

ប៊ូលូរាប់បច្ចុប្បន្ន

ARTIFICIAL INTELLIGENCE

លេកសារគោលនៅវិទ្យាល័យ 2110654

បុណ្យសេរីម កិចិត្រុក្តុល
ភាគវិទ្យាពាណិជ្ជកម្មគម្រោង
ជូនុយក្រសួងអប់រំ



ปัญญาประดิษฐ์

Artificial Intelligence

เอกสารคำสอนวิชา 2110654

บุญเสริม กิจศิริกุล
ภาควิชาวิศวกรรมคอมพิวเตอร์
จุฬาลงกรณ์มหาวิทยาลัย

แด่... นายหลี กิจศิริกุลและนางพเยาว์ แซ่เตี้ยว

เตี้ยและแม่ผู้เป็นสุดที่รัก

คำนำ

ทพ.อรรถพร ลิมปัญญาเลิศ นิสิตปริญญาโทเมื่อหลายปีก่อนที่ลงเรียนวิชาปัญญาประดิษฐ์ (เดิมเปิดสอนในรายวิชาชื่อ Directed Studies in CS.) หัวเครื่องคอมพิวเตอร์โน้ตบุ๊กเข้ามาเรียนวิชานี้ทุกสัปดาห์ แล้วจดบันทึกคำสอนลงในเครื่องได้อย่างรวดเร็วไม่น่าเชื่อ การบันทึกคำสอนของอรรถพรเป็นจุดเริ่มต้นของการเขียนเอกสารคำสอนเล่มนี้

เอกสารคำสอนนี้ใช้ในการเรียนการสอนวิชาปัญญาประดิษฐ์ (2110654) เนื้อหาครอบคลุมปัญญาประดิษฐ์เบื้องต้น ปรัชญาสตานะและการค้นหา การแทนความรู้โดยตรรกะเพรดิคเตต โปรดักอกเบื้องต้น การประมวลผลภาษาธรรมชาติ และการเรียนรู้ของเครื่อง เมื่อเรียนแต่ละบทแล้ว นิสิตสามารถฝึกทำแบบฝึกหัดท้ายบท ซึ่งบางส่วนถูกนำมาใช้เป็นการบ้านในรายวิชานี้ นอกจากนั้นนิสิตที่ต้องการเรียนรู้เพิ่มในหัวข้อที่สนใจ สามารถศึกษาเพิ่มเติมในหนังสือหรือตำราที่แนะนำไว้ที่ท้ายบท

ไฟล์ของเอกสารคำสอนเล่มนี้อยู่ในรูปแบบของพีดีเอฟสามารถดาวน์โหลดไฟล์ได้ที่

www.cp.eng.chula.ac.th/~boonserm/teaching/artificial.html

ในเอกสารนี้ใช้ตัวอักษรสีน้ำเงิน เช่น "... ดูรูปที่ 2-8" แสดงไฮเปอร์ลิงค์ไปยังตำแหน่งที่อ้างถึง เพื่อให้ผู้อ่านสามารถกระโดดไปยังตำแหน่งที่อ้างได้อย่างสะดวก และใช้ตัวอักษรสีแดงเพื่อแสดงถึงคำพที่ใหม่ หากผู้อ่านพบข้อผิดพลาดใดๆ หรือมีข้อเสนอแนะใดๆ เกี่ยวกับเอกสารเล่มนี้ กรุณาส่งข้อความทางอีเมลมาได้ที่ boonserm.k@chula.ac.th

เอกสารคำสอนเล่มนี้สำเร็จลุล่วงไปได้ด้วยความเห็นอุปถัมภ์ ความมุ่งมั่น ความเสียสละ และความรักจากบรรณาธิการและลูกๆ ที่รักทุกคน ภรรยาต้องช่วยดูแลลูกที่ยังเล็ก โดยให้สามีมีเวลาทำงานเขียนเอกสาร ต้องอดทนต่อความเห็นด้วยกันที่ต้องทำงานอยู่เวรจนค่ำ และยังมีภาระดูแลสมาชิกทุกคนในครอบครัว และขอใจลุกรักทั้งสองที่ต้อง (จำ) ยอมให้พ่อเมียเวลาทำงาน ไม่ได้เล่นสนุกด้วยกัน ไม่ได้รับการสอนหนังสืออย่างเต็มที่ในเวลาที่ควรได้อย่างยิ่ง

บุญเสริม กิจศิริกุล
ภาควิชาวิศวกรรมคอมพิวเตอร์
จุฬาลงกรณ์มหาวิทยาลัย

12 ธันวาคม 2546

ประมวลรายวิชา

รหัสวิชา	2110654
จำนวนหน่วยกิต	3 หน่วยกิต
ชื่อวิชา	ปัญญาประดิษฐ์
ภาควิชา	วิศวกรรมคอมพิวเตอร์
ภาคการศึกษา	ต้น
ปีการศึกษา	2546
ชื่อผู้สอน	ผศ.ดร.บุญเสริม กิจศิริกุล
เงื่อนไขรายวิชา	—
สถานภาพรายวิชา	วิชาเลือก
ชื่อหลักสูตร	วิทยาศาสตรมหาบัณฑิต
วิชาระดับ	ปริญญาโท
จำนวนชั่วโมงที่สอนต่อสัปดาห์	บรรยาย 3 ชั่วโมง

เนื้อหารายวิชา

นิยามของปัญญาประดิษฐ์ การประยุกต์ทางปัญญาประดิษฐ์ ปรัชญาสถานะและการค้นหา การแทนความรู้โดยตรรกะเพรดิเกต โปรดล็อกเบื้องต้น การประมวลผลภาษาธรรมชาติ การเรียนรู้ของเครื่อง

ประมวลการเรียนรายวิชา

วัตถุประสงค์ทั่วไป

- เพื่อให้นิสิตสามารถเข้าใจและอธิบายถึงปัญญาประดิษฐ์และการประยุกต์ใช้งาน
- เพื่อให้นิสิตสามารถอธิบายการแทนความรู้และนำไปประยุกต์ใช้งานได้
- เพื่อให้นิสิตสามารถเขียนโปรแกรมภาษาโปรดล็อกได้
- เพื่อให้นิสิตสามารถอธิบายถึงเทคนิคการประมวลผลภาษาธรรมชาติและสามารถนำไปประยุกต์ใช้งานได้
- เพื่อให้นิสิตสามารถอธิบายถึงเทคนิคการเรียนรู้ของเครื่องและนำไปประยุกต์ใช้งานได้

เนื้อหารายวิชาโดยละเอียด

สัปดาห์ที่ 1: นิยามของปัญญาประดิษฐ์ การทดสอบทั่วไป ห้องจีน งานประยุกต์ทางปัญญาประดิษฐ์

สัปดาห์ที่ 2-3: ปริภูมิสถานะและการค้นหา การนิยามปัญหาในรูปของการค้นหาในปริภูมิสถานะ เทคนิคการค้นหาในปัญญาประดิษฐ์ การค้นหาแนววิ่งก่อน การค้นหาแนวลึกก่อน การค้นหาแบบอิริสติก อัลกอริทึมปีนเข้า อัลกอริทึมออบเนีย จำลอง อัลกอริทึมดีสุดก่อน อัลกอริทึม A* การค้นหาตาม

สัปดาห์ที่ 4-5: การแทนความรู้โดยตัวกราฟเพรดิเคต การแปลความหมายของสูตรอะตอม ตัวเชื่อมและตัวบ่งป्रีเมต กฎการอนุมาน การแทนค่าและการทำให้เท่ากัน รีโซลูชัน การปฏิเสธแบบรีโซลูชัน

สัปดาห์ที่ 6: ภาษาโปรแกรม องค์ประกอบของภาษาโปรแกรม การโปรแกรมแบบเรียกซ้ำ นิเสธและตัวตัด

สัปดาห์ที่ 7: สอบกлагาก

สัปดาห์ที่ 8-9: การประมวลผลภาษาธรรมชาติ ขั้นตอนในการเข้าใจภาษาธรรมชาติ การวิเคราะห์ทางภาษาโดยสัมพันธ์ ตัวแจงส่วนแบบบันลังล่าง ตัวแจงส่วนตารางไวยากรณ์ข่ายงานเปลี่ยนสถานะ

สัปดาห์ที่ 10: การเรียนรู้ของเครื่องเบื้องต้น ขั้นตอนวิธีเชิงพันธุกรรม

สัปดาห์ที่ 11: การเรียนรู้โดยการจำ การเรียนรู้โดยการวิเคราะห์ความต่าง

สัปดาห์ที่ 12: เวอร์ชันสเปช การเรียนรู้ต้นไม้ตัดสินใจ

สัปดาห์ที่ 13: การเรียนรู้โดยการอธิบาย ข่ายงานประสาทเทียม

สัปดาห์ที่ 14: การเรียนรู้แบบเบส

สัปดาห์ที่ 15: เทคนิคการเรียนรู้ของเครื่องอื่นๆ

สัปดาห์ที่ 16: สอบกлагาก

วิธีจัดการเรียนการสอน บรรยาย

สื่อการสอน แผ่นใส / กระดาษ / เครื่องฉาย

การวัดผล รายงาน 40% สอบกлагาก 30% สอบปลายภาค 30%

รายชื่อหนังสืออ่านประกอบ

หนังสือบังคับ Rich, E. and Knight, K. (1991) *Artificial Intelligence*. Second Edition, McGraw-Hill.

เว็บไซต์วิชา <http://www.cp.eng.chula.ac.th/boonserm/teaching/artificial.html>

สารบัญ

1. ปัญญาประดิษฐ์เบื้องต้น	1
1.1 นิยามของปัญญาประดิษฐ์	1
1.2 การทดสอบทั่วไป	2
1.3 ห้องจีน	3
1.4 งานประยุกต์ทางปัญญาประดิษฐ์	4
เอกสารอ่านเพิ่มเติม	5
2. ปรัชญาสถานะและการค้นหา	7
2.1 การนิยามปัญหาในรูปของการค้นหาในปรัชญาสถานะ	7
2.2 เทคนิคการค้นหาในปัญญาประดิษฐ์	10
2.3 การค้นหาแบบบอต	11
2.3.1 การค้นหาแนวกว้างก่อน	12
2.3.2 การค้นหาแนวลึกก่อน	13
2.4 การค้นหาแบบอิวาริสติก	16
2.4.1 ตัวอย่างของฟังก์ชันอิวาริสติก	17
ฟังก์ชันอิวาริสติก h1 สำหรับปัญหาโลโกของบล็อก	17
ฟังก์ชันอิวาริสติก h2 สำหรับปัญหาโลโกของบล็อก	19
ฟังก์ชันอิวาริสติกสำหรับปัญหา 8-Puzzle	20
2.4.2 อัลกอริทึมปีนเขา	22
2.4.3 อัลกอริทึมมองเห็นiyawjalon	25
2.4.4 อัลกอริทึมดีสุดก่อน	28
2.4.5 อัลกอริทึม A*	30

2.4.6 การค้นหาตามหน่วยความจำระยะสั้น	31
ตัวอย่างปัญหาต้นไม้ k กิ่งน้อยสุด	33
การเลือกคุณสมบัติที่ใช้กำหนดสถานภาพต้องห้ามในการสร้างต้นไม้	34
เกณฑ์แห่งความหวัง	35
หน่วยความจำระยะยาว	37
เอกสารอ่านเพิ่มเติมและแบบฝึกหัด	38
	41
3. การแทนความรู้โดยตรรกะเพรดิเกต	45
3.1 ตรรกะเพรดิเกต	45
การแปลความหมายของสูตรอะตอม	46
ตัวเชื่อมและตัวบ่งบริมาณ	47
3.2 กฎการอนุมาน	49
3.3 การแทนค่าและการทำให้เท่ากัน	50
3.4 รีชูลัชัน	54
รีชูลัชันของอนุประโยคพื้นฐาน	56
รีชูลัชันทั่วไป	57
การปฏิเสธแบบรีชูลัชัน	58
ตัวอย่างการทำรีชูลัชัน	59
เอกสารอ่านเพิ่มเติมและแบบฝึกหัด	60
4. โปรดีอกเบื้องต้น	63
4.1 องค์ประกอบของโปรดีอก	63
ข้อเท็จจริง	64
ค่าคงที่	66
ข้อคำถาม	66
ตัวแปร	66
พจน์และการแทนค่า	67
ตัวเชื่อม 'และ'	67

กฏ	67
ความหมายของกฏ 'P :- Q, R.'	68
การจับคู่และการย้อนรอย	69
ตัวอย่างปัญหาลิงกินกล้วย	71
4.2 การโปรแกรมแบบเรียกซ้ำ	73
4.2.1 โปรแกรมเรียกซ้ำกับตัวเลข	74
โปรแกรมจำนวนธรรมชาติ	74
โปรแกรมบวกจำนวนธรรมชาติ	75
โปรแกรมคูณจำนวนธรรมชาติ	76
4.2.2 โปรแกรมเรียกซ้ำกับรายการ	77
โปรแกรมภาวะสมาชิก	78
โปรแกรมต่อรายการ	79
4.3 นิเสธและตัวตัด	80
4.3.1 นิเสธ	80
4.3.2 ตัวตัด	81
การใช้ตัวตัดร่วมกับเพรดิเคต 'fail'	83
ตัวตัดเขียว	83
ตัวตัดแดง	86
เอกสารอ่านเพิ่มเติมและแบบฝึกหัด	88
5. การประมวลผลภาษาธรรมชาติ	91
5.1 ขั้นตอนในการเข้าใจภาษาธรรมชาติ	92
ตัวอย่างการประมวลผลภาษาธรรมชาติ	93
5.2 การวิเคราะห์ทางภาษาสัมพันธ์	96
5.3 ตัวแจงส่วนแบบบูลจังส่วน	97
อัลกอริทึมของตัวแจงส่วนแบบบูลจังส่วนอย่างง่าย	99
5.4 ตัวแจงส่วนตาราง	102
ตัวอย่างของการแจงส่วนตาราง	105
5.5 ไวยากรณ์ข่ายงานเปลี่ยนสถานะ	110

อัลกอริทึมแจงส่วนบุนงล่างสำหรับอาร์ทีเอ็น	111
ตัวอย่างการแจงส่วนด้วยอาร์ทีเอ็น	113
เอกสารอ่านเพิ่มเติมและแบบฝึกหัด	116
6. การเรียนรู้ของเครื่อง	119
6.1 ขั้นตอนวิธีเชิงพัฒนารูป	119
โครงโฉมกำหนดลักษณะพิเศษที่สืบทอดได้	120
6.1.1 การออกแบบขั้นตอนวิธีเชิงพัฒนารูป	120
6.1.2 ค่าความหมายมาตรฐาน	122
6.1.3 การจำลองการคัดเลือกโดยธรรมชาติ	123
6.1.4 การไขว้เปลี่ยนเพื่อเอาชนะจุดดีสุดเฉพาะที่	126
6.1.5 ปรับปรุงจีเอด้วยฟังก์ชันความหมายแบบลำดับและการใช้ความหลากหลาย	128
ปรับปรุงจีเอด้วยฟังก์ชันความหมายแบบลำดับ	128
เพิ่มประสิทธิภาพจีเอให้สูงขึ้นโดยความหลากหลาย	129
6.2 การเรียนรู้โดยการจำ	136
6.3 การเรียนรู้โดยการวิเคราะห์ความแตกต่าง	141
6.4 เวอร์ชันสเปช	147
6.4.1 ตัวอย่างการเรียนรู้มโนทัศน์ car	150
6.4.2 ข้อจำกัดของเวอร์ชันสเปช	152
6.5 การเรียนรู้ต้นไม้ตัดสินใจ	153
6.5.1 ฟังก์ชันเกณฑ์สำหรับการเลือกบัพทดสอบ	157
6.5.2 ฟังก์ชันเกณฑ์	160
6.5.3 การเปลี่ยนต้นไม้เป็นกฎ	162
6.6 การเรียนรู้โดยการอธิบาย	164
ตัวอย่างการเรียนรู้มโนทัศน์ cup	166
6.7 ข่ายงานประสาทเทียม	169
6.7.1 เพอร์เซปตรอน	169
6.7.2 ตัวอย่างการเรียนฟังก์ชัน AND และ XOR ด้วยกฎเรียนรู้เพอร์เซปตรอน	174
6.7.3 ข่ายงานหลายชั้นและการแพร่กระจายย้อนกลับ	181

6.8 การเรียนรู้แบบเบส์	186
6.8.1 ทฤษฎีของเบส์	186
6.8.2 สูตรพื้นฐานของความน่าจะเป็น	189
6.8.3 การจำแนกประเภทที่น่าจะเป็นที่สุดสำหรับตัวอย่าง	189
6.8.4 ตัวจำแนกประเภทเบส์อย่างง่าย	190
การเรียนรู้เพื่อจำแนกประเภทข้อความโดยเบส์อย่างง่าย	193
6.8.5 ข่ายงานความเชื่อเบส์	195
6.8.6 การเรียนรู้ข่ายงานเบส์	200
การเรียนรู้ข่ายงานเบส์ในกรณีที่รู้โครงสร้างและข้อมูลครบ	200
การเรียนรู้ข่ายงานเบส์ในกรณีที่โครงสร้างรู้และข้อมูลมีค่าหาย	201
เอกสารอ่านเพิ่มเติมและแบบฝึกหัด	206
ดัชนีศัพท์	211
Index	214

ปัญญาประดิษฐ์เบื้องต้น

1

1.1 นิยามของปัญญาประดิษฐ์

ปัญญาประดิษฐ์ – เอไอ (Artificial Intelligence – AI) มีคำนิยามมากมาย นิยามที่ใช้ในที่นี่คือปัญญาประดิษฐ์เป็นวิชาที่ว่าด้วยการศึกษาเพื่อให้เข้าใจถึงความฉลาดและสร้างระบบคอมพิวเตอร์ที่ชาญฉลาด และนำมาทำงานแทนหรือช่วยมนุษย์ทำงานที่ต้องใช้ความฉลาดนั้นๆ

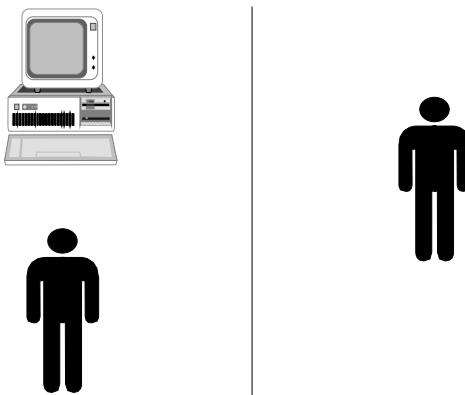
อย่างไรก็ดีงานบางอย่างที่ปัจจุบันเครื่องคอมพิวเตอร์ทำได้ดีกว่าอยู่แล้ว อย่างเช่นการคำนวณทางตัวเลข บวกหรือลบเลข ฯลฯ จะไม่อู้ในความสนใจของสาขานี้ โดยจะเน้นที่งานที่มนุษย์ทำได้ดีกว่าและงานนั้นๆ เกี่ยวข้องกับความฉลาด (intelligence) ด้วยพฤติกรรมที่แสดงความฉลาดของมนุษย์มีมากมาย ยกตัวอย่างดังด้านล่างนี้

- การเรียนรู้และเข้าใจจากประสบการณ์
- การตอบสนองต่อข้อความที่คุณมีความต้องการ
- ความสามารถที่จะตอบสนองต่อสถานการณ์ใหม่ๆ ได้อย่างรวดเร็วและประสบผลสำเร็จ
- ความสามารถให้เหตุผลในการแก้ไขปัญหาได้อย่างมีประสิทธิภาพ
- ความสามารถจัดการและแก้ไขสถานการณ์ที่ซับซ้อนได้
- ความสามารถที่จะเข้าใจและทำงานได้ในทิศทางที่ถูกต้อง
- ความสามารถที่จะใช้ความรู้เพื่อจัดการกับสภาพแวดล้อมได้
- ความสามารถที่จะหาความรู้และใช้ความรู้นั้นได้
- ความสามารถที่จะคิดและใช้เหตุผล

ระบบเอไอที่เราต้องการพัฒนาขึ้นนี้ เรา มีจุดมุ่งหมายว่าต้องทำงานได้ดีกว่ามนุษย์ เรา แล้วบังต้องทำงานที่เกี่ยวข้องกับความฉลาดด้านนิดนึง คำถามคือว่าเมื่อเราสร้างระบบ ขึ้นมาระบบหนึ่งแล้ว จะทราบได้อย่างไรว่าระบบนี้สามารถเรียกว่าระบบเอไอได้หรือไม่ หรือ เป็นระบบเอไอหรือไม่?

1.2 การทดสอบทั่วเริง

ในปี 2493 อัลัน ทั่วเริง (Alan Turing) ได้เสนอวิธีการในการทดสอบสิ่งที่เรียกว่า ปัญญาประดิษฐ์ขึ้น วิธีการทดสอบนี้มีชื่อเรียกว่า **การทดสอบทั่วเริง (Turing test)** มี จุดมุ่งหมายเพื่อแยกแยะระหว่างเครื่องคอมพิวเตอร์กับคน ถ้าหากว่าเราแยกระหว่างเครื่อง กับคนไม่ได้ก็ถือว่าระบบนั้นเป็นระบบปัญญาประดิษฐ์ วิธีการนี้อาศัยคนสองคนและระบบที่ จะทดสอบ โดยที่คนหนึ่งจะทำหน้าที่เป็นผู้ซักถาม และผู้ซักถามนี้จะถูกแยกอยู่คนละห้อง กับคนอีกคนหนึ่งและระบบที่จะทดสอบดังแสดงในรูปที่ 1-1



รูปที่ 1-1 การทดสอบทั่วเริง

ผู้ซักถามจะต้องตั้งคำถามเพื่อถามระบบหรือคนก็ได้ โดยที่คนถามจะไม่ทราบว่าห้องใด เป็นระบบหรือคน อาจจะทราบเพียงแต่ว่าเป็น A กับ B เท่านั้น หน้าที่ของผู้ซักถามคือ จะต้องถามทั้ง A และ B เพื่อจำแนกให้ได้ว่า A และ B ใครเป็นระบบ ใครเป็นคน ถ้าระบบที่ นำเข้ามาทำการทดสอบสามารถทำให้ผู้ซักถามเข้าใจว่าเป็นคน ระบบนั้นก็จะถูกพิจารณาได้ ว่ามีความสามารถในการพูดคุยเข้าใจข้อซักถาม คิดเหตุผลและเป็นระบบเอไอ

1.3 ห้องจีน

ห้องจีน (Chinese room) เป็นปัญหาทางปรัชญาในเรื่องปัญหาหนึ่งโดยปัญหานี้โดยปัญหามีอยู่ว่า มีห้องๆ หนึ่งตั้งอยู่ในประเทศจีนและมีคนนั่งอยู่ในห้องนั้น 1 คน คนนี้พูดภาษาอังกฤษได้เพียงภาษาเดียว รอบๆ ผาผนังในห้องจะมีอักษรจีนเป็นคู่ๆ แบบไร้โดยรอบดังรูปที่ 1-2

僕	→	貴方
行	→	來

⋮

รูปที่ 1-2 ชุดคู่ตัวอักษรในห้องจีน

ถ้ามีคนเงินที่อยู่นอกห้องยกแผ่นป้ายแผ่นหนึ่งมา คนอังกฤษคนนั้นจะต้องเทียบแผ่นป้ายนั้นกับแผ่นป้ายด้านซ้ายของคู่ตัวอักษรในห้อง และยกแผ่นป้ายด้านขวาของคู่อักษรที่สอดคล้องที่แบบไร้ข้างฝาออกมายื่นให้คนเงินที่อยู่นอกห้อง การทำงานของคนอังกฤษนี้บอกถึงการแก้ไขปัญหาและตอบคำถามของคนเงิน คนเงินข้างนอกอาจถามคำถามใดๆ ก็ได้ การพูดคุยกันโดยผ่านแผ่นป้ายถ้าสามารถดำเนินไปได้ด้วยดี ถ้ามีคนอังกฤษที่นั่งอยู่ในห้อง จะลากหรือไม่? หรือเขารู้ภาษาจีนหรือไม่? อาจไม่ทั้งสองอย่าง แต่อย่างไรก็ตามเขาก็สามารถที่จะให้คำตอบกลับไปได้และบางครั้งอาจให้ความรู้ใหม่ๆ กับคนถามก็ได้ ระบบทางเรื่องก็มีลักษณะคล้ายกับห้องจีนนี้เช่นกันโดยตัวระบบเองอาจไม่ได้เข้าใจสิ่งที่พูดคุยกัน แต่พฤติกรรมที่ระบบแสดงออกมาอาจเป็นพฤติกรรมที่แสดงถึงความฉลาดคล้ายกับมนุษย์ได้

1.4 งานประยุกต์ทางปัญญาประดิษฐ์

เทคนิคทางปัญญาประดิษฐ์ได้ถูกนำไปใช้ในงานประยุกต์ต่างๆ มากมาย ดังต่อไปนี้

- การประมวลผลภาษาธรรมชาติ (natural language processing) เป็นการประมวลผลข้อความในภาษาธรรมชาติหรือภาษามนุษย์ ใช้ในงานประยุกต์อย่างเช่นการทำความเข้าใจข้อความด้วยคอมพิวเตอร์ การแปลภาษาจากภาษาอังกฤษเป็นภาษาไทยด้วยคอมพิวเตอร์ เป็นต้น
- การทำเหมืองข้อมูล (data mining) เป็นการประยุกต์ใช้เทคนิคทางปัญญาประดิษฐ์เพื่อการขุดค้นข้อมูลอย่างฉลาดจากฐานข้อมูลทำหน้าที่ดึงความรู้ที่แฝงอยู่ใน

ฐานข้อมูล ตัวอย่างเช่นธนาคารแห่งหนึ่งเก็บข้อมูลลูกค้าจำนวนมากไว้ในฐานข้อมูลเพื่อช่วยตัดสินใจในการออกบัตรเครดิตให้กับลูกค้า ซึ่งฐานข้อมูลดิบนั้น ทำความเข้าใจยากมากว่าลูกค้าตัวกับไม่ได้ต่างกันอย่างไร การทำเหมือนข้อมูลสามารถชุดความสัมพันธ์ที่แฟ้มในฐานข้อมูลและสรุปคุณสมบัติของลูกค้าที่ได้ที่ ต่างจากลูกค้าไม่ได้ออกมาได้ และใช้ประโยชน์ต่อการอนุมัติบัตรให้กับลูกค้ารายอื่นๆ ในอนาคตเพื่อให้ผลกำไรของธนาคารสูงสุด เป็นต้น

- ระบบผู้เชี่ยวชาญ (expert system) เป็นระบบเอไอที่ทำหน้าที่เสมือนผู้เชี่ยวชาญเฉพาะด้าน เช่นผู้เชี่ยวชาญในการประกอบเครื่องคอมพิวเตอร์ตามที่ลูกค้าต้องการ ตัวอย่างที่มีชื่อเสียงของระบบนี้ก็เช่น MYCIN [Shortliffe, 1976] ซึ่งเป็นระบบผู้เชี่ยวชาญสมมิลแพรท์ทำหน้าที่วินิจฉัยโรคที่ติดเชื้อจากแบคทีเรียพร้อมทั้งบอกชื่อยาที่สอดคล้องกับการติดเชื้อด้วย พัฒนาขึ้นโดยเอ็ดวาร์ด ชอร์ตลิฟฟ์ (E. Shortliffe) สามารถวินิจฉัยโรคทางด้านอายุรกรรม (โรคที่รักษาทางยา) ได้อย่างมีประสิทธิภาพ เมื่อจากรับดังกล่าวแล้วได้บรรจุ “ความรู้” ในด้านนี้ไว้มากกว่าสายแพทย์ที่เป็นมนุษย์จะจำจำได้หมดและให้ผลที่ถูกต้องแม่นยำมาก
- การพิสูจน์ทฤษฎี (theorem proving) ซึ่งนับเป็นเรื่องที่ยากมาก ถ้าให้มนุษย์ทำแต่ความสามารถใช้เอไอทำได้ค่อนข้างดี
- วิทยาการหุ่นยนต์ (robotics) เป็นการทำหุ่นยนต์ให้ทำงานได้มีความคล่องสามารถตัดสินใจได้
- การโปรแกรมอัตโนมัติ (automatic programming) เป็นการเขียนโปรแกรมโดยอัตโนมัติ เช่นเราป้อนคุณลักษณะนิพุตต์กับเราต์พุตของโปรแกรมที่ต้องการ เพื่อแสดงว่าอินพุตแบบนี้เรามุ่งหวังว่าจะได้เราต์พุตอย่างไร โดยคุณลักษณะที่ป้อนเข้าไปมีจำนวนมากพอ แล้วระบบเอไอจะเขียนโปรแกรมที่ตรงกับคุณลักษณะอินพุตของเราต์พุตให้โดยอัตโนมัติ
- ปัญหาการจัดตาราง (scheduling problem) เทคนิคทางปัญญาประดิษฐ์ได้รับการประยุกต์ใช้กับปัญหาการจัดตารางเวลา เช่นการจัดตารางเวลาในสายการผลิตว่า จะทำอย่างไรให้เกิดประสิทธิภาพสูงสุด หรือจะจัดตารางการขึ้นลงของเครื่องบินอย่างไรให้เกิดประโยชน์สูงสุด เป็นต้น
- ปัญหาทางมโนธรรม (perception problem) เทคนิคทางปัญญาประดิษฐ์ได้ถูกนำไปใช้แก้ปัญหาเกี่ยวกับการมองเห็น การฟัง การได้ยิน เช่นเมื่อหุ่นยนต์มองจะทราบได้อย่างไรว่าอันนี้เป็นกล่อง อันนี้เป็นสิ่งกีดขวาง ฯลฯ

เอกสารอ่านเพิ่มเติม

หนังสือของ Russell และ Norvig [Russell & Norvig, 1995] ให้รายละเอียดเกี่ยวกับประวัติของปัญญาประดิษฐ์ไว้อย่างละเอียด คำบรรยายเกี่ยวกับการศึกษาวิจัยเรื่องความฉลาดและปัญหาทางปัญญาประดิษฐ์แสดงไว้อย่างดีใน [Pfeifer & Scheier, 1999] นอกจากนั้น หนังสือของ Luger [Luger, 2002] ก็ได้อธิบายถึงประวัติของปัญญาประดิษฐ์และการประยุกต์ใช้ปัญญาประดิษฐ์ได้ดี

บรรณานุกรม

- Luger, G. F. (2002) *Artificial Intelligence: Structures and Strategies for Complex Problem Solving (Fourth Edition)*, Addison Wesley.
- Pfeifer, R. and Scheier, C. (1999) *Understanding Intelligence*, The Mit Press.
- Russell, S. and Norvig, P. (1995) *Artificial Intelligence: A Modern Approach*, Prentice Hall.
- Shortliffe, E. H. (1976) *Computer-Based Medical Consultations: MYCIN*. Elsevier.

