบผู้เชี่ยวชาญ

องค์ประกอบสำคัญในระบบผู้เชี่ยวชาญ คือ กลไกอนุมาน (Inference Engine) ซึ่งเป็นกระบวนการในการค้นหาคำตอบภายในองค์ประกอบข้อมูลที่สำคัญ เช่น ฐานความรู้ ประกอบกับองค์ประกอบอื่น ๆ อีกคือ ส่วนอนุมานความรู้ ส่วนอธิบายความ และส่วนติดต่อกับผู้ใช้ โดยใน

กระบวนการพัฒนาสิ่งเหล่านี้สามารถเขียนจากโปรแกรมภาษาใด ๆ ได้ตั้งแต่เริ่มต้น แต่เป็นขั้นตอนที่ยากและเสียเวลา (Jackson, 1999; Salim, Villavicencio, and Timmerman, 2002) ดังนั้นการพัฒนาเปลือกระบบผู้เชี่ยวชาญจึงมีบทบาทสำคัญมากในปัจจุบัน โดยเฉพาะในเชิงพาณิชย์

แนวคิดเกี่ยวกับเปลือกระบบผู้เชี่ยวชาญคล้ายกับโปรแกรมกึ่งสำเร็จรูปพร้อมใช้งาน ขาดเพียงแค่ฐานความรู้ที่เปรียบได้กับฐานข้อมูลในโปรแกรมนั้น การนำเปลือกระบบผู้เชี่ยวชาญที่พัฒนาขึ้นแล้วไปใช้นั้น ผู้ใช้พัฒนาเพิ่มเติมในส่วนของฐานความรู้ขึ้นมาใช้ภายในองค์กรนั้น ๆ ก็เพียงพอแล้ว โดยปกติความรู้ที่อยู่ในฐานความรู้นั้นจะได้จากการพยายามกลั่นกรองความรู้ของผู้เชี่ยวชาญโดยวิศวกรความรู้ ซึ่งเป็นกระบวนการที่ยากและเสียเวลา

ในปีพ.ศ. 2552 ผลงานวิจัยเรื่อง การประมวลผลหลังกระบวนการทำเหมืองข้อมูล (นิตยา เกิดประสพ, 2552) ได้นำเสนอการใช้ประโยชน์จากข้อมูลที่ได้หลังจากการทำเหมืองข้อมูล ซึ่งอยู่ในรูปแบบของความรู้ที่มีลักษณะเป็น กฎถ้า-แล้ว (IF-THEN Rule) สอดคล้องกับลักษณะของฐานความรู้ที่ประกอบไปด้วย ข้อเท็จจริง และ กฎ โดยหลักการแล้วการทำเหมืองข้อมูลเป็นการสร้างรูปแบบความสัมพันธ์ระหว่างข้อมูลเพื่อใช้ในการพยากรณ์กับข้อมูลใหม่ในอนาคต เมื่อนำมาประยุกต์ใช้กับฐานความรู้ในระบบผู้เชี่ยวชาญ ทำให้สามารถลดขั้นตอนในการคัดกรองความรู้จากผู้เชี่ยวชาญได้ ด้วยการนำข้อมูลที่เกิดขึ้นในอดีตมาผ่านกระบวนการทำเหมือง ซึ่งมีความเป็นไปได้ที่อาจจะค้นพบความสัมพันธ์ระหว่างข้อมูลในแง่มุมที่ไม่เคยเห็นมาก่อน เรียกได้ว่าเป็นความก้าวหน้าในการพัฒนารูปแบบในการสร้างฐานความรู้ โดยโครงสร้างของระบบดังแสดงในรูปที่ 1.1



รูปที่ 1.1 กรอบแนวคิดของงานวิจัยเรื่อง การประมวลผลหลังกระบวนการทำเหมืองข้อมูล

จากรูปจะเห็นว่าในส่วนของเปลือกระบบผู้เชี่ยวชาญถูกเขียนขึ้นมาด้วยภาษาโปรล็อก (Prolog) ซึ่งเป็นภาษาโปรแกรมเชิงตรรกะที่นิยมใช้ในงานทางด้านปัญญาประดิษฐ์ และจากการศึกษาวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับและงานวิจัยอื่นที่เกี่ยวข้อง ทำให้ผู้วิจัยมีแนวคิดที่จะปรับเปลี่ยนในส่วนของเปลือกระบบผู้เชี่ยวชาญที่เขียนด้วยภาษาโปรล็อกนี้ เป็นการเขียนด้วยภาษาโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับแทน เนื่องมาจากการศึกษาความแตกต่างระหว่างภาษาโปรล็อก และวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ หรือภาษา CLP (Constraint Logic Programming) นั้นพบว่า ภาษา CLP มีข้อได้เปรียบหลาย ๆ อย่างที่เหนือกว่าและดีกว่าภาษาโปรล็อก เช่น ภาษา CLP มีกระบวนการค้นหาที่มีประสิทธิภาพมากกว่า เมื่อเปรียบเทียบกับ Depth First Search ในภาษาโปรล็อก ภาษา CLP มีกระบวนการ Search ที่หลากหลายและเทคนิค Constraint Propagation ที่ถูกรวมกันไว้ได้อย่างมีประสิทธิภาพ และง่ายต่อการใช้งาน Propagation Solver และ Built-in Predicate ที่ให้บริการแก่ผู้ใช้ของภาษา CLP ก็มีปริมาณมาก และมีประสิทธิภาพมากกว่าภาษาโปรล็อก

ดังนั้นการพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ จึงเป็นแนวทางใหม่ที่น่าสนใจ โดยผู้วิจัยมีความคาดหวังว่า เปลือกระบบผู้เชี่ยวชาญที่ได้รับการพัฒนาขึ้นในงานวิจัยชิ้นนี้ จะสามารถเพิ่มประสิทธิภาพในการทำงานของระบบผู้เชี่ยวชาญได้ ทั้งในเรื่องของเวลาที่ใช้ในการพัฒนาโปรแกรมต้นแบบที่ควรจะลดลงเนื่องจากความยาวของโค้ดโปรแกรมที่สั้นลง และกระบวนการค้นหาคำตอบที่มีประสิทธิภาพที่สูงกว่าแบบเดิมที่ใช้ภาษาโปรล็อก

1.2 วัตถุประสงค์ของการวิจัย

งานวิจัยนี้มีวัตถุประสงค์หลักเพื่อศึกษาและพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ โดยมีจุดประสงค์ย่อยดังนี้

1.2.1 เพื่อศึกษาและพัฒนาเปลือกระบบผู้เชี่ยวชาญที่มีลักษณะยืดหยุ่น สามารถใช้ได้กับฐานความรู้ทั่วไปด้วยการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ

1.2.2 เพื่อเพิ่มประสิทธิภาพในกระบวนการค้นหาความรู้ ลดเวลาที่ใช้ในกลไกอนุมานความรู้ ปรับปรุงโค้ดโปรแกรมให้สั้นลง ลดการใช้พื้นที่หน่วยความจำขณะประมวลผล ด้วยการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ

1.2.3 เพื่อทดสอบเปรียบเทียบประสิทธิภาพและเวลาที่ใช้ของเปลือกระบบผู้เชี่ยวชาญที่พัฒนาด้วยภาษาโปรล็อกกับเปลือกระบบผู้เชี่ยวชาญที่พัฒนาด้วยการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ

1.3 ขอบเขตของการวิจัย

การวิจัยนี้เป็นการศึกษาและพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ เพื่อเพิ่มประสิทธิภาพในด้านของเวลาและปริมาณการใช้พื้นที่หน่วยความจำที่ลดลง ซึ่งเป็นผลจากขนาดข้อมูลระบบที่จะเล็กลงและประสิทธิภาพของกระบวนการค้นหาที่ดีกว่าเดิม โดยเปลือกระบบผู้เชี่ยวชาญที่พัฒนาขึ้นจะมีคุณสมบัติดังต่อไปนี้

1.3.1 สามารถเปลี่ยนฐานความรู้เป็นความรู้ในขอบเขต (Domain) อื่น ๆ ได้ โดยฐานความรู้ที่ใช้ต้องมีรูปแบบที่เข้ากันได้กับกลไกการทำงานของเปลือกระบบผู้เชี่ยวชาญ ซึ่งจะช่วยลดเวลาและค่าใช้จ่ายที่อาจจะต้องสูญเสียไปกับการพัฒนาระบบผู้เชี่ยวชาญขึ้นมาใหม่

1.3.2 ระบบที่พัฒนาขึ้นนี้มีส่วนติดต่อกับผู้ใช้ เพื่อสอบถามข้อมูลไปใช้ในการค้นหาความรู้ภายในกลไกอนุมาน ซึ่งสามารถอนุมานความรู้ได้อย่างเหมาะสมตามข้อมูลที่ผู้ใช้ระบุเข้ามาในระบบ พร้อมทั้งมีการอธิบายความรู้สนับสนุนคำตอบของผู้ใช้ได้อย่างเหมาะสมกับฐานความรู้ขอบเขตใด ๆ

ผลงานที่ได้จากงานวิจัยชิ้นนี้จะถูกเปรียบเทียบประสิทธิภาพในด้านเวลา และพื้นที่หน่วยความจำที่ใช้ขณะประมวลผล กับเปลือกระบบผู้เชี่ยวชาญที่เขียนด้วยภาษาโปรล็อกเพื่อวัดผลสำเร็จของการวิจัย

1.4 ประโยชน์ที่คาดว่าจะได้รับ

งานวิจัยนี้เป็นการศึกษาและพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ โดยคาดว่าจะได้รับประโยชน์ดังต่อไปนี้

1.4.1 ระบบที่พัฒนาขึ้นสามารถลดเวลาที่ใช้ในกระบวนการต่าง ๆ ได้

1.4.2 ระบบที่พัฒนาขึ้นสามารถลดหน่วยความจำที่ใช้ทั้งในด้านการจัดเก็บระบบและการประมวลผลระบบ

1.4.3 ระบบที่พัฒนาขึ้นมีประสิทธิภาพดีกว่าระบบเดิมที่พัฒนาด้วยภาษาโปรล็อก ส่งผลให้ระบบมีการทำงานที่ดียิ่งขึ้น

1.4.4 ระบบที่พัฒนาขึ้นสามารถอนุมานความรู้ได้อย่างรวดเร็วและแม่นยำยิ่งขึ้น

1.4.5 ระบบที่พัฒนาขึ้นสามารถใช้งานได้จริงกับฐานความรู้ใด ๆ

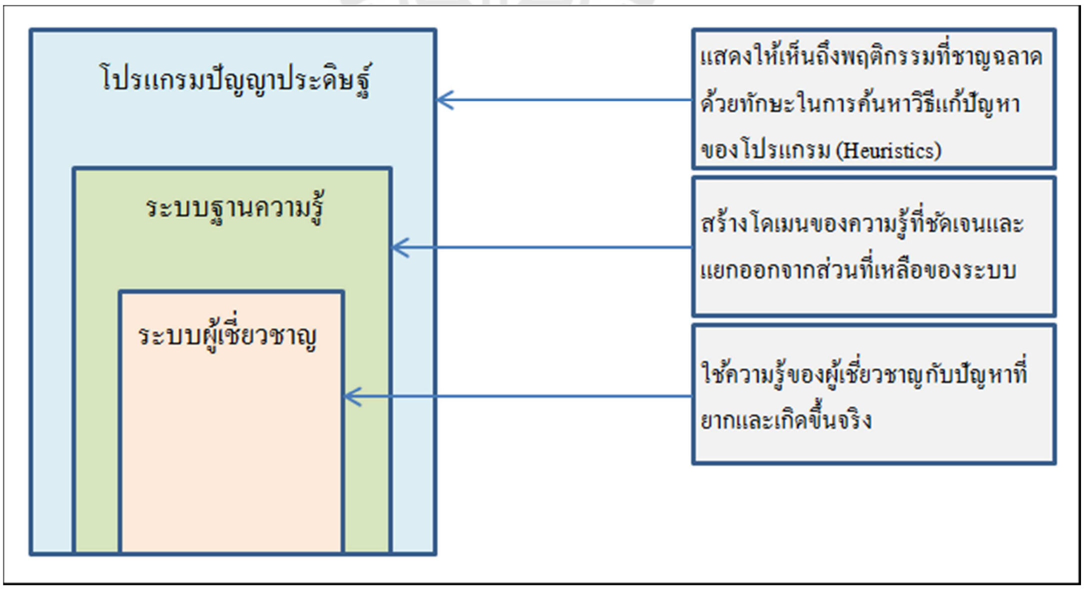
1.4.6 ระบบที่พัฒนาขึ้นสามารถใช้เป็นต้นแบบในการพัฒนาระบบอื่น ๆ ในอนาคต

บทที่ 2
ปริทัศน์วรรณกรรมและงานวิจัยที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึงปริทัศน์วรรณกรรมและงานวิจัยที่เกี่ยวข้อง โดยมีรายละเอียดของระบบผู้เชี่ยวชาญ (Expert System) เปลือกระบบผู้เชี่ยวชาญ (Expert System Shell) การเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ (Constraint Logic Programming) และงานวิจัยที่เกี่ยวข้อง

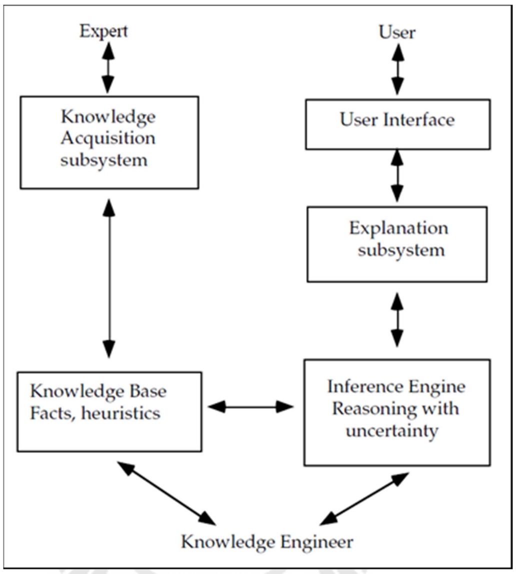
2.1 ระบบผู้เชี่ยวชาญ (Expert System)

โปรแกรมคอมพิวเตอร์ที่ประสบความสำเร็จด้วยเทคโนโลยีของปัญญาประดิษฐ์ (Artificial Intelligence) ส่วนใหญ่จะเป็นโปรแกรมที่เรียกว่า ระบบผู้เชี่ยวชาญ (รูปที่ 2.1) ระบบผู้เชี่ยวชาญเป็นโปรแกรมคอมพิวเตอร์ที่ใช้ความรู้และเหตุผลในการแก้ปัญหาที่ซับซ้อน (Feigenbaum, 1992) ด้วยการจำลองความสามารถในการตัดสินใจของมนุษย์ที่เป็นผู้เชี่ยวชาญมาไว้ในระบบคอมพิวเตอร์เพื่อช่วยในการแก้ปัญหาที่ต้องการคำแนะนำจากผู้เชี่ยวชาญโดยเฉพาะ



รูปที่ 2.1 คุณลักษณะของโปรแกรมปัญญาประดิษฐ์ ระบบฐานความรู้ และระบบผู้เชี่ยวชาญ (Saad, 2008).

ระบบผู้เชี่ยวชาญช่วยในการตัดสินใจได้ด้วยกระบวนการทางคอมพิวเตอร์ที่มีการรวบรวมเหตุผลเชิงตรรกะเข้าด้วยกันและนิยมบันทึกในลักษณะของกฎ (Rules) ระบบผู้เชี่ยวชาญจะเรียกใช้ความรู้เฉพาะด้านหนึ่ง ๆ จากฐานความรู้ (Knowledge Base) โดยขึ้นอยู่กับค่าความจริงของเหตุการณ์ (Facts) ที่ต้องการตัดสินใจ ผ่านกลไกอนุมาน (Inference Engine) มาใช้ในการสรุปข้อมูลและให้เหตุผล เพื่อให้คำแนะนำรวมทั้งอธิบาย (Explanation) คำแนะนำนั้นแก่ผู้ใช้ โดยโครงสร้างของระบบผู้เชี่ยวชาญจะแสดงดังรูปที่ 2.2



รูปที่ 2.2 โครงสร้างพื้นฐานของระบบผู้เชี่ยวชาญ (Feigenbaum, 1992)

คำว่า “ผู้เชี่ยวชาญ” ในระบบผู้เชี่ยวชาญ แสดงถึงความตั้งใจของผู้ริเริ่มออกแบบระบบผู้เชี่ยวชาญ ที่จะพัฒนาระบบให้มีความสามารถในการแก้ปัญหาในบางขอบเขตของงานเทียบเท่ากับประสิทธิภาพในการแก้ปัญหาของมนุษย์ที่เป็นผู้เชี่ยวชาญด้านนั้น ๆ เพื่อให้บรรลุจุดประสงค์นี้ระบบผู้เชี่ยวชาญจะต้องมีความรู้ที่ผู้เชี่ยวชาญในด้านดังกล่าวมีอยู่ ซึ่งการบันทึกความรู้นี้เป็นหน้าที่ของวิศวกรความรู้ (Knowledge Engineer) (Feigenbaum, 1992) เป็นผู้กลั่นกรองความรู้จากผู้เชี่ยวชาญด้านนั้น

ลักษณะปัญหาที่ใช้ระบบผู้เชี่ยวชาญแก้นั้น ส่วนใหญ่จะเป็นปัญหาที่ยุ่งยากและไม่ค่อยมีโครงสร้าง ซึ่งคำตอบที่เป็นไปได้ของปัญหาลักษณะนี้มีหลายคำตอบขึ้นอยู่กับสถานการณ์ในขณะนั้นและข้อมูลที่ได้รับ วิธีการแก้ปัญหาในลักษณะนี้ไม่สามารถกำหนดขั้นตอนแก้ปัญหาอย่างชัดเจนไว้ล่วงหน้าได้ ดังนั้นวิธีแก้ปัญหาในลักษณะอัลกอริทึมที่มีการกำหนดขั้นตอนในการปัญหา

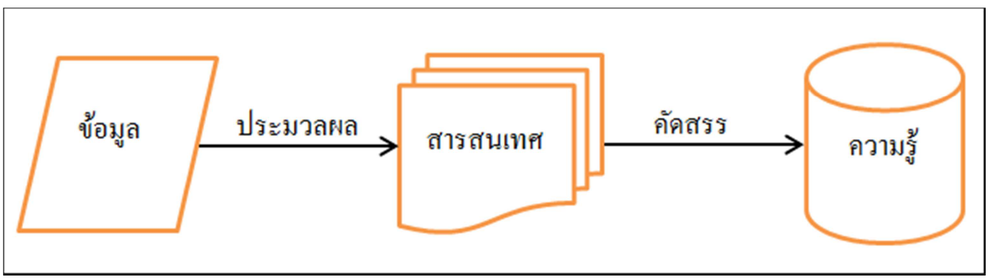
ไว้จึงประยุกต์ใช้กับปัญหาลักษณะนี้ไม่ได้ ระบบผู้เชี่ยวชาญเป็นโปรแกรมคอมพิวเตอร์ที่มีโครงสร้างและเทคนิคในการสร้างต่างจากโปรแกรมอื่น ๆ และเป้าหมายในการประยุกต์ใช้ก็แตกต่างกัน การประยุกต์ใช้ระบบผู้เชี่ยวชาญที่ผ่านมา เช่น การวินิจฉัยโรค การสำรวจทรัพยากรธรณี การวิเคราะห์โครงสร้างสารอินทรีย์เคมี และการแนะนำระบบคอมพิวเตอร์ เป็นต้น (วิลาศ วูวงศ์และบุญเจริญ ศิริเนาวกุล, 2535)

จากข้อมูลเบื้องต้นสามารถสรุปได้ว่า ระบบผู้เชี่ยวชาญ คือ ระบบที่พัฒนาเพื่อเลียนแบบการคิดและให้เหตุผลในการแก้ปัญหาบางอย่างเหมือนมนุษย์ โดยระบบจะมีความรู้ในบางขอบเขตเป็นพิเศษเทียบเท่ากับผู้เชี่ยวชาญที่เป็นมนุษย์ที่มีความรู้ในด้านนั้น ๆ เหมาะใช้ในการแก้ปัญหาที่ไม่ค่อยมีโครงสร้างและมีโอกาสเกิดคำตอบขึ้นหลายคำตอบ ระบบผู้เชี่ยวชาญทุกระบบจะประกอบไปด้วยสองส่วนหลัก ๆ คือ ฐานความรู้และเหตุผลหรือกลไกอนุมาน นอกจากนี้ยังประกอบด้วยส่วนย่อยอื่น ๆ ได้แก่ ส่วนแสวงหาความรู้ ส่วนอธิบายความ และส่วนติดต่อกับผู้ใช้ ดังแสดงในภาพที่ 2.2 (Feigenbaum, 1992)

2.1.1 ฐานความรู้ (Knowledge Base)

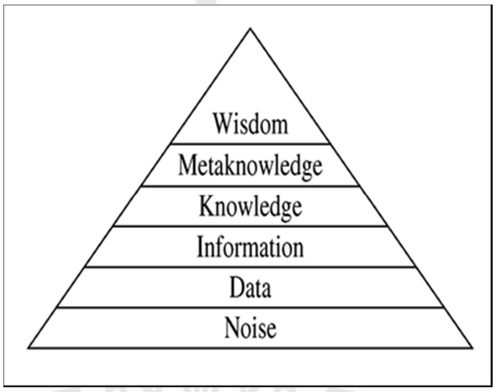
ความรู้ เป็นส่วนสำคัญในการทำงานด้านปัญญาประดิษฐ์ เพราะสามารถนำไปใช้เป็นข้อมูลเพื่อคิดหรือตัดสินใจได้ ซึ่งมีคำสำคัญดังนี้

1. ข้อมูล (Data) เป็นข้อเท็จจริงของเหตุการณ์เกี่ยวกับสิ่งต่าง ๆ และยังไม่ถูกประมวลผล
2. สารสนเทศ (Information) เป็นข้อมูลที่ผ่านการประมวลผลแล้ว และถูกจัดการให้มีความถูกต้องและทันสมัย
3. ความรู้ (Knowledge) เป็นสารสนเทศที่ผ่านการคัดสรร เพื่อนำไปใช้ในการแก้ปัญหาในสถานการณ์ต่าง ๆ ที่มีความต้องการได้อย่างมีประสิทธิภาพ (Choopan Rattanapoka, 2010)



รูปที่ 2.3 แสดงความสัมพันธ์ระหว่างข้อมูล สารสนเทศ และ ความรู้

ความสัมพันธ์ของข้อมูล ความรู้ และสารสนเทศ แสดงได้ดังรูปที่ 2.3 และรูปที่ 2.4 แสดงภาพของปิรามิดความรู้ จะเห็นว่านอกเหนือจากข้อมูล สารสนเทศ และความรู้แล้ว ในระดับล่างสุดมี Noise ซึ่งหมายถึง ข้อมูลที่ไม่ชัดเจนไม่สามารถนำมาใช้ได้ จากนั้นจึงเป็นระดับของข้อมูลที่มีศักยภาพพอที่จะใช้ประโยชน์ได้ ส่วนระดับชั้นของสารสนเทศ หมายถึงข้อมูลที่มีประโยชน์เหมาะที่จะใช้เป็นความรู้ได้ ในระดับชั้นของความรู้นั้น เป็นเหมือนกฎเกณฑ์เกี่ยวกับการใช้สารสนเทศ ในระดับที่สูงขึ้นคือ ความรู้เชิงเมต้า (Metaknowledge) ซึ่งเป็นกฎเกณฑ์เกี่ยวกับความรู้ และในระดับสูงสุด คือ ปัญญา หรือ การตัดสินใจอย่างเฉียบแหลม หมายถึง การใช้ความรู้ในหนทางที่ก่อให้เกิดประโยชน์

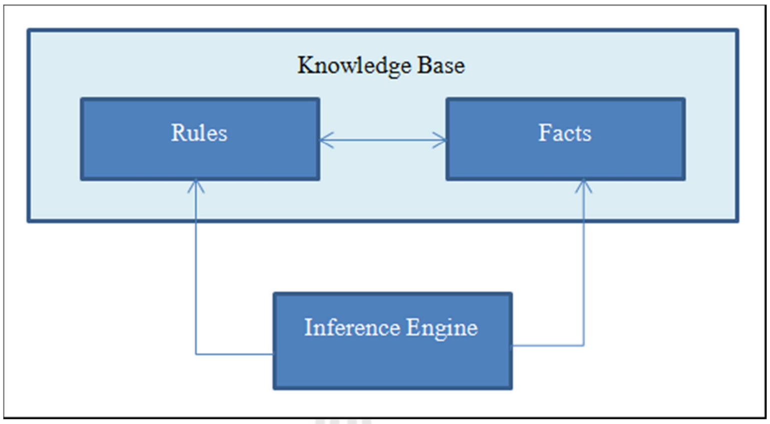


รูปที่ 2.4 ปิรามิดความรู้ (Misiaszek, 2007b)

ฐานความรู้มีทั้งความรู้ที่เป็นข้อเท็จจริง (Factual Knowledge) และ ความรู้ที่ได้จากการเรียนรู้จริง (Heuristic Knowledge) ความรู้ที่เป็นข้อเท็จจริงนั้นจะเป็นความรู้ที่รู้กันอย่างกว้างขวางโดยอยู่ในขอบเขตที่กำหนดและมักจะมีการตกลงกันไว้ระหว่างผู้เชี่ยวชาญ ส่วนความรู้ที่ได้จากการเรียนรู้จริง เช่น ความรู้ที่ไม่มีกฎเกณฑ์บังคับอย่างเข้มงวด ความรู้ที่เกิดจากประสบการณ์ ความรู้เกี่ยวกับการตัดสินใจที่ดี (The Rules-of-thumb) เป็นต้น ดังนั้นความรู้ที่ได้จากการเรียนรู้จริงจะประกอบไปด้วย "ศิลปะในการคาดเดาที่ดี" โดยอยู่ในขอบเขตที่กำหนด (Feigenbaum, 1992)

ในด้านเทคโนโลยีของระบบผู้เชี่ยวชาญ ฐานความรู้จะแสดงด้วยภาษาธรรมชาติ (Natural Language) ในรูปแบบของกฎ IF-THEN ตัวอย่างเช่น "IF it is living THEN it is mortal" หรือ "IF his age = known THEN his date of birth = date of today - his age in years" เป็นต้น รูปแบบนี้มีข้อได้เปรียบเนื่องจากเป็นรูปแบบที่สอดคล้องกับภาษาพูดในชีวิตประจำวัน กฎที่ใช้ในการแสดงความรู้จะถูกเรียกใช้โดยระบบผู้เชี่ยวชาญ (รูปที่ 2.5) โดยรูปแบบของกฎจะถูกปรับให้เหมาะสมกับการออกแบบของระบบแต่ละระบบ ปัญหาสำคัญที่เกิดขึ้นในระหว่างพัฒนาระบบ

ผู้เชี่ยวชาญ คือ ปัญหาในการรวบรวมความรู้ของผู้เชี่ยวชาญซึ่งอาจจะกลั่นกรองออกมาได้ไม่หมด (Wikipedia, 2011f)



รูปที่ 2.5 แสดงความสัมพันธ์ระหว่าง Rules Facts และ Inference Engine

2.1.2 ส่วนแสวงหาความรู้ (Knowledge Acquisition Subsystem)

กระบวนการแสวงหาและจัดการความรู้ในระบบผู้เชี่ยวชาญ มีข้อจำกัดที่สำคัญ คือ การที่วิศวกรความรู้จะได้ความรู้มานั้น ต้องผ่านกระบวนการที่ยากลำบากและเสียเวลา ที่เรียกว่า "การแทนความรู้" (Knowledge Representation) (กิดานันท์ มลิทอง, 2539) และต้องมีการจัดการความรู้เพื่อนำไปใช้กับกลไกอนุมาน (Inference Engine) ในระบบผู้เชี่ยวชาญได้อย่างเหมาะสม (Feigenbaum, 1992)

การแทนความรู้ คือ การนำเสนอความรู้ในรูปแบบที่เครื่องจักรสามารถนำไปใช้งานได้ ภาษาที่ใช้แทนความรู้มีอยู่หลายภาษา ซึ่งลักษณะของภาษาที่ใช้แทนความรู้ที่ดีควรจะมีคุณลักษณะดังนี้

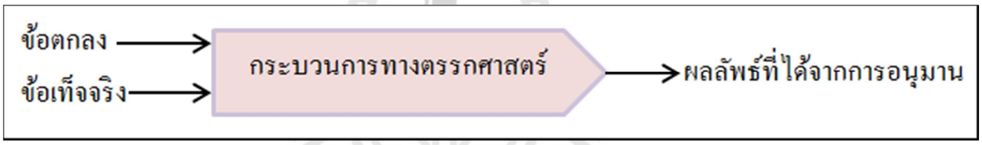
1. สามารถแทนความรู้ที่ซับซ้อนได้ (Expressive)
2. ความรู้ที่ถูกแทนควรอยู่ในรูปแบบที่สั้นกระชับ (Compact)
3. ความรู้สามารถถูกนำไปคิดหาเหตุผลหรืออนุมาน (Inference) เพื่อใช้ในการแก้ปัญหาต่าง ๆ ได้ง่ายและเป็นอัตโนมัติ (Automated Reasoning)
4. สามารถเรียนรู้ความรู้ใหม่ได้ ด้วยเทคนิคการเรียนรู้ของเครื่องจักร ซึ่งอยู่ในรูปแบบภาษาที่มนุษย์สามารถใช้งานได้ง่าย (Automated Learning)

ในปัจจุบันไม่มีภาษาที่ดีที่สุดในการใช้แทนความรู้ แต่มีภาษาที่มี "ศักยภาพในการแทนความรู้" หรือ "ความสามารถในการแสดงความรู้" (Expressiveness) ได้ในระดับที่แตกต่างกัน

กล่าวคือความรู้บางอย่างที่มีความซับซ้อนจะไม่สามารถใช้กับเครื่องจักรที่ใช้ภาษาที่ง่ายเกินไปได้ แต่การแทนความรู้ด้วยภาษาที่ซับซ้อนเกินไปก็จะนำไปใช้งานและเรียนรู้ได้ยาก ดังนั้นผู้พัฒนาจึงควรเลือกด้วยว่า ต้องการนำเสนอความรู้ในรูปแบบที่ซับซ้อนได้ดี หรือต้องการใช้งานความรู้นั้นได้อย่างสะดวก (วิกิพีเดีย สารานุกรมเสรี, 2554ก) การแทนความรู้มีอยู่หลายลักษณะดังนี้

1. การแทนความรู้เชิงตรรกะ (Logical Knowledge Representation) อาศัยแนวคิดเชิงตรรกะสำหรับแสดงความรู้ โดยการนำสารสนเทศที่สร้างขึ้นในรูปแบบของข้อตกลง (Premise) หรือข้อเท็จจริง (Fact) มาเข้าสู่กระบวนการทางตรรกศาสตร์ แล้วได้ผลลัพธ์ที่เป็นผลของการอนุมาน (Inference)

ข้อดีของการแทนความรู้เชิงตรรกะ คือ ค่าที่ใส่ในประพจน์หรือประโยคจะมีความเป็นอิสระต่อกัน มีความแม่นยำ และความสมบรูณ์สูงเมื่อนำไปใช้ในข้อมูลขนาดเล็ก ส่วนข้อเสียของการแทนความรู้เชิงตรรกะ คือ มีการแยกส่วนวิธีการจัดรูปแบบและการประมวลผลออกจากกัน ไม่เหมาะกับข้อมูลขนาดใหญ่ เพราะจะทำให้ทำงานได้ค่อนข้างช้า (Rattanapoka, C., 2010) โครงสร้างของการแทนความรู้เชิงตรรกะแสดงได้ดังรูปที่ 2.6



รูปที่ 2.6 การแทนความรู้เชิงตรรกะ

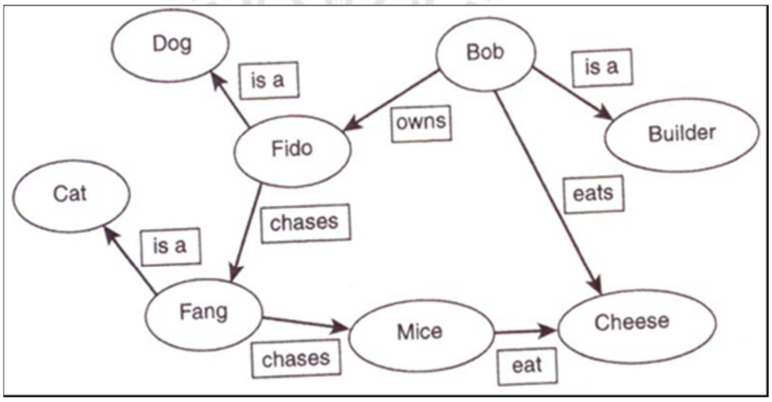
2. การแทนความรู้เชิงระเบียบวิธี (Procedural Knowledge Representation) มีการจัดรูปแบบโดยการนำกลุ่มของคำสั่งหรือเงื่อนไข (Condition) และการกระทำ (Action) มาใช้ร่วมกัน เรียกว่า “กฎ” (Rule) ซึ่งแต่ละกฎจะมีความเป็นอิสระต่อกันทำให้สามารถเพิ่มกฎใหม่ได้เรื่อย ๆ และมีความคล้ายคลึงกับพฤติกรรมในการเรียนรู้ของมนุษย์ รูปแบบการใช้งานจะอยู่ในลักษณะของกฎ “ถ้า-แล้ว” (IF-THEN Rule) ซึ่งในกรณีที่กฎมีความซับซ้อนจะมีตัวกระทำที่เป็น “และ” (AND) “หรือ” (OR) เข้ามาร่วมด้วย

ข้อดีของการแทนความรู้เชิงระเบียบวิธี คือ ง่ายต่อการทำความเข้าใจและการสื่อสาร สามารถทำการอนุมานและอธิบายได้ง่าย สะดวกในการบำรุงรักษาคุณสมบัติของความสัมพันธ์ สามารถผสมผสานกฎต่าง ๆ เข้าด้วยกันได้ และกฎแต่ละข้อมีความเป็นอิสระต่อกัน ส่วนข้อเสียในการแทนความรู้เชิงระเบียบวิธี คือ ความรู้ที่ได้มีความซับซ้อนและยากต่อการใช้งาน

ในบางครั้งกฎที่ได้มาไม่สมเหตุสมผล และระบบที่มีการใช้กฎเป็นจำนวนมากจะยากต่อการค้นหาความรู้ (Choopan Rattanapoka, 2010)

3. การแทนความรู้เชิงเครือข่าย (Network Knowledge Representation) เป็นการแทนความรู้ด้วยโครงสร้างของกราฟ ประกอบด้วย โหนด (Node) และกิ่ง (Branch, Link or Arc) โดยโหนดจะแทนวัตถุซึ่งแบ่งได้ 2 ประเภท คือ วัตถุที่จับต้องได้ เช่น หนังสือ รถยนต์ เป็นต้น และวัตถุที่จับต้องไม่ได้ เช่น แนวคิด การกระทำ หรือเหตุการณ์ นอกจากนี้แล้วยังสามารถแทนรายละเอียด หรือคุณลักษณะของวัตถุ เช่น สีหรือขนาดได้ ส่วนกิ่งจะมีหน้าที่ในการเชื่อมโยงความสัมพันธ์ระหว่างโหนดต่าง ๆ เข้าด้วยกัน โดยใช้คำหรือวลีที่สื่อความหมายในการกำกับกิ่งนั้น เช่นคำว่า กิน เล่น มี คือ เป็นต้น ตัวอย่างของการแทนความรู้เชิงเครือข่ายแสดงได้ดังรูปที่ 2.7

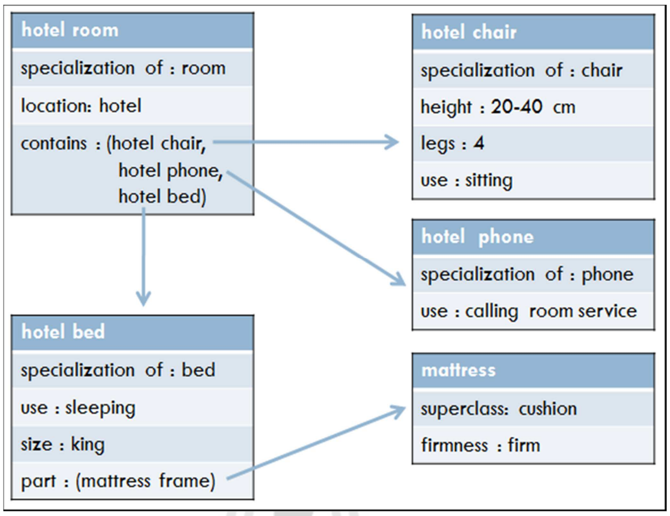
ข้อดีของการแทนความรู้เชิงเครือข่าย คือ ง่ายต่อการติดตามโครงสร้างเป็นระดับชั้น สามารถติดตามการเชื่อมโยงได้ง่าย มีความยืดหยุ่นในเรื่องของขนาดเครือข่าย และมีการถ่ายทอดคุณลักษณะภายในเครือข่าย ส่วนข้อเสียในการแทนความรู้เชิงเครือข่าย คือ ความหมายของแต่ละโหนดอาจเบี่ยงเบนได้ เนื่องจากคำแต่ละคำอาจจะมีหลายความหมาย ยากต่อการตรวจจับข้อผิดพลาด นำมาเขียนโปรแกรมได้ยาก และขนาดของหน่วยความจำที่ใช้อาจไม่เพียงพอ (Choopan Rattanapoka, 2010)



รูปที่ 2.7 ตัวอย่างการแทนความรู้เชิงเครือข่าย (Misiaszek, 2007a)

4. การแทนความรู้เชิงโครงสร้าง (Structured Knowledge Representation) เป็นการแทนโดยการอธิบายความรู้ในลักษณะของโครงสร้าง สามารถใช้ควบคู่กับการแทนความรู้เชิงเครือข่ายได้ เมื่อโหนดแต่ละโหนดนั้นมีความซ้ำซ้อน ซึ่งรูปแบบของการแทนความรู้เชิงโครงสร้างที่เป็นที่นิยม คือ เฟรม (Frame) ดังตัวอย่างในรูปที่ 2.8

เฟรม คือ กรอบที่ใช้กำหนดขอบเขตของความรู้เชิงวัตถุที่มีความซับซ้อน เพื่อนำไปใช้แก้ปัญหาใดปัญหาหนึ่ง โดยเฟรมจะถูกแบ่งออกเป็นส่วนย่อย ๆ ที่เรียกว่า “Slot” โดย Slot จะมีหน้าที่แสดงคุณลักษณะและค่าของ Slot ที่เรียกว่า “Facet” ไว้



รูปที่ 2.8 ตัวอย่างการแทนความรู้เชิงโครงสร้าง (Choopan Rattanapoka, 2010)

ข้อดีของการแทนความรู้เชิงโครงสร้าง คือ Slot สามารถสร้างเพิ่มหรือลดลงได้ง่าย สามารถกำหนดค่า Default ได้ สามารถนำเฟรมไปประยุกต์ใช้ร่วมกับเครือข่ายเชิงความหมายได้ และมีคุณลักษณะในการถ่ายทอด ส่วนข้อเสียของการแทนความรู้เชิงโครงสร้าง คือ เขียนโปรแกรมได้ยาก ยากต่อการสรุปความ และเครื่องมือที่ใช้ในการพัฒนามีน้อย (Choopan Rattanapoka, 2010)

5. การแทนความรู้เชิงผสมผสาน (Multiple Knowledge Representation) ในการแทนความรู้นั้นหากใช้เพียงรูปแบบเดียวอาจจะไม่เพียงพอต่อการนำไปใช้งานกับงานทุกประเภทได้ เนื่องจากการแทนความรู้แต่ละรูปแบบมีข้อดีและข้อเสียแตกต่างกัน ซึ่งการแทนความรู้เชิงผสมผสาน หมายถึง การนำข้อดีของแต่ละวิธีมาใช้ร่วมกัน (Choopan Rattanapoka, 2010)

2.1.3 กลไกอนุมาน (Inference Engine)

กลไกอนุมานจัดว่าเป็นกลไกที่ใช้ค้นหาความรู้ในฐานความรู้ มีความสามารถในการเปรียบเทียบรูปแบบ (Pattern Matching) การสำรวจ (Investigate) การขจัด (Eliminate) การจับคู่กฎ (Matching) เพื่อหาผลลัพธ์ที่ถูกต้อง ดังนั้นในระบบผู้เชี่ยวชาญจึงใช้กลไกอนุมานในการค้นหาความรู้จากฐานความรู้โดยตรง (No-poor, 2010)

ในระบบผู้เชี่ยวชาญ ทุกกลไกอนุมานจะใช้ความรู้ในฐานความรู้ในการสร้างเส้นทางของเหตุผล ที่จะนำไปสู่การแก้ปัญหานั้น ๆ ให้ได้มากที่สุด ซึ่งวิธีการโดยทั่วไปจะเกี่ยวข้องกับการค้นหาแบบลูกโซ่ (Chaining) หรือที่เรียกอีกอย่างว่าการให้เหตุผล (Reasoning) ภายใต้กฎที่อยู่ในรูปแบบของ IF-THEN ในการเริ่มการค้นหาด้วยชุดเงื่อนไขต่าง ๆ เพื่อไปสู่ข้อสรุปบางอย่าง จะมีวิธีการที่เรียกว่า “การค้นหาแบบไปข้างหน้า” (Forward Chaining) หรือ “การค้นหาแบบย้อนหลัง” (Backward Chaining) (Feigenbaum, 1992) ซึ่งการค้นหาแบบไปข้างหน้า เป็นการค้นหาในฐานความรู้กับปัญหาเฉพาะหน้า โดยกลไกการทำงานจะพยายามหาคำตอบเอง ส่วนการค้นหาแบบย้อนหลัง เป็นการสมมติเป้าหมายขึ้นมา จากนั้นจึงทำการตรวจสอบข้อเท็จจริงและกฎเกณฑ์ต่าง ๆ ในฐานความรู้ โดยระบบจะพยายามตั้งคำถามที่จำเป็นเพื่อที่จะยืนยัน หรือพิสูจน์เพื่อหาคำตอบ กลไกในการอนุมานส่วนใหญ่จะได้แนวคิดมาจากการให้เหตุผลของมนุษย์ เนื่องจากศาสตร์ทางด้านปัญญาประดิษฐ์เป็นการศึกษาโดยต้องการเลียนแบบพฤติกรรม ความนึกคิด และการให้เหตุผลของมนุษย์ (อรอนงค์ อร วันปู, 2554)

การเลือกใช้กระบวนการค้นหาหรือการให้เหตุผลแบบไปข้างหน้า หรือย้อนกลับนั้นขึ้นอยู่กับคุณสมบัติของกฎและข้อเท็จจริงเริ่มต้นที่กำหนดไว้ การค้นหาแบบไปข้างหน้าเหมาะกับการพิสูจน์ข้อเท็จจริงเริ่มต้นขนาดเล็ก ๆ ที่มีแนวโน้มในการเลือกใช้กฎแตกต่างกันเป็นจำนวนมาก แต่สามารถให้ข้อสรุปเดียวกันได้ และกฎจำนวนมากนั้นมีสิทธิ์ถูกตัดทิ้งได้ในการค้นหาคำตอบแต่ละรอบ (Anwar, 2011) ตัวอย่างภาษาเขียนโปรแกรมที่มีกระบวนการค้นหาแบบไปข้างหน้า คือ CLIPS (C Language Integrated Production System) (Wikipedia, 2011g) ซึ่งมีการพัฒนาขึ้นครั้งแรกในปีค.ศ. 1985 ที่ NASA-Johnson Space Center ซึ่งเป็นศูนย์ฝึก ค้นคว้า และควบคุมการบินของนักบินอวกาศ จนกระทั่งในปีค.ศ. 1990 ก็มีการนำไปใช้ในเทคโนโลยีระบบผู้เชี่ยวชาญ (Wikipedia, 2011b)

การค้นหาแบบย้อนกลับควรเลือกใช้เมื่อต้องการพิสูจน์ข้อเท็จจริงเพียงอย่างเดียวจากชุดข้อเท็จจริงเริ่มต้นขนาดใหญ่ ในกรณีนี้จะมีการกำหนดเป้าหมายที่ต้องการพิสูจน์ไว้ก่อน ถ้าหากสามารถหลีกเลี่ยงข้อสรุปที่ไม่เกี่ยวข้องได้จะทำให้การค้นหาแบบย้อนกลับนั้นมีประสิทธิภาพยิ่งขึ้น แต่ในกรณีที่การพิสูจน์นั้นจะต้องทำการค้นหาจากข้อมูลทั้งหมด อันเนื่องมากจากวิธีการพิสูจน์สิ่งที่ตั้งไว้นั้นมีจำนวนมาก จะส่งผลเสียทำให้สิ้นเปลืองทรัพยากรได้ (Anwar, 2011) ตัวอย่างภาษาเขียนโปรแกรมที่มีกระบวนการค้นหาแบบย้อนกลับ เช่น Prolog, Knowledge Machine และ ECLiPSe (Wikipedia, 2011a) ซึ่งรายละเอียดเกี่ยวกับภาษาโปรแกรมที่มีกลไกอนุมานแบบย้อนกลับนี้จะกล่าวถึงในหัวข้อถัดไป

2.1.4 ส่วนอธิบายความ (Explanation Subsystem)

ส่วนอธิบายมีหน้าที่อธิบายรายละเอียดขั้นตอนของการแก้ปัญหาต่อผู้ใช้ระบบว่าข้อสรุปหรือคำตอบที่ได้นั้น ได้มาอย่างไรและทำไม (วิลาศ วูวงศ์และบุญเจริญ ศิริเนาวกุล, 2535) ซึ่งวัตถุประสงค์ของการอธิบายความมีดังต่อไปนี้

1. เพื่อสร้างระบบให้มีความฉลาด
2. เพื่อแสดงกฎและฐานความรู้ที่อาจผิดพลาด ซึ่งจำเป็นต้องแก้ไข
3. เพื่ออธิบายเหตุการณ์ที่คาดหวังแก่ผู้ใช้ระบบ
4. เพื่อสร้างความพอใจและความมั่นใจในระบบแก่ผู้ใช้ระบบ
5. เพื่อหักล้างสมมติฐานและแสดงความชัดเจนแก่ผู้ใช้และผู้สร้าง
6. เพื่อวิเคราะห์ความอ่อนไหวของข้อมูล สำหรับพยากรณ์และทดสอบผลกระทบต่อการเปลี่ยนแปลงของระบบ

การอธิบายความนั้นมีความยุ่งยากซับซ้อน ดังนั้นเพื่อให้การอธิบายนั้นมีประสิทธิภาพจึงจำเป็นต้องเตรียมคำอธิบายสำหรับไขข้อข้องใจสำหรับปัญหาแต่ละปัญหาที่ขึ้นต้นด้วยคำว่า “ทำไม” และ “อย่างไร” ซึ่งสองคำถามนี้เป็นคำถามที่ต้องการผู้อธิบาย เช่น “ทำไมถึงต้องการข้อเท็จจริงนี้” และ “ข้อสรุปและข้อแนะนำนี้มีที่มาอย่างไร” เป็นต้น

