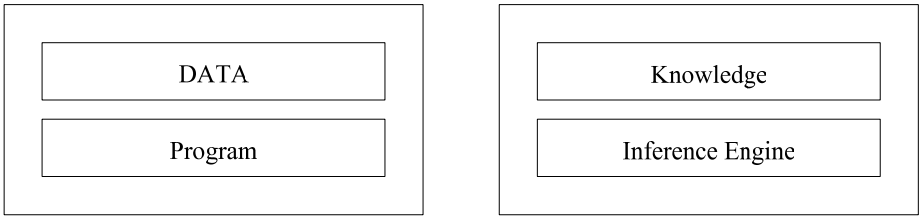
# บทที่ 3
# ทฤษฎีของระบบผู้เชี่ยวชาญ

ระบบผู้เชี่ยวชาญ (Expert system) คือโปรแกรมคอมพิวเตอร์ที่ได้รับการออกแบบและพัฒนาขึ้นให้มีความเฉลียวฉลาดกว่าโปรแกรมทั่วๆไป สามารถแก้ไขปัญหาโดยใช้ความรู้ (Knowledge) และกระบวนการอนุมาน (Inference process) สามารถแก้ไขปัญหาที่ยุ่งยากซึ่งโดยปกติแล้วต้องใช้ความรู้ความชำนาญของผู้เชี่ยวชาญจึงจะแก้ปัญหาได้ ความรู้ในที่นี้ก็คือความจริง ซึ่งอาจจะมาจากเอกสารทางวิชาการ ตำราต่างๆ และประสบการณ์ของผู้เชี่ยวชาญที่เป็นมนุษย์ กระบวนการอนุมานในการแก้ปัญหามักจะไม่สามารถกำหนดขั้นตอนไว้ล่วงหน้าเหมือนอย่างโปรแกรมทั่วๆไป ต้องใช้สถานการณ์ และความรู้ประกอบเข้าด้วยกัน จึงจะสามารถแก้ปัญหานั้นๆ ได้ [23]

ศาสตราจารย์ Edward Feigenbaum แห่งมหาวิทยาลัยแสตนฟอร์ด ซึ่งเป็นนักค้นคว้าชั้นแนวหน้าในสาขาปัญญาประดิษฐ์ (Artificial intelligence) ได้ให้คำจำกัดความของระบบผู้เชี่ยวชาญไว้ว่า ระบบผู้เชี่ยวชาญคือโปรแกรมคอมพิวเตอร์ที่มีความฉลาดด้วยการใช้ ความรู้ และกระบวนการอนุมาน ในการแก้ปัญหาที่ยุ่งยากขนาดที่ต้องใช้ประสบการณ์ความชำนาญการของมนุษย์จึงจะแก้ได้ ปัญหาที่ระบบผู้เชี่ยวชาญจะแก้ส่วนใหญ่จะเป็นปัญหาที่ยุ่งยากและไม่ค่อยมีโครงสร้าง (Semi-structured หรือ ill-structured problem) คำตอบของปัญหาประเภทนี้จะมีโอกาสเป็นได้หลายอย่างขึ้นอยู่กับสภาพข้อมูลที่เข้ามาและลักษณะของปัญหา ในการแก้ปัญหาประเภทนี้มักจะไม่สามารถกำหนดขั้นตอนในการแก้ไขปัญหาไวอย่างชัดเจนล่วงหน้าได้ ดังนั้นการแก้ปัญหาโดยใช้โปรแกรมธรรมดาซึ่งเขียนโปรแกรมเป็นขั้นตอนในการแก้ปัญหาหรือที่เรียกว่าขั้นตอนวิธีการทางคอมพิวเตอร์ (Algorithm) ดูรายละเอียดเพิ่มเติมได้ที่ [24] จึงไม่สามารถจะประยุกต์ใช้ในการแก้ปัญหาซับซ้อนประเภทนี้ได้ และได้มีการนำระบบผู้เชี่ยวชาญไปประยุกต์ใช้กับงานในหลายลักษณะ คือ การจำลองแบบ การวางแผน การวินิจฉัย การตรวจสอบ การควบคุม การศึกษา และการจัดการอบรม เป็นต้น [23]

## 3.1 ข้อแตกต่างระหว่างโปรแกรมธรรมดากับโปรแกรมระบบผู้เชี่ยวชาญ

โปรแกรมธรรมดากับโปรแกรมระบบผู้เชี่ยวชาญมีข้อแตกต่างกันดังรูป 3.1




ก) โปรแกรมธรรมดา ข) โปรแกรมระบบผู้เชี่ยวชาญ

รูป 3.1 ลักษณะโปรแกรมธรรมดากับโปรแกรมระบบผู้เชี่ยวชาญ

### 3.1.1 โปรแกรมธรรมดา

โปรแกรมธรรมดา จะใช้ข้อมูล (Data) เช่น อุณหภูมิ ความดัน อัตราการไหล เป็นต้น โดยนำข้อมูลไปประมวลผล และอาจมีการเก็บข้อมูลไว้ในฐานข้อมูล (Database) ส่วนโปรแกรม (Program) คือคำสั่งที่เขียนเป็นลำดับ สั่งให้คอมพิวเตอร์ปฏิบัติตามเรียงลำดับตั้งแต่ต้นจนจบ มักรู้ผลที่จะเกิดได้ก่อน

### 3.1.2 โปรแกรมระบบผู้เชี่ยวชาญ

โปรแกรมระบบผู้เชี่ยวชาญ จะใช้ความรู้ (Knowledge) ที่เราสนใจหรือเกี่ยวข้อง โดยอาจเก็บความรู้เป็นแบบกฎ ซึ่งมีลักษณะเงื่อนไข เช่น IF (ส่วนเงื่อนไข) THEN (ส่วนข้อสรุปหรือส่วนการปฏิบัติ) ส่วนกลไกการอนุมาน (Inference Engine) เป็นการนำความรู้ มาหาข้อเท็จจริง กลไกการอนุมานแบ่งออกเป็น 2 ประเภทคือ การอนุมานเป็นห่วงโซ่แบบไปข้างหน้า (Forward chaining inference) และการอนุมานเป็นห่วงโซ่แบบย้อนหลัง (Backward chaining inference)

## 3.2 ความแตกต่างระหว่างความรู้ (Knowledge) กับ ข้อมูล (Data)

ความแตกต่างระหว่างความรู้กับข้อมูล มีดังนี้คือ ความรู้ส่วนใหญ่แสดงออกในรูปของภาษาธรรมชาติ (Natural language) เช่นภาษาไทย ภาษาอังกฤษ มีความหมายอยู่ในตัวเอง เช่นถ้าฝนตก ไม่ควรขับรถจักรยานยนต์ ซึ่งหมายความว่าการขับรถจักรยานยนต์ขณะฝนตก ทัศนวิสัยไม่ดี พื้นถนนเปียก แฉะ ลื่น การทรงตัวไม่ดี และจะเกิดอุบัติเหตุได้ง่าย ผู้พูดเป็นห่วง เป็นต้น ส่วนข้อมูลจะหมายถึงสิ่งของ ปริมาณหรือขนาดของค่าที่วัด เป็นต้น เช่นจำนวนเครื่องคอมพิวเตอร์ 5 เครื่อง ความดันไอ 120 bar (g) เป็นต้น ไม่ได้แฝงความหมายในตัวเองไว้ ดังนั้นก่อนที่จะสามารถสร้างระบบคอมพิวเตอร์ที่บันทึกและประมวลผลความรู้ได้ จำเป็นจะต้องพัฒนาระบบคอมพิวเตอร์ที่

