บทที่ 3

ทฤษฎี และหลักการของระบบผู้เชี่ยวชาญ

3.1 คำจำกัดความของระบบผู้เชี่ยวชาญ

ระบบผู้เชี่ยวชาญ คือระบบของโปรแกรมคอมพิวเตอร์ที่เก็บรวบรวมฐานความรู้ที่ได้จากผู้เชี่ยวชาญมาใส่ให้เป็นฐานความรู้ให้กับระบบและมีกระบวนการอนุมานเพื่อที่จะนำไปสู่เป้าหมายหรือเป้าประสงค์คำตอบของปัญหานั้นๆ โดยกรรมวิธีในการนำไปสู่คำตอบเราเปรียบให้เป็นเสมือนกับการทำงานของระบบผู้เชี่ยวชาญนั้นเอง

ดังนั้นในการที่จะสร้างระบบผู้เชี่ยวชาญขึ้นมาใช้สำหรับในการแก้ปัญหาที่สลับซับซ้อนนั้นสิ่งที่ผู้สร้างจะต้องเข้าใจคือกรรมวิธีการทำงานของระบบผู้เชี่ยวชาญเป็นอย่างดี ซึ่งนั้นก็คือจะเป็นผลดีในการที่จะเก็บรวบรวมเอาความรู้ที่จะนำมาใช้ในกระบวนการอนุมานได้อย่างถูกต้องและสมบูรณ์ที่สุด

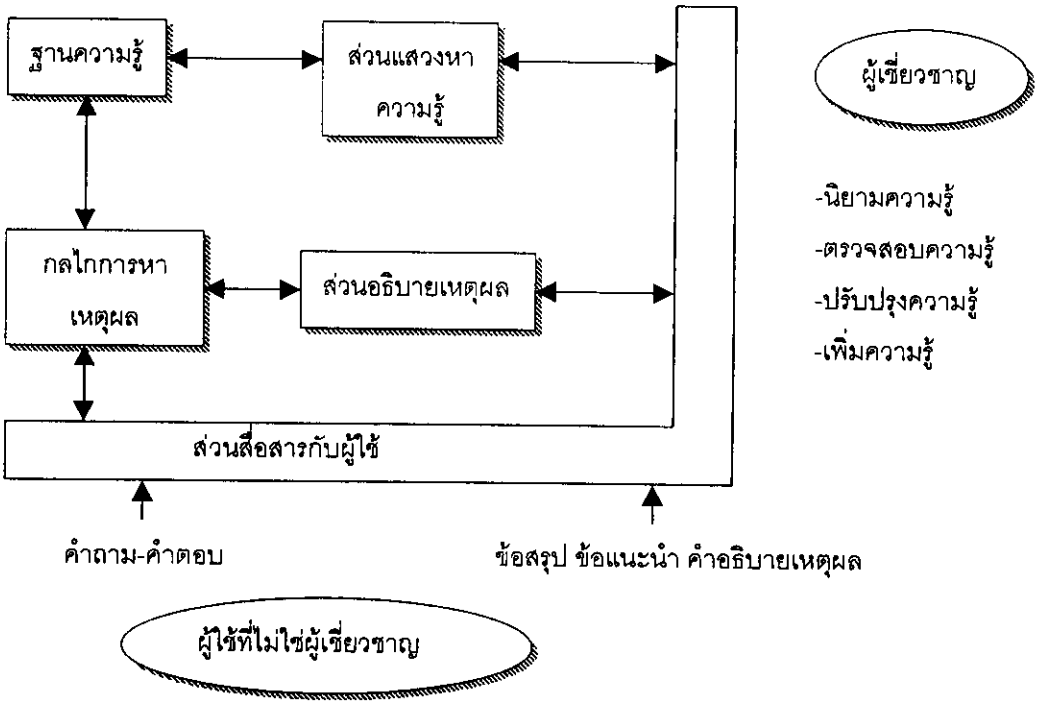
ดังนั้นเราพอที่จะสรุปได้ว่าระบบผู้เชี่ยวชาญเป็นโปรแกรมคอมพิวเตอร์ที่ใช้แก้ปัญหาโดยอาศัยความรู้ในสาขาใด สาขาหนึ่ง ที่รวบรวมความรู้จากผู้เชี่ยวชาญในสาขานั้นๆ และเก็บไว้เป็นฐานความรู้ในคอมพิวเตอร์ โปรแกรมดังกล่าวต้องเป็นโปรแกรมเพื่อแก้ปัญหาโดยเฉพาะ ความสามารถในการแก้ปัญหาโปรแกรมนี้จะเทียบเท่าความสามารถของผู้มีความชำนาญระดับผู้เชี่ยวชาญนั้นๆ

3.2 องค์ประกอบของระบบผู้เชี่ยวชาญ

ระบบผู้เชี่ยวชาญโดยทั่วไปประกอบด้วยส่วนประกอบพื้นฐาน 5 ส่วน ดังภาพประกอบที่ 3-1 ส่วนที่เป็นหัวใจที่ขาดเสียมิได้ คือ ฐานความรู้ และเครื่องอนุมาน รายละเอียดโดยย่อของแต่ละส่วนสามารถอธิบายได้ดังนี้

1. ฐานความรู้ (Knowledge Base) เป็นส่วนที่เก็บความรู้ในสาขาต่างๆ ที่ต้องการแก้ปัญหาไว้ในรูปแบบที่คอมพิวเตอร์สามารถเข้าใจและสามารถประมวลผลได้ และจะต้องมีการดงความรู้นี้ให้มีความทันสมัยอยู่เสมอซึ่งโดยปกติแล้วความรู้ที่รวบรวมมาจะมีจำนวนมาก ซึ่งปัญหาที่สำคัญอีกประการหนึ่งในการสร้างฐานความรู้ขึ้นมา คือทำอย่างไรจึงจะแทนความรู้จำนวนมากเหล่านั้นเก็บไว้ในคอมพิวเตอร์โดยไม่สิ้นเปลืองเนื้อที่มากนักในขณะเดียวกันก็อยู่ในรูปที่สามารถวิเคราะห์หาเหตุผลจากความรู้นั้นออกมาได้

สถาปัตยกรรมของระบบผู้เชี่ยวชาญจะสามารถแสดงดังในภาพประกอบที่ 3-1



ภาพประกอบ 3-1 องค์ประกอบของระบบผู้เชี่ยวชาญ

2. กลไกการหาเหตุผล (Inference Engine) เป็นส่วนของโปรแกรมที่ใช้ในการคิดหาเหตุผลโดยอาศัยความรู้จากฐานความรู้ข้างต้น เพื่อให้ได้วิธีแก้ปัญหาออกมา วิธีการหาเหตุผลปกติจะสัมพันธ์กับลักษณะการแทนความรู้ที่เลือกใช้ในการสร้างฐานข้อมูล

3. ส่วนการสื่อสารกับผู้ใช้ (User Interface Module) เป็นส่วนของโปรแกรมที่ใช้สื่อสารระหว่างผู้ใช้กับระบบ เช่น อาจเป็นการตั้งคำถามมายังผู้ใช้เพื่อให้ผู้ใช้ป้อนข้อมูลที่จำเป็นเพื่อนำไปใช้ในการแก้ปัญหาต่อไป หรือผู้ใช้อาจตั้งคำถามให้ระบบอธิบายว่าระบบกำลังทำอะไรอยู่หรือระบบช่วยหาคำตอบให้ ถ้าการสื่อสารระหว่างผู้ใช้กับระบบผู้เชี่ยวชาญทำได้ง่ายระบบผู้เชี่ยวชาญนั้นก็จะมีประโยชน์ต่อผู้ใช้มากขึ้น

4. ส่วนแสวงหาความรู้ (Knowledge Acquisition Module) ในการนำความรู้จากผู้เชี่ยวชาญมาสร้างความรู้จำเป็นต้องมีการรวบรวมความรู้ของผู้เชี่ยวชาญแทนความรู้นั้นให้อยู่ในรูปที่คอมพิวเตอร์สามารถประมวลผล คิดค้นหาเหตุผลหรือคำตอบจากความรู้นั้นได้ และรวบรวมให้เป็นฐานความรู้ขึ้นมา กระบวนการเหล่านี้เรียกว่ากระบวนการแสวงหาความรู้ (Knowledge Acquisition) ดังนั้น

โปรแกรมส่วนนี้ก็คือส่วนที่ช่วยผู้ชำนาญการในการนิยามฐานความรู้ขึ้นมา รวมทั้งช่วยในการปรับปรุงฐานความรู้นั้นให้ทันสมัยอยู่เสมอ

