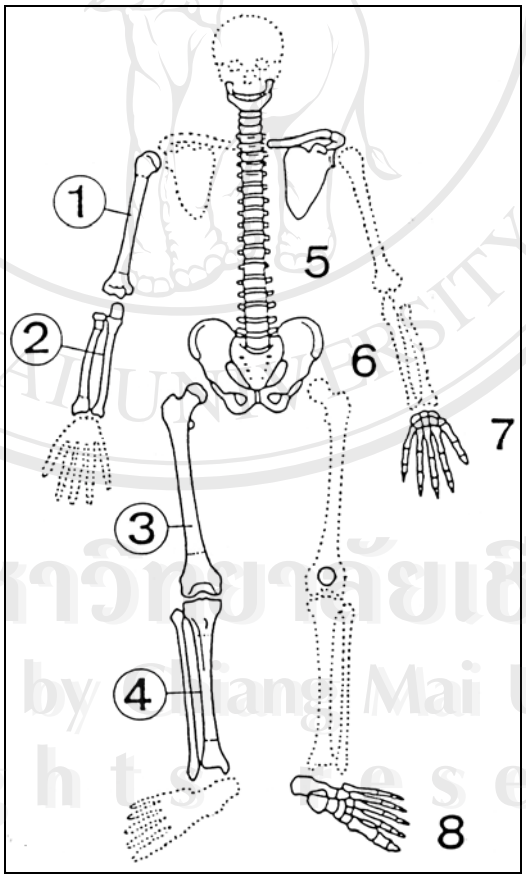
บทที่ 2
สาระสำคัญและเอกสารที่เกี่ยวข้อง

ร่างกายของมนุษย์ประกอบด้วยกระดูก 206 ชิ้น ที่มีลักษณะที่แตกต่างกันออกไป โดยกระดูกแต่ละชิ้นมีลักษณะเฉพาะตัว มีหน้าที่ที่สำคัญคือเป็นโครงสร้างของร่างกาย โดยกระดูกแต่ละชิ้นจะมีเนื้อเยื่อเกี่ยวพัน ได้แก่ เยื่อหุ้มกระดูก เส้นเลือด เส้นประสาท กล้ามเนื้อ ผิวหนัง หุ้มล้อมรอบ กระดูกโดยทั่วไปมีลักษณะที่พอจะแบ่งได้คร่าวๆคือ กระดูกที่มีลักษณะเป็นแท่งยาว กระดูกที่มีลักษณะเป็นก้อน และกระดูกที่มีลักษณะเป็นแผ่น บริเวณที่อยู่ในการศึกษาค้นคว้าอิสระนี้คือบริเวณที่มีลักษณะของกระดูกเป็นแท่งยาว คือบริเวณที่ 1, 2, 3, และ 4 ตามรูปที่ 2.1



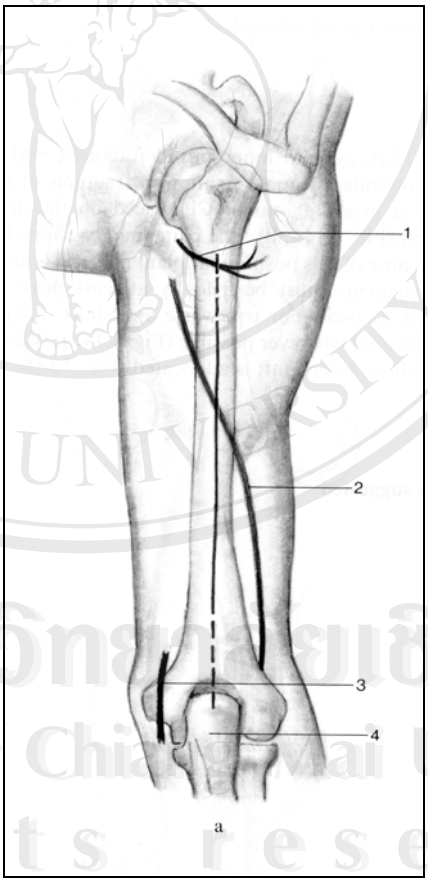
รูปที่ 2.1 ภาพตำแหน่งของกระดูกของร่างกาย

2.1 ความรู้ทางกายวิภาคพื้นฐาน

เพื่อให้เกิดความเข้าใจในลักษณะการเรียกชื่อบริเวณต่างๆของร่างกายทางการแพทย์ ผู้ศึกษาจึงขออธิบายชื่อ บริเวณตำแหน่งต่างๆของร่างกายที่เกี่ยวข้องกับการศึกษา เพื่อให้ความเข้าใจในการเรียกชื่อบริเวณของร่างกายที่กำลังศึกษาอยู่ตรงกัน

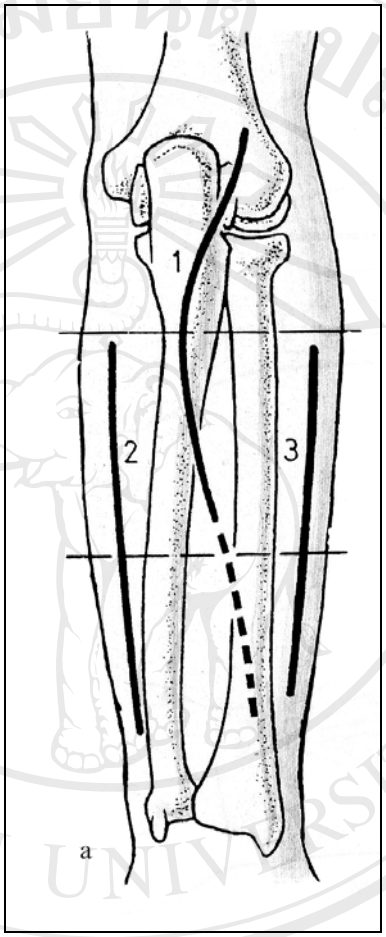
แขนในความหมายทางกายวิภาคจะหมายถึงต้นแขนและแขนส่วนปลายไม่รวมมือซึ่งถือว่าเป็นรยางค์เฉพาะ โดยแขนประกอบด้วย 2 ส่วนคือ

1. ต้นแขน(ตำแหน่งที่ 1 ตามรูปที่ 2.1) ซึ่งประกอบด้วยกระดูก 1 ชิ้น คือ กระดูกฮิวเมอร์รัส(Humerus)



รูปที่ 2.2 บริเวณต้นแขนที่ประกอบด้วยกระดูกฮิวเมอร์รัส

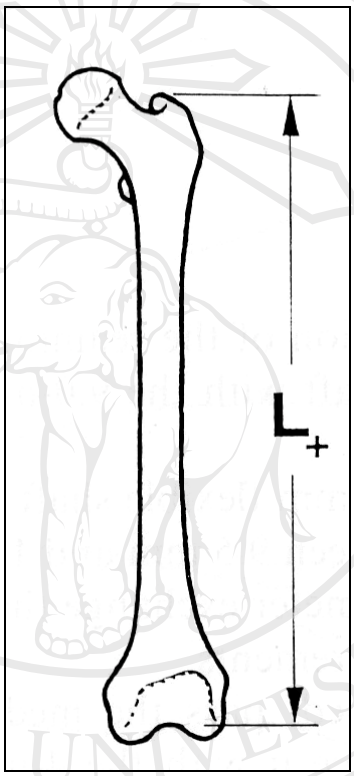
2. ปลายแขน(ตำแหน่งที่ 2 ตามรูปที่ 2.1)ประกอบด้วยกระดูก 2 ชิ้น คือ กระดูกเรเดียส (Radius) และกระดูกอัลนาร์ (Ulnar)



รูปที่ 2.3 บริเวณแขนซึ่งประกอบด้วยกระดูกสองชิ้นคือ กระดูกเรเดียส และกระดูกอัลนาร์

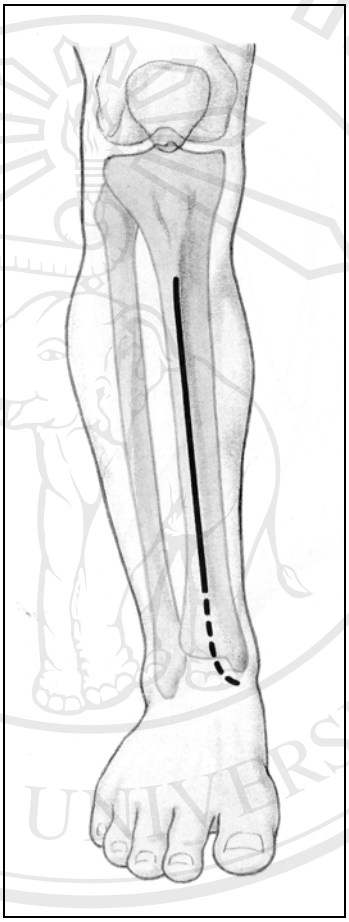
ขาในความหมายทางกายวิภาคจะหมายถึงต้นขาและขาส่วนปลายไม่รวมเท้าซึ่งถือเป็นรยางค์เฉพาะโดยขาประกอบด้วย 2 ส่วนคือ

1. ต้นขา(ตำแหน่งที่ 3 ตามรูปที่ 2.1) ซึ่งประกอบด้วยกระดูก 1 ชิ้น คือ กระดูกฟีเมอร์ (femur)