เข้าใจภาษาธรรมชาติได้ก่อน ซึ่งเป็นเรื่องที่ยากมาก เนื่องจากภาษาธรรมชาติมีลักษณะกำกวมอยู่มากซึ่งเป็นสิ่งที่คอมพิวเตอร์จัดการได้ยาก

ตาราง 3.1 ผลสรุปข้อแตกต่างระหว่างโปรแกรมธรรมดากับโปรแกรมระบบผู้เชี่ยวชาญ

| **ลักษณะ (Characteristic)** | **โปรแกรมธรรมดา (Conventional program)** | **โปรแกรมระบบผู้เชี่ยวชาญ (Expert System)** |
|---|---|---|
| ควบคุมการทำงานโดย | คำสั่งตามลำดับ | กลไกอนุมาน |
| การควบคุมและข้อมูล | รวมกันอยู่ ไม่แยกชัดเจน | แยกกันอย่างชัดเจน |
| การประมวลผลโดยการค้นหา | ขั้นตอนวิธีทางคอมพิวเตอร์ | กฎและการอนุมาน |
| การค้นหา | เล็กน้อย หรือไม่มี | มาก |
| แก้ปัญหาโดย | ขั้นตอนวิธีทางคอมพิวเตอร์ที่ถูกต้อง | กฎ |
| ข้อมูลนำเข้า | ต้องถูกต้อง | อาจไม่สมบูรณ์ อาจไม่ถูกต้อง |
| ข้อมูลนำเข้าที่ผิดปกติ | ทำงานผิด | ทำงานได้ตามปกติ |
| ผลลัพธ์ | ถูกต้อง | เปลี่ยนแปลงขึ้นกับปัญหา |
| การออกแบบโปรแกรม | มีโครงสร้าง | ไม่มีโครงสร้าง |
| การแก้ไขดัดแปลง | ยาก | ง่ายโดยเพิ่มเหตุผลเข้าไป |

## 3.3 ส่วนประกอบของระบบผู้เชี่ยวชาญ

ระบบผู้เชี่ยวชาญโดยทั่วไปจะประกอบด้วยส่วนประกอบ 7 ส่วน มีรายละเอียดดังนี้

### 3.3.1 ฐานความรู้ (Knowledge base)

ส่วนนี้เปรียบเสมือนกับข้อมูลในซอฟต์แวร์ธรรมดาหรือฐานข้อมูล (Database) ในระบบสารสนเทศ (Information system) เป็นส่วนที่ใช้เก็บความรู้ทุกประเภท ไม่ว่าจะเป็นความรู้ที่ได้จากตำรา หรือความรู้ที่ได้จากประสบการณ์ ปัญหาหลักของฐานความรู้คือการเลือกวิธีการแสดงความรู้ หรือโครงสร้างสำหรับเก็บความรู้ที่เหมาะสม ปัญหานี้เปรียบได้กับการเลือกโครงสร้างข้อมูล หรือโครงสร้างฐานข้อมูลที่เหมาะสม ในระบบ ซอฟต์แวร์ธรรมดา

### 3.3.2 กลไกการอนุมาน (Inference engine)

ส่วนนี้เปรียบได้กับขั้นตอนวิธีทางคอมพิวเตอร์ เป็นส่วนที่ควบคุมการใช้ความรู้เพื่อแก้ไขปัญหาอย่างมีประสิทธิภาพ กลไกการอนุมานมีหลายแบบ แต่แยกเป็นประเภท

ใหญ่ๆ ได้ 2 ประเภท คือ การอนุมานเป็นห่วงโซ่แบบไปข้างหน้า (Forward chaining inference) และการอนุมาน เป็นห่วงโซ่แบบย้อนหลัง (Backward chaining inference) ทั้งสองวิธีนี้ต่างก็มีจุดดีและจุดเสีย ทั้งนี้ขึ้นอยู่กับลักษณะของปัญหา ในระบบผู้เชี่ยวชาญบางระบบจะใช้วิธีทั้งสองวิธีนี้รวมกัน

**3.3.3 ส่วนเพิ่มเติมความรู้ (Knowledge acquisition module)**

ส่วนนี้เป็นส่วนของระบบผู้เชี่ยวชาญที่ใช้ช่วยในการดึงเอาความรู้จากตำรา หรือฐานข้อมูล และจากผู้เชี่ยวชาญ การดึงเอาความรู้จากตำรา หรือฐานข้อมูลนั้นทำได้ไม่ยาก ถ้าหากเราสามารถจัดความรู้จากแหล่งดังกล่าวให้เป็นระบบ และเข้ากันได้กับโครงสร้างของฐานความรู้ เราก็จะสามารถบรรจุความรู้เหล่านี้เข้าไปในฐานข้อมูลได้ แต่การดึงเอาความรู้จากผู้เชี่ยวชาญนั้น ทำได้ยาก จำเป็นต้องใช้เทคนิคต่างๆ เข้าช่วย หรือไม่ก็ทำให้ระบบผู้เชี่ยวชาญสามารถเรียนรู้ด้วยตนเองในบางส่วนได้

**3.3.4 ส่วนให้คำอธิบาย (Explanation module)**

ส่วนนี้ทำหน้าที่อธิบายรายละเอียดขั้นตอนการวินิจฉัย ให้แก่ผู้ใช้งาน ข้อสรุป หรือคำตอบนั้น ได้มาอย่างไร และทำไม รวมทั้งใช้แสดงเส้นทางการอนุมาน

**3.3.5 ส่วนติดต่อกับผู้ใช้งาน (User interface unit)**

ส่วนนี้เป็นส่วนที่เป็นตัวกลางระหว่างผู้ใช้งาน กับระบบผู้เชี่ยวชาญ เพื่อทำให้การสื่อสารระหว่างผู้ใช้งาน กับระบบผู้เชี่ยวชาญ เป็นไปได้อย่างราบรื่นและเป็นที่ยอมรับ

**3.3.6 ส่วนสอบถามความรู้ (Knowledge query module)**

เป็นส่วนที่นำความรู้ที่มีอยู่ในระบบ แสดงและให้ความรู้แก่ผู้ใช้งาน

ในระบบผู้เชี่ยวชาญบางระบบจะไม่มีส่วนประกอบทั้ง 7 ส่วนดังกล่าว แต่ที่ขาดไม่ได้ คือ ฐานความรู้และกลไกการอนุมาน

## 3.4 ปัญหาสำคัญในสาขาปัญญาประดิษฐ์

ในการค้นคว้าเกี่ยวกับ ปัญญา หรือความรู้ นั้น แยกออกเป็นปัญหาได้ 3 ปัญหา ดังนี้

**3.4.1 ปัญหาการแสดงความรู้**

คือปัญหาการเลือกแบบในการแสดงความรู้ ความรู้นี้อาจจะได้แก่ความรู้ที่ปรากฏในตำรา หรือเอกสารทางวิชาการต่างๆ หรืออาจจะเป็นความรู้ที่ได้จากประสบการณ์ที่สะสมมานานของผู้เชี่ยวชาญ รูปแบบการแสดงความรู้ ควรเหมาะแก่การเก็บในคอมพิวเตอร์ และควรเป็นรูปแบบที่ทำให้การนำความรู้ไปใช้ ทำได้ง่าย ปัญหาการแสดงความรู้อาจจะพูดได้อีกอย่างว่า เป็นปัญหาการออกแบบฐานความรู้