5. ส่วนอธิบายเหตุผล (Explanation Module) คือ เมื่อผู้ใช้ปรึกษาระบบชำนาญการและคำตอบออกมาเพียงคำตอบเดียว ผู้ใช้อาจไม่มั่นใจว่าคำตอบที่ได้นั้นจะนำไปสู่การแก้ไขปัญหาได้จริง การอธิบายเหตุผลให้ผู้ใช้ทราบอาจทำให้ผู้ใช้นำคำตอบที่ได้ไปใช้งานอย่างมั่นใจโปรแกรมส่วนนี้จึงเป็นส่วนที่จำเป็นที่ใช้อธิบายเหตุผลให้ผู้ใช้ได้ทราบว่าข้อสรุปของระบบที่เป็นคำตอบออกมาได้อย่างไร

ในระบบผู้เชี่ยวชาญบางระบบจะไม่มีส่วนประกอบครบทั้ง 5 ส่วนดังกล่าวข้างต้นก็ได้ แต่ที่ขาดไม่ได้แน่ๆ คือ ส่วนของฐานความรู้และเครื่องอนุมาน ซึ่งจะกล่าวในรายละเอียดต่อไป

3.3 ฐานความรู้

ฐานความรู้เป็นส่วนที่เก็บความรู้ในสาขาที่ต้องการแก้ไขไว้ในรูปแบบที่คอมพิวเตอร์สามารถประมวลผลได้ และมีการบำรุงรักษาฐานความรู้ให้ทันสมัยอยู่เสมอ โดยปกติแล้วความรู้ที่รวบรวมมามักจะมีมากมาย ปัญหาที่สำคัญประการหนึ่งในการสร้างฐานความขึ้นมารู้คือ ทำอย่างไรจึงจะแทนความรู้จำนวนมากเหล่านั้น เพื่อเก็บไว้ในคอมพิวเตอร์ได้โดยไม่เปลืองเนื้อที่มากนัก ในขณะเดียวกันก็อยู่ในรูปที่วิเคราะห์หาเหตุผลจากความรู้นั้นออกมาได้ โดยรูปแบบของกฎจะมีลักษณะเป็นลิสต์ซึ่งประกอบด้วยชื่อของกฎ เงื่อนไข และผลสรุป

ในระบบผู้เชี่ยวชาญจำเป็นต้องมีฐานความรู้ที่จะต้องนำมาสำหรับการอนุมานในระบบ ซึ่งฐานความรู้เป็นส่วนที่ใช้เก็บความรู้ต่างๆ ทุกประเภทที่เกี่ยวข้องกับปัญหาไม่ว่าจะได้มาจากตำราหรือมาจากประสบการณ์ของผู้ชำนาญการเอง ซึ่งมีการแบ่งฐานความรู้ออกเป็น 2 ชนิดใหญ่ คือ ฐานความจริง และฐานกฎ ซึ่งสามารถที่จะอธิบายได้ดังต่อไปนี้ คือ

3.3.1 ความรู้ที่อยู่ในลักษณะของความจริง (Fact) หรือ สิ่งที่ปฏิเสธไม่ได้ คือความรู้ที่ได้มาจากตำราหรือได้จากประสบการณ์ของผู้ชำนาญการ

3.3.2 ความรู้ที่อยู่ในลักษณะของกฎ (Rule) คือการแสดงความรู้ในรูปของกลุ่มความจริง ส่วนที่บอกถึงขั้นตอน หรือการปฏิบัติในการแก้ไขปัญหา

3.4 กลไกการหาเหตุผล

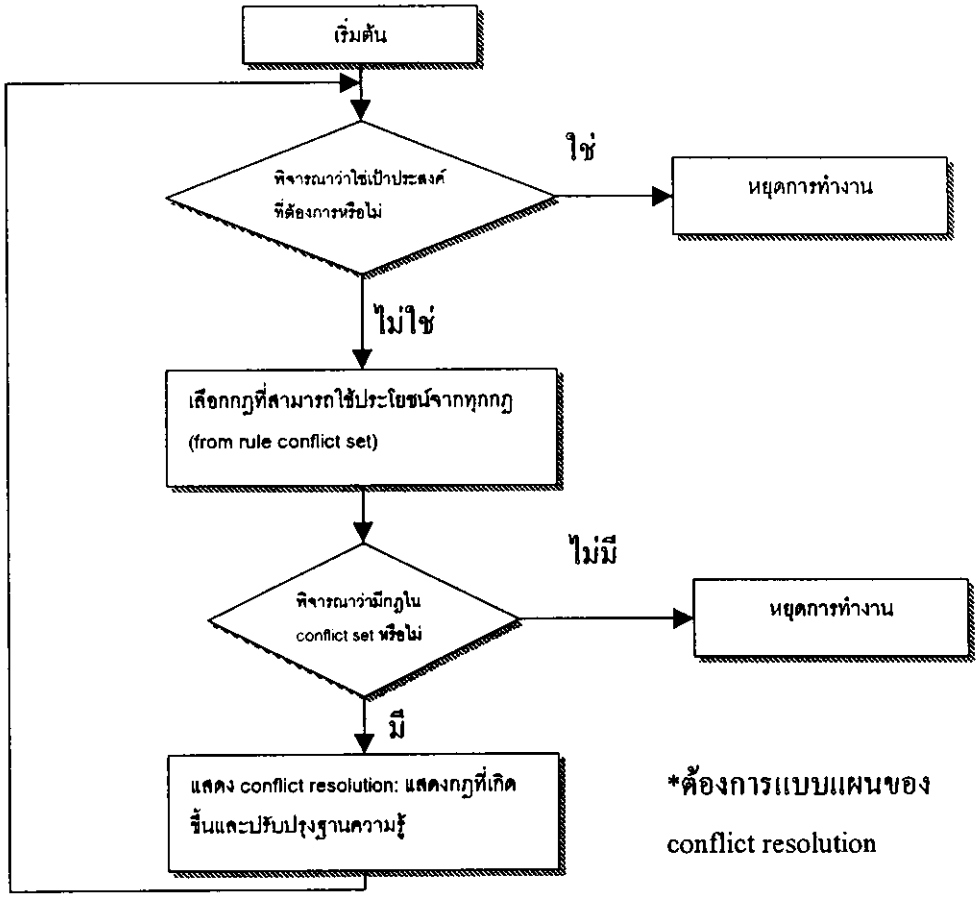
เครื่องอนุมานเป็นส่วนที่สำคัญที่สุดเมื่อเราทำการออกแบบฐานความรู้เสร็จสิ้นเรียบร้อยแล้ว ส่วนนี้เป็นส่วนหนึ่งของระบบเปลือกผู้เชี่ยวชาญ ซึ่งเครื่องอนุมานจะเป็นตัวกำหนดความสามารถของระบบ กำหนดความเร็ว และความถูกต้องของการแก้ปัญหา โดยหน้าที่หลักของเครื่องอนุมานคือ เป็นตัวดึงความรู้จากฐานความรู้มาใช้ในการให้คำปรึกษาหรือแนะนำต่อผู้ใช้ โดยการดำเนินการนำเอาเงื่อนไขของแต่ละกฎมาถามต่อผู้ใช้ เพื่อพิสูจน์ค่าความจริงของแต่ละเงื่อนไข ในกฎเพื่อดึงเอาข้อสรุปที่ได้มาใช้เป็นคำตอบของผู้ชำนาญการ พร้อมแสดงถึงความน่าจะเป็นของข้อสรุปนั้นๆ ด้วย

โดยการอนุมานที่ใช้ในระบบผู้เชี่ยวชาญนั้นจะมีอยู่ด้วยกันหลายวิธีขึ้นอยู่กับลักษณะของปัญหาและรูปแบบของการแทนความรู้ เช่น ในระบบผู้เชี่ยวชาญที่มีการทำงานเป็นลำดับขั้นตอนและใช้การแทนความรู้ในรูปของกฎ จะมีวิธีการอนุมานอยู่ด้วยกัน 3 วิธี คือ การอนุมานแบบเดินหน้า (Forward Chaining Inference) การอนุมานแบบถอยหลัง (Backward Chaining Inference) และการอนุมานแบบผสม (Mix Chaining Inference) ซึ่งเราสามารถอธิบายหลักการของการอนุมานแต่ละอย่างได้ดังต่อไปนี้ คือ