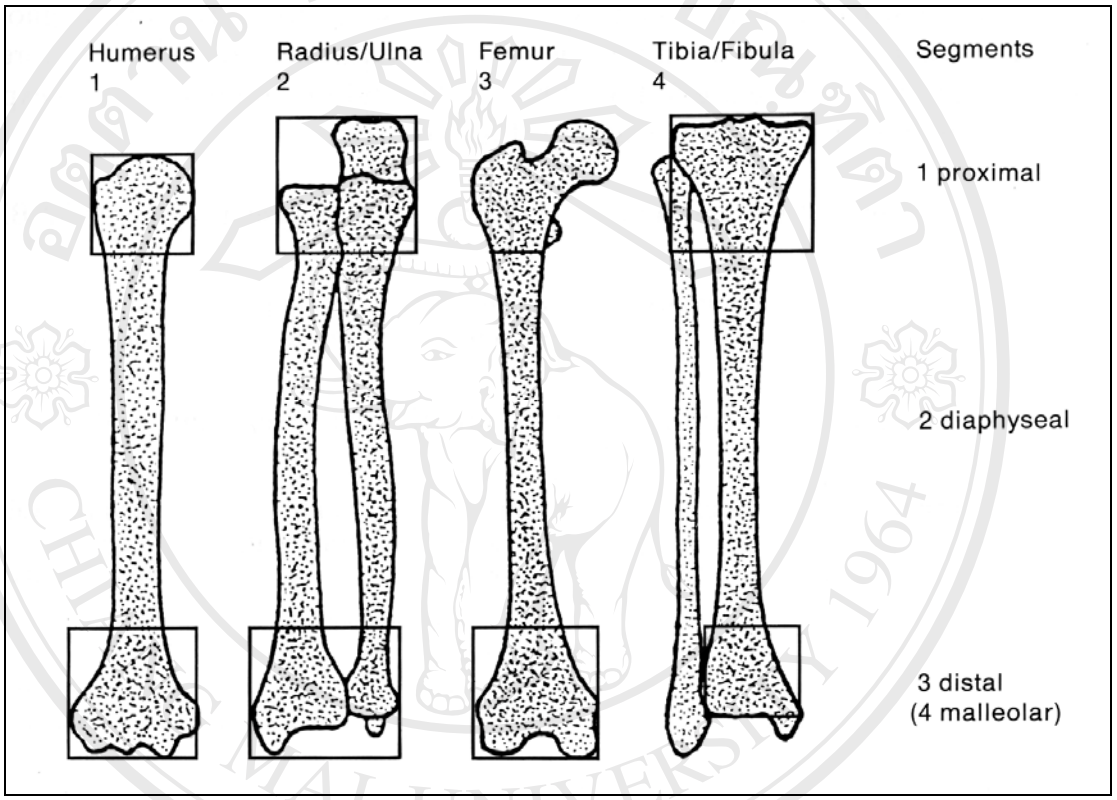
รูปที่ 2.4 กระดูกบริเวณต้นขาที่ประกอบด้วยกระดูกฟีเมอร์

2. ขา(ตำแหน่งที่ 4 ตามรูปที่ 2.1) ประกอบด้วยกระดูก 2 ชิ้น คือ กระดูกทิเบีย (Tibia) และ กระดูกฟิบูลา (Fibula)



รูปที่ 2.5 บริเวณขาที่ประกอบด้วยกระดูกสองชิ้นคือกระดูกทิเบียและกระดูกฟิบูลา

กระดูกส่วนตรงกลาง (Shaft) หมายถึงบริเวณตรงกลางของกระดูก ที่มีชื่อเรียกทางกายวิภาคว่า ไดอะไฟสิส (Diaphysis) ดังรูปที่ 2.6 ซึ่งบริเวณนี้จะมีเฉพาะกระดูกที่มีลักษณะเป็นแท่งยาว (Long bone) เท่านั้น



รูปที่ 2.6 ตำแหน่งบริเวณส่วนกลางของกระดูกแท่งยาว

2.2 ระบบผู้เชี่ยวชาญ

ระบบผู้เชี่ยวชาญ(Expert System) คือ โปรแกรมคอมพิวเตอร์ที่เก็บทั้งความรู้เกี่ยวกับปัญหาที่จะแก้ และขบวนการอนุมานเพื่อนำไปสู่ผลสรุปหรือคำตอบของปัญหานั้น ความรู้ที่เก็บมีทั้งความรู้ที่เป็นความจริงที่อาจจะถูกบันทึกไว้ในรูปของตำราหรือเอกสารทางวิชาการ และความรู้ที่ได้จากประสบการณ์ที่อาจจะไม่อยู่ในรูปของตำราหรือเอกสารทางวิชาการ แต่จะต้องดึงออกมาจากผู้เชี่ยวชาญหรือผู้ชำนาญที่มีประสบการณ์นั้น

ปัญหาที่ระบบผู้เชี่ยวชาญจะแก้ส่วนใหญ่จะเป็นปัญหาที่ยุ่งยาก และไม่ค่อยมีโครงสร้าง (Semi-structured หรือ ill-structured problem) ในปัญหาประเภทนี้คำตอบจะมีโอกาสเป็นได้หลายอย่าง ทั้งนี้ขึ้นอยู่กับสภาพขณะนั้นของปัญหาและข้อมูลที่เข้ามา ปัญหาประเภทนี้อาจจะอุปมาได้เหมือนกับ การเล่นหมากรุก การเดินหมากครั้งต่อไปนั้นเดินได้หลายวิธีด้วยกัน แต่ตัวหมากที่จะเดินดีที่สุดตัดสินจากสภาพของกระดานหมากในขณะนั้น และหมากที่คิดว่าคู่ต่อสู้จะเดินในครั้งต่อไป ในการแก้ปัญหาประเภทนี้เรามักไม่สามารถจะกำหนดขั้นตอนในการแก้อย่างชัดเจนไว้ล่วงหน้าได้ แต่จะต้องอาศัยความรู้ ประสบการณ์และสภาพของปัญหาในขณะนั้นรวมกันจึงจะแก้ได้ ดังนั้นวิธีการแก้ปัญหาแบบ ที่มีมาซึ่งเป็นแบบเขียนโปรแกรมเป็นขั้นตอนการแก้ปัญหาหรืออัลกอริธึม (Algorithm) จึงไม่สามารถจะนำมาประยุกต์ใช้ในปัญหาประเภทนี้ได้ ระบบผู้เชี่ยวชาญถึงแม้จะเป็นโปรแกรมคอมพิวเตอร์ชนิดหนึ่ง แต่โครงสร้าง และเทคนิคที่ใช้ในการสร้างหรือพัฒนาต่างจากของโปรแกรมที่มีมาและเป้าหมายใน การประยุกต์ใช้ก็แตกต่างกัน การประยุกต์ใช้ระบบผู้เชี่ยวชาญที่ประสบความสำเร็จเท่าที่มีมาได้แก่ การวินิจฉัยโรค การสำรวจทรัพยากรธรณี การวิเคราะห์โครงสร้างสารอินทรีย์เคมี และการแนะนำระบบคอมพิวเตอร์

ปัญหาสำคัญในการค้นคว้าเกี่ยวกับ “ปัญญา” หรือ “ความรู้” นั้นแยกออกเป็นปัญหาย่อยได้ 3 ปัญหาที่เกี่ยวข้องกัน คือ ปัญหาการแสดงความรู้ ปัญหาการใช้ความรู้ และปัญหาการรับความรู้

1. ปัญหาการแสดงความรู้ คือปัญหาการเลือกแบบในการแสดงความรู้ ความรู้นี้อาจจะได้แก่ความรู้ที่เป็นความจริงที่ปรากฏในตำรา หรือเอกสารทางวิชาการต่าง ๆ หรืออาจจะเป็นความรู้ที่ได้จากประสบการณ์ที่สะสมมานานของผู้รู้หรือผู้เชี่ยวชาญ รูปแบบการแสดงความรู้ควรจะต้องสามารถใช้ได้กับความรู้ทั้ง 2 ชนิด ควรเหมาะแก่การเก็บในคอมพิวเตอร์ และควรเป็นรูปแบบที่ทำให้การนำความรู้ไปใช้ทำได้ง่าย ปัญหาการแสดงความรู้อาจจะพูดได้อีกอย่างว่า เป็นปัญหาการออกแบบฐานความรู้ (Knowledge base)

2. ปัญหาการใช้ความรู้ คือปัญหาวิธีการนำเอาความรู้ที่เก็บอยู่ในรูปแบบหนึ่งมาใช้ในการแก้ปัญหา เรียกปัญหานี้ได้อีกอย่างหนึ่งว่าเป็นปัญหาการออกแบบเครื่องอนุมาน (Inference engine) ความสัมพันธ์ระหว่างการแสดงความรู้กับการใช้ความรู้นั้นเปรียบเสมือนกับความสัมพันธ์ระหว่างโครงสร้างข้อมูล (Data structure) กับอัลกอริธึม (Algorithm)

3. ปัญหาการรับความรู้ คือ ปัญหาการถ่ายเทความรู้ที่เป็นทั้งความรู้ที่ได้จากตำราและความรู้ที่ได้จากประสบการณ์เกี่ยวกับปัญหาที่ต้องการแก้ไขไปยังฐานความรู้ หัวข้อสำคัญ ๆ ในปัญหานี้ ได้แก่ วิธีการดึงเอาความรู้จากผู้เชี่ยวชาญ การสร้างฐานความรู้ที่สมบูรณ์ และไม่ขัดแย้งในตัวเอง และวิธีการรับความรู้แบบอัตโนมัติด้วยการเรียนด้วยตนเอง