### 3.4.2 ปัญหาการใช้ความรู้

คือปัญหาวิธีการนำเอาความรู้ที่เก็บอยู่ในรูปแบบใดรูปแบบหนึ่งมาใช้ในการแก้ปัญหา สามารถเรียกปัญหานี้ได้อีกอย่างหนึ่งว่า เป็นปัญหาการออกแบบกลไกการอนุมาน ความสัมพันธ์ระหว่างการแสดงความรู้กับการใช้ความรู้นั้นเปรียบเสมือนกับความสัมพันธ์ระหว่างโครงสร้างข้อมูล กับ ขั้นตอนวิธีทางคอมพิวเตอร์

### 3.4.3 ปัญหาการรับความรู้

คือปัญหาการถ่ายเทความรู้ที่เป็นทั้งความรู้ที่ได้จากตำรา และความรู้ที่ได้จากประสบการณ์ของผู้เชี่ยวชาญ เกี่ยวกับปัญหาที่ต้องการแก้ไขไปยังฐานความรู้ หัวข้อสำคัญๆ ในปัญหานี้ได้แก่ วิธีการดึงเอาความรู้จากผู้เชี่ยวชาญ การสร้างฐานความรู้ที่สมบูรณ์และไม่ขัดแย้งในตัวเอง

ปลายปี พ.ศ. 2513 (ค.ศ. 1970) เริ่มมีการสร้างระบบผู้เชี่ยวชาญขึ้น ถือได้ว่าเป็นการเริ่มประยุกต์ใช้วิชาปัญญาประดิษฐ์ จากนั้นเริ่มแบ่งวิชาปัญญาประดิษฐ์ออกเป็น 2 แขนง คือ แขนงที่เน้นการประยุกต์ซึ่งก็คือ วิศวกรรมความรู้ และแขนงที่เน้นทางด้านทฤษฎี

## 3.5 วิศวกรรมความรู้ (Knowledge engineering)

วิศวกรรมความรู้ เป็นแขนงย่อยอีกอันหนึ่งของวิชาปัญญาประดิษฐ์ ที่เกี่ยวข้องกับการรับการแสดง และการใช้ความรู้ของมนุษย์ในรูปของสัญลักษณ์ ความรู้ของมนุษย์นั้นนอกจากจะอยู่ในรูปของความสามารถในการให้ข้อเท็จจริงต่างๆ แล้วยังรวมถึงความสามารถในการติดตามเหตุผล (Reasoning) หรือการอนุมาน (Inference) โดยเป้าหมายหลักของวิศวกรรมความรู้ คือความต้องการสร้างระบบคอมพิวเตอร์ หรือเรียกกันทั่วไปว่า ซอฟต์แวร์ ที่ใช้งานได้จริงๆ สำหรับแก้ปัญหาที่ยุ่งยากซับซ้อน และเป็นปัญหาเฉพาะที่มีขอบข่ายแคบ เช่น การวิเคราะห์ปัญหาการผลิตของโรงไฟฟ้า การออกแบบสินค้าให้เข้ากับความต้องการของลูกค้า เป็นต้น

มนุษย์สามารถแก้ปัญหาที่ไม่เคยพบมาก่อน หรือปัญหาที่มีลักษณะพิเศษได้ เพราะมนุษย์มีความรู้พื้นฐานที่เกี่ยวข้องทั้งโดยตรงและทางอ้อมกับปัญหาเหล่านั้น ในทำนองคล้ายกัน ถ้าหากจะสร้างโปรแกรมที่มีความยืดหยุ่นสามารถแก้ปัญหาที่ยุ่งยากซับซ้อนได้ นอกจากโปรแกรมนั้นจะต้องประกอบด้วยขั้นตอนวิธีทางคอมพิวเตอร์ที่ดี และมีประสิทธิภาพแล้ว ยังจำเป็นจะต้องมีความรู้โดยความรู้ที่กล่าวขึ้นนี้ ใช้เป็นทั้งข้อมูลในการแก้ปัญหา และเป็นตัวชี้นำสำหรับขั้นตอนวิธีทางคอมพิวเตอร์ต่างๆ

วิศวกรรมความรู้ เน้นศึกษาค้นคว้าเกี่ยวกับ 4 หัวข้อ ดังนี้

**3.5.1 การแสดงความรู้ (Knowledge representation)**

เป็นการออกแบบการบันทึกความรู้ให้สามารถนำไปใช้งานได้ง่าย

**3.5.2 วิธีการใช้ความรู้หรือกลไกการอนุมาน**

เป็นการผสมผสานความรู้ แล้วใช้งานให้บรรลุจุดประสงค์

**3.5.3 การรับเอาและจัดการความรู้ (Knowledge acquisition and management)**

เป็นส่วนที่จะทำให้ฐานความรู้ สามารถเพิ่มความรู้ได้โดยอัตโนมัติ หรือกึ่งอัตโนมัติ และจัดการฐานความรู้โดยไม่เกิดความขัดแย้งในระหว่างความรู้ที่อยู่ในฐานความรู้ หรือความรู้ที่รับเข้ามาใหม่

**3.5.4 การติดต่อกับผู้ใช้งาน (User interface)**

ส่วนนี้ จะสร้างส่วนติดต่อกับผู้ใช้งาน ที่ทำให้ผู้ใช้งานรู้สึกใช้ง่าย และยอมรับระบบ รวมทั้งการประมวลผลแบบกราฟิก และภาษาธรรมชาติด้วย

## 3.6 ประเภทของความรู้

ความรู้ แบ่งออกเป็น 4 ประเภท คือ

3.6.1 ความรู้ที่บอกความจริง ลักษณะ หรือคุณสมบัติ เช่น มนุษย์มี 4 ขา

3.6.2 ความรู้ที่บอกความสัมพันธ์ เช่น ยิ่งเพิ่มเชื้อเพลิง อุณหภูมิยิ่งสูง

3.6.3 ความรู้ที่บอกขั้นตอน หรือวิธีการ เช่น หลังเลิกงานให้ปิดเครื่องคอมพิวเตอร์

3.6.4 ความรู้ที่เกี่ยวกับความรู้ (Meta knowledge) เช่น ความรู้เกี่ยวกับคุณสมบัติของความรู้อื่น หรือเกี่ยวกับความรู้ (Meta knowledge) เช่น ความรู้เกี่ยวกับคุณลักษณะของความรู้อื่น หรือเกี่ยวกับวิธีการใช้ความรู้อื่น