3.4.1 การอนุมานแบบเดินหน้า

การอนุมานจะเริ่มต้นจากข้อมูลความจริงที่มีอยู่ในฐานข้อมูล โดยจะถูกเปรียบเทียบกับเงื่อนไขของกฎ ซึ่งกฎที่เงื่อนไขตรงกับข้อเท็จจริงที่มีอยู่ก็จะถูกปฏิบัติการตามข้อสรุป ซึ่งกฎต่างๆ จะถูกอนุมานในลักษณะนี้ไปเรื่อยๆ จนกว่าจะได้คำตอบหรือบรรลุเป้าหมาย ซึ่งเป็นกระบวนการในการสร้างฐานความรู้ขึ้นมาใหม่

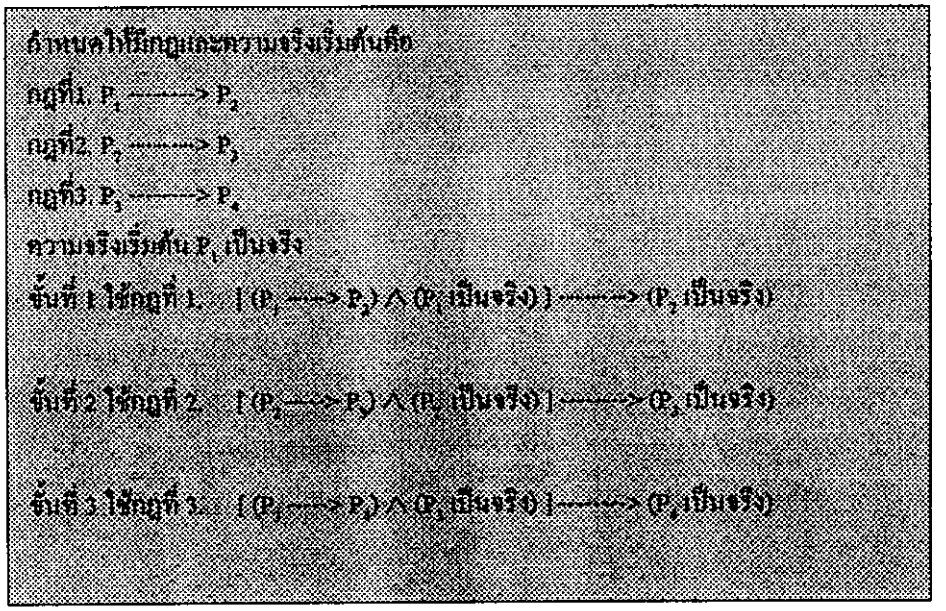
ข้อดีของการอนุมานแบบเดินหน้า คือ จะค้นหาทุกเป้าประสงค์ที่เป็นไปได้ โดยมีข้อเสียคือในบางครั้งเป้าประสงค์ที่ได้ไม่ใช่สิ่งที่ต้องการ ดังนั้นจึงเป็นการเสียเวลาอย่างมาก ต่อระบบที่ต้องค้นหาคำตอบของแต่ละปัญหา และเราสามารถแสดงการอนุมานแบบเดินหน้าได้ดังภาพประกอบ 3-2 ซึ่งแสดงการทำงานของการอนุมาน และภาพประกอบ 3-3 และ 3-4 เป็นตัวอย่างของการอนุมานแบบเดินหน้า กฎและความจริงที่กำหนดขึ้นจะเป็นส่วนประกอบของฐานความรู้เริ่มต้นและลำดับของการพิจารณาเหตุผลแบบเดินหน้า ตามลำดับ



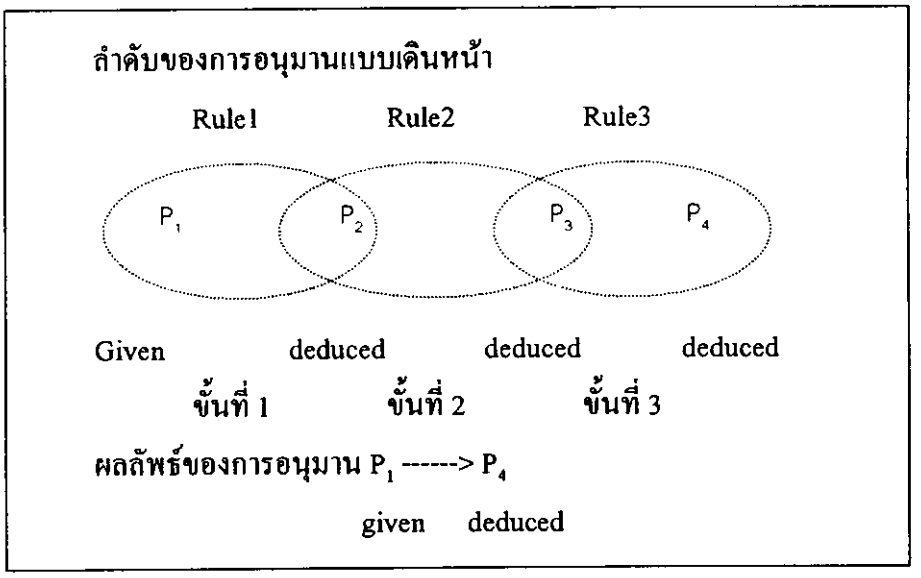
ภาพประกอบ 3-2 การทำงานของการอนุมานแบบเดินหน้า [Robert, 1990]

3.4.2 การอนุมานแบบถอยหลัง

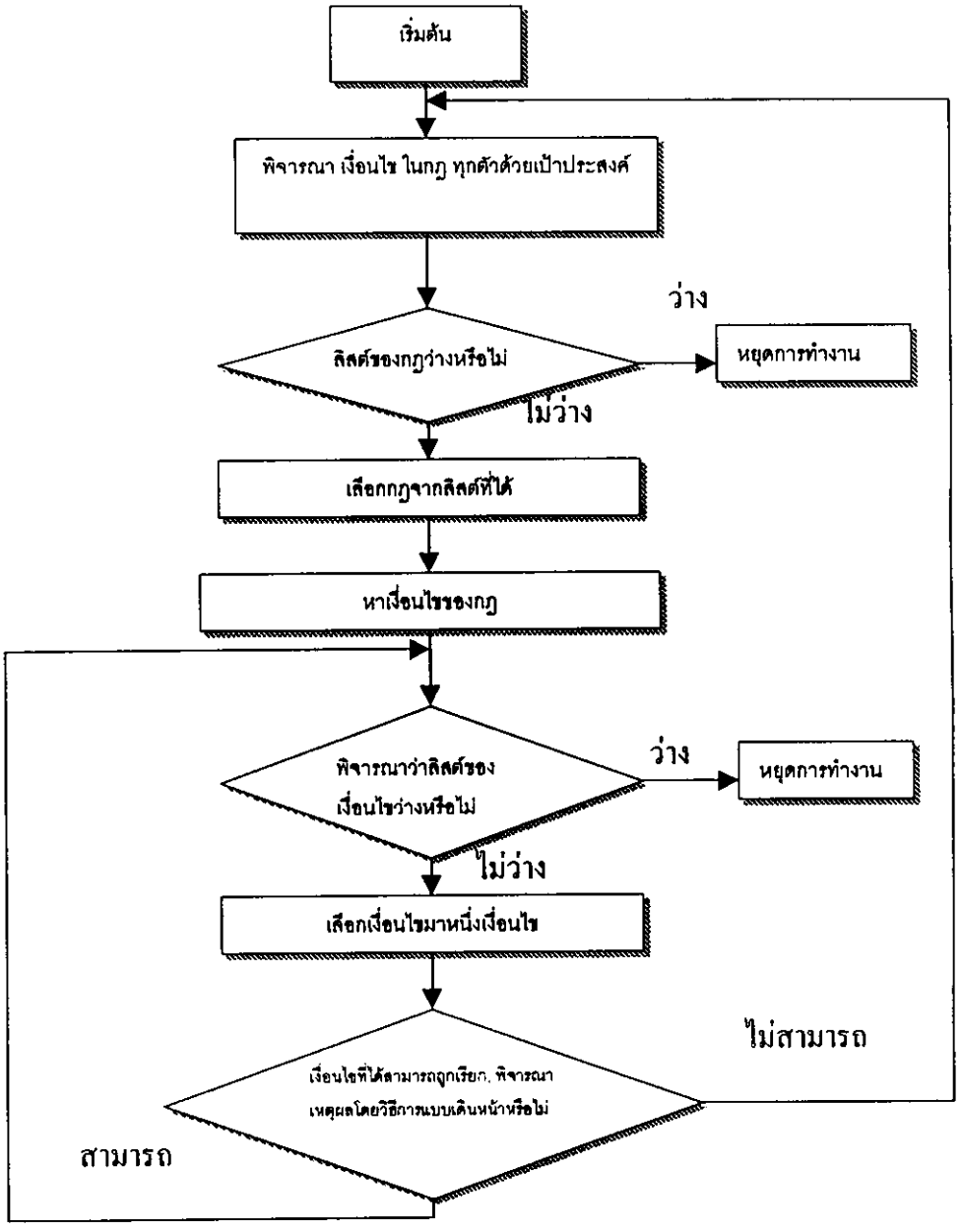
การอนุมานแบบถอยหลัง จะเริ่มต้นจากเป้าหมาย ซึ่งเป็นสมมติฐานที่สนใจไปสู่กฎที่จะนำให้เป้าหมายบรรลุผล ซึ่งกฎดังกล่าวจะต้องมีข้อสรุปเหมือนกับเป้าหมาย หลังจากนั้นเงื่อนไขของกฎนี้จะต้องถูกตรวจสอบว่าเป็นจริงตามข้อเท็จจริงที่มีอยู่ในฐานข้อมูลหรือไม่ ถ้าทุกเงื่อนไขของกฎนี้เป็นจริง เป้าหมายดังกล่าวก็บรรลุผลเป็นจริง แต่ถ้ามีบางเงื่อนไขหรือทุกเงื่อนไขของกฎนี้ไม่เป็นจริงตามข้อเท็จจริงในฐานข้อมูล เงื่อนไขที่ไม่เป็นจริงเหล่านี้ก็จะกลายเป็นเป้าหมายย่อยๆ เพื่อการอนุมานในขั้นต่อๆ ไป ซึ่งจะดำเนินไปในแบบลูกโซ่ จนกว่าจะพบกฎที่มีเงื่อนไขจากข้อเท็จจริงและทำให้เป้าหมายย่อยเหล่านั้นบรรลุผล ซึ่งก็เป็นการพิสูจน์สมมติฐานดังภาพประกอบ 3-5 และภาพประกอบ 3-6