การสร้างวิธีอธิบายความมีอยู่ 2 รูปแบบด้วยกัน รูปแบบแรก คือ แบบคงที่ (Static Explanation) เป็นการเขียนคำอธิบายเตรียมไว้ล่วงหน้าแล้วแทรกเข้าไปในระบบ ซึ่งระบบจะต้องแสดงคำอธิบายที่เกี่ยวข้องกับคำถามของผู้ใช้ได้อย่างเหมาะสม รูปแบบที่สอง คือ แบบพลวัต (Dynamic Explanation) เป็นการอธิบายโดยอาศัยจากการเปรียบเทียบรูปแบบของกฎ ด้วยการให้เหตุผล

การอธิบายความในระบบผู้เชี่ยวชาญสามารถแบ่งออกเป็น 3 ประเภทด้วยกัน ประเภทแรก คือ การอธิบายความด้วยการติดตามร่องรอยหรือไล่เรียง (Trace or Line of Reasoning) โดยการอ้างถึงความสัมพันธ์ที่เกี่ยวข้องกันของกฎตามลำดับ แบบเป็นขั้นตอนก่อนที่จะได้ข้อสรุปในจุดสุดท้าย ประเภทที่สอง คือ การอธิบายความด้วยการให้เหตุผล (Justification Reasoning) ซึ่งวิธีนี้จะมีการอธิบายและชี้แจงอย่างชัดเจนโดยปราศจากข้อโต้แย้ง หรือเป็นการให้เหตุผลได้กับทุกกรณีที่มีการกล่าวอ้างถึง เพื่อหักล้างข้อสมมติฐานต่าง ๆ วิธีนี้จะให้ผลลัพธ์ที่ดูสมเหตุสมผล และสร้างความเชื่อมั่นแก่ผู้ใช้ ประเภทสุดท้าย คือ การอธิบายความเชิงกลยุทธ์ (Strategy Reasoning) เป็นการอธิบายด้วยการใช้โครงสร้างแบบลดหลั่น โดยมีเป้าหมายที่จุดสูงสุด แล้วใช้การพิจารณาว่าจะทำงานอย่างไรให้บรรลุเป้าหมายที่กำหนดไว้ วิธีนี้จะเกี่ยวข้องกับการแทนความรู้เชิงระเบียบวิธีและค่อนข้างเป็นไปได้ยากที่จะนำมาใช้กับระบบผู้เชี่ยวชาญ (No-poor, 2010)

2.1.5 ส่วนติดต่อกับผู้ใช้ระบบ (User Interface)

กลุ่มผู้ใช้ระบบผู้เชี่ยวชาญนั้นมีความรู้เกี่ยวกับการใช้สารสนเทศแตกต่างกัน รวมถึงผู้ใช้อาจไม่คุ้นเคยกับการรับคำแนะนำจากระบบสารสนเทศ ตลอดจนความต้องการที่หลากหลายของผู้ใช้ระบบ ทำให้การพัฒนาระบบผู้เชี่ยวชาญจะต้องคำนึงถึงความสะดวกในการโต้ตอบกันระหว่างผู้ใช้ระบบและระบบผู้เชี่ยวชาญ เพื่อให้เกิดความพึงพอใจแก่ผู้ใช้ และสามารถใช้ระบบได้อย่างชำนาญ ซึ่งจะทำให้การทำงานของผู้ใช้ระบบมีประสิทธิภาพดียิ่งขึ้น (วิภาดา เวทย์ประสิทธิ์, 2547)

จากข้อมูลเบื้องต้นเกี่ยวกับส่วนประกอบของระบบผู้เชี่ยวชาญสามารถสรุปได้ว่า ส่วนประกอบของระบบผู้เชี่ยวชาญประกอบไปด้วย 2 ส่วนหลักที่สำคัญ คือ ฐานความรู้และกลไกอนุมาน ฐานความรู้ เป็นเหมือนฐานข้อมูลที่เก็บไว้พร้อม รอให้ผู้ใช้มาค้นหา แต่ว่าในระบบผู้เชี่ยวชาญจะเป็นหน้าที่ของกลไกอนุมานในการคิด ค้นหาคำตอบ หรือเหตุผลภายในฐานความรู้ นอกจากสองส่วนนี้แล้วมีส่วนแสวงหาความรู้ ส่วนอธิบาย และส่วนติดต่อกับผู้ใช้ที่ทำให้ระบบผู้เชี่ยวชาญดูน่าเชื่อถือ (ส่วนอธิบาย) และเป็นมิตรกับผู้ใช้ (ส่วนติดต่อกับผู้ใช้)

2.1.6 ตัวอย่างของระบบผู้เชี่ยวชาญ

ในปีค.ศ. 1960 งานทางด้านปัญญาประดิษฐ์ได้เปลี่ยนความสนใจจากประสิทธิภาพในการทำงานทั่ว ๆ ไป เป็นการหาวิธีการสร้างกลไกสำหรับแก้ปัญหาที่สามารถแก้ปัญหาที่หลากหลายของปัญหาที่เกิดขึ้น ในความพยายามของการศึกษาทางปัญญาประดิษฐ์นี้มีตัวอย่างที่รู้จักกันดีของโปรแกรมแก้ปัญหาทั่วไปทั้งโปรแกรม Heuristic และ Theorem Provers ซึ่งโปรแกรมเหล่านี้มีความสามารถในการแก้ปัญหาในเรื่องง่าย ๆ เช่น ปัญหาความสูงของเกมตัวต่อและการถนอมพลังงานแบตเตอร์รี่ (Feigenbaum, 1992)

1. DENDRAL

ในปีค.ศ. 1965 Feigenbaum, Lederberg และ Buchanan (Feigenbaum, 1992) นักวิจัยของมหาวิทยาลัยสแตนฟอร์ด ริเริ่มโครงการในการให้เหตุผลทางวิทยาศาสตร์ ด้วยการสร้างแบบจำลองที่มีประสิทธิภาพสูง ทำให้งานทางด้านนี้ได้รับความสนใจอีกครั้ง โดยผลงานที่ได้จากโครงการนี้ คือ โปรแกรมที่เรียกว่า DENDRAL เป็นโปรแกรมแปลความหมายมวลสารสเปกตรัมของโมเลกุลอินทรีย์ในแง่ของสมมติฐานของโครงสร้างโมเลกุลอินทรีย์ มีผลงานตีพิมพ์เผยแพร่เครื่องมือชิ้นนี้ในชื่อ Applications of Artificial Intelligence for Organic Chemistry: The DENDRAL Project เมื่อปีค.ศ. 1980 ระดับความยากของงานชิ้นนี้คือความสามารถระดับผู้เชี่ยวชาญจากผลการทดลองทำให้ DENDRAL เป็นที่รู้จักภายใต้สมมติฐานที่ว่า “ความรู้คืออำนาจ” และในภายหลังเรียกว่าหลักการของความรู้ ในช่วงเวลาสั้น ๆ โปรแกรม DENDRAL คือ ต้นกำเนิดของ

โปรแกรมระบบผู้เชี่ยวชาญ และการพัฒนาของ DENDRAL ได้แสดงหลักสำคัญของปัญหาและข้อจำกัดของระบบผู้เชี่ยวชาญ (Lindsay et al., 1980)

2. MYCIN

MYCIN เป็นระบบผู้เชี่ยวชาญที่มีชื่อเสียงอย่างรวดเร็วและรู้จักกันมากที่สุด ระบบนี้แสดงให้เห็นถึงสถาปัตยกรรมแบบบูรณาการครั้งแรก ด้วยการให้คำปรึกษาแบบโต้ตอบกันระหว่างผู้ใช้ระบบกับระบบผู้เชี่ยวชาญ รวมทั้งมีการอธิบายเหตุผลเป็นบรรทัด ๆ เป็นครั้งแรกที่มีการนำระบบผู้เชี่ยวชาญไปใช้ในการวินิจฉัยทางการแพทย์ (Buchanan and Shortliffe, 1984) และเป็นระบบต้นแบบของเปลือกระบบผู้เชี่ยวชาญ (Expert System Shell) การเขียนโปรแกรมเปลือกระบบผู้เชี่ยวชาญได้แฝงขั้นตอนการให้เหตุผลและกรอบสำหรับการแทนความรู้ได้โดยทั่วไป เพื่อที่จะสามารถเปลี่ยนโดเมนของความรู้ และพัฒนาหรือสร้างระบบใหม่ได้ง่าย และมีต้นแบบที่เป็นเปลือกระบบผู้เชี่ยวชาญชื่อ EMYCIN ที่นำมาใช้ในเชิงพาณิชย์ (Melle et al., 1984)

3. XCON

XCON เป็นหนึ่งในสองของระบบผู้เชี่ยวชาญที่ดีที่สุดในช่วงปีค.ศ. 1991 ที่มีการประยุกต์ใช้ระบบผู้เชี่ยวชาญในระบบอุตสาหกรรม หน้าที่ของ XCON คือการกำหนดค่าภายใต้ข้อจำกัดของส่วนประกอบย่อย ๆ จำนวนมากของคอมพิวเตอร์ขนาดเล็กของ DEC การกำหนดค่าด้วย XCON สามารถทำได้อย่างรวดเร็วและถูกต้อง ทำให้ในด้านการผลิตและการดำเนินการขายของ DEC สามารถประหยัดได้ถึงล้านดอลลาร์ต่อปี XCON ทำการกำหนดค่าของคอมพิวเตอร์ขนาดเล็กจำนวนประมาณ 300 ครั้งต่อนาทีเร็วกว่าความเร็วของวิศวกรที่เป็นมนุษย์และแทบจะไม่มีข้อผิดพลาด (Feigenbaum, McCorduck, and Nii, 1988)

XCON ได้รับการขยายขอบเขตสู่งานที่กว้างและใช้ได้ทั่วไปขึ้น กลายเป็นระบบผู้เชี่ยวชาญที่เรียกว่า XSEL เพื่อช่วยให้พนักงานขายของ DEC สามารถกำหนดค่าและราคาขายของอุปกรณ์ในเวลาที่ติดต่อกับลูกค้าได้ทันที (Feigenbaum, 1992)

4. Dipmeter Advisor (DA)

DA เป็นอีกหนึ่งในสองของระบบผู้เชี่ยวชาญที่ดีที่สุดในช่วงปีค.ศ. 1991 คิดค้นโดย Schlumberger โปรแกรม DA ทำการวิเคราะห์การไหลของข้อมูลและนำเสนอสมมติฐานเกี่ยวกับบัญชีและการอธิบายข้อมูล DA ตีความข้อมูลที่เข้ามาจากเครื่องมือเจาะหลุมสำหรับค้นหาน้ำมันและก๊าซ ซึ่งมีสมมติฐานเกี่ยวกับความลาดเอียงหรือที่เรียกว่า Dip ของชั้นหินภายใต้พื้นผิวโลก การเรียนรู้ความลาดเอียงในทุก ๆ ร้อยชั้นของหินภายในหลุมเจาะนั้นมีประโยชน์อย่างมากในการสำรวจหาน้ำมัน (Buchanan and Smith, 1989)

5. Charley

Charley เป็น ตัวอย่างของโปรแกรมตีความสัญญาณข้อมูลเพื่อสร้างสมมติฐานของ General Motor มีไว้สำหรับวิเคราะห์การสั่นสะเทือนของเครื่องจักรกล ซึ่งเป็นวิธีที่ใช้ในการแก้ปัญหาของเครื่องจักรกล โดยเฉพาะอย่างยิ่งในอุตสาหกรรมยานยนต์ และการผลิตอุปกรณ์ (Bajpai and Marczewski, 1989)

6. Lend Lease

อีกระบบผู้เชี่ยวชาญที่มีแรงจูงใจคล้ายกับ Charley ซึ่งสร้างโดย DEC สำหรับบริษัท Lend Lease บริษัทก่อสร้างที่ใหญ่ที่สุดในออสเตรเลีย เพื่อใช้ในการประมาณเวลาในการก่อสร้างอาคารสูงจนเสร็จสิ้น โดยมีความถูกต้องอยู่ที่ ±10% ซึ่งใช้เวลาคำนวณเพียงไม่กี่ชั่วโมงในระหว่างทำการหารือกับลูกค้า (Feigenbaum et al., 1988)

7. NKK Steel Co.

ในทำนองเดียวกันบริษัท NKK Steel จำกัดในประเทศญี่ปุ่น ได้ใช้รูปแบบความเชี่ยวชาญด้านการทำงานกับเตาเผาของผู้เชี่ยวชาญระดับอาวุโส สร้างระบบผู้เชี่ยวชาญที่คาดการณ์ความน่าจะเป็นของความแตกต่างกันของความล้มเหลว และหายนะของการทำงานของเตาเผาในช่วงเวลาสั้น ๆ ที่มีการรับส่งข้อมูลระหว่างกันของคอมพิวเตอร์ (Real-time) ซึ่งระบบผู้เชี่ยวชาญนี้ถูกสร้างเพื่อขายให้กับผู้ซื้อเตาเผาระเบิดของ NKK (Feigenbaum et al., 1988)

8. Authorizers' Assistant (AA)

AA คือ ระบบผู้เชี่ยวชาญของ American Express เป็นตัวอย่างของการใช้ระบบผู้เชี่ยวชาญเพื่อปรับปรุงคุณภาพของการตัดสินใจของมนุษย์ ใช้ในการมอบอำนาจของมนุษย์ในการตัดสินใจที่จะอนุญาตหรือไม่อนุญาตให้ลูกค้าชำระเงินค่าธรรมเนียม AA ทำการวิเคราะห์ข้อมูลจำนวนมากจากแฟ้มฐานข้อมูลของลูกค้าที่ AmEx เก็บไว้ให้แล้ว ปัญหาที่ AA ทำคือการแนะนำการตัดสินใจ และมีคำอธิบายเหตุผลสำหรับข้อเสนอแนะ พร้อมมอบอำนาจให้กับมนุษย์ที่มีข้อมูลที่สนับสนุนข้อเสนอแนะนั้น ซึ่งได้ผลตอบแทนในแง่ของการหลีกเลี่ยงหนี้เสียที่มาจากการทุจริต และสร้างเงินได้เป็นจำนวนถึงล้านดอลลาร์ต่อปี (Feigenbaum et al., 1988)

9. ระบบการดำเนินการสำหรับการดูแลรักษาความปลอดภัยในสังคมอังกฤษ

เปลือกระบบผู้เชี่ยวชาญถูกสร้างขึ้นเพื่อช่วยในการประมวลผลที่ถูกต้องและทันเวลา ในกรณีของระบบที่ซับซ้อนมาก เช่น กฎหมายราชการ กฎเกณฑ์ และกฎระเบียบ เป็นต้น ระบบนี้ช่วยเสมียนในการตอบแบบสอบถามเป็นลายลักษณ์อักษรจากประชาชนเกี่ยวกับบำนาญของพวกเขา (Feigenbaum et al., 1988)

10. Taxpayers' Assistant Expert System (Internal Revenue Service) ในประเทศสหรัฐอเมริกา

หน่วยบริการรายได้จะช่วยให้บุคลากรของ IRS สามารถให้ข้อมูลภาษีที่ถูกต้องผ่านทางโรศัพท์ เพื่อตอบสนองต่อความต้องการของผู้เสียภาษีโทรศัพท์ ซึ่งใน Fresno และ Tulare ของรัฐแคลิฟอร์เนียเปลือกระบบผู้เชี่ยวชาญช่วยเหลือในกรณีการตัดสินใจเกี่ยวกับสวัสดิการสังคม รวมถึงผู้ขอสวัสดิการที่มีคุณสมบัติภายใต้กฎระเบียบที่ซับซ้อน (Feigenbaum, 1992)

จากตัวอย่างเบื้องต้นผู้สามารถสรุปได้ว่า ตัวอย่างของระบบผู้เชี่ยวชาญตั้งแต่อดีตถึงปัจจุบันมักจะเกี่ยวข้องกับการพยายามแก้ปัญหาบางอย่างของมนุษย์ ซึ่งในมนุษย์ทั่วไปอาจจะไม่มีความรู้ความชำนาญที่มากพอในการแก้ปัญหานั้น จึงต้องพึ่งผู้เชี่ยวชาญในด้านนั้นในการให้คำแนะนำในการตัดสินใจหรือแก้ปัญหาต่าง ๆ ระบบผู้เชี่ยวชาญเรียกได้ว่าเป็นระบบที่พัฒนาขึ้นเพื่อมาทำงานแทนมนุษย์ที่เป็นผู้เชี่ยวชาญเฉพาะด้านที่มีอยู่เป็นจำนวนน้อย

2.2 เปลือกระบบผู้เชี่ยวชาญ (Expert System Shell)

ระบบผู้เชี่ยวชาญเป็นโปรแกรมคอมพิวเตอร์ที่มีการทำงานในรูปแบบของตรรกศาสตร์คล้ายกับกระบวนการคิดโดยผู้เชี่ยวชาญที่เป็นมนุษย์ การสร้างระบบผู้เชี่ยวชาญสามารถทำได้ตั้งแต่เริ่มต้น โดยการใช้การเขียนโปรแกรมด้วยภาษามาตรฐานเช่น C/C++, COBOL หรือภาษาทั่วๆไป ซึ่งเป็นกระบวนการที่ยากลำบากและเสียเวลา เหตุผลเนื่องมาจากความซับซ้อนของโปรแกรมระบบ (Jackson, 1999; Salim et al., 2002)

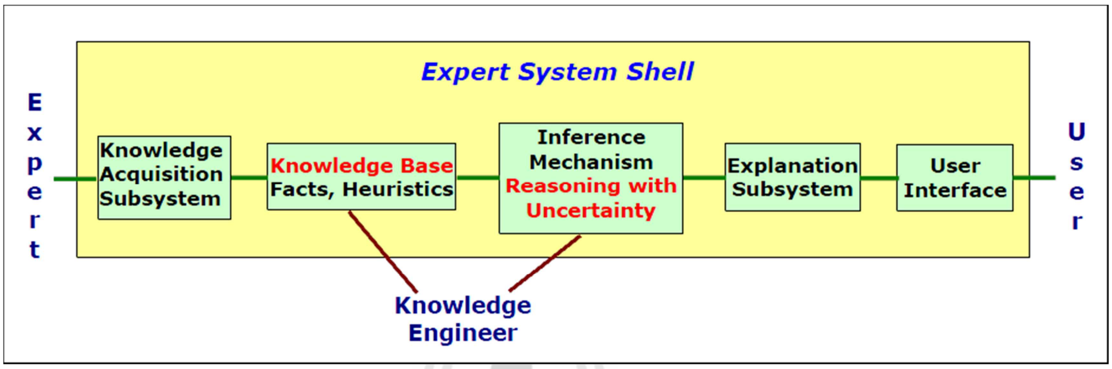
เปลือกระบบผู้เชี่ยวชาญเป็นโปรแกรมเชิงพาณิชย์ที่ใช้ในธุรกิจอุตสาหกรรม รัฐบาล และในหลักสูตรการศึกษา โดยเปลือกระบบผู้เชี่ยวชาญมีความสำคัญในการลดเวลาและแรงงานที่ใช้ในการสร้างระบบผู้เชี่ยวชาญโดยไม่จำเป็นต้องสร้างระบบใหม่ทั้งหมด

ตัวอย่างที่พบเห็นโดยทั่วไปของโปรแกรมระบบผู้เชี่ยวชาญจะเป็นระบบที่สนับสนุนการตัดสินใจ ระบบสนับสนุนการตัดสินใจเป็นโปรแกรมที่ให้ข้อมูลในการตัดสินใจโดยอยู่บนพื้นฐานการให้เหตุผลเหมือนมนุษย์ในรูปแบบของข้อมูลต่าง ๆ การประยุกต์ใช้ระบบตัดสินใจที่พบบ่อยคือ “การระบุความผิดพลาด” โปรแกรมจะแนะนำผู้ใช้ที่ไม่มีความรู้ในเรื่องนั้นด้วยการระบุสาเหตุของความผิดปกติของอุปกรณ์ โดยการแจ้งให้ทราบเพื่อทำการวัดและสังเกตหาความผิดปกติ

การนำปัญญาประดิษฐ์มาประยุกต์ใช้ทำให้บริษัทต่าง ๆ มีแนวทางที่ดีที่สุดในการจัดสรรทรัพยากรบุคคล และมีความน่าสนใจทางเศรษฐกิจในหลายภาคธุรกิจและอุตสาหกรรม ส่วนเปลือกระบบผู้เชี่ยวชาญนั้นแตกต่างออกไปอย่างมากทั้งในเรื่องค่าใช้จ่าย ประสิทธิภาพในการทำงาน และ

ความเป็นมิตรกับผู้ใช้โดยระบบปัญญาประดิษฐ์ส่วนใหญ่จะขึ้นอยู่กับการใช้งานเปลือกระบบผู้เชี่ยวชาญ (Salim et al., 2002)

จากข้อมูลเบื้องต้นสามารถสรุปได้ว่า เปลือกระบบผู้เชี่ยวชาญ ช่วยลดเวลาและแรงงานที่ใช้ในการพัฒนาระบบผู้เชี่ยวชาญ พร้อมทั้งส่งผลให้ลดค่าใช้จ่ายที่อาจจะสูญเสียไปในการพัฒนาระบบใหม่ได้ และเป็นโปรแกรมเชิงพาณิชย์ที่นิยมใช้ร่วมกับโปรแกรมปัญญาประดิษฐ์อย่างแพร่หลาย



รูปที่ 2.9 แสดงโครงสร้างของเปลือกระบบผู้เชี่ยวชาญ (Chakraborty, 2010)

เปลือกระบบผู้เชี่ยวชาญเปรียบเหมือนกับระบบผู้เชี่ยวชาญกึ่งสำเร็จรูปที่สามารถเปลี่ยนขอบเขตความรู้ได้ ดังนั้นในการพัฒนาเปลือกระบบผู้เชี่ยวชาญจะเน้นไปที่การออกแบบให้ระบบสามารถรองรับฐานความรู้ที่หลากหลาย เพื่อที่จะสามารถนำมาใช้ได้ในงานทั่วไปทันที เพียงแค่พัฒนาเฉพาะส่วนของฐานความรู้ในด้านนั้น ๆ ขึ้นมาใหม่ให้สัมพันธ์กับการทำงานของเปลือกระบบผู้เชี่ยวชาญ ก็จะได้ระบบผู้เชี่ยวชาญสำหรับความชำนาญการนั้น

ตัวอย่างโปรแกรมเปลือกระบบผู้เชี่ยวชาญ

1. **FOCL (First Order Combined Learner)**

เป็นเปลือกระบบผู้เชี่ยวชาญที่ใช้งานการเรียนรู้ของเครื่องจักร (Machine Learning) โปรแกรมเขียนด้วยภาษา Common Lisp ส่วนของระบบผู้เชี่ยวชาญประกอบด้วยการค้นหากฎแบบย้อนกลับ และส่วนติดต่อแบบกราฟิกให้กับกฎและข้อเท็จจริง (Kantrowitz, 1997a)

2. **BABYLON**

เป็นโปรแกรมสำหรับพัฒนาระบบผู้เชี่ยวชาญ ซึ่งได้รวมเฟรม ข้อจำกัด ภาษาโปรล็อก และภาษาอธิบายสำหรับใช้ในงานวินิจฉัย โปรแกรมถูกเขียนด้วยภาษา Common Lisp และมีพอร์ตแยกเพื่อรองรับฮาร์ดแวร์ที่มีอยู่หลากหลายแพลตฟอร์ม (Kantrowitz, 1997c)

3. MOBAL

เป็นระบบสำหรับพัฒนารูปแบบการดำเนินงานของโปรแกรมในโดเมนของการแสดงตรรกะของการสั่งซื้อครั้งแรก โดยการใช้ส่วนแสวงหาความรู้ของ MOBAL ทำให้สามารถพัฒนารูปแบบของโดเมนในแง่ของข้อเท็จจริงเชิงตรรกะและเชิงระเบียบวิธี สามารถตรวจสอบความรู้ได้ในขณะป้อน เพิ่มความรู้หรือว่าเปลี่ยนแปลงได้ตลอดเวลา มีกลไกอนุมานสามารถทำงานร่วมได้ทันที (Kantrowitz, 1997a)

4. RT-Expert

เป็นโปรแกรมระบบผู้เชี่ยวชาญประเภท Shareware สำหรับนักพัฒนาระบบผู้เชี่ยวชาญภาษา C/C++ โดย RT-Expert ประกอบด้วย Rule-compiler สำหรับภาษา C ทำงานร่วมกับ Borland Turbo C, Borland C++ และ Microsoft C/C++ โปรแกรม RT-Expert มีรุ่นสำหรับใช้เพื่อการศึกษาวิจัย หรือใช้ในงานส่วนบุคคล และมีรุ่นมืออาชีพที่ใช้ในงานเชิงพาณิชย์บนระบบปฏิบัติการ DOS, Windows และ Unix (Kantrowitz, 1997a)

5. FuzzyCLIPS

เป็นเปลือกระบบผู้เชี่ยวชาญที่เขียนด้วยภาษา CLIP เหมาะสำหรับแทนความรู้และจัดการกับข้อเท็จจริงและกฎที่คลุมเครือ FuzzyCLIPS สามารถจัดการด้วยเงื่อนไขที่คลุมเครือ หรือไม่แน่นอน และสามารถใช้ร่วมกันกับกฎและข้อเท็จจริงแบบปกติได้อย่างอิสระ (Kantrowitz, 1997b)

2.3 การเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ (Constraint Logic Programming : CLP)

CLP เป็นรูปแบบของการเขียนโปรแกรมด้วยเงื่อนไขบังคับ ซึ่งมีการเขียนในเชิงตรรกะโดยขยายแนวคิดไปถึงการทำให้ในเงื่อนไขบังคับเป็นจริง CLP เป็นการเขียนโปรแกรมเหมือนโปรแกรมเชิงตรรกะปกติเพียงแต่ได้เพิ่มส่วนของเงื่อนไขบังคับเข้าไปด้วย (Wikipedia, 2011c)

การเขียนโปรแกรมในรูปแบบภาษา CLP และ Prolog มีข้อแตกต่างสรุปได้ ดังนี้ (Niederlinski, 2011)

1. ในโปรแกรมโปรล็อก ขอบเขตของตัวแปรจะถูกประกาศแบบอ้อม ๆ โดยองค์ประกอบของลิสต์ ซึ่งอยู่อย่างกระจัดกระจายใน Predicate ต่าง ๆ ที่อยู่ในตำแหน่งต่าง ๆ ของโปรแกรม ส่วนในโปรแกรม CLP จะบรรจุขอบเขตของตัวแปรไว้ที่ส่วนต้นของโปรแกรม โดยมีการประกาศขอบเขตของตัวแปรที่จะใช้ในโปรแกรมไว้อย่างชัดเจน

2. โปรล็อกสามารถควบคุมหรือจัดการได้เฉพาะตัวแปรที่ถูกนิยามไว้ในขอบเขตของพจน์ (Term) ส่วน CLP สามารถควบคุมตัวแปรจากขอบเขตกว้าง ๆ ที่แน่นนอนได้ เช่น ขอบเขตของเลขจำนวนเต็ม (Integer Domain) ขอบเขตของเลขจำนวนจริง (Real Domain) ขอบเขตของสัญลักษณ์ (Symbolic Domain)

3. ในโปรล็อกกระบวนการ Constraint Propagation จะถูกกระทำผ่านการทำให้เป็นหน่วยเดียวกัน (Unification) ส่วน CLP ใช้กระบวนการ Constraint Propagation ที่มีประสิทธิภาพกว่า เรียกว่า Consistency Techniques

4. ภาษา CLP ใช้กระบวนการค้นหาที่มีประสิทธิภาพมากกว่า เมื่อเปรียบเทียบกับ Depth First Search ของโปรล็อก

5. ภาษา CLP รวบรวมกระบวนการ Mentioned Search และเทคนิค Constraint Propagation เข้าไว้ด้วยกันได้อย่างมีประสิทธิภาพ และง่ายต่อการใช้งาน Search และ Propagation Solver

6. ในโปรแกรมโปรล็อกกระบวนการค้นหาจะเริ่มขึ้นโดยอัตโนมัติเมื่อไรก็ตามที่ Query ถูกเรียก ส่วนภาษา CLP กระบวนการค้นหาจะเริ่มขึ้นโดย การทำงานของเพรดิเคตที่เป็น Built-in อยู่ภายในโปรแกรม เช่น เพรดิเคต labeling/1 ที่ถูกนิยามไว้ดังนี้

```
labeling([H|T]):-
        indomain(H),
        labeling (T).
labeling([]).
```

7. ภาษา CLP มีขอบเขตของ Built-in ขนาดใหญ่มากที่ให้บริการแก่ผู้ใช้ และมีประสิทธิภาพมากกว่าภาษาโปรล็อก

8. เมื่อประมวลผลโปรแกรมภาษาโปรล็อก ใน ECLiPSe นั้นไม่ต้องใส่ Library ไว้ในโปรแกรม แต่ในทางกลับกันถ้าเป็นการทำงานกับโปรแกรมภาษา CLP การทำงานจะต้องเริ่มจากการประกาศ Library ที่ต้องการก่อน ตัวอย่างของ Library ใน CLP มีดังนี้ ic, ic_global, cumulative, ic_edge_finder, ic_edge_finder3, ic_sets และ ic_symbolic เป็นต้น

2.3.1 ไวยากรณ์ (Syntax)

พื้นฐานภาษาที่ใช้ใน CLP นั้นมาจากภาษาโปรล็อก ซึ่งไวยากรณ์ในภาษาโปรล็อกแบ่งออกเป็น 2 กลุ่ม คือ Base Terms กับ Compound Terms มีรายละเอียด ดังนี้ (Apt and Wallance, 2006)

1. Base Terms

- ตัวแปร (Variables) คือ สายอักขระ (Strings) ที่ขึ้นต้นด้วยตัวอักษรพิมพ์ใหญ่ (A-Z) หรือ ใช้เส้นขีดล่าง "_" (Underscore) แทนตัวแปรที่ไม่สนใจ เช่น X3
- ตัวเลข (Numbers) คือ ตัวเลขจำนวนเต็ม เช่น -1, 1, 0 หรือ จำนวนจริง เช่น 1.34
- สายอักขระ (Strings) คือ การเรียงกันของตัวอักษรภายในเครื่องหมายอัญประกาศคู่ (Double Quote) ซึ่งภายในจะเป็นลิสต์ของรหัส ACSII เช่น "My Name"
- อะตอม (Atoms) คือ สายอักขระที่ขึ้นต้นด้วยตัวอักษรพิมพ์เล็ก (a-z) เช่น x3 หรือ การเรียงกันของตัวอักษรภายในเครื่องหมายอัญประกาศเดี่ยว (Single Quote) เช่น 'My Name'

สำหรับค่าคงที่ (Constant) จะหมายถึง ตัวเลข สายอักขระ และอะตอม เท่านั้น โดยจะเรียกข้อมูลนี้ว่า Ground Terms

2. Compound Terms

ประกอบไปด้วย ฟังก์เตอร์ (Functor) และอาร์กิวเมนต์ (Arguments) มีรายละเอียด ดังนี้

- ฟังก์เตอร์ (Functor) คือ ส่วนประกอบอะตอมที่อยู่หน้าวงเล็บเปิด ซึ่งภายในวงเล็บจะประกอบไปด้วยอาร์กิวเมนต์ต่าง ๆ ที่ถูกคั่นด้วยลูกน้ำ "," (Comma) เช่น f(a,b,c) หมายถึง ฟังก์เตอร์ชื่อ f มี อาร์กิวเมนต์ อยู่ 3 ตัว คือ a b และ c
- อาร์กิวเมนต์ (Arguments) คือ สิ่งที่อยู่ภายในวงเล็บของฟังก์เตอร์ ซึ่งเป็นได้ทั้ง ตัวแปร ตัวเลข สายอักขระ อะตอม หรือ ฟังก์เตอร์อื่น ๆ ได้ เช่น h(A, f(g, 'Twenty' , X) , X).
- อาริตี้ (Arity) คือ จำนวนอาร์กิวเมนต์ของฟังก์เตอร์ เช่น f(a,b). มีอาริตี้เป็น 2 สามารถเขียนเป็น f/2 ฟังก์เตอร์ที่มีจำนวนอาร์กิวเมนต์ที่แตกต่างกัน แม้ว่าฟังก์เตอร์นั้นจะชื่อเดียวกัน ถือว่า เป็นฟังก์เตอร์ที่แตกต่างกัน เช่น f(a,b). เป็น f/2 จะไม่เหมือนกับ f(a,b,c). ที่เป็น f/3

2.3.2 ส่วนประกอบพื้นฐานของโปรแกรมภาษาโปรล็อกที่ใช้ใน CLP

โปรแกรม (Program) สร้างจากกระบวนการ (Procedure) หลาย ๆ กระบวนการรวมกัน กระบวนการสร้างจากประโยคเล็ก ๆ (Clause) หลายประโยคที่จบประโยคด้วยเครื่องหมาย “.” และประโยค คือ ข้อเท็จจริง (Fact) หรือ กฎ (Rule) โดยในโปรแกรมไม่ต้องมีส่วนเฉพาะเจาะจงในการระบุจุดเริ่มโปรแกรม เช่น ฟังก์ชัน main เหมือนในโปรแกรมภาษาอื่น ๆ ส่วนประกอบพื้นฐานของโปรแกรม มีดังนี้ (Apt and Wallance, 2006)

1. Atomic Goal คือข้อความเดี่ยวประกอบไปด้วยเพรดิเคต (Predicate) และอาร์กิวเมนต์ของเพรดิเคต เพรดิเคต คือ ฟังก์เตอร์ที่ถูกตั้งชื่อตามการกระทำที่จะเกิดขึ้นกับอาร์กิวเมนต์ที่อยู่ภายในวงเล็บของฟังก์เตอร์นั้น ประกอบไปด้วย ชื่อเพรดิเคต (Predicate Name) และ เพรดิเคตอาริตี้ (Predicate Arity) เช่น eat(‘Tom’, ‘Orange’). หมายถึง ทอมกินส้ม โดยมีเพรดิเคตชื่อ eat เป็นตัวบอกการกระทำ มีอาร์กิวเมต์ คือ ‘Tom’ และ ‘Orange’ นั่นคือ เพรดิเคตอาริตี้ เป็น 2 สามารถเขียนแทนด้วย eat/2

2. Query หรือ Goal คือ ข้อความที่ประกอบขึ้นจากหนึ่งหรือหลาย Atomic Goal และลงท้ายด้วย “.” ใช้ทำหน้าที่เป็นข้อคำถาม หรือเป็นจุดเริ่มต้นการประมวลผลโปรแกรม

3. กฎ (Rule) คือข้อความที่ประกอบด้วยส่วนของ Head ซึ่งเป็น Atomic Goal ตามด้วย “:-” ซึ่งอยู่หน้าส่วนของ Body และจบประโยคด้วย “.” เช่น p(b,Y) :- q(Y) , r(Y,c). หมายถึง กฎนี้มี p/2 เป็นส่วน Head และมี q/1 กับ r/2 เป็น Body

4. ข้อเท็จจริง (Fact) คือ Atomic Goal ที่จบด้วย “.” เช่น p(a,b).

2.3.3 โครงสร้างของโปรแกรมภาษาโปรล็อกที่ใช้ใน CLP

1. ตัวดำเนินการ (Operators) หมายถึง ฟังก์ชันของสัญลักษณ์ หรือ เพรดิเคตที่ถูกประกาศเป็นมาตรฐานเพื่อเป็นทางเลือกในการใช้รูปแบบใดรูปแบบหนึ่ง เช่น 1+2 หรือ +(1,2) โดยที่ “+” เป็นตัวดำเนินการ สามารถแบ่งเป็น 2 รูปแบบ คือ Binary Operators กับ Unary Operator มีรายละเอียด ดังนี้ (Apt and Wallance, 2006)

- Binary Operators : Infix Notation หมายถึง การประกาศตัวดำเนินการโดยกำหนดให้เป็นตำแหน่งอยู่ตรงกลาง ระหว่างตัวถูกดำเนินการ (Operand) สองตัว เช่น เพรดิเคต before/2. หรือ before(10,11). ถ้าต้องการเขียนในลักษณะ 10 before 11 สามารถทำได้โดยการประกาศตัวดำเนินการ (Operator Declaration) ดังนี้ :-op(700, xfx, before).

- Binary Operators : Binding Order หมายถึง ลำดับความสำคัญของตัวดำเนินการแต่ละตัว วัดจากค่าลำดับความสำคัญ หรือ ไพรออริตี้ (Priority) ที่กำหนดไว้ตั้งแต่ตอนประกาศตัวดำเนินการนั้นขึ้น เช่น เพรดิเคต :/2. มีไพรออริตี้เป็น 600 ส่วนเพรดิเคต before/2. มี

ไพรออริตี้เป็น 700 เมื่อเขียนในลักษณะ 10:00 before 11:00 โปรแกรมจะเลือกทำงานในเพรดิเคตที่มีค่า 600 ก่อน 700 คล้ายกับการใส่วงเล็บเพื่อกำหนดการทำงานเพื่อโปรแกรมจะทำงานไม่ผิดลำดับ หรือสามารถมองในลักษณะ (10:00) before (11:00) ได้

- Binary Operators : Associativity หมายถึง การเชื่อมโยงการทำงานระหว่างตัวถูกดำเนินการมากกว่าสองตัวที่ใช้ตัวดำเนินการเดียวกัน และการกำหนดค่าไพรออริตี้ไม่สามารถทำได้ เช่น 1+2+3+4 ที่มองได้หลายแบบ คือ ((1+2)+3)+4 หรือ 1+(2+(3+4)) ฯลฯ จึงต้องกำหนดรูปแบบในการประกาศตัวดำเนินการให้สามารถบ่งบอกลำดับความสำคัญของตัวถูกดำเนินการแต่ละตัวได้ มีหลายรูปแบบ ดังนี้

:-op(500, yfx, +). คือ กำหนดให้ตัวถูกดำเนินการ y สำคัญกว่าตัวถูกดำเนินการ x (Left Associativity) เช่น 1+2+3+4 จะหมายถึง ((1+2)+3)+4

:-op(500, xfy, ++). คือ กำหนดให้ตัวถูกดำเนินการ y สำคัญกว่าตัวถูกดำเนินการ x (Right Associativity) เช่น 1++2++3++4 จะหมายถึง 1++(2++(3++4))

กรณีของเพรดิเคต mod/2. มีการประกาศในลักษณะของ Non-associative Operator คือ :-op(300, xfx, mod). การกำหนดให้ตัวถูกดำเนินการทั้งสองข้างมีความสำคัญเท่ากัน เช่น 20 mod 7 mod 3 โปรแกรมจะปฏิเสธการทำงานและบอกว่าประโยคนี้จำเป็นต้องใส่วงเล็บด้วย (Syntax Error: Bracket Necessary)

นอกจากการประกาศตัวดำเนินการแบบ Infix แล้วโปรแกรม ECLiSe ยังอนุญาตให้ประกาศตัวดำเนินการในลักษณะของ Prefix ด้วย คือ :-op(800, fxx, bin). เช่น bin 2 2. หรือ bin(2,2).

- Unary Operator หมายถึง การประกาศตัวดำเนินการที่กระทำกับตัวถูกดำเนินการเพียงตัวเดียว มีรูปแบบการประกาศ คือ fx, xf, fy และ yf เช่น :-op(500, fx, -). ซึ่งมีค่าไพรออริตี้มากกว่า before เขียนได้ว่า –T1 before T2. หรือ (-T1) before T2.

2. การควบคุม (Control) หมายถึง ตัวดำเนินการทางตรรกศาสตร์ต่าง ๆ ที่ใช้ควบคุมการทำงานของโปรแกรมด้วยการตรวจสอบค่าความจริง เช่น (Apt and Wallance, 2006)

- ถ้า-แล้ว (if-then-else) เช่น if B then S else T ในโปรแกรมภาษาโปรล็อกสามารถเขียนในรูปแบบของ B -> S ; T. หมายถึง ถ้าพจน์ B เป็นจริงแล้ว ให้ทำพจน์ S แต่ถ้าพจน์ B เป็นเท็จแล้ว ให้ทำพจน์ T แทน

- การเลือกกระทำ (Disjunction) เช่น Q or R ในโปรแกรมภาษาโปรล็อกสามารถเขียนในรูปแบบของ Q ; R หมายถึง ให้เลือกทำพจน์ Q หรือ R พจน์ใดพจน์หนึ่ง

- นิเสธ (Negation) ในโปรแกรมภาษาโปรล็อก คือ “not/1” หรือ “\+/1” เขียนเป็น “not Q” หรือ “\+ Q” เช่น not a=b หรือ \+ a=b โปรแกรมจะตอบ Yes เพราะ a นั้นไม่

เท่ากับ b ควรตอบ No แต่การใส่นิเสธไว้ข้างหน้าพจน์ทำให้ได้ผลตรงข้าม และในทำนองเดียวกัน not a=a หรือ \+ a=a โปรแกรมจะตอบว่า No

- ครั้งเดียว (Once) เป็นการระบุให้โปรแกรมเลือกพจน์ที่เป็นจริงพจน์แรก โดยจะไม่พิจารณาพจน์อื่น ๆ อีก โดยใช้ once/1. เช่น once(X=a ; X=b). โปรแกรมจะตอบ X=a และหยุดการค้นหาทันที

- การตัด (Cut) เป็นการระบุตัดเส้นทางการค้นหาของโปรแกรม จะใช้เมื่อพบว่าพจน์นั้นเป็นจริงแล้ว และไม่จำเป็นต้องค้นหาในเส้นทางอื่น ๆ อีก เช่น once/1. มีการประกาศตัวดำเนินการ คือ :-op(900, fy, once). และเขียนกฎ (Rule) ไว้ว่า once Q :- Q, !. หมายถึง เมื่อพบว่าพจน์ Q เป็นจริงแล้ว ให้ทำพจน์ Q และไม่ต้องพิจารณากรณีอื่นอีก

3. ลิสต์ (List) คือ โครงสร้างข้อมูลที่อนุญาตให้แทรกองค์ประกอบที่ด้านหน้าเพียงอย่างเดียว การประกาศลิสต์มี 3 รูปแบบ (Apt and Wallance, 2006)

- [ ] เป็น สัญลักษณ์บ่งบอกว่าข้อมูลที่อยู่ภายในเครื่องหมายนี้ ([ ]) เป็นสมาชิกของลิสต์นี้ โดยประเภทสมาชิกในลิสต์เป็นได้ทั้ง ตัวแปร ตัวเลข สายอักขระ อะตอม หรือฟังก์เตอร์ และสมาชิกแต่ละตัวจะคั่นด้วยเครื่องหมาย “,” เช่น [ ], [a], [a, b, f(a)], [“Name”, [1,2]] เป็นต้น

- [. | ..] เป็นการประกาศลิสต์แบบโครงสร้าง ถ้า T เป็นลิสต์แล้ว [H|T] ก็จะเป็นลิสต์ด้วย โดยที่ตัวแปร H เรียกว่า Head แทนสมาชิกตัวแรกของลิสต์ และตัวแปร T เรียกว่า Tail แทนลิสต์ของสมาชิกที่เหลือนอกจากสมาชิกตัวแรกที่แทนด้วยตัวแปร H เช่น [a, b, c] = [a|[b|[c]]] = [a, b|[c]] = [a|[b|[c|[ ]]]], [“Name”, [1,2]] = [“Name”|[[1,2]|[ ]]] หรือ f(X) = f([a, b, c|[d, e, f]]) โปรแกรมจะตอบว่า X = [a, b, c, d, e, f] เป็นต้น

- การนิยามแบบวนซ้ำด้วย “. /2” เช่น [a, b, c] = .(a,.(b,.(c,[]))) เป็นต้น

ภายในลิสต์สามารถมีจำนวนสมาชิกได้ตั้งแต่ 0 คือ เป็นลิสต์ว่าง (Empty List) แทนด้วยสัญลักษณ์ “[ ]” ไปจนถึงมีจำนวนไม่จำกัด

4. กระบวนการวนซ้ำ (Recursion) เป็นการเรียกเพรดิเคตเดิมซ้ำ ๆ ในส่วนของ Body ภายในกฎของเพรดิเคตชื่อเดียวกัน สามารถหยุดการวนซ้ำได้ก็ต่อเมื่อไม่ได้เรียกซ้ำอีก (Wikipedia, 2011h; Apt and Wallance, 2006) เช่น

```
app([ ], Ys, Ys).
app([X | Xs], Ys, [X | Zs]) :- app(Xs, Ys, Zs).
```

5. การประเมินค่า (Evalution) ในการประมวลผลคำสั่งภาษาโปรล็อกนั้นเมื่อมีการสอบถาม (Query) โปรแกรมประมวลผลก็จะค้นหาข้อเท็จจริงหรือกฎที่ทำให้ได้ข้อเท็จจริง กระบวนการในการค้นหาคำตอบที่โปรล็อกใช้เรียกว่า SLD Resolution เป็นกระบวนการค้นหา

แบบต้นไม้เพื่อหาคำตอบ โดยเริ่มจากโหนดรากลึกลงไปเรื่อย ๆ จนถึงโหนดใบ (Depth-first search) เมื่อทำการค้นหาถึงโหนดใบแล้วในภาษาโปรล็อกมีกระบวนการค้นหาย้อนกลับขึ้นไปทีละขั้นที่เรียกว่า Backtracking เป็นกระบวนการเพื่อหาคำตอบทั้งหมดที่เป็นไปได้ ซึ่งในส่วนนี้ทำให้โปรแกรมภาษาโปรล็อกเป็นโปรแกรมที่มีการค้นหาคำตอบแบบย้อนหลัง (Backward Chaining) (Wikipedia, 2011h; วิกิพีเดีย สารานุกรมเสรี, 2554ค)

2.3.4 โครงสร้างเฉพาะของโปรแกรมภาษา CLP

1. ไลบรารี่ (Library) เป็นการกำหนดเพื่อบอกให้โปรแกรมรับรู้ว่า ตัวแปรและข้อมูลที่กระทำกันภายในโปรแกรมนั้นเหมาะกับใช้แก้ปัญหาของข้อมูลในลักษณะใด โดยสรุปไว้ในตารางที่ 2.1 (Apt and Wallance, 2006)

ตารางที่ 2.1 สรุปไลบรารี่เบื้องต้น

ชื่อไลบรารี่	ชนิดข้อมูล	เรียกใช้	ประกาศตัวแปร	เงื่อนไขบังคับ	การทำงาน
Symbolic Domains	Symbol	:-lib(sd).	X&::[a,b,c], Y&::[a,c,d]	X&=Y, X&\=Y	Constraint Satisfaction Problem
Finite Domain Constraints	Integer	:-lib(fd).	X::[1,2,3], Y::1..5	Y#=X/2, X#>=2, Y#<4	Domain Propagation
	Symbol		X::[a,b,c], Y::[a,c,d]	X=Y, X\=Y	
Interval Constraints	Integer	:-lib(ic).	X::[1,2,3], Y::1..5	Y#=X/2, X#>=2, Y#<4	Bounds/Domain Propagation
	Real		X::[1,2,3], Y::1..5	Y$=X/2, X$>=2, Y$<4	

2. CSP (Constraint Satisfaction Problems) หมายถึง ลักษณะการเขียนโปรแกรมแก้โจทย์ปัญหาที่ต้องทำให้เงื่อนไขที่โจทย์ระบุเป็น กระบวนแก้ปัญหากำหนดลำดับตัวแปรที่แน่นอน และชุดเงื่อนไขบังคับที่แน่นอน ตัวแปรที่ใช้ใน CSP เรียกว่า ตัวแปรตัดสินใจ (Decision Variables) เพราะว่า CSP นั้นอยู่ในบริบทของการตีความตัวแปร (Apt and Wallance, 2006)