ปริภูมิสถานะและการค้นหา



การแก้ปัญหาส่วนใหญ่ในปัญญาประดิษฐ์ จะมองปัญหาในรูปของการค้นหา (search) งานหลายๆ อย่างในปัญญาประดิษฐ์ซึ่งจะกล่าวในบทต่อๆ ไป เช่นการเรียนรู้ของเครื่อง การประมวลผลภาษาธรรมชาติ ฯลฯ ใช้พื้นฐานของการค้นหาทั้งสิ้น ในการแก้ปัญหาโดยหลักการนี้ เราจะสร้างปริภูมิ (space) ขึ้นมาหนึ่งปริภูมิ สามารถแต่ละตัวในปริภูมินี้แทนตัวเลือกของคำตอบ จากนั้นก็ทำการค้นหาด้วยวิธีการค้นหาอย่างใดอย่างหนึ่งซึ่งจะกล่าวในบทนี้ เพื่อให้ได้คำตอบที่ต้องการ

สิ่งที่ต้องทำในการแก้ปัญหาหนึ่งๆ คือ

1. นิยามปัญหาอย่างชัดเจน หาสถานการณ์ปัจจุบัน (initial situation หรือ initial state) สถานการณ์สุดท้าย (final situation) ซึ่งเป็นคำตอบของปัญหา
2. วิเคราะห์ปัญหา
3. หาความรู้ที่ใช้ในการแก้ปัญหาว่ามีอะไรบ้าง
4. เลือกเทคนิคแก้ปัญหาที่เหมาะสม

2.1 การนิยามปัญหาในรูปของการค้นหาในปริภูมิสถานะ

ปัญหา

8-Puzzle

การค้นหาใน
ปริภูมิสถานะ

สมมติว่าเราต้องการแก้ปัญหา 8-Puzzle ซึ่งประกอบด้วยถาดขนาด 3×3 หน่วย ภายในถาดบรรจุแผ่นป้ายขนาด 1×1 หน่วยจำนวน 8 แผ่น ในแผ่นป้ายแต่ละแผ่นจะมีหมายเลขอกำกับอยู่ และมีช่องว่างอยู่ 1 ช่องที่แผ่นป้ายสามารถเคลื่อนเข้ามาแทนที่ได้ ปัญหาคือให้เลื่อนแผ่นป้ายเหล่านี้จากตำแหน่งเริ่มต้นให้เป็นตำแหน่งสุดท้ายซึ่งเป็นคำตอบ (ดูรูปที่ 2-1 ประกอบ) ในการนิยามปัญหาในรูปของการค้นหาในปริภูมิสถานะ (state space search) นั้นสามารถทำได้ดังนี้

8 ปัญหาประดิษฐ์

เวอร์ชัน 1.0.2: 15 มีค.2548 : 9:05 PM boonserm.k@chula.ac.th

2	4	1
8	5	6
3		7

สถานะเริ่มต้น

1	2	3
8		4
7	6	5

สถานะสุดท้าย

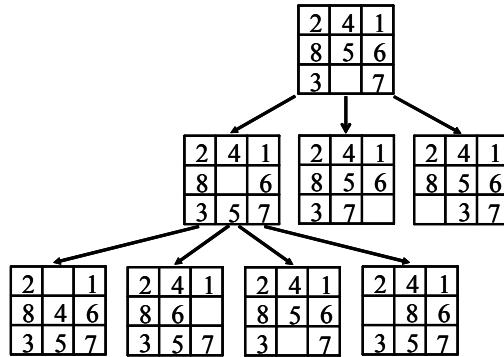
รูปที่ 2-1 8-Puzzle

สถานะเริ่มต้น
และ
สถานะสุดท้าย

ตัวกระทำการ

- นิยามปริภูมิสถานะโดยแสดงวัตถุทั้งหมดที่เกี่ยวข้องกับปัญหา ตัวอย่างเช่นในปัญหา 8-Puzzle เราอาจใช้รายการเพื่อแทนวัตถุเหล่านี้
- ให้ **สถานะเริ่มต้น (initial state)** แทนตำแหน่งเริ่มต้นของวัตถุทั้งหมดที่เกี่ยวข้องกับปัญหาและ **สถานะสุดท้ายหรือสถานะเป้าหมายหรือคำตอบ (final state or goal state or solution)** แทนตำแหน่งสุดท้าย เช่นตัวอย่างในปัญหา 8-Puzzle ให้สถานะเริ่มต้นเป็น $\{2,4,1,8,5,6,3,0,7\}$ และสถานะเป้าหมายเป็น $\{1,2,3,8,0,4,7,6,5\}$
- หากฎที่ใช้เปลี่ยนสถานะจากสถานะหนึ่งไปเป็นอีกสถานะหนึ่ง ซึ่งเราเรียกว่า **เหล่านี้ว่า ตัวกระทำการ (operator)** ในที่นี้ตัวกระทำการคือการเลื่อนแผ่นป้าย
- ใช้เทคนิคของการค้นหาในการเปลี่ยนจากสถานะเริ่มต้นไปยังสถานะเป้าหมาย ตัวอย่างตัวกระทำการในกรณีของ 8-Puzzle เป็นดังนี้
 - แผ่นป้ายหนึ่งๆ จะเลื่อนมาที่ซ่องว่างได้ถ้าอยู่ติดกับซองว่าง และสถานะจะเปลี่ยนจากสถานะเดิมไปยังสถานะใหม่ ซึ่งตำแหน่งของแผ่นป้ายสลับที่กับตำแหน่งของซ่องว่าง ตัวกระทำการมีทั้งหมด 4 ตัวดังนี้
 - ด้านบนของซ่องว่างมีแผ่นป้าย \rightarrow สลับตำแหน่งของซ่องว่างกับแผ่นป้าย
 - ด้านขวาของซ่องว่างมีแผ่นป้าย \rightarrow สลับตำแหน่งของซ่องว่างกับแผ่นป้าย
 - ด้านล่างของซ่องว่างมีแผ่นป้าย \rightarrow สลับตำแหน่งของซ่องว่างกับแผ่นป้าย
 - ด้านซ้ายของซ่องว่างมีแผ่นป้าย \rightarrow สลับตำแหน่งของซ่องว่างกับแผ่นป้าย

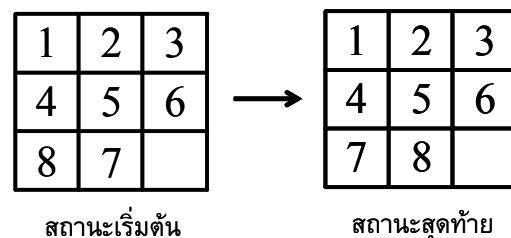
ตัวกระทำการที่ 1. มีความหมายว่า ถ้าด้านบนของซ่องว่างมีแผ่นป้าย ให้สลับตำแหน่งของซ่องว่างกับแผ่นป้าย และเกิดสถานะใหม่ซึ่งมีตำแหน่งของซ่องว่างกับแผ่นป้ายสลับที่กัน เมื่อเรา niyam ตัวกระทำการเหล่านี้แล้ว ปริภูมิสถานะก็จะเกิดขึ้นโดยปริยาย และเราจะเห็นปริภูมิสถานะว่าสถานะใดซึ่งมีตำแหน่งของซ่องว่างกับแผ่นป้ายสลับที่กัน แล้วใช้ตัวกระทำการเหล่านี้สร้างการเชื่อมต่อของสถานะ ในการกรณีของปัญหา 8-Puzzle เมื่อใช้ตัวกระทำการด้านบน เริ่มจากสถานะเริ่มต้นใน [รูปที่ 2-1](#) จะสร้างปริภูมิสถานะบางส่วน ดังแสดงใน [รูปที่ 2-2](#)



รูปที่ 2-2 บางส่วนของปริภูมิสถานะในปัญหา 8-Puzzle

จากรูปจะเห็นว่าการเชื่อมต่อของสถานะถูกกำหนดโดยตัวกระทำการ ดังนั้นในปัญหา หนึ่งๆ เมื่อเรานิยามตัวกระทำการได้แล้ว เรา ก็จะได้ปริภูมิของสถานะสำหรับปัญหานั้นๆ อย่างไรก็ได้ เราจะพบว่าปริภูมิสถานะโดยทั่วไปจะมีลักษณะเป็นโครงสร้างแบบกราฟ มากกว่าจะเป็นแบบตันไม้ เนื่องจากบางสถานะอาจเกิดซ้ำกันได้ เช่นในกรณีของตัวอย่างใน รูปที่ 2-2 จะเห็นได้ว่าสถานะที่ 3 (นับจากซ้าย) ที่แคลบล่างสุดซ้ำกับสถานะเริ่มต้น ซึ่งการ เป็นกราฟนี้เองทำให้การค้นหาในปริภูมิสถานะมีความยุ่งยากมากขึ้นกว่าปริภูมิที่เป็นแบบ ตันไม้

นอกจากนั้นสถานะเริ่มต้นตัวหนึ่งที่กำหนดให้ อาจไม่สามารถนำไปสู่สถานะเป้าหมาย บางตัวได้ เนื่องจากสถานะเป้าหมายนั้นไม่มีเส้นทางที่เชื่อมต่อจากสถานะเริ่มต้น ตัวอย่างเช่นในรูปที่ 2-3



รูปที่ 2-3 ตัวอย่างคำตอองที่เข้าถึงไม่ได้ด้วยสถานะเริ่มต้น

จากที่กล่าวข้างต้นจะเห็นได้ว่า การแก้ปัญหาพื้นฐานทางปัญญาประดิษฐ์แบบหนึ่งคือ การมองปัญหาในรูปแบบของการค้นหาในปริภูมิสถานะ โดยเริ่มจากการนิยามสถานะ นิยามโครงสร้างข้อมูลที่เก็บวัตถุที่เกี่ยวข้องกับปัญหา กำหนดสถานะเริ่มต้นและสถานะ

เป้าหมาย รวมทั้งหาตัวกระทำการว่าจะต้องใช้ตัวกระทำการอะไรบ้างในการนิยามสถานะ จากนั้นก็เป็นการเลือกเทคนิคการค้นหาที่เหมาะสมกับปัญหาที่เราต้องการ หัวข้อต่อไปนี้จะกล่าวถึงเทคนิคการค้นหาแบบต่างๆ ที่สามารถนำมาใช้ในการค้นหาในปริภูมิสถานะ

2.2 เทคนิคการค้นหาในปัญญาประดิษฐ์

เทคนิคการค้นหาสามารถแบ่งประเภทได้ดังต่อไปนี้

1. การค้นหาแบบบอต (blind search)
 - 1.1 การค้นหาทั่วหมด (exhaustive search)
 - 1.2 การค้นหาบางส่วน (partial search)
 - 1.2.1 การค้นหาแนวกว้างก่อน (breadth-first search)
 - 1.2.2 การค้นหาแนวลึกก่อน (depth-first search)
2. การค้นหาแบบอิวาริสติก (heuristic search)
 - 2.1 อัลกอริทึมปีนเข้า (hill-climbing algorithm)
 - 2.2 อัลกอริทึมขอบเหนี่ยวจัลลง (simulated annealing algorithm)
 - 2.3 การค้นหาดีสุดก่อน (best-first search)
 - 2.4 การค้นหา A* (A* search)
 - 2.5 อื่นๆ

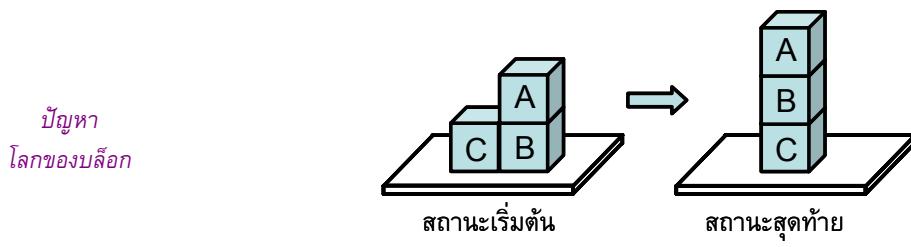
เทคนิคการค้นหาสามารถแบ่งได้เป็นสองประเภทหลักๆ คือ **การค้นหาแบบบอต (blind search)** ซึ่งเป็นเทคนิคการค้นหาที่ไม่มีตัวช่วยในการค้นหา แต่จะมีรูปแบบการค้นหาที่แน่นอนตามตัว เช่นจะค้นหาสถานะจากบนลงล่างในปริภูมิค้นหา เป็นต้น ส่วนเทคนิคการค้นหาอีกประเภทหนึ่งคือ **การค้นหาแบบอิวาริสติก (heuristic search)** ซึ่งจะใช้ความรู้รูปแบบหนึ่งที่เรียกว่าอิวาริสติกมาช่วยทำให้การค้นหามีประสิทธิภาพยิ่งขึ้น

การค้นหาแบบบอตสามารถแบ่งย่อยได้ดังนี้คือ **การค้นหาทั่วหมด (exhaustive search)** หมายถึงการค้นหาทั่วหมดทั่วทั้งปริภูมิสถานะและ **การค้นหาบางส่วน (partial search)** เป็นการค้นหาเพียงบางส่วนของปริภูมิสถานะ ปัญหาส่วนใหญ่ทางปัญญาประดิษฐ์มีปริภูมิสถานะที่มีขนาดใหญ่มาก ทำให้เราไม่สามารถค้นหาได้ทั่วทั้งปริภูมิ จำเป็นต้องค้นหาเพียงบางส่วนของปริภูมิเท่านั้น ดังนั้นจึงมีความเป็นไปได้ว่าคำตอบที่ได้อาจไม่ใช่คำตอบดีสุด การค้นหาเพียงบางส่วนโดยการค้นหาแบบบอตนั้นสามารถแบ่งได้เป็นสองประเภทคือ **การค้นหาแนวกว้างก่อน (breadth-first search)** ซึ่งเป็นการค้นหาในแนวกว้างก่อนเมื่อ

พิจารณาจากโครงสร้างต้นไม้ของปริภูมิสถานะ และ **การค้นหาแนวลึกก่อน (depth-first search)** คือหาแนวลึกก่อนเมื่อพิจารณาจากโครงสร้างต้นไม้ ส่วนการค้นหาแบบอิหริสติก (heuristic search) มีหลายเทคนิคด้วยกัน เช่น **อัลกอริทึมปีนเขา (hill-climbing search)** **อัลกอริทึมอบหนีวยาล่อง (simulated annealing algorithm)** การค้นหาดีสุดก่อน (best-first search) การค้นหา **A*** (**A* search**) เป็นต้น ในหัวข้อนี้จะกล่าวถึงเทคนิคค้นหาในแต่ละวิธีโดยเริ่มจากการค้นหาแบบบอตและตามด้วยการค้นหาแบบอิหริสติก

2.3 การค้นหาแบบบอต

ส่วนนี้อธิบายอัลกอริทึมการค้นหาแบบบอต (blind search) โดยจะเริ่มจากการค้นหาแนวว่างก่อน และต่อด้วยการค้นหาแนวลึกก่อน โดยตัวอย่างที่ใช้เพื่ออธิบายอัลกอริทึมคือ **ปัญหาโลกของบล็อก (block world problem)** ซึ่งแสดงใน [รูปที่ 2-4](#) ด้านล่างนี้



รูปที่ 2-4 ปัญหาโลกของบล็อก

ปัญหาคือกำหนดสถานะเริ่มต้นของการจัดเรียงตัวของบล็อกให้ ต้องการจัดเรียงใหม่ให้ได้ตามสถานะสุดท้าย โดยมีตัวกระทำการดังนี้

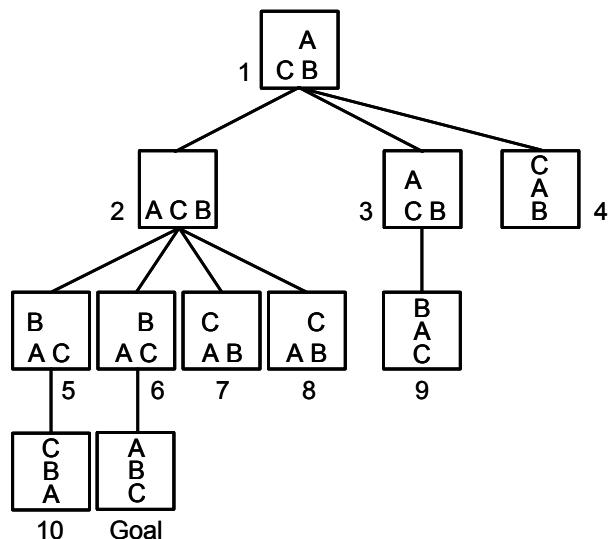
- (1) บล็อก X ไม่มีบล็อกอื่นทับ \rightarrow วาง X บนโต๊ะ
- (2) บล็อก X และ Y ไม่มีบล็อกอื่นทับ \rightarrow วาง X บน Y

โดยที่ X และ Y เป็นตัวแปรและสามารถแทนที่ด้วย 'A', 'B' หรือ 'C' เช่นเมื่อใช้ตัวกระทำการ (1) กับสถานะเริ่มต้น จะได้ว่าถ้าบล็อก 'A' ไม่มีบล็อกอื่นทับ แล้ววาง 'A' บนโต๊ะได้เป็นต้น

2.3.1 การค้นหาแบบกว้างก่อน

ในการค้นหาแบบกว้างก่อน (breadth-first search) นี้ สถานะทุกตัว (ซึ่งบางที่เรียกว่า **บ็อกซ์ (node)**) เมื่อมองปริภูมิค้นหาอยู่ในรูปของต้นไม้ ที่อยู่ในระดับเดียวกันของต้นไม้จะถูกตรวจสอบก่อนสถานะที่อยู่ในระดับถัดไป วิธีการของการค้นหาแบบนี้ทำโดยเริ่มจากการสร้างสถานะลูกของสถานะเริ่มต้นก่อน แล้วตรวจสอบว่ามีสถานะใดที่เป็นสถานะสุดท้ายหรือไม่ ถ้าหากว่ามีก็เป็นอันว่าการค้นหาสิ้นสุด ถ้าไม่มีก็จะสร้างสถานะลูกของสถานะเหล่านั้น แล้วทำการตรวจสอบสถานะลูกทุกตัวของสถานะเหล่านั้น ถ้าพบสถานะสุดท้ายการค้นหา ก็สิ้นสุด ถ้าไม่พบก็สร้างสถานะลูกของสถานะเหล่านั้นต่อไปอีก ทำเช่นนี้ไปเรื่อยๆ จนกว่าจะพบสถานะสุดท้ายหรือจนไม่สามารถสร้างสถานะลูกใหม่ได้อีก

ในการนี้ของปัญหาโลโกของบล็อก เริ่มจากสถานะเริ่มต้น เมื่อเราค้นหาด้วยการค้นหาแบบกว้างก่อน ก็จะได้สถานะที่เกิดขึ้นดังรูปที่ 2-5 ในตัวอย่างนี้กำหนดว่าลำดับของแก้วไม่มีความสำคัญ กล่าวคือสถานะ 'A C B' เท่ากับสถานะ 'C B A' เป็นต้น และในการสร้างสถานะจะไม่สร้างสถานะซ้ำเดิม เช่น จากสถานะที่ 2 เราสามารถสร้างสถานะลูกของมันตัวหนึ่งซึ่งเท่ากับสถานะที่ 1 แต่จะไม่นำสถานะนี้ไปลงเป็นสถานะลูกของสถานะที่ 2 เนื่องจากไปซ้ำกับสถานะที่ 1



รูปที่ 2-5 การค้นหาแบบกว้างก่อนในปัญหาโลโกของบล็อก

ตัวเลข 1, 2, 3, ... ในรูปด้านบนแสดงลำดับของสถานะที่ถูกสร้างขึ้น และ 'Goal' แสดงสถานะเป้าหมายหรือค่าตอบ โดยเริ่มจากสถานะที่ 1 ซึ่งเป็นสถานะเริ่มต้น จากนั้นใช้ตัว

กระทำการ (1) และตัวกระทำการ (2) และแทนที่ตัวแปร X และ Y ในตัวกระทำการทั้งสองให้เป็น 'A', 'B' หรือ 'C' ตามลำดับจะได้สถานะลูกเป็นสถานะที่ 2 สถานะที่ 3 และ สถานะที่ 4 ตามลำดับ เมื่อสร้างสถานะลูกของสถานะที่ 1 ครบทุกตัวแล้ว ก็จะลงมาในระดับถัดไปเพื่อสร้างสถานะลูกของสถานะที่ 2, 3 และ 4 เป็นเช่นนี้ไปจนกว่าจะได้สถานะสุดท้ายที่ต้องการ ซึ่งจะเห็นได้ว่าการค้นหาจะทำในแนววิ่งที่จะระดับตั้งแต่บนลงล่าง จากตัวอย่าง เราได้สถานะสุดท้ายเป็นสถานะที่ 11 และไม่ต้องทำการค้นหาต่อ เพราะการค้นหาแบบนี้เป็นการค้นหาเพียงบางส่วนเท่านั้น แม้ว่าจะมีเส้นทางอื่นอีกที่สามารถนำไปสู่สถานะสุดท้ายได้ อัลกอริทึมของการค้นหาแนววิ่งก่อนแสดงได้ในตารางที่ 2-1 ด้านล่างนี้

ตารางที่ 2-1 อัลกอริทึมการค้นหาแนววิ่งก่อน

Algorithm: Breadth-First Search

```

1. Node-list := {initial state}
2. UNTIL a goal state is found or Node-list is empty DO
    2.1 Remove the first element from Node-list and call
        it E.
    2.2 For Each operator matching E DO
        2.2.1 Apply the operator to generate a new state.
        2.2.2 IF the new state is a goal state THEN quit
            and return this state
        ELSE add the state to the end of Node-list.

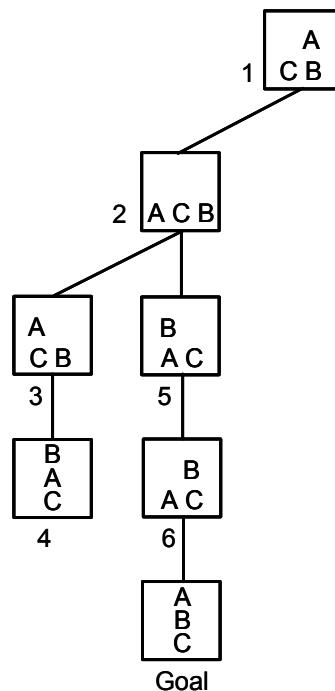
```

หมายเหตุ อัลกอริทึมด้านบนนี้ไม่ได้ตรวจสอบสถานะซ้ำ

2.3.2 การค้นหาแนวลึกก่อน

การค้นหาแนวลึกก่อน (depth-first search) จะสร้างสถานะในแนวลึกทางด้านมุมซ้ายล่างก่อน (ดูรูปที่ 2-6 ประกอบ) ถ้าสถานะตามแนวดิ่งถูกสร้างหรือกระจายจนหมดแล้วยังไม่ได้คำตอบ ก็จะไปลึกขึ้นด้านบนเพื่อหาเส้นทางอื่นที่จะเป็นไปได้

การค้นหาแบบแนววิ่งก่อนกับการค้นหาแนวลึกก่อนนั้น ไม่สามารถออกได้ว่าการค้นหาแบบไหนจะได้คำตอบเร็วกว่ากัน ขึ้นอยู่กับว่าคำตอบของเรารอยู่บริเวณไหนในปริภูมิสถานะ (กรณีของตัวอย่างในรูปที่ 2-6 นั้น การค้นหาแนวลึกก่อนพบคำตอบเร็วกว่า โดยสร้างสถานะทั้งสิ้น 7 ตัว ซึ่งคำตอบอยู่ที่บริเวณมุมซ้ายล่างของปริภูมิสถานะ) อัลกอริทึมการค้นหาแนวลึกก่อนแสดงในตารางที่ 2-2



รูปที่ 2-6 การค้นหาแบบแนวลึกก่อนในปัญหาโลกของบล็อก

ตารางที่ 2-2 อัลกอริทึมการค้นหาแนวลึกก่อน

Algorithm: Depth-First Search

```

1. IF initial state = goal state THEN quit and return
   success
   ELSE UNTIL success or failure DO
     1.1 Generate a successor, E, of the initial state
     IF there are no more successors
     THEN return failure
     ELSE call Depth-First Search with E as the
          initial state.
     1.2 IF success is return THEN return success
          ELSE continue in this loop.
  
```

หมายเหตุ อัลกอริทึมด้านบนนี้ไม่ได้ตรวจสอบสถานะซ้ำ

ข้อดีข้อเสียของอัลกอริทึมการค้นหาแนวลึกก่อน เมื่อเทียบกับการค้นหาแนวกว้างก่อน เป็นดังต่อไปนี้

ตารางที่ 2-3 เปรียบเทียบอัลกอริทึมการค้นหาแนวลึกก่อนและแนวกว้างก่อน

การค้นหาแนวลึกก่อน	การค้นหาแนวกว้างก่อน
1. ใช้หน่วยความจำน้อยกว่าการค้นหาแนวกว้างก่อน เพราะว่าสถานะในเส้นทางค้นหาปัจจุบันเท่านั้นที่ถูกเก็บ (ในขณะใดๆ จะเก็บเส้นทางเดียว พ้อไปเส้นทางอื่น เส้นทางที่ผ่านมาก่อนจะเป็นต้องเก็บไว้อีก)	1. ใช้หน่วยความจำมากกว่า เพราะต้องเก็บสถานะไว้ทุกด้วย เพื่อหาเส้นทางจากสถานะเริ่มต้นไปคำตอบ
2. อาจติดเส้นทางที่ลึกมากๆ โดยไม่พบคำตอบ เช่นในกรณีที่เส้นทางนั้นไม่มีคำตอบ และเป็นเส้นทางที่ยาวไม่สิ้นสุด ซึ่งการค้นหาแบบนี้ จะไปเส้นทางอื่นไม่ได้ (เช่นกรณีของ Prolog ซึ่งใช้วิธีค้นหาแบบนี้ จะทำการค้นหาไปเรื่อยๆ จนกว่าสถานะที่สร้างขึ้น ใช้หน่วยความจำเกิน ซึ่งจะเกิดข้อผิดพลาดขึ้น วิธีแก้ไขที่ใช้ใน Prolog คือให้ผู้ใช้กำหนดความลึกในการค้นหาคำตอบ จะได้ไม่เสียเวลา ในการค้นหาหนานเกิน จำเป็น เช่นกำหนดให้ความลึกในการค้นหาในแต่ละเส้นทางไม่มากกว่า 100 ขั้นตอน เป็นต้น หรืออาจกำหนดที่ขนาดของหน่วยความจำได้)	2. จะไม่ติดเส้นทางที่ลึกมากๆ โดยไม่พบคำตอบ
3. ถ้าคำตอบอยู่ที่ระดับ $h+1$ สถานะอื่นทุกด้วยที่อยู่ที่ระดับ 1 ถึงระดับ h ไม่จำเป็นต้องถูกกระจายจนหมด	3. ถ้าคำตอบอยู่ที่ระดับ $h+1$ สถานะทุกด้วยที่ระดับ 1 ถึงระดับ h จะต้องถูกกระจายจนหมด ทำให้มีสถานะที่ไม่จำเป็นในเส้นทางที่จะไปสู่คำตอบถูกกระจายออกด้วย
4. เมื่อพบคำตอบ ไม่สามารถรับประทานได้ว่าเส้นทางที่ได้เป็นเส้นทางสั้นสุด หรือไม่	4. ถ้ามีคำตอบจะประทานได้ว่าจะพบคำตอบแน่ และจะได้เส้นทางสั้นสุดด้วย (สมมติว่าระยะห่างหรือต้นทุนระหว่างสถานะ 2 ตัวใดๆ มีค่าเท่ากันหมด)

2.4 การค้นหาแบบอิวาริสติก

ปัญหา
การเดินทาง
ของ
พนักงานขาย

การค้นหาประเพณีอิวาริสติกนี้จะใช้ความรู้แบบหนึ่งที่เรียกว่าอิวาริสติกมาช่วยในการค้นหาให้มีประสิทธิภาพมากขึ้น โดยอิวาริสติกตัวนี้จะช่วยชี้แนะว่ากระบวนการค้นหาควรจะเลือกเส้นทางใดหรือสถานะใดเพื่อทำการค้นหาต่อไปให้ได้คำตอบอย่างมีประสิทธิภาพ พิจารณาปัญหาการเดินทางของพนักงานขาย (traveling salesman problem) ซึ่งแสดงในรูปที่ 2-7



รูปที่ 2-7 ปัญหาการเดินทางของพนักงานขาย

ในตัวอย่างของปัญหานี้ มีเมือง 7 เมือง พนักงานขายต้องการเดินทางไปให้ได้ครบทั้ง 7 เมืองและกลับมายังจุดเริ่มต้นโดยให้ได้ระยะทางโดยรวมสั้นที่สุด วิธีหนึ่งที่ทำได้คือ หาเส้นทางทั้งหมดที่เป็นไปได้ซึ่งจะมีด้วยกันทั้งสิ้น $(7-1)!/2$ ($=360$) แบบ จากนั้นวัดแต่ละเส้นทางว่าใช้ระยะทางเท่าไร แล้วก็เลือกเส้นทางที่สั้นที่สุด วิธีการนี้ไม่สามารถคำนวณได้อย่างมีประสิทธิภาพในทางปฏิบัติ เมื่อจำนวนเมืองมีมากขึ้น เช่นถ้ามีเมือง 100 เมือง จะมีเส้นทางที่เป็นไปได้ทั้งสิ้น 4.67×10^{155} แบบ

อิวาริสติก
คืออะไร?