## 3.7 การแสดงความรู้ในรูปกฎ

การแสดงความรู้ในรูปของกฎ เรียกอีกอย่างว่า การแสดงความรู้ในรูประบบโพรดักชัน (Production system: PS) เป็นโมเดลคอมพิวเตอร์ชนิดหนึ่งที่เสนอโดย E.L. Post ในปี พ.ศ. 2486 (ค.ศ. 1943) มีพื้นฐานทางทฤษฎีอยู่บนโพสท์แมชชีน (Post machine) โดยใน โพสท์ แมชชีน ขั้นตอนการปฏิบัติการ หรือการประมวลผล จะถูกบันทึกอยู่ในรูปของเซตของกฎ กฎอันไหนจะถูกใช้ก่อนหลังนั้น ไม่ได้ขึ้นอยู่กับลำดับการบันทึกกฎ แต่ขึ้นอยู่กับว่าเงื่อนไขของกฎนั้นครบสมบูรณ์หรือไม่ ถ้าครบ ก็จะมีการปฏิบัติตามกฎนั้น ในปัจจุบันเครื่องคอมพิวเตอร์มีพื้นฐานทางทฤษฎีอยู่บน ทัวริง แมชชีน (Turing machine) โดยใน ทัวริง แมชชีน จะบันทึกขั้นตอนการควบคุมไว้ทั้งหมด การปฏิบัติ หรือการประมวลผลของเครื่องจะเป็นไปตามขั้นตอนการควบคุม ทัวริง แมชชีน และ โพสท์ แมชชีน มีความสามารถในการประมวลผลเท่าเทียมกัน

แสดงว่า ความสามารถในการประมวลผลของระบบโพรดักชัน จึงเท่ากับภาษาคอมพิวเตอร์อื่นๆ เช่น ภาษาปาสคาล ภาษาเบสิก ภาษาฟอร์แทรน ฯลฯ ในทำนองเดียวกัน โปรแกรมที่เขียนด้วยภาษาคอมพิวเตอร์เหล่านี้ สามารถแทนด้วยระบบโพรดักชัน

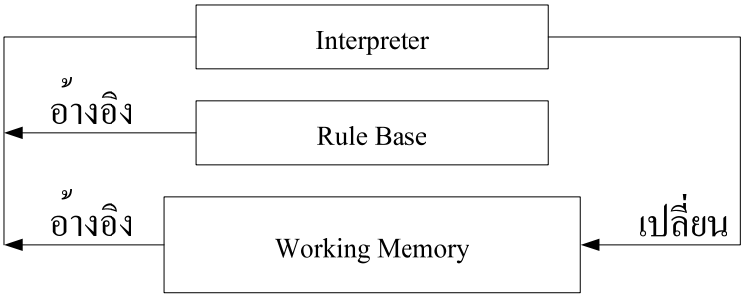
กฎในระบบโพรดักชัน จะอยู่ในรูป IF…THEN… โดยส่วนของ IF เรียกว่าส่วนเงื่อนไข และส่วนของ THEN เรียกว่า ส่วนข้อสรุปหรือส่วนการปฏิบัติ

3.7.1 โครงสร้างของระบบโพรดักชันจะประกอบด้วย 3 ส่วนด้วยกันคือ

3.7.1.1 ฐานกฎ (Rule base: RB) หรือ Production memory (PM)

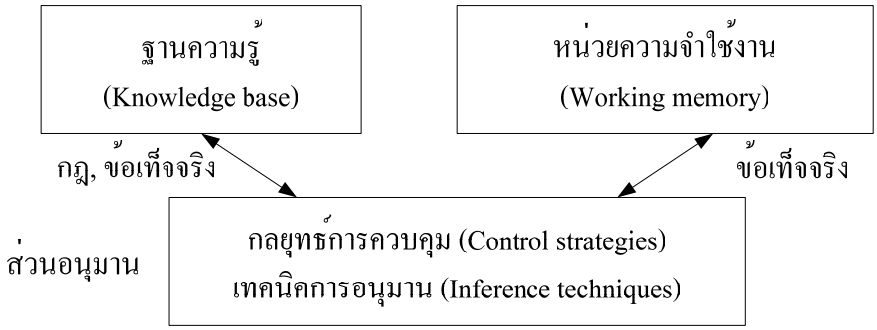
3.7.1.2 ส่วนตีความ (Interpreter) หรือส่วนอนุมาน

3.7.1.3 หน่วยความจำใช้งาน (Working memory: WM) หรือ Global database




รูป 3.2 โครงสร้างระบบโพรดักชัน [23]

ฐานกฎ (Rule base) เป็นฐานความรู้ (Knowledge base) ที่เก็บความรู้ที่อยู่ในรูปของกฎ ความจำใช้งาน (Working memory) เป็นที่เก็บข้อมูล และสถานะของระบบโพรดักชัน ข้อมูล และสถานะในความจำใช้งาน เป็นข้อมูลป้อนเข้าของส่วน IF ของกฎ จะถูกอ้างอิง และเปลี่ยนแปลงโดยกฎในฐานกฎ ส่วน IF นั้นจะตรวจดูเนื้อหาในฐานกฎ และความจำใช้งาน แล้วจะเลือกกฎใดกฎหนึ่งจากเซตของกฎที่มีเงื่อนไขครบขึ้นมาปฏิบัติการ




รูป 3.3 กลไกการอนุมานความรู้ในระบบผู้เชี่ยวชาญ

### 3.8 การออกแบบกลไกการอนุมานความรู้

ในระบบฐานกฎ ได้แบ่งกฎการอนุมานพื้นฐานที่ใช้ในการหาเหตุผล เป็น 4 แบบ ดังแสดงในตาราง 3.2

ตาราง 3.2 กฎการอนุมานพื้นฐานที่ใช้ในระบบฐานกฎ

| กฎการอนุมาน | รูปแบบ |
|---|---|
| 1. กฎการแจงผลตามเหตุ (Modus ponens) หรือ การหาเหตุผลโดยตรง (Direct Reasoning) | ถ้า $X \rightarrow Y$ เป็นจริง, X เป็นจริง<br>สรุปได้ว่า: Y เป็นจริง |
| 2. กฎการแจงผลค้านเหตุ (Modus tollens) หรือ การหาเหตุผลโดยอ้อม (Indirect Reasoning) | ถ้า $X \rightarrow Y$ เป็นจริง, ~Y เป็นจริง<br>สรุปได้ว่า: ~X เป็นจริง |
| 3. กฎลูกโซ่ (Hypothetical Syllogism หรือ chain rule) | ถ้า $X \rightarrow Y$ เป็นจริง, ถ้า $Y \rightarrow Z$ เป็นจริง<br>สรุปได้ว่า: $X \rightarrow Z$ เป็นจริง |
| 4. กฎลูกโซ่แย้งสลับที่ (Law of contrapositive) | ถ้า $X \rightarrow Y$ เป็นจริง<br>สรุปได้ว่า: $\sim Y \rightarrow \sim X$ เป็นจริง |

### 3.9 เทคนิคการทำงานของกลไกการอนุมาน

กลไกการอนุมานของระบบผู้เชี่ยวชาญ ประกอบด้วยเทคนิคการทำงาน 2 อย่างดังนี้ คือ

3.9.1 เทคนิคการอนุมาน ซึ่งเป็นวิธีการหาเหตุผลโดยนำเอาความรู้ที่มีอยู่ในฐานความรู้และข้อเท็จจริงของปัญหาในหน่วยความจำใช้งานของระบบมาใช้ประกอบการอนุมาน