ภาพประกอบ 3-3 ลำดับการพิจารณาเหตุผลของการอนุมานแบบเดินหน้า [Robert, 1990)

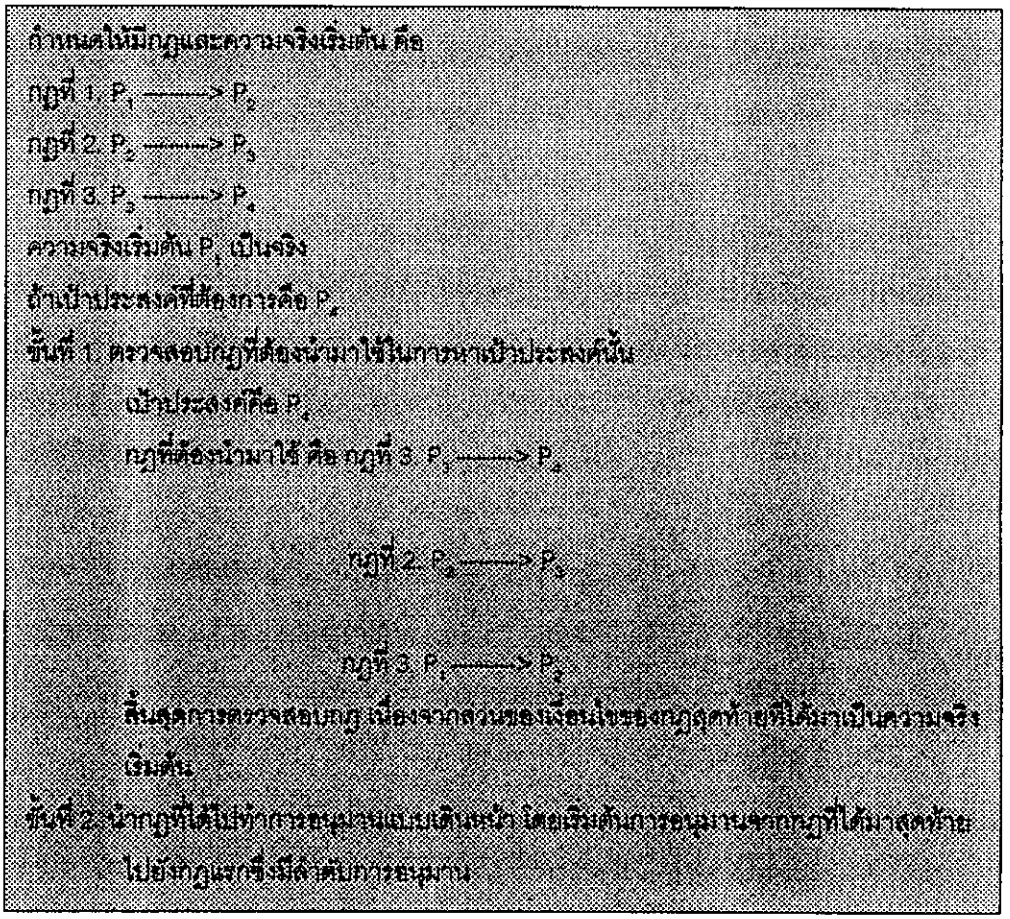


ภาพประกอบ 3-4 ลำดับการพิจารณาผลของการอนุมานแบบเดินหน้า [Robert, 1990)



ภาพประกอบ 3-5 การทำงานของการอนุมานแบบย้อนหลัง [Robert, 1990]

ข้อดีของการอนุมานแบบถอยหลัง คือ จะค้นหาเฉพาะเป้าประสงค์ที่ต้องการเท่านั้นจึงใช้เวลาน้อย แต่ก็มีข้อเสียคือ ในบางครั้งอาจจะหาเป้าหมายไม่ได้เลย เนื่องจากเงื่อนไขในการที่จะนำไปสู่เป้าประสงค์นั้นไม่ถูกต้อง



ภาพประกอบ 3-6 ตัวอย่างของอนุมานแบบย้อนหลัง

3.4.3 การอนุมานแบบผสม

ลักษณะของการอนุมานแบบผสม จะเป็นลักษณะของการอนุมานที่รวมกันระหว่างการอนุมานแบบเดินหน้าและถอยหลัง กล่าวคือ เริ่มต้นระบบจะรับเป้าประสงค์จากผู้ใช้เหมือนกับการอนุมานแบบถอยหลัง จากนั้นจะนำเป้าหมายที่รับนั้นไปทำการอนุมานแบบถอยหลัง ซึ่งถ้าเป้าประสงค์ดังกล่าวเป็นจริง ระบบก็จะนำเส้นทางของการอนุมานนั้นไปทำการอนุมานแบบเดินหน้าเพื่อให้ได้ความจริงใหม่ ตามเส้นทางของการอนุมานแบบถอยหลัง

3.5 วิธีการค้นหาคำตอบ

State Space ก็คือเซตของสถานะที่เป็นไปได้ของระบบในระหว่างกระบวนการแก้ปัญหา ดังนั้น การแก้ปัญหาที่สามารถแทนในรูปสเปซของสถานะได้ ก็คือการเคลื่อนที่จากสถานะหนึ่งไป

สู่สถานะหนึ่งโดยการใช้ตัวกระทำการที่ใช้กับสถานะเหล่านั้นได้จนกว่าจะถึงสถานะเป้าหมาย วิธีการแก้ปัญหาที่ได้ก็คือลำดับของตัวกระทำการที่นำมาใช้ต่อเนื่องกันจนนำไปสู่เป้าหมายนั้น วิธีที่ใช้แก้ปัญหาในลักษณะนี้จึงเป็นการค้นหาเส้นทางซึ่งนำไปสู่สถานะเป้าหมาย การค้นหาอย่างมีประสิทธิภาพนับว่าเป็นสิ่งสำคัญอย่างยิ่ง เพราะสถานะที่เป็นไปได้ทั้งหมดอาจมีจำนวนมาก (ใกล้อนันต์) ทำให้การค้นหากินเวลามาก

วิธีการค้นหา อาจนำไปใช้กับปัญหาใดก็ได้ที่แทนในรูปสเปซของสถานะได้ แต่ประสิทธิภาพของวิธีการค้นหาแบบต่างๆ จะขึ้นอยู่กับลักษณะของปัญหาของการใช้ความรู้เกี่ยวกับปัญหามาช่วยในการกำจัดขอบเขตของการค้นหา จุดอ่อนของวิธีแก้ปัญหาโดยการค้นหาคำตอบคือ ถ้าสเปซสถานะของปัญหามีขนาดใหญ่มาก อาจทำการค้นหาคำตอบไม่สำเร็จ จึงมักเรียกวิธีการค้นหาคำตอบว่าเป็นวิธีการแก้ปัญหาแบบไม่มั่นคง (Weak Method)