ถ้าเราใช้สามัญสำนึกโดยคาดเดาอย่างมีเหตุผลว่า เมื่อเราต้องการระยะทางโดยรวมสั้นที่สุด เรา ก็จะเลือกเมืองที่อยู่ใกล้มากที่สุดกับเมืองที่เรียกว่าในปัจจุบัน และเดินทางไปเมืองนั้นก่อน เมื่อไปถึงเมืองนั้นแล้วค่อยทำในทำนองเดียวกันอีกว่า จะเดินไปยังเมืองที่ใกล้ที่สุด เมืองถัดไป ทำเช่นนี้จนกระทั่งเดินทางครบถ้วนเมือง ก็จะได้ระยะทางโดยรวมสั้นที่สุด แม้ว่าวิธีการเช่นนี้จะทำงานได้อย่างมีประสิทธิภาพ และคำตอบที่ได้มีแนวโน้มว่าจะดี แต่อย่างไรก็ดี คำตอบที่ได้โดยวิธีนี้อาจไม่เป็นเส้นทางที่สั้นที่สุดก็ได้ วิธีการเช่นนี้ก็คือการคำนวณความรู้แบบหนึ่งมาแทรกปัญหา ความรู้แบบนี้อาจไม่ใช่ความรู้ที่สมบูรณ์ แต่ก็พอที่จะนำมาแก้ไขปัญหาให้เราได้ และช่วยแนะนำให้เรารู้ว่าควรจะค้นหาเส้นทางอย่างไร เราเรียกว่าความรู้ที่ไม่สมบูรณ์หรือการคาดเดาอย่างมีเหตุผลแบบนี้ว่า **อิวาริสติก**

การค้นหาแบบอิวาริสติกคือ การค้นหาที่นำความรู้ประเทณน์มาใช้ช่วยซึ่งแนะนำเส้นทางในการค้นหาคำตอบ โดยมีลักษณะเด่นดังนี้

- เป็นเทคนิคที่ใช้เพิ่มประสิทธิภาพของกระบวนการค้นหา โดยอาจจะต้องยอมให้ขาดความสมบูรณ์ไปบ้าง คืออาจไม่พบคำตอบที่ถูกต้อง แม้ว่าในปริภูมิสถานะจะมีคำตอบน้อย
- การนำอิวาริสติกมาใช้จะต้องนำมาใช้ในรูปแบบที่วัดค่าได้อย่างง่าย ซึ่งมักทำโดยนิยามอิวาริสติกให้อยู่ในรูปแบบของฟังก์ชัน เราเรียกว่า **ฟังก์ชันอิวาริสติก (heuristic function)** ซึ่งเป็นฟังก์ชันที่คำนวณค่าจากสถานะไปยังตัวเลขที่ชี้ว่าสถานะนั้นเข้าใกล้สถานะเป้าหมายมากเท่าไร (ยิ่งมากเท่าไร ยิ่งมีโอกาสที่จะเปลี่ยนเป็นสถานะเป้าหมายมากเท่านั้น) การค้นหา ก็จะมุ่งไปเส้นทางที่มีค่าฟังก์ชันอิวาริสติกที่ดีกว่า
- ฟังก์ชันอิวาริสติกนี้เป็นสิ่งที่ใช้ซึ่งแนะนำกระบวนการค้นหาว่าควรจะค้นหาไปในทิศทางใด ซึ่งกระบวนการค้นหาที่ใช้ฟังก์ชันอิวาริสติกสามารถออกแบบได้หลายชนิด ดังจะกล่าวต่อไป
- ในบางกรณีที่เราสามารถนิยามฟังก์ชันอิวาริสติกได้อย่างสมบูรณ์แบบ การค้นหา ก็จะสามารถมุ่งตรงไปยังสถานะเป้าหมายโดยไม่ผิดเส้นทางเลย แต่ถ้าฟังก์ชันอิวาริสติกไม่ดีก็อาจทำให้กระบวนการค้นหาหลงไปในทิศทางที่ผิดได้ ทำให้คำตอบที่ได้เมื่อใช้อิวาริสติกไม่ใช่คำตอบที่ดีที่สุด

ฟังก์ชัน
อิวาริสติก

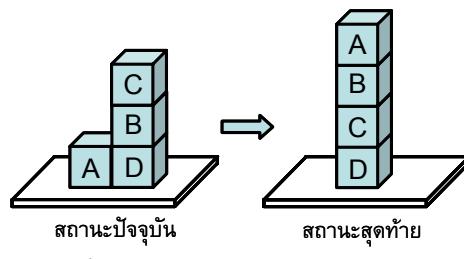
2.4.1 ตัวอย่างของฟังก์ชันอิวาริสติก

ในส่วนนี้จะยกตัวอย่างการนิยามฟังก์ชันอิวาริสติกสัก 3 ฟังก์ชันดังนี้

ฟังก์ชันอิวาริสติก h_1 สำหรับปัญหาลอกของบล็อก

ฟังก์ชันอิวาริสติกตัวแรกที่จะยกมาให้ดูเป็นฟังก์ชันสำหรับคำนวณค่าอิวาริสติก (*heuristic value*) สำหรับสถานะใดๆ ของปัญหาลอกของบล็อก ขอเรียกฟังก์ชันนี้ว่า h_1 ซึ่งนิยามดังนี้

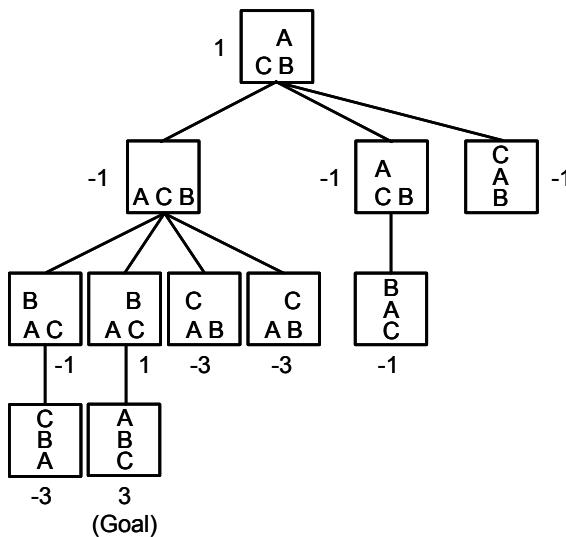
h_1 : บวกหนึ่งให้กับทุกบล็อกที่วางบนสิ่ง (บล็อกหรือโต๊ะ) ที่มีน้ำหนักอยู่ และลบหนึ่งกำไม่ใช่



รูปที่ 2-8 ตัวอย่างฟังก์ชันอิวาริสติก h_1

‘สิ่งที่มันควรอยู่’ ในที่นี่คือสถานะสุดท้ายหรือคำตอบ พิจารณา [รูปที่ 2-8](#) เราจะได้ว่าค่าพังก์ชันจะเท่ากับค่าที่บล็อกแต่ละบล็อกได้รับเมื่อเทียบกับสถานะสุดท้ายว่า มันวางอยู่บนสิ่งเดียวกันหรือไม่ ซึ่งจะได้ว่า $A = -1$ เพราะว่าในสถานะสุดท้าย A วางอยู่บน B แต่ในสถานะปัจจุบัน A วางอยู่บน D ดังนั้น เมื่อตรวจสอบบล็อกทุกบล็อกจะได้ดังนี้คือ $A = -1$, $B = -1$, $C = -1$, $D = 1$ ซึ่งทำให้ได้ค่ารวมของพังก์ชันเท่ากับ -2 หน่วย

ด้านล่างนี้แสดงค่าอิวาริสติกสำหรับสถานะต่างๆ ใน [รูปที่ 2-5](#)



รูปที่ 2-9 ค่าอิวาริสติก h_1 สำหรับสถานะต่างๆ ใน [รูปที่ 2-5](#)

ใน [รูปที่ 2-9](#) นี้ ตัวเลขที่กำกับที่แต่ละสถานะเป็นค่าอิวาริสติกที่คำนวณได้โดยพังก์ชัน h_1 จะ

เห็นว่าสถานะ $\boxed{B \quad A \quad C}$ ที่เข้าใกล้คำตอบมีค่าอิวาริสติกเท่ากับ 1 ส่วนสถานะ $\boxed{C \quad B \quad A}$ ที่ต่างจากคำตอบมากก็มีค่าอิวาริสติกน้อยสุดเท่ากับ -3 ซึ่งแสดงให้เห็นว่าพังก์ชัน h_1 สามารถวัดความใกล้เคียงคำตอบได้ดีขอข้างดี อย่างไรก็ไดเมื่อพิจารณาที่สถานะลูกของสถานะเริ่มต้นในระดับแรก จะพบว่าสถานะทุกตัวมีค่าเท่ากับ -1 ซึ่งหมายความว่าพังก์ชัน h_1 ไม่สามารถ

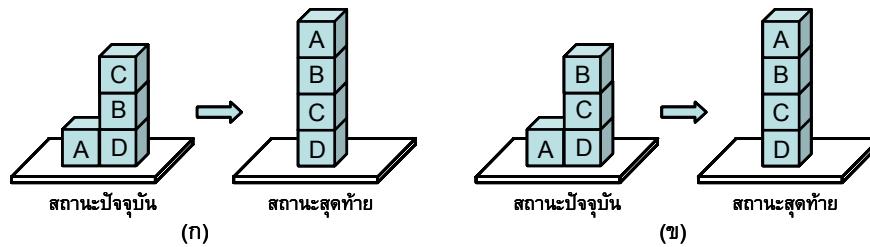
แยกความแตกต่างของสถานะทั้งสามนี้ได้ แม้ว่าสถานะ $\boxed{A \quad C \quad B}$ จะนำไปสู่คำตอบได้เร็ว

กว่าสถานะ $\boxed{A \quad C \quad B}$ และสถานะ $\boxed{C \quad B \quad A}$

พังก์ชันอิวิสติก h2 สำหรับปัญหาลอกของบล็อก

ดังจะเห็นได้ในตัวอย่างของพังก์ชันอิวิสติก h1 ที่กล่าวข้างต้นว่าไม่สามารถแยกความแตกต่างของบางสถานะได้ ในที่นี้จึงขอยกตัวอย่างพังก์ชันอิวิสติก h2 ที่มีประสิทธิภาพในการวัดความเข้าใกล้คำตอบหรือความดีของสถานะได้อย่างสมบูรณ์แบบและมีประสิทธิภาพดีกว่า h1 ซึ่ง h2 มีนิยามดังนี้

h2: สำหรับบล็อกแต่ละก้อนที่อยู่บนโครงสร้างที่ถูก บวก 1 แต้มให้กับบล็อกทุกก้อนที่อยู่ในโครงสร้างนั้นเพื่อเป็นคะแนนสำหรับบล็อกที่อยู่บนโครงสร้างที่ถูกนั้น และลบ 1 แต้มสำหรับบล็อกทุกก้อนที่อยู่บนโครงสร้างที่ผิด เพื่อเป็นคะแนนสำหรับบล็อกที่อยู่บนโครงสร้างที่ผิดนั้น ค่าอิวิสติกสำหรับสถานะคือคะแนนรวมของบล็อกทุกก้อนที่พิจารณาโดยที่โครงสร้างคือบล็อกที่เรียงตัวต่อกันตามแนวตั้ง(ไม่นับໂຕ)

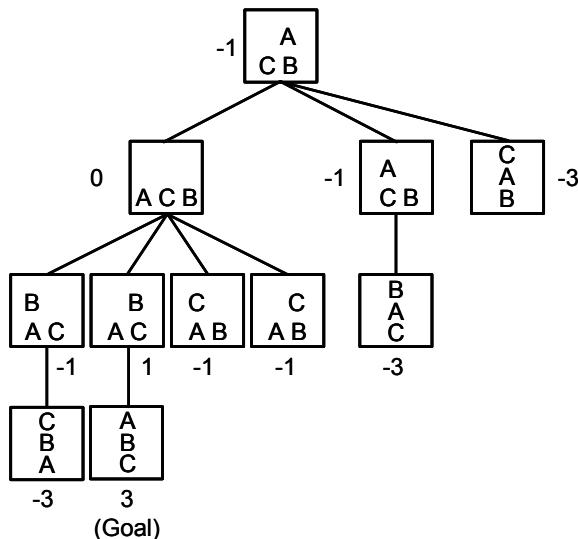


รูปที่ 2-10 ตัวอย่างพังก์ชันอิวิสติก h2

พังก์ชันอิวิสติก h2 นี้จะพิจารณาถึงค่าความดีของสถานะโดยดูที่โครงสร้างที่รองรับบล็อกไดๆ ว่าต้องเป็นโครงสร้างที่ตรงกับโครงสร้างที่รองรับบล็อกเดียวกันในคำตอบ ทำให้การวัดค่าอิวิสติกมีความแม่นยำขึ้น พิจารณาตัวอย่างในรูปที่ 2-10 (a) โครงสร้างที่รองรับ

บล็อก 'C' ในสถานะปัจจุบันคือโครงสร้าง ซึ่งต่างจากในคำตอบที่โครงสร้าง เป็นโครงสร้างที่รองรับบล็อก 'C' ดังนั้นสำหรับบล็อก 'C' ที่สถานะปัจจุบันที่มีบล็อก 2 ก้อนอยู่ในโครงสร้างรองรับที่ผิดจึงได้แต้มเท่ากับ -2 เมื่อคำนวณคะแนนสำหรับบล็อกทุกก้อนก็จะได้คะแนนตามนี้คือ $A = 0, B = -1, C = -2, D = 0$ ซึ่งทำให้ได้ค่ารวมของพังก์ชันเท่ากับ -3 หน่วย (บล็อก A ได้ศูนย์แต้ม เพราะว่าไม่มีโครงสร้างที่รองรับมันอยู่เลย เนื่องจากโครงสร้างไม่รวมถึงໂຕ) ในการนี้ของสถานะปัจจุบันในรูปที่ 2-10 (b) จะได้คะแนนตามนี้คือ $A = 0, B = 2, C = 1, D = 0$ ซึ่งทำให้ได้ค่ารวมของพังก์ชันเท่ากับ 3 หน่วย

เมื่อนำฟังก์ชัน h_2 ไปวัดค่าฮิวิสติกให้กับสถานะต่างๆ ในรูปที่ 2-5 จะได้ผลดังรูปที่ 2-11 ต่อไปนี้



รูปที่ 2-11 ค่าฮิวิสติก h_2 สำหรับสถานะต่างๆ ในรูปที่ 2-5

จะเห็นว่าฟังก์ชันฮิวิสติก h_2 สามารถแยกความแตกต่างระหว่างสถานะลูกๆ กันได้ดี ไม่เหมือนฟังก์ชันนี้ที่บอกร่วมกันว่าสถานะต่างๆ ไม่ต่างกัน แต่ h_2 สามารถเริ่มต้นได้อย่างถูกต้อง นอกเหนือจากนั้นฟังก์ชันนี้ยังบอกร่วมกันว่าสถานะที่มีค่าฮิวิสติกสูงกว่าสถานะอื่นๆ ในระดับเดียวกันล้วนเป็นสถานะที่นำไปสู่คำตอบได้โดยตรงทั้งสิ้น

ฟังก์ชันฮิวิสติกสำหรับปัญหา 8-Puzzle

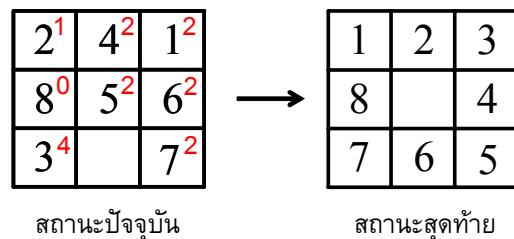
ตัวอย่างของฟังก์ชันฮิวิสติกสำหรับปัญหา 8-Puzzle แสดงในสมการด้านล่างนี้

$$h_{Man} = \sum_{i=1}^8 d_x(c_i, g_i) + \sum_{i=1}^8 d_y(c_i, g_i) \quad (2.1)$$

โดยที่ c_i , g_i , d_x , d_y คือพิกัดของแผ่นป้าย i ที่สถานะปัจจุบัน พิกัดของแผ่นป้าย i ที่สถานะเป้าหมาย ระยะห่างระหว่าง c_i กับ g_i ตามแกน x และระยะห่างระหว่าง c_i กับ g_i ตามแกน y ตามลำดับ

ฟังก์ชันนี้เรียกว่าฟังก์ชัน曼นอัตตัน ฟังก์ชันนี้สามารถแบร์คิวเมทรีอย่างง่ายคือ การหาค่าจำนวนครั้งที่ต้องขยับแผ่นป้ายตั้งแต่แผ่นที่ 1 ถึงแผ่นที่ 8 จากตำแหน่งปัจจุบันตาม

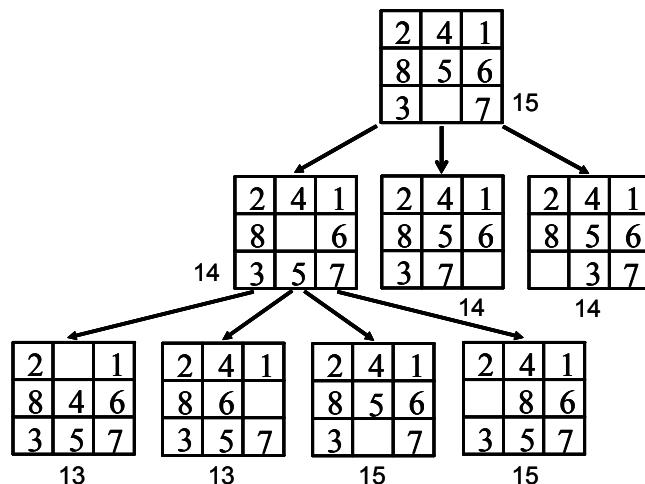
แนวแกน x และ y ไปยังตำแหน่งที่ควรจะอยู่ตามคำตобอกว่าต้องขยายทั้งสิ้นน้อยที่สุดรวมกีครั้ง โดยไม่ต้องมีเงื่อนไขว่าจะติดกับช่องว่างหรือไม่ (สามารถเลื่อนได้เลย แม้จะมีแผ่นป้ายอื่นวางอยู่แล้ว) สังเกตว่าพังก์ชันนี้ยังมีค่าน้อยยิ่งดี



รูปที่ 2-12 พังก์ชันแม่นอัตตันสำหรับปัญหา 8-Puzzle

ตัวอย่างเช่นพิจารณา [รูปที่ 2-12](#) ตัวเลขที่อยู่มุ่งความสนใจในแผ่นป้ายแต่ละแผ่นแสดงจำนวนครั้งที่ต้องเลื่อนแผ่นป้ายนั้นไปยังตำแหน่งที่มันควรอยู่เมื่อเทียบกับคำตобอกว่าต้องเลื่อน 2 ครั้ง (ตามแนวแกน x) จึงจะไปอยู่ในตำแหน่งเดียวกับในคำตобอกว่าต้องเลื่อน 1 ครั้ง (ตามแนวแกน y) แผ่นป้าย 3 ที่สถานะปัจจุบันต้องเลื่อนทั้งหมด 2+2 ครั้ง (2 ครั้งตามแนวแกน x และ 2 ครั้งตามแนวแกน y) จึงจะไปอยู่ในตำแหน่งเดียวกับในคำตобอกว่าต้องเลื่อนทั้งหมด 4 ครั้ง (2 ครั้งตามแนวแกน x และ 2 ครั้งตามแนวแกน y) ดังนั้นค่าฮาร์ดิกของสถานะปัจจุบันนี้ค่าเท่ากับ $2+1+4+2+2+2+0 = 15$ หน่วย

เมื่อนำพังก์ชัน h_{Man} ไปวัดค่าฮาร์ดิกให้กับสถานะต่างๆ ใน [รูปที่ 2-2](#) จะได้ผลดัง [รูปที่ 2-13](#) ต่อไปนี้

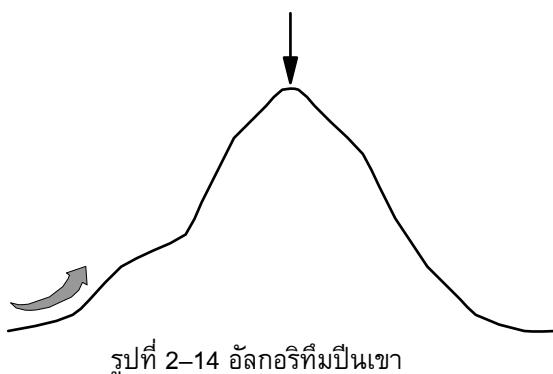


รูปที่ 2-13 ค่าพังก์ชันแม่นอัตตันสำหรับสถานะต่างๆ ใน [รูปที่ 2-2](#)

จากตัวอย่างของการนิยามฟังก์ชันอิวิสติกทั้งสามด้านบน คงจะเห็นแนวทางการนิยามฟังก์ชันอิวิสติกเหล่านี้ ซึ่งโดยมากเรามักนิยามฟังก์ชันเพื่อคำนวณความคล้าย (ความต่าง) กับคำตอบ และวัดเป็นตัวเลขไปใช้ในการค้นหาต่อไป ฟังก์ชันอิวิสติกที่ต้องคำนวณง่าย ไม่ยุ่งยากซับซ้อนมากจนกระทั่งเสียเวลาคำนวณมหาศาล ข้อสังเกตอีกอย่างก็คือการสร้างฟังก์ชันอิวิสติกอาจทำได้โดยตัดเงื่อนไขของการใช้ตัวกระทำการออก อย่างเช่นในตัวอย่างของ 8-Puzzle นั้น ตัวกระทำการเพื่อเลื่อนแผ่นป้ายมีเงื่อนไขว่า แผ่นป้ายจะเลื่อนได้ก็ต่อเมื่อแผ่นป้ายนั้นติดกับช่องว่าง การนิยามฟังก์ชันแม่นอัตตันนั้นเมื่อกับว่าแผ่นป้ายเลื่อนไปยังตำแหน่งต่างๆ ได้โดยไม่ต้องติดเงื่อนไขว่า แผ่นป้ายนั้นอยู่ติดกับช่องว่างหรือไม่ แล้วนักการเลื่อนแผ่นป้ายทั้งหมดเป็นการนิยามฟังก์ชัน เป็นต้น

2.4.2 อัลกอริทึมปีนเขา

ฟังก์ชันอิวิสติกที่นิยามขึ้นในข้างต้นนั้น สามารถนำมาช่วยกระบวนการค้นหาเพื่อให้ได้คำตอบอย่างรวดเร็วและมีประสิทธิภาพ วิธีการที่จะนำฟังก์ชันอิวิสติกมาใช้มีหลายวิธี ด้วยกันขึ้นอยู่กับว่าจะใช้ในลักษณะใด เช่นเลือกสถานะที่มีค่าอิวิสติกดีขึ้น และเดินไปยังสถานะนั้นโดยไม่ต้องสนใจสถานะที่มีค่าอิวิสติกแย่กว่าสถานะปัจจุบัน หรือว่าจะเก็บสถานะทุกตัวไว้แม้ว่าค่าอิวิสติกจะแย่ลง และพิจารณาสถานะเหล่านี้ทีหลัง เป็นต้น ในส่วนต่อไปนี้จะกล่าวถึงอัลกอริทึมต่างๆ ที่นำฟังก์ชันอิวิสติกมาช่วยในการค้นหาคำตอบ โดยเริ่มจากอัลกอริทึมปีนเขา (hill-climbing algorithm)



รูปที่ 2-14 อัลกอริทึมปีนเขา

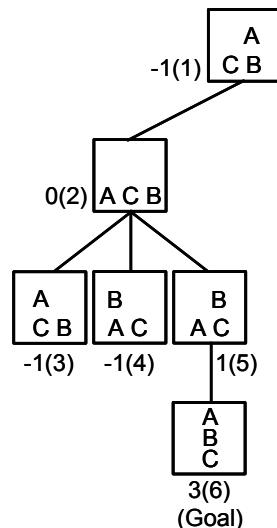
การค้นหาแบบนี้เปรียบเสมือนการปีนไปสู่ยอดเขาดังแสดงในรูปที่ 2-14 ซึ่งอัลกอริทึมจะขึ้นในแนวเดิ่งตลอด ถ้าเจอทางแยกเราจะไปแยกที่ตรงเดิ่งขึ้นไปจนกระทั่งถึงยอดเขา ความสูงจากฐานภูเขานั้นถึงตำแหน่งที่อยู่ในปัจจุบันก็จะเปรียบเหมือนค่าอิวิสติก (ในกรณีนี้เราต้องการหาค่าสูงสุด) ซึ่งในที่นี้ยิ่งมากยิ่งดี อัลกอริทึมแสดงในตารางที่ 2-4

ตารางที่ 2-4 อัลกอริทึมปีนเข้าอย่างง่าย

Algorithm: Simple Hill-Climbing Search

1. Evaluate the initial state.
2. **IF** the initial state=goal state **THEN**
 return the initial state and quit
 ELSE current state := initial state.
3. **UNTIL** a goal state is found or there are no new operators left to be applied in the current state **DO**
 - 3.1 Select an operator that has not yet been applied to the current state and apply it to produce a new state.
 - 3.2 Evaluate the new state.
IF new state=goal state **THEN**
 return the new state and quit
ELSE IF the new state is better than the current state **THEN**
 current state := new state
ELSE IF the new state is not better than the current state **THEN**
 continue in this loop.

ตัวอย่างการใช้ฟังก์ชันอิวาริสติก h_2 โดยอัลกอริทึมปีนเข้าอย่างง่ายใน [ตารางที่ 2-4](#) กับปัญหาโลกของบล็อกแสดงใน [รูปที่ 2-15](#)



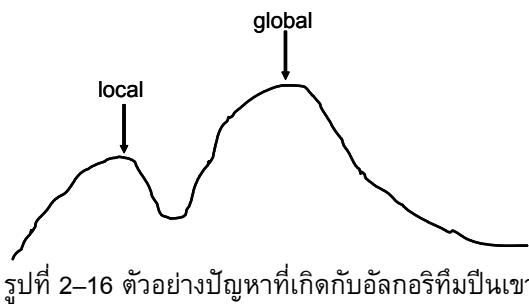
รูปที่ 2-15 ตัวอย่างอัลกอริทึมปีนเข้ากับปัญหาโลกของบล็อก

ตัวเลข h(i) ในรูปแสดงว่า สถานะที่ i มีค่าฮิวิสติกเท่ากับ h จากนั้นจะเห็นได้ว่า เริ่มต้นจากสถานะที่ 1 ที่มีค่าฮิวิสติกเท่ากับ -1 อัลกอริทึมปีนเข้าใช้ตัวกระทำการเพื่อสร้างสถานะลูกตัวแรกของสถานะที่ 1 แล้ววัดค่าฮิวิสติกได้ 0 ซึ่งมีค่าดีขึ้น ถ้าสังเกตจากรูปที่ 2-5 จะพบว่าสถานะที่ 1 มีสถานะลูกทั้งหมด 3 ตัว แต่ในการนี้ของอัลกอริทึมปีนเขานี้ เมื่อได้สถานะลูกตัวแรกซึ่งมีค่าฮิวิสติกดีขึ้น อัลกอริทึมจะไม่สร้างสถานะลูกที่เหลืออีก 2 ตัว และจะไม่มีการย้อนกลับมาที่สถานะลูกทั้งสองนี้ แม้ว่าหลังจากนี้อัลกอริทึมจะค้นไม่พบคำตอบ กล่าวคือเป็นการตัดทางเลือกทิ้งไปเลย ซึ่งการทำเช่นนี้แม้ว่าจะมีโอกาสไม่พบคำตอบ แต่ก็มีข้อดีที่เป็นการช่วยลดเวลาและปริภูมิที่ทำการค้นหาจะลดลงอย่างมาก

จากนั้นอัลกอริทึมมาที่สถานะที่ 2 แล้วเริ่มสร้างสถานะลูก ได้สถานะที่ 3 ที่มีค่าฮิวิสติก -1 ซึ่งยังคงในกรณีที่แปลงเช่นนี้ อัลกอริทึมจะไม่ไปยังสถานะลูกตัวนี้ และสร้างสถานะลูกตัวต่อไป โดยใช้ตัวกระทำการที่เหลือ ได้สถานะที่ 4 มีค่าฮิวิสติกเท่ากับ -1 ไม่ดีขึ้นเช่นกัน จึงสร้างสถานะลูกตัวดีไป เป็นสถานะที่ 5 มีค่าฮิวิสติกเท่ากับ 1 เป็นค่าที่ดีขึ้น อัลกอริทึมจะมายังสถานะนี้และค้นพบคำตอบในที่สุด

อัลกอริทึมปีนเขานี้จะมีประสิทธิภาพมากดังนี้ แสดงในตัวอย่างนี้ซึ่งกระจายสถานะทั้งสิบเพียง 6 ตัวแล้วพบคำตอบ เปรียบเทียบกับอัลกอริทึมการค้นหาแนววังก์ก่อนซึ่งใช้สถานะทั้งสิบถึง 11 ตัว อย่างไรก็ได้อัลกอริทึมนี้จะมีประสิทธิภาพมาก ถ้าใช้ฟังก์ชันฮิวิสติกที่ดีมากๆ ดังเช่นฟังก์ชัน h2 ซึ่งเป็นฟังก์ชันที่สมบูรณ์มาก ในกรณีที่ฟังก์ชันฮิวิสติกไม่ดีนัก อัลกอริทึมนี้ก็อาจหลงเส้นทางได้ และอาจไม่พบคำตอบแม้ว่าปริภูมิที่กำลังค้นหามีคำตอบอยู่ด้วยก็ตาม สาเหตุของการหลงเส้นทางประการหนึ่งจากการเลือกสถานะลูก ซึ่งอัลกอริทึมจะไม่ได้พิจารณาสถานะลูกทุกตัว โดยเมื่อพบสถานะลูกตัวใดตัวหนึ่งที่ดีขึ้น ก็จะเลือกเส้นทางนั้นไปทันที อัลกอริทึมนี้สามารถตัดแปลงเล็กน้อยให้พิจารณาสถานะลูกทุกตัวให้ครบก่อน แล้วเลือกสถานะลูกตัวที่มีค่าฮิวิสติกสูงสุด เมื่อทำเช่นนี้ก็จะทำให้อัลกอริทึมได้พิจารณาเส้นทางที่ดีที่สุด ณ ขณะหนึ่งๆ ได้ดีขึ้น เราเรียกอัลกอริทึมที่ตัดแปลงนี้ว่าอัลกอริทึมปีนเข้าชันสุด (steepest ascent hill-climbing)

2.4.3 อัลกอริทึมอปบเนี้ยวยำล่อง

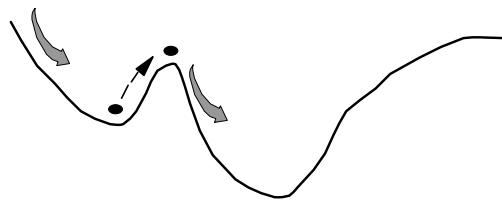


รูปที่ 2-16 ตัวอย่างปัญหาที่เกิดกับอัลกอริทึมปีนเข่า

ค่าดีสุดเฉพาะที่
และ
ค่าดีสุดวงกว้าง

พิจารณาตัวอย่างในรูปที่ 2-16 ซึ่งแสดงค่าอิวิสติกวัดความสูงจากฐานที่เกิดขึ้นระหว่างการค้นหาด้วยอัลกอริทึมปีนเข่า ปัญหาที่เกิดขึ้นก็คือ อัลกอริทึมปีนเข่าจะค้นพบสถานะที่มีค่าดีสุดเฉพาะที่ (*local optimum*) เท่านั้น ไม่สามารถค้นพบค่าดีสุดวงกว้าง (*global optimum*) ได้ เนื่องจากเมื่อการค้นหามาตกลิ้งสถานะดีสุดเฉพาะที่แล้ว พบว่าสถานะใหม่ที่สร้างขึ้นจะมีค่าอิวิสติกแย่ลงทั้งหมด ทำให้การค้นหาหยุดที่สถานะนั้น ปริภูมิสถานะจำนวนมากที่มีลักษณะของค่าอิวิสติกดังรูปด้านบนนี้ และในปัญหาจำนวนมาก เรากับพบปริภูมิสถานะที่มีสถานะดีสุดเฉพาะที่มากกว่าหนึ่งสถานะ ทำให้การค้นหาประสบความลำบากในการค้นให้พบสถานะดีสุดวงกว้าง

อัลกอริทึมการอปบเนี้ยวยำล่อง (simulated annealing algorithm) ถูกออกแบบมาเพื่อให้สามารถหลุดออกได้จากสถานะดีสุดเฉพาะที่ โดยใช้แนวคิดอุณหพลศาสตร์ของกระบวนการการอปบเนี้ยวยา ซึ่งเป็นขั้นตอนการลดอุณหภูมิลงอย่างช้าๆ ระหว่างการหลอม เพื่อให้ได้โลหะที่อยู่ในสภาพที่เหมาะสมที่สุด เป็นโลหะเนี้ยวยา ไม่เประ แนวคิดอธิบายให้เข้าใจได้โดยรูปที่ 2-17



รูปที่ 2-17 อัลกอริทึมการอปบเนี้ยวยำล่อง

เนื่องจากแนวคิดนี้เป็นการหาค่าต่ำสุด จึงใช้รูปที่กลับหัวกลับหางกับรูปที่ 2-16 ปัญหาของค่าดีสุดเฉพาะที่ (ในรูปนี้คือค่าต่ำสุดเฉพาะที่) ก็คือเมื่อการค้นหาแบบปีนเข่า (ในกรณีนี้

คือการค้นหาแบบบลังเหว) มาตกที่สถานะนี้ การค้นหา ก็จะหยุด วิธีที่จะแก้ปัญหาการตกที่สถานะนี้ได้ก็คือ ต้องดึงการค้นหาให้ขึ้นมาให้ได้เพื่อจะได้ค้นต่อไปจนพบค่าดีสุดของกว้าง

อัลกอริทึมการออมเหนี่ยวจำลองจะยอมให้การค้นหาวิ่งไปในทิศทางที่ไม่ดีในช่วงเริ่มต้นของกระบวนการค้นหา เพื่อเป็นการสำรวจทั่วๆ แบบหยาบก่อน แล้วจึงค่อยๆ ค้นหาอย่างละเอียดเมื่อเวลาผ่านไป ด้วยแนวคิดเช่นนี้ทำให้เราคาดหวังว่า ผลลัพธ์สุดท้ายที่ได้จะไม่ขึ้นกับสถานะเริ่มต้นมากนัก เพื่อให้เข้าใจถึงอัลกอริทึมนี้ ขออธิบายการออมเหนี่ยวของโลหะโดยย่อตั้งนี้

การออมเหนี่ยวของโลหะเป็นการหลอมโลหะจนละลาย (ทำให้โลหะอยู่ในสถานะที่มีพลังงานสูง) แล้วค่อยๆ ลดอุณหภูมิลงทีละน้อยจนโลหะเปลี่ยนกลับมาอยู่ในสถานะของแข็ง จุดมุ่งหมายคือพยายามทำให้โลหะกลับมาเป็นของแข็งในสถานะสุดท้ายที่มีพลังงานต่ำสุด ซึ่งโดยทั่วไปตามธรรมชาติ สาระพยายามเปลี่ยนตัวเองจากสถานะที่มีพลังงานสูงไปสู่สถานะพลังงานต่ำ แต่ก็มีความน่าจะเป็นที่สาระเปลี่ยนจากพลังงานต่ำไปพลังงานสูงอยู่บ้าง ความน่าจะเป็นนี้สามารถคำนวณได้โดยสมการด้านล่างนี้

$$p = e^{-\Delta E/kT} \quad (2.2)$$

โดยที่ ΔE เป็นระดับพลังงานที่เปลี่ยนไป (เป็นค่าบวก) T เป็นอุณหภูมิ และ k เป็นค่าคงที่ของโบลต์ซมันน์ (Boltzmann) การเปลี่ยนแปลงจากระดับพลังงานสูงไปต่ำนั้น มีความน่าจะเป็นที่จะเกิดในช่วงเริ่มต้นมากกว่าในช่วงปลายของการออมเหนี่ยว อัตราการลดอุณหภูมิในการออมเหนี่ยวเรียกว่า **หมายกำหนดการออมเหนี่ยว (annealing schedule)** ถ้าหมายกำหนดการออมเหนี่ยวเร็ว กล่าวคือลดอุณหภูมิลงอย่างรวดเร็ว โลหะก็มีโอกาสเข้าสู่สถานะสุดท้ายที่มีพลังงานสูงอยู่ เราจึงต้องพิจารณาหมายกำหนดการออมเหนี่ยวไม่ให้เร็วเกินไป แต่ถ้าช้าไปก็ทำให้เสียเวลาโดยไม่จำเป็นเช่นกัน