2.2.1. โครงสร้างพื้นฐานของระบบผู้เชี่ยวชาญ

ระบบผู้เชี่ยวชาญโดยทั่วไปจะประกอบด้วยส่วนประกอบพื้นฐาน 5 ส่วน รายละเอียดของแต่ละส่วนสามารถอธิบายได้ดังนี้

1. ฐานความรู้ ส่วนนี้เปรียบเสมือนกับข้อมูลในซอฟต์แวร์ธรรมดา หรือฐานข้อมูล (database) ในระบบสารสนเทศ (Information system) เป็นส่วนที่ใช้เก็บความรู้ทุกประเภทไม่ว่าจะเป็นความรู้ที่ได้จากตำราหรือความรู้ที่ได้จากประสบการณ์ ปัญหาหลักของฐานความรู้คือ การเลือกวิธีการแสดงความรู้หรือโครงสร้างสำหรับเก็บความรู้ที่เหมาะสมปัญหานี้เปรียบได้กับการเลือกโครงสร้างข้อมูลหรือโครงสร้างฐานข้อมูลที่เหมาะสมในระบบซอฟต์แวร์ธรรมดา

2. เครื่องอนุมาน ส่วนนี้เปรียบได้กับอัลกอริธึม เป็นส่วนที่ควบคุมการใช้ความรู้ในฐาน ความรู้เพื่อแก้ไขปัญหาอย่างมีประสิทธิภาพ วิธีการอนุมานมีหลายแบบแต่แยกเป็นประเภทใหญ่ ๆ ได้ 2 ประเภท คือ อนุมานแบบเดินหน้า (Forward chaining inference) และอนุมานแบบย้อนหลัง (Backward chaining inference) ทั้งสองวิธีนี้ต่างก็มีจุดดี และจุดเสีย ทั้งนี้ขึ้นอยู่กับลักษณะของปัญหา ในระบบผู้เชี่ยวชาญบางระบบ จะใช้วิธีอนุมานทั้งสองวิธีรวมกัน

3. ส่วนดึงความรู้ (Knowledge acquisition subsystem) เป็นส่วนของระบบผู้เชี่ยวชาญที่ใช้ช่วยในการดึงเอาความรู้จากตำราหรือฐานข้อมูลและจากผู้เชี่ยวชาญ การดึงเอาความรู้จากตำราหรือฐานข้อมูลนั้นทำได้ไม่ยาก ถ้าหากเราสามารถจัดความรู้จากแหล่งดังกล่าวให้เป็นระบบ และเข้ากันได้กับโครงสร้างของฐานความรู้ เราก็จะสามารถบรรจุความรู้เหล่านั้น เข้าไปในฐานข้อมูลได้ แต่ทว่าการดึงเอาความรู้จากผู้เชี่ยวชาญนั้นทำได้ยาก จำเป็นต้องใช้เทคนิคต่าง ๆ เข้าช่วยหรือไม่ก็ทำให้ระบบผู้เชี่ยวชาญสามารถเรียนรู้ด้วยตนเองใน บางส่วนได้ ปัจจุบันการเรียนรู้เป็นหัวข้อค้นคว้าที่นักค้นคว้าในสาขาปัญญาประดิษฐ์ให้ความสนใจมากที่สุดหัวข้อหนึ่ง

4. ส่วนอธิบาย (Explanation sub-system) ส่วนนี้ทำหน้าที่อธิบายรายละเอียดของขั้นตอน การวินิจฉัยต่อผู้ใช้ว่าข้อสรุปหรือคำตอบนั้นได้มาอย่างไรและทำไม

5. ส่วนติดต่อกับผู้ใช้ (User interface) เป็นส่วนที่เป็นตัวกลางระหว่างผู้ใช้กับระบบเพื่อทำให้การสื่อสารระหว่างผู้ใช้กับระบบเป็นไปได้อย่างราบรื่น และช่วยทำให้ผู้ใช้ยอมรับระบบมากขึ้น

ในระบบผู้เชี่ยวชาญบางระบบจะไม่มีส่วนประกอบครบทั้งห้าส่วนดังกล่าวข้างต้น แต่ที่ขาด ไม่ได้แน่ ๆ คือ ฐานความรู้ และเครื่องอนุมาน

2.2.2. คุณลักษณะของระบบผู้เชี่ยวชาญ

1. High Performance ต้องมีความสามารถในการตอบสนองให้ดีกว่าหรือเท่ากับผู้เชี่ยวชาญที่เป็นมนุษย์

2. Adequate response time ต้องแสดงการตอบสนองได้รวดเร็วกว่า หรือเท่ากับผู้เชี่ยวชาญจริง

3. Good reliability ควรมีความน่าเชื่อถือและไม่เอนเอียง

2.2.3. ส่วนประกอบของระบบเชี่ยวชาญ

1. user interface เป็นกลไกที่ช่วยให้ user และ expert system ติดต่อสื่อสารระหว่างกันได้

2. explanation memory อธิบายเหตุผลของระบบให้ user ได้รับรู้

3. working memory เป็น global memory ของ facts ที่ถูกใช้โดย rules

4. inference engine ทำการตัดสินใจว่า rules ใดที่ match กับ facts หรือ objects ถ้ามีหลาย rules ที่ match ก็จะเรียงลำดับตามความสำคัญก่อนหลังและทำการ execute rule ที่มีความสำคัญสูงสุด

5. agenda เป็น list ของ rules ที่เรียงลำดับตามความสำคัญก่อนหลังและทำการ execute rule ที่มีความสำคัญสูงสุด

6. knowledge acquisition facility เป็นวิธีการที่ user จะสามารถนำความรู้เข้าสู่ระบบได้โดยอัตโนมัติ

2.2.4. Expert system programming paradigm

1. Programming languages โดยทั่วไปเช่น FORTRAN และ C ถูกออกแบบมาให้เหมาะกับ การจัดการข้อมูลแบบ procedural (เช่น numbers และ arrays) ในการแก้ปัญหาที่ซับซ้อนของมนุษย์ บ่อยครั้งใช้กลยุทธ์ที่เป็นนามธรรมหรือสัญลักษณ์ ซึ่งภาษาการโปรแกรมทั่วๆไป จะต้องมีการแปลงข้อมูลให้อยู่ในรูปแบบที่สามารถใช้ได้โดยหลัก procedural

2. จากการค้นคว้าในแขนงวิชา Artificial Intelligence (AI) ได้มีการพัฒนาเทคนิคซึ่งอนุญาตให้ทำการสร้างข้อมูลสารสนเทศในรูปของนามธรรมระดับสูงขึ้นได้

3. เทคนิคเหล่านี้ถูกประกอบเข้าด้วยกันใน language หรือ tools ซึ่งอนุญาตให้โปรแกรมที่จะสร้างขึ้นมีความคล้ายคลึงกับตรรกะของมนุษย์ จึงเป็นการง่ายต่อการพัฒนาและบำรุงรักษามากกว่า

4. โปรแกรมเหล่านี้จะเจริญรอยตามความเชี่ยวชาญของมนุษย์ในการระบุของขอบข่ายของปัญหาที่ดี เรียกว่า expert system tool เช่น CLIPS ซึ่งช่วยลดความเพียรพยายาม และค่าใช้จ่ายในการพัฒนา expert system ลงอย่างมาก

2.2.5 Rule-based programming

1. เป็นหนึ่งในเทคนิคทั่วๆไปที่ใช้สำหรับการพัฒนาระบบผู้เชี่ยวชาญ

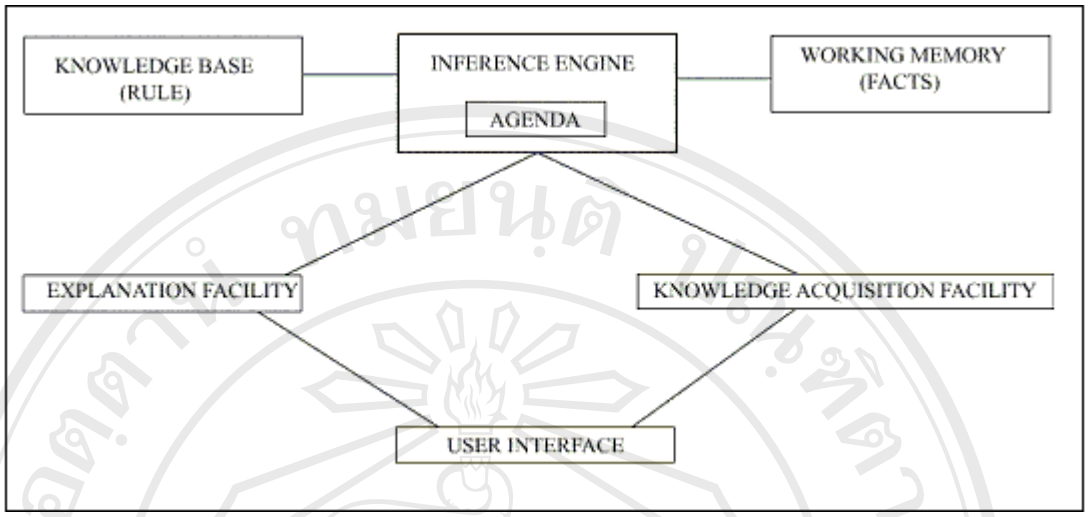
2. หลักวิธีการโปรแกรมนี้ rules จะแสดงการค้นหาได้ด้วยตัวเอง (heuristics) โดยเซตของ actions ที่จำเพาะเจาะจงจะถูกแสดงในสถานการณ์ต่างๆที่ให้มา