3.5.1 การแทนสเปซสถานะในรูปต้นไม้หรือกราฟ

เมื่อนิยามปัญหาในรูปสเปซสถานะแล้ว อาจแทนสเปซสถานะนั้นในรูปต้นไม้ (Tree) หรือ กราฟ (Graph) ก็ได้ สถานะต่างๆ จะแทนด้วยโหนด (Node) เมื่อใช้ตัวกระทำการทำให้เปลี่ยนสถานะหนึ่งไปสู่อีกสถานะหนึ่ง สถานะที่เปลี่ยนไปจะแทนด้วยโหนดซึ่งมีเส้นโยงกับสถานะเก่า

ตัวอย่าง 3.1 น้ำในถังสองถัง ความจุ 6 และ 8 แกนลอนตามลำดับ ทั้ง 2 ถัง ไม่มีมาตราบอกปริมาตรน้ำ ถ้าต้องการให้ถังที่จุ 8 แกนลอนมีน้ำบรรจุอยู่ 4 แกนลอน โดยตัวกระทำต่อไปนี้

ตัวกระทำการ

1	เติมน้ำในถัง 8 แกนลอน ให้เต็ม
2	เติมน้ำในถัง 6 แกนลอนให้เต็ม
3	เทน้ำในถัง 8 แกนลอนออกหมด
4	เทน้ำในถัง 6 แกนลอนออกหมด
5	เทน้ำจากถัง 8 แกนลอนใส่ถัง 6 แกนลอน
6	เทน้ำจากถัง 6 แกนลอนไปใส่ถัง 8 แกนลอน
7	เทน้ำจากถัง 6 แกนลอนไปใส่ถัง 8 แกนลอนจนเต็มถัง 8 แกนลอน
8	เทน้ำจากถัง 8 แกนลอนไปใส่ถัง 6 แกนลอนจนเต็มถัง 6 แกนลอน

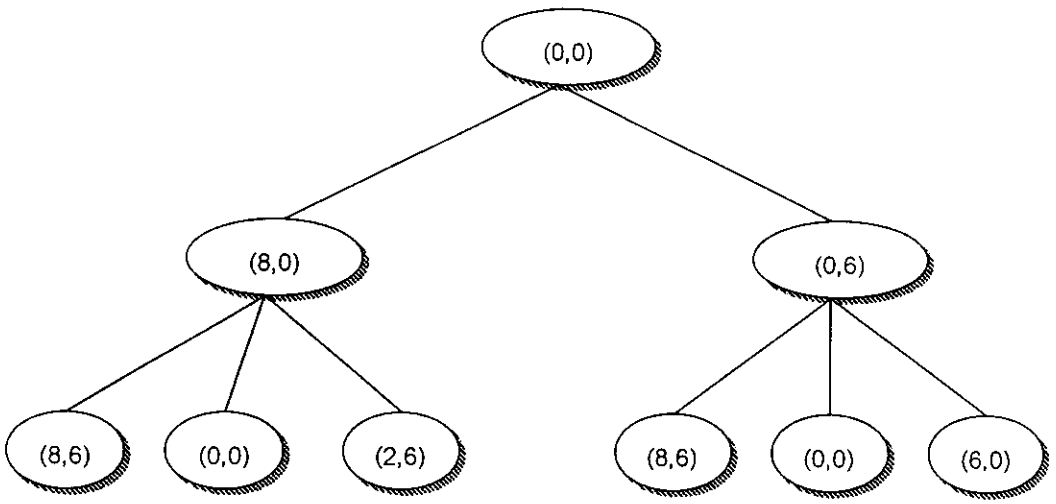
วิธีการแก้ปัญหาแบบที่ 1

วิธีการแก้ปัญหานี้ง่ายๆ แบบหนึ่ง คือ ก่อนจะเขียนโปรแกรมคอมพิวเตอร์ เราลองมาไล่ดูว่าตัวกระทำการใดบ้างจะทำให้ถัง 8 แกนลอนมีน้ำ 4 แกนลอน ตามต้องการ เราอาจพบว่าถ้าใช้ตัว

กระทำการ 2, 6, 2, 7, 3, 6 ตามลำดับ จะได้ผลตามต้องการ เสร็จแล้วก็เขียนลำดับของตัวกระทำการนี้ลงในโปรแกรมคอมพิวเตอร์รับได้ วิธีการนี้เรียกว่าเป็นการแก้ปัญหาแบบกำหนดวิธีแก้ไว้เรียบร้อยแล้ว โปรแกรมสามารถทำงานได้อย่างมีประสิทธิภาพมากที่สุด จะนำไปสู่เป้าหมายได้ทันทีโดยไม่ต้องเสียเวลาลองหรือเรียนรู้อะไรเพิ่มเติม

วิธีการแก้ปัญหาแบบที่ 2

วิธีการนี้ แทนที่จะให้ผู้แก้ปัญหาคิดวิธีแก้ปัญหาจนเสร็จแล้วจึงเขียนโปรแกรมแก้ปัญหาที่คิดได้นั้น อาจแทนปัญหาที่ต้องการแก้ในรูปที่เป็นสากลมากขึ้น และให้คอมพิวเตอร์ค้นหาคำตอบจากปัญหานั้นเอง เช่นให้ (x, y) แทนปริมาณน้ำในถัง 8 และ 6 แกนลอนตามลำดับ เริ่มต้นปัญหาด้วยสถานะ (0,0) จากสถานะเริ่มต้น ตรวจสอบดูว่ามีตัวกระทำการใดบ้างที่จะนำมาใช้กับสถานะเริ่มต้น และทำให้สถานะของปัญหาเปลี่ยนไป นำตัวกระทำการดังกล่าวมาใช้ เมื่อได้สถานะใหม่ก็ตรวจสอบว่าใช่สถานะที่ต้องการหรือไม่ ถ้าไม่ใช่ก็ตรวจสอบตัวกระทำการที่จะนำมาใช้ได้กับสถานะนั้นว่ามีตัวกระทำการใดบ้างและนำมาลองใช้ ทำซ้ำไปเช่นนี้จนกว่าจะได้สถานะที่ต้องการออกมาดังนั้นการหาคำตอบในลักษณะนี้ก็คือการหาเส้นทางที่นำไปสู่สถานะที่ต้องการนั่นเองดังภาพประกอบ 3-7



ภาพประกอบ 3-7 ต้นไม้แสดงสถานะต่างๆ ของปัญหา จากตัวอย่างที่ 3.1

เมื่อแทนปัญหาในรูปต้นไม้ การค้นหาคำตอบก็คือการสำรวจโหนดต่างๆ ในต้นไม้ (Tree Traversal) เมื่อตรวจสอบว่าโหนดใดเป็นโหนดเป้าหมาย เนื่องจากสถานะที่เป็นไปได้ทั้งหมดมักมีจำนวนมาก ในการค้นหาจึงมักไม่สร้างต้นไม้ที่สมบูรณ์ทั้งต้นออกมา แต่ก็จะค่อยๆ กระจายโหนดที่ต้องการสำรวจทีละระดับ จนกระทั่งพบโหนดเป้าหมายหรือกระจายต่อไปอีกไม่ได้แล้ว

การแทนปัญหาในรูปแบบต้นไม้ บางครั้งหลายโหนดที่ใช้แทนสถานะเดียวกัน เช่นในภาพประกอบ 3-7 จะเห็นว่ามีหลายโหนดที่แทนสถานะ (0, 0) ทั้งนี้อาจเป็นเพราะมีตัวกระทำหลายตัวด้วยกันที่เมื่อใช้แล้วนำไปสู่สถานะเดียวกัน ดังนั้นจึงก่อให้เกิดการทำงานซ้ำซ้อน ปัญหานี้จะไม่เกิดถ้าแทนในรูปของกราฟ