แนวคิดของการออมเหนี่ยวจำลองก็ได้จากการล้อเลียนการออมเหนี่ยวของโลหะ โดยเป็นการค้นหาแบบบีนเข้า (ลงเหว) แบบหนึ่ง ซึ่งการค้นหาสามารถไปในทิศทางที่ไม่ดีได้โดยเฉพาะในช่วงต้นของการออมเหนี่ยว เพื่อสำรวจบริเวณที่น่าจะนำไปสู่ค่าตอบดีสุด เราได้สมการความน่าจะเป็นสำหรับการออมเหนี่ยวจำลองดังนี้

$$p = e^{-\Delta E/T} \quad (2.3)$$

โดยที่ ΔE เป็นค่าอิควิตี้ที่เปลี่ยนไป (เป็นค่าบวก) T เป็นอุณหภูมิ เนื่องจาก k ในสมการ (2.2) เป็นค่าคงที่ จึงรวมเข้าไปใน T ได้

สมการนี้เมื่อนำไปใช้ร่วมกับการค้นหาแบบปีนเข้า ก็จะช่วยให้กระบวนการค้นหาสามารถหลุดออกจากค่าดีสุดเฉพาะที่ได้ตั้งกับความต้องการของเรา ตัวอย่างเช่นเมื่อเราอยู่ที่สถานะปัจจุบันมีค่าอิควิตี้ต่ำกว่า A และเมื่อสร้างสถานะลูกนี้มีค่าอิควิตี้ต่ำกว่า B ซึ่งยังคง 0 การค้นหาอาจไปยังสถานะลูกนี้ได้ โดยคำนวณค่าความน่าจะเป็น (p) ตามสมการด้านบน ได้เป็น $p = e^{-|A-B|/T}$ ค่าที่ได้นี้จะอยู่ระหว่าง 0 ถึง 1 จากนั้นเราจะสุ่มตัวเลข (random(0,1)) ขึ้นมาหนึ่งตัว ถ้าค่า p มีค่ามากกว่า ก็จะรับสถานะลูกเป็นสถานะต่อไปได้ ค่า T เป็นพารามิเตอร์ของอัลกอริทึมที่ความสามารถปรับแต่งให้เหมาะสมสมสำหรับปัญหาหนึ่งๆ ที่เราสนใจ โดยช่วงเริ่มต้นกำหนดให้เป็นค่ามากแล้วค่อยๆ ลดลงเมื่อการค้นหาดำเนินต่อไป

เมื่อพิจารณาสมการนี้ เราจะได้คุณสมบัติที่เราต้องการดังนี้คือ เมื่อ T มาก (ในช่วงต้นของการค้นหา) จะได้ p มีค่ามาก คือเรายอมให้การค้นหาไปในทิศทางที่ไม่ดีได้ง่าย หน่อยในช่วงต้นการค้นหา แต่เมื่อ T น้อย (ในช่วงปลายของการค้นหา) จะได้ p มีค่าน้อยคือการค้นหาเริ่มเข้าสู่คำตอบ ก็ไม่ควรให้การค้นหาระดูไปยังสถานะที่ยังคง เมื่อพิจารณา ΔE จะพบว่า เมื่อ ΔE มีค่ามากจะได้ว่า p มีค่าน้อย ก่อให้การค้นหาไม่ดี จึงเป็นคุณสมบัติที่น่าพอใจ เนื่องจากเราเชื่อว่าสถานะที่เป็นคำตอบมักจะเป็นหลุมลึกๆ ถ้าจะหลุดออกจากหลุมลึกได้ต้องก้าวใหญ่ ซึ่งไม่ควรให้เกิดขึ้นได้ง่าย ส่วนสถานะดีสุดเฉพาะที่มักเป็นหลุมลึกๆ ดังนั้นจึงให้โอกาสหลุดลอดได้ง่ายหน่อย อัลกอริทึมการอุบหนี่บวจำลองแสดงในตารางที่ 2-5

อัลกอริทึมจะมีลักษณะคล้ายกับอัลกอริทึมปีนเข้า แต่สามารถเลือกสถานะที่มีค่าอิควิตี้แย่ลงได้ ดังนั้นมีสิ่นสุดกระบวนการค้นหา สถานะตัวสุดท้ายที่พิจารณาอยู่อาจไม่ใช่สถานะที่มีค่าอิควิตี้ดีสุด ดังนั้นเราจึงจำเป็นต้องจำสถานะที่มีค่าอิควิตี้ดีสุดที่ผ่านมาไว้ โดยเก็บค่าไว้ในตัวแปรชื่อ BEST-SO-FAR ในตอนเริ่มกระบวนการค้นหา จะกำหนดค่าคงที่ตัวหนึ่งเป็นอุณหภูมิการอุบหนี่บว (T) และลดอุณหภูมิลงตามความเหมาะสม (ในขั้นตอนที่ 5.3 ในอัลกอริทึม) ในกรณีที่เราพิจารณาสถานะใหม่ (new state) ตัวหนึ่งอยู่ ถ้าพบว่าสถานะนี้มีค่าอิควิตี้ดีขึ้น ก็จะค้นหาต่อไปยังสถานะใหม่นี้ และปรับค่าตัวแปร BEST-SO-FAR แต่ถ้าหากว่าค่าอิควิตี้แย่ลง เราจะคำนวณค่าความน่าจะเป็น ($e^{-\Delta E/T}$) ที่จะไปยังสถานะใหม่นี้ การประยุกต์ใช้อัลกอริทึมนี้ ผู้ใช้จำเป็นต้องกำหนดพารามิเตอร์ในอัลกอริทึมให้เหมาะสมสมสำหรับปัญหาที่กำลังพิจารณาอยู่ด้วย พารามิเตอร์นี้ได้แก่ ค่าอุณหภูมิเริ่มต้นและปริมาณการลดอุณหภูมิ

ตารางที่ 2-5 อัลกอริทึมการออบหนี่งวจำลอง

Algorithm: Simulated Annealing Search

```

1. Evaluate the initial state.
2. IF initial state=goal state THEN
    return the initial state and quit
  ELSE current state := initial state.
3. BEST-SO-FAR := current state
4. T := constant
5. UNTIL a goal state is found or there are no new
   operators left to be applied in the current state DO
  5.1 Select an operator that has not yet been applied
      to the current state and apply it to produce a
      new state.
  5.2 Evaluate the new state.
    IF new state=goal state THEN
      return new state and quit
    ELSE IF the new state is better than the current
      state THEN {
      current state := new state
      IF the new state is better than BEST-SO-FAR
      THEN BEST-SO-FAR := new state }
    ELSE IF the new state is not better than the
      current state THEN {
       $\Delta E := |(\text{value of the current state}) -$ 
       $(\text{value of the new state})|$ 
      IF  $e^{-\Delta E/T} > \text{random}(0,1)$  THEN
        current state := new state }
  5.3 Revise T as necessary.
6. Return BEST-SO-FAR as the answer.

```

2.4.4 อัลกอริทึมดีสุดก่อน

อัลกอริทึมดีสุดก่อน (best-first search) จะเก็บสถานะทุกตัวโดยไม่มีการตัดทิ้งไป ต่างจาก อัลกอริทึมปีนเขาที่เมื่อเลือกเส้นทางหนึ่งแล้ว ตัวเลือกอื่นที่เป็นลูกของสถานะปัจจุบันจะถูกตัดทิ้งไป การไม่ตัดทิ้งทำให้อัลกอริทึมนี้ไม่พลาดเส้นทางที่นำไปสู่คำตอบ โดยในแต่ละขั้นตอนจะเลือกสถานะที่มีค่าฮีวิสติกดีสุด โดยพิจารณาสถานะทุกตัวที่ยังไม่ถูกกระจาย (สถานะที่ยังไม่ได้สร้างสถานะลูก) อัลกอริทึมแสดงใน [ตารางที่ 2-6](#)

ตารางที่ 2-6 อัลกอริทึมดีสุดก่อน

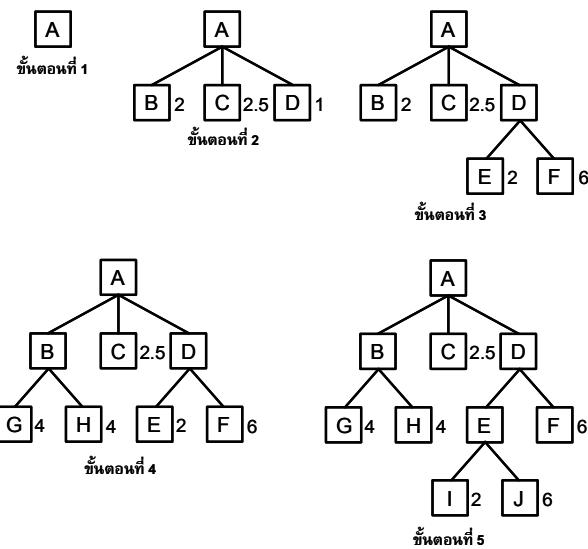
Algorithm: Best-First Search

```

1. OPEN := {initial state}
2. UNTIL a goal state is found or there are no states
   left on OPEN DO
   2.1 Pick the best node on OPEN.
   2.2 Generate its successors.
   2.3 FOR EACH successor DO
      IF the successor has not been generated THEN
         evaluate it, add it to OPEN, and record its
         parent.
      IF the successor has been generated before
      THEN change the parent if this new path is
      better than the previous one.

```

ตัวแปร OPEN ในอัลกอริทึมเก็บสถานะทุกตัวที่ถูกสร้างขึ้นแล้วแต่ยังไม่ถูกกระจาย (ยังไม่ได้สร้างสถานะลูกของมัน) ขั้นตอนสุดท้าย (IF STATEMENT) มีไว้เพื่อปรับเส้นทางและตันทุนใหม่ เนื่องจากว่าสถานะหนึ่งๆ อาจเข้าถึงจากหลายเส้นทาง (ตั้งที่ได้กล่าวข้างต้นแล้วว่าปรัชญาสถานะโดยทั่วไปเป็นกราฟ) ในกรณีที่เราพบเส้นทางใหม่ที่นำมาสู่สถานะที่เคยสร้างแล้ว และเส้นทางใหม่ดีกว่าหรือมีตันทุนหรือจำนวนครั้งจากสถานะเริ่มต้นมากยังสถานะนั้นอย่างว่าเส้นทางเดิม ก็ให้แก้ไขเส้นทางให้ถูกต้อง ตัวอย่างของการค้นหาแบบอัลกอริทึมดีสุดก่อนแสดงในรูปที่ 2-18



รูปที่ 2-18 อัลกอริทึมดีสุดก่อน

สมมติว่า 'A' เป็นสถานะเริ่มต้น สถานะลูกทุกตัวของ 'A' (คือ 'B', 'C' และ 'D') ถูกสร้างขึ้นในขั้นตอนที่ 2 จากนั้นเมื่อวัดค่าอิวิสติกของสถานะทุกตัวได้ว่า 'D' มีค่าดีสุด (ในที่นี้ยังน้อยยิ่งดี) จึงนำ 'D' มากระจายสถานะลูก ได้สถานะ 'E' และ 'F' ซึ่งมีค่าอิวิสติก 2 และ 6 ตามลำดับ เมื่อเปรียบเทียบค่าอิวิสติกของสถานะทุกตัวที่ยังไม่ได้กระจาย (เก็บไว้ใน OPEN ในตารางที่ 2-6) จะได้ว่า 'B' มีค่าดีสุด (เท่ากับ 'E' แต่ในที่นี้กำหนดให้กรณีค่าเท่ากันให้นำสถานะที่สร้างก่อนมาทำก่อน) จึงนำ 'B' มากระจายต่อ และต่อจากนั้นในขั้นตอนที่ 5 สถานะ 'E' ถูกกระจายต่อไปตามลำดับ เป็นเช่นนี้จนกระทั่งพบคำตอบหรือไม่สามารถสร้างสถานะใหม่ได้อีก

2.4.5 อัลกอริทึม A*

อัลกอริทึม A* (A* Search) เป็นการขยายอัลกอริทึมดีสุดก่อนโดยพิจารณาเพิ่มเติมถึงต้นทุนจากสถานะเริ่มต้นmany สถานะปัจจุบันเพื่อใช้คำนวณค่าอิวิสติกด้วย ในกรณีของอัลกอริทึม A* เราต้องการหาค่าต่ำสุดของฟังก์ชัน $f(s)$ ของสถานะ s นิยามดังนี้

$$f(s) = g(s) + h'(s) \quad (2.4)$$

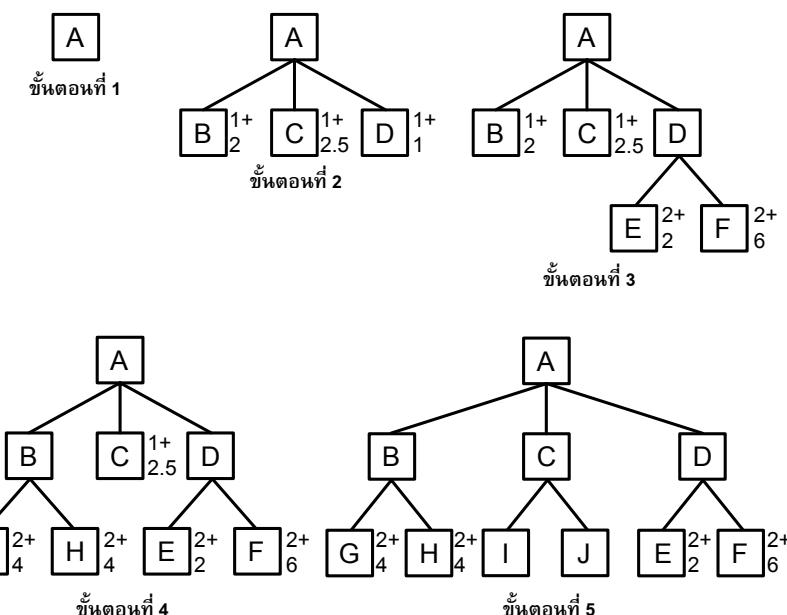
โดยที่ g คือฟังก์ชันที่คำนวณต้นทุนจากสถานะเริ่มต้นmany สถานะปัจจุบัน h' คือฟังก์ชันที่ประมาณต้นทุนจากสถานะปัจจุบันไปยังคำตอบ ดังนั้น f จึงเป็นฟังก์ชันที่ประมาณต้นทุนจากสถานะเริ่มต้นไปยังคำตอบ (ยิ่งน้อยยิ่งดี)

เรามองได้ว่าฟังก์ชัน h' คือฟังก์ชันอิวิสติกที่เราเคยใช้ในการค้นหาอื่นๆ ก่อนหน้านี้ เช่นอัลกอริทึมปีนเขา อัลกอริทึมดีสุดก่อน เป็นต้น ในที่นี้เราใส่เครื่องหมาย ' เพื่อแสดงว่า ฟังก์ชันนี้เป็นฟังก์ชันประมาณของฟังก์ชันจริงที่ไม่รู้ (เราทำได้แค่ประมาณว่า h' คือต้นทุนจากสถานะปัจจุบันไปยังคำตอบ เราจะรู้ต้นทุนจริงก็ต่อเมื่อเราได้ทำการค้นหาจริงจนไปถึงคำตอบแล้ว) ส่วน g เป็นฟังก์ชันที่คำนวณต้นทุนจริงจากสถานะเริ่มต้นmany สถานะปัจจุบัน (จึงไม่ได้ใส่เครื่องหมาย ') เพราะเราสามารถหาต้นทุนจริงได้เนื่องจากได้ค้นหาจากสถานะเริ่มต้นจนมาถึงสถานะปัจจุบันแล้ว ส่วน f ก็เป็นเพียงแค่ฟังก์ชันประมาณโดยการรวมต้นทุนทั้งสองคือ h' กับ g

อัลกอริทึม A* จะทำการค้นหาโดยวิธีเดียวกันกับอัลกอริทึมดีสุดก่อนทุกประการ ยกเว้นฟังก์ชันอิวิสติกที่ใช้เปลี่ยนมาเป็น f (ต่างจากอัลกอริทึมดีสุดก่อนที่ใช้ h') และด้วยการใช้ f อัลกอริทึม A* จึงให้ความสำคัญกับสถานะหนึ่งๆ 2 ประการคือ (1) สถานะที่ดีต้องมี h' ดีคือต้นทุนเพื่อจะนำไปสู่คำตอบหลังจากนี้ต้องน้อย และ (2) ต้นทุนที่จ่ายไปแล้วกว่าจะถึงสถานะนี้ (g) ต้องน้อยด้วย เราจึงได้ว่า A* จะค้นหาเส้นทางที่ให้ต้นทุนโดยรวมน้อยสุดตาม

ค่า f ซึ่งต่างจากอัลกอริทึมดีสุดก่อนที่เน้นความสำคัญของสถานะที่ตั้นทุนหลังจากนี้ที่จะนำไปสู่คำตอบต้องน้อย โดยไม่สนใจว่าตั้นทุนที่จ่ายไปแล้วกว่าจะนำมาถึงสถานะนี้ต้องเสียไปเท่าไร

รูปที่ 2-19 แสดงการค้นหาด้วยอัลกอริทึม A* กับสถานะในรูปที่ 2-18 โดยสมมติให้ตั้นทุนหรือระยะห่างระหว่างสถานะพ่อแม่ไปยังสถานะลูกเท่ากับ 1 หน่วย เช่นตั้นทุนจริง (g) จาก 'A' ไปยัง 'B', 'C' หรือ 'D' มีค่าเท่ากับ 1 หน่วย



รูปที่ 2-19 อัลกอริทึม A*

จากรูปจะเห็นได้ว่าในขั้นตอนที่ 4 สถานะ 'C' จะถูกเลือกมาบรรจายโดยอัลกอริทึม A* เนื่องจากมีค่า f น้อยสุดเท่ากับ 3.5 ซึ่งน้อยกว่า 'E' ที่มีค่าเท่ากับ 4 แม้ว่าค่า h ของ E จะน้อยกว่า ซึ่งต่างจากการสร้างสถานะของอัลกอริทึมดีสุดก่อน

2.4.6 การค้นหาตาม

ตาม (tabu, taboo) แปลว่า ต้องห้าม (เช่นสิ่งของที่เป็นของศักดิ์สิทธิ์ จึงห้ามแตะต้อง) การค้นหาตามเป็นเทคนิคการค้นหาที่ค่อนข้างใหม่ มักไม่ถูกอธิบายไว้ในหนังสือเกี่ยวกับปัญญาประดิษฐ์ ในหัวข้อนี้เราจะใช้เนื้อที่ค่อนข้างมากเพื่อบรรยายถึงวิธีการนี้

ในกระบวนการค้นหาตามนั้น จะทำเครื่องหมายบนเส้นทางบางเส้นทางที่ไม่สนใจจะค้นหา การทำเครื่องหมายนี้อาจทำในระดับของตัวกระทำการ หรือหน่วยย่อยของตัวกระทำการที่ได้ หน่วยย่อยได้ที่ถูกทำเครื่องหมายไว้จะเปลี่ยนสถานภาพต้องห้าม (*tabu status*) ให้อยู่ในภาวะต้องห้าม (*tabu active*) กล่าวคือหน่วยย่อยนี้จะไม่ถูกนำมาใช้เพื่อสร้างเส้นทางในการค้นหา อาจเป็นเพราะเส้นทางนี้คงไม่นำไปสู่คำตอบหรืออาจเป็นเส้นทางที่เคยค้นหามาแล้ว เป็นต้น

การค้นหาตามจะกำหนดสถานภาพของหน่วยย่อยโดยขึ้นกับหน่วยความจำ ซึ่งหน่วยความจำนี้จะเปลี่ยนแปลงตามเวลาและสภาพแวดล้อมในระหว่างกระบวนการค้นหา การค้นหาตามมีแนวคิดที่ว่า การค้นหาที่ตลาดจะต้องพิจารณาถึงสิ่งเหล่านี้คือ

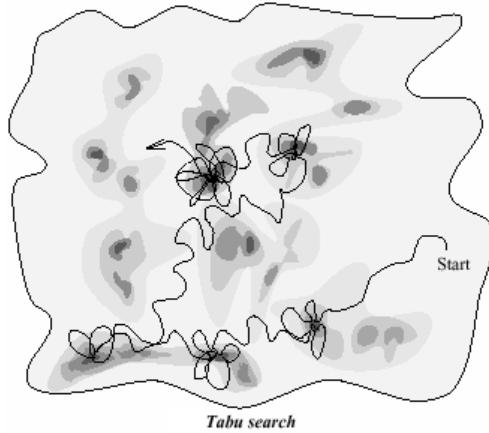
1. หน่วยความจำปรับตัว (*adaptive memory*) กล่าวคือหน่วยความจำจะต้องมีการปรับตัว เพื่อให้การค้นหาเป็นไปอย่างมีประสิทธิภาพตามสภาพการค้นหา ณ ตำแหน่งนั้นๆ
2. การสำรวจแบบตอบสนอง (*responsive exploration*) ซึ่งการค้นหานั้น เราจะต้องทำการสำรวจอย่างถี่ถ้วนในเส้นทางที่ดี เพื่อหวังว่าอาจจะได้เส้นทางที่ดีกว่าเดิม นอกจากนั้นแล้ว เรายังต้องค้นหาในเส้นทางที่ไม่ดีด้วย เพราะบางครั้งเส้นทางที่เลวร้ายจะให้ข้อมูลมากกว่าเส้นทางที่ดีเพื่อหาเส้นทางใหม่ที่ดีขึ้นได้

หน่วยความจำที่ใช้ในการค้นหานี้มี 2 ชนิดคือ

1. หน่วยความจำตามเวลา (*recency-based memory*) เป็นหน่วยความจำที่จะปรับเปลี่ยนไปตามขั้นตอนในการค้นหา
2. หน่วยความจำตามความถี่ (*frequency-base memory*) เป็นหน่วยความจำที่ใช้สำหรับจั่วตัวกระทำการตัวไหนใช้งานบ่อยๆ

หน่วยความจำทั้งสองนี้จะถูกนำมาใช้เพื่อปรับสถานภาพของตัวกระทำการ หน่วยความจำทั้งสองนี้ใช้สำหรับการค้นหาในสองลักษณะคือ (1) ความละเอียด (*intensification*) และ (2) ความหลากหลาย (*diversification*) ความละเอียดหมายถึงว่า เมื่อเราพบว่าผลเฉลย (*solution*)¹ อยู่บริเวณใดบริเวณหนึ่ง เราจะพยายามค้นหาบริเวณใกล้เคียงให้มากขึ้นเพื่อหาผลเฉลยที่ดีกว่า ส่วนความหลากหลายหมายถึง เมื่อเราค้นหาผลเฉลยพบแล้วว่าอยู่ในบริเวณใดบริเวณหนึ่ง ให้เลือกเส้นทางที่แตกต่างจากเดิมบ้าง เพื่อที่เราอาจจะได้ผลเฉลยที่ดีขึ้น แม้ว่าเส้นทางนั้นจะเป็นเส้นทางที่เลว (เมื่อประเมินจากค่าอิหริสติก)

¹ 'ผลเฉลย' ในบริบทของการค้นหาตามเทียบได้กับ 'สถานะ' ของการค้นหาในปริภูมิสถานะ ซึ่งผลเฉลยนี้อาจไม่เป็นผลเฉลยดีสัก



รูปที่ 2-20 การค้นหาตาม

รูปที่ 2-20 แสดงแนวคิดการค้นหาตาม ในรูปบริเวณที่มีสีเข้มแสดงถึงบริเวณที่มีค่า อิวาริสติกที่ดี จากรูปจะเห็นว่าจากจุดเริ่มต้น (Start ในรูป) การค้นหาจะพยายามไปสู่บริเวณ ที่มีค่าอิวาริสติกสูง (สีเข้ม) ซึ่งก็จะพบจุดสูงสุดเฉพาะที่ จากนั้นก็จะสำรวจบริเวณรอบๆ และ ค่อยเปลี่ยนไปยังเส้นทางใหม่ที่ต่างไปจากเดิม ซึ่งอาจต้องไปในเส้นทางใหม่ที่มีค่าอิวาริสติก ไม่ดีด้วย ในที่สุดก็คาดหวังว่าการค้นหาจะสามารถไปสู่จุดสูงสุดด้วยหวังว่างได้

หน่วยความจำระยะสั้น

ปัญหาที่เราสนใจคือ หาค่า x ที่ทำให้ฟังก์ชัน $f(x)$ มีค่าต่ำสุด โดยกำหนดให้ $N(x)$ เป็น สถานะข้างเคียง (neighborhood) ของ x ในปริภูมิสถานะ X กล่าวคือ $N(x)$ คือสถานะลูกที่ สามารถสร้างได้จากตัวกระทำการ

การค้นหาตามใช้หน่วยความจำ 2 ชนิดเพื่อเปลี่ยนค่า $N(x)$ คือ **หน่วยความจำระยะสั้น (short term memory)** เพื่อใช้สำหรับเป็นหน่วยความจำตามเวลา และ **หน่วยความจำระยะยาว (long term memory)** ใช้เป็นหน่วยความจำตามความที่ โดยให้ $N^*(x)$ แทนจุดข้างเคียง ของ x ที่เปลี่ยนไป เมื่อใช้หน่วยความจำเหล่านี้มาช่วยสร้างสถานะข้างเคียง

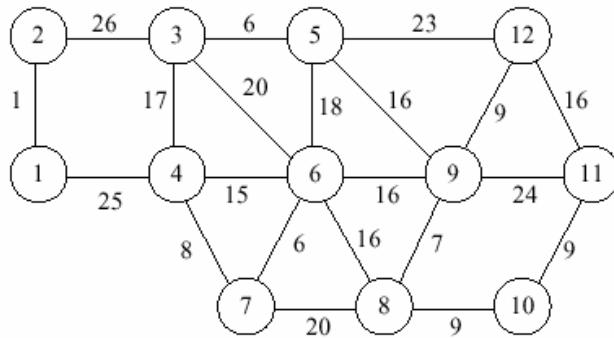
ในการนี้ของหน่วยความจำระยะสั้น หน่วยความจำนี้ใช้เป็นหน่วยความจำตามเวลาเพื่อ เก็บผลเฉลยหรือคุณสมบัติของผลเฉลย (*solution attribute*) ในการค้นหาที่เพิ่งจะผ่านมา และคุณสมบัติที่ปรากฏในผลเฉลยที่เพิ่งจะค้นหาไปจะถูกกำหนดให้มีสถานภาพเป็น “**ภาวะต้องห้าม**” ซึ่งภาวะต้องห้ามก็คือการทำเครื่องหมายไว้ว่าเราไม่ต้องค้นหาเส้นทางหรือผล เฉลยอื่นๆ ถ้าเส้นทางหรือผลเฉลยนั้นๆ มีคุณสมบัติเหมือนกับผลเฉลยที่มีเพิ่งค้นหาไปเมื่อ เวลาๆ นี้ เพราะจะได้ผลเฉลยที่ใกล้เคียงกันนั่นเอง และผลเฉลยอื่นๆ ที่จะพบในอนาคตที่มี คุณสมบัติเป็นภาวะต้องห้ามก็จะมีสถานภาพเป็นภาวะต้องห้ามด้วย (ก็เพราะว่ามันมี

คุณสมบัติเมื่อมีนัก จึงไม่จำเป็นจะต้องไปเสียเวลาค้นหามันอีก) ดังนั้นหน้าที่ของหน่วยความจำจะสั้นก็คือ การป้องกันการสร้างผลเฉลยบางตัวไม่ให้อยู่ใน $N^*(x)$ กล่าวคือ $N^*(x)$ จะเป็นเซตย่อยของ $N(x)$ โดยมีสถานะบางตัวที่มีสถานภาพเป็นภาวะต้องห้ามถูกตัดออกไป (ในกรณีของหน่วยความจำจะขยาย $N^*(x)$ อาจเป็นซูเปอร์เซตของ $N(x)$) $N^*(x)$ นี้จะถูกกำหนดโดยหน่วยความจำจะสั้น ซึ่งจะเปลี่ยนแปลงในแต่ละครั้งของการค้นหา

ตัวอย่างปัญหาต้นไม้ k กิ่งน้อยสุด

ในที่นี้จะยกตัวอย่างปัญหาต้นไม้ k กิ่งน้อยสุด เพื่อแสดงกระบวนการค้นหาตามด้วยหน่วยความจำจะสั้น

ปัญหาต้นไม้ k กิ่งน้อยสุด (minimum k-tree problem) คือ การหาต้นไม้ที่มี k กิ่งจากกราฟโดยให้ผลรวมของน้ำหนักของกิ่งน้อยที่สุด (ในที่นี้ให้ k=4) ดังรูปที่ 2-21



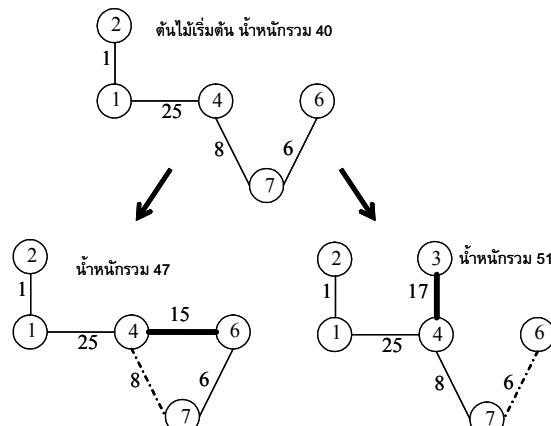
รูปที่ 2-21 ตัวอย่างปัญหาต้นไม้ k กิ่งน้อยสุด

จากราฟข้างบน เราจะสร้างต้นไม้ที่มี 4 กิ่งโดยใช้อัลกอริทึมตัดกราม (greedy algorithm) ซึ่งเริ่มจากการหาต้นไม้ที่มี 2 กิ่ง คือ $(1,2)$ จากนั้นหาต้นไม้ที่เพิ่มกับต้นไม้ที่มี 2 กิ่ง ให้มีน้ำหนักน้อยสุด ทำเช่นนี้ไปจนครบ 4 กิ่ง จะได้ผลดังตารางข้างล่างนี้

ตารางที่ 2-7 การสร้างผลเฉลยเริ่มต้นด้วยอัลกอริทึมตัดกราม

ขั้นตอนที่	ตัวเลือก	กิ่งที่เลือก	น้ำหนักร่วม
1	(1,2)	(1,2)	1
2	(1,4), (2,3)	(1,4)	26
3	(2,3), (3,4), (4,6), (4,7)	(4,7)	34
4	(2,3), (3,4), (4,6), (6,7), (7,8)	(6,7)	40

เมื่อได้ต้นไม้ที่มี 4 กิ่งเรียบร้อยแล้ว พบร่วมน้ำหนักร่วมที่หาได้คือ 40 เมื่อได้ผลเฉลยเริ่มต้นแล้ว ต่อไปเราก็จะสร้างผลเฉลยข้างเคียงใหม่ โดยแทนที่กิ่ง 1 กิ่งในต้นไม้ด้วยกิ่งใหม่โดยที่ผลที่ได้ต้องเป็นต้นไม้ [รูปที่ 2-22](#) แสดงการสร้างผลเฉลยใหม่ 2 ตัวโดยการลบกิ่งหนึ่งกิ่ง (แสดงด้วยเส้นประในรูป) ออกจากต้นไม้เดิม และเพิ่มกิ่งหนึ่งกิ่ง (แสดงด้วยเส้นทึบในรูป) เข้าไป



รูปที่ 2-22 การสร้างผลเฉลยข้างเคียงโดยลบกิ่งหนึ่งกิ่งและเพิ่มกิ่งหนึ่งกิ่ง

การเลือกคุณสมบัติที่ใช้กำหนดสถานภาพต้องห้ามในการสร้างต้นไม้ในปัญหานี้กำหนดให้ กิ่งเพิ่มเข้า (added edge) และ กิ่งลบออก (dropped edge) เป็นตัวกำหนดสถานภาพต้องห้าม โดยที่

- กิ่งเพิ่มเข้าหมายถึงคำตอบอื่นๆ ใน $N(x)$ ที่จะลบกิ่งนี้ จะมีสถานภาพเป็นภาวะต้องห้าม (ไม่ต้องนำมาพิจารณา)
- กิ่งลบออกหมายถึงคำตอบอื่นๆ ใน $N(x)$ ที่จะเพิ่มกิ่งนี้ จะมีสถานภาพเป็นภาวะต้องห้าม (ซึ่งไม่นำมาพิจารณาอีกเช่นเดียวกัน)

และกำหนดให้ **ระยะเวลาต้องห้าม (tabu-tenure)** คือระยะเวลาที่กิ่งเพิ่มเข้าหรือกิ่งลบออกจะส่งผลต่อสถานภาพต้องห้าม โดยกำหนดไว้ว่า