3. COP (Constraint Optimization Problems) ลักษณะโจทย์ปัญหาที่เรียกว่า COP โดยทั่วไปแล้ว คือ การหาคำตอบที่เหมาะสมที่สุด (Optimal Solutions) ที่ทำให้ชุดของเงื่อนไขบังคับเป็นจริง และให้ cost ที่ดีที่สุดตามที่ระบุโดยฟังก์ชัน cost (Apt and Wallance, 2006)

4. การวนซ้ำ (Iteration) การวนซ้ำใน CLP แตกต่างจากโปรล็อก ในโปรล็อกจะเป็นการเรียกชื่อเพรดิเคตเดิมซ้ำ ๆ กัน แต่ใน CLP รูปแบบการเขียนโค้ดจะคล้ายกับ Loop แบบต่าง ๆ ในภาษาซี เช่น (Apt and Wallance, 2006)

- (foreach(El, [a,b,c]) do write (El)). ผลลัพธ์ที่ได้ abc
- (count(I,1,4) do write (I)). ผลลัพธ์ที่ได้ 1234
- (foreach(El, [a,b,c]), count(I,1,Max) do write (I-El)). ผลลัพธ์ที่ได้ 1-a2-b3-c
- (for(I,1,3) do (for(J,5,9), param(I) do K is I*J, write(K), write(' '))). ผลลัพธ์ที่ได้ 5 6 7 8 9 10 12 14 16 18 15 18 21 24 27
- (multifor([I,J], [1,5], [3,9]) do K is I*J, write(K), write(' ')). ผลลัพธ์ที่ได้ 5 6 7 8 9 10 12 14 16 18 15 18 21 24 27
- (foreacharg(Arg, p(a,b,c)) do write(Arg)). ผลลัพธ์ที่ได้ acb
- (formto([a,b,c],[Head|Tail],Tail,[ ]) do write(Head)). ผลลัพธ์ที่ได้ abc
- ฯลฯ

2.3.5 โปรแกรม ECLiPSe

ECLiPSe เป็นซอฟต์แวร์สำหรับการพัฒนาและการใช้งานของการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ (Constraint Logic Programming : CLP) ในขอบเขตของงานด้านการเพิ่มประสิทธิภาพ การวางแผน การตั้งเวลา การจัดสรรทรัพยากร การจัดตารางเวลา และการขนส่ง เป็นต้น เหมาะกับการเรียนการสอนด้านการแก้ปัญหาคณิตศาสตร์ เช่น การสร้างแบบจำลองปัญหา การเขียนโปรแกรมด้วยเงื่อนไขบังคับ การเขียนโปรแกรมทางคณิตศาสตร์ และเทคนิคในการค้นหา เป็นต้น (Wikipedia, 2011d)

ภาษา ECLiPSe มีกระบวนการค้นหาแบบย้อนกลับที่เข้ากันได้กับภาษาโปรล็อก และมีการสนับสนุนภาษาที่แตกต่างกันอีกหลายภาษา เช่น Ada, C, C++, COBOL, Perl, PHP, Python, R, Ruby, Scala, Clojure, Groovy และ Scheme (Wikipedia, 2011e)

2.4 งานวิจัยที่เกี่ยวข้อง

ในการศึกษาและพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ ผู้วิจัยได้ทำการศึกษาค้นคว้างานวิจัยในอดีตที่เกี่ยวข้องกับการพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับดังสรุปในตารางที่ 2.2 และมีรายละเอียดดังต่อไปนี้

วิทยานิพนธ์ของ Muli (1992) แห่งมหาวิทยาลัย Ottawa ได้นำเสนอการประเมินผลประโยชน์ที่ได้จากการเขียนโปรแกรมทางคณิตศาสตร์รวมกับระบบผู้เชี่ยวชาญ เพื่อแก้ปัญหางบประมาณโดยใช้วิธีเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ จากการศึกษาโปรแกรมประยุกต์อย่างง่ายในการแก้ปัญหาการจัดลำดับใน VP-Expert ทำให้พบข้อจำกัดของเปลือกระบบผู้เชี่ยวชาญบางระบบ ซึ่งสามารถทำให้มีศักยภาพมากขึ้นได้โดยการเขียนโปรแกรมเชิงตรรกะที่มีการเขียนโปรแกรมทางคณิตศาสตร์ภายในระบบผู้เชี่ยวชาญ ซึ่งผู้วิจัยเลือกใช้ภาษาโปรล็อกในการสร้างโปรแกรมทดสอบโดยเขียนด้วยโปรแกรม Turbo-Prolog (ตัวแปลภาษาโปรล็อกที่บริษัทบอร์แลนด์เป็นผู้พัฒนาขึ้นเพื่อใช้กับไมโครคอมพิวเตอร์) อีกผลงานหนึ่งในวิทยานิพนธ์นี้คือ การใช้วิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับในการแก้ปัญหางบประมาณเงินทุน ซึ่งจากการทดลองสร้างพบว่าโปรแกรมจะมีประสิทธิภาพสูงสุด เมื่อมีวิธีที่ใช้ในการแก้ปัญหาเพียงไม่กี่วิธี โดยผู้วิจัยได้พัฒนาในส่วนนี้ด้วย CLP(R) ซึ่งเป็นส่วนขยายที่เหมาะสมสำหรับโปรล็อก และตอบสนองต่อข้อบังคับเกี่ยวกับพีชคณิตที่จำเป็นต้องทำ เพื่อแก้ปัญหาการโปรแกรมเชิงเส้น

งานวิจัยของ Roach and Berghel (1992) ได้นำเสนอวิธีการสำหรับรวมขั้นตอนทางคณิตศาสตร์แสดงในส่วนประกาศของ rule-base ได้โดยที่ยังเป็นส่วนสำคัญของ proof tree ที่สร้างขึ้นโดยกลไกอนุมาน โดยในงานวิจัยชิ้นนี้ได้อธิบายถึงการใช้การเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ (CLP) ในการจัดการกระบวนการทางคณิตศาสตร์ และกล่าวถึงวิธีที่ใช้ในการจัดการส่วนอธิบายของระบบผู้เชี่ยวชาญที่อยู่ใน proof tree และผู้วิจัยได้ใช้ภาษาโปรล็อกในการสร้างระบบผู้เชี่ยวชาญ

งานวิจัยของ Lee and Kwon (1995) ได้นำเสนอแผนพัฒนาระบบผู้เชี่ยวชาญที่เลือกใช้แนวคิดของข้อจำกัด และ ปัญหาของความพึงพอใจในกฎ (constraint and rule satisfaction problems) ภายใต้วัตถุประสงค์ที่หลากหลาย สาเหตุเนื่องมาจากในการพัฒนาระบบผู้เชี่ยวชาญแต่ละระบบไม่ได้ประสบความสำเร็จเหมือนซอฟต์แวร์ทั่ว ๆ ไปทั้งหมด บางระบบอาจจะล้มเหลวด้วยสาเหตุต่าง ๆ รวมถึงความล้มเหลวที่เกิดจากการวางแผนระบบที่ทำได้ยาก และได้เสนอกระบวนการให้เหตุผลแบบครบวงจรเพื่อให้สอดคล้องกับแผนพัฒนาระบบผู้เชี่ยวชาญซึ่งมีคุณสมบัติดังต่อไปนี้

1. ส่วนสร้างการตัดสินใจ สามารถโต้ตอบเพื่อรับข้อมูลเกี่ยวกับปัญหาของสถานการณ์และสามารถควบคุมกระบวนการแก้ปัญหาที่ขัดแย้งกันระหว่างหลาย ๆ วัตถุได้

2. มีการให้เหตุผลพร้อม ๆ กันด้วยการเริ่มต้นจากจุดต่าง ๆ กัน เพื่อเพิ่มประสิทธิภาพในการค้นหา
3. วิธีการให้เหตุผลสำหรับระบบ rule-base และวิธีการกระจายสำหรับ CSP

โดยงานวิจัยชิ้นนี้มีการพัฒนาด้วย UNIK-Object และ ภาษา LISP ซึ่งได้มีการประยุกต์เป็นโปรแกรมที่ปรึกษาในการวางแผนภาษี ซึ่งได้จากการวางแผนพัฒนาระบบผู้เชี่ยวชาญในงานวิจัยนี้

Eronen and Zitting (2001) ได้นำเสนอการพัฒนาระบบผู้เชี่ยวชาญสำหรับวิเคราะห์กฎของไฟร์วอล โดยใช้การเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับบนโปรแกรม ECLiPSe ในการพัฒนาระบบ ผู้วิจัยได้นำเสนอเครื่องมือที่ช่วยในการบริหารจัดการสำหรับผู้ดูแลระบบในการวิเคราะห์กฎของไฟร์วอล โดยออกแบบให้สามารถโต้ตอบกับผู้ใช้ระบบได้เพื่อถามคำถามเกี่ยวกับเครือข่าย

นิตยา เกิดประสพ (2552) ได้นำเสนอการประมวลผลหลังกระบวนการทำเหมืองข้อมูล เป็นการใช้ประโยชน์จากข้อมูลที่ได้หลังจากการทำเหมืองข้อมูล ซึ่งอยู่ในรูปแบบของความรู้ที่มีลักษณะเป็น กฎถ้า-แล้ว (IF-THEN Rule) สอดคล้องกับลักษณะของฐานความรู้ที่ประกอบไปด้วยข้อเท็จจริง และ กฎ โดยหลักการแล้วการทำเหมืองข้อมูลเป็นการสร้างรูปแบบความสัมพันธ์ระหว่างข้อมูลเพื่อใช้ในการพยากรณ์กับข้อมูลใหม่ในอนาคต เมื่อนำมาประยุกต์ใช้กับฐานความรู้ในระบบผู้เชี่ยวชาญ ทำให้สามารถลดขั้นตอนในการคัดกรองความรู้จากผู้เชี่ยวชาญได้ ด้วยการนำข้อมูลที่เกิดขึ้นในอดีตมาผ่านกระบวนการทำเหมือง ซึ่งมีความเป็นไปได้ที่อาจจะค้นพบความสัมพันธ์ระหว่างข้อมูลในแง่มุมที่ไม่เคยเห็นมาก่อน

จากการศึกษางานวิจัยที่เกี่ยวข้องกับการพัฒนาระบบผู้เชี่ยวชาญที่มีการบูรณาการแนวคิดเกี่ยวกับการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับพบว่า งานวิจัยส่วนใหญ่เห็นความสำคัญในการใช้แนวคิดของการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับจัดการกับส่วนที่เป็นการคำนวณทางคณิตศาสตร์ หรือฐานความรู้ของระบบผู้เชี่ยวชาญ ซึ่งระบบที่พัฒนาขึ้นนั้นจะกระทำร่วมกับฐานความรู้เพียงโดเมนเดียวที่สามารถควบคุมได้โดยการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ โดยเครื่องมือและวิธีการที่ใช้ในการเขียนโปรแกรมของแต่ละงานวิจัยนั้นมีหลากหลายรูปแบบ ทั้งในลักษณะของการนำแนวคิดมาใช้ร่วมในภาษาโปรล็อก หรือใช้โปรแกรมเฉพาะสำหรับพัฒนาโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ และผลงานที่ได้จากงานวิจัยเหล่านั้นมีทั้งลักษณะของการวางแผนพัฒนา หรือการสร้างต้นแบบระบบผู้เชี่ยวชาญที่ส่วนใหญ่ออกแบบให้ระบบมีความเป็นมิตรกับผู้ใช้สามารถโต้ตอบเพื่อรับข้อมูลจากผู้ใช้ได้ มีการวิเคราะห์ข้อมูลและ

ค้นหาคำตอบพร้อมนำเสนอเหตุผลประกอบเพื่อสนับสนุนการตัดสินใจของผู้ใช้ได้อย่างเหมาะสม ซึ่งส่วนนี้โดยส่วนใหญ่จะใช้เครื่องแปลภาษาที่เข้าใจภาษาโปรล็อก ซึ่งเป็นภาษาโปรแกรมเชิงตรรกะที่นิยมใช้ในงานวิจัยทางด้านปัญญาประดิษฐ์ แต่อย่างไรก็ตามงานวิจัยที่ผู้วิจัยได้ทำการศึกษานี้ยังมุ่งประเด็นไปที่การสร้างระบบผู้เชี่ยวชาญเพื่อแก้ปัญหาภายในโดเมนใดโดเมนหนึ่ง ไม่ได้พัฒนาออกมาในลักษณะของเปลือกระบบผู้เชี่ยวชาญที่สามารถเปลี่ยนฐานความรู้ที่ใช้ได้ และโดเมนของปัญหาที่แก้ไขจะเกี่ยวข้องกับกระบวนการทางคณิตศาสตร์เป็นส่วนใหญ่

ตารางที่ 2.2 แสดงการสรุปเปรียบเทียบประเด็นต่าง ๆ ของงานวิจัยที่เกี่ยวข้อง โดยบทความวิจัยที่เกี่ยวข้องประกอบด้วย “ก” แทนงานวิทยานิพนธ์ของ Muli(1992) “ข” แทนงานวิจัยของ Roach and Berghel (1992) “ค” แทนงานวิจัยของ Lee and Kwon (1995) “ง” แทนงานวิจัยของ Eronen and Zitting (2001) “จ” แทนโครงการวิจัยของนิตยา เกิดประสพ (2552) และ “ฉ” แทนงานวิจัยเรื่อง การพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ (งานวิจัยของวิทยานิพนธ์ฉบับนี้)

ตารางที่ 2.2 สรุปเปรียบเทียบงานวิจัยที่เกี่ยวข้องกับการพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยเงื่อนไขบังคับ

กระบวนการทำงาน	**งานวิจัยที่เกี่ยวข้อง**					
	ก	**ข**	**ค**	**ง**	**จ**	**ฉ**
ฐานความรู้ของระบบ						
ลักษณะของฐานความรู้						
ฐานความรู้เกี่ยวข้องกับกระบวนการทางคณิตศาสตร์	✓	✓	✓			
ฐานความรู้เกี่ยวข้องกับกฎที่ต้องอาศัยผู้เชี่ยวชาญเฉพาะด้าน				✓		
ฐานความรู้เกี่ยวข้องกับข้อความและสัญลักษณ์					✓	✓
ฐานความรู้ได้จากข้อมูลที่ผ่านกระบวนการทำเหมืองข้อมูล					✓	✓
การพัฒนา						
พัฒนาโดยใช้รูปแบบของภาษาโปรล็อก					✓	
พัฒนาโดยใช้แนวคิดของการกฎข้อบังคับ (Constraint)	✓		✓			✓
พัฒนาโดยใช้เครื่องมือ CLP(R)	✓					
พัฒนาโดยใช้รูปแบบของภาษา CLP		✓		✓		✓
ระบบผู้เชี่ยวชาญ						
ฐานความรู้						

ตารางที่ 2.2 สรุปเปรียบเทียบงานวิจัยที่เกี่ยวข้องกับการพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยเงื่อนไขบังคับ(ต่อ)

กระบวนการทำงาน	**งานวิจัยที่เกี่ยวข้อง**					
	ก	**ข**	**ค**	**ง**	**จ**	**ฉ**
ระบบสามารถใช้ได้กับข้อมูลภายในขอบเขตใดขอบเขตหนึ่ง	✓	✓	✓	✓		
ระบบสามารถเปลี่ยนฐานความรู้เป็นขอบเขตอื่น ๆ ได้					✓	✓
การแทนความรู้						
ปรับรูปแบบให้เหมาะกับกระบวนการทางคณิตศาสตร์	✓	✓	✓			
ปรับรูปแบบให้สอดคล้องกับกฎของรายการเข้าถึงข้อมูล				✓		
ปรับรูปแบบให้สามารถใช้ได้กับฐานความรู้ใด ๆ					✓	✓
กลไกอนุมาน						
ใช้กลไกอนุมานของโปรแกรม Turbo-Prolog	✓					
ใช้กลไกอนุมานของโปรแกรม SWI-Prolog					✓	
ใช้กลไกอนุมานของเครื่องมือ UNIK-Object			✓			
ใช้กลไกอนุมานของโปรแกรม ECLiPSe				✓		✓
ส่วนอธิบายความ						
มีการอธิบายถึงความเป็นมาของคำตอบที่ได้		✓	✓	✓	✓	✓
ส่วนติดต่อกับผู้ใช้						
สามารถโต้ตอบกับผู้ใช้ระบบได้			✓	✓	✓	✓
ใช้ภาษา Perl ในการเขียนคลาสแยกไฟล์รายการเข้าถึงข้อมูล				✓		
การพัฒนา						
พัฒนาในรูปแบบของระบบผู้เชี่ยวชาญ	✓	✓	✓	✓		
พัฒนาในรูปแบบของเปลือกระบบผู้เชี่ยวชาญ					✓	✓
พัฒนาระบบโดยใช้ภาษาโปรล็อก	✓	✓			✓	
พัฒนาระบบโดยใช้ภาษา CLP				✓		✓
พัฒนาระบบโดยใช้ภาษา LISP			✓			
การประยุกต์ใช้ระบบ						
วิจัยเพื่อทดสอบประสิทธิภาพ	✓					✓
วิจัยเพื่อวางแผนพัฒนาระบบ			✓			✓
มีการประยุกต์ใช้ระบบกับฐานความรู้จริง	✓	✓	✓	✓	✓	

บทที่ 3
วิธีดำเนินการวิจัย

ในบทนี้จะกล่าวถึงวิธีการวิจัย เครื่องมือที่ใช้ในการวิจัย และการเปรียบเทียบประสิทธิภาพกับเวลาที่ใช้ของระบบ โดยมีรายละเอียดดังนี้

3.1 วิธีการวิจัย

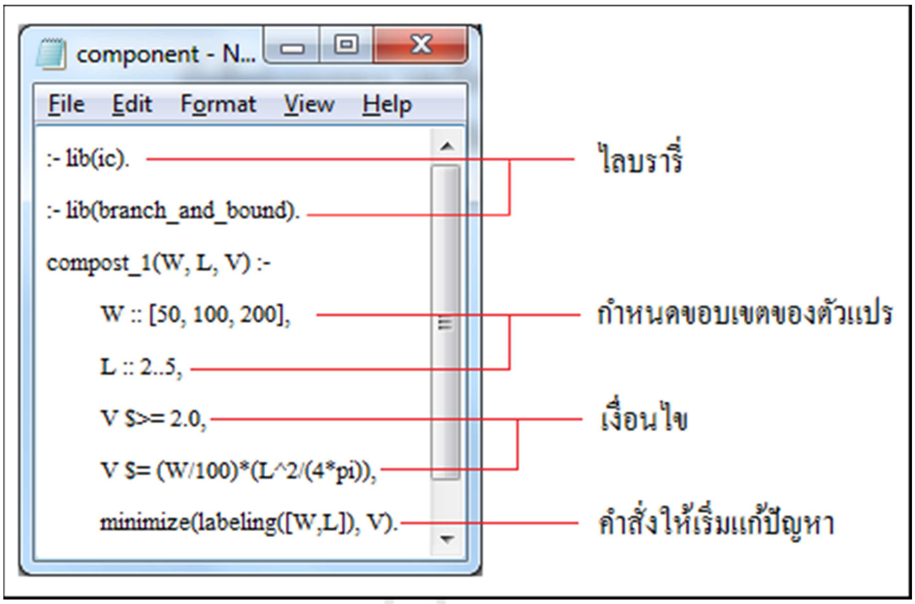
ในการพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับมีขั้นตอนในการศึกษาและพัฒนา ดังนี้

1. ศึกษารูปแบบการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ
2. ออกแบบฐานความรู้ในรูปแบบของการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ
3. ออกแบบเปลือกระบบผู้เชี่ยวชาญในรูปของการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ
4. การพัฒนาเปลือกระบบผู้เชี่ยวชาญ ตรวจสอบความถูกต้อง และทดสอบประสิทธิภาพ

โดยรายละเอียดของแต่ละขั้นตอนอธิบายในหัวข้อ 3.1.1 ถึง 3.1.4 ตามลำดับ

3.1.1 การศึกษารูปแบบการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ

รูปแบบการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับในโปรแกรม ECLiPSe นั้นมีรูปแบบเฉพาะ และมีการแบ่งสัดส่วนการทำงานไว้อย่างชัดเจน เช่น คำสั่งเรียกใช้ไลบรารี่ที่เกี่ยวข้องตามลักษณะของปัญหาที่จะแก้ ส่วนประกาศและกำหนดค่าขอบเขตของตัวแปร ส่วนที่เป็นข้อจำกัดหรือเงื่อนไขที่ใช้ในการแก้ปัญหา และคำสั่งที่ใช้เริ่มกระบวนการแก้ปัญหาของโปรแกรม ดังแสดงในรูปที่ 3.1



รูปที่ 3.1 โครงสร้างและส่วนประกอบของรูปแบบการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ

ในการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับนั้น การเลือกใช้ไลบรารี่จะต้องสัมพันธ์กับประเภทของเงื่อนไขที่ตั้งไว้ ซึ่งที่นิยมใช้โดยทั่วไปแล้วจะแบ่งเป็น 2 ชนิด คือ ic (Interval Constraint) เหมาะสำหรับแก้ปัญหาสมการคณิตศาสตร์ และ sd (Symbolic Domain) เหมาะสำหรับแก้ปัญหาเชิงข้อความหรือสัญลักษณ์ นอกจากความแตกต่างในการเลือกใช้ตามประเภทของเงื่อนไขแล้ว รูปแบบของสัญลักษณ์ที่ใช้ใน “คำสั่งดำเนินการ” จะต้องสัมพันธ์กับประเภทของเงื่อนไขด้วยเช่นกัน เช่น “#=” ใช้ในเงื่อนไขที่เป็นสมการเลขจำนวนเต็ม “$=” ใช้ในเงื่อนไขที่เป็นสมการเลขจำนวนจริง และ “&=” ใช้ในเงื่อนไขที่เป็นตัวอักษร ข้อความ หรือสัญลักษณ์ ส่วนสัญลักษณ์ที่ใช้สำหรับกำหนดค่าของตัวแปรนั้น ก็มีความแตกต่างตามประเภทของตัวแปร เช่น “::” ใช้ในการกำหนดค่าตัวแปรประเภทตัวเลข และ “&::” ใช้ในการกำหนดค่าตัวแปรประเภทตัวอักษร ข้อความ หรือสัญลักษณ์ โดยจะแสดงตัวอย่างในรูปที่ 3.2 สำหรับเงื่อนไขบังคับประเภทสมการเลขจำนวนเต็ม รูปที่ 3.3 สำหรับเงื่อนไขบังคับประเภทสมการเลขจำนวนจริง และรูปที่ 3.4 สำหรับเงื่อนไขบังคับประเภทตัวอักษร สัญลักษณ์ หรือข้อความ

```
:-lib(ic).
send([S,E,N,D,M,O,R,Y]):-
        S::0..9,E::0..9,N::0..9,D::0..9,
        M::0..9,O::0..9,R::0..9,Y::0..9,
        S#>=0,M#>=0,
        [C1,C2,C3,C4]::0..1,
        alldifferent([S,E,N,D,M,O,R,Y]),
        D+E#=Y+10*C1,
        C1+N+R#=E+10*C2,
        C2+E+O#=N+10*C3,
        C3+S+M#=O+10*C3,
        C4#=M.
```

```
?- send([S, E, N, D, M, O, R, Y]).
S = S{0 .. 9}
E = E{0 .. 9}
N = N{0 .. 9}
D = D{0 .. 9}
M = M{[0, 1]}
O = O{0 .. 9}
R = R{0 .. 9}
Y = Y{0 .. 9}
There are 12 delayed goals.
Yes (0.00s cpu)
```

(a) โปรแกรม (b) ผลลัพธ์

รูปที่ 3.2 ตัวอย่าง (a) รูปแบบการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับประเภทสมการเลขจำนวนเต็ม และ (b) ผลลัพธ์ของการประมวลผลโปรแกรม

จากรูปที่ 3.2 เป็นการเขียนโปรแกรมเพื่อแก้ปัญหาการหาเลขจำนวนเต็มระหว่าง 0 ถึง 9 แทนในอักขระ S, E, N, D, M, O, R, Y โดยตัวเลขที่แทนในแต่ละตัวอักษรจะไม่ซ้ำกัน และเงื่อนไขการแทนค่าต้องทำให้สมการ SEND + MORE = MONEY เป็นจริง

```
:-lib(ic).
:-lib(branch_and_bound).
compost_1(W,L,V):-
        W :: [50, 100, 200],
        L :: 2..5,
        V$>=2.0,
        V$=(W/100)*(L^2/(4*pi)),
        minimize(labeling([W,L]),V).
```

(a) โปรแกรม

```
?- compost_1(W, L, V).
W = 200
L = 4
V = 2.5464790894703251__2.546479089470326
There are 6 delayed goals.
Yes (0.02s cpu)

Output and Error Messages
lists.eco   loaded in 0.00 seconds
Found a solution with cost 2.5464790894703251__2.546479089470326
Found no solution with cost 2.5464790894703251 .. 1.546479089470326
```

(b) ผลลัพธ์

รูปที่ 3.3 ตัวอย่าง (a) รูปแบบการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับประเภทสมการเลขจำนวนจริง และ (b) ผลลัพธ์ของการประมวลผลโปรแกรม

จากรูปที่ 3.3 เป็นการเขียนโปรแกรมเพื่อแก้ปัญหาการหาเลขจำนวนจริงเพื่อแทนค่าตัวแปร W และ L และเงื่อนไขการแทนค่าต้องทำให้สมการ $V = (W/100) \times (L^2/4\pi)$ เป็นจริง โดย V หมายถึง Volume หรือปริมาตรของรูปทรงกระบอกที่สร้างจากลวดความหนา W และความยาว L ซึ่งความหนาของลวดมีให้เลือกใช้ 3 ขนาด คือ 50, 100 และ 200 เซนติเมตร ความยาวของลวดอยู่ระหว่าง 2-5 เมตร เมื่อนำลวดมาตัดเป็นรูปทรงกระบอกเพื่อใช้เป็นโครงของถังขยะจะต้องมีปริมาตรไม่ต่ำกว่า 2 ลูกบาศก์เมตร

```
:- lib(sd).
solve1(X,Y,Z) :-
        X &:: [a,c,d],
        Y &:: [a,b,c,d],
        Z &:: [c],
        X &= Y, Y &\= Z.
```

```
?- solve1(X, Y, Z).
X = X{[a, b]}
Y = Y{[a, b]}
Z = c
There is 1 delayed goal.
Yes (0.00s cpu)
```

(a) โปรแกรม (b) ผลลัพธ์

รูปที่ 3.4 ตัวอย่าง (a) รูปแบบการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับประเภทตัวอักษร สัญลักษณ์ หรือข้อความ และ (b) ผลลัพธ์ของการประมวลผลโปรแกรม

จากรูปที่ 3.4 เป็นการเขียนโปรแกรมเพื่อแก้ปัญหาการหาสมาชิกของตัวแปร X, Y, Z และเงื่อนไขการคงความเป็นสมาชิกต้องทำให้สมการ X&=Y และ Y&\=Z เป็นจริง

3.1.2 การออกแบบฐานความรู้ในรูปแบบของการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ

จากการศึกษารูปแบบการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ ผู้วิจัยได้ทำการออกแบบรูปแบบของฐานความรู้ใหม่ โดยแปลงจากรูปแบบเดิมที่เขียนด้วยภาษาโปรล็อก ซึ่งในกระบวนการนี้ได้เลือกฐานความรู้ที่ชื่อว่า vacation เป็นต้นแบบของฐานความรู้ที่ใช้ในการทดสอบเปลือกระบบผู้เชี่ยวชาญ โดยภายในฐานความรู้ vacation เป็นข้อมูลแนะนำจังหวัดสำหรับไปพักผ่อนในวันหยุด มีข้อมูลสำหรับคำถามเพื่อสอบถามผู้ใช้ทั้งหมด 4 คำถามเกี่ยวกับ ที่พัก (stay) สิ่งที่ชื่นชอบ (like) ของฝาก (buy) และสิ่งที่ต้องการเยี่ยมชม (will_see) โดยตัวอย่างการเขียนฐานความรู้นี้จะแสดงในรูปที่ 3.5

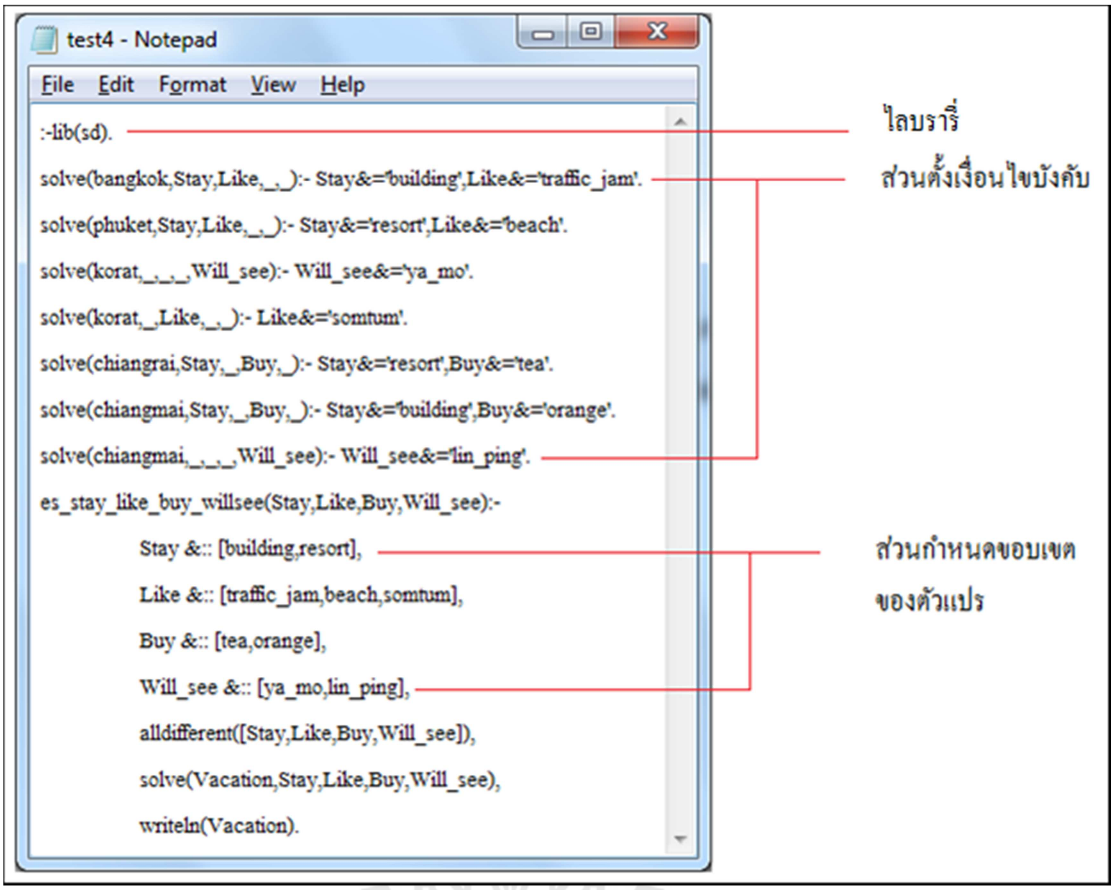
```
top_goal(X) :- vacation(X).


vacation(bangkok) :- stay(building), like(traffic_jam).
vacation(phuket) :- stay(resort), like(beach).
vacation(korat) :- will_see(ya_mo).
vacation(korat) :- like(somtom).
vacation(chiangrai) :- stay(resort), buy(tea).
vacation(chiangmai) :- stay(building), buy(orange).
vacation(chiangmai) :- will_see(lin_ping).


stay(X) :- menuask(stay, X, [building, resort]).
buy(X) :- menuask(buy, X, [tea,coffee, orange]).
like(X) :- menuask(like, X, [traffic_jam, beach, somtam]).
will_see(X) :- menuask(will_see, X, [thai_boxing, ya_mo, lin_ping]).
```

รูปที่ 3.5 ตัวอย่างฐานความรู้ vacation ที่เขียนด้วยภาษาโปรล็อก

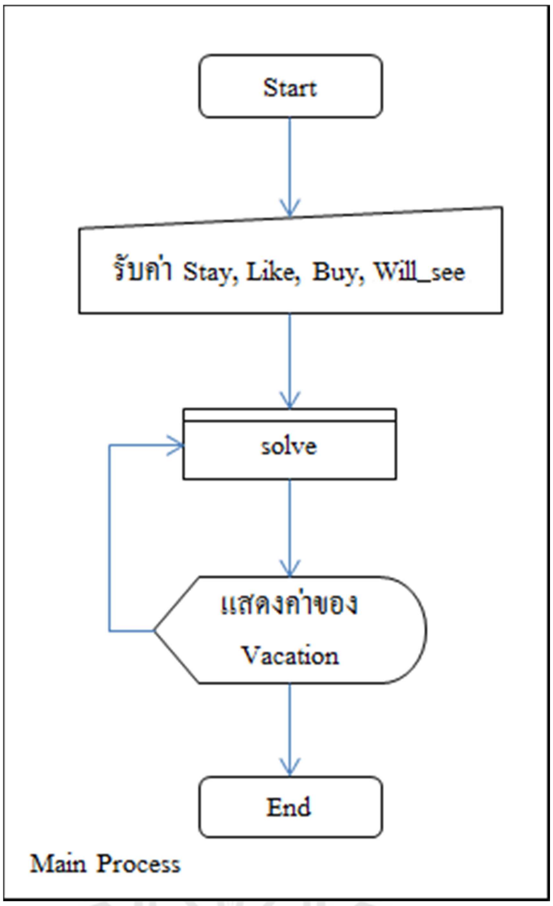
จากข้อมูลในรูปที่ 3.5 จะเห็นว่าขอบเขตของตัวแปรเป็นประเภทข้อความ ดังนั้น ไลบรารี่ที่จะใช้ในการพัฒนาฐานความรู้ควรที่จะเป็น sd (Symbolic Domain) ซึ่งใช้ในการแก้ปัญหาประเภทตัวอักษร ข้อความ หรือสัญลักษณ์ และข้อมูลส่วนนี้มีลักษณะที่สอดคล้องกับเงื่อนไขบังคับ (Constraint) มีความเป็นไปได้ที่จะนำข้อมูลส่วนนี้ไปเขียนที่ส่วนของเงื่อนไขบังคับ โดยกำหนดให้ขอบเขตของตัวแปร ดังแสดงในรูปที่ 3.6 ส่วนของการกำหนดขอบเขตของตัวแปร และตั้งเงื่อนไขบังคับในการแก้ปัญหาให้สอดคล้องกับรูปแบบเดิม และเมื่อทดสอบฐานความรู้ที่พัฒนาขึ้นใหม่แล้วต้องได้คำตอบตรงกันกับฐานความรู้เดิมที่ใช้เป็นต้นแบบในการพัฒนา ซึ่งผู้วิจัยได้ทำการทดลองพัฒนาด้วยภาษาโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับที่เรียกใช้ฐานความรู้ vacation ที่อยู่ในรูปลักษณ์ของเงื่อนไขบังคับการแทนค่าของตัวแปร โดยแสดงตัวอย่างโปรแกรมในรูปที่ 3.6



```
test4 - Notepad
File Edit Format View Help
:-lib(sd).
solve(bangkok,Stay,Like,_,_):- Stay&='building',Like&='traffic_jam'.
solve(phuket,Stay,Like,_,_):- Stay&='resort',Like&='beach'.
solve(korat,_,_,_,Will_see):- Will_see&='ya_mo'.
solve(korat,_,Like,_,_):- Like&='somtum'.
solve(chiangrai,Stay,_,Buy,_):- Stay&='resort',Buy&='tea'.
solve(chiangmai,Stay,_,Buy,_):- Stay&='building',Buy&='orange'.
solve(chiangmai,_,_,_,Will_see):- Will_see&='lin_ping'.
es_stay_like_buy_willsee(Stay,Like,Buy,Will_see):-
        Stay &:: [building,resort],
        Like &:: [traffic_jam,beach,somtum],
        Buy &:: [tea,orange],
        Will_see &:: [ya_mo,lin_ping],
        alldifferent([Stay,Like,Buy,Will_see]),
        solve(Vacation,Stay,Like,Buy,Will_see),
        writeln(Vacation).
```

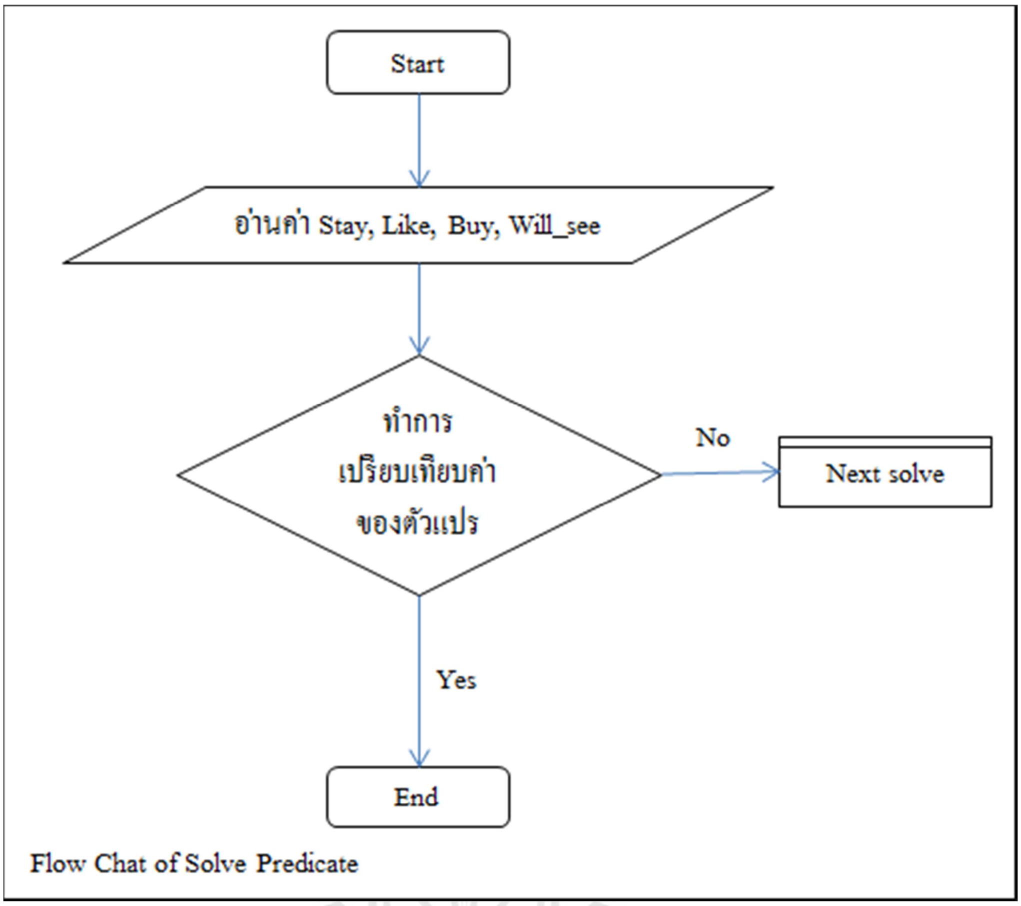

รูปที่ 3.6 ตัวอย่างการทดลองสร้างฐานความรู้ที่ใช้การเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ

จากรูปที่ 3.6 ผู้วิจัยได้ออกแบบให้โปรแกรมรับค่าของตัวแปร 4 ค่า คือ Stay, Like, Buy และ Will_see จากผู้ใช้ แล้วส่งค่าไปที่เพรดิเคต ที่เรียกใช้เงื่อนไขบังคับ คือ solve ซึ่งจะทำการค้นหาคำตอบจากเงื่อนไขบังคับที่มีรายละเอียดระบุความสัมพันธ์ระหว่างข้อมูลในฐานความรู้อย่างชัดเจน จากนั้นโปรแกรมก็จะแสดงคำตอบออกทางหน้าจอ ผังงานของโปรแกรมนี้สามารถแสดงได้ตามรูปที่ 3.7 และ 3.8



รูปที่ 3.7 แสดงผังงานของระบบหลัก

จากรูปที่ 3.7 แสดงให้เห็นรายละเอียดโดยรวมว่าเมื่อเริ่มโปรแกรม ได้ทำการรับค่าของตัวแปรเข้ามาจากนั้นทำการค้นหาคำตอบด้วยการส่งค่าไปที่ solve ซึ่งเป็นเพรดิเคต ที่ผู้วิจัยเขียนขึ้นเองสำหรับจำลองกระบวนการค้นหาคำตอบในฐานความรู้ และเมื่อได้คำตอบก็จะทำการแสดงค่าของตัวแปร Vacation ออกมาทางหน้าจอแสดงผลลัพธ์ และทำการวนซ้ำจนกว่าจะไม่พบคำตอบอื่นอีกจึงจบโปรแกรม รายละเอียดโดยรวมของ solve จะแสดงให้เห็นในรูปที่ 3.8



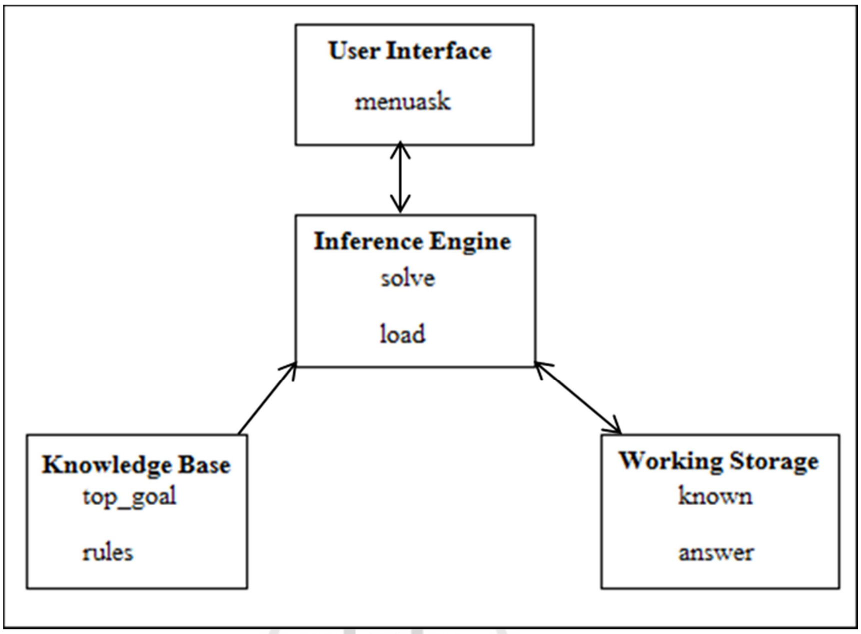
รูปที่ 3.8 แสดงผังงานของการจัดการข้อมูลในเงื่อนไขบังคับ

จากรูปแสดงให้เห็นว่าส่วนนี้ได้ทำการรับค่าของตัวแปรที่ผู้ใช้ป้อนเข้ามา แล้วทำการเปรียบเทียบกับเงื่อนไขที่ตั้งไว้แล้ว ถ้าตรงตามเงื่อนไขก็จะทำการส่งค่าของ Vacation กลับไปที่กระบวนการหลัก แต่ถ้าไม่ตรงตามเงื่อนไขก็จะทำการตรวจสอบเงื่อนไขถัดไป

3.1.3 การออกแบบเปลือกระบบผู้เชี่ยวชาญในรูปแบบของการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ

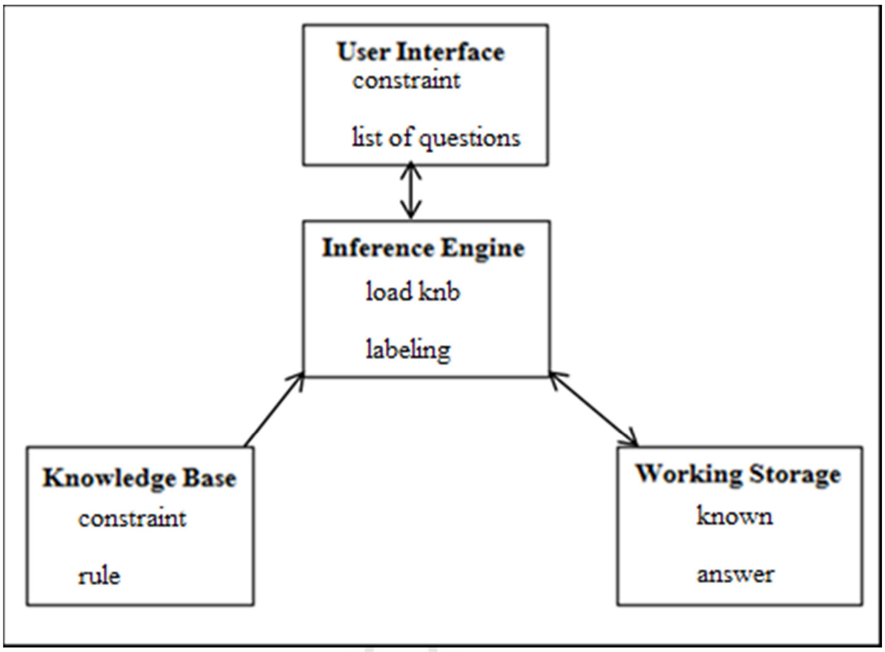
เนื่องจากโปรแกรม ECLiPSe 6.0 ที่จะใช้ในการพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ มีลักษณะการเขียนของโปรแกรมบนพื้นฐานของโปรแกรมเชิงตรรกะภาษาโปรล็อก ดังนั้นจึงทำการการศึกษาเปลือกระบบผู้เชี่ยวชาญที่เขียนด้วยโปรแกรมเชิงตรรกะภาษาโปรล็อกทั้งในด้านของโครงสร้าง และกระบวนการทำงาน เพื่อใช้เป็นแนวทางในการออกแบบเปลือกระบบผู้เชี่ยวชาญในรูปแบบของการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ และสามารถทำการวัดประสิทธิภาพของโปรแกรมทั้งสองรูปแบบภายใต้

องค์ประกอบที่คล้ายคลึงกัน โดยรูปที่ 3.9 จะแสดงโครงสร้างของเปลือกระบบผู้เชี่ยวชาญ ที่พัฒนาในงานวิจัยการประมวลผลหลังกระบวนการทำเหมือง (นิตยา เกิดประสพ, 2552) ซึ่งพัฒนาด้วยโปรแกรมเชิงตรรกะโดยใช้ภาษาโปรล็อก



รูปที่ 3.9 โครงสร้างของเปลือกระบบผู้เชี่ยวชาญ (นิตยา เกิดประสพ, 2552)