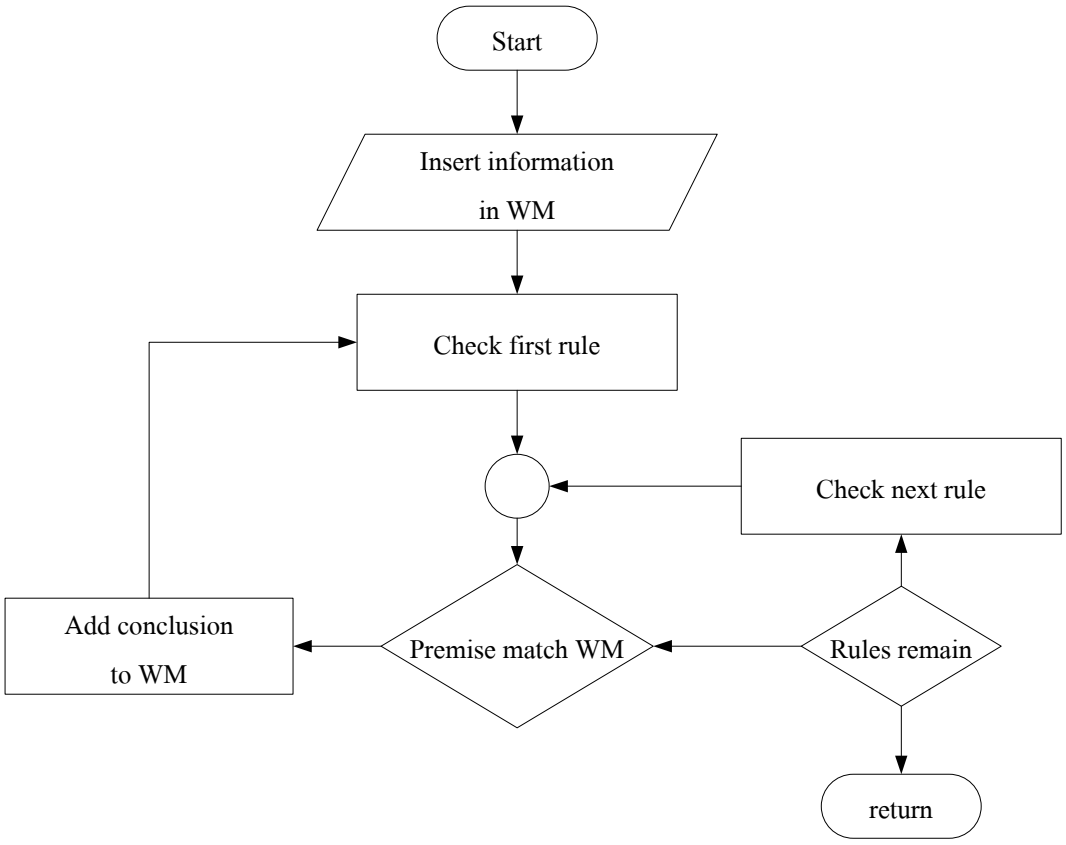
3.9.2 กลยุทธ์การควบคุม เป็นวิธีการกำหนดทิศทางการเข้าสู่กระบวนการหาเหตุผลของระบบโดยควบคุมการเริ่มต้นกระบวนการอนุมาน จากเป้าประสงค์ (Goal) หรือข้อเท็จจริงของปัญหา

### 3.10 การอนุมาน

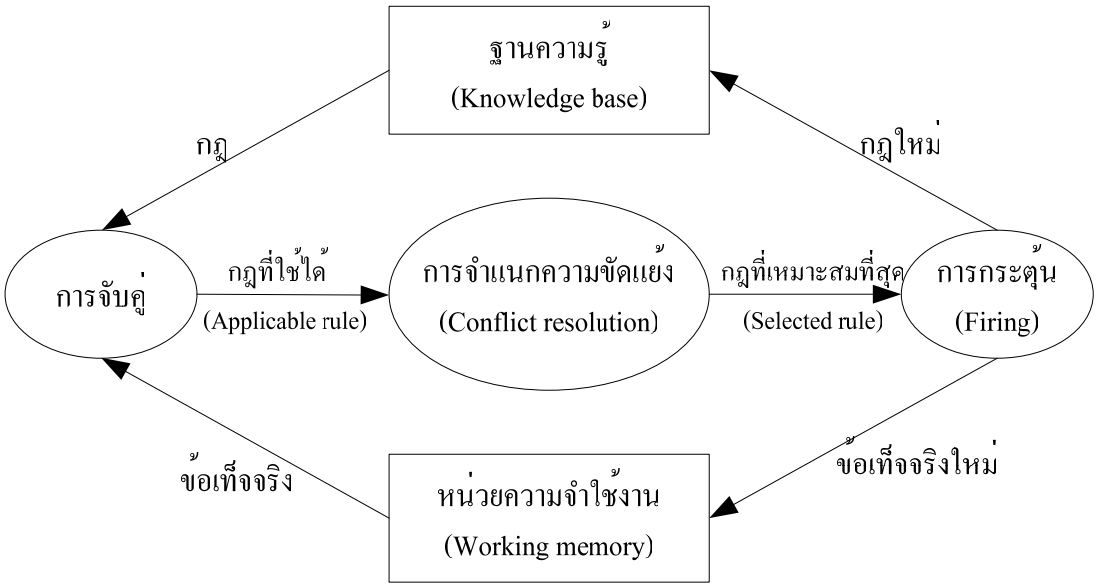
การอนุมาน ถือเป็นเทคนิค หรือวิธีการจำลองกระบวนการคิด และหาเหตุผลแบบมนุษย์ของระบบผู้เชี่ยวชาญ หรือเป็นวิธีการหาสารสนเทศใหม่ๆ จากสารสนเทศเดิมที่มีอยู่ในฐานความรู้ของระบบผู้เชี่ยวชาญ

### 3.10.1 การอนุมานเป็นห่วงโซ่แบบไปหน้า

การอนุมานเป็นห่วงโซ่แบบไปหน้า เป็นการอนุมานจากปัจจุบันไปสู่อนาคต โดยกระบวนการอนุมานจะเริ่มต้นจากข้อเท็จจริงของปัญหา และทำงานไปข้างหน้าเพื่อหาว่าคำตอบคืออะไร กลไกการอนุมานแบบนี้ เหมาะสมกับปัญหาที่ไม่ทราบคำตอบล่วงหน้า ในระบบผู้เชี่ยวชาญที่ใช้ระบบฐานกฎเป็นแบบจำลองของการแก้ปัญหา สามารถแสดงผังงานของขั้นตอนและกระบวนการอนุมาน ดังรูป 3.4 และ 3.5 ตามลำดับ (สามารถดูรายละเอียดเพิ่มเติมได้ที่ [25])