3. rule จะประกอบไปด้วยส่วนของ IF และส่วนของ THEN

4. ในส่วนของ IF จะเป็นชุดของ patterns ที่ระบุ facts (หรือ data) ที่จำเพาะ ซึ่งเป็นเหตุให้ rule นั้นๆสามารถปฏิบัติงานได้

5. กระบวนการในการ match facts เข้ากับ patterns นี้เรียกว่า pattern matching

6. expert system tool ได้จัดเตรียมกลไกที่เรียกว่า inference engine ไว้ ซึ่งจะทำการ match facts เข้ากับ patterns ให้โดยอัตโนมัติ และตัดสินใจว่า rule ใดจะสามารถนำไปปฏิบัติงานได้บ้าง



รูปที่ 2.7 โครงสร้างของ Rule-based Expert System

รูปที่ 2.7 แสดงโครงสร้างของ Rule-Base Expert system ซึ่งประกอบด้วย ส่วนสำคัญคือ Inference engine, Knowledge Acquisition Facility, Explanation Facility, User interface โดยในส่วนของ Inference engine ประกอบด้วยสิ่งสำคัญ 3 ส่วนคือ Knowledge base (Rule) ที่มีหน้าที่เก็บส่วนของ actions ที่เฉพาะเจาะจงต่อสถานการณ์ต่างๆ Working memory (Facts) คือเหตุที่จำเพาะซึ่งเป็นเหตุให้ Rule นั้นๆทำงาน และส่วนสุดท้ายคือ inference engine ที่ทำหน้าที่ macth facts กับ Rule

2.2.6 Rule-based system

1. knowledge base : บรรจุความรู้ต่างๆที่จำเป็นต้องใช้ในการแก้ปัญหา เขียนอยู่ในรูปของ rules

2. rules : หลักวิธีการที่ใช้ในการแสดงความรู้ โดยทั่วไปจะเป็นแบบ IF THEN ดังรูปที่ 2.8

IF the light is red THEN stop

รูปที่ 2.8 หลักวิธีการที่ใช้ในการแสดงความรู้

- ถ้า fact ที่มีอยู่คือ light is red
- fact นั้นจะไป match กับ pattern " light is red " ในส่วน IF ของ rule
- ทำให้เกิด action " stop " ในส่วนของ THEN ของ rule

สำหรับในส่วนของ THEN นั้นจะเป็นชุดของ actions ที่จะถูก execute ได้ โดย inference engine จะทำการเลือก rule ขึ้นมา action ของ rule ที่เลือกขึ้นมานี้ก็ถูก execute ต่อ inference engine จะทำการเลือก rule ที่ถูกพิจารณาไว้แล้วมา execute อีกเรื่อยๆจนกระทั่ง rule ที่พิจารณาไว้ว่าสามารถปฏิบัติงานได้

2.3 CLIPS

CLIPS ย่อมาจาก C Language Integrated Production System คือ เครื่องมือที่สามารถใช้ในการพัฒนา และมอบส่งระบบผู้เชี่ยวชาญได้ (expert system) และเป็น multiparadigm programming language ซึ่งได้จัดเตรียมสภาวะแวดล้อมที่สมบูรณ์ไว้ให้สำหรับสนับสนุนการเขียนโปรแกรมแบบ rule-based, object-oriented และ procedural ถูกสร้างขึ้นในปี ค.ศ.1985 โดยองค์การ NASA สาขา Software Technology ของ Johnson Space Center เมืองฮุสตัน รัฐเทกซัส ด้วยความสามารถจำเพาะของการจัดหาภาษาที่มี high portability ค่าใช้จ่ายต่ำ และรวมเข้ากับระบบภายนอก(external system)ได้ง่าย ซึ่งพัฒนาขึ้นโดยใช้ภาษา C ปัจจุบันมีการใช้งานกันอย่างกว้างขวางทั้งในภาครัฐบาล อุตสาหกรรม และการศึกษา

2.3.1 คุณลักษณะของ CLIPS

1. Knowledge Representation

1) CLIPS ได้จัดเตรียมเครื่องมือที่ช่วยจัดการความรู้ได้กว้างขวาง และหลากหลายโดยสนับสนุนการโปรแกรมที่แตกต่างกันทั้ง 3 แบบคือ rule-based, object-oriented และ procedural

2) โดยหลักของ Rule-based programming ความรู้ถูกแสดงออกมาโดยการค้นหาได้ด้วยตัวเอง (heuristics) ซึ่งจะระบุเซตของ actions ที่จะถูกกระทำในแต่ละ สถานการณ์ (situation) ที่ได้รับมา

2. Portability

1) CLIPS ถูกเขียนขึ้นโดยภาษา C เพื่อให้มี portability , speed และติดตั้งได้หลากหลาย operating system ที่ต่างกัน โดยปราศจากการเปลี่ยนแปลง code

2) Operating system ที่ CLIPS ได้ทำการทดสอบแล้วประกอบไปด้วย Window 95/98/NT, MacOS X และ Unix

3) CLIPS สามารถนำไปใช้ได้ในระบบต่างๆที่มีมาตรฐาน ANSI ในการยอมรับ compiler ของภาษา C หรือ C++ source code ทั้งหมดของ CLIPS สามารถทำการปรับปรุงเปลี่ยนแปลงเพื่อให้เป็นไปตามความต้องการที่จำเพาะเจาะจงของผู้ใช้ได้

3. Integration/Extensibility

1) CLIPS สามารถฝังตรึงเข้าใน procedural code ได้ เรียกว่า "subroutine"

2) และรวมเข้าด้วยกันกับภาษาอย่างเช่น C, Java, FORTRAN และ Ada ได้

3) CLIPS สามารถขยายออกไปโดย user ได้ง่าย ผ่านทาง Protocol ต่างๆ

4. Interactive Development

1) standard version ของ CLIPS จะมีการจัดเตรียมในเรื่องของ interactive, text oriented development environment, ความช่วยเหลือในการ debugging , On-line help และ integrated editor ไว้ให้

2) สำหรับ interfaces นั้น ได้มีการจัดเตรียมลักษณะต่างๆอย่างเช่น pull down menus, integrated editors และ multiple windows ซึ่งถูกพัฒนาขึ้นสำหรับสภาวะแวดล้อมของ MacOS, Windows 95/98/NT และ X Window

5. Verification/Validation

1) CLIPS ได้รวมเอาการตรวจสอบความถูกต้องตามที่ได้ออกแบบมา และความถูกต้องตามความต้องการของ expert system เข้าไว้ด้วย

2) รวมถึงการสนับสนุนสำหรับ modular design และ การแบ่ง knowledge base ออกเป็นส่วนๆ

3) static และ dynamic constraint ในการตรวจสอบของ slot values และ function arguments

4) และสุดท้าย semantic analysis (การวิเคราะห์เกี่ยวกับความหมายของคำ) ของ rule patterns เพื่อพิจารณาว่าทำให้เกิด error ขึ้นได้หรือไม่

6. Fully Documented CLIPS จะมาพร้อมกับเอกสารต่างๆรวมถึง reference manual และ user's guide

7. Low Cost CLIPS เป็น public domain software

2.3.2 Programming Paradigm

Programming Paradigmใน CLIPS โดย CLIPS สนับสนุนทั้ง 3 รูปแบบของ programming paradigm ดังนี้

1. CLIPS's rule-based programming language:
   1) มีความคล้ายคลึงกับภาษา OPS5 แต่มีความสามารถมากกว่า
   2) Syntax ของ CLIPS นั้นมีความใกล้เคียงกันอย่างมากกับ rule ในภาษาอย่างเช่นART, ART-IM, Eclipse และ Cognate
   3) CLIPS สนับสนุน Forward chaining rules เท่านั้น
2. CLIPS's object-oriented programming language:

   CLIPS Object-Oriented Language (COOL) จะมีลักษณะต่างๆที่พบได้ใน object-oriented language อื่นๆเช่น Common Lisp Object System (CLOS) และ Smalltalk แต่จะมีแนวคิดใหม่ๆเพิ่มขึ้นมาด้วย
3. CLIPS's procedural programming language:

   Procedural programming language ที่สนับสนุนโดย CLIPS จะมีลักษณะคล้ายกับในภาษาอย่างเช่น C, Ada และPascal และมี syntax คล้ายกับภาษาLISP มาก

2.3.3 Rule-based expert system ที่พบใน CLIPS

มีส่วนประกอบพื้นฐานคือ

- fact list บรรจุ data ที่ได้รับการแสดง
- knowledge base บรรจุ rule ทั้งหมด
- inference engine : ควบคุมการ execute ทั้งหมด

สามารถอธิบายภาพรวมได้ดังนี้

1. knowledge base บรรจุความรู้ต่างๆที่จำเป็นต้องใช้ในการแก้ปัญหา เขียนอยู่ในรูปของ rules โดย rules หลักวิธีการที่ใช้ในการแสดงความรู้ โดยทั่วไปจะเป็นแบบ IF THEN

2. fact list จะบรรจุ data ที่ได้รับการแสดงเช่น บุคคลชื่อ Bob อาจให้เป็น fact หนึ่งใน fact list ได้