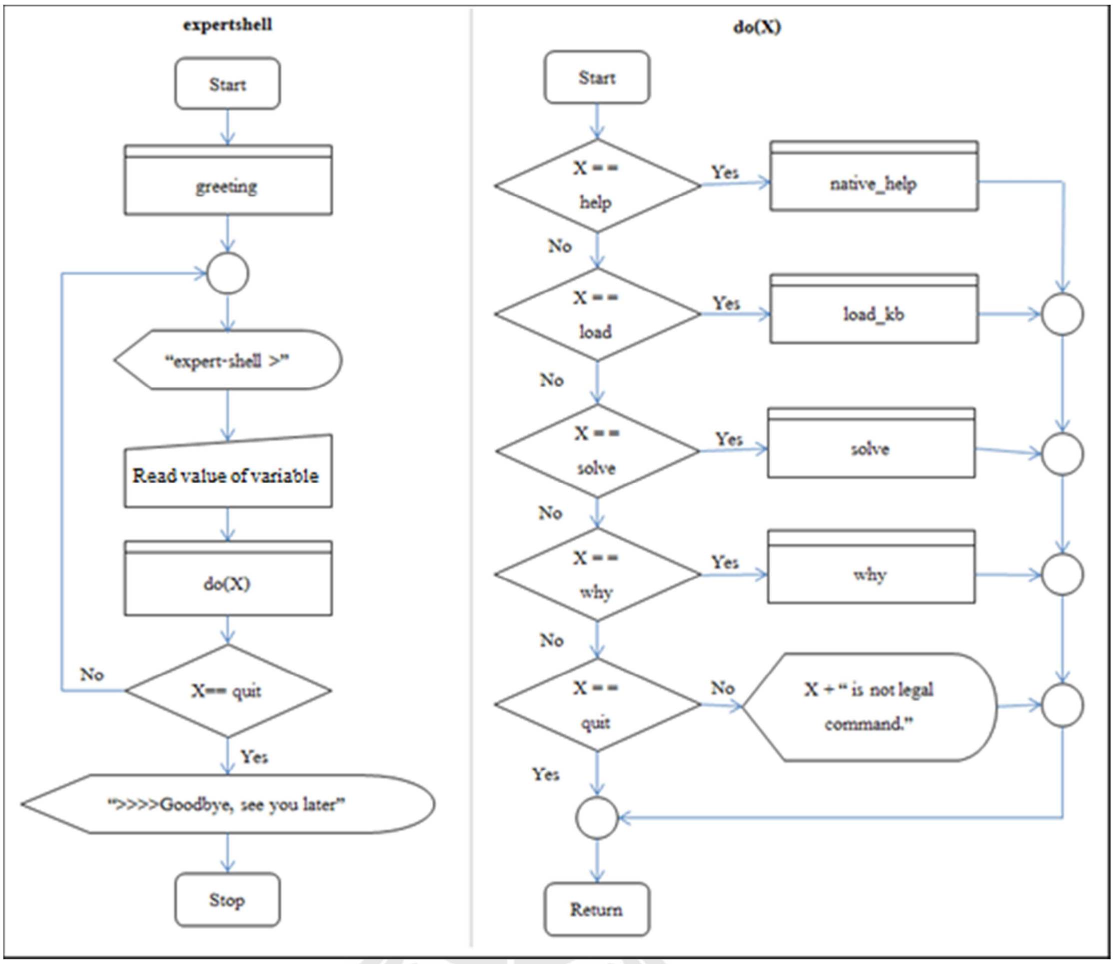
จากรูปที่ 3.9 เป็นโครงสร้างของเปลือกระบบผู้เชี่ยวชาญที่เขียนด้วยภาษาโปรล็อก โดยแยกตามองค์ประกอบที่สำคัญของเปลือกระบบผู้เชี่ยวชาญ เช่น ฐานความรู้ (Knowledge Base) และกลไกอนุมาน (Inference Engine) ซึ่งได้แสดงส่วนประกอบหลักของส่วนติดต่อกับผู้ใช้ และข้อมูลแบบพลวัต (Dynamic) ที่จะมีการจัดเก็บขณะที่ระบบทำงาน เช่น คำตอบจากผู้ใช้สำหรับแต่ละคำถาม และข้อมูลความรู้ที่ได้จากการสอบถามกับผู้ใช้ จากการศึกษาโครงสร้างดังกล่าวมีบางส่วนที่จะต้องปรับแก้ให้เข้ากับการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ เช่น ส่วนของฐานความรู้จากตอนแรกจะเป็นข้อมูลที่เขียนตามโครงสร้างของโปรแกรมภาษาโปรล็อกธรรมดา เมื่อเขียนด้วยการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ จะอยู่ในลักษณะของเงื่อนไขบังคับ (Constraint) การเรียกใช้ฐานความรู้ร่วมกับเปลือกระบบผู้เชี่ยวชาญก็อาจจะมีรูปแบบที่แตกต่างออกไป ในรูปที่ 3.10 จะแสดงโครงสร้างของเปลือกระบบผู้เชี่ยวชาญที่เขียนด้วยภาษาโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ



รูปที่ 3.10 โครงสร้างของเปลือกระบบผู้เชี่ยวชาญที่เขียนด้วยภาษาโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ

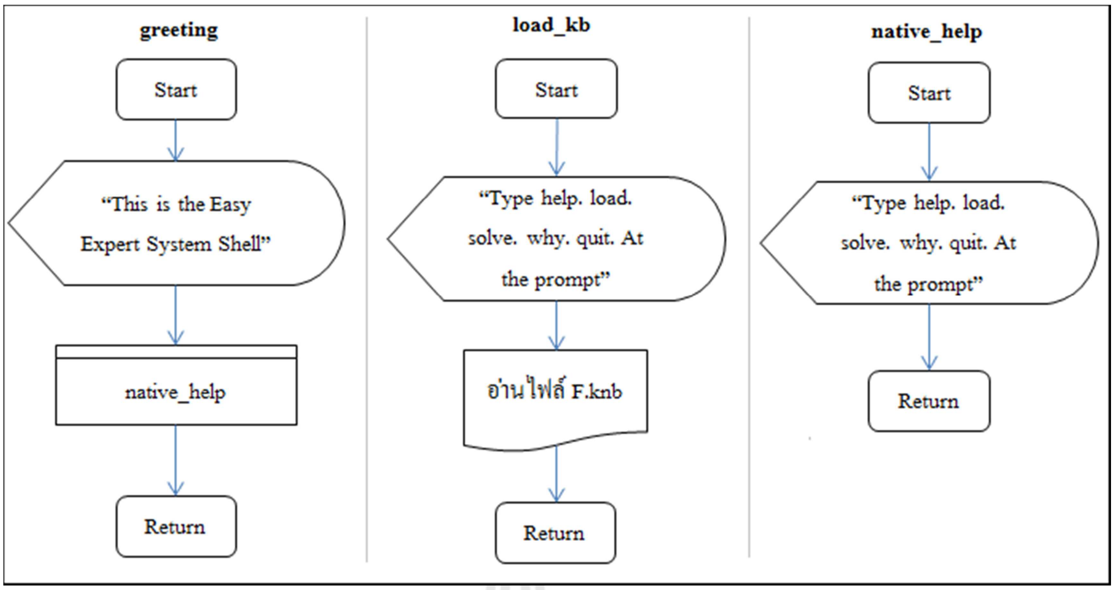
จากรูปที่ 3.10 เริ่มจากส่วนติดต่อกับผู้ใช้ (User Interface) ข้อมูลที่จะแสดงในส่วนนี้เกิดจากกระบวนการดึงความรู้ภายในกลไกอนุมานดึงมาจากฐานความรู้ ซึ่งในเปลือกระบบที่ออกแบบใหม่นี้ ฐานความรู้จะอยู่ในรูปแบบของเงื่อนไขบังคับ (Constraint) ในส่วนของกลไกอนุมานมีการเรียกใช้ฐานความรู้ และกระบวนการค้นหาคำตอบ ซึ่งในการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับมีคำสั่งพื้นฐานที่เรียกว่า labeling ช่วยในการค้นหา

หลังจากศึกษาโครงสร้างของเปลือกระบบผู้เชี่ยวชาญก็ทำการวิเคราะห์กระบวนการทำงานของเปลือกระบบผู้เชี่ยวชาญดังแสดงในรูปที่ 3.11 ถึงรูปที่ 3.15 ซึ่งเป็นกระบวนการทำงานของเปลือกระบบผู้เชี่ยวชาญ ที่พัฒนาในงานวิจัยการประมวลผลหลังกระบวนการทำเหมือง (นิตยา เกิดประสพ, 2552) ด้วยโปรแกรมภาษาโปรล็อก



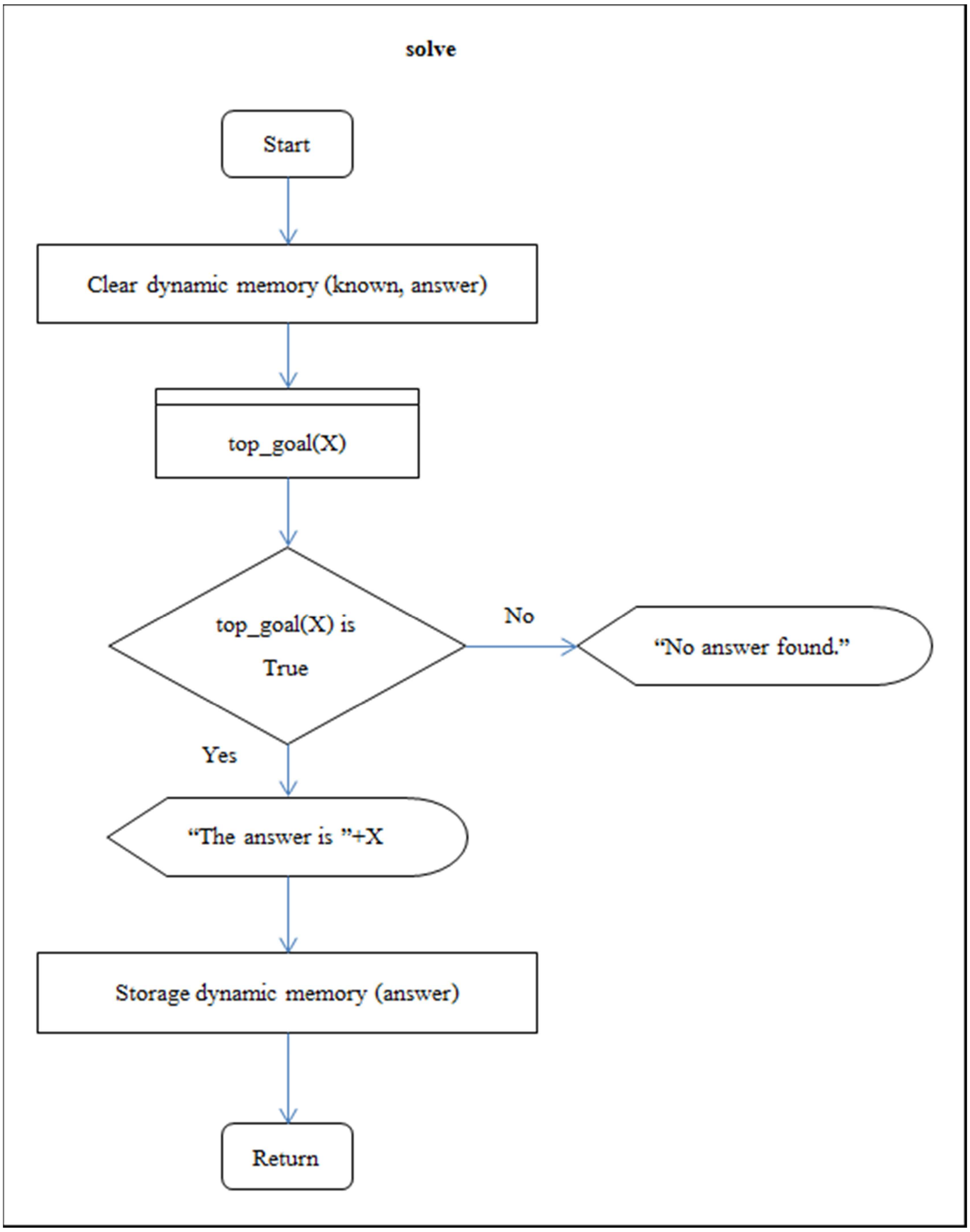
รูปที่ 3.11 ผังงานทางซ้ายแสดงกระบวนการทำงานของระบบเมื่อป้อนคำสั่ง expertshell และผังงานทางขวาแสดงกระบวนการทำงานของเพรดิเคต ชื่อ do(X)

จากรูปที่ 3.11 เมื่อผู้ใช้ป้อนคำสั่ง expertshell โปรแกรมก็จะเรียกใช้ Predicate ที่ชื่อ greeting เพื่อแสดงข้อความต้อนรับ ในส่วนของ expertshell หน้าที่หลักคือการรอรับคำสั่งว่า จะให้ระบบทำอะไร แล้วส่งคำสั่งไปยัง do(X) เพรดิเคต do(X) จะมีหน้าที่ในการเลือกการทำงานของโปรแกรมว่าจะให้ทำอะไร เช่น เมื่อผู้ป้อนคำว่า help เพรดิเคตก็จะทำการเรียกใช้ native_help ให้ทำงาน ซึ่งเปลือกระบบนี้ไม่หยุดทำงานจนกว่าจะป้อนคำสั่ง quit ให้แก่ระบบ



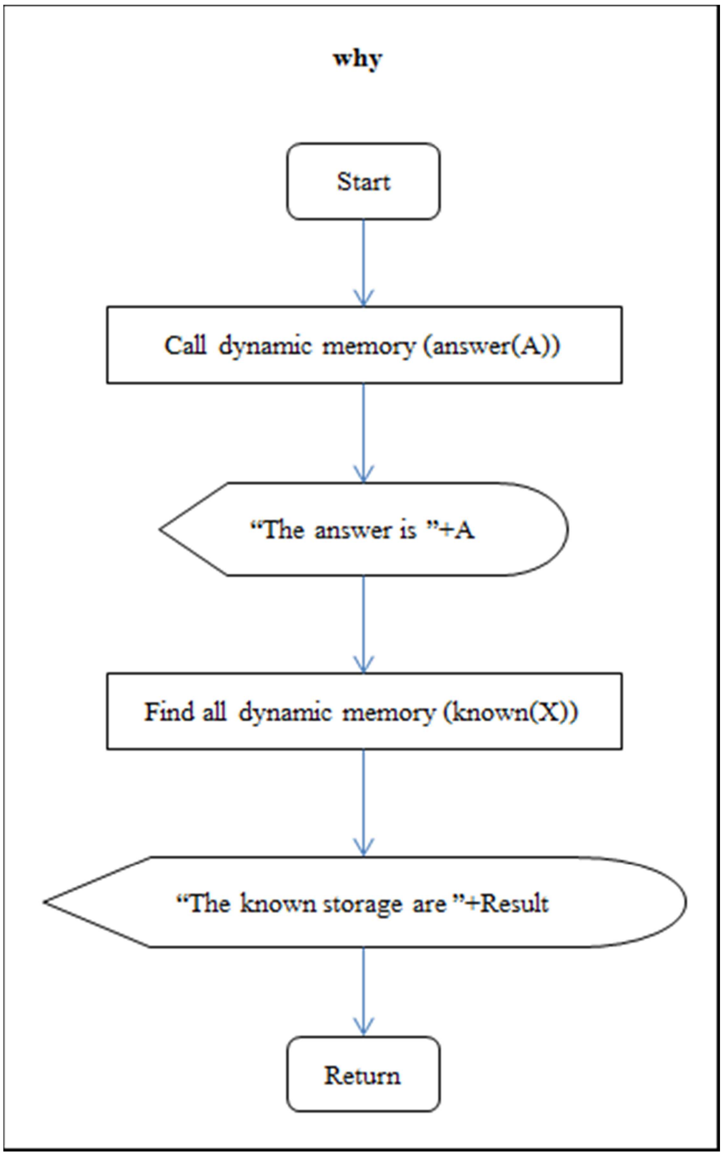
รูปที่ 3.12 ผังงานทางซ้ายแสดงกระบวนการทำงานของเพรดิเคต ชื่อ greeting ผังงานตรงกลาง แสดงกระบวนการทำงานของเพรดิเคต ชื่อ load_kb และผังงานทางขวาแสดงกระบวนการทำงานของเพรดิเคต ชื่อ native_help

จากรูปที่ 3.12 greeting ทำหน้าที่แสดงข้อความต้อนรับแล้วเรียกใช้ native_help ซึ่งจะแสดงคำสั่งที่สามารถใช้ทำงานกับระบบ ส่วน load_kb มีหน้าที่อ่านข้อมูลฐานความรู้เข้ามารวมอยู่ในเปลือกระบบผู้เชี่ยวชาญ ซึ่งตามคุณสมบัติของเปลือกระบบผู้เชี่ยวชาญทำให้คำสั่ง load_kb มีบทบาทสำคัญทำให้ผู้ใช้สามารถเปลี่ยนฐานความรู้ที่ต้องการค้นได้



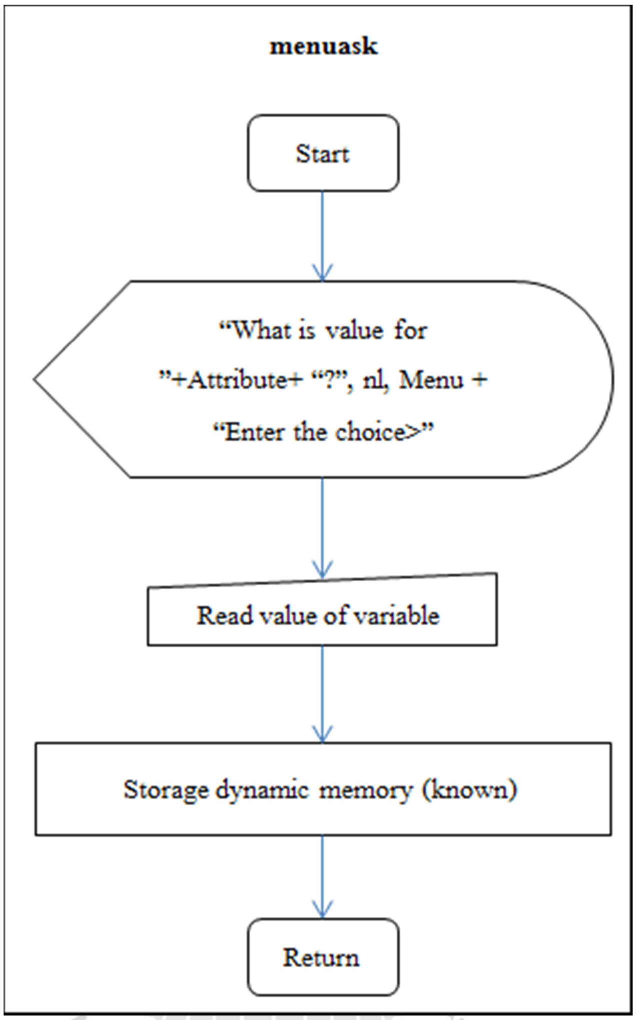
รูปที่ 3.13 ผังงานแสดงกระบวนการทำงานของเพรดิเคตชื่อ solve

จากรูปที่ 3.13 เรียกได้ว่าเป็น ผังงานของกลไกอนุมาน โดยเริ่มจากการล้างค่าข้อมูลแบบพลวัต (Dynamic) ที่อาจเก็บไว้ในการทำงานครั้งก่อน จากนั้นทำการอนุมานความรู้แล้วแสดงคำตอบแก่ผู้ใช้ระบบ แล้วทำการเก็บคำตอบไว้ใช้กับคำสั่งอื่น เช่น why



รูปที่ 3.14 ผังงานแสดงกระบวนการทำงานเพรดิเคตชื่อ why

จากรูปที่ 3.14 การทำงานจะเริ่มจากการแสดงคำตอบที่ได้จากการอนุมานความรู้และแสดงเหตุผลสนับสนุนด้วยการเรียกความรู้ที่ได้จากการโต้ตอบกับผู้ใช้ ส่วนนี้ถือว่าเป็นส่วนของการอธิบายความรู้ ซึ่งทำหน้าที่สร้างความน่าเชื่อถือในคำตอบที่ค้นได้ของระบบผู้เชี่ยวชาญ



รูปที่ 3.15 ผังงานแสดงกระบวนการทำงานของเพรดิเคตชื่อ menuask

จากรูปที่ 3.15 เป็นกระบวนการทำงานของส่วนติดต่อกับผู้ใช้ สำหรับเปลือกระบบผู้เชี่ยวชาญในส่วนนี้จะมีการออกแบบคำถามให้ถามในรูปแบบกว้าง ๆ ไม่เฉพาะเจาะจง และคอยรับคำตอบของผู้ใช้มาเก็บไว้ในส่วนอนุมานความรู้ โดยตัวอย่างโค้ดในโปรแกรมภาษาโปรล็อกของเพรดิเคตชื่อ menuask นั้นแสดงในรูปที่ 3.16

```
menuask(Pred, Value, Menu) :- menuask(Pred,Menu), known([Pred,Value]), !.
menuask(Pred,_) :- known([Pred,_]), !.
menuask(Attribute,Menu) :-   write('What is the value for '), write(Attribute), writeln('?'),
                             writeln(Menu), write('Enter the choice > '), read(V),writeln(V),
                             asserta(known([Attribute,V])).
```

รูปที่ 3.16 แสดงเพรดิเคตชื่อ menuask จากเปลือกระบบผู้เชี่ยวชาญอย่างง่ายที่เขียนด้วยภาษาโปรล็อก

```
What is the value for stay?
[building, resort]
Enter the choice > resort
What is the value for like?
[traffic_jam, beach, somtam]
Enter the choice > beach
The answer is phuket
expert-shell>
The answer is phuket
The known storage are
[[like, beach], [stay, resort]]
expert-shell>
>>>>Goodbye, see you later<<<<
```

รูปที่ 3.17 ตัวอย่างผลลัพธ์ที่ได้จากเปลือกระบบผู้เชี่ยวชาญที่เขียนด้วยภาษาโปรล็อก

ในรูปที่ 3.17 เป็นผลลัพธ์ที่ได้จากการรันโปรแกรมเปลือกระบบผู้เชี่ยวชาญภาษาโปรล็อกและใช้ฐานความรู้ vacation ที่พัฒนาด้วยภาษาโปรล็อกเป็นฐานความรู้ของระบบ

จากการศึกษาโปรแกรมและการวิเคราะห์ออกมาเป็นผังงานของส่วนต่าง ๆ ช่วยให้กำหนดรูปแบบและตำแหน่งในการจัดวางองค์ประกอบต่าง ๆ ได้ชัดเจนขึ้น รูปแบบการเขียนโปรแกรมของภาษาโปรล็อก และภาษา CLP มีการจัดการที่แตกต่างกัน ดังนั้นในการออกแบบเปลือกระบบผู้เชี่ยวชาญด้วยการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับจะเน้นไปที่การวางแผนการจัดวางองค์ประกอบในส่วนต่าง ๆ ของโค้ดโปรแกรมโดยจะสรุปไว้ในตารางที่ 3.1 ดังนี้

ตารางที่ 3.1 ตารางแสดงการวางแผนจัดวางองค์ประกอบสำหรับออกแบบเปลือกระบบผู้เชี่ยวชาญที่เขียนด้วยภาษา CLP

ส่วนประกอบของโปรแกรม	องค์ประกอบ	หมายเหตุ
Library ของโปรแกรม	lib(sd)	เลือกใช้ sd (Symbolic Domain) เพราะเป็นการทำงานกับข้อความ หรือสัญลักษณ์
ฟังก์ชันของโปรแกรม	do(C) help load solve why menuask	ประกอบไปด้วยเพรดิเคตต่าง ๆ เช่น do(C) โดยที่ C คือชื่อเพรดิเคตที่ทำงานอยู่ในส่วนเดียวกันนี้ เช่น help, load, solve, why และเพรดิเคตสำหรับโต้ตอบกับผู้ใช้ คือ menuask และเนื่องจากการจัดวางองค์ประกอบให้ load อยู่ในส่วนนี้ ทำให้เมื่ออ่านไฟล์ฐานความรู้แล้ว การทำงานก็จะอยู่ในส่วนนี้เช่นกัน
ส่วนเริ่มการทำงาน	expertshell	เรียกใช้ expertshell โดยไม่ต้องระบุค่าใด ๆ
ส่วนประกาศตัวแปร	C (Command) Stay Like Buy Will_see	ในส่วนนี้จะประกาศขอบเขตของตัวแปรประกอบด้วย C หมายถึง คำสั่งที่ผู้ใช้ระบุว่าต้องการให้ระบบทำอะไร เช่น help, load, solve และ why และส่วนนี้ประกาศขอบเขตของตัวแปรที่เป็นคำตอบของแต่ละคำถาม เช่น Stay, Like, Buy และ Will_see
ส่วนโปรแกรมหลัก	repeat print ask command read(C) do(C) print exit program	กำหนดให้มีการวนซ้ำด้วยคำสั่ง repeat ซึ่งจะหยุดก็ต่อเมื่อ C มีค่าเป็น quit โดยส่วนนี้จะประกอบไปด้วยข้อความถามคำสั่งที่ผู้ใช้ต้องการให้ระบบทำ ซึ่งจะส่งค่า C ไปให้ do(C) เลือกการทำงาน

จากตารางแสดงถึงส่วนประกอบภายในโปรแกรมเปลือกระบบผู้เชี่ยวชาญ ส่วนฐานความรู้ของเปลือกระบบผู้เชี่ยวชาญนี้ผู้วิจัยได้ทำการปรับเปลี่ยนให้อยู่ในรูปแบบของเงื่อนไขบังคับของโปรแกรมระบบ โดยที่เงื่อนไขบังคับนี้สามารถปรับเปลี่ยนโดเมนความรู้ได้อย่างอิสระ

3.1.4 การพัฒนาเปลือกระบบผู้เชี่ยวชาญและฐานความรู้ในรูปแบบวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ

ในงานวิจัยนี้ได้ทำการศึกษาและพัฒนาเปลือกระบบผู้เชี่ยวชาญและฐานความรู้ขึ้นมาทั้งหมด 4 รูปแบบ ดังรายละเอียดต่อไปนี้

รูปแบบที่ 1 เป็นการพัฒนาโดยการปรับเปลี่ยนรูปแบบของฐานความรู้ทั้งหมดให้อยู่ในรูปแบบวิธีเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ ดังแสดงในรูปที่ 3.18 และปรับเปลี่ยน menuask ให้กระชับขึ้นด้วยการตัดส่วนที่ไม่จำเป็นออก แสดงในรูปที่ 3.19

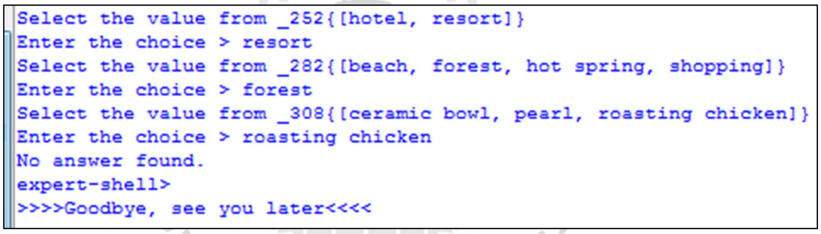
```
top_goal(Vout) :-
        Sin &:: ['hotel','resort'],                                  %Menu Domain start%
        Lin &:: ['forest','hot spring','shopping','beach'],
        Bin &:: ['pearl','ceramic bowl','roasting chicken'],
        Win &:: ['stone castle','coal mine','coral'],                %Menu Domain end%
        Vout &:: ['Phuket','Bangkok','Korat','Lumpang'],              %Goal Domain %
        P &:: ['resort','beach','pearl','coral'],               %Knowledge Domain start%
        K &:: ['resort','forest','roasting chicken','stone castle'],
        B &:: ['hotel','shopping'],
        L &:: ['hotel','hot spring','ceramic bowl','coal mine'],  %Knowledge Domain end%
        menuask(Sin),                                                 %Constraint Start%
        (Sin&=P ->
                (menuask(Lin),Lin&=P ->
                                (menuask(Bin),Bin&=P ->
                                                (menuask(Win),Win&=P -> Vout='Phuket'
                                                )
                                )
                ;
                        (menuask(Lin),Lin&=L ->
                                (menuask(Bin),Bin&=L ->
                                                (menuask(Win),Win&=L -> Vout='Lumpang'
                                                )
                                )
                ;
                                (Lin&=B -> Vout='Bangkok'
                                )
                )
).                                                                  % Constraint end%
```

รูปที่ 3.18 ตัวอย่างฐานความรู้โดยใช้ข้อมูลแบบง่าย ๆ เพื่อออกแบบในรูปแบบที่ 1

จากรูปที่ 3.18 จะเห็นว่ามีการกำหนดขอบเขตของตัวแปรแทนค่าที่เป็นไปได้ทั้งหมดของตัวแปรแต่ละตัว โดยแบ่งเป็น 3 กลุ่มหลัก ๆ คือ กลุ่มของตัวแปรสำหรับรับค่าจากผู้ใช้ (Sin Lin Bin และ Win) กลุ่มของตัวแปรสำหรับค่าเป้าหมายที่เราต้องการ (Vout) และ กลุ่มของตัวแปรที่ระบุค่าสำหรับประกอบการค้นหาคำตอบที่เป็นไปได้ เพื่อใช้ในการแทนค่าเป้าหมายที่ต้องการ (P K B และ L) จากนั้นในส่วนของเงื่อนไขบังคับเบื้องต้นได้ออกแบบในลักษณะจำเพาะกับข้อมูลก่อน โดยใช้รูปแบบของ “A -> B ; C” หมายถึง “if A then B else C” มาช่วยในการตั้งเงื่อนไขเพื่อกำหนดค่าเป้าหมายในกรณีต่าง ๆ

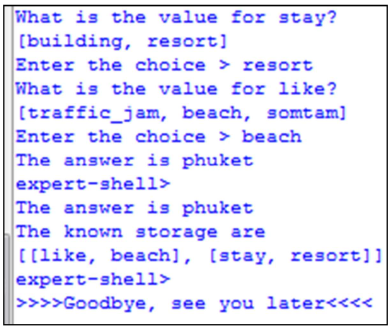
```
:-lib(sd).
menuask(Value) :-  write('Select the value from '),writeln(Value),
                   write('Enter the choice > '), read(Value),writeln(Value),
                   asserta(known(Value)).
```

รูปที่ 3.19 แสดงไลบรารี่ที่จำเป็นคือ lib(sd) และ menuask ที่ถูกดัดแปลงให้กระชับขึ้น



```
Select the value from _252{[hotel, resort]}
Enter the choice > resort
Select the value from _282{[beach, forest, hot spring, shopping]}
Enter the choice > forest
Select the value from _308{[ceramic bowl, pearl, roasting chicken]}
Enter the choice > roasting chicken
No answer found.
expert-shell>
>>>>Goodbye, see you later<<<<
```

(a) Expert Shell พัฒนาด้วย CLP รูปแบบที่ 1



```
What is the value for stay?
[building, resort]
Enter the choice > resort
What is the value for like?
[traffic_jam, beach, somtam]
Enter the choice > beach
The answer is phuket
expert-shell>
The answer is phuket
The known storage are
[[like, beach], [stay, resort]]
expert-shell>
>>>>Goodbye, see you later<<<<
```

(b) Expert Shell พัฒนาด้วย Prolog

รูปที่ 3.20 แสดงผลลัพธ์ที่ได้จากระบบที่พัฒนาตามรูปแบบที่ 1

จากผลลัพธ์ในรูปแบบ CLP เมื่อเทียบกับฐานความรู้ในรูปแบบของโปรล็อก (รูปที่ 3.19) จะเห็นว่าโปรแกรมยังมีส่วนที่ผิดพลาด คือ คำตอบที่ได้รับจากระบบเป็น “No answer found.” แทนที่จะเป็น “Korat” สาเหตุเกิดจากวิธีที่ใช้ในการเปรียบเทียบค่าของ “ตัวแปรที่รับจากผู้ใช้” กับ “ตัวแปรที่ระบุค่าสำหรับประกอบการค้นหาคำตอบที่เป็นไปได้” ด้วยเครื่องหมาย “&=” นั้น ทำให้ค่าของตัวแปรที่ระบุค่าสำหรับประกอบการค้นหาคำตอบที่เป็นไปได้ถูกกำหนดค่าให้ “เท่ากันกับ” ค่าของตัวแปรที่รับจากผู้ใช้ และไม่สามารถนำตัวแปรที่ระบุค่าสำหรับประกอบการค้นหาคำตอบที่เป็นไปได้นี้ไปตรวจสอบค่ากับตัวแปรที่รับจากผู้ใช้ตัวอื่นได้อีก

<u>รูปแบบที่ 2</u> เป็นการพัฒนาโดยเปลี่ยนวิธีตรวจสอบเงื่อนไขใหม่เป็นการตรวจสอบว่าข้อมูลที่ได้รับมานั้นเป็น “สับเซต” ของค่าที่เป็นได้ของแต่ละเป้าหมายหรือไม่แทน และออกแบบตัวกระทำสำหรับใช้ตรวจสอบเงื่อนไขขึ้นใหม่ เพื่อให้ง่ายต่อการออกแบบฐานความรู้ โดยกลุ่มตัวแปรที่ระบุค่าสำหรับประกอบการค้นหาคำตอบที่เป็นไปได้จะเป็นการกำหนดค่าเป็นลิสต์แทนการกำหนดในลักษณะของโดเมน

```
% more Code Here%
P &= ['resort','beach','pearl','coral'],
K &= ['resort','forest','roasting chicken','stone castle'],
B &= ['hotel','shopping'],
L &= ['hotel','hot spring','ceramic bowl','coal mine'],
        type(Sin),
        (if ord_subset([Sin],B)
        then
                (type(Lin),
                if ord_subset([Sin,Lin],B)
                then
                        Vout &= 'Bangkok'
                else
%more Constraint Here%
end).
```

รูปที่ 3.21 แสดงตัวอย่างบางส่วนของฐานความรู้ในรูปแบบที่ 2

```
:- lib(ordset).
:- op(701,fx,if).
:- op(702,yfx,then).
:- op(703,yfx,else).
:- op(704,yf,end).
if A then B end:- A -> B.
if A then B else C end:- A ->B;C.
```

รูปที่ 3.22 แสดงไลบรารี่ที่จำเป็น และวิธีออกแบบตัวกระทำ if-then-else

```
Select the value from _254{[hotel, resort]}
Enter the choice > resort
Select the value from _284{[beach, forest, hot spring, shopping]}
Enter the choice > forest
Select the value from _310{[ceramic bowl, pearl, roasting chicken]}
Enter the choice > roasting chicken
Select the value from _336{[coal mine, coral, stone castle]}
Enter the choice > stone castle
The answer is Korat
expert-shell>
The answer is Korat
The known storage are
[stone castle, roasting chicken, forest, resort]
expert-shell>
>>>>Goodbye, see you later<<<<
```

รูปที่ 3.23 แสดงผลลัพธ์ที่ได้จากระบบที่พัฒนาตามรูปแบบที่ 2

จากรูปที่ 3.23 พบว่ารูปแบบที่ 2 นั้นสามารถแสดงคำถามและคำตอบได้อย่างถูกต้องตามที่กำหนดไว้ แต่เนื่องจากเป็นการพัฒนาต่อยอดจากรูปแบบที่ 1 ซึ่งสามารถใช้ได้กับข้อมูลชุดนี้เท่านั้น ผู้วิจัยจะต้องออกแบบการแสดงคำถามและคำตอบของระบบใหม่ให้สามารถใช้ได้กับข้อมูลทั่ว ๆ ไป ตามวัตถุประสงค์ที่กล่าวไว้ข้างต้น จึงเกิดการพัฒนาในรูปแบบที่ 3

รูปแบบที่ 3 เป็นการพัฒนาให้ระบบสามารถรองรับฐานความรู้ใด ๆ ได้ โดยออกแบบฐานความรู้ใหม่ ในส่วนของการตรวจสอบเงื่อนไข และออกแบบเพรดิเคตใหม่ชื่อ liken ให้มีการทำงานสัมพันธ์กับฐานความรู้ใหม่

```
liken([_],[Db],[Att],Flag):-
        known([Att,Old]),!,
        (if Old==Db then
                Flag = true
        else
                Flag = fail
        end).
```

```
liken([_|Tn],[Db|Td],[Att|Ta],Flag):-
        known([Att,Old]),!,
        (if Old==Db then
                liken(Tn,Td,Ta,Flag)
        else
                Flag = fail
        end).
```

```
liken([In],[Db],[Att],Flag):-
        \+known([Att,_]),
        menuask(Att,In),!,
        (if In==Db then
                Flag = true
        else
                Flag = fail
        end).
```

```
liken([In|Tn],[Db|Td],[Att|Ta],Flag):-
        \+known([Att,_]),
        menuask(Att,In),!,
        (if In==Db then
                liken(Tn,Td,Ta,Flag)
        else
                Flag = fail
        end).
```

รูปที่ 3.24 แสดงเพรดิเคตชื่อ liken ที่ออกแบบใหม่

โดยเพรดิเคตที่ออกแบบใหม่นี้แบ่งออกเป็น 4 เพรดิเคตย่อยสำหรับรองรับเหตุการณ์ที่อาจจะเกิดขึ้น 4 เหตุการณ์ คือ

1. กรณีที่คำถามนั้นเป็นคำถามสุดท้ายและได้ถามไปแล้ว ให้ตรวจสอบว่าค่าที่บันทึกไว้ในเพรดิเคต know ตรงกับในฐานความรู้หรือไม่ ถ้าตรงให้ค่า Flag = true แต่ถ้าไม่ตรงให้ค่า Flag = fail (รูปที่ 3.24 ซ้าย-บน)

2. กรณีที่คำถามนั้นไม่เป็นคำถามสุดท้ายและได้ถามไปแล้ว ให้ตรวจสอบว่าค่าที่บันทึกไว้ในเพรดิเคต know ตรงกับในฐานความรู้หรือไม่ ถ้าตรงให้เรียกเพรดิเคต like ซ้ำ แต่ถ้าไม่ตรงให้ค่า Flag = fail (รูปที่ 3.24 ขวา-บน)

3. กรณีที่คำถามนั้นเป็นคำถามสุดท้ายและไม่เคยถามมาก่อน ให้เรียกเพรดิเคต menuask แล้วตรวจสอบว่าค่าที่รับมานั้นตรงกับในฐานความรู้หรือไม่ ถ้าตรงให้ค่า Flag = true แต่ถ้าไม่ตรงให้ค่า Flag = fail (รูปที่ 3.24 ซ้าย-ล่าง)

4. กรณีที่คำถามนั้นไม่เป็นคำถามสุดท้ายและไม่เคยถามมาก่อน ให้เรียกเพรดิเคต menuask แล้วตรวจสอบว่าค่าที่รับมานั้นตรงกับในฐานความรู้หรือไม่ ถ้าตรงให้เรียกเพรดิเคต like ซ้ำแต่ถ้าไม่ตรงให้ค่า Flag = fail (รูปที่ 3.24 ขวา-ล่าง)

```
%more Code Here%
            In2 = [Sin,Lin],
            In4 = [Sin,Lin,Bin,Win],
            Att2 = [stay,like],
            Att4 = [stay,like,buy,will_see],
(           if liken(In2,B,Att2,true) then Vout &= 'Bangkok' end;
            if liken(In4,L,Att4,true) then Vout &= 'Lumpang' end;
            if liken(In4,P,Att4,true) then Vout &= 'Phuket' end;
            if liken(In4,K,Att4,true) then Vout &= 'Korat' end).
```

รูปที่ 3.25 แสดงตัวอย่างลักษณะฐานความรู้ที่ใช้ได้กับ liken

```
Please select value of stay from  : _262{[hotel, resort]}
Your input is  : resort
Please select value of like from  : _292{[beach, forest, hot_spring, shopping]}
Your input is  : forest
Please select value of buy from  : _318{[ceramic_bowl, pearl, roasting_chicken]}
Your input is  : roasting_chicken
Please select value of will_see from  : _344{[coal_mine, coral, stone_castle]}
Your input is  : stone_castle
The answer is Korat
expert-shell>
The answer is Korat
The known storage are
[[stay, resort], [like, forest], [buy, roasting_chicken], [will_see, stone_castle]]
expert-shell>
>>>>Goodbye, see you later<<<<
```

รูปที่ 3.26 แสดงผลลัพธ์ที่ได้จากระบบที่พัฒนาตามรูปแบบที่ 3

ระบบจากรูปแบบที่ 3 สามารถแสดงผลลัพธ์ได้เช่นเดียวกันกับรูปแบบที่ 2 ตามตัวอย่างในรูปที่ 3.26 แต่จะมีรายละเอียดของแอททริบิวต์สำหรับตัวแปรรับค่าแต่ละตัวเพิ่มเข้ามา แต่เมื่อผู้วิจัยได้นำวิธีนี้ไปทดลองกับข้อมูลเดียวกันกับที่พัฒนาด้วยภาษาโปรล็อกพบว่าลำดับในการแสดงผลของระบบไม่เป็นไปตามที่คาดเดาไว้ และรูปแบบการทำงานของระบบซับซ้อนและมีการวนซ้ำที่เพรดิเคตชื่อ liken มากเกินไป ทำให้ไม่สามารถยอมรับได้ว่ารูปแบบที่ 3 นี้จะเป็นรูปแบบที่ดีที่สุดได้

<u>รูปแบบที่ 4</u> จากการวิจัยโดยทดลองปรับเปลี่ยนรูปแบบของฐานความรู้ใหม่ในรูปแบบทั้งสามที่กล่าวไปแล้วนั้น ทำให้ต้องเปลี่ยนแนวคิดในการออกแบบมาสนใจปรับเปลี่ยนที่ตัวเปลือกระบบผู้เชี่ยวชาญแทน ซึ่งจากการวิจัยพบว่ารูปแบบที่ 4 นี้มีการเปลี่ยนแปลงแค่ส่วนเล็ก ๆ ภายใน menuask เพียงอย่างเดียว และฐานความรู้ที่ใช้ก็ไม่จำเป็นต้องออกแบบใหม่ สามารถใช้รูปแบบเดียวกันกับฐานความรู้ของระบบที่พัฒนาด้วยภาษาโปรล็อกตามที่แสดงในรูปที่ 3.5

```
menuask(Pred,Value,Menu):- C&::Menu, menuask(Pred,C),  known([Pred,Value]),!.
menuask(Pred,_) :- known([Pred,_]),!.
menuask(Attribute,C) :-  write('What is the value for '), write(Attribute), writeln('?'),
                    writeln(C), write('Enter the choice > '), read(C),writeln(C),
                    asserta(known([Attribute,C])).
```

(a) เพรดิเคต menuask ในรูปแบบ CLP

```
menuask(Pred, Value, Menu) :- menuask(Pred,Menu), known([Pred,Value]), !.
menuask(Pred,_) :- known([Pred,_]), !.
menuask(Attribute,Menu) :-   write('What is the value for '), write(Attribute), writeln('?'),
                    writeln(Menu), write('Enter the choice > '), read(V),writeln(V),
                    asserta(known([Attribute,V])).
```

(b) เพรดิเคต menuask ในรูปแบบภาษาโปรล็อก

รูปที่ 3.27 แสดง menuask ที่ออกแบบด้วย (a) วิธีเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับเปรียบเทียบกับ (b) วิธีเชิงตรรกะที่เขียนด้วยภาษาโปรล็อก

เมื่อเปรียบเทียบโค้ดของ CLP และภาษาโปรล็อก จะเห็นความแตกต่างที่การส่งตัวแปร บรรทัดแรกในรูปที่ 3.27 (b) Menu ที่ส่งมานั้นถูกส่งต่อไปยังเพรดิเคต menuask/2 ส่วนบรรทัดแรกในรูปที่ 3.27 (a) Menu ที่ส่งมานั้นจะอยู่ในลักษณะของลิสต์ สามารถใช้กำหนดขอบเขตของตัวแปรตามวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับได้ และบรรทัดที่สี่ในรูปที่ 3.27 (a) จะเห็นว่าสามารถใช้ตัวแปรที่มีการกำหนดขอบเขตนั้นแสดงเมนูของคำตอบได้ และใช้เป็นตัวแปรรับค่าจากผู้ใช้ด้วย ส่วนในรูปที่ 3.27 (b) Menu ที่ส่งมานั้นใช้แสดงเป็นตัวเลือกสำหรับคำถามเท่านั้น เมื่อจะรับค่าจากผู้ใช้จำเป็นต้องเพิ่มตัวแปรใหม่ขึ้นมารับค่า

รูปแบบฐานความรู้สำหรับรูปแบบที่ 4 เป็นรูปแบบที่มีความยืดหยุ่นสูง สามารถออกแบบเป็นภาษาโปรล็อกได้ทันที ดังแสดงตัวอย่างจากข้อมูล Nursery ในรูปที่ 3.28 หรืออาจจะออกแบบให้อยู่ในรูปแบบวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับได้โดยแสดงในรูปที่ 3.29 ก็ได้

```
class(priority):-
health(recommended),has_nurs(critical),parents(usual),(social(nonprob);social(slightly_prob)).
```

รูปที่ 3.28 แสดงตัวอย่างฐานความรู้ภาษาโปรล็อก

```
class(priority):-
health(recommended),has_nurs(critical),parents(usual),social(S),S&::[nonprob,slightly_prob].
```

รูปที่ 3.29 แสดงตัวอย่างฐานความรู้ที่ออกแบบด้วยวิธีเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ

3.2 เครื่องมือที่ใช้ในการวิจัย

3.2.1 เครื่องมือสำหรับการพัฒนาระบบ

1. เครื่องคอมพิวเตอร์สำหรับพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ โดยมีรายละเอียดดังนี้

- หน่วยประมวลผลกลาง : AMD Athlon™ Neo Processor MV-40 1.60 GHz 512 KB L2 Cache
- หน่วยความจำสำรอง : 250 GB 5400 RPM
- หน่วยความจำหลัก : 2.00 GB DDR2
- อุปกรณ์เสริมอื่น ๆ เช่น เมาส์ แป้นพิมพ์ เป็นต้น

2. ระบบปฏิบัติการและโปรแกรมประยุกต์สำหรับการพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ ประกอบด้วย

- ระบบปฏิบัติการ : Window 7 Ultimate 64-bit Operating System
- เครื่องมือในการพัฒนา : Notepad และ ECLiPEs 6.0 ซึ่งมีสถาปัตยกรรมระบบ เป็น ตัวแปลโปรแกรมส่วนเพิ่ม (Incremental Compiler) คือ โปรแกรมพิเศษที่มีหน้าที่เป็นตัวแปลคำสั่งโปรแกรมให้เป็นรหัสภาษาเครื่องทันทีที่พิมพ์ “ข้อความคำสั่ง” จนจบบรรทัด ตัวแปลโปรแกรมส่วนเพิ่มนี้จะทำหน้าที่แปลในขณะที่พิมพ์คำสั่งแต่ละคำสั่ง (หมายเหตุ

Compiler คือ โปรแกรมที่แปลภาษาโปรแกรมต่าง ๆ เป็นภาษาเครื่อง โดยการแปลทั้งโปรแกรม แล้วนำคำแปลเก็บไว้ในหน่วยความจำ จึงลงมือปฏิบัติการไปทีละคำสั่ง ถ้ามีการสั่งให้ทำบางคำสั่งซ้ำ ก็ไม่จำเป็นต้องแปลใหม่ ส่วน Interpreter คือ โปรแกรมแปลภาษาที่อ่านคำสั่งในโปรแกรมทีละคำสั่ง เพื่อแปลเป็นภาษาเครื่องแล้วปฏิบัติตามคำสั่งนั้นทันที โดยไม่รอจนพิมพ์จบโปรแกรม ซึ่งจะรายงานทันทีว่า คำสั่งนั้นถูกหรือผิด ถ้าสั่งให้ทำอะไรก็จะทำทันที ข้อเสีย ของตัวแปลภาษาประเภทนี้ คือ คอมพิวเตอร์จะไม่เก็บคำแปลไว้ และหากจะต้องทำคำสั่งนั้นอีกก็จะต้องแปลใหม่ทุกครั้ง) (ECLiPSe, 2012; W3dictionary, 2012a; W3dictionary, 2012b; W3dictionary, 2012c)