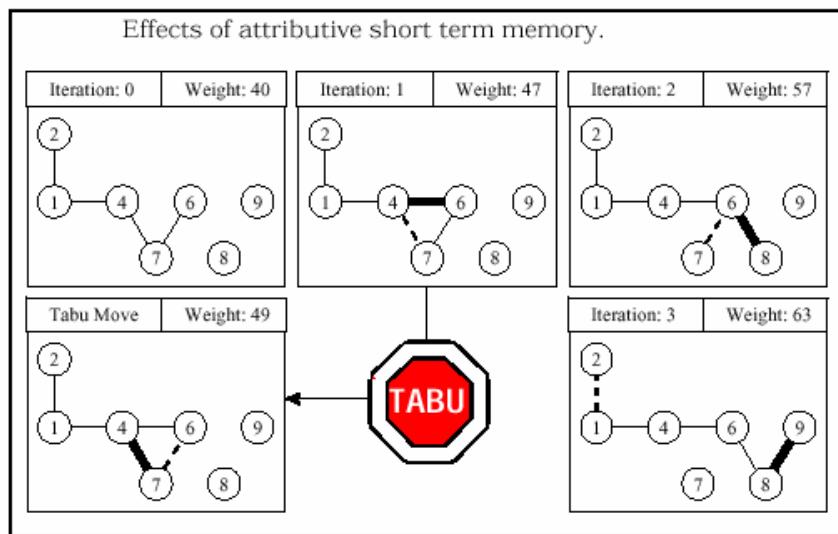
- ให้ระยะเวลาต้องห้ามของกิ่งลบออกเท่ากับ 2 หมายความว่าห้ามเพิ่มกิ่งนี้เข้าไปในอีก 2 รอบ
- ให้ระยะเวลาต้องห้ามของกิ่งเพิ่มเข้าเท่ากับ 1 หมายความว่าห้ามลบกิ่งนี้ออกไปในอีก 1 รอบ

ดังนั้นเมื่อผ่านไป 3 รอบ จะได้ผลเฉลยดัง [ตารางที่ 2-8](#) ต่อไปนี้

ตารางที่ 2-8 ผลเฉลยที่ได้เมื่อผ่านไป 3 รอบ

รอบที่	ระยะเวลาต้องห้าม		กิ่งเพิ่มเข้า	กิ่งลบออก	น้ำหนัก
	1	2			
1			(4,6)	(4,7)	47
2	(4,6)	(4,7)	(6,8)	(6,7)	57
3	(6,8), (4,7)	(6,7)	(8,9)	(1,2)	63

สังเกตที่ระยะเวลาต้องห้าม ในรอบแรกจะยังไม่มีค่าใดๆ เก็บไว้ ในช่อง ‘กิ่งเพิ่มเข้า’ จะเก็บค่ากิ่ง (4,6) และ ‘กิ่งลบออก’ เก็บค่า (4,7) เพื่อให้ทราบว่าขณะนี้ได้เพิ่มและลบกิ่งใด และน้ำหนักร่วมเป็นเท่าไร ในรอบที่สองจะมีการเก็บค่าระยะเวลาต้องห้ามแล้ว ในช่อง ‘กิ่งเพิ่มเข้า’ ก็จะเพิ่มเข้าที่หมายเลขอีก 1 ซึ่งตรงตามข้อตกลงข้างต้นว่า ถ้าเป็น ‘กิ่งเพิ่มเข้า’ จะให้ระยะเวลาต้องห้ามมีค่าเป็น 1 หมายความว่าห้ามไปยุ่งกับกิ่งที่ (4,6) หนึ่งรอบและห้ามยุ่งกับกิ่ง (4,7) สองรอบ ในขณะที่รอบนี้มีการเพิ่มกิ่ง (6,8) และลบกิ่ง (6,7) ดู [รูปที่ 2-23](#) ประกอบ ในรอบที่สามจะเห็นว่า (4,6) หายไปจากช่อง 1 ในระยะเวลาต้องห้าม และ (4,7) จะย้ายมาอยู่ในช่อง 1 แทน เพราะมันถูกห้ามสองรอบ แต่ผ่านไปหนึ่งรอบจึงยังคงถูกห้ามอีกเพียง 1 รอบเท่านั้น นอกจากนั้นในช่อง 1 ยังมีกิ่ง (6,8) เพิ่มเข้ามาอีกด้วย ทำเช่นนี้ไปเรื่อยๆ

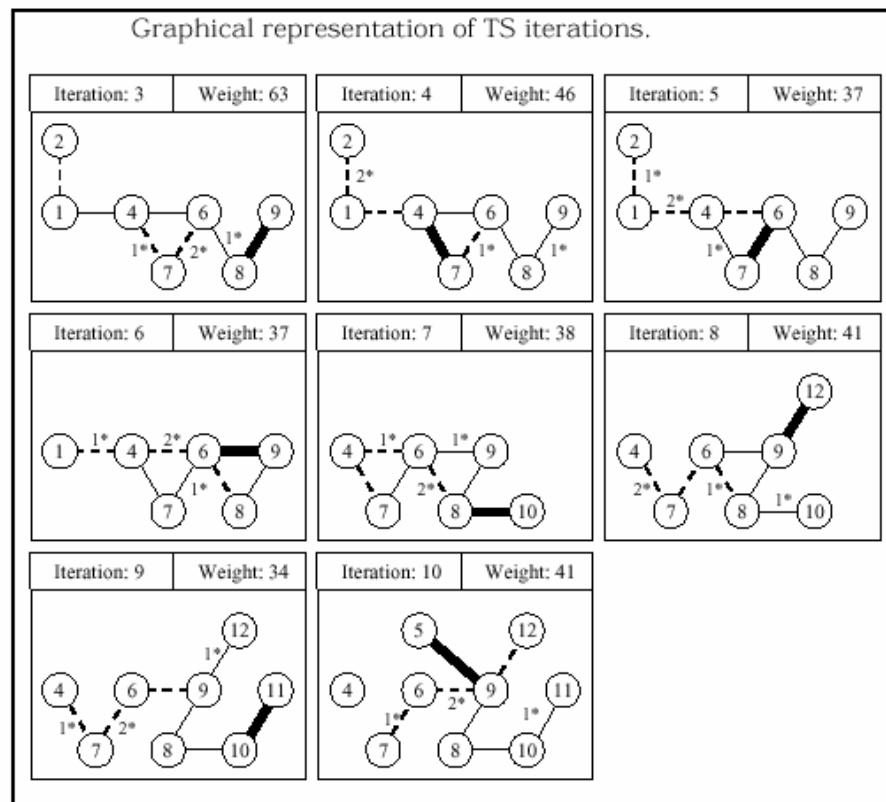


รูปที่ 2-23 ผลของการใช้หน่วยความจำระยะสั้น

เกณฑ์แห่งความหวัง

เกณฑ์แห่งความหวัง (aspiration criteria) คือเกณฑ์ที่ใช้เปลี่ยนสถานภาพต้องห้ามของผลเฉลยได้ๆ จากภาวะต้องห้าม (tabu active) เป็นภาวะไม่ห้าม (tabu non-active) เนื่องจากในบางครั้งผลเฉลยที่มีสถานภาพเป็นภาวะต้องห้ามเป็นคำตوبที่ดี และโดยทั่วไปจะใช้เกณฑ์ที่ว่าถ้าเป็นผลเฉลยที่ให้ค่า $f(x)$ น้อยที่สุดเท่าที่เคยมีมาจะยอมรับผลเฉลยนั้นได้ (เปลี่ยนจาก 'ต้องห้าม' เป็น 'ไม่ห้าม')

การค้นหาตามจะทำการค้นหาไปจนกระทั่งจำนวนรอบเกินกว่าค่าขีดแม่งที่กำหนดไว้ (ในตัวอย่างด้านบนกำหนดให้เป็น 10 รอบ) กระบวนการค้นหาแสดงใน [รูปที่ 2-24](#) และ [ตารางที่ 2-9](#) ซึ่งจะพบว่าการใช้ภาวะต้องห้ามจะทำให้ประหยัดเวลาในการไปในเส้นทางต่างๆ ได้มากพอครับ ตัวเลข 1* และ 2* ใน [รูปที่ 2-24](#) แสดงระยะเวลาต้องห้ามของกิจกรรม



รูปที่ 2-24 ต้นไม้ที่ได้ในรอบที่ 3 ถึงรอบที่ 10 ของการค้นหาตาม

ตารางที่ 2-9 ผลเฉลยที่ได้เมื่อผ่านไป 10 รอบ

รอบที่	ระยะเวลาต้องห้าม		กิ่งเพิ่ม เข้า	กิ่งลบ ออก	น้ำหนัก
	1	2			
0					40
1			(4,6)	(4,7)	47
2	(4,6)	(4,7)	(6,8)	(6,7)	57
3	(6,8), (4,7)	(6,7)	(8,9)	(1,2)	63
4	(6,7), (8,9)	(1,2)	(4,7)	(1,4)	46
5	(1,2), (4,7)	(1,4)	(6,7)	(4,6)	37
6	(1,4), (6,7)	(4,6)	(6,9)	(6,8)	37
7	(4,6), (6,9)	(6,8)	(8,10)	(4,7)	38
8	(6,8), (8,10)	(4,7)	(9,12)	(6,7)	41
9	(4,7), (9,12)	(6,7)	(10,11)	(6,9)	34
10	(6,7), (10,11)	(6,9)	(5,9)	(9,2)	41

หน่วยความจำระยะยาว

การค้นหาที่กล่าวมาข้างต้นเป็นการค้นหาในบริเวณใกล้ๆ กับผลเฉลยที่ได้ (ผลเฉลยที่ได้จากการอธิบายตามธรรมชาติ) กล่าวคือเมื่อพบผลเฉลยที่ดีเฉพาะที่แล้ว ก็จะค้นหาให้ทั่วๆ ในบริเวณนั้นโดยใช้หน่วยความจำระยะยาว

ในส่วนนี้จะกล่าวถึงการค้นหาโดยใช้หน่วยความจำระยะยาว (long term memory) เพื่อค้นหาคำตอบใหม่ที่แตกต่างจากเดิม ซึ่งจะหยุดกระบวนการค้นหาของหน่วยความจำระยะสั้นแล้วเริ่มต้นกระบวนการค้นหาที่จุดใหม่ รูปแบบที่นิยมใช้ของหน่วยความจำระยะยาวคือ **หน่วยความจำเหตุการณ์วิกฤต (critical event memory)** เพื่อจดจำเหตุการณ์สำคัญที่ผ่านมาแล้วนำมาใช้เป็นข้อมูลสำหรับการสร้างสถานภาพต้องห้ามสำหรับจุดใหม่ที่จะใช้เป็นจุดเริ่มต้นของการค้นหาครั้งใหม่ และนอกจากนั้นหน่วยความจำเหตุการณ์วิกฤตนี้จะใช้กำหนดความหลากหลายอีกด้วย

สำหรับปัญหานี้กำหนดให้เหตุการณ์สำคัญคือ

- จุดเริ่มต้นของการค้นหาแต่ละครั้ง (รอบที่ 0 ในกรณีตัวอย่าง)
- จุดให้ค่าต่ำสุดเฉพาะที่ซึ่งเกิดขึ้นในการค้นหาแต่ละครั้งที่ให้ค่า $f(x)$ น้อยกว่าหรือเท่ากับจุดก่อนหน้าและจุดด้านหลัง (รอบที่ 5, 6, 9 ในกรณีของตัวอย่าง)
- ในตัวอย่างรอบที่ 9 คือรอบที่ให้ค่าต่ำสุด ดังนั้นเราต้องการเริ่มการค้นหาครั้งใหม่ ก่อนรอบนี้โดยไม่นำรอบที่ 9 นำมาพิจารณา

ในกรณีนี้เหตุการณ์สำคัญคือเหตุการณ์ที่เกิดในรอบที่ 0, 5, 6 เราจะรวบรวมข้อมูลทั้งหมดของทั้งสามรอบไว้ในหน่วยความจำระยะยาว ซึ่งก็คือกิ่ง (1,2), (1,4), (4,7), (6,7), (6,8), (8,9) และ (6,9) และในการนี้ที่ใช้หน่วยความจำตามความถี่ (frequency-based memory) เป็นหน่วยความจำระยะยาว เราจะใช้จำนวนครั้งเพื่อกำหนดความสำคัญของแต่ละกิ่งด้วย ในกรณีนี้สมมติว่าไม่นำความถี่มาพิจารณาจะได้ดังนี้

รอบที่ 0 ประกอบด้วย (1,2), (1,4), (4,7), (6,7)

รอบที่ 5 ประกอบด้วย (4,7), (6,7), (8,9), (6,8)

รอบที่ 6 ประกอบด้วย (4,7), (6,7), (8,9), (6,9)

จากนั้นจะให้กิ่งเหล่านี้มีสถานภาพเป็นภาวะต้องห้าม เพื่อใช้เป็นตัวป้องกันการสร้างจุดเริ่มต้นใหม่ที่มีกิ่งเหมือนกับกิ่งในหน่วยความจำนี้ อย่างไรก็ได้ในแต่ละขั้นตอนของการสร้างจุดเริ่มต้นใหม่นั้น (ในกรณีของตัวอย่างเราวิธีอัลกอริทึมต่อไปนี้จะทำทั้งหมด 4 ขั้นตอน – มี 4 กิ่ง) จะทำการป้องกันมากถ้าเป็นขั้นตอนต้นๆ และป้องกันน้อยถ้าเป็นขั้นตอนท้ายๆ ของการสร้างจุดเริ่มต้นใหม่ (เพราะหากป้องกันมากไปอาจทำให้ไม่สามารถสร้างจุดใหม่ได้เลย)

ในกรณีของตัวอย่าง กำหนดให้การป้องกันเป็นดังต่อไปนี้

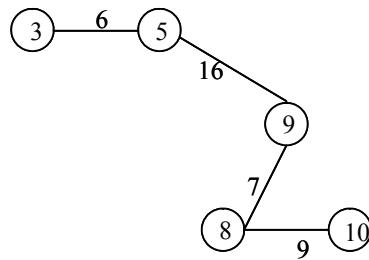
- ใน 2 ขั้นตอนแรก ห้ามมีกิ่งที่เราเก็บไว้ในหน่วยความจำแลย (คือห้ามมีกิ่งที่เราจำไว้ข้างบนของรอบ 0, 5, 6 ซึ่งก็คือกิ่ง (1,2), (1,4), (4,7), (6,7), (6,8), (8,9) และ (6,9) ดังนั้นเราจะเริ่มจาก (3,5) เป็นกิ่งแรก (ดูตารางที่ 2-10 ประกอบ)
- หลังจากนั้นยอมให้มีกิ่งในหน่วยความจำได้

ตารางที่ 2-10 กระบวนการหาจุดเริ่มต้นใหม่โดยใช้หน่วยความจำระยะยาว

ขั้นตอนที่	ตัวเลือก	กิ่งที่เลือก	นำหน้ารวม
1	(3,5)	(3,5)	6
2	(2,3), (3,4), (3,6), (5,6), (5,9), (5,12)	(5,9)	22
3	(2,3), (3,4), (3,6), (5,6), (5,12), (6,9), (8,9), (9,12)	(8,9)	29
4	(2,3), (3,4), (3,6), (5,6), (5,12), (6,8), (6,9), (7,8), (8,10), (9,12)	(8,10)	38

ในขั้นตอนที่ 2 เลือก (5,9) ก็ เพราะว่ามีนำหน้ากันน้อยที่สุดในทางเลือกทั้งหมดที่เป็นไปได้ ส่วนขั้นตอนที่ 3 และ 4 ก็ เช่นเดียวกัน เราจะใช้ต้นไม้ที่ได้ (ในรูปที่ 2-25) เป็นจุดเริ่มต้น

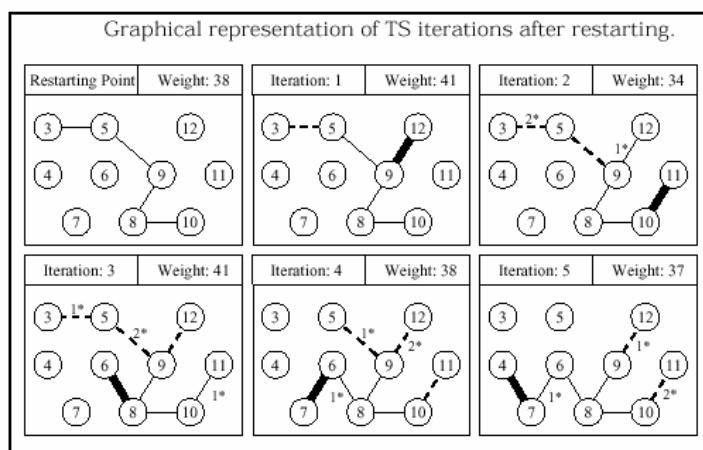
ใหม่ และใช้กระบวนการค้นหาด้วยหน่วยความจำระยะสั้น เช่นเดียวกับที่ผ่านมา ซึ่งจะได้ผลดังตารางที่ 2-11 และรูปที่ 2-26



รูปที่ 2-25 ต้นไม้ที่ได้สำหรับเริ่มการค้นหาใหม่

ตารางที่ 2-11 กระบวนการค้นหาตามเมื่อเริ่มจากผลเฉลยตัวใหม่

รอบที่	ระยะเวลาต้องห้าม		ก้าวเพิ่มเข้า	ก้าวออก	ค่าที่เปลี่ยน	น้ำหนัก
	1	2				
1			(9,12)	(3,5)	3	41
2	(9,12)	(3,5)	(10,11)	(5,9)	-7	34
3	(3,5), (10,11)	(5,9)	(6,8)	(9,12)	7	41
4	(5,9), (6,8)	(9,12)	(6,7)	(10,11)	-3	38
5	(9,12), (6,7)	(10,11)	(4,7)	(8,10)	-1	37



รูปที่ 2-26 ต้นไม้ที่ได้ในแต่ละรอบของกระบวนการค้นหาตามเมื่อเริ่มจากผลเฉลยตัวใหม่

ความสามารถเขียนอัลกอริทึมของการค้นหาตามโดยใช้แนวคิดด้านบนได้ดังตารางที่ 2-12

ตารางที่ 2-12 อัลกอริทึมการค้นหาตาม

Algorithm: Tabu Search

```

1. Choose an initial (possibly random) solution  $x \in X$ 
2.  $x^* := x$ ,  $k := 1$ 
3. Initialize tabu short-term memory and long-term
   memory
4. WHILE the stopping condition is not met DO {
   k := k+1
   Generate a candidate set  $N^*(x)$  including  $x'$ 
   with tabu-active which satisfies aspiration
   criteria.
   Find a best  $x' \in N^*(x)$ 
   IF local optimum reached THEN {
      IF no improvement made over a period THEN {
         Apply long term memory to restart the
         process, and find a new solution  $x'$ .
      }
   }
    $x := x'$ 
   Update tabu memory and adjust search parameters.
   IF  $f(x') < f(x^*)$  THEN  $x^* := x'$ 
}

```

เอกสารอ่านเพิ่มเติมและแบบฝึกหัด

การศึกษาเรื่องอิวาริสติกดูเพิ่มเติมได้ใน [Polya, 1971] อัลกอริทึม A* มีคุณสมบัติที่จะรับประกันได้ว่าเส้นทางที่ได้จากอัลกอริทึมจะเป็นเส้นทางสั้นสุด (ใช้ต้นทุนน้อยสุด) ถ้าใช้ฟังก์ชันอิวาริสติกที่ประเมินต่ำกว่าความจริง รายละเอียดและการพิสูจน์ดูได้จาก [Russell & Norvig, 1995] การค้นหาตามดูรายละเอียดเพิ่มเติมได้ใน [Glover & Laguna, 2002] นอกจากค้นหาในปรัชญาสถานะแล้ว ยังมีการแก้ปัญหาแบบที่เรียกว่า mean-end analysis ซึ่งสามารถหาอ่านได้ใน [Rich & Knight, 1991]

บรรณานุกรม

- Glover, F. and Laguna, M. (2002) Tabu search. *The Handbook of Applied Optimization*, P. M. Pardalos and M. G. C. Resende (Eds.), Oxford Academic Press. pp. 194-208.
- Polya, G. (1971) *How to Solve It: A New Aspect of Mathematical Method*. (Second Edition) Princeton University Press.
- Rich, E. and Knight, K. (1991) *Artificial Intelligence*. (International Edition), McGraw-Hill.
- Russell, S. and Norvig, P. (1995) *Artificial Intelligence: A Modern Approach*, Prentice Hall.

แบบฝึกหัด

- จงอธิบายว่าทำไม่เส้นทางไปสู่คำตอบที่ได้จากอัลกอริทึมการค้นหาแนวว่างก่อน จึงเป็นเส้นทางสั้นสุด โดยกำหนดให้ระยะห่างหรือค่าใช้จ่ายระหว่างสถานะสองตัวใดๆ มีค่าเท่ากันหมด
- จงเปรียบเทียบข้อดีข้อเสียระหว่างอัลกอริทึมค้นหา 3 ตัวนี้คือ อัลกอริทึมค้นหาแนวว่างก่อน อัลกอริทึมปีนเข้า และอัลกอริทึมตาม
- กำหนดให้สถานะเริ่มต้นและสถานะสุดท้ายของปัญหา 8-puzzle เป็นดังต่อไปนี้

<table border="1"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>8</td><td>5</td><td>6</td></tr> <tr><td>4</td><td>7</td><td></td></tr> </table>	1	2	3	8	5	6	4	7		<table border="1"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>4</td><td>5</td><td>6</td></tr> <tr><td>7</td><td>8</td><td></td></tr> </table>	1	2	3	4	5	6	7	8	
1	2	3																	
8	5	6																	
4	7																		
1	2	3																	
4	5	6																	
7	8																		

จงออกแบบพังก์ชันอิวิสติกที่ใช้กับการค้นหาปีนเข้าให้สามารถค้นหาจากสถานะเริ่มต้นและค้นพบสถานะสุดท้ายด้านบนนี้ได้

- พิจารณาปัญหาต่อไปนี้

9	1	2	3	4	5	4	3	2	1
8	2	3	4	5	6	5	4	3	2
7	3	4	0	0	0	0	0	4	3
6	4	5	0	7	8	7	0	5	4
5	5	6	0	8	9	8	0	6	5
4	4	5	0	7	8	7	0	5	4
3	3	4	0	0	0	0	0	4	3
2	2	3	4	5	6	5	4	3	2
1	1	2	3	4	5	4	3	2	1
	1	2	3	4	5	6	7	8	9

X

กำหนดให้

- สถานะหนึ่งๆ ในปริภูมิสถานะแทนด้วย $s(x,y)$ โดยที่ $x=1,\dots,9$ $y=1,\dots,9$
- สถานะเริ่มต้นคือ $s(1,1)$ และสถานะสุดท้ายคือ $s(5,5)$
- สถานะลูกของสถานะหนึ่งๆ ได้จากการเพิ่มหรือลดค่า 1 หน่วยของ x หรือ y เช่นสถานะลูกของ $s(3,4)$ คือ $s(2,4)$, $s(4,4)$, $s(3,5)$, $s(3,3)$
- ฟังก์ชันวิธีสติก h ของสถานะ s ได้ๆ แสดงโดยรูปด้านบน เช่น $h(s(1,1))=1$, $h(s(1,3))=3$, $h(s(3,3))=0$, $h(s(5,5))=9$ เป็นต้น

จงเขียนโปรแกรมอัลกอริทึมการออบหนี้夷จำลอง เพื่อค้นหาสถานะสุดท้ายจากสถานะเริ่มต้น

5. จงเขียนโปรแกรมเพื่อแก้ปัญหาการคำนวณค่าใช้จ่ายที่น้อยที่สุดในการจัดการแข่งขัน เล่นหมากเก็บชิงแชมป์โลก

- ในการแข่งขันเล่นหมากเก็บชิงแชมป์โลกนี้ มีผู้สมัครแข่งขันทั้งหมด n คนจาก คณะประเทศ ค่าใช้จ่ายในการจัดการแข่งขันสำหรับผู้สมัครจากประเทศ i กับ ประเทศ j เสียค่าใช้จ่ายเป็น c_{ij} โดยที่ $c_{ij} > 0$ จงเขียนโปรแกรมเพื่อคำนวณ ค่าใช้จ่ายที่น้อยที่สุดสำหรับการจัดการแข่งขันในรอบแรก (เฉพาะรอบแรกซึ่งมี การแข่งทั้งหมด $n/2$ ครั้ง ไม่ต้องคำนวณของรอบอื่นๆ)
- กำหนดให้ n เป็นเลขคู่ และโปรแกรมรับอินพุตจากไฟล์มีรูปแบบดังต่อไปนี้

8
0.0 5.2 2.0 11.0 9.4 11.9 13.1 15.1
5.2 0.0 4.6 9.9 10.7 13.2 14.8 16.2
2.0 4.6 0.0 9.1 8.1 10.6 11.9 13.8
11.0 9.9 9.1 0.0 6.4 7.7 9.8 9.7
9.4 10.7 8.1 6.4 0.0 3.7 5.3 6.8
11.9 13.2 10.6 7.7 3.7 0.0 3.3 4.5
13.1 14.8 11.9 9.8 5.3 3.3 0.0 3.8
15.1 16.2 13.8 9.7 6.8 4.5 3.8 0.0

บรรทัดแรกแสดงจำนวนผู้สมัครทั้งหมด บรรทัดที่เหลือแสดงค่าใช้จ่าย (หน่วย เป็นล้านบาท) ใน การจัดการแข่งขันสำหรับผู้สมัครคนหนึ่งกับคนอื่นที่เหลือ ตัวอย่างเช่น

- ในข้อมูลด้านบนมีผู้สมัครทั้งหมด 8 คนจาก 8 ประเทศ
- ค่าใช้จ่ายในการจัดการแข่งขันระหว่างผู้สมัครคนที่ 1 กับผู้สมัครคนที่ 2 เท่ากับ 5.2 ล้านบาท

-
- ค่าใช้จ่ายในการจัดการแข่งขันระหว่างผู้สมัครคนที่ 1 กับผู้สมัครคนที่ 3 เท่ากับ 2.0 ล้านบาท
 - ค่าใช้จ่ายในการจัดการแข่งขันระหว่างผู้สมัครคนที่ 8 กับผู้สมัครคนที่ 6 เท่ากับ 4.5 ล้านบาท
 - ค่าใช้จ่ายในการจัดการแข่งขันระหว่างผู้สมัครคนที่ 8 กับผู้สมัครคนที่ 7 เท่ากับ 3.8 ล้านบาท
-

การแทนความรู้โดยตรรกะเพรดิคเตต

3

การแทนความรู้ (knowledge representation) ในปัญญาประดิษฐ์มีหลายวิธีด้วยกัน เช่น กรอบ (frame) ข่ายงานความหมาย (semantic network) เป็นต้น ในบทนี้จะกล่าวถึงการแทนความรู้ที่มีการใช้กันอย่างแพร่หลายวิธีหนึ่งในปัญญาประดิษฐ์ ซึ่งเรียกว่า **ตรรกะเพรดิคเตต (predicate logic)**

3.1 ตรรกะเพรดิคเตต

สูตรรูปดี

ตรรกะเพรดิคเตตเป็นภาษาหนึ่งซึ่งมีภาษาสัมพันธ์ (syntax) และความหมาย (semantics) ของภาษาที่กำหนดขึ้นอย่างชัดเจน เราเรียกสูตรที่ถูกต้องตามภาษาสัมพันธ์ของภาษาหนึ่งว่า **สูตรรูปดี (well form formula)** ภาษาหนึ่งประกอบด้วยสัญลักษณ์พื้นฐานดังนี้คือ

- สัญลักษณ์แสดงเพรดิคเตต (predicate symbol) ใช้สัญลักษณ์ตัวอักษรตัวใหญ่ เช่น P, Q, R เป็นต้น
- สัญลักษณ์แสดงตัวแปร (variable symbol) ใช้สัญลักษณ์ตัวอักษรตัวเล็ก เช่น x, y, z เป็นต้น
- สัญลักษณ์แสดงฟังก์ชัน (function symbol) ใช้สัญลักษณ์ตัวอักษรตัวเล็ก เช่น f, g, h เป็นต้น
- สัญลักษณ์แสดงค่าคงที่ (constant symbol) ใช้สัญลักษณ์ตัวอักษรตัวใหญ่ เช่น A, B, C เป็นต้น
- เครื่องหมายวงเล็บ เช่น [], { }, () เป็นต้น

เครื่องหมายวงเล็บทั้งสามประเภทนี้ไม่มีความแตกต่างกันทางความหมาย เพรดิคเตตเป็นสัญลักษณ์ที่ใช้แสดงความสัมพันธ์ในโดเมนที่กล่าวถึง เช่น

สูตรอะตอม

FATHER(SOMCHAI, SOMSRI)

(3.1)

สูตร (3.1) เรียกว่า **สูตรอะตอม (atomic formula)** ซึ่งเป็นสูตรรูปดิที่เล็กที่สุดที่ถูกต้องตามวากยสัมพันธ์ของภาษาฯ นี้ สูตรอะตอมประกอบด้วยสัญลักษณ์เพรดิเกตในที่นี่คือ 'FATHER' ซึ่งแสดงความสัมพันธ์ 'พ่อ' ความสัมพันธ์นี้จะรับ **อาร์กิวเมนต์ (argument)** 2 ตัว คือ 'SOMCHAI' และ 'SOMSRI' ซึ่งอาร์กิวเมนต์ทั้งสองนี้เป็นค่าคงที่ สูตรอะตอมที่ถูกต้องตามวากยสัมพันธ์จะต้องประกอบด้วยเพรดิเกตและอาร์กิวเมนต์ตั้งแต่ 0 ตัวขึ้นไป โดยที่อาร์กิวเมนต์แต่ละตัวคันด้วยเครื่องหมาย ';' และอาร์กิวเมนต์ทั้งหมดต้องถูกคลุมด้วยเครื่องหมายวงเล็บ ลำดับของอาร์กิวเมนต์มีความสำคัญ ผู้เขียนต้องกำหนดลำดับเอง ในตัวอย่างนี้เราต้องการให้มีหมายความว่า คน 2 คนในโดเมนนี้ที่ชื่อ 'SOMCHAI' และ 'SOMSRI' มีความสัมพันธ์กันโดยที่ 'SOMCHAI' เป็นพ่อของ 'SOMSRI'

ภาษาฯสามารถใช้ตัวแปรเพื่อให้แทนถึงค่าคงที่ได้ ได้ และสามารถใช้พังก์ชันเพื่อรับหาพจน์ (term) หนึ่งจากพจน์อื่นๆ ได้ ดังแสดงในตัวอย่างด้านล่างนี้ตามลำดับ

FATHER(x, y)

(3.2)

x และ y แทนตัวแปร ซึ่งสูตรนี้แสดงว่า x เป็นพ่อของ y

HAS-MONEY(SOMCHAI, salary(SOMCHAI))

(3.3)

สูตรนี้แสดงความสัมพันธ์ 'HAS-MONEY' และมีพังก์ชัน 'salary' ที่ระบุหาพจน์จาก 'SOMCHAI' ซึ่งพจน์นี้อาจเป็นค่าคงที่ค่านึง เช่น 10850 เป็นต้น

การแปลความหมายของสูตรอะตอม

การแปลความหมาย (interpretation) คือการกำหนดค่าให้กับเพรดิเกต ค่าคงที่ ตัวแปร พังก์ชันในโดเมนนั้น ซึ่งการกำหนดค่าเหล่านี้จะเป็นตัวนิยามความหมายของสูตรอะตอม เช่นการแปลความหมายของสูตรที่ (3.3) คือการกำหนดให้ 'SOMCHAI' มีค่าคือคนที่ชื่อ สมชายในโดเมนของเรา กำหนดให้ 'salary(SOMCHAI)' มีค่าเท่ากับ 10850 บาท และกำหนดให้ 'HAS-MONEY' มีค่าเป็นการที่สมชายมีเงินเท่ากับ 10850 บาท เป็นต้น เมื่อเรานิยามการแปลความหมายแล้ว เราสามารถอภิปรายได้ว่าสูตรอะตอมตัวหนึ่ง มีค่าเป็น T (จริง) ถ้าความสัมพันธ์ที่ถูกแสดงด้วยสูตรนั้นเป็นจริงในโดเมนที่กล่าวถึง และจะมีค่าเป็น F (เท็จ) ถ้าไม่ใช่