3. Inference engine ควบคุมการ execute ทั้งหมดดังนี้

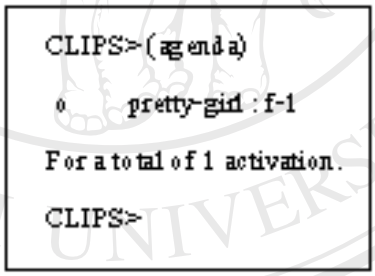
1) rules ที่มี facts มา match กับส่วน pattern ของมัน จะทำให้เกิด activation ขึ้น

2) activation นี้จะถูกเก็บไว้ใน agenda

3) และมี property ที่เรียกว่า Refraction มาช่วยป้องกัน rule จากการถูก active โดย facts เดิมอีกด้วย

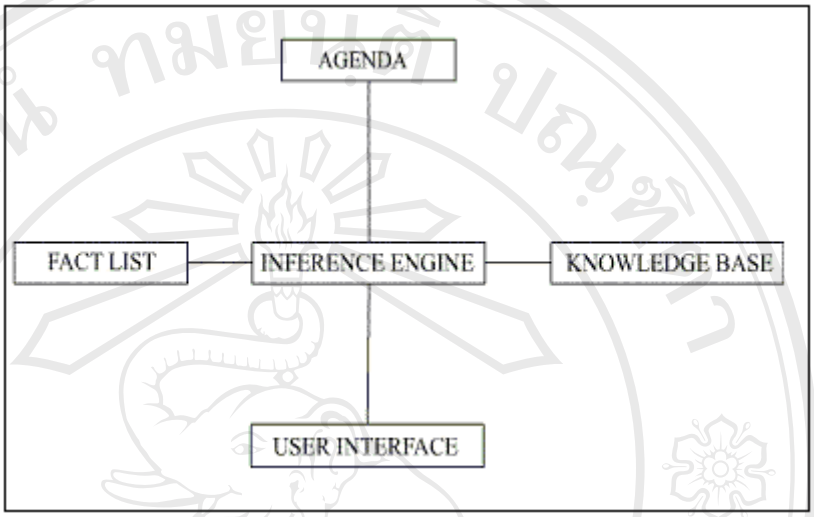
4) Agenda คือ list ของ rule ที่เรียงลำดับตามความสำคัญ ถูกสร้างขึ้นมาโดย inference engine ซึ่ง rule เหล่านี้จะต้องเป็น rule ที่มี pattern match กับ facts ที่มีอยู่ใน fact list แต่ละ entry ใน agenda แบ่งเป็น 3 ส่วนดังนี้

1. salience บอกลำดับความสำคัญของ rule ที่มีอยู่ใน agenda
2. rule name บอกชื่อของ rule ที่มี อยู่ใน agenda
3. fact identifier บอก identifier ที่ใช้ชี้ fact ที่ match กันกับ rule นั้นๆซึ่งอาจจะมีมากกว่า 1 fact ก็ได้เช่น



รูปที่ 2.9 fact identifier

ส่วนประกอบของ CLIPS rule-based expert system นี้ในตำราบางเล่มอาจจะเพิ่มในส่วนของ agenda และ user interface เข้าไว้ด้วย ดังรูปที่ 2.10



รูปที่ 2.10 ส่วนประกอบของ CLIPS rule-based expert system

2.3.4 Construct ใน CLIPS

Construct จากส่วนกลางของ program คือ ความรู้ที่เพิ่มลงในสภาวะแวดล้อมของ CLIPS ซึ่งจะต่างจาก function และ commands โดย Construct ใน CLIPS มีอยู่ 3 รูปแบบ คือ

1. Deftemplate construct :

1) ก่อนที่จะสร้าง facts ขึ้นมาได้นั้น ต้องมีการกำหนดรูปแบบโครงสร้างของ facts ก่อน

2) เพื่อนิยามกลุ่มของ facts ที่มีชื่อเดียวกัน เก็บ information ทั่วๆไป เหมือนกัน

3) ทั้งหมดจะอธิบายโดยใช้ deftemplate construct ซึ่งแบ่งเป็น 2 ประเภทคือ

- Implied deftemplate (โดยนัย) : เมื่อ CLIPS พบ ordered fact มันจะทำการสร้าง deftemplate ให้กับ fact นั้นๆโดยอัตโนมัติ
- Explicit deftemplate (โดยชัดแจ้ง) : เป็น deftemplate ที่สร้างขึ้นก่อนการสร้าง facts ใช้อธิบายรายละเอียดและรูปแบบของ fact ที่จะสร้าง

2. Deffacts construct

1) เป็นการ assert ชุดของ fact โดยอัตโนมัติ แทนการพิมพ์ใส่ใน top level

2) ใช้กับ fact ที่เป็นจริงเสมอ เป็น initial knowledge ก่อนการ run program

3) ซึ่ง fact นี้จะถูกนิยามโดยใช้ deffacts construct

4) แต่ในการ assert fact ลงไปใน fact list นั้นต้องใช้คำสั่ง reset ข้อมูล initial knowledge จึงจะถูก assert เข้าไปใน fact list ได้

3. Defrule construct

1) เป็นการสร้างกฎขึ้นมา ซึ่งถ้า rule name ซ้ำกันก็จะทับ rule เดิม

2) กฎเหล่านี้จะทำงานในส่วนของ actions เมื่อมี fact ที่อยู่ใน fact list มา match กับส่วนของ pattern ของมัน โดยทำการตรวจสอบ patterns กับทุกๆ fact ที่อยู่ใน list

2.3.5 การสร้าง Construct

การสร้าง Construct เหล่านี้ ทำได้ 2 วิธี

1. Key เข้าไปโดยตรงที่ Top level
2. Load จาก file ที่สร้าง constructs ไว้ใน text editor โดยใช้คำสั่ง load Facts

CLIPS Program จะต้องมี data หรือ information ในการที่จะคิดหาเหตุผลเพื่อแก้ปัญหาต่างๆ information ชิ้นใหญ่ๆชิ้นหนึ่งใน CLIPS จะถูกเรียกว่า fact ซึ่งจะประกอบด้วย relation name ตามด้วย zero or more slots และค่าของมัน (values) สำหรับลำดับของ slot นั้นจะวางอย่างไรก็ได้ CLIPS จะไม่สนใจ

2.3.6 fact

fact มีอยู่ด้วยกัน 2 ประเภท คือ

1. Ordered facts คือ facts ที่ไม่มีการอธิบายไว้ใน deftemplate จะประกอบไปด้วย multifield slot เพียงอัน เดียว ซึ่ง value จะเป็น zero or more ใช้ใน 2 กรณีคือ

1) fact ที่ประกอบไปด้วย relation name เพียงอย่างเดียว(ไม่มีvalue) : จะมีประโยชน์ ใช้เป็นสัญญาณ (flag) ในการบอก status ต่างๆ

2) fact ที่มีเพียง slot เดียว (โดยทั่วไปแล้ว slot name จะใช้เป็นเหมือนกับ relation name) : ใช้เก็บทุกๆค่าที่ตามหลัง relation name (เนื่องจากมีเพียง 1 slot จึงไม่มีความจำเป็นต้องใช้ slot name ในการนิยาม fact)

2. Deftemplate facts คือ fact ที่สร้างขึ้นตามรูปแบบที่ deftemplate ได้อธิบายไว้ โดยทั่วไปแล้ว ถ้าเป็นไปได้ควรจะใช้ deftemplate fact เนื่องจาก slot name จะช่วยให้ facts นั้นๆ อ่านได้ง่ายและใช้งานง่าย ดังตัวอย่าง

ตัวอย่าง

```
CLIPS> (assert (duck))
<Fact-0>
CLIPS> for a total of 1 fact.
CLIPS> (clear)
CLIPS> (assert (duck))
<Fact-0>
CLIPS> (facts)
f-0     (duck)
For a total of 1 fact.
CLIPS> (reset)
CLIPS> (facts)
f-0     (initial-fact)
For a total of 1 fact.
CLIPS> (assert (duck))
<Fact-1>
CLIPS> (facts)
f-0     (initial-fact)
f-1     (duck)
```

```
For a total of 2 facts.

CLIPS> (assert (duck))

FALSE

CLIPS> (facts)

f-0     (initial-fact)

f-1     (duck)

For a total of 2 facts.

CLIPS> (assert (quack))

<Fact-2>

CLIPS> (facts)

f-0     (initial-fact)

f-1     (duck)

f-2     (quack)

For a total of 3 facts.

CLIPS> (facts)

f-0     (initial-fact)

f-1     (duck)

f-2     (quack)

For a total of 3 facts.

CLIPS> (clear)
```

ปัจจุบันได้มีการนำภาษาระบบผู้เชี่ยวชาญ CLIPS มาพัฒนา Software expert system อย่าง กว้างขวาง โดยใช้ประโยชน์ในสาขาวิชาหลายแขนง ซึ่งมีตัวอย่างดังนี้

1. ด้านการแพทย์ ได้แก่ การพัฒนาต้นแบบผู้เชี่ยวชาญเพื่อแก้ปัญหายาเม็ด (นภดล ชะลอธรรม, 2546)

2. ด้านวิศวกรรมศาสตร์ ได้แก่ ระบบให้คำปรึกษาและการวิเคราะห์โครงสร้าง (Joseph Giarratano,1998)