3.2.2 เครื่องมือที่ใช้ในการวัดประสิทธิภาพ

ประสิทธิภาพที่ต้องการวัดประกอบไปด้วย หน่วยความจำที่ใช้เก็บไฟล์โปรแกรมเปลือกระบบผู้เชี่ยวชาญ หน่วยความจำที่ใช้ขณะประมวลผล และเวลาที่ใช้ในการประมวลผล โดยเครื่องมือที่ใช้ในการวัดประสิทธิภาพมีรายละเอียดดังต่อไปนี้

1. ขนาดของไฟล์ : ดูจากขนาดไฟล์ที่แสดงในหน้าคุณสมบัติของไฟล์
2. ปริมาณการใช้หน่วยความจำขณะใช้ระบบ : ใช้คำสั่ง statistics ในการสังเกตขนาดของหน่วยความจำที่ global_stack_used และ gc_collected ใช้ไปขณะใช้ระบบ (global_stack_used แสดงขนาดของหน่วยความจำที่ถูก Global Stack ใช้อยู่ขณะนั้น ส่วน gc_collected แสดงขนาดของหน่วยความจำที่ถูกสะสมจาก Global Stack ไว้ใน Garbage Collections ภายในรอบการทำงานนั้น)
3. เวลาที่ใช้ในการประมวลผล : ระบุให้โปรแกรมแสดงเวลาด้วยไลบรารี่ util แล้วสังเกตและบันทึกค่า
4. เวลาที่ใช้ในการแสดงผลส่วนต่าง ๆ : แบ่งเป็น expertshell top_goal และ menuask โดยระบุให้โปรแกรมแสดงเวลาด้วยไลบรารี่ util แล้วสังเกตและบันทึกค่า

ในการวัดประสิทธิภาพด้านปริมาณการใช้ซีพียูขณะประมวลผล และเวลาที่ใช้ในการประมวลผลจะเป็นการทดลองและบันทึกผลอย่างน้อย 3 ครั้ง และแสดงกราฟเปรียบเทียบกับประสิทธิภาพของระบบที่เขียนด้วยภาษาโปรล็อก ส่วนเวลาที่ใช้ในการแสดงผลส่วนต่าง ๆ จะเป็นการทำงานร่วมกันของเปลือกระบบผู้เชี่ยวชาญและฐานความรู้รูปแบบต่าง ๆ

บทที่ 4
ผลการวิเคราะห์ข้อมูล และการอภิปรายผล

งานวิจัยนี้เป็นการทดลองเปรียบเทียบ ขนาดของไฟล์ข้อมูล ปริมาณการใช้หน่วยความจำขณะใช้ระบบ และ เวลาที่ใช้ในขั้นตอนต่าง ๆ ของโปรแกรมระหว่างระบบที่พัฒนาด้วย CLP ในงานวิจัยนี้กับระบบที่พัฒนาด้วยภาษาโปรล็อก ภายใต้องค์ประกอบเดียวกันในเรื่อง รูปแบบการทำงานที่คล้ายคลึงกันของระบบ และฐานความรู้ที่มาจากข้อมูลเดียวกัน คือ ฐานความรู้ของ Nursery ประกอบไปด้วยข้อมูลที่ต้องรับจากผู้ใช้จำนวน 8 ค่า (health, has_nurs, parents, social, housing, finance, form, children)

4.1 การวิเคราะห์ข้อมูลในด้านขนาดไฟล์ข้อมูล

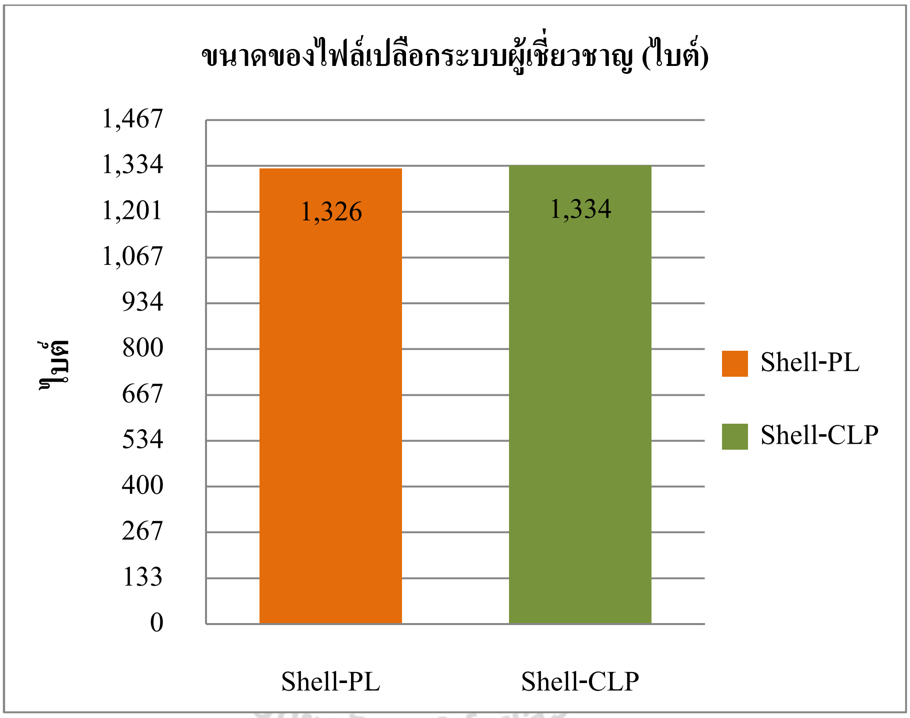
การวิเคราะห์ในประเด็นนี้เป็นการเปรียบเทียบขนาดไฟล์ข้อมูล ของเปลือกระบบผู้เชี่ยวชาญและฐานความรู้ที่ใช้ในการทดสอบระบบ โดยเป็นการเปรียบเทียบระหว่างเปลือกระบบผู้เชี่ยวชาญ และ ฐานความรู้ Nursery ของทั้งภาษาโปรล็อก (PL) และวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ (CLP) โดยข้อมูลนี้เป็นฐานข้อมูลของการคัดเลือกนักเรียนเข้าเรียนระดับอนุบาล ของประเทศสโลวีเนีย ซึ่งได้มาจากแบบจำลองในการตัดสินใจ แบบลำดับชั้นในการจัดอันดับใบสมัคร ที่มีการยื่นใบสมัครเข้าเรียนที่มากเกินไป มีจำนวนข้อมูลทั้งหมด 12,960 แถว และสามารถดาวน์โหลดข้อมูลนี้ได้ที่ http://archive.ics.uci.edu/ml/datasets/Nursery

ตารางที่ 4.1 เปรียบเทียบขนาดไฟล์ข้อมูล

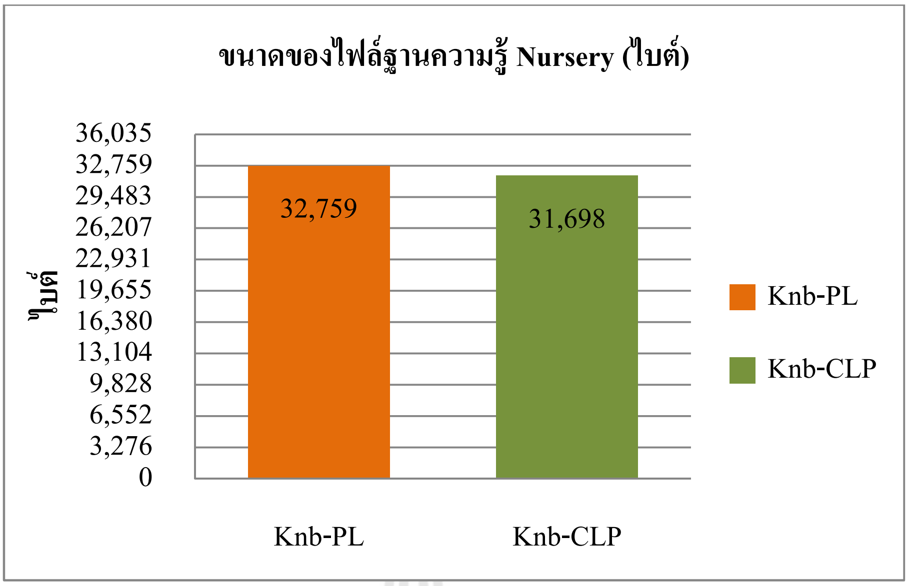
	เปลือกระบบผู้เชี่ยวชาญ (Shell)		ฐานความรู้ (Knb)		หน่วย
	PL	CLP	PL	CLP	
ขนาดของไฟล์	1,326	1,334	32,759	31,698	ไบต์

ข้อมูลในตารางที่ 4.1 แสดงให้เห็นว่า ขนาดของไฟล์ข้อมูล (รูปที่ 4.1 และ 4.2) มีความแตกต่างอย่างชัดเจน คือ ขนาดของไฟล์เปลือกระบบผู้เชี่ยวชาญที่พัฒนาด้วยวิธีเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับจะ “ใหญ่กว่า” ของภาษาโปรล็อกเพียง 8 ไบต์เท่านั้น ซึ่งขนาดข้อมูลในส่วนนี้จะแน่นอนและไม่มีโอกาสขยายได้อีกในอนาคต ไม่เหมือนกับขนาดของไฟล์

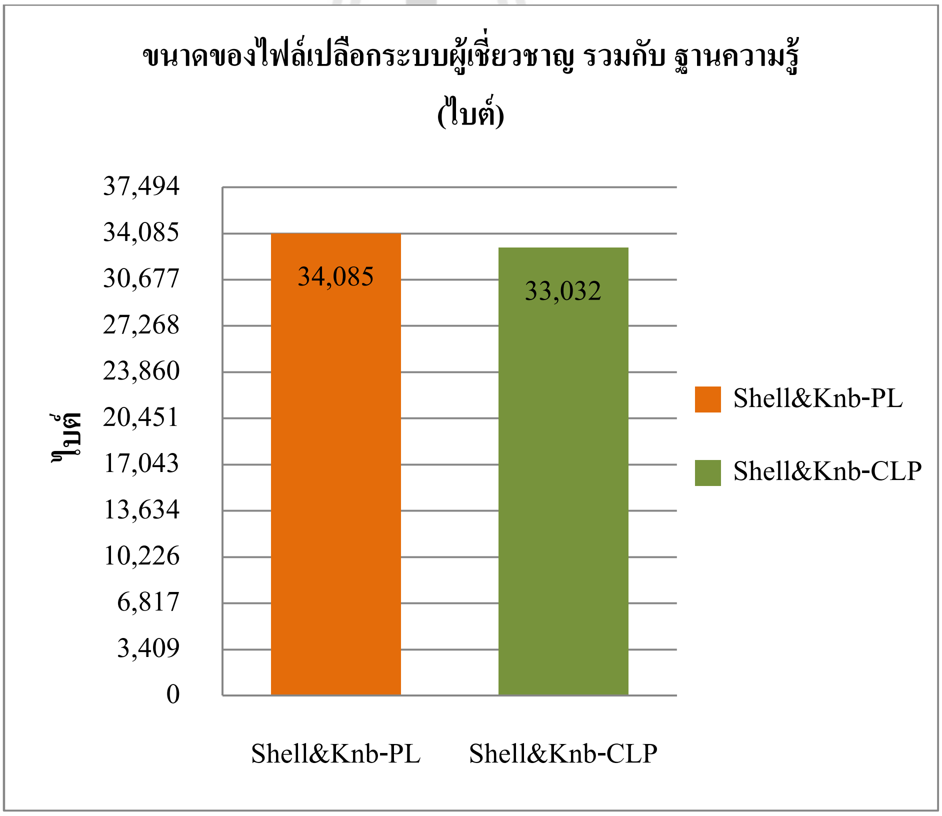
ฐานความรู้ที่มีโอกาสขยายขนาดเพิ่มขึ้นอีกในอนาคต และจากข้อมูลในตารางที่ 4.1 แสดงให้เห็นว่าขนาดของไฟล์ฐานความรู้ที่อยู่ในรูปแบบวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับนั้น "เล็กกว่า" ของภาษาโปรล็อกอยู่ 1,061 ไบต์ ซึ่งตรงกับที่คาดเดาไว้ก่อนทำการทดลอง ดังนั้นเมื่อคิดรวมทั้งขนาดของไฟล์เปลือกระบบผู้เชี่ยวชาญและขนาดของไฟล์ฐานความรู้ (รูปที่ 4.3) จะเห็นว่าโปรแกรมที่เขียนด้วยวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับจะมีขนาดที่เล็กกว่าโปรแกรมที่เขียนด้วยภาษาโปรล็อก



รูปที่ 4.1 แสดงกราฟเปรียบเทียบขนาดของไฟล์เปลือกระบบผู้เชี่ยวชาญ



รูปที่ 4.2 แสดงกราฟเปรียบเทียบขนาดของไฟล์ฐานความรู้ Nursery



รูปที่ 4.3 แสดงกราฟเปรียบเทียบขนาดของไฟล์เปลือกระบบรวมกับฐานความรู้

4.2 การวิเคราะห์ข้อมูลในด้านของปริมาณการใช้หน่วยความจำขณะใช้ระบบ

ในการสังเกตและบันทึกค่าปริมาณการใช้หน่วยความจำขณะใช้เปลือกระบบผู้เชี่ยวชาญที่ใช้ในการทดสอบระบบนั้น ใช้คำสั่ง statistics ในการสังเกตขนาดของหน่วยความจำที่ global_stack_used และ gc_collected ใช้ไปขณะใช้ระบบอยู่ ซึ่งมีการเปลี่ยนแปลงตลอดเวลาตามการใช้งานของผู้ใช้ในโปรแกรม ECLiPSe 6.0

ในประเด็นของการเปรียบเทียบปริมาณการใช้หน่วยความจำขณะใช้เปลือกระบบผู้เชี่ยวชาญที่ใช้ทดสอบระบบ เป็นการเปรียบเทียบปริมาณการใช้หน่วยความจำขณะใช้เปลือกระบบผู้เชี่ยวชาญ ที่เรียกใช้ฐานความรู้ Nursery ทั้งในรูปแบบของภาษาโปรล็อก และวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ ซึ่งในการใช้งานแต่ละครั้งมีการใช้มีการใช้ปริมาณหน่วยความจำเหมือนกันทุกครั้ง ดังนั้นในตารางที่ 4.2 และ 4.3 จะแสดงข้อมูลที่บันทึกมาจากการสังเกตค่าเพียงครั้งเดียว ไม่จำเป็นต้องแสดงข้อมูลของทั้ง 3 ครั้งที่ทำการสังเกต และไม่จำเป็นต้องหาค่าเฉลี่ยใด ๆ

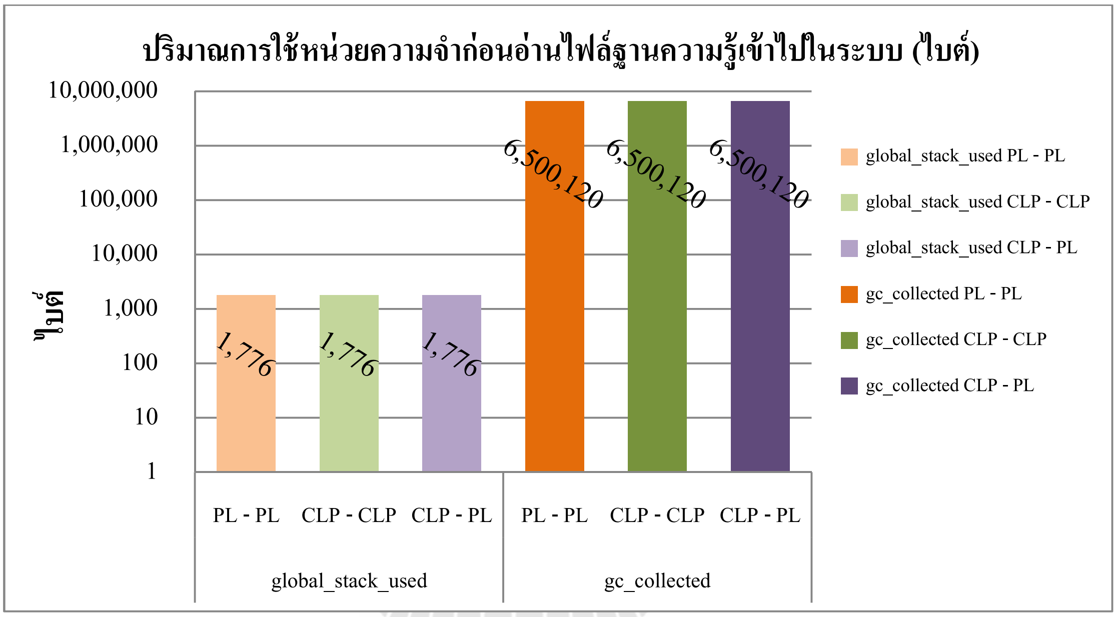
ตารางที่ 4.2 เปรียบเทียบปริมาณการใช้หน่วยความจำขณะใช้ระบบของ global_stack_used

ปริมาณการใช้หน่วยความจำ	รูปแบบของเปลือกระบบ - รูปแบบของฐานความรู้ที่ใช้ทดสอบ			หน่วย
	PL - PL	CLP - CLP	CLP - PL	
ก่อนอ่านไฟล์ knb	1,776	1,776	1,776	ไบต์
หลังอ่านไฟล์ knb	625,240	5,660,088	625,240	ไบต์
ก่อนกลไกอนุมาน	920	920	920	ไบต์
หลังกลไกอนุมาน	3,240	4,144	3,792	ไบต์

ตารางที่ 4.3 เปรียบเทียบปริมาณการใช้หน่วยความจำขณะใช้ระบบของ gc_collected

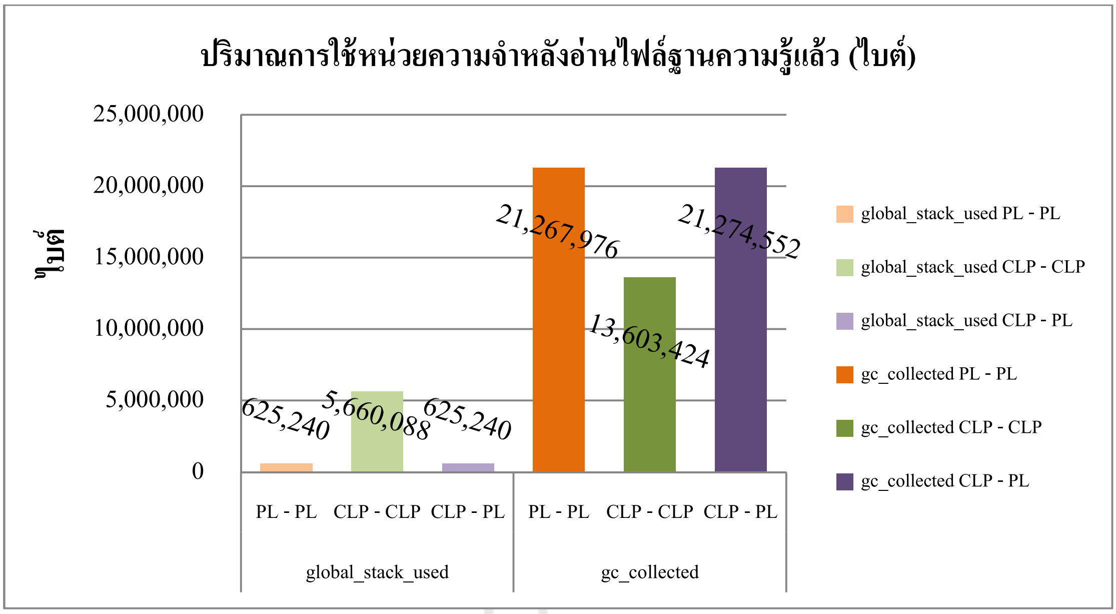
ปริมาณการใช้หน่วยความจำ	รูปแบบของเปลือกระบบ - รูปแบบของฐานความรู้ที่ใช้ทดสอบ			หน่วย
	PL - PL	CLP - CLP	CLP - PL	
ก่อนอ่านไฟล์ knb	6,500,120	6,500,120	6,500,120	ไบต์
หลังอ่านไฟล์ knb	21,267,976	13,603,424	21,274,552	ไบต์
ก่อนกลไกอนุมาน	21,267,976	13,603,424	21,274,552	ไบต์
หลังกลไกอนุมาน	21,267,976	13,603,424	21,274,552	ไบต์

ข้อมูลในตารางที่ 4.2 และ 4.3 แสดงปริมาณของหน่วยความจำที่ถูกใช้ไปขณะที่ระบบกำลังทำงานอยู่ โดยการใช้คำสั่ง statistics เก็บข้อมูลในขั้นตอนการทำงานของระบบทั้งหมด 4 ช่วง คือ ก่อนอ่านไฟล์ฐานความรู้เข้าไปในระบบ (ก่อนอ่านไฟล์ knb) หลังอ่านไฟล์ข้อมูลฐานความรู้แล้ว (หลังอ่านไฟล์ knb) ก่อนเริ่มกลไกอนุมานเพื่อถาม-ตอบคำถาม (ก่อนกลไกอนุมาน) และ หลังผ่านกลไกอนุมานแล้ว (หลังกลไกอนุมาน) ซึ่งข้อมูลนี้มีหน่วยเป็น ไบต์ โดยจะแสดงกราฟเปรียบเทียบในรูปที่ 4.4, 4.5, 4.6 และ 4.7 ตามลำดับขั้นตอนการทำงานของระบบทั้ง 4 ช่วง



รูปที่ 4.4 แสดงกราฟเปรียบเทียบปริมาณการใช้หน่วยความจำก่อนอ่านไฟล์ฐานความรู้

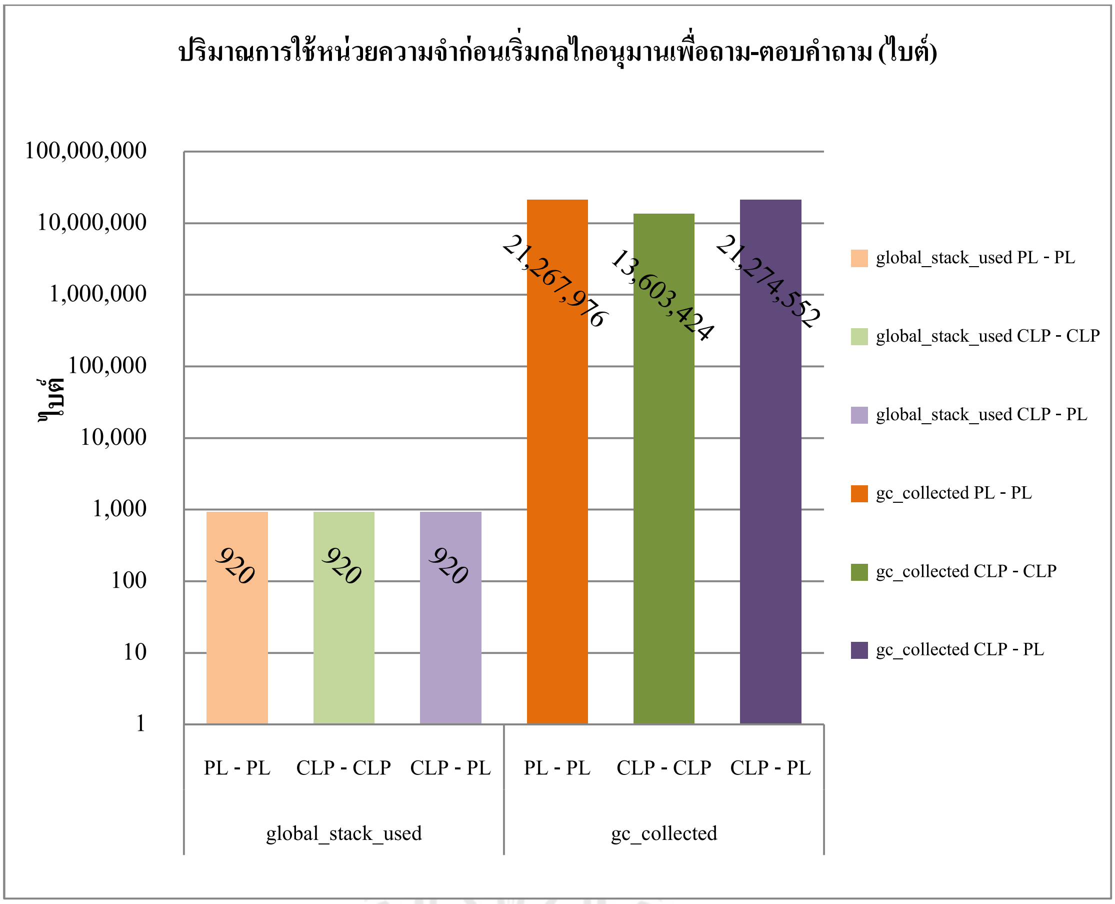
ในรูปที่ 4.4 แสดงปริมาณการใช้หน่วยความจำก่อนอ่านไฟล์ฐานความรู้เข้าไปในระบบ โดยกราฟแต่ละแท่งแทนการจับคู่ระหว่าง เปลือกระบบผู้เชี่ยวชาญกับฐานความรู้ แบ่งออกเป็น 3 คู่ คือ เปลือกระบบผู้เชี่ยวชาญกับฐานความรู้ภาษาโปรล็อก (PL-PL) เปลือกระบบผู้เชี่ยวชาญกับฐานความรู้แบบวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ (CLP-CLP) และ เปลือกระบบผู้เชี่ยวชาญแบบวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับกับฐานความรู้ภาษาโปรล็อก (CLP-PL) ซึ่งปริมาณการใช้หน่วยความจำในส่วนของ global_stack_used ทั้ง 3 คู่มีค่าเท่ากัน คือ 1,776 ไบต์ และ gc_collected ของทั้ง 3 คู่ก็มีค่าเท่ากัน คือ 6,500,120 ไบต์ (เปลือกระบบผู้เชี่ยวชาญภาษาโปรล็อกไม่สามารถใช้ร่วมกับฐานความรู้แบบวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับได้จึงไม่ได้นำมาทดลอง)



รูปที่ 4.5 แสดงกราฟเปรียบเทียบปริมาณการใช้หน่วยความจำหลังอ่านไฟล์ฐานความรู้

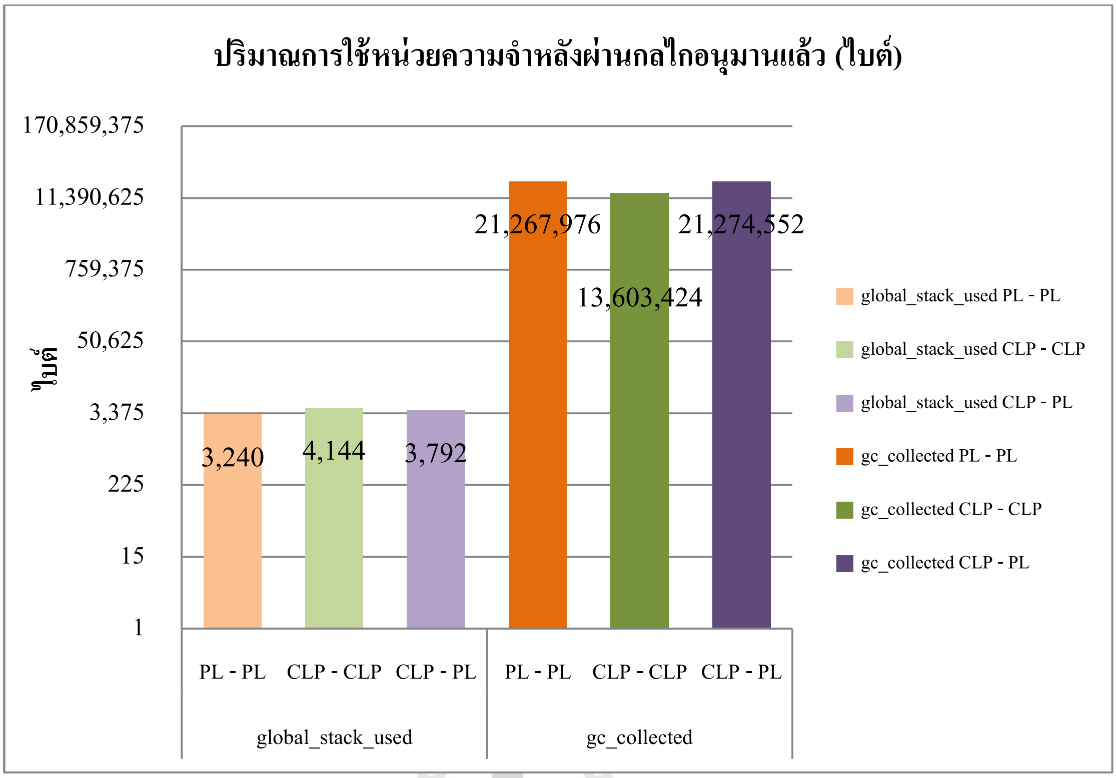
เมื่อดูค่าของ global_stack_used ในรูปที่ 4.5 จะเห็นว่าเปลือกระบบผู้เชี่ยวชาญที่จับคู่กับฐานความรู้แบบวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ (CLP-CLP) นั้นจะใช้หน่วยความจำ “มากกว่า” เปลือกระบบผู้เชี่ยวชาญที่จับคู่กับฐานความรู้ภาษาโปรล็อก (PL-PL และ CLP-PL) อยู่ 5,034,848 ไบต์ เนื่องจากรูปแบบการเขียนโปรแกรมที่มีการเขียนเงื่อนไขบังคับด้วยนั้น ทำให้เมื่ออ่านไฟล์เข้ามาแล้ว ระบบจำเป็นจะต้องอ่านค่าของไลบรารี่ต่าง ๆ ที่เกี่ยวข้อง ซึ่งขาดไม่ได้เมื่อเขียนโปรแกรมด้วยรูปแบบนี้ แต่ด้วยเหตุผลเดียวกันนี้ทำให้ gc_collected ของคู่เปลือกระบบผู้เชี่ยวชาญกับฐานความรู้แบบวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ (CLP-CLP) นั้นใช้หน่วยความจำ “น้อยกว่า” คู่ของเปลือกระบบผู้เชี่ยวชาญกับฐานความรู้ภาษาโปรล็อก (PL-PL) อยู่ 7,664,552 ไบต์ และ “น้อยกว่า” คู่ของเปลือกระบบผู้เชี่ยวชาญแบบวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับกับฐานความรู้ภาษาโปรล็อกอยู่ 7,671,128 ไบต์

เนื่องจากรูปแบบวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับที่มีการอ่านไลบรารี่เก็บไว้ที่ global_stack_used แล้ว ทำให้ไม่ต้องเก็บค่าไว้ที่ gc_collected อีก คิดเป็นส่วนต่างที่สามารถลดปริมาณการใช้หน่วยความจำได้ประมาณ 2,629,704 ถึง 2,636,280 ไบต์ หลังจากระบบได้อ่านไฟล์ฐานความรู้เข้ามาแล้ว



รูปที่ 4.6 แสดงกราฟเปรียบเทียบปริมาณการใช้หน่วยความจำก่อนเริ่มกลไกอนุมาน

ในรูปที่ 4.6 แสดงให้เห็นว่า ภายหลังการอ่านไฟล์ฐานความรู้ที่ทำให้การใช้หน่วยความจำในส่วน global_stack_used มีความแตกต่างกัน แต่ก็เพียงชั่วคราวเท่านั้น เพราะเมื่อทำการเก็บข้อมูลในขั้นตอนก่อนการทำงานของกลไกอนุมาน พบว่า ปริมาณการใช้หน่วยความจำของ global_stack_used ทั้ง 3 คู่ที่ทดลองนั้นกลับไปมีค่าเท่ากันอีกครั้ง และมีปริมาณลดน้อยลงมาก ส่วนของ gc_collected ก็แสดงค่าออกมาเท่ากันกับในขั้นตอนหลังอ่านไฟล์ฐานความรู้ ซึ่งเพียงพอที่จะสรุปได้ว่า ปริมาณการใช้หน่วยความจำของคู่เปลือกระบบผู้เชี่ยวชาญแบบวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับกับฐานความรู้รูปแบบเดียวกันนั้น จะใช้ “น้อยกว่า” เปลือกระบบผู้เชี่ยวชาญที่คู่กับฐานความรู้ภาษาโปรล็อก เช่นเดียวกับในรูปที่ 4.5



รูปที่ 4.7 แสดงกราฟเปรียบเทียบปริมาณการใช้หน่วยความจำหลังผ่านกลไกอนุมาน

สำหรับปริมาณการใช้หน่วยความจำของ gc_collected ในรูปที่ 4.7 นั้นแสดงผลลัพธ์เป็นค่าเดียวกันกับที่แสดงในรูปที่ 4.5 และ 4.6 สรุปว่าค่าของ gc_collected เปลี่ยนแปลงไปหลังขั้นตอนการอ่านไฟล์ฐานความรู้ และมีค่าคงที่ในระดับนี้ถึงขั้นตอนหลังผ่านกลไกอนุมาน ซึ่งผลที่แสดงออกมานั้นชี้ให้เห็นว่า การจับคู่กันของเปลือกระบบผู้เชี่ยวชาญและฐานความรู้แบบวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับนั้นมีการใช้หน่วยความจำ “น้อยกว่า” การจับคู่ของเปลือกระบบผู้เชี่ยวชาญทั้ง 2 แบบกับฐานความรู้ภาษาโปรล็อก

ส่วนปริมาณการใช้หน่วยความจำของ global_stack_used ในรูปที่ 4.7 มีความเปลี่ยนแปลงอีกครั้ง แต่มีการใช้หน่วยความจำในปริมาณที่น้อยกว่าการเปลี่ยนแปลงครั้งแรกที่แสดงในรูปที่ 4.5 แต่ผลการเปรียบเทียบระหว่างการจับคู่เปลือกระบบกับฐานความรู้ทั้ง 3 คู่ จะเห็นว่าเปลือกระบบผู้เชี่ยวชาญแบบวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับนั้นใช้พื้นที่หน่วยความจำมากกว่าเปลือกระบบผู้เชี่ยวชาญภาษาโปรล็อก และเมื่อใช้ร่วมกับฐานความรู้ที่มีรูปแบบวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ ก็ทำให้จำเป็นต้องใช้พื้นที่หน่วยความจำเพิ่มขึ้นอีก แต่เป็นปริมาณที่น้อยมากเมื่อเทียบกับปริมาณที่ลดการใช้ลงได้ของ gc_collected

4.3 การวิเคราะห์ข้อมูลในด้านของเวลา

ในการบันทึกเวลาที่ใช้ไปของแต่ละขั้นตอนนั้น เป็นการเก็บข้อมูลโดยอาศัยเครื่องมือที่มีอยู่แล้วในโปรแกรม ECLiPSe 6.0 คือ ไลบรารี่ util “lib(util)” แล้วเรียกใช้คำสั่ง time(ชื่อpredicate) โดยสามารถแทรกคำสั่งนี้ในส่วนต่าง ๆ ของโปรแกรม เมื่อโปรแกรมทำงานในส่วนนั้นเสร็จก็จะพิมพ์แสดงเวลาออกมา มีหน่วยเป็นวินาที

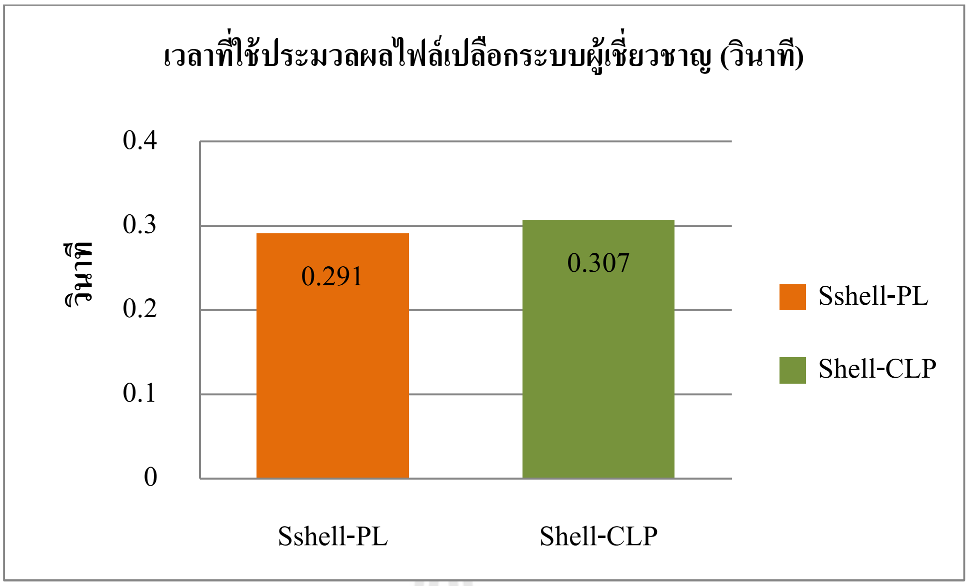
4.3.1 การเปรียบเทียบเวลาที่ใช้ประมวลผลไฟล์เปลือกระบบผู้เชี่ยวชาญ

เป็นการเปรียบเทียบเวลาที่ได้หลังจากประมวลผลไฟล์เปลือกระบบผู้เชี่ยวชาญของทั้งภาษาโปรล็อก และวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ โดยทำการสังเกตและบันทึกค่าเวลาทั้งหมด 3 ครั้งเพื่อหาค่าเฉลี่ย

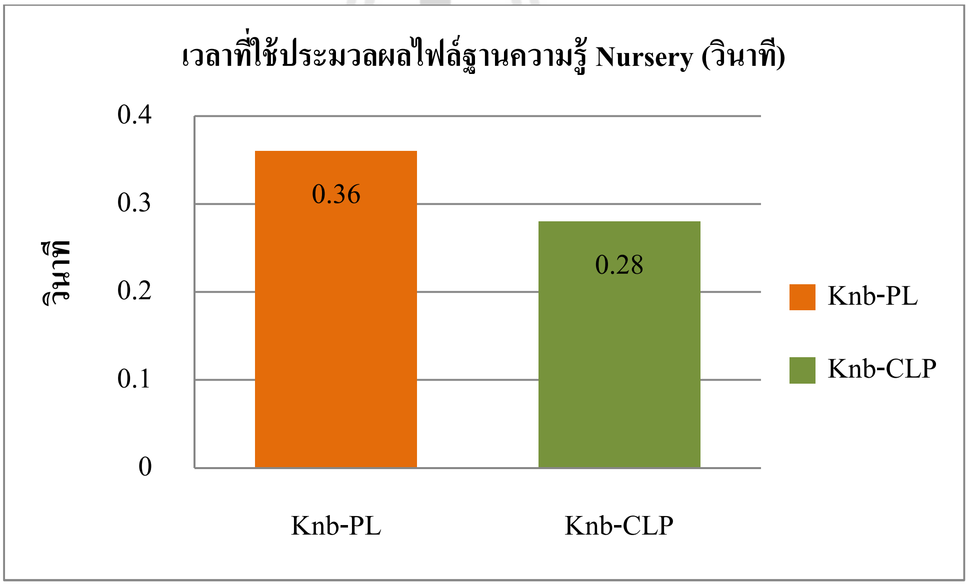
ตารางที่ 4.4 เปรียบเทียบเวลาที่ได้หลังจากประมวลผลไฟล์เปลือกระบบผู้เชี่ยวชาญ

เวลาที่ใช้ประมวลผล	เปลือกระบบผู้เชี่ยวชาญ (Shell)		ฐานความรู้ (Knb)		หน่วย
	PL	CLP	PL	CLP	
การทดสอบ ครั้งที่ 1	0.265	0.265	0.36	0.27	วินาที
การทดสอบ ครั้งที่ 2	0.312	0.296	0.34	0.31	วินาที
การทดสอบ ครั้งที่ 3	0.296	0.359	0.38	0.27	วินาที
ค่าเฉลี่ย	0.291	0.307	0.36	0.28	วินาที

จากข้อมูลในตารางที่ 4.4 แสดงให้เห็นว่า เวลาเฉลี่ยที่ใช้ประมวลผลไฟล์ (compile) เปลือกระบบผู้เชี่ยวชาญที่พัฒนาด้วยวิธีเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ “มากกว่า” ของภาษาโปรล็อกอยู่ 0.016 วินาที และเวลาเฉลี่ยที่ใช้ประมวลผลไฟล์ฐานความรู้ที่เขียนด้วยวิธีเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ “น้อยกว่า” ของภาษาโปรล็อกอยู่ 0.08 วินาที จะสังเกตเห็นว่าผลที่ได้จะสัมพันธ์กับขนาดของไฟล์ข้อมูลที่บันทึกไว้ในตารางที่ 4.1 แสดงกราฟเปรียบเทียบดังรูปที่ 4.8 และ 4.9



รูปที่ 4.8 แสดงกราฟเปรียบเทียบเวลาเฉลี่ยที่ใช้ขณะประมวลผลไฟล์เปลือกระบบผู้เชี่ยวชาญ



รูปที่ 4.9 แสดงกราฟเปรียบเทียบเวลาเฉลี่ยที่ใช้ขณะประมวลผลไฟล์ฐานความรู้ Nursery

4.3.2 การเปรียบเทียบเวลาที่ใช้ขณะแสดงผลส่วนต่าง ๆ ของระบบ

เป็นการเปรียบเทียบเวลาที่ใช้ไปขณะแสดงผลส่วนต่าง ๆ ของระบบ โดยในงานวิจัยนี้ได้ทำการสังเกตและบันทึกค่าส่วนประกอบดังต่อไปนี้ expertshell top_goal และ menuask ซึ่งเป็นเพรดิเคตที่มีอยู่ในเปลือกระบบทั้งสองแบบทดสอบกับฐานความรู้ Nursery

ตารางที่ 4.5 เปรียบเทียบเวลาที่ใช้ในการแสดงผลของ expertshell

เวลาที่ใช้แสดงผล	รูปแบบของเปลือกระบบ - รูปแบบของฐานความรู้ที่ใช้ทดสอบ			หน่วย
	PL - PL	CLP - CLP	CLP - PL	
การทดสอบครั้งที่ 1	0.562	0.436	0.499	วินาที
การทดสอบครั้งที่ 2	0.562	0.515	0.593	วินาที
การทดสอบครั้งที่ 3	0.515	0.437	0.499	วินาที
ค่าเฉลี่ย	0.546	0.463	0.530	วินาที

ตารางที่ 4.6 เปรียบเทียบเวลาที่ใช้ในการแสดงผลของ top_goal

เวลาที่ใช้แสดงผล	รูปแบบของเปลือกระบบ - รูปแบบของฐานความรู้ที่ใช้ทดสอบ			หน่วย
	PL - PL	CLP - CLP	CLP - PL	
การทดสอบครั้งที่ 1	0.078	0.047	0.078	วินาที
การทดสอบครั้งที่ 2	0.063	0.046	0.109	วินาที
การทดสอบครั้งที่ 3	0.093	0.031	0.094	วินาที
ค่าเฉลี่ย	0.078	0.041	0.094	วินาที

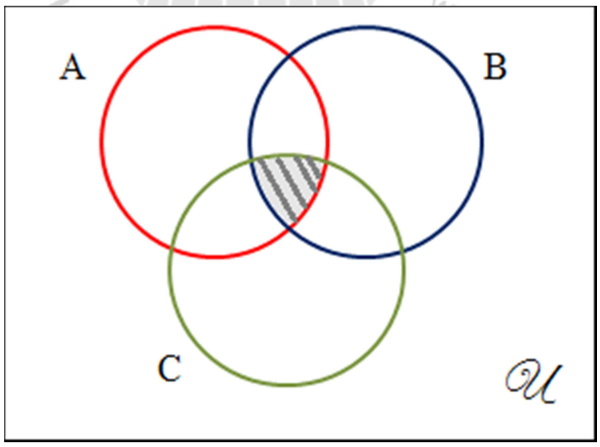
ตารางที่ 4.7 เปรียบเทียบเวลาที่ใช้ในการแสดงผลของ menuask

เวลาที่ใช้แสดงผล	รูปแบบของเปลือกระบบ - รูปแบบของฐานความรู้ที่ใช้ทดสอบ			หน่วย
	PL - PL	CLP - CLP	CLP - PL	
การทดสอบครั้งที่ 1	0.029	0.016	0.021	วินาที
การทดสอบครั้งที่ 2	0.021	0.015	0.036	วินาที
การทดสอบครั้งที่ 3	0.031	0.010	0.031	วินาที
ค่าเฉลี่ย	0.027	0.014	0.029	วินาที

จากข้อมูลในตารางที่ 4.5 ถึง 4.7 เมื่อทำการทดสอบเปลือกระบบผู้เชี่ยวชาญที่มีรูปแบบวิธีเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับกับฐานความรู้ที่เป็นรูปแบบเดียวกัน (CLP - CLP) จะเห็นว่าค่าเฉลี่ยเวลาที่ใช้ในการแสดงผลส่วนต่าง ๆ ของระบบนั้น “น้อยกว่า” ของเปลือกระบบผู้เชี่ยวชาญภาษาโปรล็อกที่ทดสอบกับฐานความรู้ภาษาโปรล็อก (PL - PL) อยู่ 0.123 วินาที 0.037 วินาที และ 0.013 วินาที ตามค่าที่ปรากฏในตารางที่ 4.5 4.6 และ 4.7 ตามลำดับ

นอกจากการทดสอบเปลือกระบบผู้เชี่ยวชาญที่มีรูปแบบวิธีเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับกับฐานความรู้ที่เป็นรูปแบบเดียวกันแล้ว ในงานวิจัยนี้ได้ทดสอบเปลือกระบบผู้เชี่ยวชาญที่มีรูปแบบวิธีเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับกับฐานความรู้ภาษาโปรล็อก (CLP - PL) เพื่อหารูปแบบที่ดีที่สุดในการนำไปใช้ต่อไป ซึ่งจากข้อมูลในตารางที่ 4.6 และ 4.7 แสดงให้เห็นว่า เมื่อจับคู่ CLP - PL แล้วจะได้ค่าเฉลี่ยเวลาที่ “มากกว่า” คู่ของ PL - PL อยู่ 0.016 วินาทีและ 0.002 วินาทีตามลำดับ และเมื่อเปรียบเทียบกับคู่ของ CLP - CLP ก็พบว่า CLP - PL จะใช้เวลามากกว่าอยู่ 0.053 วินาทีและ 0.015 วินาทีตามลำดับ