ความแตกต่างระหว่างกระบวนการค้นหาคำตอบของปัญหาที่แทนในรูปต้นไม้และแทนในรูปกราฟคือการค้นหาในรูปของกราฟนั้น เวลากระจายโหนดปัจจุบันได้โหนดลูก (Successor) ก่อนที่จะนำโหนดลูกเหล่านี้เพิ่มเข้าในกราฟ จะต้องตรวจสอบว่าโหนดลูกเหล่านี้ เหมือนกับโหนดเก่าที่สร้างไว้แล้วหรือไม่ ถ้าไม่เหมือนก็ให้เพิ่มโหนดเข้าในกราฟได้ แต่ถ้าเหมือนกับที่มีอยู่แล้วไม่ต้องเพิ่มโหนดใหม่เข้าไปในกราฟ แต่ให้โหนดปัจจุบันชี้กลับไปที่โหนดที่มีอยู่แล้วนั้น และถ้าในการค้นหา ทำการเปรียบเทียบด้วยว่าเส้นทางใดดีที่สุด จะต้องตรวจสอบด้วยว่าเส้นทางที่ได้ใหม่ กับเส้นทางเดิม เส้นทางใดดีกว่ากันและบันทึกไว้

ปัญหาหนึ่งที่อาจพบในการค้นหากราฟ คือ ถ้ากราฟมีวัฏจักรจะทำให้เกิดเส้นทางที่มีความยาวเป็นอนันต์ ทำให้การค้นหาไม่สิ้นสุด

การค้นหากราฟดีกว่าการค้นหาต้นไม้ในแง่การช่วยลดเวลาในการสำรวจเส้นทางเดียวกันซ้ำๆ ลงไป แต่จำเป็นต้องเสียเวลาตรวจสอบเพิ่มว่า โหนดใหม่เหมือนกับโหนดที่เคยมีแล้วหรือไม่ ดังนั้น การจะแทนปัญหาในรูปต้นไม้หรือในรูปกราฟจึงขึ้นอยู่กับว่ามีการสร้างโหนดที่เหมือนเดิมบ่อยหรือไม่ ถ้าบ่อยก็ควรแทนในรูปของกราฟแทน

3.5.2 วิธีการค้นหาเป้าหมายหรือคำตอบ

การค้นหาคำตอบก็คือการเลือกเส้นทางที่นำไปสู่จุดหมายปลายทาง หากพบว่าเส้นทางที่เลือก ไม่ได้นำไปสู่จุดหมายปลายทาง อาจมีการย้อนกลับไปลองเส้นทางใหม่ วิธีการค้นหาอาจแบ่งตามลักษณะหลักๆ ได้ 3 ประเภท คือ

(1) การค้นหาเป้าหมายแบบพื้นฐาน

- การค้นหาแบบสร้างและทดสอบ

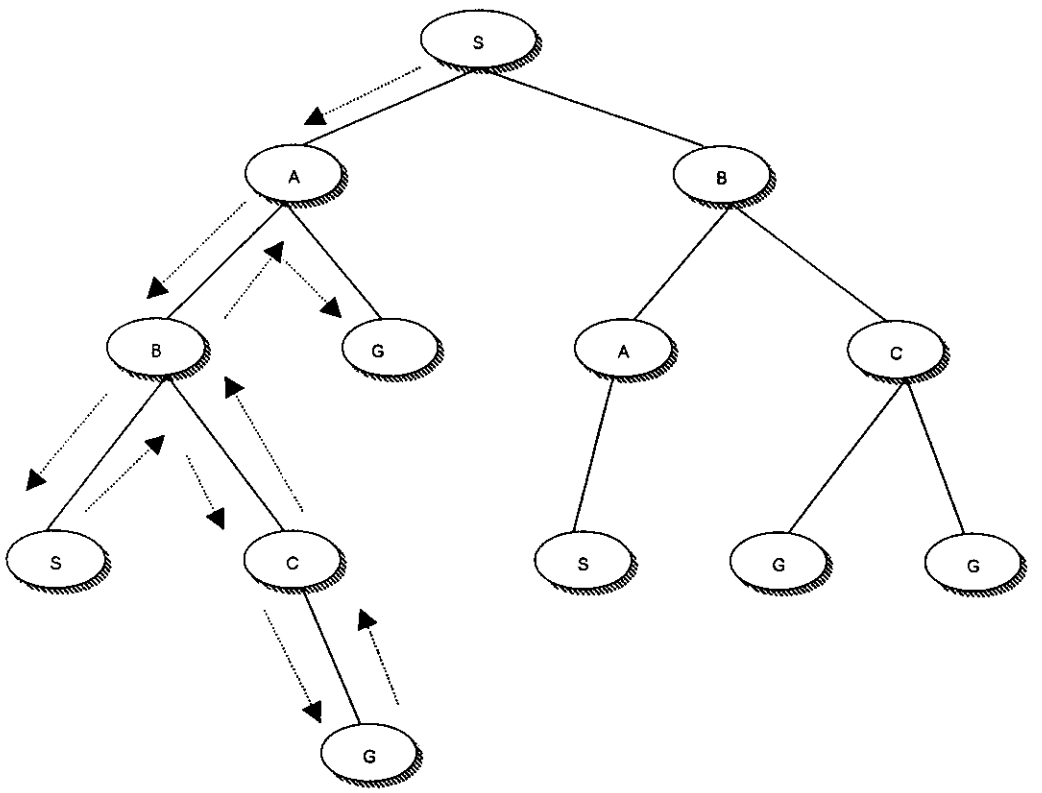
การค้นหาแบบนี้ เป็นการค้นหาแบบที่ง่ายที่สุด วิธีการคือนำตัวกระทำการมาใช้กับสถานะปัจจุบัน ตรวจสอบว่าสถานะที่ได้ใหม่เป็นสถานะเป้าหมายหรือไม่ ถ้าใช่จึงหยุดการค้นหา มิเช่นนั้น

ก็ทำเช่นเดิมต่อไป การเลือกตัวกระทำการมาใช้ จะเลือกสุ่ม ไม่มีกฎหรือระบบว่าต้องเลือกใช้ตัวกระทำใดก่อน ดังนั้น การค้นหาแบบนี้อาจไม่พบเส้นทางที่เป็นคำตอบก็ได้

- การค้นหาแบบลงลึกก่อน

การค้นหาแบบนี้ ต่างจากแบบสร้างและทดสอบตรงที่ว่า การเลือกจะตรวจสอบสถานะใดก่อนหลังทำอย่างมีระบบ ถ้าจำนวนสถานะมีจำกัดและปัญหามีคำตอบ การค้นหาแบบนี้จะนำไปสู่คำตอบได้

ก่อนอื่น จะพิจารณากรณีการค้นหาต้นไม้ก่อน การค้นหาต้นไม้แบบค้นหาลงทางลึกก่อนนี้ถือว่าที่แต่ละจุดที่มีทางเลือกให้นั้น ทางเลือกทุกทางมีโอกาสนำไปสู่คำตอบได้เท่าๆ กัน ดังนั้นในการค้นหาจึงมุ่งลงไปข้างล่างอย่างเดียว โดยไม่สนใจทางเลือกอื่น ถ้าลงไปถึงโหนดปลายกิ่ง (Terminal Node) แล้ว ไม่พบเป้าหมาย จึงย้อนกลับขึ้นไปที่จุดใกล้ที่สุดที่มีทางเลือก และเลือกเดินไปตามทางที่ยังไม่ได้เลือกทางแรกสุดลึกลงไปใหม่อีก ถ้ายังไม่พบเป้าหมายเมื่อลงไปจนสุดแล้ว ก็ย้อนกลับใหม่ตามกฎเดิม ถ้าใช้การค้นหาแบบลงทางลึกก่อนกับต้นไม้ดังภาพประกอบ 3-8 ลำดับการค้นหาจะเป็นตามลูกศรดังในรูป ลำดับการค้นหาจะเริ่มต้นที่ S จนพบโหนดเป้าหมาย G คือ A, B, S, C, G การค้นหาแบบนี้ บางครั้งเรียก Vertical Search