3. ด้านวิทยาศาสตร์ ได้แก่ การสำรวจทรัพยากรธรณีวิทยา การวิเคราะห์โครงสร้างสารอินทรีย์เคมี และการแนะนำระบบคอมพิวเตอร์(Joseph Giarratano,1998)

CLIPS เป็น Expert System Tool ตัวหนึ่งที่มีองค์ประกอบพื้นฐานกลไก และความสามารถ ในการที่จะพัฒนาระบบผู้เชี่ยวชาญขึ้นมาได้ เช่น facts, knowledge base, agenda และ inference engine มีความเบาของภาษา สามารถติดตั้งใช้งานได้ตั้งแต่เครื่อง PC ไปจนถึง CRAY Supercomputer โดย ไม่จำเป็นต้องเปลี่ยนแปลง code แต่จะแตกต่างที่รูปแบบบางประการขึ้นอยู่กับ OS ที่ใช้ เช่น การระบุ path ของ file CLIPS สามารถนำไปประยุกต์ใช้งานได้ในหลากหลายสาขาวิชาชีพ เช่น แพทย์ , วิศวกรรมศาสตร์ , วิทยาศาสตร์ โดย expert system ที่สร้างขึ้นจะเป็นที่ปรึกษาที่รวดเร็ว แน่นอน ค่าใช้จ่ายต่ำ และไม่มีวันตาย ส่วนข้อจำกัดทางด้าน GUI ควรมีการศึกษาเพิ่มเติม โดยใช้ tool อื่นๆ เข้ามาช่วยเช่น JESS หรือ Visual basic ที่ผู้ศึกษาใช้อยู่เป็นต้น

2.4 CLIPS ActiveX Control

ActiveX เป็นเทคโนโลยีอีกชนิดหนึ่งซึ่งบริษัทไมโครซอฟท์ ได้ค้นคิดขึ้น โดยมีวัตถุประสงค์หลักเพื่อสร้างระบบซอฟต์แวร์ที่มุ่งเน้นการทำงานเฉพาะอย่าง โดยสามารถนำกลับมาใช้ใหม่ได้โดยง่าย มีความยืดหยุ่นสูง สามารถติดต่อกับซอฟต์แวร์ Platform ได้โดยง่าย

ActiveX Control หมายถึงคอนโทรลที่เพิ่มเติมจากออบเจ็คพื้นฐานที่ Visual Basic ได้เตรียมไว้แล้วในกล่องเครื่องมือ มีหลายบริษัทมากมายพัฒนาคอนโทรลที่ทำงานเฉพาะอย่างออกมาขายมากมายทั่วโลก CLIPS ActiveX เป็นคอนโทรลอีกชนิดหนึ่ง ที่ทำงานในหน้าที่ของ CLIPS Expert System Shell บนระบบปฏิบัติการวินโดวส์ 32 บิต มีความเข้ากันได้กับการเขียนไวยากรณ์แบบCLIPSที่ทำงานแบบ command line ทุกประการ

CLIPS ActiveX control ที่ผู้ศึกษาได้นำมาใช้ในการศึกษาค้นคว้าอิสระนี้เป็นผลผลิตของบริษัท Athena Information Service, Inc Oviedo มลรัฐ Florida ประเทศสหรัฐอเมริกา โดยรุ่นที่ได้จัดซื้อมาเพื่อประกอบการศึกษาค้นคว้าอิสระนี้คือรุ่นที่ 1.8.1.2 โดยลิขสิทธ์ที่ได้เพื่อการศึกษาเท่านั้น ไม่สามารถเอาโปรแกรมที่ต้องใช้ ActiveX Control รุ่นนี้ไปในเชิงการค้าได้

2.5 การจัดทำระบบเปลือกระบบผู้เชี่ยวชาญคลิปส์

การพัฒนาต้นแบบระบบผู้เชี่ยวชาญเพื่อการช่วยในการให้คำปรึกษาแนวทางการรักษาภาวะกระดูกส่วนแขนและขาหัก ที่มีแผลเปิดสำหรับแพทย์เวชปฏิบัติทั่วไป รพ.สมเด็จพระเจ้าตากสินมหาราช จ.ตาก นี้ได้เลือกใช้เปลือกระบบผู้เชี่ยวชาญคลิปส์แบบ ActiveX control มาช่วยในการพัฒนา โดยมีการเชื่อมต่อกับโปรแกรมที่พัฒนาโดยใช้ Visual Basic เนื่องจากการใช้คำสั่งของ CLIPS ActiveX Control นั้นเข้ากันได้ 100 % กับเปลือกระบบผู้เชี่ยวชาญคลิปส์บนระบบปฏิบัติการแบบดอสหรือวินโดวส์ ดังนั้นจึงจะขอกล่าวถึงรายละเอียดการทำงานของเปลือกระบบผู้เชี่ยวชาญคลิปส์พอสังเขปดังนี้

เปลือกระบบผู้เชี่ยวชาญคลิปส์มีทั้งแบบที่ทำงานบนระบบปฏิบัติการแบบดอสและแบบที่ทำงานระบบปฏิบัติการแบบวินโดวส์ การสั่งให้เปลือกระบบผู้เชี่ยวชาญคลิปส์ทำงานสามารถทำได้ทั้ง 2 รูปแบบคือ

แบบที่ 1 การป้อนคำสั่งให้ คลิปส์ ทำงานทีละคำสั่งจากท็อปเลเวล (top level) โดย

- เริ่มต้นการทำงาน คลิปส์ โดยการเรียกใช้คำสั่งให้เปลือกระบบผู้เชี่ยวชาญ คลิปส์ ทำงาน

  A:\> clipsdos กด Enter

- คลิปส์จะให้ command prompt รอรับคำสั่งที่ผู้ใช้งาน ป้อนคำสั่งที่ต้องการลงไป

  CLIPS>

- เมื่อต้องการออกจาก คลิปส์ใช้คำสั่ง exit

  CLIPS> (exit)

แบบที่ 2 การบรรจุชุดคำสั่งเก็บไว้เป็นไฟล์นามสกุล clp หรือแบทไฟล์ นามสกุล bat สามารถเรียกใช้ได้โดยคำสั่ง load ตัวอย่างเช่น

CLIPS> (load “a:\txmenu.clp”)

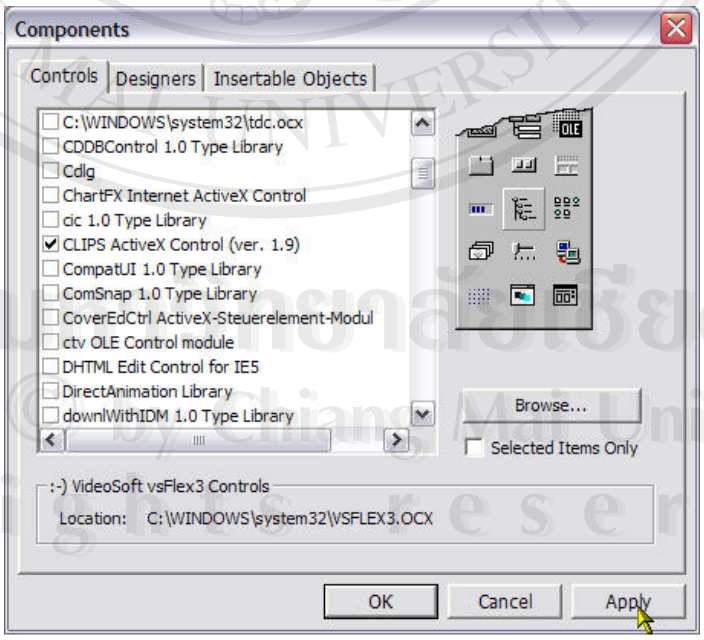
การแทนความรู้ใน คลิปส์ ทำได้ 3 วิธีคือ

- การเขียนแบบกฎ (Rule – Based) โดยการเขียนแบบกฎนี้คลิปส์จะสนับสนุนเฉพาะ การอนุมานแบบเดินหน้า (Forward chaining)
- การเขียนแบบโปรแกรมเชิงวัตถุ (Objected Oriented)
- การเขียนแบบเป็นลำดับขั้นตอน (Procedural)