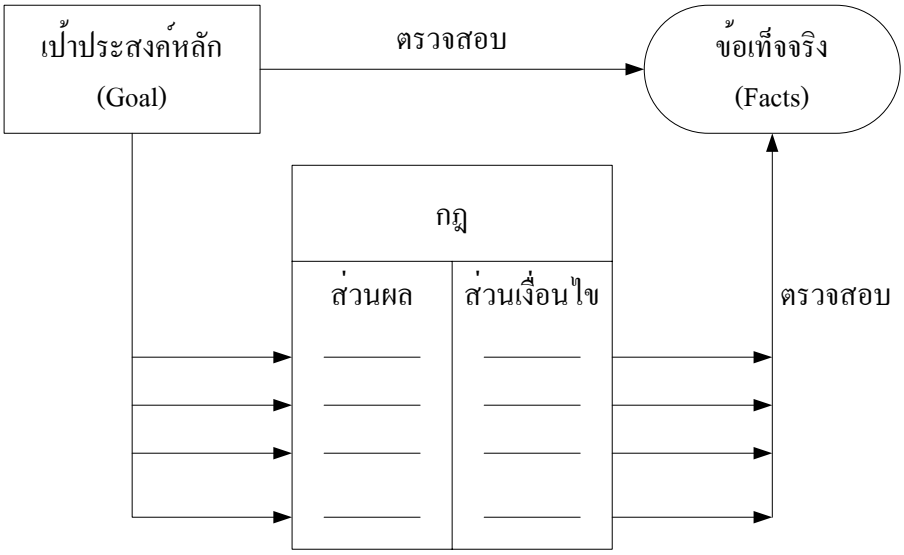

รูป 3.4 ผังงานขั้นตอนการอนุมานเป็นห่วงโซ่แบบไปหน้า




รูป 3.5 กระบวนการอนุมานเป็นห่วงโซ่แบบไปหน้า

### 3.10.2 การอนุมานเป็นห่วงโซ่แบบย้อนหลัง

การอนุมานเป็นห่วงโซ่แบบย้อนหลัง เป็นการอนุมานจากปัจจุบันย้อนกลับไปสู่อดีต โดยกระบวนการอนุมานจะเริ่มต้นจากเป้าประสงค์หลัก และทำงานย้อนหลังเพื่อหาข้อเท็จจริงมาสนับสนุนเป้าประสงค์หลักนั้น กลไกการอนุมานแบบนี้ เหมาะสมกับปัญหาที่ทราบคำตอบได้ล่วงหน้า ในระบบผู้เชี่ยวชาญที่ใช้ระบบฐานกฎเป็นแบบจำลองของการแก้ปัญหา สามารถแสดงกระบวนการอนุมาน ดังรูป 3.6




รูป 3.6 กระบวนการอนุมานเป็นห่วงโซ่แบบย้อนหลัง

### 3.11 หลักการออกแบบฐานความรู้

ความรู้ คือ ความเข้าใจในสาขาวิชาหนึ่งๆ โดยนักวิจัยสาขาปัญญาประดิษฐ์ได้จัดแบ่งความรู้ออกเป็น 2 ลักษณะคือ ความรู้ในแนวลึก (Deep knowledge) เช่น ความรู้จากตำรา หลักการ กฎเกณฑ์ หรือทฤษฎีต่างๆ และความรู้ในแนวกว้าง (Surface knowledge) เช่นความรู้จากประสบการณ์ของผู้เชี่ยวชาญมนุษย์ (Heuristic knowledge)

วิธีแปลงส่งความรู้เข้าสู่ฐานความรู้ของระบบผู้เชี่ยวชาญ สามารถทำได้ โดยการแทนความรู้หรือการเข้ารหัสความรู้ให้อยู่ในรูปที่เหมาะสมกับการนำไปประมวลผล หรือการอนุมาน การแทนความรู้ในระบบฐานกฎอยู่ในรูปของกฎและข้อเท็จจริงเท่านั้น

โครงสร้างของกฎ ประกอบด้วย 2 ส่วนคือ ส่วนเงื่อนไข (IF part) และส่วนผล (THEN part) ดังสมการ

$$\text{IF } A \text{ THEN } B \equiv A \rightarrow B \qquad (3.1)$$

โดยที่ A และ B แทนประพจน์ (Proposition) ซึ่งเป็นข้อความที่เป็นอนุประโยค (Clause) บอกเล่าหรือปฏิเสธ ที่มีค่าความจริง (Truth value) เป็นจริง หรือเป็นเท็จอย่างใดอย่างหนึ่ง หรือแทน ประพจน์ประกอบ (Compound statements) หรือข้อความที่ประกอบด้วยอนุประโยคตั้งแต่ 2 ประโยคขึ้นไป โดยแต่ละอนุประโยคเชื่อมต่อกันด้วยตัวเชื่อมทางตรรกะ (Connective) จากสมการ (3.1) ประพจน์ A เป็นอนุประโยคแสดงเงื่อนไข หลักฐาน หรือข้อพิสูจน์ที่นำมาเป็นเหตุ และประพจน์ B เป็นอนุประโยคแสดงการกระทำ ข้อสมมุติ หรือข้อสรุปที่เป็นผลตามมาของกฎนั้นๆ

3.11.1 รูปแบบของอนุประโยคที่เป็นองค์ประกอบอยู่ในเงื่อนไขหรือส่วนผลของกฎ โดยทั่วไปมีอยู่ 3 ชนิดคือ

3.11.1.1 อนุประโยคปฏิบัติการ (Executable clause) หรือ $E_c$ เป็นกระบวนคำสั่ง (Procedure) ที่สามารถปฏิบัติการอย่างใดอย่างหนึ่งได้ อนุประโยคชนิดนี้จะใช้ในส่วนผลของกฎโดยมีรูปแบบทั่วไป ดังสมการ

$$E_c = \{\text{procedure\_name; parameters}\} \qquad (3.2)$$

3.11.1.2 อนุประโยคทั่วไป (Regular clause) หรือ $R_c$ เป็นประโยคบอกเล่า หรือปฏิเสธ ที่มีโครงสร้างทั่วไป ดังสมการ

$$R_c = A \text{ is } B \qquad (3.3)$$

โดยที่ A แทนประธานของประโยค

B แทนกรรมของประโยค ซึ่งอาจเป็นคำ (Word) ตัวเลข หรือช่วงจำนวนจริง (Real interval)

3.11.1.3 อนุประโยคคณิตศาสตร์ (Mathematical clause) หรือ $M_c$ อนุประโยคชนิดนี้เป็นฟังก์ชันทางคณิตศาสตร์ที่มีอยู่ด้วยกัน 2 รูปแบบคือ นิพจน์เลขคณิต (Arithmetic expression) และ นิพจน์ทางตรรกะ (Logical expression)

1) นิพจน์เลขคณิต เป็นอนุประโยคที่ใช้ในส่วนผลของกฎ มีรูปแบบดังสมการ

$M_c$ = [ X : = math_expression ] (3.4)

2) นิพจน์ทางตรรกะ เป็นอนุประโยคที่ใช้ในส่วนเงื่อนไขของกฎ มีรูปแบบดังสมการ

$M_c$ = [ X <op> Y ] (3.5)

โดยที่

X หรือ Y แทนนิพจน์ทางคณิตศาสตร์

<op> แทนตัวดำเนินการทางตรรกะในเซต เช่น $<$, $>$, $\leq$, $\geq$, $=$ และ $\neq$ เป็นต้น

รูปแบบทั่วไปของฐานความรู้ที่ใช้การแทนความรู้ในรูปกฎ ประกอบด้วย รายการของกฎ (List of rule) ดังแสดงในตาราง 3.3 โดยที่กฎ IF A THEN B แทนกฎใดๆ และ $a_i$ และ $b_i$ แทนเงื่อนไขและข้อสรุปของกฎ ตามลำดับ

ตาราง 3.3 รายการของกฎ

| **เลขที่กฎ** | **กฎ** |
|---|---|
| 1 | IF $a_1$ THEN $b_1$ |
| 2 | IF $a_2$ THEN $b_2$ |
| 3 | IF $a_3$ THEN $b_3$ |
| . . . | . . . |
| I | IF $a_i$ THEN $b_i$ |
| . . . | . . . |
| n | IF $a_n$ THEN $b_n$ |