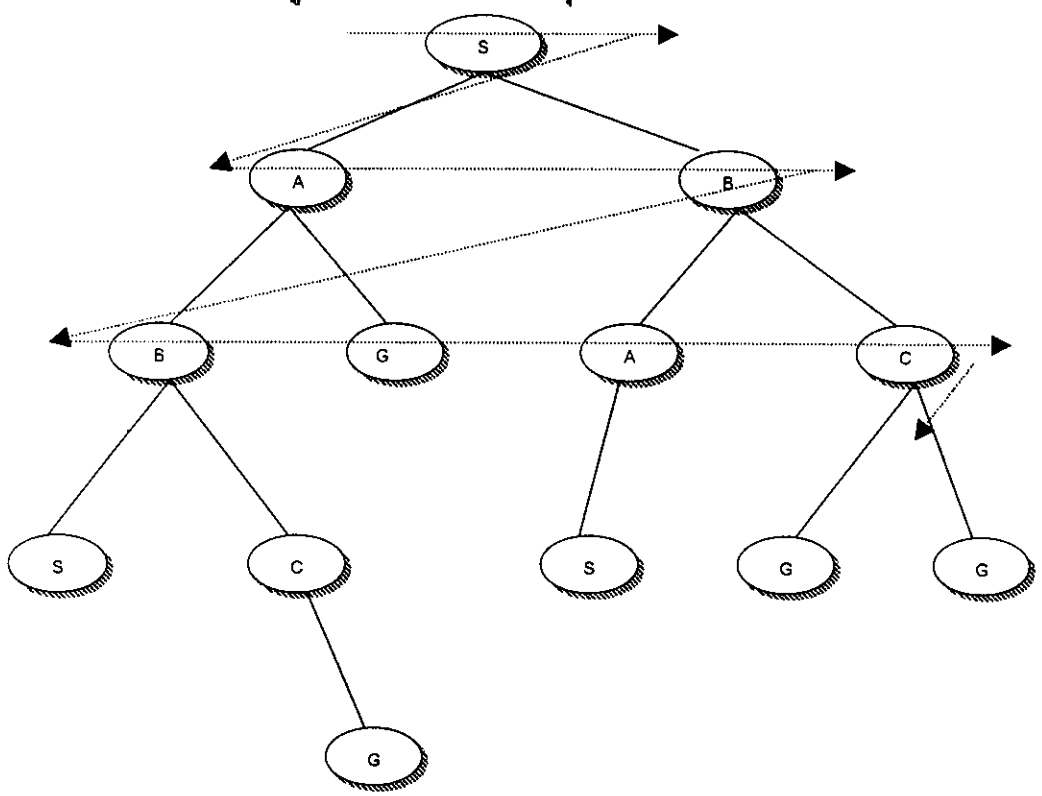


ภาพประกอบ 3-8 การค้นหาต้นไม้ตามแนวลึก

ข้อควรระวังในการใช้วิธีการค้นหาตามแนวลึกก่อนคือ ถ้าเส้นทางบางเส้นทางมีความยาวลงไปลึกมาก ในขณะที่บางเส้นทางลงไป 2-3 ระดับ ก็ถึงโหนดเป้าหมายแล้ว อาจทำให้เสียเวลากับเส้นทางที่มีความลึกมากเกินไปนั้นจนไม่พบเป้าหมายเสียที เพราะมุ่งลงไปข้างล่างเพียงอย่างเดียวมากเกินไป

-การค้นหาตามแนวระดับก่อน

ในการค้นหาต้นไม้ตามแนวระดับก่อน จะค้นหาโหนดที่อยู่ในระดับเดียวกันก่อนโดยจะเริ่มจากโหนดในระดับแรกสุดก่อน เมื่อตรวจสอบโหนดในระดับเดียวกันหมดแล้ว จึงจะค้นหาโหนดในระดับถัดไป การค้นหาในลักษณะนี้บางครั้งเรียก Horizontal Search ตัวอย่างลำดับของการค้นหาอาจดูได้จากภาพประกอบ 3-9 ลำดับของโหนดที่ตรวจสอบคือ A, B, B, G ในกรณีการค้นหากราฟ จะตรวจสอบโหนดที่อยู่ใกล้กับโหนดเริ่มต้นที่สุดก่อนตามลำดับ



ภาพประกอบ 3-9 การค้นหาต้นไม้ตามแนวระดับก่อน

(2) การค้นหาแบบฮิวริสติก

ในกรณีที่สเปซสถานะที่ต้องค้นหาขนาดใหญ่ การค้นหาโดยวิธีพื้นฐานที่กล่าวมาแล้วทั้ง 3 วิธีจะกินเวลามาก ดังนั้นจึงได้มีการพิจารณาตัดโหนดที่ไม่จำเป็นต้องสำรวจออกไปเสียบ้าง เพื่อลด

เวลาในการสำรวจลงไปโดยอาจใช้ความรู้เกี่ยวกับโหนดต่างๆ มาช่วยในการเลือกเส้นทางเพื่อให้ถึงเป้าหมายโดยเร็วขึ้น วิธีการค้นหาที่นำความรู้เกี่ยวกับปัญหามาช่วยในการเลือกเส้นทางนี้เรียกว่าการค้นหาแบบฮิวริสติก

3.6 ภาษา LISP

ภาษา LISP เป็นภาษาทางคอมพิวเตอร์ชั้นสูงภาษาหนึ่ง เป็นภาษาที่ใช้ได้ดีในการประมวลผลทางสัญลักษณ์ (Symbol) จึงเป็นภาษาที่ใช้กันอย่างแพร่หลายในทางด้านปัญญาประดิษฐ์ (Artificial Intelligence :AI) โดยลักษณะพิเศษของภาษานี้คือ เป็นภาษาที่มีการตอบรับทันทีที่โปรแกรมถูกป้อนเข้าไป และจะมีการตรวจสอบไวยากรณ์ของภาษาด้วยตัวมันเองโดยที่ผู้ใช้ ไม่จำเป็นต้องออกจากสภาพแวดล้อมของภาษา LISP จะส่งผลลัพธ์ของการหาค่า (Evaluation) ทันทีที่มีการพิมพ์โปรแกรมสิ้นสุด ดังนั้นในการโปรแกรมของภาษา เราไม่จำเป็นต้องมีการประกาศชนิดของตัวแปรที่ถูกใช้ในโปรแกรม (นิตยา นินทรกิจ, 2543)

เหตุผลในการเลือกใช้ภาษา LISP

- โครงสร้างทางข้อมูลของภาษา LISP คือ List ซึ่งเป็นโครงสร้างที่โปรแกรม AI ส่วนมากใช้ในการแทนความรู้

- สามารถรวบรวมข้อเท็จจริงเกี่ยวกับสิ่งต่างๆ ได้ง่าย โดยการแทนข้อมูลต่างๆ อยู่ในรูป Property List

- โครงสร้างในการควบคุมภาษา LISP ที่สำคัญคือ Recursion ซึ่งเหมาะกับการแก้ปัญหาหลายแบบ

- การกำหนดค่าให้กับตัวแปรจะกำหนดช่วงไหนก็ได้

ในงานวิจัยนี้จะใช้ภาษา LISP เป็นภาษาหลักในการเขียนโปรแกรมควบคุมส่วนต่างๆ เช่น กลไกการอนุมาน ฐานความรู้ หรือส่วนที่ใช้สำหรับการติดต่อกับผู้ใช้ แต่อาจจะมีโปรแกรมภาษาอื่นบ้างที่ต้องนำมาใช้ร่วมกัน เช่น ภาษาซี เป็นต้น ซึ่งภาษา LISP ที่ใช้นี้ทำงานบนระบบปฏิบัติการ LINUX

3.7 การพัฒนาระบบผู้ชำนาญการ

ดังกล่าวแล้ว ระบบผู้ชำนาญการเป็นระบบที่ใช้แก้ปัญหาที่จำเป็นต้องใช้ความรู้ความชำนาญระดับสูง งานที่ต้องใช้ความรู้ความชำนาญระดับสูงพวกนี้มีหลายลักษณะด้วยกันเช่น เป็น

3.7.1 การตีความ (Interpretation) ได้แก่การวิเคราะห์ข้อมูลวิเคราะห์ข้อมูลเพื่อหาว่าความหมายที่ได้จากข้อมูลคืออะไร เช่น ดูข้อมูลจาก Mass Spectrometer และพยายามตีความจากข้อมูลว่าโครง

สร้างทางเคมีคืออะไร งานในลักษณะนี้จะต้องตีความให้ถูกต้อง การตีความควรมีความคงเส้นคงวา วิธีการโดยปกติมักพิจารณาตีความที่เป็นไปได้ทั้งหมดก่อน แล้วจึงตัดข้อใดข้อหนึ่งทิ้งไป เมื่อมีหลักฐานเพียงพอว่าการตีความแบบนั้นผิด ปัญหาของงานในลักษณะนี้คือ ข้อมูลมักมีความคลาดเคลื่อนรวมอยู่ด้วย ดังนั้นอาจกล่าวได้ว่าผู้ตีความมีข้อเท็จจริงอยู่เพียงบางส่วน ถ้าข้อมูลที่ได้มาเชื่อถือไม่ได้ การตีความก็จะเชื่อถือไม่ได้ไปด้วย จึงจำเป็นที่ผู้ตีความจะต้องจำแนกให้ได้ว่าข้อมูลส่วนไหนเชื่อถือไม่ได้หรือไม่สมบูรณ์