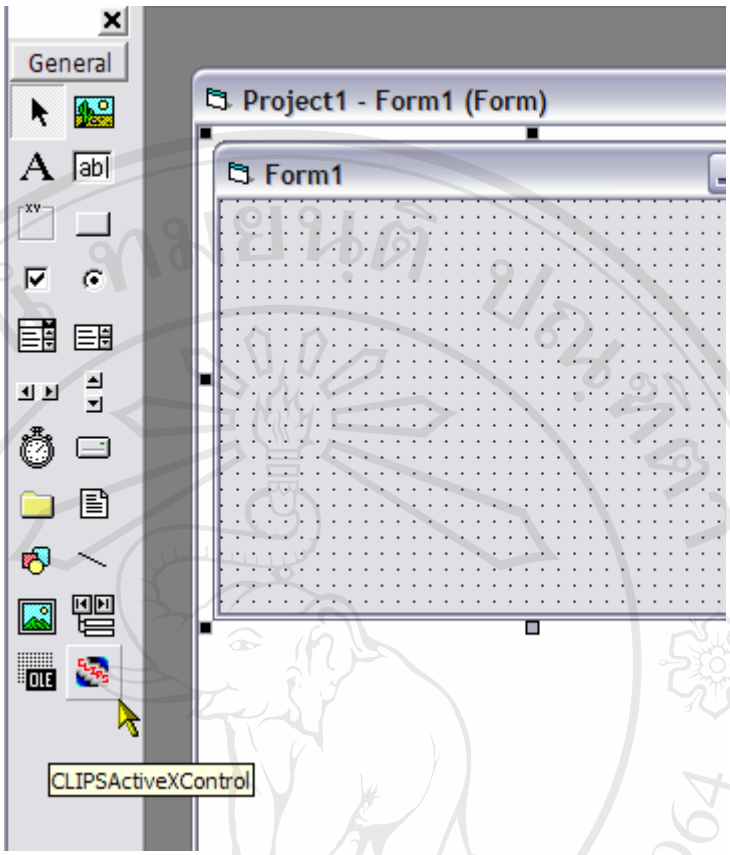
ในการศึกษานี้ผู้พัฒนาเลือกใช้การแทนความรู้โดยวิธีเขียนเป็นกฎ เก็บไว้ในไฟล์ชื่อ OprnRx.clp โดยกฎทั้งหมดจะครอบคลุมการให้แนวทางการรักษาโดยอนุมานจากตำแหน่งที่ได้รับบาดเจ็บกับระดับความรุนแรงของภาวะกระดูกหักที่มีแผลเปิด ร่วมกับการมีภาวะพิเศษที่ทำให้การวินิจฉัยหรือการรักษาเปลี่ยนแปลงไป

CLIPS ActiveX control ที่ทำงานอยู่ในวิชวลเบสิกนั้นมีการเรียกการใช้งานที่แตกต่างออกไปจาก คลิปส์บนดอสหรือวินโดวส์เล็กน้อยคือ ผู้พัฒนาต้องมีความรู้ในการเรียกใช้ ActiveX ดังนี้

ขั้นตอนแรก เริ่มต้นหลังจากเปิดโปรแกรม Visual Basic ขึ้นมาทำงานแล้ว หลังจากเลือกชนิดของ Project ที่ต้องการแล้ว ผู้พัฒนาต้องมีการเพิ่ม component ของ CLIPS โดยเปิดเมนู Project แล้วเลือก CLIPS component ดังรูปที่ 2.11 หลังจากนั้นจะพบ CLIPS ActiveX Control ที่ Toolbox

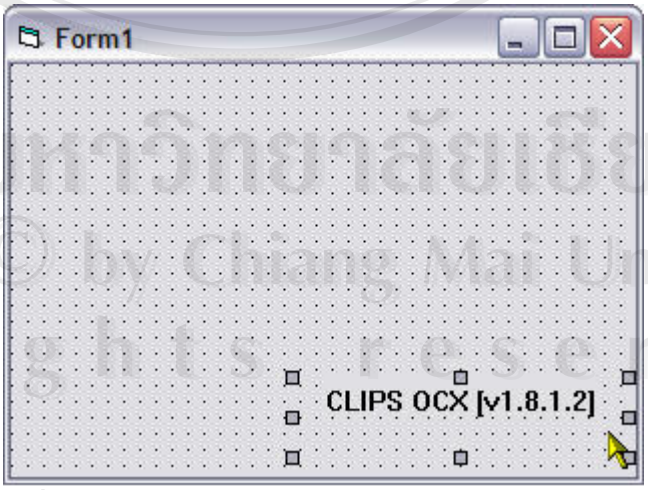


รูปที่ 2.11 การเลือก CLIPS ActiveX control เข้ามาใน Project



รูปที่ 2.12 CLIPS ActiveX Control ที่ Toolbox

ขั้นตอนที่ 2 เลือก CLIPS ActiveX control ที่ Toolbox ดังรูปที่ 2.12 แล้ว คอนโทรลลงในฟอร์มที่ต้องการจะให้เปลือกระบบผู้เชี่ยวชาญทำงาน ดังรูปที่ 2.13



รูปที่ 2.13 CLIPS ActiveX control พร้อมทำงานบน Form1

จากนั้นผู้พัฒนาสามารถทำการเรียกใช้คอนโทรลได้ตามไวยากรณ์ ของ Visual Basic ทุกประการโดยคำสั้ง CLIPS ActiveX จะเข้ากันได้กับ CLIPS ใน Platform อื่นทุกประการ เช่น

ถ้าต้องการให้ CLIPS ActiveX ทำการ Load file ชื่อ Test.clp ก็ทำการเพิ่มคำสั่ง CLIPS. Load ("Test.clp") ลงใน form ที่ได้เพิ่ม CLIPS ActiveX ไว้ตั้งแต่แรก ซึ่งเหมือนกับใน DOS ที่ต้องพิมพ์คำสั่ง (load "a:\txmenu.clp") ทุกประการ ส่วนการเขียนชุดคำสั่ง clp ผู้พัฒนาสามารถเขียนเหมือนเดิม แต่ต้องเรียนรู้คำสั่งขั้นสูงเพิ่มเติม ในการส่งข้อมูลกลับมาให้ Visual Basic

โดยการพัฒนาสามารถใช้วิธีการแทนค่าความรู้วิธีใดวิธีหนึ่งเพียงอย่างเดียว หรือหลายวิธีผสมกันก็ได้ แต่ในที่นี้จะขออธิบายรายละเอียดของการเขียนฐานความรู้แบบกฎเท่านั้น การเขียนฐานความรู้แบบกฎมีองค์ประกอบพื้นฐาน 3 อย่างคือ

1. รายการข้อเท็จจริง (Fact list) ประกอบด้วยข้อมูลที่เก็บไว้ในหน่วยความทรงจำ ซึ่งจะถูกนำไปใช้โดยกลไกการอนุมาน (Inference engine) และมีคำสั่งที่ใช้บ่อยๆ เกี่ยวกับข้อเท็จจริงดังนี้ คือ
   - การกำหนดโครงสร้างของข้อเท็จจริงให้โปรแกรมรู้จักโดยคำสั่งนี้ สามารถกำหนดรายละเอียดของข้อเท็จจริง ดังนี้

     (deftemplate <relation-name> [<optional-comment>]
     <slot-definition>*)

     ตัวอย่างการกำหนดโครงสร้างของข้อเท็จจริง

     (deftemplate person "An example deftemplate"
     (slot name)
     (slot age)
     (slot eye-color)
     (slot hair-color)

   - การเพิ่มข้อเท็จจริงเข้าไปในระบบ (Adding) โดยใช้คำสั่ง assert ซึ่งมีโครงสร้างการใช้คำสั่งดังนี้

     (assert <fact>+)

     ตัวอย่างการเพิ่มข้อเท็จจริงเข้าไปในระบบ

     (assert (person (name "John Q. Public")
     (age 23)
     (eye-color blue)
     (hair-color black)))

สามารถเพิ่มข้อเท็จจริงจำนวนมากๆเข้าไปในระบบ โดยใช้คำสั่ง defects ซึ่งมีโครงสร้างการใช้คำสั่งดังนี้

(deffacts <deffacts-name> [<optional comment>]
<facts>")

ตัวอย่างการเพิ่มข้อเท็จจริงจำนวนมากๆเข้าไปในระบบ

(deffact people "Some people we know"
(person (name "John Q. Public") (age 23)
(eye-color blue) (hair-color black))
(person (name "Jack S. Public") (age 24)
(eye-color blue) (hair-color black))
(person (name "Jane Q. Public") (age 36)
(eye-color green) (hair-color red))

- การลบข้อเท็จจริงออกจากระบบ (Removing) โดยใช้คำสั่ง retract ซึ่งมีโครงสร้างการใช้คำสั่งดังนี้

(retract <fact-index>+)

ตัวอย่างการลบข้อเท็จจริงเข้าไปในระบบ

(retract 0)

- การแก้ไขข้อเท็จจริง (Modifying) โดยใช้คำสั่ง modify ซึ่งมีโครงสร้างการใช้คำสั่งดังนี้

(modify <fact-index> <slot-modifier>+)

ตัวอย่างการแก้ไขข้อเท็จจริง โดยต้องการแก้อายุของ John Q. Public จาก 23 เป็น 24 ปี

(modify 0 (age 24))

- การแก้ไขข้อเท็จจริง (Modifying) โดยใช้คำสั่ง modify ซึ่งมีโครงสร้างการใช้คำสั่งดังนี้

(modify <fact-index> <slot-modifier>+)

ตัวอย่างการแก้ไขข้อเท็จจริง โดยต้องการแก้อายุของ John Q. Public จาก 23 เป็น 24 ปี

(modify 0 (age 24))

เมื่อใส่ข้อเท็จจริงเข้าไปในคลิปส์ข้อเท็จจริงแต่ละอันจะมีหมายเลขกำกับเพื่อใช้อ้างอิงในโปรแกรมเรียกว่าดัชนีข้อเท็จจริง (fact index) หรืออาจอ้างอิงจากที่อยู่ของข้อเท็จจริง (fact address) หรืออ้างจากชื่อของข้อเท็จจริงที่เรียกว่ารีเลชั่นเนม (relation name) ก็ได้ กลุ่มข้อเท็จจริงเหล่านี้มีทั้งเป็นแบบมีชื่อหรือไม่มีชื่อ แบบมีสล๊อต (slot) เดียวหรือมีหลายสล๊อต และใน สล๊อตจะมีข้อมูลเพียงค่าเดียว (single-field slot) หลายค่า (multifield slot) หรือไม่มีค่าอะไรเลยก็ได้ โดยชนิดของข้อมูลในสล๊อต เป็นได้ทั้ง float, integer, symbol, string, external, address, instance name และ instance address การป้อนข้อเท็จจริงให้กับคลิปส์สามารถใช้วิธีป้อนเข้าโดยตรงโดยคำสั่ง assert หรือ deffact ที่ท็อปเลเวล (top level) หรือจะใช้วิธีเรียกไฟล์ (file) จากภายนอกเข้ามาซึ่งมีนามสกุล bat หรือ clp โดยใช้คำสั่ง load ในกรณีที่ต้องการใช้ข้อเท็จจริงที่เหมือนกันทุกๆครั้งที่สั่งให้โปรแกรมทำงาน