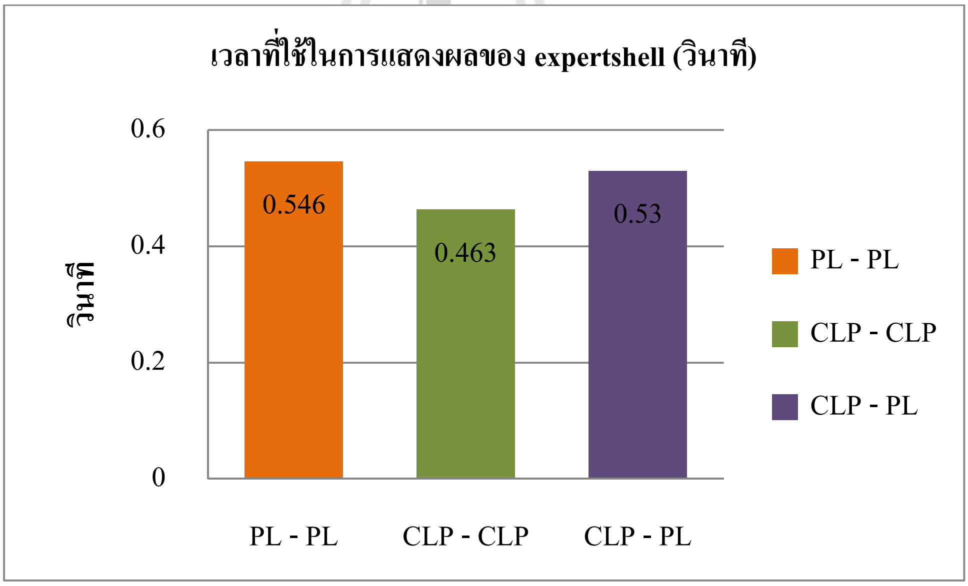
ปัจจัยสำคัญที่ทำให้ CLP - CLP ใช้เวลาน้อยกว่าการจับคู่อื่น ๆ คือ การประกาศข้อมูลในลักษณะของโดเมน ทั้งเปลือกระบบผู้เชี่ยวชาญและฐานความรู้ ทำให้กลไกอนุมานที่เกิดขึ้นเป็นลักษณะของการตรวจสอบ Intersection ของเซต 3 เซต คือ เซตของคำตอบที่เป็นไปได้ทั้งหมด (A) เซตของคำตอบที่เป็นไปได้สำหรับเป้าหมายที่ต้องการ (B) และเซตของคำตอบที่รับมาจากผู้ใช้ (C) ดังแสดงด้วยแผนภาพในรูปที่ 4.10



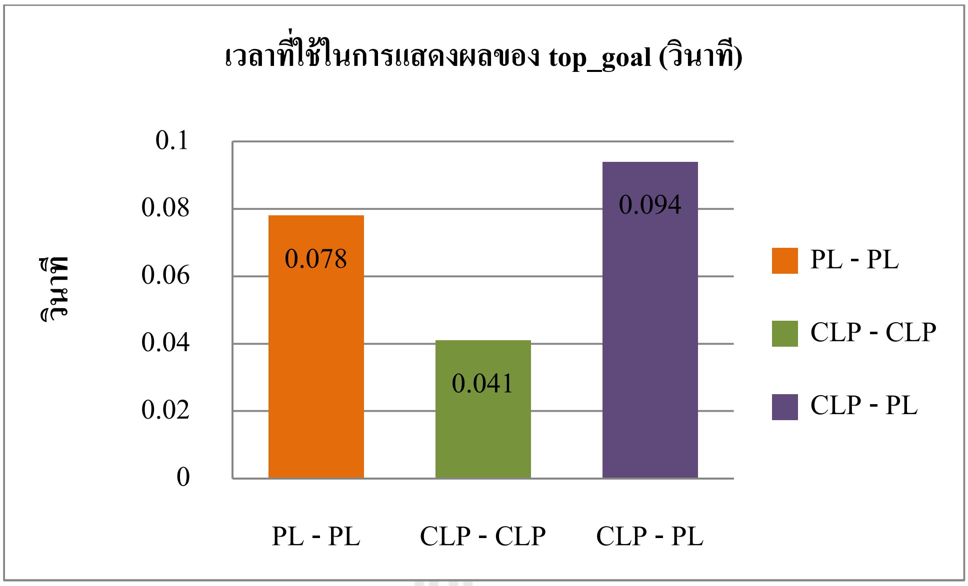
รูปที่ 4.10 แสดงการ Intersection กันของเซต 3 เซต และส่วนที่แรเงา คือ ค่าของตัวแปรที่เหมือนกันทั้ง 3 เซต

ส่วนในตารางที่ 4.5 จะมีความแตกต่างอยู่เล็กน้อย คือ CLP - PL “น้อยกว่า” PL - PL อยู่ 0.016 วินาทีและ “มากกว่า” CLP - CLP อยู่ 0.067 วินาที เนื่องจากเวลาในส่วนของ expertshell คือการจับเวลาการทำงานของระบบโดยรวมทั้งหมด ตั้งแต่ 1. เริ่มการทำงานของระบบ 2. เรียกใช้คำสั่ง load เพื่อประมวลผลไฟล์ฐานความรู้เข้าไปรวมในระบบ 3. เรียกใช้คำสั่ง solve ซึ่งเป็นส่วนเริ่มต้นของ top_goal และ menuask 4. ใช้คำสั่ง why เพื่อทดสอบการให้เหตุผลของระบบให้ครบกระบวนการทำงาน และสิ้นสุดลงในส่วนที่ 5. คำสั่ง quit เพื่อสิ้นสุดการทำงานของระบบ

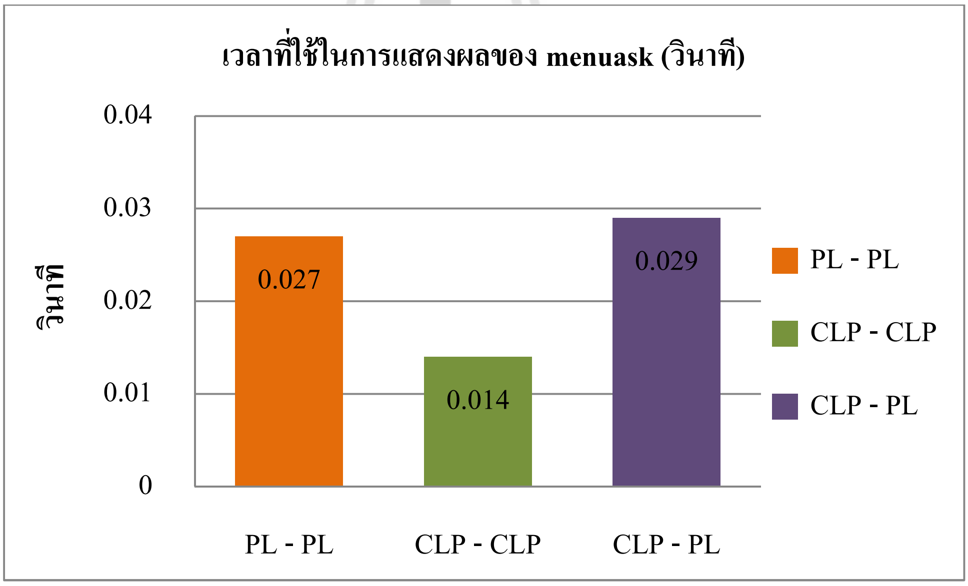
ปัจจัยที่ทำให้ CLP - PL ใช้เวลามากกว่าในส่วนของ top_goal และ menuask คือ รูปแบบการเขียนโปรแกรมที่เปลือกระบบมีการระบุให้ลิสต์ของคำตอบที่เป็นไปได้เป็นโดเมนของตัวแปร เมื่อมีการรับคำตอบจากผู้ใช้ระบบก็จะเสียเวลาเล็กน้อยในการตรวจสอบว่าค่าที่ได้รับมานั้นตรงกับในโดเมนหรือไม่ (CLP) จากนั้นก็จะไปตรวจสอบในฐานความรู้ที่จะเสียเวลาในส่วนของการเรียกซ้ำเพื่อตรวจสอบกับข้อมูลทีละค่า (PL) ซึ่งจากการสังเกตค่าที่บันทึกไว้ในการทดลอง 3 ครั้ง ในตารางที่ 4.5 4.6 และ 4.7 จะเห็นว่า ค่าเฉลี่ยเวลาของ CLP - PL และ PL - PL นั้นดูใกล้เคียงกัน ดังแสดงภาพเปรียบเทียบได้ดังรูปที่ 4.11 4.12 และ 4.13



รูปที่ 4.11 แสดงกราฟเปรียบเทียบค่าเฉลี่ยเวลาที่ใช้ในการแสดงผล expertshell



รูปที่ 4.12 แสดงกราฟเปรียบเทียบค่าเฉลี่ยเวลาที่ใช้ในการแสดงผล top_goal



รูปที่ 4.13 แสดงกราฟเปรียบเทียบค่าเฉลี่ยเวลาที่ใช้ในการแสดงผล menuask

4.4 สรุปการเปรียบเทียบผลการทดสอบ

ในตารางที่ 4.8 ได้เปรียบเทียบผลการทดสอบทั้งหมดและสรุปได้ว่า เมื่อเปรียบเทียบในด้านของขนาดไฟล์ และเวลาที่ใช้ประมวลผลของเปลือกระบบผู้เชี่ยวชาญพบว่าของ CLP (วิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ) มากกว่า PL (ภาษาโปรล็อก) แต่เมื่อเปรียบเทียบฐานความรู้ทั้งสองรูปแบบในทำนองเดียวกับเปลือกระบบพบว่าของ CLP น้อยกว่า PL ส่วนปริมาณการใช้หน่วยความจำของ global_stack_used พบว่าทั้งเปลือกระบบกับฐานความรู้ของ CLP มากกว่า เปลือกระบบ CLP และ PL ที่คู่กับฐานความรู้ PL และปริมาณการใช้หน่วยความจำของ gc_collected พบว่าทั้งเปลือกระบบกับฐานความรู้ของ CLP น้อยกว่า เปลือกระบบ CLP และ PL ที่คู่กับฐานความรู้ PL และเมื่อเปรียบเทียบในด้านเวลาแสดงผลของเพรดิเคตทั้งของ expertshell top_goal และ menuask พบว่า การจับคู่ระหว่างเปลือกระบบผู้เชี่ยวชาญ CLP กับฐานความรู้ CLP นั้นใช้เวลาแสดงผลน้อยที่สุด

ตารางที่ 4.8 ตารางสรุปการเปรียบเทียบผลการทดสอบ

<table>
<tr><th>ชื่อรายการ</th><th colspan="2">การเปรียบเทียบ</th><th>ชื่อรายการ</th><th>การเปรียบเทียบ</th></tr>
<tr><td>ขนาดของไฟล์ Shell</td><td colspan="2">CLP > PL</td><td>ขนาดของไฟล์ KNB</td><td>CLP < PL</td></tr>
<tr><td>เวลาประมวลผล Shell</td><td colspan="2">CLP > PL</td><td>เวลาประมวลผล KNB</td><td>CLP < PL</td></tr>
<tr><th colspan="3">ชื่อรายการ</th><th colspan="2">การเปรียบเทียบ</th></tr>
<tr><td colspan="3">เวลาแสดงผล expertshell</td><td colspan="2">CLP - CLP < CLP - PL < PL - PL</td></tr>
<tr><td colspan="3">เวลาแสดงผล top_goal</td><td colspan="2">CLP - CLP < PL - PL < CLP - PL</td></tr>
<tr><td colspan="3">เวลาแสดงผล menuask</td><td colspan="2">CLP - CLP < PL - PL < CLP - PL</td></tr>
<tr><th rowspan="2">ขั้นตอนการทำงานของระบบ</th><th colspan="4">การเปรียบเทียบปริมาณการใช้หน่วยความจำ</th></tr>
<tr><th colspan="2">global_stack_used</th><th colspan="2">gc_collected</th></tr>
<tr><td>ก่อนอ่านไฟล์ KNB</td><td colspan="2">CLP - CLP = CLP - PL = PL - PL</td><td colspan="2">CLP - CLP = CLP - PL = PL - PL</td></tr>
<tr><td>หลังอ่านไฟล์ KNB</td><td colspan="2">CLP - CLP > CLP - PL = PL - PL</td><td colspan="2">CLP - CLP < PL - PL < CLP - PL</td></tr>
<tr><td>ก่อนกลไกอนุมาน</td><td colspan="2">CLP - CLP = CLP - PL = PL - PL</td><td colspan="2">CLP - CLP < PL - PL < CLP - PL</td></tr>
<tr><td>หลังกลไกอนุมาน</td><td colspan="2">CLP - CLP > CLP - PL > PL - PL</td><td colspan="2">CLP - CLP < PL - PL < CLP - PL</td></tr>
</table>

<u>หมายเหตุ</u> Shell คือ เปลือกระบบผู้เชี่ยวชาญ KNB คือ ฐานความรู้ PL คือ ไฟล์ข้อมูลรูปแบบภาษาโปรล็อก และ CLP คือ ไฟล์ข้อมูลรูปแบบวิธีเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ

บทที่ 5
สรุปและข้อเสนอแนะ

งานวิจัยนี้เป็นการศึกษาและพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ แล้ววัดประสิทธิภาพด้วยการเปรียบเทียบผลการทดลองกับเปลือกระบบผู้เชี่ยวชาญภาษาโปรล็อก โดยการสังเกตและบันทึกค่าในด้านขนาดไฟล์ของโปรแกรมเปลือกระบบผู้เชี่ยวชาญและฐานความรู้ ปริมาณการใช้หน่วยความจำขณะใช้ระบบของ global_stack_used กับ gc_collected เวลาที่ใช้ในการประมวลผลไฟล์ และเวลาที่ใช้ในการแสดงผลส่วนของ expertshell top_goal และ menuask โดยเลือกใช้รูปแบบการพัฒนาเปลือกระบบผู้เชี่ยวชาญและฐานความรู้ในรูปแบบที่ 4 ซึ่งเป็นรูปแบบที่ดีที่สุดเป็นโปรแกรมทดสอบ ซึ่งสามารถสรุปผลการวิจัยได้ดังนี้

5.1 สรุปผลการวิจัย

5.1.1 ประสิทธิภาพที่วัดได้จากการปรับรูปแบบของเปลือกระบบผู้เชี่ยวชาญ

ข้อเสียที่เกิดจากการแปลงรูปแบบของเปลือกระบบผู้เชี่ยวชาญภาษาโปรล็อก (ESS - PL) เป็นรูปแบบการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ (ESS - CLP) แล้วทำให้ขนาดไฟล์ข้อมูลใหญ่ขึ้นอีก 8 ไบต์ ซึ่งกล่าวได้ว่าไม่มีผลกระทบในด้านการจัดเก็บข้อมูลมากนัก แต่ทำให้เวลาที่ใช้ประมวลผลไฟล์ ESS - CLP นั้นมากกว่า ESS - PL และเนื่องจากรูปแบบการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ ทำให้ปริมาณการใช้หน่วยความจำของ global_stack_used ที่เป็น ESS - CLP นั้นมากกว่า ESS-PL และมากขึ้นอีกถ้าใช้คู่กับฐานความรู้รูปแบบการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ (KNB - CLP)

ข้อดีที่เกิดจากการแปลงรูปแบบของเปลือกระบบผู้เชี่ยวชาญ คือ เมื่อทำการวัดริมาณการใช้หน่วยความจำของ gc_collected พบว่า ESS - CLP ที่ใช้ KNB - CLP นั้นใช้หน่วยความจำน้อยกว่า ESS – PL และ ESS – CLP ที่ใช้ฐานความรู้ภาษาโปรล็อก (KNB - PL) และมีผลต่อเวลาที่ใช้ในการแสดงผลระบบโดยรวม (เวลาที่ใช้ในการแสดงผล expertshell) ของรูปแบบการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับนั้นเร็วกว่าของภาษาโปรล็อกเมื่อใช้ร่วมกับฐานข้อมูลภาษาโปรล็อก และจะเร็วขึ้นอีกเมื่อใช้ร่วมกับฐานข้อมูลที่มีรูปแบบการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ (เทียบกับเปลือกระบบผู้เชี่ยวชาญภาษาโปรล็อก)

5.1.2 ประสิทธิภาพที่วัดได้จากการปรับรูปแบบของฐานความรู้

ข้อเสียที่เกิดจากการแปลงรูปแบบของฐานความรู้จากภาษาโปรล็อก (KNB - PL) ให้อยู่ในรูปของการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ (KNB - CLP) คือ เมื่อใช้ KNB - CLP กับ ESS - CLP มีผลทำให้ใช้ปริมาณหน่วยความจำเพิ่มขึ้นในส่วนของ global_stack_used เนื่องจากจำเป็นจะต้องเรียกใช้ไลบรารี่ต่าง ๆ ที่เกี่ยวข้อง

ข้อดี คือ ขนาดของไฟล์ KNB - CLP เล็กลง ทำให้ประหยัดพื้นที่ในการจัดเก็บข้อมูลที่อาจจะมีเพิ่มขึ้นอีกในอนาคตได้ ทำให้เวลาที่ใช้ในการประมวลผลไฟล์น้อยลง เมื่อวัดประสิทธิภาพด้านเวลาที่ใช้ในการแสดงผลทั้งระบบโดยรวม (expertshell) กลไกอนุมานทั้งหมด (top_goal) กลไกอนุมานของแต่ละคำถาม (menuask) พบว่า เมื่อ ESS - CLP จับคู่กับ KNB - CLP แล้วจะช่วยลดเวลาลงได้ดีกว่าการจับคู่แบบอื่น ๆ และมีผลทำให้ปริมาณการใช้หน่วยความจำลดลงมากในส่วนของ gc_collected เมื่อจับคู่ระหว่าง ESS - CLP กับ KNB - CLP เท่านั้น เพราะโปรแกรมสามารถเรียกใช้คำสั่งที่จำเป็นได้ผ่านไลบรารี่จึงไม่จำเป็นต้องเก็บค่าไว้ใน gc_collected

5.2 ข้อเสนอแนะ

ในการพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับนี้ มีรูปแบบการทำงานและการแสดงผลของระบบแบบอย่างง่าย ซึ่งเพียงพอที่จะใช้วัดประสิทธิภาพของโปรแกรม โดยเปรียบเทียบระหว่างเปลือกระบบผู้เชี่ยวชาญที่พัฒนาด้วย CLP และภาษาโปรล็อก หากจะพัฒนาโปรแกรมให้มีประสิทธิภาพยิ่งขึ้นสามารถทำได้หลายแนวทาง อาทิ เช่น

- พัฒนาเปลือกระบบผู้เชี่ยวชาญให้มีรูปแบบการถาม-ตอบที่หลากหลายขึ้น เช่น การแสดงคำถามให้ตอบแค่ yes หรือ no การแสดงคำถามให้ตอบเป็นตัวเลือก ก ข ค ง ฯลฯ โดยไม่จำเป็นต้องพิมพ์ข้อความของคำตอบทั้งหมด
- พัฒนาเปลือกระบบผู้เชี่ยวชาญจากแบบ Stand-Alone ให้เป็น Web-Base เพื่อรองรับการใช้งานของผู้ใช้หลาย ๆ คนพร้อมกัน
- ด้วยวิธีเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับสามารถพัฒนาโดยเพิ่มข้อมูลที่ต้องผ่านกระบวนการคำนวณทางคณิตศาสตร์ก่อนได้ เช่น การกำหนดค่าน้ำหนัก (Weight) เพื่อระบุลำดับความสำคัญของฐานความรู้ ว่าข้อมูลใดมีโอกาสเกิดขึ้นบ่อยกว่ากัน หรือ การเพิ่มเพรดิเคตสำหรับคำนวณสมการคณิตศาสตร์โดยเฉพาะในเปลือกระบบผู้เชี่ยวชาญ เป็นต้น

รายการอ้างอิง

กิดานันท์ มลิทอง (2539). **อธิบายศัพท์คอมพิวเตอร์อินเตอร์เนตมัลติมีเดีย [ออนไลน์].** ที่มา : http://www.nukul.ac.th/it/itdic/11.htm

นิตยา เกิดประสพ (2552). การประมวลผลหลังกระบวนการทำเหมืองข้อมูล. **โครงการวิจัยโดยทุนอุดหนุนจากมหาวิทยาลัยเทคโนโลยีสุรนารี รหัสโครงการ SUT7-705-48-24-13.** หน้า 1-5

วิกิพีเดีย สารานุกรมเสรี (2554ก). **การแทนความรู้ [ออนไลน์].** ที่มา : http://th.wikipedia.org/wiki/การแทนความรู้

วิกิพีเดีย สารานุกรมเสรี (2554ข). **ปัญญาประดิษฐ์ [ออนไลน์].** ที่มา : http://th.wikipedia.org/wiki/ปัญญาประดิษฐ์

วิกิพีเดีย สารานุกรมเสรี (2554ค). **ภาษาโปรล็อก [ออนไลน์].** ที่มา : http://th.wikipedia.org/wiki/ภาษาโปรล็อก

วิภาดา เวทย์ประสิทธิ์ (2547). Expert System. **เอกสารประกอบการบรรยาย วิชาปัญญาประดิษฐ์ประจำภาควิชาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยสงขลานครินทร์ [ออนไลน์].** ที่มา : http://staff.cs.psu.ac.th/wiphada/Teaching%20Coureses/sem%202-2547/344-471%20AI-ES/Present-AI/Expert%20System/Expert%20System%C3%C7%C1.ppt

วิลาศ วูวงศ์และบุญเจริญ ศิริเนาวกุล (2535). **ระบบผู้เชี่ยวชาญ Expert System.** กรุงเทพฯ : ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ

อรอนงค์ อร วันปู (2554). **กลไกอนุมาน (Inference engine) หมายถึง กลไกลอนุมานได้รับการพัฒนาอย่างไร [ออนไลน์].** ที่มา : http://www.learners.in.th/blog/qqw

Anwar, Z. (2011). **Forward and Backward Chaining [Online].** Available URL : www.docstoc.com/docs/5693324/Forward-and-Backward-Chaining

Apt, K.R. and Wallance, M. (2006). **Constraint Logic Programming using ECLiPSe [Online].** Available URL : http://araku.ac.ir/~d_comp_engineering/88892/1323163/Eclipse.pdf

Bajpai, A. and Marczewski, R. (1989). Charley : An Expert System for Diagnostics of Manufacturing Equipment. Innovative Applications of Artificial Intelligence. AAAI Press/MIT Press.

Buchanan, B.G. and Shortliffe, E.H. (1984). **Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.** Reading, MA, Addison-Wesley.

Buchanan, B.G. and Smith, R.G. (1989). Fundamentals of Expert Systems. **The Handbook of Artificial Intelligence.** Vol. IV. Addison-Wesley.

Chakraborty, R.C. (2010). **Expert System [Online].** Available URL : http://www.myreaders.info/07-Expert_Systems.pdf

ECLiPSe (2012). **ECLiPSe Features [Online].** Available URL : http://eclipseclp.org/features.html

Eronen, P. and Zitting, J. (2001). An Expert System for Analyzing Firewall Rules. **CiteSeerX Proc. Conf.**

Feigenbaum, E.A. (1992). **Expert Systems: Principles and Practice* [Online].** Available URL : http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.34.9207

Feigenbaum, E., McCorduck, P. and Nii, H. P. (1988). The Rise of the Expert Company. **Times Books.** New York.

Jackson, P. (1999). **Introduction to Expert Systems.** Harlow, England., Addison-Wesley.

Kantrowitz, M. (1997a). [1-5a] Free/Cheap Expert System Shells. **FAQ: Expert System Shells 1/1 [Online].** Available URL : http://www.cs.cmu.edu/Groups/AI/html/faqs/ai/expert/part1/faq-doc-5.html

Kantrowitz, M. (1997b). [1-5b] Free/Cheap Expert System Shells: CLIPS and Related Systems. **FAQ: Expert System Shells 1/1 [Online].** Available URL : http://www.cs.cmu.edu/Groups/AI/html/faqs/ai/expert/part1/faq-doc-6.html

Kantrowitz, M. (1997c). [1-6] Commercial Expert System Shells. **FAQ: Expert System Shells 1/1 [Online].** Available URL : http://www.cs.cmu.edu/Groups/AI/html/faqs/ai/expert/part1/faq-doc-7.html

Lee, J.K. and Kwon, S.B. (1995). ES*: An Expert Systems Development Planner Using a Constraint and Rule-Based Approach. **Expert Systems With Applications,** Vol. 9, No. I, pp. 3-14.

Lindsay, R.K, Buchanan, B.G., Feigenbaum, E.A. and Lederberg, J. (1980). **Applications of Artificial Intelligence for Organic Chemistry: The DENDRAL Project.** McGraw-Hill.

Melle, V.W., Shortliffe, E.H. and Buchanan, B.G. (1984). **"EMYCIN: A Knowledge engineer's Tool for Constructing Rule-Based Expert Systems," in Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.** Addison-Wesley.

Misiaszek, M. (2007a). **Expert Systems: Principles and [Online].** Available URL : http://www. slideshare.net/artintelligence/expert-systems-expert-systems-principles-and/

Misiaszek, M. (2007b). The Representation of Knowledge. **Expert Systems : Principles and Programming, Fourth Edition [Online].** Available URL : http://www.csie.ntu.edu.tw/ ~sylee/courses/clips/PowerPoint/chapter02.ppt

Muli, H.K. (1992). Optimization Methods in Logic Programming Applied to Expert Systems for Capital Budgeting. **Masters of Science Thesis,** Systems Science Programme, University of Ottawa, Canada.

Niederlinski, A. (2011). **CLP with Elementary Constraints for Feasible Solutions. A Quick and Gentle Guide to Constraint Logic Programming via ECLiPSe.** Page : 103-104.

No-poor (2010). **Inference Techniques [Online].** Available URL : http://www.no-poor.com/Java andAi/bookAI/week7InferenceTechniques_Finanl.pdf

Rattanapoka, C. (2010). **Knowledge Representation [Online].** Available URL : http://ect.cit. kmutnb.ac.th/~choopan/2010/357353/07-KR.ppt

Roach, D. and Berghel, H. (1992). An Alternative to Constraint Logic Programming for Managing Domain Arithmetics in Prolog Expert Systems. **ACM Proc. Conf. SIGAPP Symposium on Applied Computing: Technological Challenges of the 1990's,** pp. 328 - 332.

Saad, M.K. (2008**). Introduction to CLIPS Expert System [Online].** Available URL : http:// www.slideshare.net/mksaad/introduction-to-clips-expert-system-presentation

Salim, M.D., Villavicencio, A. and Timmerman, M.A. (2002). A Method for Evaluating Expert System Shells for Classroom Instruction. **Journal of Industrial Technology,** Vol. 19, No. I, pp. 1-11.

W3dictionary (2012a). **Compiler [Online].** Available URL : http://th.w3dictionary.org/index.php?q=compiler

W3dictionary (2012b). **Incremental Compiler [Online].** Available URL : http://th.w3dictionary.org/index.php?q=incremental+compiler

W3dictionary (2012c). **Interpreter [Online].** Available URL : http://th.w3dictionary.org/index.php?q=interpreter

Wikipedia, The Free Encyclopedia (2011a). **Backward Chaining [Online].** Available URL : http://en.wikipedia.org/wiki/Backward_chaining

Wikipedia, The Free Encyclopedia (2011b). **CLIPS [Online].** Available URL : http://en.wikipedia.org/wiki/CLIPS

Wikipedia, The Free Encyclopedia (2011c). **Constraint Logic Programming [Online].** Available URL : http://en.wikipedia.org/wiki/Constraint_logic_programming

Wikipedia, The Free Encyclopedia (2011d). **ECLiPSe [Online].** Available URL : http://en.wikipedia.org/wiki/ECLiPSe

Wikipedia, The Free Encyclopedia (2011e). **Eclipse (software) [Online].** Available URL : http://en.wikipedia.org/wiki/Eclipse_(software)

Wikipedia, The Free Encyclopedia (2011f). **Expert System [Online].** Available URL : http://en.wikipedia.org/wiki/Expert_system

Wikipedia, The Free Encyclopedia (2011g). **Forward Chaining [Online].** Available URL : http://en.wikipedia.org/wiki/Forward_chaining

Wikipedia, The Free Encyclopedia (2011h). **Prolog [Online].** Available URL : http://en.wikipedia.org/wiki/Prolog

ภาคผนวก ก

โค้ดเปลือกระบบผู้เชี่ยวชาญภาษาโปรล็อก

```
:- dynamic known/1.
:- dynamic answer/1.

expertshell :-
        greeting, repeat,
        writeln('expert-shell> '),
        read(X),do(X),
        X==quit,
        writeln('>>>>Goodbye, see you later<<<<'), !.

greeting :-
        write('This is the Easy Expert System shell.'), nl,
        native_help.

do(help) :- native_help, !.
do(load) :- load_kb, !.
do(solve) :- solve, !.
do(why) :- why, !.
do(quit).
do(X) :-        write(X),
        write(' is not a legal command.'), nl,
        fail.

native_help :-
        write('Type help. load. solve. why. or quit.'), nl,
        write('at the prompt.'), nl.

load_kb :-
        writeln('Enter file name in single qoutes (ex. "vacation.knb".) : '),
        read(F),
        compile(F).

solve :-
        retractall(known(_)), retractall(answer(_)),
        time(top_goal(X)),
        write('The answer is '), write(X), assert(answer(X)), nl.
solve :-
        write('No answer found. '), nl.

menuask(Pred, Value, Menu) :- menuask(Pred,Menu), known([Pred,Value]), !.

menuask(Pred,_) :- known([Pred,_]), !.

menuask(Attribute,Menu) :-
        write('What is the value for '), write(Attribute), write('?'), nl,
        writeln(Menu), write('Enter the choice > '), read(V),writeln(V),
        asserta(known([Attribute,V])).

why :- answer(A), write('The answer is '), writeln(A),
        findall( X, known(X), Result), writeln('The known storage are'),
        writeln(Result).
```

ภาคผนวก ข

โค้ดเปลือกระบบผู้เชี่ยวชาญที่พัฒนาด้วยวิธีเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ

```
:-lib(sd).
:- dynamic known/1.
:- dynamic answer/1.

expertshell :-
        greeting, repeat,
        writeln('expert-shell> '),
        read(X),do(X),
        X==quit,
        writeln('>>>>Goodbye, see you later<<<<'), !.

greeting :-
        write('This is the Easy Expert System shell.'), nl,
        native_help.

do(help) :- native_help, !.
do(load) :- load_kb, !.
do(solve) :- solve, !.
do(why) :- why, !.
do(quit).
do(X) :-        write(X),
        write(' is not a legal command.'), nl,
        fail.

native_help :-
        write('Type help. load. solve. why. or quit.'), nl,
        write('at the prompt.'), nl.

load_kb :-
        writeln('Enter file name in single qoutes (ex. "vacation.knb".) : '),
        read(F),
        compile(F).

solve :-
        retractall(known(_)), retractall(answer(_)),
        time(top_goal(X)),
        write('The answer is '), write(X), assert(answer(X)), nl.
solve :-
        write('No answer found. '), nl.

menuask(Pred,Value,Menu):- C&::Menu, menuask(Pred,C),  known([Pred,Value]),!.

menuask(Pred,_) :- known([Pred,_]),!.

menuask(Attribute,C) :-
        write('What is the value for '), write(Attribute), writeln('?'),
        writeln(C), write('Enter the choice > '), read(C),writeln(C),
        asserta(known([Attribute,C])).

why :- answer(A), write('The answer is '), writeln(A),
        findall( X, known(X), Result), writeln('The known storage are'),
        writeln(Result).
```

ภาคผนวก ค

โค้ดฐานความรู้ Nursery ในรูปแบบภาษาโปรล็อก

```
health(X):-time(menuask(health,X,[recommended,priority,not_recom])).
has_nurs(X):-time(menuask(has_nurs,X,[proper,less_proper,improper,critical,very_crit])).
parents(X):-time(menuask(parents,X,[usual,pretentious,great_pret])).
social(X):-time(menuask(social,X,[nonprob,slightly_prob,problematic])).
housing(X):-time(menuask(housing,X,[convenient,less_conv,critical])).
finance(X):-time(menuask(finance,X,[convenient,inconv])).
form(X):-time(menuask(form,X,[complete,completed,incomplete,foster])).
children(X):-time(menuask(children,X,[1,2,3,more])).

top_goal(X):-class(X).

%1%
class(not_recom):-health(not_recom).

%3%
        %p%
class(priority):-health(priority),(has_nurs(proper);has_nurs(less_proper)),
                (parents(usual);parents(pretentious)).
class(priority):-health(priority),has_nurs(improper),parents(usual).
        %sp%
class(spec_prior):-health(priority),has_nurs(very_crit),
                   (form(completed);form(incomplete);form(foster)).

%4%
        %p%
class(priority):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                parents(great_pret),(social(nonprob);social(slightly_prob)).
class(priority):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                (parents(usual);parents(pretentious)),social(problematic).
class(priority):-health(recommended),has_nurs(improper),parents(usual),social(problematic).
class(priority):-health(recommended),has_nurs(improper),parents(pretentious),
                (social(nonprob);social(slightly_prob)).
class(priority):-health(recommended),has_nurs(critical),parents(usual),
                (social(nonprob);social(slightly_prob)).
        %sp%
class(spec_prior):-health(priority),has_nurs(very_crit),form(complete),
                   (children(2);children(3);children(more)).
class(spec_prior):-health(recommended),has_nurs(very_crit),social(problematic),
                   (form(completed);form(incomplete);form(foster)).
class(spec_prior):-health(priority),has_nurs(improper),parents(great_pret),
                   (form(completed);form(incomplete);form(foster)).
class(spec_prior):-health(priority),has_nurs(critical),(parents(pretentious);parents(great_pret)),
                   (form(completed);form(incomplete);form(foster)).

%5%
        %p%
```

```
class(priority):-health(recommended),has_nurs(very_crit),
                 (social(nonprob);social(slightly_prob)),housing(convenient),
                 finance(convenient).
class(priority):-health(priority),(has_nurs(proper);has_nurs(less_proper)),
                 parents(great_pret),housing(convenient),finance(convenient).
class(priority):-health(priority),has_nurs(improper),parents(pretentious),housing(convenient),
                 finance(convenient).
class(priority):-health(priority),has_nurs(critical),parents(usual),housing(convenient),
                 finance(convenient).
        %sp%
class(spec_prior):-health(priority),has_nurs(very_crit),form(complete),children(1),
                   (housing(less_conv);housing(critical)).
class(spec_prior):-health(priority),(has_nurs(proper);has_nurs(less_proper)),parents(great_pret),
                   housing(critical),(form(completed);form(incomplete);form(foster)).
class(spec_prior):-health(priority),has_nurs(improper),parents(pretentious),housing(critical),
                   (form(completed);form(incomplete);form(foster)).
class(spec_prior):-health(priority),has_nurs(critical),parents(usual),housing(critical),
                   (form(completed);form(incomplete);form(foster)).
class(spec_prior):-health(priority),has_nurs(improper),parents(great_pret),form(complete),
                   (children(2);children(3);children(more)).
class(spec_prior):-health(priority),has_nurs(critical),(parents(pretentious);parents(great_pret)),
                   form(complete),(children(2);children(3);children(more)).
class(spec_prior):-health(priority),(has_nurs(proper);has_nurs(less_proper)),parents(great_pret),
                   housing(less_conv),(children(3);children(more)).
class(spec_prior):-health(priority),has_nurs(improper),parents(pretentious),housing(less_conv),
                   (children(3);children(more)).
class(spec_prior):-health(priority),has_nurs(critical),parents(usual),housing(less_conv),
                   (children(3);children(more)).
class(spec_prior):-health(recommended),has_nurs(improper),parents(great_pret),
                   social(problematic),(form(complete);form(completed);form(incomplete)).
class(spec_prior):-health(recommended),has_nurs(critical),
                   (parents(pretentious);parents(great_pret)),social(problematic),
                   (form(complete);form(completed);form(incomplete)).
class(spec_prior):-health(recommended),has_nurs(very_crit),social(problematic),
                   form(complete),(children(2);children(3);children(more)).
class(spec_prior):-health(recommended),has_nurs(very_crit),
                   (social(nonprob);social(slightly_prob)),housing(less_conv),
                   (children(3);children(more)).
class(spec_prior):-health(recommended),has_nurs(very_crit),
                   (social(nonprob);social(slightly_prob)),housing(critical),
                   (form(complete);form(completed);form(incomplete)).

%6%
        %p%
class(priority):-health(priority),has_nurs(very_crit),form(complete),children(1),
                 housing(convenient),finance(convenient).
```

```
class(priority):-health(priority),(has_nurs(proper);has_nurs(less_proper)),parents(great_pret),
                    housing(critical),form(complete),children(1).
class(priority):-health(priority),has_nurs(improper),parents(pretentious),housing(critical),
                    form(complete),children(1).
class(priority):-health(priority),has_nurs(critical),parents(usual),housing(critical),
                    form(complete),children(1).
class(priority):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                    (parents(usual);parents(pretentious)),(social(nonprob);social(slightly_prob)),
                    housing(less_conv),(children(3);children(more)).
class(priority):-health(recommended),has_nurs(improper),parents(usual),
                    (social(nonprob);social(slightly_prob)),housing(less_conv),
                    (children(3);children(more)).
class(priority):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                    (parents(usual);parents(pretentious)),(social(nonprob);social(slightly_prob)),
                    housing(critical),(form(completed);form(incomplete);form(foster)).
class(priority):-health(recommended),has_nurs(improper),parents(usual),
                    (social(nonprob);social(slightly_prob)),housing(critical),
                    (form(completed);form(incomplete);form(foster)).
class(priority):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                    parents(great_pret),social(problematic),housing(convenient),
                    finance(convenient).
class(priority):-health(recommended),has_nurs(improper),parents(pretentious),
                    social(problematic),housing(convenient),finance(convenient).
class(priority):-health(recommended),has_nurs(improper),parents(great_pret),
                    (social(nonprob);social(slightly_prob)),housing(convenient),
                    finance(convenient).
class(priority):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
                    housing(convenient),finance(convenient).
class(priority):-health(recommended),has_nurs(critical),
                    (parents(pretentious);parents(great_pret)),
                    (social(nonprob);social(slightly_prob)),housing(convenient),
                    finance(convenient).
class(priority):-health(priority),(has_nurs(proper);has_nurs(less_proper)),parents(great_pret),
                    housing(less_conv),children(1),
                    (form(complete);form(completed);form(incomplete)).
class(priority):-health(priority),(has_nurs(proper);has_nurs(less_proper)),parents(great_pret),
                    housing(less_conv),children(2),(form(complete);form(completed)).
class(priority):-health(priority),has_nurs(improper),parents(pretentious),housing(less_conv),
                    children(1),(form(complete);form(completed);form(incomplete)).
class(priority):-health(priority),has_nurs(improper),parents(pretentious),housing(less_conv),
                    children(2),(form(complete);form(completed)).
class(priority):-health(priority),has_nurs(critical),parents(usual),housing(less_conv),children(1),
                    (form(complete);form(completed);form(incomplete)).
class(priority):-health(priority),has_nurs(critical),parents(usual),housing(less_conv),children(2),
                    (form(complete);form(completed)).
```

```
class(priority):-health(recommended),has_nurs(very_crit),
                    (social(nonprob);social(slightly_prob)),housing(less_conv),children(1),
                    (form(complete);form(completed);form(incomplete)).
class(priority):-health(recommended),has_nurs(very_crit),
                    (social(nonprob);social(slightly_prob)),housing(less_conv),children(2),
                    (form(complete);form(completed)).
class(priority):-health(recommended),has_nurs(very_crit),
                    (social(nonprob);social(slightly_prob)),housing(critical),form(complete),
                    children(1).
          %sp%
class(spec_prior):-health(priority),has_nurs(very_crit),form(complete),children(1),
                      housing(convenient),finance(inconv).
class(spec_prior):-health(priority),(has_nurs(proper);has_nurs(less_proper)),parents(great_pret),
                      housing(critical),form(complete),(children(2);children(3);children(more)).
class(spec_prior):-health(priority),has_nurs(improper),parents(pretentious),housing(critical),
                      form(complete),(children(2);children(3);children(more)).
class(spec_prior):-health(priority),has_nurs(critical),parents(usual),housing(critical),
                      form(complete),(children(2);children(3);children(more)).
class(spec_prior):-health(recommended),has_nurs(very_crit),
                      (social(nonprob);social(slightly_prob)),housing(convenient),
                      finance(inconv),(children(3);children(more)).
class(spec_prior):-health(priority),has_nurs(improper),parents(great_pret),form(complete),
                      children(1),(housing(less_conv);housing(critical)).
class(spec_prior):-health(priority),has_nurs(critical),(parents(pretentious);parents(great_pret)),
                      form(complete),children(1),(housing(less_conv);housing(critical)).
class(spec_prior):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                      parents(great_pret),social(problematic),housing(less_conv),
                      (children(3);children(more)).
class(spec_prior):-health(recommended),has_nurs(improper),parents(pretentious),
                      social(problematic),(housing(convenient);housing(less_conv)),
                      (children(3);children(more)).
class(spec_prior):-health(recommended),has_nurs(improper),parents(great_pret),
                      (social(nonprob);social(slightly_prob)),housing(less_conv),
                      (children(3);children(more)).
class(spec_prior):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
                      housing(less_conv),(children(3);children(more)).
class(spec_prior):-health(recommended),has_nurs(critical),
                      (parents(pretentious);parents(great_pret)),
                      (social(nonprob);social(slightly_prob)),housing(less_conv),
                      (children(3);children(more)).
class(spec_prior):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                      parents(great_pret),social(problematic),housing(critical),
                      (form(completed);form(incomplete);form(foster)).
class(spec_prior):-health(recommended),has_nurs(improper),parents(pretentious),
                      social(problematic),housing(critical),
                      (form(completed);form(incomplete);form(foster)).
```

```
class(spec_prior):-health(recommended),has_nurs(improper),parents(great_pret),
                  (social(nonprob);social(slightly_prob)),housing(critical),
                  (form(completed);form(incomplete);form(foster)).
class(spec_prior):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
                  housing(critical),(form(completed);form(incomplete);form(foster)).
class(spec_prior):-health(recommended),has_nurs(critical),
                   (parents(pretentious);parents(great_pret)),
                   (social(nonprob);social(slightly_prob)),housing(critical),
                   (form(completed);form(incomplete);form(foster)).
class(spec_prior):-health(priority),(has_nurs(proper);has_nurs(less_proper)),
                  parents(great_pret),housing(convenient),finance(inconv),
                  (children(3);children(more)).
class(spec_prior):-health(priority),has_nurs(improper),parents(pretentious),
                  housing(convenient),finance(inconv),(children(3);children(more)).
class(spec_prior):-health(priority),has_nurs(critical),parents(usual),housing(convenient),
                  finance(inconv),(children(3);children(more)).
class(spec_prior):-health(priority),(has_nurs(proper);has_nurs(less_proper)),parents(great_pret),
                  housing(less_conv),children(1),form(foster).
class(spec_prior):-health(priority),(has_nurs(proper);has_nurs(less_proper)),parents(great_pret),
                  housing(less_conv),children(2),(form(incomplete);form(foster)).
class(spec_prior):-health(priority),has_nurs(improper),parents(pretentious),housing(less_conv),
                  children(1),form(foster).
class(spec_prior):-health(priority),has_nurs(improper),parents(pretentious),housing(less_conv),
                  children(2),(form(incomplete);form(foster)).
class(spec_prior):-health(priority),has_nurs(critical),parents(usual),housing(less_conv),
                  children(1),form(foster).
class(spec_prior):-health(priority),has_nurs(critical),parents(usual),housing(less_conv),
                  children(2),(form(incomplete);form(foster)).
class(spec_prior):-health(recommended),has_nurs(improper),parents(great_pret),
                  social(problematic),form(complete),
                  (children(2);children(3);children(more)).
class(spec_prior):-health(recommended),has_nurs(critical),
                   (parents(pretentious);parents(great_pret)),social(problematic),
                  form(complete),(children(2);children(3);children(more)).
class(spec_prior):-health(recommended),has_nurs(very_crit),social(problematic),
                  form(complete),children(1),(housing(less_conv);housing(critical)).

class(spec_prior):-health(recommended),has_nurs(very_crit),
                  (social(nonprob);social(slightly_prob)),housing(less_conv),
                  children(1),form(foster).
class(spec_prior):-health(recommended),has_nurs(very_crit),
                  (social(nonprob);social(slightly_prob)),housing(less_conv),children(2),
                  (form(incomplete);form(foster)).
class(spec_prior):-health(recommended),has_nurs(very_crit),
                  (social(nonprob);social(slightly_prob)),housing(critical),form(complete),
                  (children(2);children(3);children(more)).
```

```
        %vr%
class(very_recom):-health(recommended),has_nurs(proper),parents(pretentious),
                        (social(nonprob);social(slightly_prob)),housing(convenient),
                        finance(convenient).
class(very_recom):-health(recommended),has_nurs(less_proper),
                        (parents(usual);parents(pretentious)),
                        (social(nonprob);social(slightly_prob)),housing(convenient),
                        finance(convenient).
class(very_recom):-health(recommended),has_nurs(improper),parents(usual),
                        (social(nonprob);social(slightly_prob)),housing(convenient),
                        finance(convenient).

%7%
        %p%
class(priority):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                    (parents(usual);parents(pretentious)),
                    (social(nonprob);social(slightly_prob)),housing(convenient),finance(inconv),
                    (children(3);children(more)).
class(priority):-health(recommended),has_nurs(improper),parents(usual),
                    (social(nonprob);social(slightly_prob)),housing(convenient),
                    finance(inconv),(children(3);children(more)).
class(priority):-health(recommended),has_nurs(very_crit),
                    (social(nonprob);social(slightly_prob)),housing(convenient),finance(inconv),
                    children(1),(form(complete);form(completed);form(incomplete)).
class(priority):-health(recommended),has_nurs(very_crit),
                    (social(nonprob);social(slightly_prob)),housing(convenient),finance(inconv),
                    children(2),(form(complete);form(completed)).
class(priority):-health(priority),has_nurs(improper),parents(great_pret),form(complete),
                    children(1),housing(convenient),finance(convenient).
class(priority):-health(priority),has_nurs(critical),(parents(pretentious);parents(great_pret)),
                    form(complete),children(1),housing(convenient),finance(convenient).
class(priority):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                    (parents(usual);parents(pretentious)),(social(nonprob);social(slightly_prob)),
                    housing(less_conv),children(1),form(foster).
class(priority):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                    (parents(usual);parents(pretentious)),(social(nonprob);social(slightly_prob)),
                    housing(less_conv),children(2),(form(incomplete);form(foster)).
class(priority):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                    parents(great_pret),social(problematic),housing(less_conv),children(1),
                    (form(complete);form(completed);form(incomplete)).
class(priority):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                    parents(great_pret),social(problematic),housing(less_conv),children(2),
                    (form(complete);form(completed)).
class(priority):-health(recommended),has_nurs(improper),parents(usual),
                    (social(nonprob);social(slightly_prob)),housing(less_conv),children(1),
                    form(foster).
```

```
class(priority):-health(recommended),has_nurs(improper),parents(usual),
        (social(nonprob);social(slightly_prob)),housing(less_conv),children(2),
        (form(incomplete);form(foster)).
class(priority):-health(recommended),has_nurs(improper),parents(pretentious),
        social(problematic),(housing(convenient);housing(less_conv)),children(1),
        (form(complete);form(completed);form(incomplete)).
class(priority):-health(recommended),has_nurs(improper),parents(pretentious),
        social(problematic),(housing(convenient);housing(less_conv)),children(2),
        (form(complete);form(completed)).
class(priority):-health(recommended),has_nurs(improper),parents(great_pret),
        (social(nonprob);social(slightly_prob)),housing(less_conv),children(1),
        (form(complete);form(completed);form(incomplete)).
class(priority):-health(recommended),has_nurs(improper),parents(great_pret),
        (social(nonprob);social(slightly_prob)),housing(less_conv),children(2),
        (form(complete);form(completed)).
class(priority):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
        housing(less_conv),children(1),
        (form(complete);form(completed);form(incomplete)).
class(priority):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
        housing(less_conv),children(2),(form(complete);form(completed)).
class(priority):-health(recommended),has_nurs(critical),
        (parents(pretentious);parents(great_pret)),
        (social(nonprob);social(slightly_prob)),housing(less_conv),children(1),
        (form(complete);form(completed);form(incomplete)).
class(priority):-health(recommended),has_nurs(critical),
        (parents(pretentious);parents(great_pret)),
        (social(nonprob);social(slightly_prob)),housing(less_conv),children(2),
        (form(complete);form(completed)).
class(priority):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
        (parents(usual);parents(pretentious)),
        (social(nonprob);social(slightly_prob)),housing(critical),form(complete),
        (children(2);children(3);children(more)).
class(priority):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
        parents(great_pret),social(problematic),housing(critical),form(complete),
        children(1).
class(priority):-health(recommended),has_nurs(improper),parents(usual),
        (social(nonprob);social(slightly_prob)),housing(critical),form(complete),
        (children(2);children(3);children(more)).
class(priority):-health(recommended),has_nurs(improper),parents(pretentious),
        social(problematic),housing(critical),form(complete),children(1).
class(priority):-health(recommended),has_nurs(improper),parents(great_pret),
        (social(nonprob);social(slightly_prob)),housing(critical),form(complete),
        children(1).
class(priority):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
        housing(critical),form(complete),children(1).
```

```
class(priority):-health(recommended),has_nurs(critical),
                    (parents(pretentious);parents(great_pret)),
                    (social(nonprob);social(slightly_prob)),
                    housing(critical),form(complete),children(1).
class(priority):-health(priority),(has_nurs(proper);has_nurs(less_proper)),
                    parents(great_pret),housing(convenient),finance(inconv),children(1),
                    (form(complete);form(completed);form(incomplete)).
class(priority):-health(priority),(has_nurs(proper);has_nurs(less_proper)),
                    parents(great_pret),housing(convenient),finance(inconv),children(2),
                    (form(complete);form(completed)).
class(priority):-health(priority),has_nurs(improper),parents(pretentious),housing(convenient),
                    finance(inconv),children(1),
                    (form(complete);form(completed);form(incomplete)).
class(priority):-health(priority),has_nurs(improper),parents(pretentious),housing(convenient),
                    finance(inconv),children(2),(form(complete);form(completed)).
class(priority):-health(priority),has_nurs(critical),parents(usual),housing(convenient),
                    finance(inconv),children(1),
                    (form(complete);form(completed);form(incomplete)).
class(priority):-health(priority),has_nurs(critical),parents(usual),housing(convenient),
                    finance(inconv),children(2),(form(complete);form(completed)).
class(priority):-health(recommended),has_nurs(very_crit),social(problematic),form(complete),
                    children(1),housing(convenient),finance(convenient).
          %sp%
class(spec_prior):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                      parents(great_pret),social(problematic),housing(convenient),
                      finance(inconv),(children(3);children(more)).
class(spec_prior):-health(recommended),has_nurs(improper),parents(great_pret),
                      (social(nonprob);social(slightly_prob)),housing(convenient),
                      finance(inconv),(children(3);children(more)).
class(spec_prior):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
                      housing(convenient),finance(inconv),(children(3);children(more)).
class(spec_prior):-health(recommended),has_nurs(critical),
                      (parents(pretentious);parents(great_pret)),
                      (social(nonprob);social(slightly_prob)),housing(convenient),
                      finance(inconv),(children(3);children(more)).
class(spec_prior):-health(recommended),has_nurs(very_crit),
                      (social(nonprob);social(slightly_prob)),housing(convenient),
                      finance(inconv),children(1),form(foster).
class(spec_prior):-health(recommended),has_nurs(very_crit),
                      (social(nonprob);social(slightly_prob)),housing(convenient),
                      finance(inconv),children(2),(form(incomplete);form(foster)).
class(spec_prior):-health(priority),has_nurs(improper),parents(great_pret),form(complete),
                      children(1),housing(convenient),finance(inconv).
class(spec_prior):-health(priority),has_nurs(critical),(parents(pretentious);parents(great_pret)),
                      form(complete),children(1),housing(convenient),finance(inconv).
```

```
class(spec_prior):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                   parents(great_pret),social(problematic),housing(less_conv),children(1),
                   form(foster).
class(spec_prior):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                   parents(great_pret),social(problematic),housing(less_conv),children(2),
                   (form(incomplete);form(foster)).
class(spec_prior):-health(recommended),has_nurs(improper),parents(pretentious),
                   social(problematic),(housing(convenient);housing(less_conv)),children(1),
                   form(foster).
class(spec_prior):-health(recommended),has_nurs(improper),parents(pretentious),
                   social(problematic),(housing(convenient);housing(less_conv)),children(2),
                   (form(incomplete);form(foster)).
class(spec_prior):-health(recommended),has_nurs(improper),parents(great_pret),
                   (social(nonprob);social(slightly_prob)),housing(less_conv),children(1),
                   form(foster).
class(spec_prior):-health(recommended),has_nurs(improper),parents(great_pret),
                   (social(nonprob);social(slightly_prob)),housing(less_conv),children(2),
                   (form(incomplete);form(foster)).
class(spec_prior):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
                   housing(less_conv),children(1),form(foster).
class(spec_prior):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
                   housing(less_conv),children(2),(form(incomplete);form(foster)).
class(spec_prior):-health(recommended),has_nurs(critical),
                   (parents(pretentious);parents(great_pret)),
                   (social(nonprob);social(slightly_prob)),housing(less_conv),children(1),
                   form(foster).
class(spec_prior):-health(recommended),has_nurs(critical),
                   (parents(pretentious);parents(great_pret)),
                   (social(nonprob);social(slightly_prob)),housing(less_conv),children(2),
                   (form(incomplete);form(foster)).
class(spec_prior):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                   parents(great_pret),social(problematic),housing(critical),form(complete),
                   (children(2);children(3);children(more)).
class(spec_prior):-health(recommended),has_nurs(improper),parents(pretentious),
                   social(problematic),housing(critical),form(complete),
                   (children(2);children(3);children(more)).
class(spec_prior):-health(recommended),has_nurs(improper),parents(great_pret),
                   (social(nonprob);social(slightly_prob)),housing(critical),form(complete),
                   (children(2);children(3);children(more)).
class(spec_prior):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
                   housing(critical),form(complete),(children(2);children(3);children(more)).
class(spec_prior):-health(recommended),has_nurs(critical),
                   (parents(pretentious);parents(great_pret)),
                   (social(nonprob);social(slightly_prob)),housing(critical),form(complete),
                   (children(2);children(3);children(more)).
class(spec_prior):-health(priority),(has_nurs(proper);has_nurs(less_proper)),parents(great_pret),
                   housing(convenient),finance(inconv),children(1),form(foster).
```

```
class(spec_prior):-health(priority),(has_nurs(proper);has_nurs(less_proper)),parents(great_pret),
                   housing(convenient),finance(inconv),children(2),
                   (form(incomplete);form(foster)).
class(spec_prior):-health(priority),has_nurs(improper),parents(pretentious),
                   housing(convenient),finance(inconv),children(1),form(foster).
class(spec_prior):-health(priority),has_nurs(improper),parents(pretentious),
                   housing(convenient),finance(inconv),children(2),
                   (form(incomplete);form(foster)).
class(spec_prior):-health(priority),has_nurs(critical),parents(usual),housing(convenient),
                   finance(inconv),children(1),form(foster).
class(spec_prior):-health(priority),has_nurs(critical),parents(usual),housing(convenient),
                   finance(inconv),children(2),(form(incomplete);form(foster)).
class(spec_prior):-health(recommended),has_nurs(improper),parents(great_pret),
                   social(problematic),form(complete),children(1),
                   (housing(less_conv);housing(critical)).
class(spec_prior):-health(recommended),has_nurs(critical),
                   (parents(pretentious);parents(great_pret)),social(problematic),
                   form(complete),children(1),(housing(less_conv);housing(critical)).
class(spec_prior):-health(recommended),has_nurs(very_crit),social(problematic),
                   form(complete),children(1),housing(convenient),finance(inconv).
       %vr%
class(very_recom):-health(recommended),has_nurs(proper),parents(usual),
                    (social(nonprob);social(slightly_prob)),housing(convenient),
                    finance(convenient),(form(completed);form(incomplete);form(foster)).
class(very_recom):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                    (parents(usual);parents(pretentious)),
                    (social(nonprob);social(slightly_prob)),housing(less_conv),children(1),
                    (form(complete);form(completed);form(incomplete)).
class(very_recom):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                    (parents(usual);parents(pretentious)),
                    (social(nonprob);social(slightly_prob)),housing(less_conv),children(2),
                    (form(complete);form(completed)).
class(very_recom):-health(recommended),has_nurs(improper),parents(usual),
                    (social(nonprob);social(slightly_prob)),housing(less_conv),children(1),
                    (form(complete);form(completed);form(incomplete)).
class(very_recom):-health(recommended),has_nurs(improper),parents(usual),
                    (social(nonprob);social(slightly_prob)),housing(less_conv),children(2),
                    (form(complete);form(completed)).
class(very_recom):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                    (parents(usual);parents(pretentious)),
                    (social(nonprob);social(slightly_prob)),housing(critical),form(complete),
                    children(1).
class(very_recom):-health(recommended),has_nurs(improper),parents(usual),
                    (social(nonprob);social(slightly_prob)),housing(critical),form(complete),
                    children(1).
```

```
%8%
        %p%
class(priority):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                (parents(usual);parents(pretentious)),(social(nonprob);social(slightly_prob)),
                housing(convenient),finance(inconv),children(1),form(foster).
class(priority):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                (parents(usual);parents(pretentious)),(social(nonprob);social(slightly_prob)),
                housing(convenient),finance(inconv),children(2),
                (form(incomplete);form(foster)).
class(priority):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                parents(great_pret),social(problematic),housing(convenient),finance(inconv),
                children(1),(form(complete);form(completed);form(incomplete)).
class(priority):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                parents(great_pret),social(problematic),housing(convenient),finance(inconv),
                children(2),(form(complete);form(completed)).
class(priority):-health(recommended),has_nurs(improper),parents(usual),
                (social(nonprob);social(slightly_prob)),housing(convenient),finance(inconv),
                children(1),form(foster).
class(priority):-health(recommended),has_nurs(improper),parents(usual),
                (social(nonprob);social(slightly_prob)),housing(convenient),finance(inconv),
                children(2),(form(incomplete);form(foster)).
class(priority):-health(recommended),has_nurs(improper),parents(great_pret),
                (social(nonprob);social(slightly_prob)),housing(convenient),finance(inconv),
                children(1),(form(complete);form(completed);form(incomplete)).
class(priority):-health(recommended),has_nurs(improper),parents(great_pret),
                (social(nonprob);social(slightly_prob)),housing(convenient),finance(inconv),
                children(2),(form(complete);form(completed)).
class(priority):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
                housing(convenient),finance(inconv),children(1),
                (form(complete);form(completed);form(incomplete)).
class(priority):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
                housing(convenient),finance(inconv),children(2),
                (form(complete);form(completed)).
class(priority):-health(recommended),has_nurs(critical),
                (parents(pretentious);parents(great_pret)),
                (social(nonprob);social(slightly_prob)),housing(convenient),finance(inconv),
                children(1),(form(complete);form(completed);form(incomplete)).
class(priority):-health(recommended),has_nurs(critical),
                (parents(pretentious);parents(great_pret)),
                (social(nonprob);social(slightly_prob)),housing(convenient),finance(inconv),
                children(2),(form(complete);form(completed)).
class(priority):-health(recommended),has_nurs(improper),parents(great_pret),
                social(problematic),form(complete),children(1),housing(convenient),
                finance(convenient).
class(priority):-health(recommended),has_nurs(critical),
                (parents(pretentious);parents(great_pret)),social(problematic),form(complete),
                children(1),housing(convenient),finance(convenient).
```

```
          %sp%
class(spec_prior):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                   parents(great_pret),social(problematic),housing(convenient),
                   finance(inconv),children(1),form(foster).
class(spec_prior):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                   parents(great_pret),social(problematic),housing(convenient),
                   finance(inconv),children(2),(form(incomplete);form(foster)).
class(spec_prior):-health(recommended),has_nurs(improper),parents(great_pret),
                   (social(nonprob);social(slightly_prob)),housing(convenient),
                   finance(inconv),children(1),form(foster).
class(spec_prior):-health(recommended),has_nurs(improper),parents(great_pret),
                   (social(nonprob);social(slightly_prob)),housing(convenient),
                   finance(inconv),children(2),(form(incomplete);form(foster)).
class(spec_prior):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
                   housing(convenient),finance(inconv),children(1),form(foster).
class(spec_prior):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
                   housing(convenient),finance(inconv),children(2),
                   (form(incomplete);form(foster)).
class(spec_prior):-health(recommended),has_nurs(critical),
                   (parents(pretentious);parents(great_pret)),
                   (social(nonprob);social(slightly_prob)),housing(convenient),
                   finance(inconv),children(1),form(foster).
class(spec_prior):-health(recommended),has_nurs(critical),
                   (parents(pretentious);parents(great_pret)),
                   (social(nonprob);social(slightly_prob)),housing(convenient),
                   finance(inconv),children(2),(form(incomplete);form(foster)).
class(spec_prior):-health(recommended),has_nurs(improper),parents(great_pret),
                   social(problematic),form(complete),children(1),housing(convenient),
                   finance(inconv).
class(spec_prior):-health(recommended),has_nurs(critical),
                   (parents(pretentious);parents(great_pret)),social(problematic),
                   form(complete),children(1),housing(convenient),finance(inconv).
          %vr%
class(very_recom):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                    (parents(usual);parents(pretentious)),
                    (social(nonprob);social(slightly_prob)),housing(convenient),
                    finance(inconv),children(1),
                    (form(complete);form(completed);form(incomplete)).
class(very_recom):-health(recommended),(has_nurs(proper);has_nurs(less_proper)),
                    (parents(usual);parents(pretentious)),
                    (social(nonprob);social(slightly_prob)),housing(convenient),
                    finance(inconv),children(2),(form(complete);form(completed)).
class(very_recom):-health(recommended),has_nurs(proper),parents(usual),
                    (social(nonprob);social(slightly_prob)),housing(convenient),
                    finance(convenient),form(complete),
                    (children(2);children(3);children(more)).
```

```
class(very_recom):-health(recommended),has_nurs(improper),parents(usual),
                   (social(nonprob);social(slightly_prob)),housing(convenient),
                   finance(inconv),children(1),
                   (form(complete);form(completed);form(incomplete)).
class(very_recom):-health(recommended),has_nurs(improper),parents(usual),
                   (social(nonprob);social(slightly_prob)),housing(convenient),
                   finance(inconv),children(2),(form(complete);form(completed)).
        %r%
class(recommend):-health(recommended),has_nurs(proper),parents(usual),
                   (social(nonprob);social(slightly_prob)),housing(convenient),
                   finance(convenient),form(complete),children(1).
```