ตัวเชื่อมและตัวบ่งปริมาณ

ภาษาหนึ่งมี **ตัวเชื่อม(connective)** และ **ตัวบ่งปริมาณ(quantifier)** เพื่อใช้เขียนความสัมพันธ์ได้ ซับซ้อนยิ่งขึ้น ตรงกับความต้องการของเรามากขึ้น ตัวเชื่อมที่มีในภาษาหนึ่งได้แก่

- และ แทนด้วยเครื่องหมาย \wedge
- หรือ แทนด้วยเครื่องหมาย \vee
- อิมไพล แทนด้วยเครื่องหมาย \Rightarrow
- นิสเซ แทนด้วยเครื่องหมาย \sim

ตัวเชื่อมเหล่านี้ทำหน้าที่เชื่อมสูตรอะตอมหลายตัวเข้าด้วยกัน เพื่อสร้างเป็นสูตรใหม่ที่ ซับซ้อนยิ่งขึ้น ตัวอย่างเช่นถ้าเราต้องการเขียนสูตรแทนประโยค 'John lives in a yellow house.' ก็อาจเขียนได้โดยใช้ตัวเชื่อม \wedge ดังนี้

$$\text{LIVE}(JOHN, \text{HOUSE-1}) \wedge \text{COLOR}(\text{HOUSE-1}, \text{YELLOW}) \quad (3.4)$$

ตัวเชื่อม \Rightarrow ใช้เขียนประโยคเงื่อนไข (เช่น if ... then ... เป็นต้น) เช่นถ้าเราต้องการเขียน สูตรแทนประโยค 'If the car belongs to John then it is green.' ก็อาจเขียนได้โดยใช้ ตัวเชื่อม \Rightarrow ได้ดังนี้

$$\text{OWNS}(JOHN, \text{CAR-1}) \Rightarrow \text{COLOR}(\text{CAR-1}, \text{GREEN}) \quad (3.5)$$

ตัวเชื่อม \sim ใช้เปลี่ยนค่าความจริงของสูตร เช่นถ้าต้องการเขียนสูตรแทนประโยค 'John did not write computer-chess.' ก็เขียนได้ดังนี้

$$\sim\text{WRITE}(JOHN, \text{COMPUTER-CHESS}) \quad (3.6)$$

เมื่อกำหนดการแปลความหมายของสูตรอะตอมแล้ว ค่าความจริงของสูตรที่ประกอบด้วย ตัวเชื่อมสามารถหาได้โดยใช้ตารางค่าความจริงด้านล่างนี้ โดยที่ P และ Q แทนสูตรอะตอม หนึ่งๆ

ตารางที่ 3-1 ตารางค่าความจริง

P	Q	$P \vee Q$	$P \wedge Q$	$P \Rightarrow Q$	$\sim P$
T	T	T	T	T	F
F	T	T	F	T	T
T	F	T	F	F	F
F	F	F	F	T	T

นอกจากตัวเชื่อมแล้ว เรายังสามารถใช้ตัวบ่งปริมาณในการเขียนสูตรได้ ตัวบ่งปริมาณที่ มีให้ในภาษาหนึ่งได้แก่

- ตัวบ่งปริมาณเอกภาพ (universal quantifier) แทนด้วยเครื่องหมาย \forall
- ตัวบ่งปริมาณมีอยู่ (existential quantifier) แทนด้วยเครื่องหมาย \exists

ตัวอย่างเขียนต้องการเขียนประโยค 'All elephants are gray.' ก็สามารถแสดงได้โดยสูตรด้านล่างนี้

$$\forall x (\text{ELEPHANT}(x) \Rightarrow \text{COLOR}(x, \text{GRAY})) \quad (3.7)$$

หรือในกรณีของประโยค 'There is a person who wrote computer-chess.' เขียนแทนด้วยสูตรด้านล่างนี้

$$\exists x (\text{WRITE}(x, \text{COMPUTER-CHESS})) \quad (3.8)$$

ปัญหาหนึ่งที่เกิดขึ้นในกรณีที่มีตัวบ่งปริมาณ普遍 quantifier ในสูตรก็คือ เราอาจค่าความจริงของสูตรนั้นไม่ได้ เช่นกำหนดให้สูตรเป็น ' $\forall x (P(x))$ ' และเมื่อให้การแปลความหมายของ 'P' และให้โดเมนของ x เป็นโดเมนอนันต์ เช่น x แทนเลขจำนวนจริง บางครั้งเราอาจหาค่าความจริงของสูตรนี้ไม่ได้ เนื่องจากเราไม่สามารถนำเลขจำนวนจริงมาทดสอบสูตรนี้ได้จนหมด แต่ถ้าหากว่ามีจำนวนจริงตัวหนึ่งที่ทำให้สูตรเป็นเท็จ ก็จะสรุปได้ว่าสูตรเป็นเท็จ แต่ถ้าหากว่าตราบเท่าที่จำนวนจริงที่นำมาทดสอบยังให้ค่าเป็นจริงอยู่ก็จะยังคงสรุปไม่ได้

ตรรกะเพรดิคเตต อันดับที่หนึ่ง

ตรรกะเพรดิคเตต อันดับที่หนึ่ง (first-order predicate logic) คือตรรกะเพรดิคเตตที่ไม่มีตัวบ่งปริมาณของสัญลักษณ์แสดงเพรดิคเตตหรือของสัญลักษณ์แสดงฟังก์ชัน ในบทนี้เราจะสนใจเฉพาะตรรกะเพรดิคเตต อันดับที่หนึ่งเท่านั้น ไม่สนใจตรรกะเพรดิคเตต อันดับสูง (high order predicate logic) เนื่องจากตรรกะเพรดิคเตต อันดับที่หนึ่งก็สามารถใช้เขียนความสัมพันธ์แทนความรู้ต่างๆ ได้อย่างกว้างขวางมาก และการคำนวณเชิงตรรกะของการแทนความรู้ประเภทนี้ก็ไม่สูงเกินไปนัก ตัวอย่างของสูตรที่เป็นตรรกะเพรดิคเตต อันดับที่หนึ่งก็เช่น $\forall x \forall y (y(x) \Rightarrow \text{COLOR}(x, \text{GRAY}))$ เป็นต้น ส่วนตัวอย่างของสูตรที่ไม่เป็นตรรกะเพรดิคเตต อันดับที่หนึ่งก็เช่น $\forall x \forall y (y(x) \Rightarrow \text{COLOR}(x, \text{GRAY}))$ เป็นต้น สังเกตว่าในกรณีนี้ 'y' เป็นตัวแปรแสดงเพรดิคเตต ซึ่งในกรณีเขียนนี้การแปลความหมายของสูตรนี้จะมีความยุ่งยากขับซ้อนมากขึ้น คือเราต้องหาทุกความสัมพันธ์ 'y' เพื่อนำมาตรวจสอบหาค่าความจริงของสูตร ' $y(x) \Rightarrow \text{COLOR}(x, \text{GRAY})$ '

3.2 กฎการอนุมาน

เมื่อเราไปสังเกตปรากฏการณ์ ต่างๆ หรือความสัมพันธ์ต่างๆ ในโดเมนที่เราสนใจและนำมาเขียนให้อยู่ในรูปของตรรกะเพรดิเกตได้แล้ว เราสามารถมองได้ว่า สิ่งที่เราเขียนในรูปของตรรกะเพรดิเกตคือ ความรู้ที่เราทราบในโดเมนนั้นๆ ขั้นตอนต่อไปจะเป็นข้อดีของการแทนความรู้คือ เราสามารถ **ถอดความ (inference)** เพื่อหาข้อเท็จจริงใหม่ๆ หรือผลสรุปที่แหงอยู่ในความรู้นั้นได้ เราเรียกกฎที่ใช้ออนุมานเพื่อหาความรู้ใหม่ว่า **กฎการอนุมาน (rule of inference)** ซึ่งจะทำหน้าที่สร้างสูตรใหม่จากสูตรหลายๆ ตัวที่มีอยู่ กฎการอนุมานมีอยู่หลากหลายชนิด ตัวอย่างของกฎการอนุมาน เช่น

กฎโมดัสโพเนนส์
และ
กฎเฉพาะจงตัวแปร

กฎโมดัสโพเนนส์ (Modus Ponens):

$$\frac{W1 \Rightarrow W2}{\frac{W1}{W2}}$$

กฎเฉพาะจงตัวแปรเอกภาพ (Universal Specialization):

$$\frac{\forall x (W(x))}{W(A)}$$

กฎข้อแรกหมายความว่า ถ้า ' $W1 \Rightarrow W2$ ' และ ' $W1$ ' เป็นจริงแล้ว สามารถสรุปได้ว่า ' $W2$ ' จะเป็นจริงด้วย ส่วนกฎข้อที่สองหมายความว่า ถ้า ' $\forall x (W(x))$ ' เป็นจริงแล้ว สามารถสรุปได้ว่า ' $W(A)$ ' จะเป็นจริงด้วย เมื่อ ' A ' เป็นค่าคงที่ในโดเมนที่กล่าวถึง เราเรียกสูตรใหม่ที่เกิดขึ้นเรียกว่า **ทฤษฎี (theorem)** และลำดับของกฎการอนุมานที่ใช้ในการสร้างทฤษฎีว่า **การพิสูจน์ (proof)** ของทฤษฎีนั้น

ตัวอย่างเช่นจากสูตร 2 ตัวคือ ' $\forall x (W1(x) \Rightarrow W2(x))$ ' และ ' $W1(A)$ ' เราสามารถถอดความได้ว่า ' $W2(A)$ ' เป็นจริงถ้าสูตร 2 ตัวบนเป็นจริง ' $W2(A)$ ' เป็นทฤษฎี ส่วนการพิสูจน์ก็คือ ลำดับของกฎการอนุมานด้านล่างนี้

ใช้กฎเฉพาะจงตัวแปรเอกภาพ
ได้ว่า
จากนั้นใช้กฎโมดัสโพเนนส์
ได้ว่า

$$\frac{\frac{\frac{\forall x (W1(x) \Rightarrow W2(x))}{W1(A) \Rightarrow W2(A)}}{W1(A)}}{W2(A)}$$

□

ปัญหาหนึ่งของการพิสูจน์ก็ดังเช่นที่แสดงในตัวอย่างข้างต้นนี้คือ จะรู้ได้อย่างไรว่าเวลาที่ใช้กฎ jealousy ของตัวแปรเอกภาพ จะต้องแทนด้วยค่าคงที่ตัวใด ในตัวอย่างข้างต้นเราแทนตัวแปร x ด้วยค่าคงที่ A และทำให้สามารถอนุมานต่อได้ เพราะเราจะได้ว่า ' $W1(A)$ ' ที่ด้านซ้ายมือในสูตร ' $W1(A) \Rightarrow W2(A)$ ' เท่ากันกับ ' $W1(A)$ ' ที่มีอยู่ แต่ถ้าเราแทน x ด้วยค่าคงที่อื่น เช่น B ก็จะไม่สามารถอนุมานต่อได้ ปัญหานี้เป็นปัญหาที่สำคัญที่เราต้องพิจารณาอย่างละเอียดถี่ถ้วน ดังนั้นส่วนต่อไปที่เราจะศึกษาเกี่ยวกับการแทนค่าและการทำให้เท่ากัน

3.3 การแทนค่าและการทำให้เท่ากัน

การแทนค่า (substitution) คือการแทน พจน์ (term) ให้กับตัวแปร พจน์หมายรวมถึงค่าคงที่ พังก์ชัน และตัวแปร สูตรที่ได้จากการทำการแทนค่าพจน์ในตัวแปรของสูตรใดๆ เรียกว่า ตัวอย่างการแทนค่า (substitution instance) ของสูตรนั้นๆ เช่นตัวอย่างการแทนค่าของสูตร

$$P(x, f(y), B) \quad (3.9)$$

ได้แก่

$$P(z, f(w), B) \quad (3.10)$$

และ

$$P(c, f(A), B) \quad (3.11)$$

เป็นต้น เราเขียนการแทนค่าอยู่ในรูปของเซตคู่ลำดับ

$$s = \{t_1/v_1, t_2/v_2, \dots, t_n/v_n\} \quad (3.12)$$

โดยที่ คู่ลำดับ t_i/v_i หมายถึงพจน์ t_i ถูกแทนค่าให้กับตัวแปร v_i เมื่อเราใช้การแทนค่า $s1$ และ $s2$

$$s1 = \{z/x, w/y\} \quad (3.13)$$

$$s2 = \{c/x, A/y\} \quad (3.14)$$

กระทำการกับสูตรที่ (3.9) จะได้สูตรที่ (3.10) และสูตรที่ (3.11) ตามลำดับ
เราเขียนสูตรที่ได้จากการแทนค่า s กับสูตร E ด้วย Es จากตัวอย่างที่แล้วจะได้ว่า

$$P(z, f(w), B) = P(x, f(y), B)s1 \quad (3.15)$$

และ

$$P(C, f(A), B) = P(x, f(y), B) s2 \quad (3.16)$$

จุดประสงค์หนึ่งของการแทนความรู้ในรูปของตรรกะเพรดิเกตคือ เมื่อเราเขียนความรู้อยู่ในรูปตรรกะเพรดิเกตแล้ว ทำให้เราสามารถอนุมานหาความรู้ใหม่ๆ ที่ແง່ງอยู่ในความรู้ที่มีอยู่ได้ และดังเช่นที่แสดงในกฎการอนุมานว่า ปัญหานี้ที่เราพบในการอนุมานหาความรู้ใหม่ๆ คือ การแทนค่าให้กับตัวแปรว่าจะต้องใช้ค่าคงที่ใดแทนค่าให้กับตัวแปรได้ เพื่อที่จะทำให้พจน์บางตัวเท่ากัน ซึ่งส่งผลให้การอนุมานสำเร็จ เราได้กล่าวถึงการแทนค่าไปแล้ว ส่วนต่อไปที่จะกล่าวคือ **การทำให้เท่ากัน (unification)**

เรา假設ว่า สูตร $E1$ และ $E2$ สามารถทำให้เท่ากัน (*unify*) ถ้ามีการแทนค่า s ที่ทำให้ $E1s = E2s$ และในกรณีนี้เรารอเรียก s ว่าเป็น **ตัวทำให้เท่ากัน (unifier)** ของ $E1$ และ $E2$ ตัวอย่างเช่น $P[x, f(y), B]$ และ $P[x, f(B), B]$ ทำให้เท่ากันได้โดยมีตัวทำให้เท่ากันคือ $s = \{A/x, B/y\}$ และผลของการทำให้เท่ากันคือ $P[A, f(B), B]$ สูตรสองตัวได้ๆ มักจะมีตัวทำให้เท่ากันมากกว่าหนึ่งตัว แต่ตัวที่เราสนใจคือตัวทำให้เท่ากันที่ใช้การแทนค่าไม่มากเกินความจำเป็น เราเรียกตัวทำให้เท่ากันแบบนี้ว่า **ตัวทำให้เท่ากันกว้างสุด – เอ็มจียู (most general unifier – mgu)** นิยามได้ดังนี้

g เป็นตัวทำให้เท่ากันกว้างสุดของ $E1$ และ $E2$ ก็ต่อเมื่อ ถ้ามี s ที่เป็นตัวทำให้เท่ากันอื่นของ $E1$ และ $E2$ และ จะต้องมีตัวทำให้เท่ากัน s' ที่ทำให้ $E1s = E1s'$ และ $E2s = E2s'$

จากตัวอย่างด้านบน เอ็มจียูของ $P[x, f(y), B]$ และ $P[x, f(B), B]$ คือ $\{B/y\}$ ซึ่งจะเห็นได้ว่าใช้การแทนค่าไม่มากเกินไป ต่างจาก s ($\{A/x, B/y\}$) ด้านบนที่มีการแทนค่าเกินความจำเป็น คือ ' A/x ' อัลกอริทึมสำหรับหาเอ็มจียูแสดงในตารางที่ 3-2 ต่อไปนี้

ตารางที่ 3-2 อัลกอริทึมการทำให้เท่ากัน

Algorithm: Unify(L1,L2)

```

2. IF L1 or L2 are both variables or constants THEN
    IF L1=L2 THEN return NIL
    ELSE IF L1 is a variable THEN
        IF L1 occurs in L2 THEN return {FAIL} ELSE
        return {L2/L1}
    ELSE IF L2 is a variable THEN
        IF L2 occurs in L1 THEN return {FAIL} ELSE
        return {L1/L2}
    ELSE return {FAIL}
3. IF the predicate or function symbols of L1 and L2
    are not identical
    THEN return {FAIL}
4. IF L1 and L2 have a different number of arguments
    THEN return {FAIL}
5. SUBST := NIL
6. FOR i := 1 TO number of arguments in L1 DO
    5.1 S := Unify(ith argument of L1, ith argument of L2)
    5.2 IF S contains FAIL THEN return {FAIL}
    5.3 IF S <> NIL THEN
        5.3.1 Apply S to the remainder of both L1 and
        L2
        5.3.2 SUBST := SUBST ∪ S
6. return SUBST

```

ตัวอย่างเช่นกำหนดให้ $L1 = P(A, x, h(g(z)))$ และ $L2 = P(z, h(y), h(y))$ เราต้องการทำให้ $L1$ เท่ากับ $L2$ โดยเรียกอัลกอริทึม $\text{Unify}(L1, L2)$ ซึ่งจะได้ขั้นตอนการทำดังนี้

- $L1$ และ $L2$ ไม่ใช่ตัวแปรหรือค่าคงที่ ดังนั้นจึงตรวจสอบด้วยขั้นตอนที่ 2 และ 3 ใน อัลกอริทึมว่า $L1$ กับ $L2$ มีเพรดิเคตตัวเดียวกัน ($= P$) และมีจำนวนอาร์กิวเม้นต์ เท่ากัน ($= 3$) จึงจะทำในขั้นตอนที่ 4 และ 5 ต่อไป ถ้าหากว่าสูตรสองตัวได้มี เพรดิเคตไม่เหมือนกันหรือมีจำนวนอาร์กิวเม้นต์ไม่เท่ากัน ก็จะไม่สามารถทำให้ เท่ากันได้
- ขั้นตอนที่ 4 กำหนดค่าเริ่มต้นให้กับผลลัพธ์ของการแทนค่าที่จะทำให้ $L1$ เท่ากับ $L2$ โดยเริ่มต้นให้เท่ากับเซตว่าง (แทนด้วย NIL ในอัลกอริทึม)
- ขั้นตอนที่ 5 เป็นขั้นตอนที่พยายามทำอาร์กิวเม้นต์ในตำแหน่งที่ต้องกันของ $L1$ และ $L2$ ให้เท่ากัน โดยเริ่มจากอาร์กิวเม้นต์ตัวที่ 1 ถึงตัวสุดท้าย

- เริ่มจากอาร์กิวเมนต์ตัวที่ 1 โดยเรียกอัลกอริทึม $S = \text{Unify}(A, z)$ ในการเรียกครั้งนี้จะเป็นการเรียกซ้ำ (recursive) ซึ่งจะตรวจสอบด้วยขั้นตอนที่ 1 พบว่า z เป็นตัวแปรและไม่ปรากฏใน A จึงคืนค่า $\{A/z\}$ ที่จุดนี้เราได้ $S = \{A/z\}$ เนื่องจาก S ไม่ประกอบด้วย FAIL และไม่เท่ากับ NIL เราจึงทำขั้นตอนที่ 5.3.1 โดยนำการแทนค่าตาม S ไปประกอบกับ $L1$ และ $L2$ ดังเดิม และได้ค่าตามนี้คือ

$$L1 = P(A, x, h(g(A)))$$

$$L2 = P(A, h(y), h(y))$$

สังเกตว่าอาร์กิวเมนต์ในตำแหน่งที่ 3 ของ $L1$ ซึ่งเดิมมี z อยู่ด้วยได้ถูกแทนค่าด้วย A เนื่องจากว่าตัวแปรตัวเดียวต้องถูกแทนค่าด้วยพจน์เดียวกันเสมอ เมื่อทำขั้นตอนที่ 5.3.2 จะได้ว่า $\text{SUBST} = \text{NIL} \cup \{A/z\} = \{A/z\}$

- ทำอาร์กิวเมนต์ตัวที่ 2 ของ $L1$ และ $L2$ ให้เท่ากัน โดยเรียกอัลกอริทึม $S = \text{Unify}(x, h(y))$ ในการเรียกครั้งนี้จะตรวจสอบด้วยขั้นตอนที่ 1 พบว่า x เป็นตัวแปรและไม่ปรากฏใน $h(y)$ ทำให้ได้ $S = \{h(y)/x\}$ เนื่องจาก S ไม่ประกอบด้วย FAIL และไม่เท่ากับ NIL เราจึงทำขั้นตอนที่ 5.3.1 โดยนำการแทนค่าตาม S ไปประกอบกับ $L1$ และ $L2$ ดังเดิม และได้ค่าตามนี้คือ

$$L1 = P(A, h(y), h(g(A)))$$

$$L2 = P(A, h(y), h(y))$$

เมื่อทำขั้นตอนที่ 5.3.2 จะได้ว่า $\text{SUBST} = \{A/z\} \cup \{h(y)/x\} = \{A/z, h(y)/x\}$

- ทำอาร์กิวเมนต์ตัวที่ 3 ของ $L1$ และ $L2$ ให้เท่ากัน โดยเรียกอัลกอริทึม $S = \text{Unify}(h(g(A)), h(y))$ ในการเรียกครั้งนี้เนื่องจาก h ของ $h(g(A))$ และ $h(y)$ เป็นพังก์ชันซึ่งตรวจสอบด้วยขั้นตอนที่ 2 และพบว่ามีจำนวนอาร์กิวเมนต์เท่ากัน เท่ากับ 1 จึงเรียกอัลกอริทึม $\text{Unify}(g(A), y)$ ซึ่งเมื่อทำสำเร็จจะคืนค่าเท่ากับ $\{g(A)/y\}$ ณ จุดนี้เราได้ S ของอาร์กิวเมนต์ตัวที่ 3 ของ $L1$ และ $L2$ เท่ากับ $\{g(A)/y\}$ เนื่องจาก S ไม่ประกอบด้วย FAIL และไม่เท่ากับ NIL เราจึงทำขั้นตอนที่ 5.3.1 โดยนำการแทนค่าตาม S ไปประกอบกับ $L1$ และ $L2$ ดังเดิม และได้ค่าตามนี้คือ

$$L1 = P(A, h(g(A)), h(g(A)))$$

$$L2 = P(A, h(g(A)), h(g(A)))$$

เมื่อทำขั้นตอนที่ 5.3.2 จะได้ว่า $\text{SUBST} = \{A/z, h(y)/x\} \cup \{g(A)/y\} = \{A/z, h(y)/x, g(A)/y\}$ ซึ่งเป็นผลลัพธ์สุดท้ายของการทำให้เท่ากันครั้งนี้

ในการทำให้เท่ากันนี้ เราสามารถทำให้พจน์ด้านล่างนี้คือ

x กับ $f(x)$

เท่ากันได้หรือไม่? สมมติว่าเราใช้การแทนค่า $\{f(A)/x\}$ ก็จะได้เป็น

$f(A)$ กับ $f(f(A))$

หรือถ้าแทนด้วย $\{f(f(A))/x\}$ ก็จะได้เป็น

$f(f(A))$ กับ $f(f(f(A)))$

ซึ่งจะเห็นว่าไม่สามารถทำให้เท่ากันได้ ในอัลกอริทึม Unify ข้างต้นนั้นมีการตรวจสอบที่ขั้นตอนที่ 1 ว่าถ้า $L1$ (ในที่นี้คือ x) เป็นตัวแปรและปรากฏใน $L2$ (ในที่นี้คือ $f(x)$) แล้ว จะคืนค่า FAIL หมายถึงว่าไม่สามารถทำให้เท่ากันได้

3.4 รีโซลูชัน

วิธีการอนุมานทางตรรกเพรดิคเตตได้กล่าวไปแล้วสองวิธีคือกฎโมดัลไฟเนนส์และกฎเจาะจงตัวแปรเอกภาพ ซึ่งกฎทั้งสองข้อนี้มีข้อจำกัดอยู่ เช่นกฎโมดัลไฟเนนส์จะใช้ได้กับสูตรที่ต้องอยู่ในรูปแบบที่กำหนดเท่านั้นคือ ตัวแปรอยู่ในรูป $W1 \Rightarrow W2$ และตัวที่สองเป็น $W1$ ซึ่งเหมือนกันกับด้านซ้ายมือของสูตรแรก หรือกฎเจาะจงตัวแปรเอกภาพก็ใช้ได้กับสูตรที่มีตัวแปรแล้วแทนที่ด้วยค่าคงที่เท่านั้น การใช้งานเจึงจำกัดเช่นกัน ในขณะที่ความรู้ที่เขียนอยู่ในตรรกเพรดิคเตตโดยทั่วไปมีหลากหลายรูปแบบ ทำให้การใช้กฎดังกล่าวไม่ก่อวังข่าวงเพียงพอหรือมีโอกาสพบรูปแบบที่กฎใช้ได้น้อยมาก

หัวข้อนี้จะกล่าวถึงการอนุมานอีกชิ้นหนึ่งซึ่งสามารถใช้ได้อย่างกว้างขวางเรียกว่า **รีโซลูชัน (resolution)** ที่สามารถใช้กับสูตรทุกด้วยที่เป็นอนุประโยค (clause)

อนุประโยคคือสูตรที่เป็น **การหรือของสัญลักษณ์ (disjunction of literals)** สัญลักษณ์คือสูตร

อะตอมซึ่งอาจมีเครื่องหมายนิเสธอยู่หน้าหรือไม่ก็ได้ กล่าวคืออนุประโยคก็คือสูตรอะตอมหลายตัวมาเชื่อมกันด้วยเครื่องหมาย \vee และสูตรอะตอมบางตัวอาจมีเครื่องหมายนิเสธอยู่บางตัวอาจไม่มี ตัวอย่างของอนุประโยคก็เช่น $P(x) \vee Q(x,y) \vee \neg R(A)$ เป็นต้น ในทางปฏิบัติสูตรที่เราเขียนแทนความรู้อาจไม่อยู่ในรูปของอนุประโยค ดังนั้นก่อนที่เราจะสามารถใช้รีโซลูชันเพื่ออนุมานได้นั้น เราจำเป็นต้องแปลงสูตรให้อยู่ในรูปของอนุประโยคก่อน โดยใช้ขั้นตอนพร้อมทั้งยกตัวอย่างการแปลงสูตรให้เป็นอนุประโยคดังต่อไปนี้

$(\forall x) \{P(x) \Rightarrow \{(\forall y) [P(y) \Rightarrow P(f(x,y))] \wedge \sim (\forall y) [Q(x,y) \Rightarrow P(y)]\}\}$

1. กำหนดเครื่องหมาย \Rightarrow : เปลี่ยนรูปของ $X \Rightarrow Y$ เป็น $\sim X \vee Y$

$$(\forall x) \{ \sim P(x) \vee \{ (\forall y) [\sim P(y) \vee P(f(x, y))] \wedge \sim (\forall y) [\sim Q(x, y) \vee P(y)] \}$$
2. ลดขอบเขตของเครื่องหมายนิเสธ: นิเสธแต่ละตัวจะมีขอบเขตไม่เท่ากัน ถ้าพนิเสธที่คุณบอกรวบกับ \sim ให้ลดขอบเขตให้แคบสุดโดยกระจายนิเสธเข้าไปข้างในบริเวณที่มันคุณอยู่ และเปลี่ยนเครื่องหมายอื่นๆ เป็นตรงข้ามให้หมด เช่นนิเสธของ ' \wedge ' เป็น ' \vee ' นิเสธของ ' \forall ' เป็น ' \exists ' ฯลฯ

$$(\forall x) \{ \sim P(x) \vee \{ (\forall y) [\sim P(y) \vee P(f(x, y))] \wedge (\exists y) [Q(x, y) \wedge \sim P(y)] \}$$
3. ทำตัวแปรเป็นมารตฐาน: เปลี่ยนชื่อตัวแปรตามขอบเขตของตัวบ่งปริมาณ เช่นถ้าเราพูดว่ามี y ซึ่กันสองที่ ให้เปลี่ยนชื่อตัวได้ตัวหนึ่ง และความหมายของสูตรที่ได้จะไม่เปลี่ยนไปจากเดิม

$$(\forall x) \{ \sim P(x) \vee \{ (\forall y) [\sim P(y) \vee P(f(x, y))] \wedge (\exists w) [Q(x, w) \wedge \sim P(w)] \}$$
4. กำหนดตัวบ่งปริมาณมีอยู่: แทนค่าตัวแปรด้วย **ฟังก์ชันสกอเลม (Skolem function)** ซึ่งเป็นฟังก์ชันที่แทนค่าตัวแปรตัวหนึ่งด้วยฟังก์ชันของตัวแปรอื่นๆ ที่ตัวแปรตัวนั้นขึ้นอยู่กับมัน ในกรณีของตัวอย่างสูตรในขั้นตอนที่ 3 ที่ทำแทนง่ายของ $(\forall x) \{ \sim P(x) \vee \{ (\forall y) [\sim P(y) \vee P(f(x, y))] \wedge (\exists w) [Q(x, w) \wedge \sim P(w)] \}$ ซึ่งจะอ่านได้ว่าสำหรับ x ทุกตัวจะมี w บางตัวที่ทำให้ $Q(x, w)$ และ $\sim P(w)$ เป็นจริง แสดงว่าถ้าเลือก x มาหนึ่งตัวจะต้องมี w 1 ตัวที่ทำให้สูตรเป็นจริง หมายความว่า w ขึ้นกับ x หรือเป็นฟังก์ชันของ x ซึ่งก็คือ $w = g(x)$ เมื่อ g เป็นฟังก์ชันสกอเลม

$$(\forall x) \{ \sim P(x) \vee \{ (\forall y) [\sim P(y) \vee P(f(x, y))] \wedge [Q(x, g(x)) \wedge \sim P(g(x))] \}$$
5. แปลงให้อยู่ในรูปแบบพรีเน็กซ์ (prenex form): ย้ายตัวบ่งปริมาณออกจากทุกตัวมาอยู่หน้าสุด และรูปแบบที่ได้ใหม่นี้เรียกว่ารูปแบบพรีเน็กซ์

$$(\forall x) (\forall y) \{ \sim P(x) \vee \{ [\sim P(y) \vee P(f(x, y))] \wedge [Q(x, g(x)) \wedge \sim P(g(x))] \}$$
6. จัดรูปของพรีเน็กซ์ใหม่ให้อยู่ในรูปทั่วไปแบบและ (conjunctive normal form): รูปที่สูตรทุกตัวเข้มกันด้วยเครื่องหมาย ' \wedge ' แต่ภายในสูตรมีแต่เครื่องหมาย ' \vee ' โดยใช้ความสมมูลของสูตรต่อไปนี้ $P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$

$$(\forall x) (\forall y) \{ [\sim P(x) \vee \sim P(y) \vee P(f(x, y))] \wedge [\sim P(x) \vee Q(x, g(x)) \wedge [\sim P(x) \vee \sim P(g(x))] \}$$

7. ละเครื่องหมายตัวบ่งปริมาณเอกพก: เนื่องจากว่า ณ จุดนี้ตัวแปรทุกตัวจะมีตัวบ่งปริมาณเป็นแบบเอกพกเท่านั้น
8. ที่จุดนี้เราจะได้อันุประโยคตั้งแต่ 1 ประโยคขึ้นไป โดยที่แต่ละประโยคเชื่อมกันด้วยเครื่องหมาย \wedge นำอนุประโยคเหล่านั้นมาเขียนเรียงกัน

$$(1) \sim P(x) \vee \sim P(y) \vee P(f(x, y))$$

$$(2) \sim P(x) \vee Q(x, g(x))$$

$$(3) \sim P(x) \vee \sim P(g(x))$$

9. เปลี่ยนชื่อตัวแปร: เปลี่ยนชื่อตัวแปรโดยตัวแปรเดียวกันที่ปรากฏในหลายอนุประโยคให้เขียนใหม่ด้วยตัวแปรคนละตัว

$$(1) \sim P(x_1) \vee \sim P(y) \vee P(f(x_1, y))$$

$$(2) \sim P(x_2) \vee Q(x_2, g(x_2))$$

$$(3) \sim P(x_3) \vee \sim P(g(x_3))$$

รีโซลูชันของอนุประโยคพื้นฐาน

ก่อนที่จะอธิบายถึงรีโซลูชันของอนุประโยคทั่วไป ขอกล่าวถึงรีโซลูชันของ **อนุประโยคพื้นฐาน (ground clause)** ก่อน ซึ่งจะง่ายกว่าของอนุประโยคทั่วไป อนุประโยคพื้นฐานคือ อนุประโยคที่ไม่มีตัวแปร การทำรีโซลูชันสำหรับอนุประโยคพื้นฐานจะรับอินพุตเป็น อนุประโยคพ่อแม่ (parent clause) 2 ประโยค และจะให้อันุประโยคเป็นเอาต์พุต 1 ประโยค สามารถแสดงได้ดังต่อไปนี้

$$\frac{P_1 \vee P_2 \vee \dots \vee P_n \\ \sim P_1 \vee Q_2 \vee \dots \vee Q_m}{P_2 \vee \dots \vee P_n \vee Q_2 \vee \dots \vee Q_m} \quad (3.17)$$