2. ฐานความรู้ (knowledge base) ประกอบด้วยกฎต่างๆ กฎเหล่านี้สามารถป้อนเข้าไปในคลิปส์ โดยใช้คำสั่ง defrule ซึ่งมีโครงสร้างดังนี้

```
(defrule <rule-name> [<comment>]
<pattern>* ; Left-Hand Side (LHS) of the rule
=>
<action>*) ; Right-Hand Side (RHS) of the rule
```

ตัวอย่างการใช้คำสั่ง defrule

```
(defrule fire-emergency “An example rule”
(emergency (type fire))
=>
(assert (response
(action activate-spinkler-system))))
```


โดยถ้ารูปแบบ (Pattern) ทั้งหมดสอดคล้องกับกลุ่มข้อเท็จจริง กฎนั้นจะอยู่ในสภาพถูกกระตุ้น (activated) คือส่วนที่เป็นการกระทำ (action) จะถูกนำไปปฏิบัติ (fires) คลิปส์สามารถตัวแปรในการเขียนกฎได้ ความสามารถในการใช้ตัวแปรนี้ทำให้ผู้ใช้งานสามารถเขียนระบบผู้เชี่ยวชาญให้ทำงานได้ตามต้องการได้ง่าย โดยใน คลิปส์ ตัวแปรจะขึ้นต้นด้วยเครื่องหมายคำถาม (?) วิธีการใส่ค่าให้ตัวแปรทำโดยใช้ตรรกะ (Logic) คือ ถ้ามีข้อเท็จจริงที่ทำให้รูปแบบใน

กฎเป็นจริงได้ ตัวแปรจะมีค่าเท่ากับค่าที่อยู่ใน ข้อเท็จจริง เรียกว่า การบอนด์(bound) นอกจากนี้ยังสามารถกำหนดค่าที่เป็นที่อยู่ของข้อเท็จจริงให้กับตัวแปรได้ โดยใช้เครื่องหมาย pattern binding operator (<-)

3. กลไกการอนุมาน (Inference engine) ใช้ควบคุมการทำงานทั้งหมด เครื่องอนุมานจะเป็นตัวตัดสินว่ากฎข้อใดจะถูกนำมาใช้และปฏิบัติโดยใช้วิธีการค้นหากฎที่มีความสอดคล้องกับข้อเท็จจริงมาปฏิบัติ ดังรายละเอียดต่อไปนี้

- ใช้คำสั่ง รัน (run) สั่งให้เปลือกระบบผู้เชี่ยวชาญคลิปส์ นำระบบผู้เชี่ยวชาญมาเริ่มทำงานแล้ว
- จากนั้นป้อน (input) กลุ่มข้อเท็จจริง เข้าระบบ

ถ้าข้อเท็จจริงที่เราป้อนให้ระบบตรงกับทุกรูปแบบในกฎ (Left hand side : LHS) กฎจะถูกกระตุ้นแล้วส่วนที่เป็นการกระทำของกฎ (Right hand side : RHS) จะถูกนำไปเก็บเรียงไว้อะเจนด้า (agenda) แบบสแตก (stack) คือ LIFO (last in first out) นั่นคือ การกระทำใดถูกนำเข้าไปเก็บทีหลังก็จะถูกนำมาปฏิบัติก่อนแต่หากกฎเหล่านั้นมีการกำหนดความสำคัญที่เรียกว่า เซเลียนท (salience) การกระทำในอะเจนด้า (agenda) ก็จะถูกนำมาปฏิบัติตามค่าของเซเลียนท (salience) จากมากไปหาน้อยจนหมดหรือจนกว่าจะได้รับคำสั่งหยุดการทำงาน แต่เนื่องจากลักษณะการทำงานของ คลิปส์ จะต่างจากโปรแกรมแบบลำดับขั้นตอน(Procedural Language) ตัวอย่างเช่น ทุกครั้งที่สั่งให้โปรแกรมโปรแกรมแบบลำดับขั้นตอนก็จะเริ่มทำงานใหม่ทุกครั้ง แต่ในคลิปส์หากมีการสั่งให้โปรแกรมทำงานอีก จะไม่เกิดการกระทำใดๆทั้งสิ้น แม้ว่าเงื่อนไขจะเป็นจริงก็ตาม ทั้งนี้เพราะว่าไม่มีกฎที่อยู่ในสถานะถูกกระตุ้นอยู่ในอะเจนด้า(agenda) เลยเพราะในคลิปส์กฎจะอยู่ในสภาวะถูกกระตุ้นก็ต่อเมื่อ

- รูปแบบในกฎนั้นเป็นรูปแบบใหม่มีการเปลี่ยนแปลงหรือเพิ่งเกิดขึ้น
- รูปแบบนั้นเคยเกิดขึ้นมาก่อนแต่ได้ถูกลบออกและเพิ่มขึ้นมาใหม่

ดังนั้นในการเขียนระบบผู้เชี่ยวชาญสามารถใช้หลักการที่สำคัญเหล่านี้ช่วยในการควบคุมการทำงานของโปรแกรมได้โดยการควบคุมการเข้าคู่กัน(Match) ระหว่างกลุ่มข้อเท็จจริงกับรูปแบบของกฎ

2.6 Review Literatures

ระบบผู้เชี่ยวชาญทางด้านการแพทย์ ได้มีพัฒนามาเป็นระยะเวลานานพอสมควร ดังเห็นได้จากในการสืบค้นเอกสารทางวิชาการย้อนหลัง พบว่าได้มีการพัฒนาเพื่อแก้ไขปัญหาต่างๆ ในหลายๆสาขาทางการแพทย์ ได้แก่ ระบบผู้เชี่ยวชาญเพื่อเพิ่มประสิทธิภาพการอ่านค่าผลตรวจทางห้องปฏิบัติการ (McNeely MD, 2002) ที่สามารถลดค่าใช้จ่ายในการส่งตรวจทางห้องทดลองเพิ่มเติมได้ถึง 20% ระบบผู้เชี่ยวชาญเพื่อการแปลผลคลื่นไฟฟ้าหัวใจ (KtoNus PY, 1996) ที่นำมาตรวจสอบหาลักษณะของคลื่นไฟฟ้าหัวใจที่ผิดปกติซึ่งไม่ได้ผลการศึกษาไว้ ระบบปัญญาประดิษฐ์เพื่อการวิเคราะห์ประสิทธิภาพทางการกีฬา (Lapham AC, Bartlett RM, 1995) ที่นำระบบคอมพิวเตอร์ผู้เชี่ยวชาญมาช่วยในการวิเคราะห์ในการทำ simulation ทางชีวกลศาสตร์ ได้สะดวกและรวดเร็วขึ้นกว่าเดิม

Chen Z และคณะ (2002) ได้กล่าวว่าระบบผู้เชี่ยวชาญทางด้านการวินิจฉัยโรคและหาแนวทางการรักษานั้น พบว่าได้มีการวิจัยและพัฒนาระบบต้นแบบระบบมาหลายด้าน มีความลำบากในการพัฒนาเนื่องจากพยาธิสภาพที่เกิดขึ้นกับผู้ป่วยมักจะประกอบกันด้วยปัจจัยหลายๆ อย่าง เช่น ผู้ป่วยคนเดียวมักมีหลายๆพยาธิสภาพร่วมกัน ทำให้เกิดความลำบากในการวินิจฉัยเพื่อหาแนวทางการรักษา โดยทั่วไปแล้วระบบผู้เชี่ยวชาญที่ประสบความสำเร็จในการนำไปใช้ มักจะเป็นการวินิจฉัยเพื่อหาคำตอบหรือแนวทางการรักษาภาวะใด ภาวะหนึ่งเท่านั้น ตามรายงานของการพัฒนาระบบผู้เชี่ยวชาญ CADIAG-1 (Adlassing KP et. Al, 1995)เพื่อการวินิจฉัยภาวะ โรครูมาติก (Rheumatic disease) กับ โรคแพนครีเอติก (Pancreatic disease) โดยมีความแม่นยำของการวินิจฉัยถึง 93.7%

การพัฒนาระบบผู้เชี่ยวชาญเพื่อการวินิจฉัยและการรักษาภาวะกระดูกหัก แบบมีแผลเปิดนั้น ผู้ศึกษาได้สืบค้นจากเอกสารที่ตีพิมพ์ในฐานข้อมูล Medline แล้วไม่พบว่ามีการศึกษาในด้านนี้หรือ การพัฒนาระบบแบบนี้เลย แต่อาจจะเป็นไปได้ว่าอาจจะมีการศึกษาแต่ยังไม่ได้มีการตีพิมพ์ลงในหนังสือหรือวรสารที่มีการอ้างอิงใน Index Medicus ก็เป็นได้