ภาคผนวก ง

โค้ดฐานความรู้ Nursery ในรูปแบบวิธีเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ

```
health(X):-time(menuask(health,X,[recommended,priority,not_recom])).
has_nurs(X):-time(menuask(has_nurs,X,[proper,less_proper,improper,critical,very_crit])).
parents(X):-time(menuask(parents,X,[usual,pretentious,great_pret])).
social(X):-time(menuask(social,X,[nonprob,slightly_prob,problematic])).
housing(X):-time(menuask(housing,X,[convenient,less_conv,critical])).
finance(X):-time(menuask(finance,X,[convenient,inconv])).
form(X):-time(menuask(form,X,[complete,completed,incomplete,foster])).
children(X):-time(menuask(children,X,[1,2,3,more])).

top_goal(X):-class(X).

%1%
class(not_recom):-health(not_recom).

%3%
        %p%
class(priority):-health(priority),has_nurs(N),parents(P),N&::[proper,less_proper],
                P&::[usual,pretentious].
class(priority):-health(priority),has_nurs(improper),parents(usual).
        %sp%
class(spec_prior):-health(priority),has_nurs(very_crit),form(M),
                   M&::[completed,incomplete,foster].

%4%
        %p%
class(priority):-health(recommended),has_nurs(N),parents(great_pret),social(S),
                N&::[proper,less_proper],S&::[nonprob,slightly_prob].
class(priority):-health(recommended),has_nurs(N),parents(P),social(problematic),
                N&::[proper,less_proper],P&::[usual,pretentious].
class(priority):-health(recommended),has_nurs(improper),parents(usual),social(problematic).
class(priority):-health(recommended),has_nurs(improper),parents(pretentious),social(S),
                S&::[nonprob,slightly_prob].
class(priority):-health(recommended),has_nurs(critical),parents(usual),social(S),
                S&::[nonprob,slightly_prob].
        %sp%
class(spec_prior):-health(priority),has_nurs(very_crit),form(complete),children(C),
                   C&::[2,3,more].
class(spec_prior):-health(recommended),has_nurs(very_crit),social(problematic),form(M),
                   M&::[completed,incomplete,foster].
class(spec_prior):-health(priority),has_nurs(improper),parents(great_pret),form(M),
                   M&::[completed,incomplete,foster].
class(spec_prior):-health(priority),has_nurs(critical),parents(P),form(M),
                   P&::[pretentious,great_pret],M&::[completed,incomplete,foster].

%5%
        %p%
```

```
class(priority):-health(recommended),has_nurs(very_crit),social(S),housing(convenient),
                 finance(convenient),S&::[nonprob,slightly_prob].
class(priority):-health(priority),has_nurs(N),parents(great_pret),housing(convenient),
                 finance(convenient),N&::[proper,less_proper].
class(priority):-health(priority),has_nurs(improper),parents(pretentious),housing(convenient),
                 finance(convenient).
class(priority):-health(priority),has_nurs(critical),parents(usual),housing(convenient),
                 finance(convenient).
        %sp%
class(spec_prior):-health(priority),has_nurs(very_crit),form(complete),children(1),housing(H),
                   H&::[less_conv,critical].
class(spec_prior):-health(priority),has_nurs(N),parents(great_pret),housing(critical),form(M),
                   N&::[proper,less_proper],M&::[completed,incomplete,foster].
class(spec_prior):-health(priority),has_nurs(improper),parents(pretentious),housing(critical),
                   form(M),M&::[completed,incomplete,foster].
class(spec_prior):-health(priority),has_nurs(critical),parents(usual),housing(critical),form(M),
                   M&::[completed,incomplete,foster].
class(spec_prior):-health(priority),has_nurs(improper),parents(great_pret),form(complete),
                   children(C),C&::[2,3,more].
class(spec_prior):-health(priority),has_nurs(critical),parents(P),form(complete),children(C),
                   P&::[pretentious,great_pret],C&::[2,3,more].
class(spec_prior):-health(priority),has_nurs(N),parents(great_pret),housing(less_conv),
                   children(C),N&::[proper,less_proper],C&::[3,more].
class(spec_prior):-health(priority),has_nurs(improper),parents(pretentious),housing(less_conv),
                   children(C),C&::[3,more].
class(spec_prior):-health(priority),has_nurs(critical),parents(usual),housing(less_conv),
                   children(C),C&::[3,more].
class(spec_prior):-health(recommended),has_nurs(improper),parents(great_pret),
                   social(problematic),form(M),M&::[completed,incomplete,foster].
class(spec_prior):-health(recommended),has_nurs(critical),parents(P),social(problematic),
                   form(M),P&::[pretentious,great_pret],M&::[completed,incomplete,foster].
class(spec_prior):-health(recommended),has_nurs(very_crit),social(problematic),
                   form(complete),children(C),C&::[2,3,more].
class(spec_prior):-health(recommended),has_nurs(very_crit),social(S),housing(less_conv),
                   children(C),S&::[nonprob,slightly_prob],C&::[3,more].
class(spec_prior):-health(recommended),has_nurs(very_crit),social(S),housing(critical),
                   form(M),S&::[nonprob,slightly_prob],M&::[completed,incomplete,foster].

%6%
        %p%
class(priority):-health(priority),has_nurs(very_crit),form(complete),children(1),
                 housing(convenient),finance(convenient).
class(priority):-health(priority),has_nurs(N),parents(great_pret),housing(critical),
                 form(complete),children(1),N&::[proper,less_proper].
class(priority):-health(priority),has_nurs(improper),parents(pretentious),housing(critical),
                 form(complete),children(1).
```

```
class(priority):-health(priority),has_nurs(critical),parents(usual),housing(critical),
                 form(complete),children(1).
class(priority):-health(recommended),has_nurs(N),parents(P),social(S),housing(less_conv),
                 children(C),N&::[proper,less_proper],P&::[usual,pretentious],
                 S&::[nonprob,slightly_prob],C&::[3,more].
class(priority):-health(recommended),has_nurs(improper),parents(usual),social(S),
                 housing(less_conv),children(C),S&::[nonprob,slightly_prob],C&::[3,more].
class(priority):-health(recommended),has_nurs(N),parents(P),social(S),housing(critical),
                 form(M),N&::[proper,less_proper],P&::[usual,pretentious],
                 S&::[nonprob,slightly_prob],M&::[completed,incomplete,foster].
class(priority):-health(recommended),has_nurs(improper),parents(usual),social(S),
                 housing(critical),form(M),S&::[nonprob,slightly_prob],
                 M&::[completed,incomplete,foster].
class(priority):-health(recommended),has_nurs(N),parents(great_pret),social(problematic),
                 housing(convenient),finance(convenient),N&::[proper,less_proper].
class(priority):-health(recommended),has_nurs(improper),parents(pretentious),
                 social(problematic),housing(convenient),finance(convenient).
class(priority):-health(recommended),has_nurs(improper),parents(great_pret),social(S),
                 housing(convenient),finance(convenient),S&::[nonprob,slightly_prob].
class(priority):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
                 housing(convenient),finance(convenient).
class(priority):-health(recommended),has_nurs(critical),parents(P),social(S),
                 housing(convenient),finance(convenient),P&::[pretentious,great_pret],
                 S&::[nonprob,slightly_prob].
class(priority):-health(priority),has_nurs(N),parents(great_pret),housing(less_conv),children(1),
                 form(M),N&::[proper,less_proper],M&::[complete,completed,incomplete].
class(priority):-health(priority),has_nurs(N),parents(great_pret),housing(less_conv),children(2),
                 form(M),N&::[proper,less_proper],M&::[complete,completed].
class(priority):-health(priority),has_nurs(improper),parents(pretentious),housing(less_conv),
                 children(1),form(M),M&::[complete,completed,incomplete].
class(priority):-health(priority),has_nurs(improper),parents(pretentious),housing(less_conv),
                 children(2),form(M),M&::[complete,completed].
class(priority):-health(priority),has_nurs(critical),parents(usual),housing(less_conv),children(1),
                 form(M),M&::[complete,completed,incomplete].
class(priority):-health(priority),has_nurs(critical),parents(usual),housing(less_conv),children(2),
                 form(M),M&::[complete,completed].
class(priority):-health(recommended),has_nurs(very_crit),social(S),housing(less_conv),
                 children(1),form(M),S&::[nonprob,slightly_prob],
                 M&::[completed,incomplete,foster].
class(priority):-health(recommended),has_nurs(very_crit),social(S),housing(less_conv),
                 children(2),form(M),S&::[nonprob,slightly_prob],M&::[complete,completed].
class(priority):-health(recommended),has_nurs(very_crit),social(S),housing(critical),
                 form(complete),children(1),S&::[nonprob,slightly_prob].
    %sp%
class(spec_prior):-health(priority),has_nurs(very_crit),form(complete),children(1),
                   housing(convenient),finance(inconv).
```

```
class(spec_prior):-health(priority),has_nurs(N),parents(great_pret),housing(critical),
        form(complete),children(C),N&::[proper,less_proper],C&::[2,3,more].
class(spec_prior):-health(priority),has_nurs(improper),parents(pretentious),housing(critical),
        form(complete),children(C),C&::[2,3,more].
class(spec_prior):-health(priority),has_nurs(critical),parents(usual),housing(critical),
        form(complete),children(C),C&::[2,3,more].
class(spec_prior):-health(recommended),has_nurs(very_crit),social(S),housing(convenient),
        finance(inconv),children(C),S&::[nonprob,slightly_prob],C&::[3,more].
class(spec_prior):-health(priority),has_nurs(improper),parents(great_pret),form(complete),
        children(1),housing(H),H&::[less_conv,critical].
class(spec_prior):-health(priority),has_nurs(critical),parents(P),form(complete),children(1),
        housing(H),P&::[pretentious,great_pret],H&::[less_conv,critical].
class(spec_prior):-health(recommended),has_nurs(N),parents(great_pret),social(problematic),
        housing(less_conv),children(C),N&::[proper,less_proper],C&::[3,more].
class(spec_prior):-health(recommended),has_nurs(improper),parents(pretentious),
        social(problematic),housing(H),children(C),H&::[convenient,less_conv],
        C&::[3,more].
class(spec_prior):-health(recommended),has_nurs(improper),parents(great_pret),social(S),
        housing(less_conv),children(C),S&::[nonprob,slightly_prob],C&::[3,more].
class(spec_prior):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
        housing(less_conv),children(C),C&::[3,more].
class(spec_prior):-health(recommended),has_nurs(critical),parents(P),social(S),
        housing(less_conv),children(C),P&::[pretentious,great_pret],
        S&::[nonprob,slightly_prob],C&::[3,more].
class(spec_prior):-health(recommended),has_nurs(N),parents(great_pret),social(problematic),
        housing(critical),form(M),N&::[proper,less_proper],
        M&::[completed,incomplete,foster].
class(spec_prior):-health(recommended),has_nurs(improper),parents(pretentious),
        social(problematic),housing(critical),form(M),
        M&::[completed,incomplete,foster].
class(spec_prior):-health(recommended),has_nurs(improper),parents(great_pret),social(S),
        housing(critical),form(M),S&::[nonprob,slightly_prob],
        M&::[completed,incomplete,foster].
class(spec_prior):-health(recommended),has_nurs(critical),parents(usual),
        social(problematic),housing(critical),form(M),
        M&::[completed,incomplete,foster].
class(spec_prior):-
health(recommended),has_nurs(critical),parents(P),social(S),housing(critical),
        form(M),P&::[pretentious,great_pret],S&::[nonprob,slightly_prob],
        M&::[completed,incomplete,foster].
class(spec_prior):-health(priority),has_nurs(N),parents(great_pret),housing(convenient),
        finance(inconv),children(C),N&::[proper,less_proper],C&::[3,more].
class(spec_prior):-
health(priority),has_nurs(improper),parents(pretentious),housing(convenient),
        finance(inconv),children(C),C&::[3,more].
class(spec_prior):-health(priority),has_nurs(critical),parents(usual),housing(convenient),
        finance(inconv),children(C),C&::[3,more].
```

```
class(spec_prior):-health(priority),has_nurs(N),parents(great_pret),housing(less_conv),
                        children(1),form(foster),N&::[proper,less_proper].
class(spec_prior):-health(priority),has_nurs(N),parents(great_pret),housing(less_conv),
                        children(2),form(M),N&::[proper,less_proper],M&::[incomplete,foster].
class(spec_prior):-health(priority),has_nurs(improper),parents(pretentious),housing(less_conv),
                        children(1),form(foster).
class(spec_prior):-health(priority),has_nurs(improper),parents(pretentious),housing(less_conv),
                        children(2),form(M),M&::[incomplete,foster].
class(spec_prior):-health(priority),has_nurs(critical),parents(usual),housing(less_conv),
                        children(1),form(foster).
class(spec_prior):-health(priority),has_nurs(critical),parents(usual),housing(less_conv),
                        children(2),form(M),M&::[incomplete,foster].
class(spec_prior):-health(recommended),has_nurs(improper),parents(great_pret),
                        social(problematic),form(complete),children(C),C&::[2,3,more].
class(spec_prior):-health(recommended),has_nurs(critical),parents(P),social(problematic),
                        form(complete),children(C),P&::[pretentious,great_pret],C&::[2,3,more].
class(spec_prior):-health(recommended),has_nurs(very_crit),social(problematic),
                        form(complete),children(1),housing(H),H&::[less_conv,critical].
class(spec_prior):-health(recommended),has_nurs(very_crit),social(S),housing(less_conv),
                        children(1),form(foster),S&::[nonprob,slightly_prob].
class(spec_prior):-health(recommended),has_nurs(very_crit),social(S),housing(less_conv),
                        children(2),form(M),S&::[nonprob,slightly_prob],M&::[incomplete,foster].
class(spec_prior):-health(recommended),has_nurs(very_crit),social(S),housing(critical),
                        form(complete),children(C),S&::[nonprob,slightly_prob],C&::[2,3,more].
        %vr%
class(very_recom):-health(recommended),has_nurs(proper),parents(pretentious),social(S),
                        housing(convenient),finance(convenient),S&::[nonprob,slightly_prob].
class(very_recom):-health(recommended),has_nurs(less_proper),parents(P),social(S),
                        housing(convenient),finance(convenient),P&::[usual,pretentious],
                        S&::[nonprob,slightly_prob].
class(very_recom):-health(recommended),has_nurs(improper),parents(usual),social(S),
                        housing(convenient),finance(convenient),S&::[nonprob,slightly_prob].

%7%
        %p%
class(priority):-health(recommended),has_nurs(N),parents(P),social(S),housing(convenient),
                    finance(inconv),children(C),N&::[proper,less_proper],P&::[usual,pretentious],
                    S&::[nonprob,slightly_prob],C&::[3,more].
class(priority):-health(recommended),has_nurs(improper),parents(usual),social(S),
                    housing(convenient),finance(inconv),children(C),S&::[nonprob,slightly_prob],
                    C&::[3,more].
class(priority):-health(recommended),has_nurs(very_crit),social(S),housing(convenient),
                    finance(inconv),children(1),form(M),S&::[nonprob,slightly_prob],
                    M&::[complete,completed,incomplete].
class(priority):-health(recommended),has_nurs(very_crit),social(S),housing(convenient),
                    finance(inconv),children(2),form(M),S&::[nonprob,slightly_prob],
                    M&::[complete,completed].
```

```
class(priority):-health(priority),has_nurs(improper),parents(great_pret),form(complete),
                  children(1),housing(convenient),finance(convenient).
class(priority):-health(priority),has_nurs(critical),parents(P),form(complete),children(1),
                  housing(convenient),finance(convenient),P&::[pretentious,great_pret].
class(priority):-health(recommended),has_nurs(N),parents(P),social(S),housing(less_conv),
                  children(1),form(foster),N&::[proper,less_proper],P&::[usual,pretentious],
                  S&::[nonprob,slightly_prob].
class(priority):-health(recommended),has_nurs(N),parents(P),social(S),housing(less_conv),
                  children(2),form(M),N&::[proper,less_proper],P&::[usual,pretentious],
                  S&::[nonprob,slightly_prob],M&::[incomplete,foster].
class(priority):-health(recommended),has_nurs(N),parents(great_pret),social(problematic),
                  housing(less_conv),children(1),form(M),N&::[proper,less_proper],
                  M&::[complete,completed,incomplete].
class(priority):-health(recommended),has_nurs(N),parents(great_pret),social(problematic),
                  housing(less_conv),children(2),form(M),N&::[proper,less_proper],
                  M&::[complete,completed].
class(priority):-health(recommended),has_nurs(improper),parents(usual),social(S),
                  housing(less_conv),children(1),form(foster),S&::[nonprob,slightly_prob].
class(priority):-health(recommended),has_nurs(improper),parents(usual),social(S),
                  housing(less_conv),children(2),form(M),S&::[nonprob,slightly_prob],
                  M&::[incomplete,foster].
class(priority):-health(recommended),has_nurs(improper),parents(pretentious),
                  social(problematic),housing(H),children(1),form(M),
                  H&::[convenient,less_conv],M&::[complete,completed,incomplete].
class(priority):-health(recommended),has_nurs(improper),parents(pretentious),
                  social(problematic),housing(H),children(2),form(M),
                  H&::[convenient,less_conv],M&::[complete,completed].
class(priority):-health(recommended),has_nurs(improper),parents(great_pret),social(S),
                  housing(less_conv),children(1),form(M),S&::[nonprob,slightly_prob],
                  M&::[complete,completed,incomplete].
class(priority):-health(recommended),has_nurs(improper),parents(great_pret),social(S),
                  housing(less_conv),children(2),form(M),S&::[nonprob,slightly_prob],
                  M&::[complete,completed].
class(priority):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
                  housing(less_conv),children(1),form(M),
                  M&::[complete,completed,incomplete].
class(priority):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
                  housing(less_conv),children(2),form(M),M&::[complete,completed].
class(priority):-health(recommended),has_nurs(critical),parents(P),social(S),
                  housing(less_conv),children(1),form(M),P&::[pretentious,great_pret],
                  S&::[nonprob,slightly_prob],M&::[complete,completed,incomplete].
class(priority):-health(recommended),has_nurs(critical),parents(P),social(S),
                  housing(less_conv),children(2),form(M),P&::[pretentious,great_pret],
                  S&::[nonprob,slightly_prob],M&::[complete,completed].
class(priority):-health(recommended),has_nurs(N),parents(P),social(S),housing(critical),
                  form(complete),children(C),N&::[proper,less_proper],P&::[usual,pretentious],
                  S&::[nonprob,slightly_prob],C&::[2,3,more].
```

```
class(priority):-health(recommended),has_nurs(N),parents(great_pret),social(problematic),
                housing(critical),form(complete),children(1),N&::[proper,less_proper].
class(priority):-health(recommended),has_nurs(improper),parents(usual),social(S),
                housing(critical),form(complete),children(C),S&::[nonprob,slightly_prob],
                C&::[2,3,more].
class(priority):-health(recommended),has_nurs(improper),parents(pretentious),
                social(problematic),housing(critical),form(complete),children(1).
class(priority):-health(recommended),has_nurs(improper),parents(great_pret),social(S),
                housing(critical),form(complete),children(1),S&::[nonprob,slightly_prob].
class(priority):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
                housing(critical),form(complete),children(1).
class(priority):-health(recommended),has_nurs(critical),parents(P),social(S),housing(critical),
                form(complete),children(1),P&::[pretentious,great_pret],
                S&::[nonprob,slightly_prob].
class(priority):-health(priority),has_nurs(N),parents(great_pret),housing(convenient),
                finance(inconv),children(1),form(M),N&::[proper,less_proper],
                M&::[complete,completed,incomplete].
class(priority):-health(priority),has_nurs(N),parents(great_pret),housing(convenient),
                finance(inconv),children(2),form(M),N&::[proper,less_proper],
                M&::[complete,completed].
class(priority):-health(priority),has_nurs(improper),parents(pretentious),housing(convenient),
                finance(inconv),children(1),form(M),M&::[complete,completed,incomplete].
class(priority):-health(priority),has_nurs(improper),parents(pretentious),housing(convenient),
                finance(inconv),children(2),form(M),M&::[complete,completed].
class(priority):-health(priority),has_nurs(critical),parents(usual),housing(convenient),
                finance(inconv),children(1),form(M),M&::[complete,completed,incomplete].
class(priority):-health(priority),has_nurs(critical),parents(usual),housing(convenient),
                finance(inconv),children(2),form(M),M&::[complete,completed].
class(priority):-health(recommended),has_nurs(very_crit),social(problematic),form(complete),
                children(1),housing(convenient),finance(convenient).
        %sp%
class(spec_prior):-health(recommended),has_nurs(N),parents(great_pret),social(problematic),
                  housing(convenient),finance(inconv),children(C),N&::[proper,less_proper],
                  C&::[3,more].
class(spec_prior):-health(recommended),has_nurs(improper),parents(great_pret),social(S),
                  housing(convenient),finance(inconv),children(C),
                  S&::[nonprob,slightly_prob],C&::[3,more].
class(spec_prior):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
                  housing(convenient),finance(inconv),children(C),C&::[3,more].
class(spec_prior):-health(recommended),has_nurs(critical),parents(P),social(S),
                  housing(convenient),finance(inconv),children(C),
                  P&::[pretentious,great_pret],S&::[nonprob,slightly_prob],C&::[3,more].
class(spec_prior):-health(recommended),has_nurs(very_crit),social(S),housing(convenient),
                  finance(inconv),children(1),form(foster),S&::[nonprob,slightly_prob].
class(spec_prior):-health(recommended),has_nurs(very_crit),social(S),housing(convenient),
                  finance(inconv),children(2),form(M),S&::[nonprob,slightly_prob],
                  M&::[incomplete,foster].
```

```
class(spec_prior):-health(priority),has_nurs(improper),parents(great_pret),form(complete),
                  children(1),housing(convenient),finance(inconv).
class(spec_prior):-health(priority),has_nurs(critical),parents(P),form(complete),children(1),
                  housing(convenient),finance(inconv),P&::[pretentious,great_pret].
class(spec_prior):-health(recommended),has_nurs(N),parents(great_pret),social(problematic),
                  housing(less_conv),children(1),form(foster),N&::[proper,less_proper].
class(spec_prior):-health(recommended),has_nurs(N),parents(great_pret),social(problematic),
                  housing(less_conv),children(2),form(M),N&::[proper,less_proper],
                  M&::[incomplete,foster].
class(spec_prior):-health(recommended),has_nurs(improper),parents(pretentious),
                  social(problematic),housing(H),children(1),form(foster),
                  H&::[convenient,less_conv].
class(spec_prior):-health(recommended),has_nurs(improper),parents(pretentious),
                  social(problematic),housing(H),children(2),form(M),
                  H&::[convenient,less_conv],M&::[incomplete,foster].
class(spec_prior):-health(recommended),has_nurs(improper),parents(great_pret),social(S),
                  housing(less_conv),children(1),form(foster),S&::[nonprob,slightly_prob].
class(spec_prior):-health(recommended),has_nurs(improper),parents(great_pret),social(S),
                  housing(less_conv),children(2),form(M),S&::[nonprob,slightly_prob],
                  M&::[incomplete,foster].
class(spec_prior):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
                  housing(less_conv),children(1),form(foster).
class(spec_prior):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
                  housing(less_conv),children(2),form(M),M&::[incomplete,foster].
class(spec_prior):-
health(recommended),has_nurs(critical),parents(P),social(S),housing(less_conv),
                  children(1),form(foster),P&::[pretentious,great_pret],
                  S&::[nonprob,slightly_prob].
class(spec_prior):-health(recommended),has_nurs(critical),parents(P),social(S),
                  housing(less_conv),children(2),form(M),P&::[pretentious,great_pret],
                  S&::[nonprob,slightly_prob],M&::[incomplete,foster].
class(spec_prior):-health(recommended),has_nurs(N),parents(great_pret),social(problematic),
                  housing(critical),form(complete),children(C),N&::[proper,less_proper],
                  C&::[2,3,more].
class(spec_prior):-health(recommended),has_nurs(improper),parents(pretentious),
                  social(problematic),housing(critical),form(complete),children(C),
                  C&::[2,3,more].
class(spec_prior):-health(recommended),has_nurs(improper),parents(great_pret),social(S),
                  housing(critical),form(complete),children(C),S&::[nonprob,slightly_prob],
                  C&::[2,3,more].
class(spec_prior):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
                  housing(critical),form(complete),children(C),C&::[2,3,more].
class(spec_prior):-
health(recommended),has_nurs(critical),parents(P),social(S),housing(critical),
                  form(complete),children(C),P&::[pretentious,great_pret],
                  S&::[nonprob,slightly_prob],C&::[2,3,more].
```

```
class(spec_prior):-health(priority),has_nurs(N),parents(great_pret),housing(convenient),
                    finance(inconv),children(1),form(foster),N&::[proper,less_proper].
class(spec_prior):-health(priority),has_nurs(N),parents(great_pret),housing(convenient),
                    finance(inconv),children(2),form(M),N&::[proper,less_proper],
                    M&::[incomplete,foster].
class(spec_prior):-health(priority),has_nurs(improper),parents(pretentious),
                    housing(convenient),finance(inconv),children(1),form(foster).
class(spec_prior):-health(priority),has_nurs(improper),parents(pretentious),
                    housing(convenient),finance(inconv),children(2),form(M),
                    M&::[incomplete,foster].
class(spec_prior):-health(priority),has_nurs(critical),parents(usual),housing(convenient),
                    finance(inconv),children(1),form(foster).
class(spec_prior):-health(priority),has_nurs(critical),parents(usual),housing(convenient),
                    finance(inconv),children(2),form(M),M&::[incomplete,foster].
class(spec_prior):-health(recommended),has_nurs(improper),parents(great_pret),
                    social(problematic),form(complete),children(1),housing(H),
                    H&::[less_conv,critical].
class(spec_prior):-health(recommended),has_nurs(critical),parents(P),social(problematic),
                    form(complete),children(1),housing(H),P&::[pretentious,great_pret],
                    H&::[less_conv,critical].
class(spec_prior):-health(recommended),has_nurs(very_crit),social(problematic),
                    form(complete),children(1),housing(convenient),finance(inconv).
        %vr%
class(very_recom):-health(recommended),has_nurs(proper),parents(usual),social(S),
                    housing(convenient),finance(convenient),form(M),
                    S&::[nonprob,slightly_prob],M&::[completed,incomplete,foster].
class(very_recom):-health(recommended),has_nurs(N),parents(P),social(S),housing(less_conv),
                    children(1),form(M),N&::[proper,less_proper],P&::[usual,pretentious],
                    S&::[nonprob,slightly_prob],M&::[complete,completed,incomplete].
class(very_recom):-health(recommended),has_nurs(N),parents(P),social(S),housing(less_conv),
                    children(2),form(M),N&::[proper,less_proper],P&::[usual,pretentious],
                    S&::[nonprob,slightly_prob],M&::[complete,completed].
class(very_recom):-health(recommended),has_nurs(improper),parents(usual),social(S),
                    S&::[nonprob,slightly_prob],housing(less_conv),children(1),form(M),
                    M&::[complete,completed,incomplete].
class(very_recom):-health(recommended),has_nurs(improper),parents(usual),social(S),
                    S&::[nonprob,slightly_prob],housing(less_conv),children(2),form(M),
                    M&::[complete,completed].
class(very_recom):-health(recommended),has_nurs(N),parents(P),social(S),housing(critical),
                    form(complete),children(1),N&::[proper,less_proper],
                    P&::[usual,pretentious],S&::[nonprob,slightly_prob].
class(very_recom):-health(recommended),has_nurs(improper),parents(usual),social(S),
                    housing(critical),form(complete),children(1),S&::[nonprob,slightly_prob].

%8%
        %p%
```

```
class(priority):-health(recommended),has_nurs(N),parents(P),social(S),housing(convenient),
                finance(inconv),children(1),form(foster),N&::[proper,less_proper],
                P&::[usual,pretentious],S&::[nonprob,slightly_prob].
class(priority):-health(recommended),has_nurs(N),parents(P),social(S),housing(convenient),
                finance(inconv),children(2),form(M),N&::[proper,less_proper],
                P&::[usual,pretentious],S&::[nonprob,slightly_prob],
                M&::[incomplete,foster].
class(priority):-health(recommended),has_nurs(N),parents(great_pret),social(problematic),
                housing(convenient),finance(inconv),children(1),form(M),
                N&::[proper,less_proper],M&::[complete,completed,incomplete].
class(priority):-health(recommended),has_nurs(N),parents(great_pret),social(problematic),
                housing(convenient),finance(inconv),children(2),form(M),
                N&::[proper,less_proper],M&::[complete,completed].
class(priority):-health(recommended),has_nurs(improper),parents(usual),social(S),
                housing(convenient),finance(inconv),children(1),form(foster),
                S&::[nonprob,slightly_prob].
class(priority):-health(recommended),has_nurs(improper),parents(usual),social(S),
                housing(convenient),finance(inconv),children(2),form(M),
                S&::[nonprob,slightly_prob],M&::[incomplete,foster].
class(priority):-health(recommended),has_nurs(improper),parents(great_pret),social(S),
                housing(convenient),finance(inconv),children(1),form(M),
                S&::[nonprob,slightly_prob],M&::[complete,completed,incomplete].
class(priority):-health(recommended),has_nurs(improper),parents(great_pret),social(S),
                housing(convenient),finance(inconv),children(2),form(M),
                S&::[nonprob,slightly_prob],M&::[complete,completed].
class(priority):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
                housing(convenient),finance(inconv),children(1),form(M),
                M&::[complete,completed,incomplete].
class(priority):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
                housing(convenient),finance(inconv),children(2),form(M),
                M&::[complete,completed].
class(priority):-health(recommended),has_nurs(critical),parents(P),social(S),
                housing(convenient),finance(inconv),children(1),form(M),
                P&::[pretentious,great_pret],S&::[nonprob,slightly_prob],
                M&::[complete,completed,incomplete].
class(priority):-health(recommended),has_nurs(critical),parents(P),social(S),
                housing(convenient),finance(inconv),children(2),form(M),
                P&::[pretentious,great_pret],S&::[nonprob,slightly_prob],
                M&::[complete,completed].
class(priority):-health(recommended),has_nurs(improper),parents(great_pret),
                social(problematic),form(complete),children(1),housing(convenient),
                finance(convenient).
class(priority):-health(recommended),has_nurs(critical),parents(P),social(problematic),
                form(complete),children(1),housing(convenient),finance(convenient),
                P&::[pretentious,great_pret].
```

```
        %sp%
class(spec_prior):-health(recommended),has_nurs(N),parents(great_pret),social(problematic),
                   housing(convenient),finance(inconv),children(1),form(foster),
                   N&::[proper,less_proper].
class(spec_prior):-health(recommended),has_nurs(N),parents(great_pret),social(problematic),
                   housing(convenient),finance(inconv),children(2),form(M),
                   N&::[proper,less_proper],M&::[incomplete,foster].
class(spec_prior):-health(recommended),has_nurs(improper),parents(great_pret),social(S),
                   housing(convenient),finance(inconv),children(1),form(foster),
                   S&::[nonprob,slightly_prob].
class(spec_prior):-health(recommended),has_nurs(improper),parents(great_pret),social(S),
                   housing(convenient),finance(inconv),children(2),form(M),
                   S&::[nonprob,slightly_prob],M&::[incomplete,foster].
class(spec_prior):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
                   housing(convenient),finance(inconv),children(1),form(foster).
class(spec_prior):-health(recommended),has_nurs(critical),parents(usual),social(problematic),
                   housing(convenient),finance(inconv),children(2),form(M),
                   M&::[incomplete,foster].
class(spec_prior):-health(recommended),has_nurs(critical),parents(P),social(S),
                   housing(convenient),finance(inconv),children(1),form(foster),
                   P&::[pretentious,great_pret],S&::[nonprob,slightly_prob].
class(spec_prior):-health(recommended),has_nurs(critical),parents(P),social(S),
                   housing(convenient),finance(inconv),children(2),form(M),
                   P&::[pretentious,great_pret],S&::[nonprob,slightly_prob],
                   M&::[incomplete,foster].
class(spec_prior):-health(recommended),has_nurs(improper),parents(great_pret),
                   social(problematic),form(complete),children(1),housing(convenient),
                   finance(inconv).
class(spec_prior):-health(recommended),has_nurs(critical),parents(P),social(problematic),
                   form(complete),children(1),housing(convenient),finance(inconv),
                   P&::[pretentious,great_pret].
        %vr%
class(very_recom):-health(recommended),has_nurs(N),parents(P),social(S),
                   housing(convenient),finance(inconv),children(1),form(M),
                   N&::[proper,less_proper],P&::[usual,pretentious],
                   S&::[nonprob,slightly_prob],M&::[complete,completed,incomplete].
class(very_recom):-health(recommended),has_nurs(N),parents(P),social(S),
                   housing(convenient),finance(inconv),children(2),form(M),
                   N&::[proper,less_proper],P&::[usual,pretentious],
                   S&::[nonprob,slightly_prob],M&::[complete,completed].
class(very_recom):-health(recommended),has_nurs(proper),parents(usual),social(S),
                   housing(convenient),finance(convenient),form(complete),children(C),
                   S&::[nonprob,slightly_prob],C&::[2,3,more].
class(very_recom):-health(recommended),has_nurs(improper),parents(usual),social(S),
                   housing(convenient),finance(inconv),children(1),form(M),
                   S&::[nonprob,slightly_prob],M&::[complete,completed,incomplete].
```

```
class(very_recom):-health(recommended),has_nurs(improper),parents(usual),social(S),
                        housing(convenient),finance(inconv),children(2),form(M),
                        S&::[nonprob,slightly_prob],M&::[complete,completed].
        %r%
class(recommend):-health(recommended),has_nurs(proper),parents(usual),social(S),
                        housing(convenient),finance(convenient),form(complete),children(1),
                        S&::[nonprob,slightly_prob].
```

ภาคผนวก จ

วิธีการรันโปรแกรมเปลือกระบบผู้เชี่ยวชาญในโปรแกรม ECLiPSe 6.0

วิธีการใช้งานเปลือกระบบผู้เชี่ยวชาญในโปรแกรม ECLiPSe 6.0

มีขั้นตอนดังต่อไปนี้

1. เปิดหน้าต่างโปรแกรม ECLiPSe 6.0 ขึ้นมา
2. คลิกเมนู File เลือก Change directory … ระบุให้โปรแกรมทำงานในแฟ้มที่เก็บข้อมูลเปลือกระบบผู้เชี่ยวชาญและฐานความรู้ไว้ (กรณีไฟล์ข้อมูลไม่ได้อยู่ในแฟ้ม My Document)
3. คลิกเมนู File เลือก Compile … ไฟล์เปลือกระบบผู้เชี่ยวชาญ
4. พิมพ์คำสั่ง expertshell ในช่อง Query Entry แล้วสั่ง run
5. พิมพ์คำสั่ง help ในช่อง Input box เมื่อต้องการทราบคำสั่งทั้งหมด
6. พิมพ์คำสั่ง load ในช่อง Input box เมื่อต้องการเรียกใช้ไฟล์ฐานความรู้
    a. พิมพ์ ‘ชื่อไฟล์.knb’ ในช่อง Input box เพื่อโหลดไฟล์เข้าระบบ
7. พิมพ์คำสั่ง solve ในช่อง Input box เมื่อต้องการใช้งานระบบผู้เชี่ยวชาญ (ต้องทำหลังจากโหลดไฟล์ฐานความรู้แล้ว)
    a. พิมพ์คำตอบสำหรับแต่ละคำถามเพื่อใช้ในกลไกอนุมาน
8. พิมพ์คำสั่ง why ในช่อง Input box เมื่อต้องการทราบที่มาของคำตอบ (จะแสดงผลก็ต่อเมื่อระบบสามารถทำการอนุมานได้สำเร็จจนได้คำตอบ)

ภาคผนวก ฉ

บทความทางวิชาการที่ได้รับการตีพิมพ์เผยแพร่ในระหว่างศึกษา

รายชื่อบทความที่ได้รับการตีพิมพ์เผยแพร่ในระหว่างศึกษา

กาญจนา อินทรชาธร, นิตยา เกิดประสพ และ กิตติศักดิ์ เกิดประสพ (2555). **การพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ.** การประชุมวิชาการระดับชาติ “ศรีนครินทรวิโรฒวิชาการ” ครั้งที่ 6., หน้า 1-10

การพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ

DEVELOPMENT OF EXPERT SYSTEM SHELL WITH CONSTRAINT LOGIC PROGRAMMING

กาญจนา อินทรชาธร*, นิตยา เกิดประสพ, กิตติศักดิ์ เกิดประสพ

Kanjana Intharachatorn*, Nittaya Kerdprasop, Kittisak Kerdprasop

สาขาวิชาวิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

School of Computer Engineering, Faculty of Engineering, Suranaree University of Technology, Thailand.