## 3.12 การค้นหา (Search)

ก่อนที่จะทำการค้นหาข้อมูล ข่าวสาร ความรู้ในระบบคอมพิวเตอร์ จะต้องทำการจัดเก็บข้อมูล ข่าวสาร หรือความรู้ไว้ และต้องเก็บอย่างเป็นระบบ มีระเบียบ มีหลักการเก็บที่ดี จึงจะทำให้การค้นหาทำได้ง่าย และรวดเร็ว

3.12.1 รูปแบบการจัดเก็บข้อมูล ข่าวสาร และความรู้

3.12.1.1 แบบรายการ (List)

3.12.1.2 แบบวางซ้อน (Stack)

3.12.1.3 แบบคิว (Queue)

3.12.1.4 แบบคิวชนิดวงรอบ

3.12.1.5 แบบต้นไม้ (Tree)

ดูรายละเอียดเพิ่มเติมได้ที่ [24]

3.12.2 การค้นหาโดยอ้างอิงข้อมูล

ในการค้นหาข้อมูลจะคำนึงถึงขั้นตอนวิธีทางคอมพิวเตอร์และหน่วยความจำ ซึ่งจะมีผลกับการใช้เวลาในการค้นหา การค้นหาโดยอ้างอิงข้อมูล แบ่งออกเป็น 4 แบบดังนี้

3.12.2.1 การค้นหาแบบลึกก่อน (Depth – first search)

การค้นหาแบบลึกก่อนเป็นการค้นหาที่ไม่ใช้ข้อมูล กฎเกณฑ์ตายตัว ในการค้นหาคำตอบ คือ จากโหนด (Node) ในต้นไม้การค้นหาให้เลือกอาร์ค (Arc) ที่อยู่ซ้ายสุดก่อนแล้วเลื่อนตัวไปยังโหนดที่อยู่ใต้ไปหนึ่งระดับ ทำเช่นนี้จนกว่าจะพบโหนดที่เป็นคำตอบ หรือลงไปจนถึงระดับล่างสุดที่กำหนดไว้ แล้วจึงย้อนกลับ (backtrack) ขึ้นมาหนึ่งระดับ เพื่อลองอาร์คที่อยู่ขวามือถัดไป ทำไปเช่นนี้จนกว่าจะพบคำตอบ

ขอแสดงตัวอย่าง กำหนดให้ระดับความลึกในการค้นหาให้ถึงระดับ 2 เมื่อ

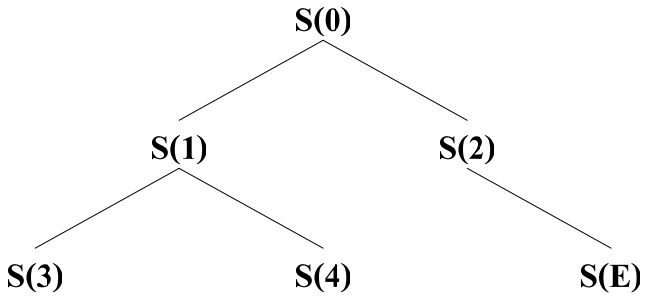
S(0) คือ โหนดเริ่มต้น

S(E) คือ โหนดเป้าหมาย

การค้นหาแบบลึกก่อนมีลำดับการตรวจสอบโหนดดังนี้

$S(0) \rightarrow S(1) \rightarrow S(3) \rightarrow S(1) \rightarrow S(4) \rightarrow S(1) \rightarrow S(0) \rightarrow S(2) \rightarrow S(E)$

ตามที่แสดงในรูป 3.7 โหนดที่มีการย้อนกลับคือ S(1) จำนวน 1 ครั้ง




รูป 3.7 การค้นหาแบบลึกก่อน

1) ขั้นตอนวิธีการค้นหาแบบลึกก่อน มีดังนี้

1.1) ใส่โหนดเริ่มต้น S(0) ไว้ในคิว (Queue)

1.2) ถ้าคิวว่างให้หยุดการค้นหาและแจ้งบอกความล้มเหลวในการค้นหา

1.3) ถ้าสมาชิกแรกสุดของคิวคือโหนดเป้าหมาย S(E) ให้หยุดการค้นหาและแจ้งบอกความสำเร็จในการค้นหา ถ้าไม่ใช่ให้ไปที่ขั้นตอนที่ 1.4)

1.4) เอาสมาชิกแรกสุดออกจากคิว แล้วเอาโหนดลูกของโหนดแรกสุดนั้น ที่ได้จากทุกทางเลือกไปใส่ไว้ข้างบนของคิวแทน

1.5) ย้อนกลับไปที่ขั้นตอนที่ 1.2)

2) ข้อดีของการค้นหาแบบลึกก่อน ได้แก่

2.1) ใช้เนื้อที่หน่วยความจำน้อย เนื่องจากการค้นหาแบบลึกก่อน จำเป็นต้องเก็บข้อมูลเกี่ยวกับโหนดเพียงเท่ากับจำนวนความลึกของการค้นหาเท่านั้น เก็บเพียงเท่านี้ก็สามารถทำการย้อนกลับ จากตัวอย่าง มีความจำเป็นต้องเก็บข้อมูลเกี่ยวกับโหนด เพียง 3 โหนด เท่านั้นเมื่อเปรียบเทียบ จำนวนความลึกของปัญหา จะน้อยกว่าจำนวนโหนดทั้งหมดมาก

2.2) การค้นหาแบบลึกก่อน อาจจะมีโอกาสพบคำตอบโดยไม่ต้องตรวจสอบโหนดจำนวนมากได้ ทำให้ประหยัดเวลาค้นหา คุณลักษณะนี้จะโดดเด่นมากขึ้นในปัญหาที่มีคำตอบอยู่จำนวนมาก และเราต้องการเพียงคำตอบเดียว การค้นหาแบบลึกก่อนสามารถหยุดได้ทันที เมื่อพบคำตอบใดคำตอบหนึ่ง

3) ข้อเสียของการค้นหาแบบลึกก่อน ได้แก่

การค้นหา อาจจะเป็นไปอย่างไม่สิ้นสุด ในกรณีที่มีวงวน (Loop) ในต้นไม้การค้นหา หรือในกรณีที่ไม่มีการกำหนดความลึกของการค้นหาไว้ ในทั้งสองกรณีเราอาจจะเสียเวลาค้นหาโดยไม่ได้คำตอบ

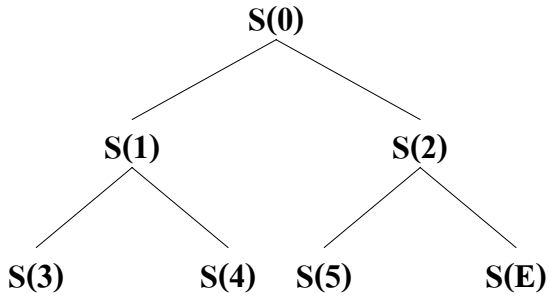
3.12.2.2 การค้นหาแบบกว้างก่อน (Breadth – first search)