เราเรียกอนุประโยค 2 ประโยคบันว่า **อนุประโยคพ่อแม่ (parent clause)** ส่วน อนุประโยคที่เป็นผลลัพธ์เรียกว่า **รีโซลูชัน (resolvent)** การทำรีโซลูชันนี้ทำได้ดังแสดงใน (3.17) ด้านบนนี้ ซึ่งหมายความว่าถ้าเรามีอนุประโยค 2 ประโยค และพบว่าอนุประโยคทั้งสองมีสัญพจน์อยู่ 1 ตัวที่เหมือนกันทุกประการยกเว้นเครื่องหมายนิเสธที่ต่างกัน (สัญพจน์แรกไม่มีนิเสธแต่สัญพจน์ที่สองมีนิเสธ (P_1 กับ $\sim P_1$) หรือกลับกัน) ผลลัพธ์ที่ได้จะเป็น อนุประโยคที่นำสัญพจน์อื่นๆ ทั้งหมดของทั้งสองอนุประโยคมารวมกัน ยกเว้นสัญพจน์ที่เหมือนกัน (ต่างเฉพาะเครื่องหมาย) นั้น ผลที่ได้จากการทำรีโซลูชันจะเป็นจริงถ้าอนุประโยคพ่อแม่ทั้งสองเป็นจริง ตัวอย่างของการใช้รีโซลูชันแสดงในตารางที่ 3-3 ด้านล่างนี้

ตารางที่ 3-3 ตัวอย่างของการทำรีโซลูชัน

อนุประโยคพ่อแม่

รีโซลูเอนท์

$P \text{ และ } \sim P \vee Q$	Q
$P \vee Q \text{ และ } \sim P \vee Q$	Q
$P \vee Q \text{ และ } \sim P \vee \sim Q$	$Q \vee \sim Q \text{ และ } P \vee \sim P$
$\sim P \text{ และ } P$	NIL
$\sim P \vee Q \text{ และ } \sim Q \vee R$	$\sim P \vee R$

ดังจะเห็นได้จากตัวอย่างทั้งห้าด้านบน รีโซลูชันสามารถใช้ได้อย่างกว้างขวางมาก ดังเช่น ตัวอย่างแรกในตารางก็เป็นกรณีหนึ่งของรีโซลูชัน ซึ่งเท่ากับกฎโมดัลส์โพเน็นท์

รีโซลูชันทั่วไป

กรณีของรีโซลูชันทั่วไปที่กระทำกับอนุประโยคที่มีตัวแปรด้วยนั้น ขั้นตอนจะซับซ้อน กว่าเดิม ซึ่งเราต้องใช้การทำให้เท่ากันร่วมด้วยเพื่อทำให้อนุประโยคพ่อแม่ประกอบด้วย **สัญพจน์เติม (complimentary literals)** สัญพจน์เติมคือสัญพจน์ที่เหมือนกันทุก ประการเว้นแต่ว่าตัวหนึ่งมีเครื่องหมายนิเสธส่วนอีกด้วยไม่มี เช่น $\sim Q(z)$ กับ $Q(z)$ เป็นต้น ใน การอธิบายการทำรีโซลูชันทั่วไป จะเขียนแทนอนุประโยคด้วยเซตของสัญพจน์ เช่น $\sim P(z, f(A)) \vee \sim Q(z)$ เขียนแทนด้วย $\{\sim P(z, f(A)), \sim Q(z)\}$ เป็นต้น ขั้นตอนของรีโซลูชันทั่วไป ทำตามขั้นตอนต่อไปนี้

- กำหนดให้อนุประโยคพ่อแม่เป็น $\{L_i\}$ และ $\{M_i\}$
- เลือก $\{l_i\}$ และ $\{m_i\}$ ที่เป็นเซตย่อยของ $\{L_i\}$ และ $\{M_i\}$ ตามลำดับ ซึ่งมี s ที่ เป็นเอ็มจีจูของ $\{l_i\}$ และ $\{\sim m_i\}$ (หมายความว่าเซตย่อยทั้งสองจะต้องสามารถทำ ให้เป็นสัญพจน์เติมได้)
- จะได้ว่ารีโซลูเอนท์ของอนุประโยคพ่อแม่ $\{L_i\}$ กับ $\{M_i\}$ คือ $\{L_i\} - \{l_i\} \cup \{M_i\} - \{m_i\}$

โดยทั่วไปรีโซลูเอนท์จากการรีโซลูชันทั่วไปอาจมีได้มากกว่าหนึ่งประยุกต์ขึ้นกับการเลือก เซตย่อยที่จะมาทำให้เป็นสัญพจน์เติมเต็ม

ตัวอย่างของการทำรีโซลูชันทั่วไปแสดงดังด้านล่างนี้

- กำหนดให้ $\{L_i\} = \{P[x, f(A)], P[x, f(y)], Q(y)\}$ $\{M_i\} = \{\sim P[z, f(A)], \sim Q(z)\}$ (กล่าวคือต้องการทำรีโซลูชันระหว่างอนุประโยค $P[x, f(A)] \vee P[x, f(y)] \vee Q(y)$ กับ $\sim P[z, f(A)] \vee \sim Q(z)$)
- เลือก $\{l_i\} = \{P[x, f(A)]\}$ และ $\{m_i\} = \{\sim P[z, f(A)]\}$ โดยที่ $s = \{z/x\}$

- ได้ไวร์โซเวนท์เท่ากับ $(\{P[x, f(A)], P[x, f(y)], Q(y)\} - \{p(x, f(A))\})\{z/x\} \cup (\{\sim P[z, f(A)], \sim Q(z)\} - \{\sim P[z, f(A)]\})\{z/x\} = \{P[z, f(y)], Q(y), \sim Q(z)\}$
- แต่ถ้าเราเลือก $\{i\} = \{[p(x, f(A)], P[x, f(y)]\}$ $\{m_i\} = \{\sim P[z, f(A)]\}$ จะได้ไวร์โซเวนท์เท่ากับ $\{Q(A), \sim Q(z)\}$

การปฏิเสธแบบไวร์โซลูชัน

จากที่ได้แสดงให้เห็นข้างต้นว่าการทำไวร์โซลูชันจะทำให้เราหาความรู้ที่แฝงอยู่ในความรู้ที่มีอยู่ได้ และในหลาย ๆ กรณีเรามีความรู้ที่แสดงอยู่ในรูปตรรกะเพรติเกตและเราต้องการพิสูจน์อนุประโยคด้วยไม่ได้ ว่าเป็นผลสรุปของความรู้ที่มีอยู่หรือไม่ วิธีการพิสูจน์ก็อาจทำโดยการเลือกอนุประโยคพ่อแม่ 2 ประโยคแล้วหารีโซเวนท์ ถ้ารีโซเวนท์เป็นความรู้ใหม่ที่เราต้องการพิสูจน์ก็แสดงว่าเราพิสูจน์สำเร็จ ถ้าไม่ใช่เราก็อาจเลือกอนุประโยคพ่อแม่ 2 ประโยคอื่น ๆ แล้วลองทำไวร์โซลูชันดู หรืออาจนำรีโซเวนท์ที่ได้ก่อนหน้านี้มาจับคู่กับอนุประโยคอื่นเพื่อเป็นอนุประโยคพ่อแม่แล้วทำไวร์โซลูชันต่อไป อย่างไรก็ได้วิธีการพิสูจน์แบบนี้อาจไม่มีเป้าหมายที่ชัดเจน ทำให้การพิสูจน์เสียเวลาในการคำนวณมาก เรามีวิธีการซึ่งเน้นที่เป้าหมายในการพิสูจน์เพื่อทำให้การพิสูจน์ทำได้ดีขึ้น เราเรียกวิธีการที่จะนำเสนอว่า

ความขัดแย้ง

การปฏิเสธแบบไวร์โซลูชัน (*resolution refutation*) ก่อนอื่นของกล่าวถึง **ความขัดแย้ง** (*contradictory*) ที่ใช้ในการพิสูจน์โดยการปฏิเสธแบบไวร์โซลูชันดังนี้

สัญพจน์ 2 ตัวใด ๆ จะขัดแย้งกัน ถ้าตัวหนึ่งสามารถทำให้เท่ากันนิเสธของอีกตัวหนึ่งได้ เช่น $MAN(x)$ กับ $\sim MAN(Spot)$ ขัดแย้งกัน เนื่องจาก $MAN(x)$ ทำให้เท่ากับ $MAN(Spot)$ ได้

วิธีการปฏิเสธแบบไวร์โซลูชันคือ การที่จะพิสูจน์ว่าสูตร W เป็นผลสรุปของเซตของสูตร K ทำได้โดยการพิสูจน์ว่า $K \cup \{\sim W\}$ ขัดแย้งกัน

ตัวอย่างเช่น

- กำหนดให้ $K = \{MAN(Marcus), \sim MAN(x) \vee MORTAL(x)\}$
- กำหนดให้ $W = MORTAL(Marcus)$
- ได้ว่า $K \cup \{\sim W\} = \{MAN(Marcus), \sim MAN(x) \vee MORTAL(x), \sim MORTAL(Marcus)\}$
- จากการทำไวร์โซลูชันกับอนุประโยค 2 ประโยคล่าง เราได้ $\sim MAN(Marcus)$ ซึ่งขัดแย้งกับอนุประโยคที่ 1 ซึ่งแสดงว่า $MORTAL(Marcus)$ เป็นผลสรุปของ K

การพิสูจน์ในลักษณะนี้จะมีเป้าหมายในการพิสูจน์ กล่าวคือ $\sim W$ จะเป็นตัวที่เราเลือกเพื่อไปการทำไวร์โซลูชันกับอนุประโยคอื่น แล้วนำรีโซเวนท์ที่ได้ไปประทำไวร์โซลูชันกับ

อนุประโยคอื่นๆ ต่อไปตามลำดับจนกระทั่งพบความขัดแย้ง ตารางด้านล่างนี้แสดงอัลกอริทึมการปฏิเสธแบบรีโซลูชัน

ตารางที่ 3-4 อัลกอริทึมการปฏิเสธแบบรีโซลูชัน

Algorithm: Resolution Refutation

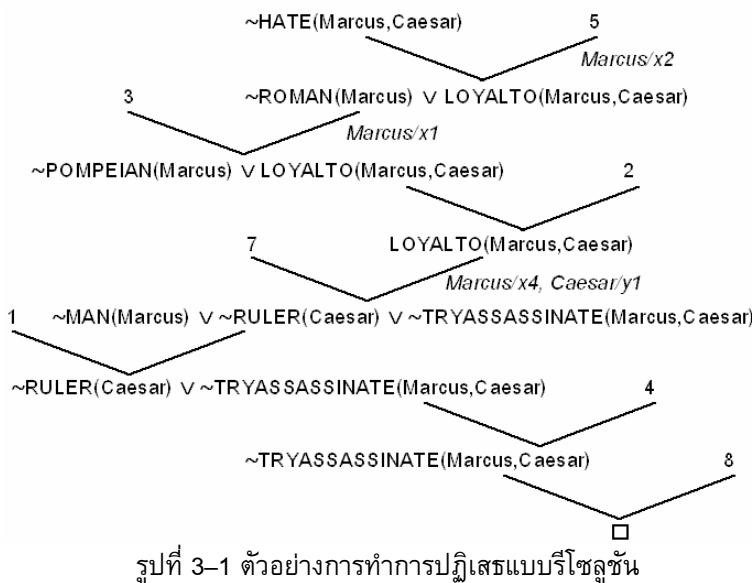
1. Convert all the statements of F to clause form.
2. Negate P and convert the result to clause form. Add it to the set of clauses obtained in 1.
3. Let $Clauses$ be the set of clauses. /* $F \cup \{\sim P\}$ */
4. **UNTIL** (NIL is a member of $Clauses$) or
($Clauses$ do not change) **DO**
 - 4.1 Select C_i and C_j which are resolvable.
 - 4.2 Calculate the resolvent of C_i and C_j , and call it R_{ij} .
 - 4.3 $Clauses := Clauses \cup \{R_{ij}\}$

อัลกอริทึมด้านบนนี้เริ่มต้นด้วยการแปลงสูตรใดๆ ให้อยู่ในรูปของอนุประโยค แล้วเปลี่ยน P ซึ่งเป็นสูตรที่เราต้องการพิสูจน์ให้อยู่ในรูปของนิเสธแล้วเพิ่มเข้าไปใน F จากนั้นจึงทำรีโซลูชันในขั้นตอนที่ 4 จนกระทั่งพบความขัดแย้ง หรือ $Clauses$ ไม่เปลี่ยนแปลง ในกรณีที่พบความขัดแย้งอัลกอริทึมจะหยุดและได้ว่า P เป็นผลสรุปของ F เมื่อสังเกตในขั้นตอน 4.3 จะเห็นได้ว่า $Clauses$ จะขยายตัวเรื่อยๆ ทุกครั้งที่ได้รีโซลูชันตัวใหม่ อย่างไรก็ได้หากพบว่าไม่มีรีโซลูชันใหม่เกิดขึ้นอีกหรือไม่สามารถทำรีโซลูชันได้อีก $Clauses$ จะหยุดขยายตัว ซึ่งเป็นกรณีอีกหนึ่งกรณีที่อัลกอริทึมนี้จะหยุด และในกรณีนี้ได้ว่า P ไม่ใช่ผลสรุปของ F อย่างไรก็ได้ในบางครั้งเราอาจพบกรณีที่ $Clauses$ ยังคงขยายตัวต่อเรื่อยๆ ซึ่งในกรณีเช่นนี้เราไม่สามารถสรุปอะไรได้ กล่าวคือ P อาจเป็นผลสรุปของ F หรือไม่ก็ได้

ตัวอย่างการทำรีโซลูชัน

- กำหนดอนุประโยคด้านล่างนี้ให้
 1. MAN(Marcus)
 2. POMPEIAN(Marcus)
 3. $\sim POMPEIAN(x_1) \vee ROMAN(x_1)$
 4. RULER(Caesar)
 5. $\sim ROMAN(x_2) \vee LOYALTO(x_2, Caesar) \vee HATE(x_2, Caesar)$
 6. LOYALTO(x_3, f1(x_3))
 7. $\sim MAN(x_4) \vee \sim RULER(y_1) \vee \sim TRYASSASSINATE(x_4, y_1) \vee \sim LOYALTO(x_4, y_1)$
 8. TRYASSASSINATE(Marcus, Caesar)
- ต้องการพิสูจน์ $HATE(Marcus, Caesar)$

- เติม ~HATE(Marcus, Caesar) เข้าไปในฐานความรู้แล้วทำรีโซลูชันดังแสดงในรูปที่ 2-14 ต่อไปนี้



รูปที่ 3-1 ตัวอย่างการทำการปฏิเสธแบบรีโซลูชัน

หมายเหตุในรูปด้านบนแสดงหมายเหตุประโยคที่เลือกมาทำรีโซลูชัน และเครื่องหมาย ‘∨’ แสดงการทำรีโซลูชันของอนุประโยค 2 ประโยคที่อยู่ด้านบนของเครื่องหมายส่วนรีโซลูชันที่ค่อนประโยคด้านล่างของเครื่องหมาย ที่ด้านข้างของเครื่องหมายแสดงการแทนค่าที่ทำให้อนุประโยคพ่อแม่สามารถทำรีโซลูชันกันได้ จากตัวอย่างแสดงให้เห็นว่า HATE(Marcus, Caesar) เป็นผลสรุปของฐานความรู้ที่มีอยู่

เอกสารอ่านเพิ่มเติมและแบบฝึกหัด

หนังสือของ Lloyd [Lloyd, 1984] แม้ว่าจะตีพิมพ์ไว้นานแล้ว แต่ก็เป็นหนังสือที่อธิบายเรื่องตรรกะเพรดิคเตตไว้อย่างดีมาก นอกจากนั้นหนังสือ [Genesereth & Nilsson, 1987] ได้แสดงการใช้ตรรกะเพรดิคเตตในการแก้ปัญหาต่างๆ ของปัญญาประดิษฐ์ได้อย่างดีเยี่ยม

บรรณานุกรม

- Lloyd, J. W. (1984) *Foundations of Logic Programming*, Springer-Verlag.
 Genesereth, M. R. and Nilsson, N. J. (1987) *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann.

แบบฝึกหัด

1. เน้นยามตัวเชื่อมตัวใหม่ในตรรกะเพรดิเกตเรียกว่า **exclusive-or** เขียนแทนด้วย **สัญลักษณ์** \oplus โดยมีตารางค่าความจริงดังนี้

P	Q	$P \oplus Q$
T	T	F
T	F	T
F	T	T
F	F	F

จงเขียนสูตรที่มีความหมายสมมูลกับ $P \oplus Q$ โดยใช้ตัวเชื่อม \wedge , \vee และ \sim เท่านั้น

2. จงบอกว่าคู่ของสูตรในแต่ละข้ออยู่ต่อไปนี้ว่าสามารถทำให้เท่ากันได้หรือไม่ ถ้าได้ให้เขียนแสดงເວັມຈີໍຍຂອງສູດທັງສອງ (ตัวอักษรເລີກແສດງตัวແປຣหรือຟັງກົດ ตัวอักษรໃໝ່ແກ່ນຫຼືເພື່ອພົຈິເຕີຕໍ່ຫຼືອຳນວຍກີ່)

- 2.1 $P(x, B, B) \quad P(A, y, z)$
 2.2 $P(g(f(v)), g(u)) \quad P(x, x)$
 2.3 $P(x, f(x)) \quad P(y, y)$
 2.4 $R(f(y), x) \quad R(x, f(B))$
 2.5 $R(f(y), y, x) \quad R(x, f(A), f(v))$

3. พิจารณาประโยคต่อไปนี้

All horses are faster than every dog.

There is a greyhound that is faster than every rabbit.

If x is faster than y and y is faster than z , then x is faster than z .

If x is a greyhound then x is a dog.

HaHa is a horse.

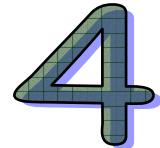
RaRa is a rabbit.

(a) จงเขียนประโยคด้านบนทั้งหมดให้อยู่ในรูปของสูตรทางตรรกะเพรดิเกต

(b) จงแปลงสูตรที่ได้ให้อยู่ในรูปของอนุประโยค

(c) จงพิสูจน์ว่า "HaHa is faster than RaRa." โดยใช้การปฏิเสธແບບຮູບຈຸດ

โปรดีอกเบื้องต้น



ภาษาโปรดีอกมาจากคำว่าการโปรดแกรมในตรรกศาสตร์ (PROLOG – PROgramming in LOGic) เป็นภาษาคอมพิวเตอร์ที่ใช้ในการแก้ปัญหาทางด้านสัญลักษณ์ (symbol) โดยใช้พื้นฐานของตรรกศาสตร์ โปรดีอกเป็นภาษาคอมพิวเตอร์ขั้นสูงที่มีจุดเด่นเหนือภาษาชั้นสูงตัวอื่นๆ เกี่ยวกับการจัดการเรื่องความสัมพันธ์ของสัญลักษณ์ต่างๆ นอกจากนี้ โปรดีอกยังมีลักษณะเป็นเอกสารลักษณ์พิเศษของตัวเองอีกด้วยที่เป็นภาษาเชิงพรรณนา (descriptive language) ซึ่งแตกต่างจากภาษาชั้นสูงทั่วไปประเภทภาษาเชิงกระบวนการคำสั่ง (procedural language) อย่างสิ้นเชิง ดังจะกล่าวต่อไปในบทนี้

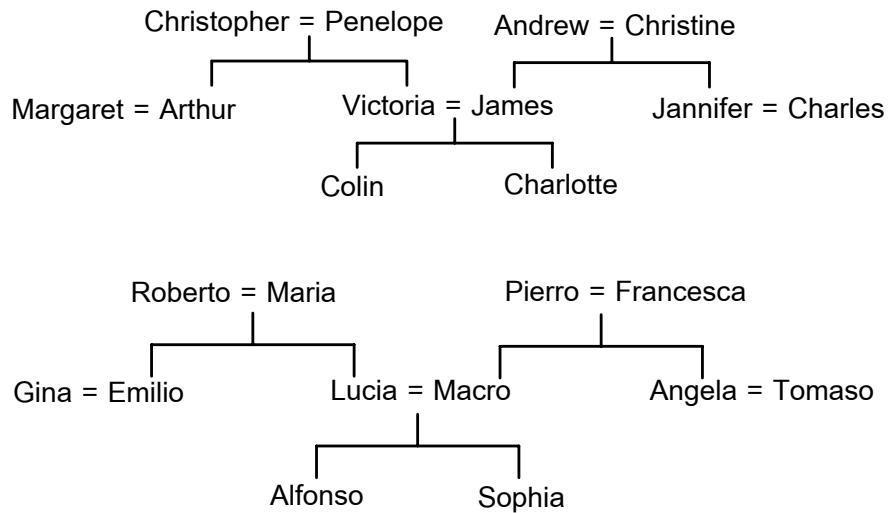
4.1 องค์ประกอบของโปรดีอก

องค์ประกอบที่สำคัญของภาษาโปรดีอกประกอบด้วย **ข้อเท็จจริง (fact)** **ข้อคำถาม (query)** **ตัวแปร (variable)** **สันฐาน (conjunction)** และ **กฎ (rule)** ใช้อธิบายวัตถุ (object) และความสัมพันธ์ (relation) ระหว่างวัตถุ เราใช้โปรดีอกในการจัดการกับตรรกศาสตร์ที่กล่าวไปในบทที่ 3 โปรดีอกหมายความว่าเป็นโปรแกรมที่อธิบายความสัมพันธ์และหาข้อสรุปเกี่ยวกับความสัมพันธ์นั้น แต่ไม่หมายความกับการนำมาระบบทางคณิตศาสตร์ โปรดีอกประกอบด้วยสามส่วนหลักคือ

ข้อเท็จจริง
กฎ
และ
ข้อคำถาม

- ข้อเท็จจริง: ใช้อธิบายเกี่ยวกับข้อเท็จจริงที่เกี่ยวข้องกับความสัมพันธ์
- กฎ: ใช้เขียนความสัมพันธ์ที่เกี่ยวข้องในโดเมน
- ข้อคำถาม: เมื่อใส่ข้อมูลครบถ้วนแล้ว ก็สามารถตั้งข้อคำถามเพื่อให้โปรดีอกหาคำตอบให้ได้

พิจารณาแผนภาพต้นไม้ครอบครัวในรูปที่ 4-1



รูปที่ 4-1 ต้นไม้ครอบครัว

เครื่องหมาย ‘ $A = B$ ’ ในรูปหมายถึง A แต่งงานกับ B และต้นไม้นี้แสดงความสัมพันธ์ภายในครอบครัว 2 ครอบครัว เช่น Christopher แต่งงานกับ Penelope มีลูกชื่อ Arthur กับ Victoria คนชื่อ James แต่งงานกับ Victoria มีลูกชื่อ Colin กับ Charlotte และ Colin เป็นหลานของ Christopher กับ Penelope เป็นต้น

จากความสัมพันธ์ในรูป เราอาจไม่จำเป็นต้องเขียนความสัมพันธ์ที่เกิดขึ้นทั้งหมด โดยเขียนแค่ความสัมพันธ์หลัก ส่วนความสัมพันธ์อื่นๆ ที่เหลือก็เขียนให้อยู่ในรูปความสัมพันธ์หลัก แล้วให้ปรับลักษณะตามให้ สมมติว่าเราต้องการเขียนความสัมพันธ์ภายในครอบครัว ด้วยภาษาโปรแกรมจะสามารถเขียนได้ดังจะกล่าวต่อไป

ข้อเท็จจริง

จากตัวอย่างในรูปที่ 4-1 สมมติว่าเราเลือกความสัมพันธ์ฟ่อแม่และความสัมพันธ์เพศมาเป็นความสัมพันธ์หลัก เราเขียนข้อเท็จจริงได้ดังตารางที่ 4-1

ตารางที่ 4-1 ข้อเท็จจริงของต้นไม้ครอบครัวในรูปที่ 4-1

```

01: parent(christopher, arthur).
02: parent(christopher, victoria).
03: parent(penelope, arthur).
04: parent(penelope, victoria).
05: parent(andrew, james).
06: parent(andrew, jennifer).
07: parent(christine, james).
08: parent(christine, jennifer).
09: parent(victoria, colin).
10: parent(victoria, charlotte).
11: parent(james, colin).
12: parent(james, charlotte).
13: parent(roberto, emilio).
14: parent(roberto, lucia).
15: parent(maria, emilio).
16: parent(maria, lucia).
17: parent(pierro, macro).
18: parent(pierro, angela).
19: parent(francesca, macro).
20: parent(francesca, angela).
21: parent(lucia, alfonso).
22: parent(lucia, sophia).
23: parent(macro, alfonso).
24: parent(macro, sophia).
25: male(christopher).
26: male(andrew).
27: male(arthur).
28: male(james).
29: male(charles).
30: male(colin).
31: male(roberto).
32: male(pierro).
33: male(emilio).
34: male(macro).
35: male(tomaso).
36: male(alfonso).
37: female(penelope).
38: female(christine).
39: female(margaret).
40: female(victoria).
41: female(jennifer).
42: female(charlotte).
43: female(maria).
44: female(francesca).
45: female(gina).
46: female(lucia).
47: female(angela).
48: female(sophia).

```

ตัวเลข 01:...48: ใน ตารางที่ 4-1 แสดงหมายเลขอารทัดและไม่ต้องเขียนลงในโปรแกรม แต่ละบรรทัดในตารางคือ ข้อเท็จจริง 1 ข้อ เช่น 'parent(christopher, arthur).' โดยมี 'parent' เป็นชื่อเพредิคเต (predicate) และมี 'christopher' กับ 'arthur' เป็น อาร์กิวเมนต์ (argument) ภายในเครื่องหมายงเล็บ ข้อเท็จจริงทุกข้อต้องจบด้วยเครื่องหมายมหัพภาค ' .' ในตารางมี

ข้อเท็จจริงแสดงความสัมพันธ์ 'parent', 'male' และ 'female' ทั้งหมด 24 ข้อ 12 ข้อ และ 12 ข้อตามลำดับ ในภาษาโปรล็อกทั้งชื่อเพรดิคเตตและอาร์กิวเมนต์ที่เป็นค่าคงที่ต้องขึ้นต้นด้วยตัวอักษรเล็ก

ค่าคงที่

ค่าคงที่ (constant) ในโปรล็อกมี 2 ประเภทหลักๆ คือ

อะตอม
และ
ตัวเลข

ข้อคำถาม

เมื่อเรายูไนตัวแปลภาษาโปรล็อก (Prolog interpreter) จะเห็นสัญลักษณ์ '?' ซึ่งแสดงให้ป้อนข้อคำถาม เราสามารถป้อนข้อคำถามได้ เช่นถ้าต้องการรู้ว่า 'christopher' เป็นพ่อแม่ของ 'arthur' หรือไม่ก็ป้อนดังนี้ (โดยเราต้องบรรจุ (load) ข้อเท็จจริงที่เขียนไว้ในตารางที่ 4-1 ลงในตัวแปลภาษาโปรล็อกก่อน)

?- parent(christopher, arthur).

Yes

เพื่อให้เห็นชัดเจนว่าข้อความส่วนใดเป็นของตัวแปลภาษาโปรล็อก ส่วนใดเป็นข้อความส่วนที่ผู้ใช้ป้อน ในที่นี้ขอใช้ตัวอักษรหนาและตัวอักษรปกติแสดงข้อความของตัวแปลภาษาโปรล็อกและส่วนของผู้ใช้ตามลำดับ

คำตอบจากโปรล็อกในกรณีนี้เป็น 'Yes' หมายความว่าข้อคำถามเป็นจริง การหาคำตอบของโปรล็อกจะใช้วิธีการจับคู่หรือเปรียบเทียบกับข้อเท็จจริงที่เราป้อนไว้ตั้งแต่แรก ซึ่งในกรณีนี้ข้อคำถามตรงกับข้อเท็จจริงที่ป้อนไว้ จึงให้คำตอบเป็น 'Yes' หรือถ้าเราป้อนข้อคำถามที่ไม่ตรงกับข้อเท็จจริงที่ให้ไว้ คำตอบก็จะเป็น 'No' ดังนี้

?- parent(andrew,victoria).

No

ตัวแปร

ในโปรล็อกเราสามารถเขียนตัวแปรเพื่อใช้แสดงวัตถุใดๆ ได้ โดยขึ้นต้นด้วยตัวอักษรใหญ่ เช่นถ้าต้องการถามว่า ใครเป็นพ่อแม่ของ 'christopher' ก็ใช้ข้อคำถามดังนี้

?- parent(christopher,X).

X = arthur;

X = victoria;

No

เมื่อเราตั้งข้อคำถาม 'parent(christopher,X)' เพื่อให้โปรแกรมหาคำตอบบนสำหรับตัวแปร X โปรแกรมจะค้นหาข้อเท็จจริงที่จับคู่ได้กับ X ในฐานความรู้ (โปรแกรม) และจะให้คำตอบเป็น 'arthur' ถ้าเราต้องการหาคำตอบอื่น เราสามารถถามต่อได้ว่า มีคำตอบอื่นอีกรึไม่ โดยใช้เครื่องหมายอัพภาค ';' ดังแสดงในตัวอย่างด้านบน ซึ่งโปรแกรมจะให้คำตอบที่สองคือ 'victoria' และถ้าถามต่อด้วย ';' ซึ่งจะไม่พบค่าคงที่ตัวอื่นอีกที่เมื่อแทนใน X แล้วทำให้ 'parent(christopher,X)' เป็นจริง ดังนั้นโปรแกรมจะตอบ 'No'

พจน์และการแทนค่า

พจน์ (term) คือสิ่งต่อไปนี้

- ค่าคงที่และตัวแปรเป็นพจน์
- พจน์ประกอบ (compound term – structure) เป็นพจน์ที่ประกอบด้วย
 - ฟังก์เตอร์ (functor) (หรือเรียกว่าฟังก์ชัน)
 - อาร์กิวเมนต์ที่เป็นพจน์

การแทนค่า (substitution) คือเซตของคู่ล้ำดับ $\{X = t\}$ โดยที่ X เป็นตัวแปรและ t เป็นพจน์ เราเรียก A ว่าเป็น **ตัวอย่าง (instance)** ของ B ถ้ามีการแทนค่า θ บางตัวที่ทำให้ $A = B\theta$ ตัวอย่างเช่น

$A = \text{parent}(\text{penelope}, \text{arthur})$

$B = \text{parent}(X, Y)$

$\theta = \{X = \text{penelope}, Y = \text{arthur}\}$

ซึ่งได้ว่า $A = B\theta$

ตัวเชื่อม 'และ'

ตัวเชื่อมและ (conjunction) เป็นแทนด้วย ',' เป็นเครื่องหมายแสดง 'และ' ทางตรรกศาสตร์ ตัวอย่างเช่น

?- $\text{parent}(\text{penelope}, X), \text{parent}(X, Y).$

คือข้อคำถามให้หา X และ Y ซึ่ง X มีพ่อแม่เป็น penelope และ X เป็นพ่อแม่ของ Y ข้อคำถามนี้เป็น 1 ข้อคำถามแต่มี **เป้าหมาย (goal)** 2 ตัวคือ ' $\text{parent}(\text{penelope}, X)$ ' และ ' $\text{parent}(X, Y)$ '

กฎ

กฎ (rule) หรือที่เราเรียกว่า **อนุประโยค (clause)** ในตรรกะเพรดิคเตตัน ในโปรแกรมจะเขียนอยู่ในรูปด้านล่างนี้

$H :- B_1, B_2, \dots, B_n.$

โดยที่ ‘:-’ แทนคำว่า ‘ถ้า’ ส่วนเครื่องหมาย ‘;’ ที่อยู่ระหว่าง **สัญพจน์** B_1 ก็คือตัวเชื่อม ‘และ’ ดังที่กล่าวแล้วข้างต้น ดังนั้นก็จะนิอ่านว่า ถ้า B_1 และ B_2 และจนกระทั่งถึง B_n ทุกด้วยเป็นจริงแล้ว สัญพจน์ H ก็จะเป็นจริงด้วย

ก็หรืออนุประโยคในภาษาโปรแกรมจะมีสัญพจน์อยู่ตัวเดียวที่ไม่มีนิเสธคือ H ก็จะเป็นจริงด้วยในรูปอนุประโยคในตรรกะเพรีดิเกตได้เป็น

$$H \vee \sim B_1 \vee \sim B_2 \vee \dots \vee \sim B_n.$$

อนุประโยค
ของขอร์น

เราเรียกอนุประโยคที่มีสัญพจน์ตัวเดียวที่ไม่มีนิเสธว่า **อนุประโยคของขอร์น (Horn clause)** ดังนั้นก็จะไม่เป็นอนุประโยคของขอร์น ซึ่งต้องมีสัญพจน์หนึ่งตัวที่ไม่มีนิเสธ ส่วนสัญพจน์อื่นๆ ที่เหลือทั้งหมด (ดังแต่ 0 ตัวเป็นต้นไป – กรณีที่เป็น 0 ตัวคือข้อเท็จจริง) ต้องมีนิเสธ และเราเรียก H ว่าเป็น **ส่วนหัว (head)** ของก็ และเรียก ' B_1, B_2, \dots, B_n ' รวมกันว่าเป็น **ลำตัว (body)** ของก็ ตัวอย่างของก็ เช่น

grandparent (X, Y) :- parent (X, Z), parent (Z, Y).