***Corresponding author, E-mail:** b4900313kanjana@gmail.com

บทคัดย่อ

งานวิจัยชิ้นนี้มีวัตถุประสงค์เพื่อศึกษาและพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับสำหรับฐานความรู้ใดๆ ที่อยู่ในรูปแบบของเงื่อนไขบังคับ และมีขอบเขตข้อมูลเชิงสัญลักษณ์ โดยระบบสามารถโต้ตอบกับผู้ใช้ระบบเพื่อรับข้อมูลของปัญหา และสามารถอธิบายถึงที่มาของคำตอบที่ได้ ในการพัฒนาระบบจะใช้โปรแกรม ECLiPSe 6.0 เป็นตัวแปลภาษา เพราะมีภาษาโปรล็อกเป็นภาษาพื้นฐานและเหมาะสำหรับพัฒนาระบบเชิงตรรกะด้วยเงื่อนไขบังคับโดยเฉพาะ สำหรับการวัดผลการวิจัยจะเป็นการเปรียบเทียบกับเปลือกระบบผู้เชี่ยวชาญที่เขียนด้วยภาษาโปรล็อกในด้านของระยะเวลา ปริมาณหน่วยความจำ และจำนวนบรรทัด ภายใต้กระบวนการทำงานที่คล้ายคลึงกัน และใช้ฐานความรู้ที่ได้จากแหล่งเดียวกันในการทดสอบระบบ แล้วทำการสังเกตและบันทึกค่าผลการทำงานเพื่อสรุปผลการวิจัย

คำสำคัญ: การเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ ขอบเขตของตัวแปรเชิงสัญลักษณ์ ฐานความรู้ เปลือกระบบผู้เชี่ยวชาญ

Abstract

This research aims to study and develop expert system shell with constraint logic programming for any knowledge base, which has constraints' format and symbolic domain. The system can interact with users to obtain information's issue and it can explain all sources of answer. The system uses ECLiPSe 6.0 as a compiler because it is based on Prolog language and suitable for development into constraint logic programming. For evaluation of the research, the system has been compared with Prolog's expert system shell. Evaluation has been done in terms of time, memory and lines of code, under similar procedures and use knowledge base that originate from the same source. Then observe and record the results of the work to summarize the research.

Keywords: Constraint Logic Programming, Expert System Shell, Knowledge Base, Symbolic Domain

บทนำ

เปลือกระบบผู้เชี่ยวชาญนั้นเป็นโปรแกรมเชิงพาณิชย์ [1] ที่มีความสำคัญมากขึ้นต่อองค์กรทุกองค์กรไม่ว่าขนาดเล็ก หรือขนาดใหญ่ ที่ต้องการสร้างระบบผู้เชี่ยวชาญขึ้นใช้ภายในองค์กร เพราะคุณสมบัติของเปลือกระบบผู้เชี่ยวชาญที่ยอมให้ผู้ใช้เปลี่ยนฐานความรู้ของระบบได้ ทำให้ไม่จำเป็นต้องพัฒนาระบบใหม่ทั้งหมด เหลือเพียงแค่ขั้นตอนการกลั่นกรองความรู้จากผู้เชี่ยวชาญและการแปลงความรู้นั้นให้กลายเป็นฐานความรู้ของระบบผู้เชี่ยวชาญ ซึ่งสองขั้นตอนนี้เป็นขั้นตอนที่ทำได้ยากและเสียเวลามาก [2] เพราะผู้เชี่ยวชาญนั้นบางครั้งอาจจะไม่สามารถถ่ายทอดความรู้และประสบการณ์ที่มีให้วิศวกรความรู้ได้ในคราวเดียว อาจจะต้องเสียเวลาในการเก็บรวบรวมและกลั่นกรองข้อมูลหลายครั้ง แต่กระบวนการนี้ก็มีการนำเสนอการวิจัยวิธีการกลั่นกรองข้อมูลโดยใช้วิธีการทำเหมืองข้อมูลด้วย ID3 [3] แล้วทำการแปลงกฎที่ได้จากการทำเหมืองข้อมูลนั้นเป็นฐานความรู้ของเปลือกระบบผู้เชี่ยวชาญอย่างง่ายที่พัฒนาด้วยภาษาโปรล็อก

วิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับนั้นมีพื้นฐานภาษาเป็นภาษาโปรล็อก แต่มีรูปแบบในการจัดการขอบเขตของตัวแปรที่ชัดเจนกว่าภาษาโปรล็อก ที่ไม่มีการประกาศตัวแปรอย่างชัดเจน และค่าของตัวแปรก็ใช้ได้เฉพาะใน predicate ที่ได้รับค่าตัวแปรนั้นๆ มา นอกจากนี้วิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับนั้นยังมีข้อดีอีกหลายอย่างที่ทำให้มีความน่าสนใจเมื่อเทียบกับภาษาโปรล็อก [4] เช่น วิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับมีกระบวนการค้นหาที่มีประสิทธิภาพมากกว่าเมื่อเปรียบเทียบกับ Depth First Search ในภาษาโปรล็อก วิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับมีกระบวนการค้นหาคำตอบที่หลากหลาย มีเทคนิคการกำหนดเงื่อนไข การแทนค่าตัวแปร ที่ถูกรวมกันไว้ได้อย่างมีประสิทธิภาพ และง่ายต่อการใช้งาน เป็นต้น

จากการศึกษางานวิจัยเกี่ยวกับการพัฒนาระบบผู้เชี่ยวชาญด้วยวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ [5-8] พบว่า ระบบที่ได้จากการศึกษาและพัฒนาในงานวิจัยเหล่านั้นยังอยู่ในรูปของระบบผู้เชี่ยวชาญที่สามารถใช้ได้กับข้อมูลเชิงคณิตศาสตร์ และเป็นข้อมูลที่มีสมการเฉพาะเพื่อแก้โจทย์ปัญหาทางคณิตศาสตร์ที่โปรแกรมภาษาโปรล็อกไม่สามารถทำได้ และจากการค้นคว้างานวิจัยเกี่ยวกับระบบผู้เชี่ยวชาญที่พัฒนาด้วยวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับยังไม่พบกรณี นำวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับไปพัฒนาเป็นเปลือกระบบผู้เชี่ยวชาญเพื่อใช้กับฐานความรู้ใดๆ เลย

จากที่กล่าวมาข้างต้นผู้วิจัยจึงได้ทำการศึกษาและออกแบบเปลือกระบบผู้เชี่ยวชาญด้วยวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับขึ้นมา ซึ่งเปลือกระบบที่จะพัฒนานี้ต้องสามารถใช้ได้กับฐานความรู้ใดๆ โดยระบบจะแบ่งออกเป็น 2 ส่วน คือ เปลือกระบบผู้เชี่ยวชาญ และฐานความรู้ของระบบ เนื่องจากข้อได้เปรียบบางอย่างของวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับทำให้มีความเป็นไปได้ว่า เปลือกระบบที่จะพัฒนานี้จะดีกว่าเปลือกระบบเดิมที่พัฒนาด้วยภาษาโปรล็อก ในด้านของจำนวนบรรทัดของโค้ดที่อาจจะสั้นลง ขนาดของข้อมูลที่อาจจะเล็กลงเพราะจำนวนโค้ดที่น้อยลง และเวลาที่ใช้ในการประมวลผลของระบบที่อาจจะน้อยลง

วัตถุประสงค์ของการวิจัย

การวิจัยนี้มีวัตถุประสงค์เพื่อศึกษาและพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ ที่ใช้งานร่วมกับฐานความรู้ใดๆ ภายในขอบเขตของตัวแปรเชิงสัญลักษณ์ที่มีรูปแบบข้อมูลเดียวกันกับกลไกการทำงานของระบบ โดยระบบนี้จะมีส่วนติดต่อกับผู้ใช้ เพื่อรับข้อมูลสอบถามไปใช้ในการค้นหาความรู้ภายในกลไกอนุมาน ซึ่งมีการอนุมานความรู้ได้อย่างเหมาะสมกับข้อมูลที่ผู้ใช้ระบุเข้ามา พร้อมทั้งมีการอธิบายความรู้สนับสนุนคำตอบของผู้ใช้ได้อย่างเหมาะสม การวัดผลวิจัยจะใช้การเก็บข้อมูลขนาดของหน่วยความจำ จำนวนบรรทัดของข้อมูล และเวลาที่ใช้ประมวลผลและแสดงผลออกหน้าจอของระบบที่พัฒนาขึ้นเปรียบเทียบกับเปลือกระบบผู้เชี่ยวชาญที่เขียนด้วยภาษาโปรล็อก โดยทดสอบภายใต้การทำงานที่คล้ายคลึงกันและใช้ข้อมูลจากฐานความรู้เดียวกัน

วิธีดำเนินการวิจัย

งานวิจัยนี้เป็นการศึกษาและพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ ระบบที่ได้ศึกษาและพัฒนาขึ้นนี้มีส่วนประกอบหลัก 2 ส่วน คือ เปลือกระบบผู้เชี่ยวชาญ และ ฐานความรู้ของระบบ โดยใช้โปรแกรม ECLiPSe 6.0 เป็นตัวแปลภาษา เพราะสามารถใช้ได้กับภาษาโปรล็อก และภาษาโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ

1) การศึกษารูปแบบการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ

ในการวิจัยนี้ได้มีการศึกษารูปแบบการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ ที่ใช้โปรแกรม ECLiPSe 6.0 เป็นตัวแปลภาษา เพราะมีภาษาโปรล็อกเป็นภาษาพื้นฐานที่ผู้วิจัยมีพื้นฐานความรู้อยู่แล้ว และเหมาะสำหรับพัฒนาโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับโดยเฉพาะ ลักษณะของโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับมีวิธีเขียนแตกต่างกันตามประเภทของตัวแปรที่ประกาศขอบเขตไว้ แบ่งออกเป็น 3 ประเภทหลักๆ คือ ขอบเขตของข้อมูลเชิงสัญลักษณ์ ขอบเขตของข้อมูลเลขจำนวนเต็ม และขอบเขตของข้อมูลเลขทศนิยม ซึ่งสามารถสรุปลักษณะที่แตกต่างกันได้ ดังไปต่อนี้

ตารางที่ ฉ.1 สรุปลักษณะที่แตกต่างกันของโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับภายใต้ขอบเขตข้อมูลที่ต่างกัน

คำอธิบาย	ตัวอย่างของขอบเขตข้อมูล			หมายเหตุ
	เชิงสัญลักษณ์	เลขจำนวนเต็ม	เลขทศนิยม	
ไลบารี่ที่เรียกใช้	:- lib(sd).	:- lib(ic).	:- lib(fd).	ประกาศที่ส่วนต้นของไฟล์
ประกาศตัวแปร	X &:: [a,b,c]	X :: [3,4,5,6]	X :: 0.0..10.0	ประกาศตัวแปรเดียว
	[Y,Z] &:: [a,b,c,d]	[Y,Z] :: [1..3,5]	[Y,Z] :: 0.0..20.0	ขอบเขตเดียวหลายตัวแปร
เขียนเงื่อนไขบังคับ	X &= Y, Y &\= Z	X #= Y, X #> Z	X $= Z – Y, Y $< Z	ใช้ตัวกระทำระหว่างตัวแปร

งานวิจัยนี้ผู้วิจัยเลือกใช้ขอบเขตข้อมูลเชิงสัญลักษณ์ เนื่องจากความไม่แน่นอนของฐานความรู้ที่ต้องใช้กับเปลือกระบบผู้เชี่ยวชาญในอนาคตทำให้ไม่อาจคาดเดาได้ว่าข้อมูลเหล่านั้นจะอยู่ในลักษณะใด จึงต้องเลือกประเภทข้อมูลที่รองรับข้อมูลที่ผ่านการดัดแปลงให้อยู่ในรูปเชิงสัญลักษณ์ได้คือ ขอบเขตข้อมูลเชิง

สัญลักษณ์

2) การศึกษาลักษณะการออกแบบฐานความรู้ และเปลือกระบบผู้เชี่ยวชาญภาษาโปรล็อก

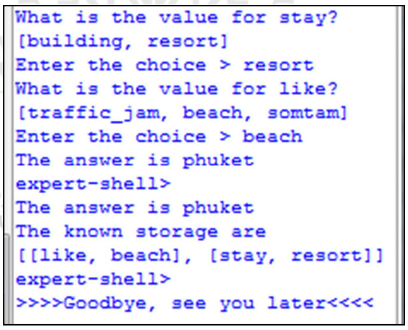
การศึกษาเพื่อพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับนั้น ผู้วิจัยได้เริ่มจากศึกษาการทำงานของเปลือกระบบผู้เชี่ยวชาญอย่างง่ายที่เขียนด้วยภาษาโปรล็อกก่อน

```
top_goal(X) :- vacation(X).
vacation(phuket) :- stay(resort), like(beach).
vacation(korat) :- will_see(ya_mo).
vacation(korat) :- like(somtam).
%%% More Knowledge Here %%%
stay(X) :- menuask(stay, X, [building, resort]).
buy(X) :- menuask(buy, X, [tea,coffee, orange]).
like(X) :- menuask(like, X, [traffic_jam, beach, somtam]).
will_see(X) :- menuask(will_see, X, [ya_mo, lin_ping]).
```

ภาพที่ ฉ.1 แสดงตัวอย่างฐานความรู้ของเปลือกระบบผู้เชี่ยวชาญอย่างง่ายที่เขียนด้วยภาษาโปรล็อก

```
menuask(Pred, Value, Menu) :- menuask(Pred,Menu), known([Pred,Value]), !.
menuask(Pred,_) :- known([Pred,_]), !.
menuask(Attribute,Menu) :-   write('What is the value for '), write(Attribute), writeln('?'),
                             writeln(Menu), write('Enter the choice > '), read(V),writeln(V),
                             asserta(known([Attribute,V])).
```

ภาพที่ ฉ.2 แสดง predicate ชื่อ menuask จากเปลือกระบบผู้เชี่ยวชาญอย่างง่ายที่เขียนด้วยภาษาโปรล็อก



ภาพที่ ฉ.3 ตัวอย่างผลลัพธ์ที่ได้จากเปลือกระบบผู้เชี่ยวชาญที่เขียนด้วยภาษาโปรล็อก

จากการศึกษาโค้ดตัวอย่างในภาพที่ 1 และ 2 ทำให้ผู้วิจัยสามารถวางแนวทางการการพัฒนาระบบในงานวิจัยนี้ออกเป็น 2 ส่วน คือ ฐานความรู้ และเปลือกระบบผู้เชี่ยวชาญ โดยระบบที่พัฒนาขึ้นจะมีลักษณะการทำงานของส่วนติดต่อผู้ใช้เหมือนกับระบบเดิมที่เป็นภาษาโปรล็อก และเนื่องจากวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับนั้นมีพื้นฐานมาจากภาษาโปรล็อก จึงสามารถรักษาโค้ดบางส่วนให้คงอยู่ในลักษณะเดิม และมีการเปลี่ยนแปลงเฉพาะ predicate ที่ชื่อ menuask ในโค้ดของเปลือกระบบผู้เชี่ยวชาญตามความเหมาะสม ซึ่งจะสัมพันธ์กับโค้ดของฐานความรู้ที่ผู้วิจัยได้นำเสนอรูปแบบการพัฒนาฐานความให้อยู่ในรูปของการโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับไว้ในหลายแง่มุม โดยจะเสนอเป็นลำดับถัดไป

3) แนวทางการพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยวิธีเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ

รูปแบบที่ 1: ผู้วิจัยทำการปรับเปลี่ยนฐานความรู้ทั้งหมดให้อยู่ในลักษณะการเขียนโปรแกรมเชิงตรรกะด้วย

เงื่อนไขบังคับ และได้ปรับเปลี่ยน menuask ให้กระชับขึ้น ตัวอย่างฐานความรู้รูปแบบที่ 1 จะแสดงให้เห็นในภาพที่ 4 ส่วน menuask ที่ดัดแปลงแล้วจะแสดงไว้ในภาพที่ 5 และตัวอย่างการประมวลผลโปแกรมจะแสดงในภาพที่ 6

```
top_goal(Vout) :-
        Sin &:: ['hotel','resort'],    Lin &:: ['forest','hot spring','shopping','beach'],
%%% More Domain for Input %%%
        Vout &:: ['Phuket','Bangkok','Korat','Lumpang'],
        P &:: ['resort','beach','pearl','coral'],
        K &:: ['resort','forest','roasting chicken','stone castle'],
%%% More Domain for Goal's Component%%%
menuask(Sin), (Sin&=P ->
                (menuask(Lin),Lin&=P ->
                    (menuask(Bin),Bin&=P ->
                  (menuask(Win),Win&=P -> Vout='Phuket')
%%% Do Something %%% ).
```

ภาพที่ ฉ.4 ตัวอย่างฐานความรู้โดยใช้ข้อมูลแบบง่ายๆ เพื่อออกแบบในรูปแบบที่ 1

จากภาพที่ 4 จะเห็นว่า ผู้วิจัยใช้การกำหนดขอบเขตของตัวแปรแทนค่าที่เป็นไปได้ทั้งหมดของตัวแปรแต่ละตัว โดยแบ่งเป็น 3 กลุ่มหลักๆ คือ กลุ่มของตัวแปรสำหรับรับค่าจากผู้ใช้ (Sin, Lin, Bin, Win) กลุ่มของตัวแปรสำหรับค่าเป้าหมายที่เราต้องการ (Vout) และกลุ่มของตัวแปรที่ระบุค่าสำหรับประกอบการค้นหาคำตอบที่เป็นไปได้ เพื่อใช้ในการแทนค่าเป้าหมายที่ต้องการ (P, K, B, L) จากนั้นในส่วนของเงื่อนไขบังคับเบื้องต้นได้ออกแบบในลักษณะจำเพาะกับข้อมูลก่อน โดยใช้รูปแบบของ "A -> B ; C" หมายถึง "if A then B else C" มาช่วยในการตั้งเงื่อนไขเพื่อกำหนดค่าเป้าหมายในกรณีต่างๆ

```
:-lib(sd).
menuask(Value) :-  write('Select the value from '),writeln(Value),
                   write('Enter the choice > '), read(Value),writeln(Value),
                   asserta(known(Value)).
```

ภาพที่ ฉ.5 แสดงไลบรารี่ที่จำเป็นคือ lib(sd) และ menuask ที่ถูกดัดแปลงให้กระชับขึ้น

```
Select the value from _252{[hotel, resort]}
Enter the choice > resort
Select the value from _282{[beach, forest, hot spring, shopping]}
Enter the choice > forest
Select the value from _308{[ceramic bowl, pearl, roasting chicken]}
Enter the choice > roasting chicken
No answer found.
expert-shell>
>>>>Goodbye, see you later<<<<
```

ภาพที่ ฉ.6 แสดงผลลัพธ์ที่ได้จากระบบที่พัฒนาตามรูปแบบที่ 1

จากผลลัพธ์ในภาพที่ 6 เมื่อเทียบกับฐานความรู้ในภาพที่ 4 จะเห็นว่า โปรแกรมยังมีส่วนที่ผิดพลาด สาเหตุเกิดจากวิธีที่ใช้ในการเปรียบเทียบค่าของ "ตัวแปรที่รับจากผู้ใช้" กับ "ตัวแปรที่ระบุค่าสำหรับประกอบการค้นหาคำตอบที่เป็นไปได้" ด้วยเครื่องหมาย "&=" นั้น ทำให้ค่าของตัวแปรที่ระบุค่าสำหรับประกอบการค้นหาคำตอบที่เป็นไปได้ถูกกำหนดค่าให้ "เท่ากันกับ" ค่าของตัวแปรที่รับจากผู้ใช้ และไม่สามารถนำตัว

แปรที่ระบุค่าสำหรับประกอบการค้นหาคำตอบที่เป็นไปได้นี้ไปตรวจสอบค่ากับตัวแปรที่รับจากผู้ใช้ตัวอื่นได้อีก ทำให้คำตอบที่ได้รับจากระบบเป็น “No answer found.” แทน

รูปแบบที่ 2: เนื่องจากความผิดพลาดที่พบในรูปแบบที่ 1 ผู้วิจัยจึงเปลี่ยนวิธีตรวจสอบเงื่อนไขใหม่เป็นการตรวจสอบว่า ข้อมูลที่ได้รับมานั้นเป็น “สับเซต” ของค่าที่เป็นได้ของแต่ละเป้าหมายหรือไม่แทน และออกแบบตัวกระทำสำหรับใช้ตรวจสอบเงื่อนไขขึ้นใหม่ เพื่อให้ง่ายต่อการออกแบบฐานความรู้ โดยกลุ่มตัวแปรที่ระบุค่าสำหรับประกอบการค้นหาคำตอบที่เป็นไปได้จะเป็นการกำหนดค่าเป็นลิสต์แทนการกำหนดในลักษณะของโดเมน

```
:- lib(ordset).
:- op(701,fx,if).
:- op(702,yfx,then).
:- op(703,yfx,else).
:- op(704,yf,end).
if A then B end:- A -> B.
if A then B else C end:- A ->B;C.
```

ภาพที่ ฉ.7 แสดงไลบรารีที่จำเป็น และวิธีออกแบบตัวกระทำ

```
%%% More Domain for Input %%%
K &= ['resort','forest','roasting_chicken','stone_castle'],
%%% More Domain for Goal's Component%%%
menuask(Sin), (if ord_subset([Sin],B) then (menuask(Lin), if ord_subset([Sin,Lin],B) then
                                                                  Vout &= 'Bangkok' else
%%% Do Something %%% end).
```

ภาพที่ ฉ.8 แสดงตัวอย่างบางส่วนของฐานความรู้ในรูปแบบที่ 2

```
Select the value from _254{[hotel, resort]}
Enter the choice > resort
Select the value from _284{[beach, forest, hot spring, shopping]}
Enter the choice > forest
Select the value from _310{[ceramic bowl, pearl, roasting chicken]}
Enter the choice > roasting chicken
Select the value from _336{[coal mine, coral, stone castle]}
Enter the choice > stone castle
The answer is Korat
expert-shell>
The answer is Korat
The known storage are
[stone castle, roasting chicken, forest, resort]
expert-shell>
>>>>Goodbye, see you later<<<<
```

ภาพที่ ฉ.9 แสดงผลลัพธ์ที่ได้จากระบบที่พัฒนาตามรูปแบบที่ 2

จากภาพที่ 9 พบว่า รูปแบบที่ 2 นั้นสามารถแสดงคำถามและคำตอบได้อย่างถูกต้องตามที่กำหนดไว้ แต่เนื่องจากเป็นการพัฒนาต่อยอดจากรูปแบบที่ 1 ซึ่งสามารถใช้ได้กับข้อมูลชุดนี้เท่านั้น ผู้วิจัยจะต้องออกแบบการแสดงคำถามและคำตอบของระบบใหม่ให้สามารถใช้ได้กับข้อมูลทั่วๆ ไป ตามวัตถุประสงค์ที่กล่าวไว้ข้างต้น จึงเกิดการพัฒนาในรูปแบบที่ 3

รูปแบบที่ 3: ในการพัฒนาให้ระบบสามารถรองรับฐานความรู้ใดๆ ได้นั้น ผู้วิจัยต้องออกแบบ

ฐานความรู้ใหม่ในส่วนของการตรวจสอบเงื่อนไข และออกแบบ predicate ใหม่ชื่อ liken ให้มีการทำงานสัมพันธ์กับฐานความรู้ใหม่

```
Please select value of stay from  : _262{[hotel, resort]}
Your input is  : resort
Please select value of like from  : _292{[beach, forest, hot_spring, shopping]}
Your input is  : forest
Please select value of buy from  : _318{[ceramic_bowl, pearl, roasting_chicken]}
Your input is  : roasting_chicken
Please select value of will_see from  : _344{[coal_mine, coral, stone_castle]}
Your input is  : stone_castle
The answer is Korat
expert-shell>
The answer is Korat
The known storage are
[[stay, resort], [like, forest], [buy, roasting_chicken], [will_see, stone_castle]]
expert-shell>
>>>>Goodbye, see you later<<<<
```

ภาพที่ ฉ.10 แสดงผลลัพธ์ที่ได้จากระบบที่พัฒนาตามรูปแบบที่ 3

```
(if liken([Sin,Lin],B, [stay,like],true) then Vout &= 'Bangkok' end;
%%% More Knowledge Here%%%
 if liken([Sin,Lin,Bin,Win],K, [stay,like,buy,will_see],true) then Vout &= 'Korat' end ).
```

ภาพที่ ฉ.11 แสดงตัวอย่างลักษณะฐานความรู้ที่ใช้ได้กับ liken

```
liken([_],[Db],[Att],Flag):-
        known([Att,Old]),!,
        (if Old==Db then
                Flag = true
        else
                Flag = fail
        end).
```

```
liken([_|Tn],[Db|Td],[Att|Ta],Flag):-
        known([Att,Old]),!,
        (if Old==Db then
                liken(Tn,Td,Ta,Flag)
        else
                Flag = fail
        end).
```

```
liken([In],[Db],[Att],Flag):-
        \+known([Att,_]),
        menuask(Att,In),!,
        (if In==Db then
                Flag = true
        else
                Flag = fail
        end).
```

```
liken([In|Tn],[Db|Td],[Att|Ta],Flag):-
        \+known([Att,_]),
        menuask(Att,In),!,
        (if In==Db then
                liken(Tn,Td,Ta,Flag)
        else
                Flag = fail
        end).
```

ภาพที่ ฉ.12 แสดง predicate ชื่อ liken ที่ออกแบบใหม่

ระบบจากรูปแบบที่ 3 สามารถแสดงผลลัพธ์ได้เช่นเดียวกันกับรูปแบบที่ 2 ตามตัวอย่างในภาพที่ 10 แต่จะมีรายละเอียดของแอททริบิวต์สำหรับตัวแปรรับค่าแต่ละตัวเพิ่มเข้ามา แต่เมื่อผู้วิจัยได้นำวิธีนี้ไปทดลองกับข้อมูลเดียวกันกับที่พัฒนาด้วยภาษาโปรล็อกพบว่า ลำดับในการแสดงผลของระบบไม่เป็นไปตามที่คาดเดาไว้ และรูปแบบการทำงานของระบบซับซ้อนและมีการวนซ้ำที่ predicate ชื่อ liken มากเกินไปทำให้ไม่สามารถยอมรับได้ว่ารูปแบบที่ 3 นี้จะเป็นรูปแบบที่ดีที่สุดได้

รูปแบบที่ 4: จากการวิจัยโดยทดลองปรับเปลี่ยนรูปแบบของฐานความรู้ใหม่ในรูปแบบทั้งสามที่กล่าวไปแล้วนั้น ทำให้ผู้วิจัยต้องเปลี่ยนแนวคิดในการออกแบบมาสนใจปรับเปลี่ยนที่ตัวเปลือกระบบ

ผู้เชี่ยวชาญแทน ซึ่งจากการวิจัยพบว่า รูปแบบที่ 4 นี้มีการเปลี่ยนแปลงแค่ส่วนเล็กๆ ภายใน menuask เพียงอย่างเดียว และฐานความรู้ที่ใช้ก็ไม่จำเป็นต้องออกแบบใหม่ สามารถใช้รูปแบบเดียวกันกับฐานความรู้ของระบบที่พัฒนาด้วยภาษาโปรล็อกตามที่แสดงในภาพที่ 1

```
menuask(Pred,Value,Menu):- C&::Menu, menuask(Pred,C),  known([Pred,Value]),!.
menuask(Pred,_) :- known([Pred,_]),!.
menuask(Attribute,C) :-  write('What is the value for '), write(Attribute), writeln('?'),
                         writeln(C), write('Enter the choice > '), read(C),writeln(C),
                         asserta(known([Attribute,C])).
```

ภาพที่ ฉ.13 แสดง menuask ที่ออกแบบด้วยวิธีเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ

เมื่อเปรียบเทียบโค้ดในภาพที่ 1 กับภาพที่ 13 จะเห็นความแตกต่างที่การส่งตัวแปร บรรทัดแรกในภาพที่ 13 Menu ที่ส่งมานั้นจะอยู่ในลักษณะของลิสต์ (ดูภาพที่ 1) สามารถใช้กำหนดขอบเขตของตัวแปรตามวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับได้ และบรรทัดที่สี่ในภาพที่ 13 จะเห็นว่าสามารถใช้ตัวแปรที่มีการกำหนดขอบเขตนั้นแสดงเมนูของคำตอบได้ และใช้เป็นตัวแปรรับค่าจากผู้ใช้ด้วย

ผลการวิจัย

งานวิจัยนี้จะเป็นการทดลองเปรียบเทียบจำนวนบรรทัดของโค้ด ขนาดของไฟล์ข้อมูล และเวลาที่ใช้ในการประมวลผลระหว่างระบบที่พัฒนาในงานวิจัยนี้กับระบบที่พัฒนาด้วยภาษาโปรล็อก ภายใต้องค์ประกอบเดียวกันในเรื่อง รูปแบบการทำงานที่คล้ายคลึงกันของระบบ และฐานความรู้ที่มาจากข้อมูลเดียวกันคือฐานความรู้ของ Nursery ประกอบไปด้วยข้อมูลที่ต้องรับจากผู้ใช้จำนวน 8 ค่า (health, has_nurs, parents, social, housing, finance, form, children) ตัวอย่างฐานความรู้ที่ใช้ในการประเมินผลการวิจัยจะแสดงในภาพที่ 14 สำหรับเปลือกระบบภาษาโปรล็อก และในภาพที่ 15 สำหรับเปลือกระบบที่พัฒนาขึ้นในงานวิจัยนี้

```
top_goal(X) :- class(X).
class(very_recom) :- health(recommended), has_nurs(proper), parents(usual), (social(nonprob);
                     social(slightly_prob)), housing(convenient), finance(convenient),
                     (form(completed); form(incomplete); form(foster)).
%%% More Knowledge Here %%%
health(X) :- menuask(health, X, [recommended, priority, not_recom]).
has_nurs(X) :- menuask(has_nurs, X, [proper, less_proper, improper, critical, very_crit]).
%%% More Attribute Here %%%
```

ภาพที่ ฉ.14 ตัวอย่างฐานความรู้สำหรับเปลือกระบบผู้เชี่ยวชาญภาษาโปรล็อก

```
%%% Do Something %%%
class(very_recom) :- health(recommended), has_nurs(proper), parents(usual), social(S),
                     housing(convenient), finance(convenient), form(M),
                     S &:: [nonprob, slightly_prob], M &:: [completed, incomplete, foster].
%%% Do Something %%%
```

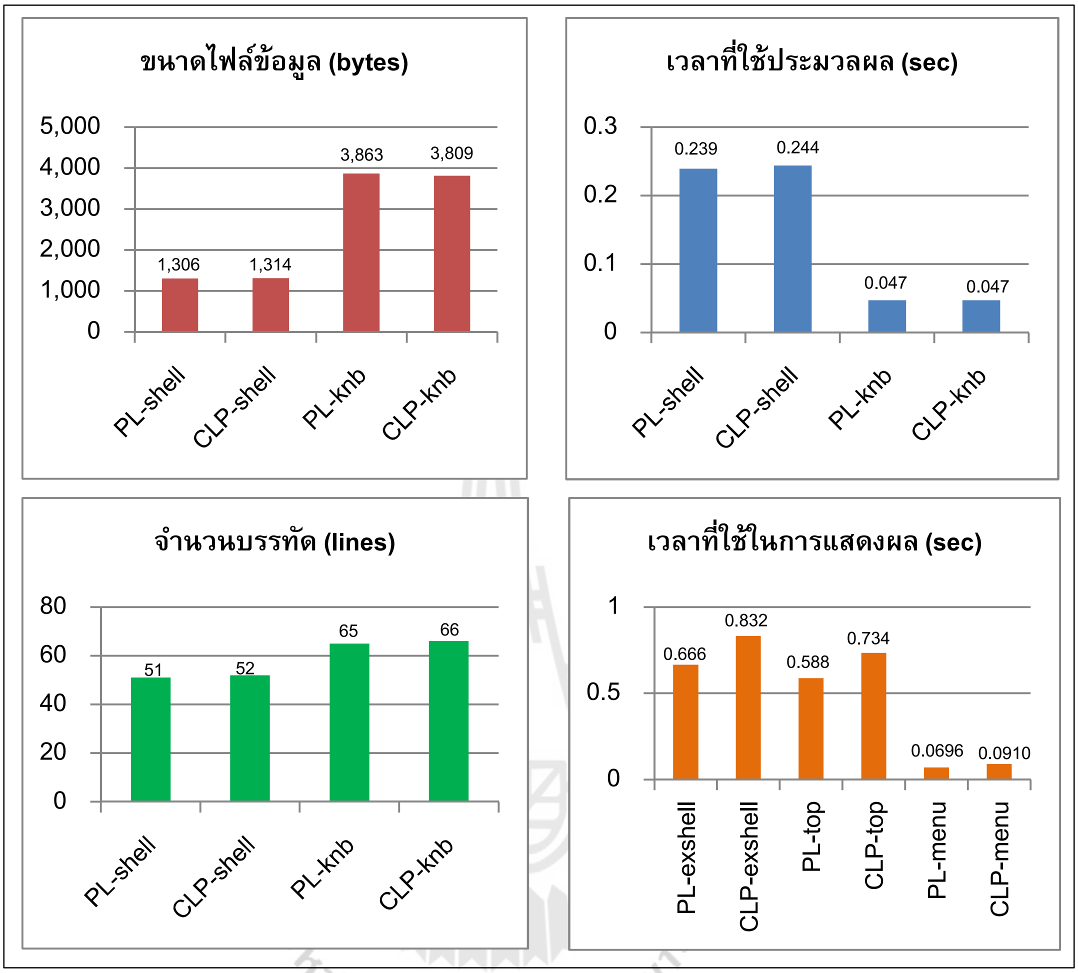
ภาพที่ ฉ.15 ตัวอย่างฐานความรู้สำหรับเปลือกระบบผู้เชี่ยวชาญที่พัฒนาขึ้น

จากภาพที่ 15 จะเห็นว่า ข้อมูลที่ “or” (;) กันตามแบบในภาพที่ 14 จะถูกแทนที่ด้วยการกำหนดขอบเขตให้ตัวแปรใหม่ ซึ่งเป็นรูปแบบในการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับที่สามารถทำได้เมื่อเปลือกระบบผู้เชี่ยวชาญนั้นมีรูปแบบในการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับเช่นเดียวกัน

ตารางที่ ฉ.2 ตารางแสดงการเปรียบเทียบปัจจัยต่างๆ ระหว่างระบบที่พัฒนาขึ้น กับระบบภาษาโปรล็อก

	ภาษาโปรล็อก			**โปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ**			**หมายเหตุ**
	เปลือกระบบ		**ฐานความรู้**	**เปลือกระบบ**		**ฐานความรู้**	
ขนาดของไฟล์	*1,306*		*3,863*	*1,314*		*3,809*	bytes
จำนวนบรรทัด	*51*		*65*	*52*		*66*	lines
เวลาประมวลผล							
ครั้งที่ 1	0.265		0.047	0.28		0.047	second
ครั้งที่ 2	0.171		0.046	0.172		0.047	second
ครั้งที่ 3	0.281		0.047	0.281		0.046	second
เฉลี่ย	*0.239*		*0.047*	*0.244*		*0.047*	second
เวลาแสดงผล	**expertshell**	**top_goal**	**Menuask**	**expertshell**	**top_goal**	**menuask**	
ครั้งที่ 1	0.858	0.749	0.0776	0.889	0.796	0.0995	second
ครั้งที่ 2	0.531	0.499	0.0669	0.905	0.796	0.0995	second
ครั้งที่ 3	0.609	0.515	0.0644	0.702	0.609	0.0741	second
เฉลี่ย	*0.666*	*0.588*	*0.0696*	*0.832*	*0.734*	*0.0910*	second

ในตารางที่ 2 จะแสดงให้เห็นว่า ในด้านของขนาดไฟล์นั้น ฐานความรู้ที่พัฒนาด้วยวิธีเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับมี “ขนาดเล็กกว่า” ฐานความรู้ภาษาโปรล็อก ซึ่งถ้าคิดถึงข้อมูลที่อาจจะมีมากขึ้นในอนาคตนับว่าส่วนนี้ส่งผลดีต่อระบบโดยรวมเป็นอย่างมาก ส่วนเปลือกระบบผู้เชี่ยวชาญที่พัฒนาด้วยวิธีเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับมี “ขนาดใหญ่กว่า” เปลือกระบบผู้เชี่ยวชาญภาษาโปรล็อกเล็กน้อย ส่วนในด้านของเวลาของเปลือกระบบผู้เชี่ยวชาญที่พัฒนาด้วยวิธีเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับทั้งที่ใช้ในการประมวลผลเวลาอ่านไฟล์ และเวลาที่ใช้ตอนแสดงผลออกหน้าจอของ predicate ต่างๆ นั้น “ช้ากว่า” ของเปลือกระบบผู้เชี่ยวชาญภาษาโปรล็อก และในด้านของจำนวนบรรทัด เนื่องจากเปลือกระบบผู้เชี่ยวชาญที่พัฒนาด้วยวิธีเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ จำเป็นจะต้องเรียกใช้ไลบารี่ของขอบเขตเชิงสัญลักษณ์ ทำให้จำนวนบรรทัดมี “มากกว่า” ของโปรล็อกอยู่หนึ่งบรรทัด



ภาพที่ ฉ.16 แสดงกราฟเปรียบเทียบค่าที่ได้จากตารางที่ 2

ในงานวิจัยนี้ได้เรียกใช้ :-lib(util) เพื่อช่วยบันทึกผลการทดลองในด้านเวลาการทำงานของ predicate ต่างๆ ตัวอย่างเช่น ถ้าต้องการบันทึกเวลาที่ใช้ใน predicate ชื่อ top_goal สามารถเขียนได้ดังนี้ “time(top_goal(X))” เมื่อ predicate ชื่อ top_goal ทำงานเสร็จโปรแกรมก็จะแสดงค่าออกทางหน้าจอ มีหน่วยเป็นวินาที

สรุปและอภิปรายผล

งานวิจัยนี้มีการออกแบบเปลือกระบบผู้เชี่ยวชาญด้วยวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับทั้งหมด 4 รูปแบบด้วยกัน รูปแบบที่ 1-3 เป็นการพัฒนาที่ยังมีความผิดพลาดบางส่วน จึงไม่เลือกเป็นโปรแกรมต้นแบบสำหรับเปรียบเทียบผลการวิจัย ซึ่งในการวัดผลการวิจัยได้เลือกรูปแบบที่ 4 ที่มีกระบวนการทำงานเหมือนกับระบบที่เป็นภาษาโปรล็อก และลักษณะการออกแบบฐานความรู้ก็คล้ายคลึงกัน จากสังเกตและบันทึกค่าผลการทดลองสรุปได้ว่า *ข้อดี*ของเปลือกระบบผู้เชี่ยวชาญที่ใช้วิธีเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับคือ ฐานความรู้ “มีขนาดของไฟล์ข้อมูลเล็กกว่า” ของเปลือกระบบ

ผู้เชี่ยวชาญภาษาโปรล็อก เพราะสามารถลดปริมาณการเขียนไฟล์ฐานความรู้ให้สั้นลงได้ และเนื่องจากการประกาศในลักษณะของขอบเขตทำให้ระบบมีการตรวจสอบขอบเขตของตัวแปรที่รับจากผู้ใช้ทันที เมื่อผู้ใช้กรอกค่าผิดระบบจะหยุดการค้นหาคำตอบช่วยลดเวลาที่อาจจะสูญเสียไปในการตอบคำถามครบทุกข้อแต่ก็ไม่ได้คำตอบเช่นกัน ส่วน*ข้อเสีย*ของเปลือกระบบผู้เชี่ยวชาญที่ใช้วิธีเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับคือ ขนาดของข้อมูลเปลือกระบบที่พัฒนาขึ้นนี้ "มีขนาดใหญ่กว่า" ของภาษาโปรล็อกเล็กน้อย ส่วนเวลาที่ใช้ประมวลผลไฟล์เข้าระบบและเวลาที่ใช้ขณะแสดงผลออกหน้าจอ "มากกว่า" เปลือกระบบผู้เชี่ยวชาญภาษาโปรล็อก และเนื่องจากการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับจำเป็นต้องเรียกใช้ไลบรารีที่จำเป็น ทำให้จำนวนบรรทัดของโค้ดเปลือกระบบผู้เชี่ยวชาญที่ใช้วิธีเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ "มากกว่า" เปลือกระบบผู้เชี่ยวชาญภาษาโปรล็อกอยู่ 1 บรรทัด

กิตติกรรมประกาศ

งานวิจัยนี้ได้รับทุนสนับสนุนจากมหาวิทยาลัยเทคโนโลยีสุรนารี

เอกสารอ้างอิง

[1] Feigenbaum, Edward A. (1992). Expert Systems: Principles and Practice. In *The Encyclopedia of Computer Science and Engineering,* Stanford: Knowledge Systems Laboratory.

[2] Salim, M.D.; Villavicencio, A.; & Timmerman, M.A. (2002). A Method for Evaluating Expert System Shells for Classroom Instruction. *Journal of Industrial Technology.* 19(1): 1-11.

[3] นิตยา เกิดประสพ. (2552). *การประมวลผลหลังกระบวนการทำเหมืองข้อมูล*. โครงการวิจัยโดยทุนอุดหนุนจากมหาวิทยาลัยเทคโนโลยีสุรนารี: รหัสโครงการ SUT7-705-48-24-13. นครราชสีมา: มหาวิทยาลัยเทคโนโลยีสุรนารี.

[4] Antoni, Niederlinski. (2011). *CLP with Elementary Constraints for Feasible Solutions. A Quick and Gentle Guide to Constraint Logic Programming via ECLiPSe*. Gliwice: pkjs.com.pl

[5] Eronen, Pasi; & Zitting, Jukka. (2001). *An Expert System for Analyzing Firewall Rules*. n.p.: CiteSeerX.

[6] Lee, Jae Kyu; & Kwon, Suhn Beom. (1995). *ES*: An Expert Systems Development Planner Using a Constraint and Rule-Based Approach*. n.p.: Expert Systems With Applications.

[7] Muli, Hastings Kyale. (1992). *Optimization Methods in Logic Programming Applied to Expert Systems for Capital Budgeting*. Dissertation, M.S. (Systems Science Programme). Ottawa: University of Ottawa.

[8] Roach, Devid; & Berghel, Hal. (1992). An Alternative to Constraint Logic Programming for Managing Domain Arithmetics in Prolog Expert Systems. In *ACM Proc. Conf. SIGAPP Symposium on Applied computing: technological challenges of the 1990's.* pp. 328-332. n.p.

ประวัติผู้เขียน

นางสาวกาญจนา อินทรชาธร เกิดเมื่อวันที่ 30 พฤษภาคม พ.ศ. 2531 ที่ อำเภอเมือง จังหวัดนครราชสีมา เริ่มเข้าศึกษาชั้นอนุบาล 3 ถึงชั้นมัธยมศึกษาปีที่ 6 ที่โรงเรียนมารีย์วิทยา จังหวัดนครราชสีมา จากนั้นได้เข้าศึกษาต่อระดับปริญญาตรีในสาขาวิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี และสำเร็จการศึกษาเมื่อปี 2552 ภายหลังสำเร็จการศึกษาในระดับปริญญาตรี ได้เข้าศึกษาต่อในระดับปริญญาโท สาขาวิชาวิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี ในปีการศึกษา 2553

ในระหว่างการศึกษาได้รับความอนุเคราะห์อย่างดียิ่งจากคณาจารย์ในสาขาวิชา และได้รับความไว้วางใจให้เป็นผู้ช่วยสอนปฏิบัติการวิชา Computer Programming

ผลงานวิจัย : ได้เสนอบทความเข้าร่วมในงานประชุมวิชาการ “ศรีนครินทรวิโรฒวิชาการ” ครั้งที่ 6 ประจำปี 2555 เรื่อง การพัฒนาเปลือกระบบผู้เชี่ยวชาญด้วยวิธีการเขียนโปรแกรมเชิงตรรกะด้วยเงื่อนไขบังคับ