การค้นหาแบบกว้างก่อนเป็นการค้นหาที่ไม่ใช้ข้อมูล กฎเกณฑ์ตายตัว โดยเริ่มจากโหนดเริ่มต้นให้สร้างทางเลือกหรืออาร์คทุกทางที่เป็นไปได้ แล้วทำการตรวจสอบดูว่าโหนดที่เกิดจากทางเลือกหรืออาร์คที่สร้างขึ้นนั้นเป็นคำตอบหรือโหนดเป้าหมายหรือไม่ ถ้าเป็นก็หยุดการค้นหา แต่ถ้าไม่เป็นก็สร้างอาร์คจากทุกโหนดที่ตรวจสอบแล้วลงไปอีกระดับหนึ่ง แล้วทำการตรวจสอบอีก ทำเช่นนี้จนกว่าจะพบโหนดเป้าหมาย หรือจนกระทั่งหมดโหนดค้นหา

การค้นหาแบบกว้างก่อนมีลำดับการตรวจสอบโหนดดังนี้

S(0)→ S(1)→ S(2)→ S(3)→ S(4)→ S(5)→ S(E)

ตามที่แสดงในรูป 3.8 โหนดที่มีการย้อนกลับคือ S(1) จำนวน 1 ครั้ง




รูป 3.8 การค้นหาแบบกว้างก่อน

1) ขั้นตอนวิธีการค้นหาแบบกว้างก่อน มีดังนี้
   1.1) ใส่โหนดเริ่มต้น S(0) ไว้ในคิว
   1.2) ถ้าคิวว่างให้หยุดการค้นหาและแจ้งบอกความล้มเหลวในการค้นหา
   1.3) ถ้าสมาชิกแรกสุดของคิวคือโหนดเป้าหมาย S(E) ให้หยุดการค้นหาและแจ้งบอกความสำเร็จในการค้นหา ถ้าไม่ใช่ให้ไปที่ขั้นตอนที่ 1.4)
   1.4) เอาสมาชิกแรกสุดออกจากคิว แล้วเอาโหนดลูกของโหนดแรกสุดนั้น ที่ได้จากทุกทางเลือกไปใส่ไว้ข้างท้ายของคิวแทน
   1.5) ย้อนกลับไปที่ขั้นตอนที่ 1.2)
2) ข้อดีของการค้นหาแบบกว้างก่อน ได้แก่
   2.1) การค้นหาแบบกว้างก่อน จะไม่มีการเข้าวงวน วนเวียนซ้ำไม่สิ้นสุด เพราะฉะนั้นเมื่อผ่านไปเวลาหนึ่งการค้นหาจะสิ้นสุดลง
   2.2) ถ้าในต้นไม้การค้นหามีคำตอบอยู่ การค้นหาแบบกว้างก่อนให้ประกันเราว่า เราจะพบคำตอบนั้น

3) ข้อเสียของการค้นหาแบบกว้างก่อน คือ

ใช้เนื้อที่ในหน่วยความจำมาก เนื่องจากต้องเก็บข้อมูลของทุกโหนดในระดับใดระดับหนึ่ง ยิ่งระดับการค้นหายิ่งลึกเท่าไหร่ จำนวนโหนดที่ต้องเก็บก็ยิ่งมากขึ้นเท่านั้น

สมมุติว่าแต่ละโหนดมีทางเลือก หรืออาร์ค ออกมาเท่ากับ A ในระดับที่ 0 จะมีโหนดเริ่มต้นเพียงโหนดเดียว แต่ในระดับที่ 1 จะมี A โหนด ระดับที่ 2 จะมี $A^2$ โหนด ระดับที่ 3 จะมี $A^3$ โหนด และระดับที่ n จะมี $A^n$

3.12.2.3 การค้นหาแบบดีที่สุดก่อน (Best – first search)

การค้นหาแบบดีที่สุดก่อน เป็นการค้นหาที่ใช้ข้อมูลช่วยประกอบในการเลือกทางเลือกต่อไป ข้อมูลที่ใช้คือ คุณค่าของแต่ละโหนด การค้นหาแบบดีที่สุดเอาคุณค่าของแต่ละโหนดไปผสมกับการค้นหาแบบกว้างก่อน โดยใช้คุณค่าของแต่ละโหนดเป็นตัวกำหนดลำดับในการตรวจสอบในแต่ละระดับความลึก แทนที่จะเป็นลำดับจากซ้ายไปขวา หรือจากโหนดแรก ไปโหนดหลัง ตามลำดับ การสร้างโหนด เหมือนกับการค้นหาแบบกว้างก่อน

การค้นหาแบบที่ดีที่สุดก่อนมักจะหาทางเลือกที่ดีที่นำไปสู่โหนดเป้าหมาย สิ่งสำคัญในการใช้การค้นหาแบบดีที่สุดก่อน คือการกำหนดวิธีการประเมินคุณค่าของแต่ละโหนด ถ้ากำหนดได้ดีก็จะทำให้การค้นหามีประสิทธิภาพ

ขั้นตอนวิธีการค้นหาแบบดีที่สุดก่อน มีดังนี้

1.1) ใส่โหนดเริ่มต้น S(0) ไว้ในคิว

1.2) ถ้าคิวว่างให้หยุดการค้นหาและแจ้งบอกความล้มเหลวในการค้นหา

1.3) ถ้าสมาชิกแรกสุดของคิวคือโหนดเป้าหมาย S(E) ให้หยุดการค้นหาและแจ้งบอกความสำเร็จในการค้นหา ถ้าไม่ใช่ให้ไปที่ขั้นตอนที่ 1.5)

1.4) เอาสมาชิกแรกสุดออกจากคิว จัดเรียงลำดับสมาชิกของคิวตามค่าของคุณค่าเพื่อให้ได้โหนด ที่มีค่าของคุณค่าดีที่สุดอยู่แรกสุดของคิว

1.5) ย้อนกลับไปที่ขั้นตอนที่ 1.3)

3.12.2.4 การค้นหาแบบ A*

การค้นหาแบบ A* (อ่านว่า เอ-สตาร์) เป็นการค้นหาที่ใช้ข้อมูลช่วยประกอบในการเลือกทางเลือกต่อไป ที่คล้ายกับการค้นหาแบบดีที่สุด กล่าวคือ ใช้การค้นหาแบบกว้างก่อนเป็นหลัก แต่แทนที่จะใช้ข้อมูลการประเมินคุณค่าของแต่ละโหนดอย่างเดียว เหมือนกับการค้นหาแบบดีที่สุดก่อน การค้นหาแบบ A* ใช้ข้อมูลเกี่ยวกับค่าใช้จ่ายในการกระทำที่เลือกประกอบด้วย ถ้าหากหน่วยที่ใช้ในการประเมินคุณค่าและในการคำนวณค่าใช้จ่ายเป็นหน่วยเดียวกัน หรือ สามารถแปลงให้เป็นหน่วยเดียวกันได้ เราจะสามารถรวมค่าของคุณค่าและค่าใช้จ่ายเข้าด้วยกัน แล้วใช้ค่ารวมนี้เป็นตัวชี้นำในการค้นหาแบบกว้างก่อน กล่าวคือ เราใช้ค่ารวมนี้ในการจัดลำดับของโหนด ในคิวที่รอการแตกแขนงโหนดย่อยต่อไป โหนดที่มีค่ารวมดีที่สุดจะอยู่ที่หัวคิว การค้นหาแบบ A* ก็คล้ายกับของแบบดีที่สุดก่อน เพียงแต่เปลี่ยนจากคุณค่ามาเป็นค่ารวมของคุณค่ากับค่าใช้จ่าย