3.7.2 การวินิจฉัย (Diagnosis) ได้แก่การตรวจสอบว่ามีอะไรผิดพลาดในระบบ (Fault Finding) เช่นการวินิจฉัยโรคติดเชื้อ งานในลักษณะนี้ผู้วินิจฉัยจำเป็นต้องเข้าใจว่าระบบประกอบกันขึ้นมาได้อย่างไรและแต่ละส่วนย่อยๆ ในระบบสัมพันธ์กันอย่างไร งานในลักษณะนี้มีความยากตรงที่ความผิดพลาดอย่างหนึ่งอาจมีอาการซึ่งเป็นสาเหตุของความผิดพลาดอีกอย่างหนึ่งปิดบังอยู่ บางครั้งข้อมูลในระบบก็ไม่สามารถดึงออกมาได้ หรือแพงเกินไปหรือเป็นอันตรายมากเกินกว่าจะดึงออกมาได้

3.7.3 การควบคุมการทำงาน (Monitoring) คือการตีความสัญญาณที่ส่งออกมาจากระบบว่ามีอะไรผิดพลาดหรือไม่และต้องแจ้งให้ผู้ทำงานทราบทันทีที่เมื่อเกิดความผิดพลาดขึ้น เช่นการควบคุมการทำงานของเครื่องช่วยหายใจของคนไข้ผ่าตัด งานในลักษณะนี้จะต้องเห็นว่ารวมเอางานวินิจฉัยเอาไว้ด้วย เพราะจำเป็นต้องวินิจฉัยว่าความผิดพลาดในระบบคืออะไร และต้องแจ้งสาเหตุแห่งความผิดพลาดให้ผู้ใช้เครื่องมือทราบทันที

3.7.4 การทำนาย (Prediction) ได้แก่การทำนายพฤติกรรมในอนาคตจากตัวแบบซึ่งได้จากอดีตและปัจจุบัน เช่น การทำนายผลกระทบเนื่องจากการเปลี่ยนแปลงนโยบายทางเศรษฐกิจ การทำนายเป็นการหาเหตุผลที่เกี่ยวเนื่องกับเวลา ดังนั้นผู้ทำนายจะต้องสามารถอ้างอิงสิ่งต่างๆ ที่เปลี่ยนแปลงไปตามเวลาได้ การทำนายจำเป็นต้องอาศัยความรู้ที่มีอยู่ซึ่งไม่สมบูรณ์มาทำนาย ในการทำนายจริงๆ จึงควรบอกทางที่จะเป็นไปได้ไว้หลายๆ ทางพร้อมทั้งบอกด้วยว่าถ้าข้อมูลมีการเปลี่ยนแปลงจะทำอะไรเปลี่ยนแปลงได้บ้าง

3.7.5 การวางแผน (Planning) การวางแผนก็คือการกำหนดลำดับของการกระทำที่สามารถนำไปสู่เป้าหมายได้ เช่นการวางแผนการทดลอง เกี่ยวกับ Molecular Genetics คนวางแผนจะต้องสร้างแผนขึ้นมาในลักษณะที่สามารถนำไปสู่เป้าหมายได้โดยไม่ใช้ทรัพยากรมากเกินไป และไม่ขัดแย้งกับข้อจำกัดต่างๆ ที่มี ถ้าเป้าหมายขัดแย้งกัน ผู้วางแผนต้องสามารถจัดลำดับความสำคัญได้ ถ้าข้อมูลหรือข้อจำกัดเกี่ยวกับแผนมีการเปลี่ยนแปลงตามเวลาหรือไม่สมบูรณ์ ผู้วางแผนก็ต้องสามารถปรับเปลี่ยนได้ ปกติแล้วในการวางแผนจะมีบางส่วนที่ต้องใช้การทำนายเข้ามาช่วย ปัญหาในการวางแผนคือ ระบบที่ต้องวางแผนมักใหญ่และซับซ้อน ผู้วางแผนอาจเข้าใจปัญหาในทันทีไม่ได้ จำเป็นต้องใช้เวลาในการศึกษาอย่างละเอียดรอบคอบ ถ้ามีรายละเอียดมาก อาจต้องแบ่งเป็นปัญหา

ย่อยๆ ก่อน ถ้าปัญหาย่อยๆ นี้มีความสัมพันธ์กัน ผู้วางแผนต้องสามารถดำเนินการกับความสัมพันธ์เหล่านี้ได้ ข้อมูลที่ใช้ในการวางแผนมักมีความไม่แน่นอนรวมอยู่ด้วย ดังนั้นผู้วางแผนควรกำหนดได้ว่าแผนที่วางไว้นั้นโอกาสที่จะต้องใช้เป็นเท่าไร

3.7.6 การออกแบบ (Designing) คือการกำหนดรายละเอียด (Specification) ของสิ่งใดสิ่งหนึ่งให้มีคุณสมบัติตามที่ต้องการ เช่น การออกแบบ Digital Circuit ลักษณะในรายละเอียดของการออกแบบจะคล้ายกับการวางแผน

การพัฒนาระบบผู้ชำนาญการจึงขึ้นอยู่กับลักษณะของปัญหาในลักษณะใดที่กล่าวมาแล้ว ซึ่ง ปัจจัยสำคัญที่จะทำให้การพัฒนาระบบชำนาญการประสบความสำเร็จหรือไม่ คือ การเลือกงานที่จะนำมาสร้างเป็นระบบผู้ชำนาญการ

ลักษณะที่สำคัญของระบบผู้ชำนาญการคือฐานความรู้มีขนาดใหญ่ เนื่องจากความรู้มีเป็นจำนวนมาก ดังนั้นในการออกแบบระบบผู้ชำนาญการ จุดที่สำคัญที่สุดจุดหนึ่งที่ต้องพิจารณาคือจะแทนความรู้นั้นในรูปใด นอกจากนั้นแล้ว เนื่องจากความรู้นั้นมีใหม่เพิ่มเติม เปลี่ยนแปลงอยู่ตลอดเวลาดังนั้นจึงมักแยกฐานความรู้ออกจากตัวโปรแกรมส่วนอื่นๆ

ในการสร้างฐานความรู้ควร

- นิยามงานที่จะทำให้ชัดเจนว่าอะไรคือข้อมูลที่ต้องการนำมาเก็บไว้และอะไรคือสิ่งที่ต้องการได้ออกมา (Output) Concept พื้นฐานเกี่ยวกับงานนั้นมีอะไรบ้างและความสัมพันธ์ระหว่าง Concept เหล่านั้นคืออะไร มีตัวแปรเกี่ยวกับความรู้ในงานนั้นหรือยัง ถ้ามีนิยามไว้ดีแล้วควรนำมาใช้
- ให้ผู้เชี่ยวชาญให้ความร่วมมือด้วย โดยผู้เชี่ยวชาญควรเป็นผู้เชี่ยวชาญจริงๆ ไม่ใช่ผู้เชี่ยวชาญสมัครเล่น และระบบผู้ชำนาญการที่จะสร้างขึ้นเพื่อเป็นเครื่องช่วยผู้เชี่ยวชาญ ไม่ใช่สร้างขึ้นเพื่อแทนผู้เชี่ยวชาญ
- เปรียบเทียบกับปัญหาทางปัญญาประดิษฐ์หลายๆ แบบ เพื่อดูแนวทางแก้ปัญหาของเขาดูข้อดีข้อเสีย อย่าพยายามแก้ปัญหาในรูปแบบที่ตัวเองชอบ ให้ดูวิธีแก้ปัญหานั้นเหมาะกับปัญหาที่ต้องการหรือไม่
- เมื่อเริ่มต้น ให้สร้างระบบเล็กๆ ขึ้นมาก่อน โดยอาจเริ่มต้นด้วยโครงการ 6 เดือน สร้างกลไกการอนุมานอย่างง่ายๆ ขึ้นมา มุ่งไปที่กฎที่ใช้ร่วมกันก่อน สร้างเครื่องมือสำหรับสร้างและแก้ไขกฎต่างๆ การเพิ่มเติม ลบข้อมูลและเอาข้อมูลออกจากฐานความรู้ แล้วจึงปรับปรุงให้ทำสิ่งยากๆ ในภายหลัง

- อย่าคอยจนระบบสมบูรณ์แล้วจึงนำมาใช้ ให้ลองใช้งานไปก่อน อะไรที่ต้องทำใหม่หรือเพิ่มเติมให้ทำภายหลัง
- ตัวอย่างที่นำมาใช้ทดสอบกับระบบที่สร้างเสร็จแล้วควรเก็บไว้เพื่อประโยชน์ในการอ้างอิงในภายหลัง