มีส่วนหัวคือ ‘grandparent(X,Y)’ และส่วนลำตัวคือ ‘parent(X,Z), parent(Z,Y)’ ก็จะนิอ่านว่า ถ้า ‘parent(X,Z), parent(Z,Y)’ เป็นจริงแล้ว ‘grandparent (X, Y)’ เป็นจริงด้วย

ความหมายของก็ ‘ $P :- Q, R$ ’

เราสามารถมองก็ที่เขียนในโปรแกรมได้สองลักษณะคือ

1. ในลักษณะของความหมายเชิงประกาศ (declarative meaning): แปลความหมายของก็ในลักษณะของความจริงทางตรรกศาสตร์ ตัวอย่างเช่น

- ‘ $P :- Q, R$ ’ คือ P จะเป็นจริงถ้า Q และ R เป็นจริง
- ‘grandparent (X, Y) :- parent (X, Z), parent (Z, Y).’ คือ ‘grandfather(X, Y)’ เป็นจริงถ้า ‘parent(X, Z)’ และ ‘parent(Z, Y)’ เป็นจริง หรือสำหรับ X, Y และ Z ทุกด้วย X เป็นปู่ย่าของ Y ถ้า X เป็นพ่อแม่ของ Z และ Z เป็นพ่อแม่ของ Y

2. ในลักษณะของความหมายเชิงกระบวนการคำสั่ง (procedural meaning) มองในลักษณะเชิงโปรแกรมที่สามารถทำงานได้ ตัวอย่างเช่น

- ‘ $P :- Q, R$ ’ คือ ถ้าจะแก้ปัญหา P จะต้องแก้ปัญหาย่อย Q ตามด้วยปัญหาย่อย R
- ‘grandparent (X, Y) :- parent (X, Z), parent (Z, Y).’ คือ ในการตอบข้อคำถามว่า X เป็นปู่ย่าของ Y หรือไม่ ให้ตอบข้อคำถามว่า X เป็นพ่อแม่ของ Z หรือไม่ และ Z เป็นพ่อแม่ของ Y หรือไม่

การจับคู่และการยื้อหน่อย

การเท่ากันและการจับคู่ (equality and matching)

พิจารณาข้อคำถามในโปรดีอก

?- X = Y.

เครื่องหมาย '=' แสดง **การจับคู่ (matching)** ซึ่งหมายถึงการจับคู่ X กับ Y โดยพยายามทำให้ X **เท่ากับ (equal)** Y ถ้าทำสำเร็จแสดงว่า X จับคู่กับ Y ได้ ถ้าไม่สำเร็จแสดงว่า X จับคู่กับ Y ไม่ได้ ข้อคำถามด้านบนนี้ไม่ได้ตรวจสอบการเท่ากันของ X กับ Y แต่เป็นการพยายามจับคู่ X กับ Y เพื่อพยายามทำให้ X เท่ากับ Y การจับคู่ในโปรดีอกกีคือ **การทำให้เท่ากัน (unify)** ในตรรกะเพรตติเคต

ກາສໍາຫຽວຈັບຄ' X ກັບ Y

ກວດສອບການຈັບຄົ່ງ X ກັບ Y ມີດັ່ງນີ້

- ถ้า X เป็นตัวแปรที่ยังไม่ได้แทนค่า และ Y ถูกแทนค่าด้วยพจน์ใดๆ แล้ว X กับ Y เท่ากันและ Y จะถูกแทนค่าด้วยพจน์นั้นด้วย
 - ตัวเลขและอะตอมเท่ากับตัวมันเองเท่านั้น
 - พจน์ประกอบ 2 ตัวใดๆ จะเท่ากันถ้าทั้งคู่มีพังก์เตอร์เดียวกัน มีจำนวนอาร์กิวเมนต์

ຕົວວິທີ ພະເຊົາ

2- triangle(point(1, 1) A point(2, 3)) ≡ triangle(X point(4, X) point(2, 7))

จังเขียนได้ และท่ากันด้วยการแทนค่า $\{X \equiv \text{point}(1, 1), A \equiv \text{point}(4, Y), Z \equiv 3\}$

ຂອບໃຈ

โปรดล็อกมีกระบวนการที่เรียกว่า **การย้อนรอย (backtracking)** เพื่อช่วยในการค้นหาคำตอบเบริญน์เมื่อการค้นหาแนวลึกก่อนในการค้นหาของบริภัณฑ์ ผู้ใช้งานกดด้านล่างนี้

$A := B_1 \cup B_2 \cup B_3$

กฎข้อหนึ่งกล่าวว่า A จะเป็นจริง ถ้า B_1 และ B_2 จนกระทั่งถึง B_n เป็นจริง ัญพจน์ A, B_1 , B_2, \dots, B_n อาจประกอบด้วยตัวแปร ถ้าเราต้องการพิสูจน์ว่า A เป็นจริงหรือไม่ เราจึงจะเริ่มพิสูจน์ B_1 ในการพิสูจน์ B_1 อาจมีการแทนค่าของตัวแปรใน B_1 ที่ทำให้ B_1 เป็นจริง และการแทนค่าเหล่านี้จะมีผลต่อไปยัง B_2, \dots, B_n และสมมติว่าในขณะที่เราพิสูจน์ B_i ด้วยการแทนค่าก่อนหน้านี้และพบว่า B_i ไม่จริง สิ่งที่เกิดขึ้นก็คือแทนที่จะตอบว่า A ไม่จริง โปรดล็อกจะพยายามทำการพิสูจน์ต่อโดยตัดการแทนค่าของตัวแปรที่ผ่านมาสำหรับตัวแปรที่ปรากฏใน B_i และย้อนกลับไปยัง B_{i-1} เพื่อหาเส้นทางพิสูจน์ใหม่ ซึ่งหมายถึงการแทนค่าใหม่ การย้อน

รอยสามารถกลับไปจนถึง B_1 และถ้ามีการแทนค่าบางตัวที่ทำให้ B_1 ถึง B_i เป็นจริงก็จะพิสูจน์ต่อไปได้ ตัวอย่างเช่นถ้าเราต้องการหาคนที่เป็นลูกของ christopher และเป็นพ่อแม่ของ colin เราใช้ข้อคำถามดังนี้

?- parent(christopher,X), parent(X,colin).

โปรแกรมจะพิสูจน์เป้าหมายตัวแรกในข้อคำถามด้านบน โดยพยายามหาการแทนค่าของ X ที่ทำให้ 'parent(christopher,X)' เป็นจริง และพบว่าข้อเท็จจริงตัวที่ 1 ในตารางที่ 4-1 'parent(christopher,arthur).' ด้วยการแทนค่า $\{X = arthur\}$ ทำให้เป้าหมายแรกเป็นจริง ได้จึงส่งการแทนค่านี้ไปยังเป้าหมายถัดไป ในกรณีพิสูจน์เป้าหมายตัวที่สอง 'parent(X,colin)' นีองจาก X ถูกแทนค่าด้วย arthur ดังนั้นโปรแกรมจะค้นหาว่ามี 'parent(arthur,colin)' ในฐานความรู้หรือไม่ ซึ่งพบว่าไม่มี ทำให้เป้าหมายตัวนี้ไม่เป็นจริง ณ จุดนี้จะเกิดการย้อนรอยกลับไปยังเป้าหมายตัวแรกพร้อมกับตัดการแทนค่า $\{X = arthur\}$ ออก เพื่อหาการแทนค่าอื่นที่ทำให้เป้าหมายตัวแรกเป็นจริงและพบว่า $\{X = victoria\}$ ทำให้เป้าหมายตัวแรกเป็นจริง ได้ พร้อมกับส่งการแทนค่าใหม่นี้เพื่อไปพิสูจน์ต่อสำหรับเป้าหมายตัวที่สอง ซึ่งครั้งนี้จะเป็น 'parent(victoria,colin)' และพบว่าเป็นจริงด้วยข้อเท็จจริงข้อที่ 9 ในตารางที่ 4-1 จึงได้การแทนค่า $\{X = victoria\}$ ทำให้ข้อคำถามเป็นจริง

กระบวนการทำงานของโปรแกรมเพื่อตอบข้อคำถามแสดงในตารางที่ 4-2
ทำงานของ
โปรแกรม

ตารางที่ 4-2 กระบวนการทำงานของโปรแกรม

<p>Algorithm: Execute([G1,G2,⋯,Gn])</p> <ol style="list-style-type: none"> 1. If the goal list is empty then terminate with success. 2. If the goal list is not empty then continue. 3. Scan the clauses in the program from top to bottom until the first clause C (H :- B1,⋯,Bm) is found such that the head of C matches G1. If no such clause then terminate with failure. Find substitution θ such that Hθ = G1. Replace G1 in the goal list with B1θ,B2θ,⋯,Bmθ, obtaining the new goal list B1θ,⋯,Bmθ,G2θ,⋯,Gnθ 4. Execute (recursively with this procedure) this new goal list. If the execution of the new goal list terminates with success then the execution of the original list also terminates with success. If not, then abandon this new goal and go back to (3). Continue scanning the next clause.
--

ຕ້ວຍໆຢ່າງປັບປຸງຫາລົງກິນກລ້າຍ

ລົງດ້ວຍໜຶ່ງອູ້ທີ່ປະຕູໃນຫ້ອງ ກລາງທ້ອງມືກລ້າຍແຂວນອູ້ທີ່ເພດານ ລົງທິວມາກແລະອຍາກໄປໜີບກລ້າຍແຕ່ມັນສູງໄມ້ພວກທີ່ຈະເຂົ້າມືຖື ທີ່ໜ້າຕ່າງໃນຫ້ອງມືກລ່ອງໆ ມີເຫັນວ່າ ລົງສາມາດເດີນບັນພື້ນປິ່ນກລ່ອງ ພລັກກລ່ອງໄປມາຮອບຫ້ອງ ແລະຫຍົບກລ້າຍຄ້າກ່ລ່ອງອູ້ໄຕກລ້າຍ ຄໍາຄາມຄືອັນຫຍົບກລ້າຍໄດ້ຫີ່ອໄມ່

ໃນການເຂົ້ານໂປຣແກຣມໂປຣລົກນ້ຳເຮົາຕ້ອງເລືອກເພຣດີເຄີດທີ່ຈະແທນຄວາມສັນພັນຮີໃນປັບປຸງຫາທີ່ເຮົາສັນໃຈ ໃນທີ່ນີ້ຈະໃຊ້ເພຣດີເຄີດ ‘state’ ເພື່ອແສດງສຕານະໜຶ່ງໆ ທີ່ແສດງຂ້ອມໜຸລ 4 ດ້ວຍຄື່ອ (1) ຕໍາແໜ່ງທີ່ລົງອູ້ (2) ລົງຍືນບັນພື້ນຫີ່ອກລ່ອງ (3) ຕໍາແໜ່ງທີ່ກ່ລ່ອງດັ່ງອູ້ແລະ (4) ລົງມືກລ້າຍຫີ່ອໄມ່ ເຮົາແທນສຕານະເຮີມຕັ້ນໄດ້ດັ່ງນີ້

state(atdoor, onfloor, atwindow, hasnot)

ແລະສຕານະສຸດທ້າຍຄື່ອ

state(_____, has)

ໂດຍທີ່ ‘_’ ຄື່ອຕ້ວແປປ່າຍໃສນ (*don't care variable*) ມາຍຄື່ອງຕ້ວແປປ່າຍທີ່ເຮົາໄມ້ສັນໃຈຄ່າທີ່ໄດ້ໜັງພັນຄໍາຕອນ ດັ່ງນັ້ນ state(_____, has) ເທົ່າກັນ state(X, Y, Z, has) ແຕ່ກຣົນີແຮກເຮົາໄມ້ສັນໃຈການແທນຄ່າທີ່ໄດ້

ເຮົາຈະໃຊ້ແນວຄີດຂອງການຄັ້ນຫາໃນບຽນສຕານະເພື່ອຫາຄໍາຕອນຂອງປັບປຸງຫານີ້ ໂດຍພິຈານາວ່າເຮົາມີສຕານະເຮີມຕັ້ນ ມີສຕານະສຸດທ້າຍ ແລະຄ້າເຮົາມີຕ້ວກະທຳການທີ່ໃຊ້ປັບປຸງຫາ ເຮົາກີຈະຄັ້ນຫາຄໍາຕອນໄດ້ເພວະວ່າໂປຣລົກຈະໃຊ້ກະບວນການຍັ້ນຮອຍເພື່ອຄັ້ນຫາເສັ້ນທາງຕ່າງໆ ໃນລັກປະນະຂອງການຄັ້ນຫາແນວລືກ ດັ່ງນັ້ນ ພະ ຈຸດນີ້ເຮົາໃຫ້ເພຣດີເຄີດ ‘move’ ແສດງການປັບປຸງຫາຈາກສຕານະໜຶ່ງໄປອັກສຕານະໜຶ່ງ ເພຣດີເຄີດນີ້ມີຮູບແບບດັ່ງນີ້

move(State1, MoveOp, State2)

ໂດຍທີ່ MoveOp ເປັນຕ້ວກະທຳການແລະອາຈເປັນ (1) ຫຍົບກລ້າຍ – grasp (2) ປິ່ນກລ່ອງ – climb (3) ພລັກກລ່ອງ – push (4) ເດີນໄປນາ – walk ຕ້ວຍໆຢ່າງເຊັ່ນ

move(state(middle, onbox, middle, hasnot), grasp, state(middle, onbox, middle, has))

ແສດງການປັບປຸງຫາໂດຍຕ້ວກະທຳການຕ້ວທີ່ໜຶ່ງ grasp ເພື່ອປັບປຸງຫາຈາກສຕານະທີ່ ‘ລົງອູ້ກລາງທ້ອງ ລົງອູ້ບຸນກລ່ອງ ກລ່ອງອູ້ກລາງທ້ອງ ລົງໄມ້ມືກລ້າຍ’ ໄປຍັງສຕານະທີ່ ‘ລົງອູ້ກລາງທ້ອງ ລົງອູ້ບຸນກລ່ອງ ກລ່ອງອູ້ກລາງທ້ອງ ລົງມືກລ້າຍ’

เมื่อเราเขียนเพรดิคเตต 'move' สำหรับตัวกระทำการที่เหลือเรียบร้อยแล้ว (ดูใน [ตารางที่ 4-3](#)) เพรดิคเตตตัวสุดท้ายที่เราต้องนิยามก็คือ 'canget' เพื่อใช้สำหรับพิสูจน์ว่าที่สถานะหนึ่งๆ ลิงจะหยิบกล้ายได้หรือไม่

```
canget(state(_,_,_,has)).  
canget(S1) :- move(S1,M,S2), canget(S2).
```

เพรดิคเตตนี้ประกอบด้วยกฎสองข้อ ข้อแรกเป็นกรณีของสถานะสุดท้ายคือกรณีที่ลิงมีกล้ายแล้วแสดงว่าลิงหยิบกล้ายได้ ส่วนกฎข้อที่สองเป็นกรณีของสถานะอื่นๆ (S1) ที่ยังไม่มีกล้าย ซึ่งนิยามว่าที่สถานะ S1 ลิงหยิบกล้ายได้ถ้ามีตัวกระทำการ M บางตัวที่เปลี่ยนสถานะจาก S1 ไปเป็น S2 และพบว่าที่ S2 ลิงหยิบกล้ายได้ กฎข้อที่สองนี้มีนิยามแบบเรียกซ้ำ (ดูรายละเอียดของโปรแกรมเรียกซ้ำใน [หัวข้อที่ 4.2](#))

เรานำเพรดิคเตตทั้งหมดเขียนเป็นโปรแกรมได้ใน [ตารางที่ 4-3](#) ด้านล่างนี้

ตารางที่ 4-3 โปรแกรมลิงกินกล้าย

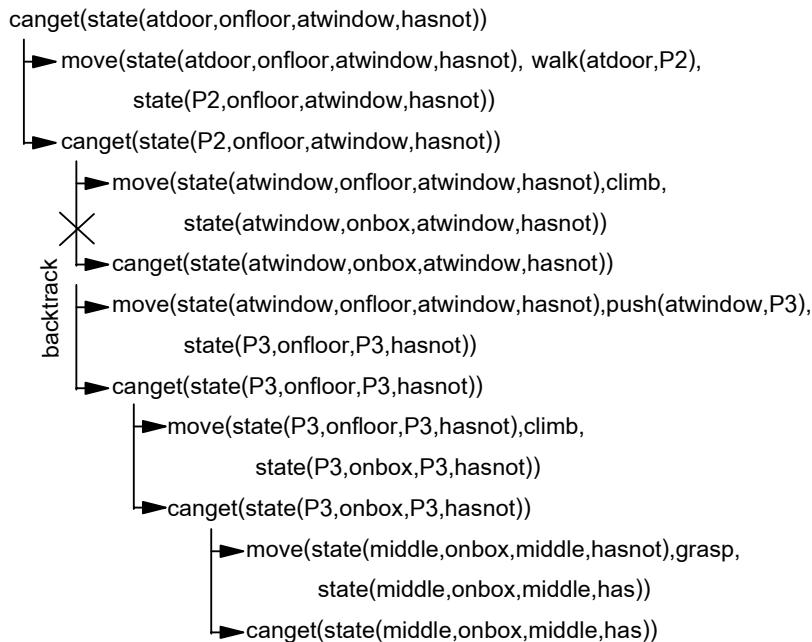
```
01: canget(state(_,_,_,has)).  
02: canget(S1) :- move(S1,M,S2), canget(S2).  
03: move(state(middle,onbox,middle,hasnot), grasp,  
04:       state(middle,onbox,middle,has)).  
05: move(state(P,onfloor,P,H), climb,  
06:       state(P,onbox,P,H)).  
07: move(state(P1,onfloor,P1,H), push(P1,P2),  
08:       state(P2,onfloor,P2,H)).  
09: move(state(P1,onfloor,B,H), walk(P1,P2),  
10:       state(P2,onfloor,B,H)).
```

เมื่อบรรจุโปรแกรมนี้เข้าไปในตัวแปลภาษาซีโปรเล็อก เราสามารถตั้งคำถามที่เราต้องการหาคำตอบได้ดังนี้

```
?- canget(state(atdoor,onfloor,atwindow,hasnot)).
```

Yes

ซึ่งแสดงให้เห็นว่าจากสถานะเริ่มต้นลิงจะหยิบกล้ายได้ ด้านล่างนี้แสดง **การตามรอย (trace)** การทำงานของโปรเล็อก



รูปที่ 4-2 ตามรอยการทำงานของໂປຣລູກສໍາຫຼັບປັບປຸງຫາລິງກິນກລ້າຍ

4.2 การໂປຣແກຣມແບບເຮືອກຫຼັກ

การໂປຣແກຣມແບບເຮືອກຫຼັກ (*recursive programming*) เป็นເຄື່ອງນີ້ອໍາທີ່ສໍາຄັນຂອງການໃຊ້ໂປຣລູກເນື້ອງຈາກໃນພາກພາໂປຣລູກໄມ້ມີຄໍາສັ່ງວັນຫຼຳ (loop) ແລ້ວອືນໃນພາກພາເອົ້າໆ ພາຍພາກພາ ດັ່ງນັ້ນການໂປຣແກຣມເຮືອກຫຼັກຈຶ່ງເປັນເຄື່ອງນີ້ທີ່ມີໜັກອືກຕ້ວາໜຶ່ງໃນການເຂົ້າໃນໂປຣແກຣມໂປຣລູກ ເຊັ່ນຄ້າເຮົາທີ່ຕ້ອງການຫາວ່າໄດ້ເປັນບໍລິຫານທຸກໆຂອງໄດ້ ເຮົາຈະເຂົ້າໃນໂປຣແກຣມໂປຣລູກທີ່ໄມ້ເປັນໂປຣແກຣມແບບເຮືອກຫຼັກໄດ້ດັ່ງນີ້

```

predesessor(X, Y) :- parent(X, Y).
predesessor(X, Y) :- parent(X, Z), parent(Z, Y).
predesessor(X, Y) :- parent(X, Z), parent(Z, W), parent(W, Y).
...
```

ຈະເຫັນໄດ້ວ່າໂປຣແກຣມດ້ານນີ້ໄມ້ມີປະສິທິພາພເພຣະຕ້ອງເຂົ້າໃນກົງຈໍານວນຫຼາຍ ແລະກີ່ໄມ້ສາມາດເຂົ້າໃນໄດ້ມາກພອທີ່ຈະຄອບຄຸມທຸກການ ແຕ່ຄ້າໃຫ້ໂປຣແກຣມແບບເຮືອກຫຼັກຈະນີ້ຍາມຄວາມສັນພັນນີ້ໄດ້ຄອບຄຸມທັງໝົດດ້ານລ່າງນີ້

`predesessor(X,Y) :- parent(X,Y).`

`predesessor(X,Y) :- parent(X,Z), predesessor(Z,Y).`

เมื่อพิจารณาโปรแกรม ‘`predecessor`’ ด้านบน เราจะเห็นลักษณะการเขียนโปรแกรมแบบเรียกซ้ำในโปรแกรมที่จะประกอบด้วย 2 ส่วนหลักคือ

- **อนุประโยคฐาน (base clause)** คือกฎที่ไม่มีการเรียกตัวมันเองและมีตั้งแต่ 1 ข้อขึ้นไป อนุประโยคฐานนี้ไว้สำหรับการหยุดการเรียกซ้ำ
- **อนุประโยคเรียกซ้ำ (recursive clause)** คือกฎที่มีการเรียกตัวของมันเอง โดยจะมีเพรดิเกตในส่วนลำตัวของกฎบางเพรดิเกตที่เหมือนกับเพรดิเกตที่ส่วนหัวโปรแกรมของเพรดิเกตหนึ่งๆ มีกฎเรียกซ้ำแบบนี้ตั้งแต่ 1 ข้อขึ้นไป ในอนุประโยคเรียกซ้ำนี้เพรดิเกตที่เรียกซ้ำต้องมีอาร์กิวเม้นต์ที่แตกต่างกับเพรดิเกตที่ส่วนหัวไม่ เช่น การเรียกซ้ำจะไม่รับ

เพื่อให้เข้าใจถึงการเขียนโปรแกรมเรียกซ้ำในโปรแกรม ในส่วนต่อไปนี้จะยกตัวอย่างการเขียนโปรแกรมเรียกซ้ำที่ใช้จัดการกับตัวเลขและรายการ (list) ตามลำดับ

4.2.1 โปรแกรมเรียกซ้ำกับตัวเลข

โปรแกรมจำนวนธรรมชาติ

โปรแกรมแรกที่เราจะเขียนด้วยโปรแกรมเรียกซ้ำคือ โปรแกรมสร้างจำนวนธรรมชาติ (natural number) เราแสดงจำนวนธรรมชาติด้วยค่าคงที่ '0' และฟังก์ชันสืบเนื่อง (successor function) 's' จำนวนธรรมชาติของเรารอเรียงตามลำดับได้ดังนี้

`0, s(0), s(s(0)), s(s(s(0))), s(s(s(s(0)))), ...`

ซึ่งหมายถึงตัวเลขจำนวนเต็ม 0, 1, 2, 3, 4, ... แม้ว่าในโปรแกรมจะมีตัวเลขจำนวนเต็มให้ใช้ได้ แต่ในหัวข้อนี้เราสนใจที่จะศึกษาการเขียนโปรแกรมเรียกซ้ำ จึงจะทดลองสร้างตัวเลขจำนวนธรรมชาติขึ้นมาใช้เอง

ในการเขียนโปรแกรมเรียกซ้ำโดยทั่วไป สิ่งที่เราต้องพิจารณาก็คือ อนุประโยคฐานเขียนได้อย่างไรและอนุประโยคเรียกซ้ำเขียนได้อย่างไร ซึ่งมีข้อสังเกตว่าอนุประโยคเรียกซ้ำมักจะทำหน้าที่เพิ่มหรือลดลำดับของข้อมูลในอาร์กิวเม้นต์ของเพรดิเกตที่เรียกซ้ำและอนุประโยคฐานมักเป็นตัวที่มีลำดับน้อยสุดหรือมากสุดของข้อมูล ลองดูตัวอย่างโปรแกรมนี้ที่ชื่อ ‘natural_number’ ในตารางที่ 4-4

ตารางที่ 4-4 โปรแกรมจำนวนธรรมชาติ

```
1: natural_number(0).  
2: natural_number(s(X)) :- natural_number(X).
```

จากโปรแกรมจะเห็นได้ว่าข้อมูลตัวมีลำดับน้อยสุดคือ '0' เป็นข้อมูลที่ถูกใช้สำหรับอนุประโยคฐานในกฎข้อที่ 1 และพบว่าในอนุประโยคเรียกช้าข้อมูลถูกลดลำดับลงหนึ่งหน่วยจาก $s(X)$ เป็น X เช่นถ้าเราตั้งข้อคำถามว่า '?- natural_number(s(s(s(0))))' (3 เป็นจำนวนธรรมชาติหรือไม่?) ข้อคำถามนี้บังคับได้กับส่วนหัวของกฎข้อ 2 โดยพยายามทำให้อาร์กิวเมนต์ที่ส่วนหัวของกฎคือ ' $s(X)$ ' เท่ากับ ' $s(s(s(0)))$ ' ได้การแทนค่า $\{X = s(s(s(0)))\}$ จากนั้นโปรแกรมจะพิสูจน์ต่อว่าส่วนลำดับเป็นจริงหรือไม่ นั่นคือ $natural_number(s(s(s(0)))$ จะเห็นได้ว่าเมื่อมีการเรียกช้าเกิดขึ้นแต่ละครั้ง ลำดับของข้อมูลจะน้อยลงหนึ่งหน่วย ดังนั้นเมื่อเรียกช้าหลายรอบจะถึงจุดที่อาร์กิวเมนต์เป็น '0' ซึ่งจะหยุดได้ด้วยอนุประโยคฐาน

โปรแกรมบวกจำนวนธรรมชาติ

โปรแกรมต่อไปคือโปรแกรมบวกจำนวนธรรมชาติ ให้โปรแกรมบวกเลขใช้เพรดิเคตชื่อ 'plus' มีรูปแบบคือ ' $\text{plus}(X,Y,Z)$ ' มีความหมายว่า Z เป็นผลบวกของ X กับ Y เช่นเมื่อตั้งข้อคำถาม '?-plus(s(s(0)),s(s(s(0))),Z)' แล้วต้องให้ $Z = s(s(s(s(s(0)))))$ เป็นเหตุน โปรแกรมที่ได้เป็นดังตารางที่ 4-5 นี้

ตารางที่ 4-5 โปรแกรมนวชาเลข

```
1: plus(0,X,X) :- natural_number(X).  
2: plus(s(X),Y,s(Z)) :- plus(X,Y,Z).
```

สมมติว่า อาร์กิวเมนต์สองตัวแรกเป็นอนิพุตและตัวที่สามเป็นเอาต์พุต อย่างไรก็ได้ โดยทั่วไปแล้วโปรแกรมป์ร็อกส์่วนมากดังเช่นโปรแกรมนี้ อาร์กิวเมนต์แต่ละตัวเป็นได้ทั้ง อนิพุตและเอาต์พุต ในโปรแกรมนี้จะเห็นว่า กฎข้อแรกซึ่งเป็นอนุประโยคฐานมีอาร์กิวเมนต์ ตัวแรกเป็นข้อมูลที่มีลำดับต่ำสุดคือ '0' และเมื่อสังเกตอาร์กิวเมนต์ตัวเดียว กัน (ตัวแรก) ใน กฎข้อที่สองจะเห็นได้ว่า อาร์กิวเมนต์ตัวแรกที่ส่วนหัวมีลำดับมากกว่าอาร์กิวเมนต์ตัวแรกที่ ส่วนลำดับ ซึ่งก็คือมีการลดลำดับลงที่ละหนึ่ง ซึ่งโปรแกรมเรียกซ้ำจะมีลักษณะเช่นนี้ คือ อนุประโยคเรียกซ้ำจะถูกเรียกวันซ้ำ จนกระทั่งถึงจุดหนึ่งที่ อาร์กิวเมนต์ตรงกับอนุประโยค ฐานก็จะหยุด

ກວ່ານີ້ແກ່ທີ່ມາຍຄວາມວ່າ '0' ບວກກັບຕົວເລີນໄດ້ ຈະໄດ້ຕົວເລີນນັ້ນ ສ່ວນກວ່ານີ້ທີ່ສອງ
ໝາຍຄວາມວ່າ ຄ້າ X ບວກ Y ໄດ້ Z ($\text{plus}(X, Y, Z)$) ແລ້ວ $X+1$ ($\text{s}(X)$ ກີ່ຄື່ອງ $X+1$) ບວກ Y ຕ້ອງໄດ້
 $Z+1$

การเขียนโปรแกรมเรียกช้า้นี้สามารถเขียนได้โดยเรากำหนดอนุประโยคฐานก่อน และต้องคำนึงว่าเราจะลดลำดับของอาร์กิวเม้นต์ตัวใด สมมติว่าในกรณีนี้จะลดลำดับที่อาร์กิวเม้นต์ตัวแรก (หรือเราจะลดที่อาร์กิวเม้นต์ตัวที่สองก็ได้) ดังนั้นเราจะได้ในขั้นตอนแรกว่าก្នុងข้อแรกควรเป็น ‘plus(0,X,?)’ และเราก็มาหาค่าของ ? ว่าควรเป็นอะไร ซึ่งจากคณิตศาสตร์ร่ายๆ เราจะได้ว่า ? ก็คือ X (เพราะ 0 บวกตัวเลขใดก็ได้ตัวเลขนั้น) ดังนั้นก្នុងข้อแรกจึงเป็น ‘plus(0,X,X) :- natural_number(X).’ ส่วนลำดับของก្នុងนี้เพิ่มเข้าไปเพื่อตรวจสอบว่า X ต้องเป็นจำนวนธรรมชาติเท่านั้น กล่าวคือเราจะไม่ใช้โปรแกรมนี้สำหรับบวกเลขที่อยู่ในรูปจำนวนเต็มเช่น ‘?- plus(5,2,Z)’ เป็นต้น

จากนั้นเรามาเขียนอนุประโยคเรียกช้า ดังที่กล่าวแล้วว่าเราต้องการลดลำดับของอาร์กิวเม้นต์ตัวแรก ดังนั้นส่วนหัวของก្នុងจึงเป็น ‘plus(s(X),Y,?)’ ซึ่งเราต้องหาเอาต์พุต ? ต่อไปว่าควรเป็นอะไร ณ จุดนี้เรารู้ว่ากำลังเขียนโปรแกรมเรียกช้า ดังนั้นส่วนลำดับของก្នុงนี้ จึงต้องมีเพรดิคเตต plus โดยที่มีการลดลำดับของอาร์กิวเม้นต์ตัวแรก เราจึงได้ก្នុងเป็น ‘plus(s(X),Y,?) :- plus(X,Y,??).’ ให้เรากำหนดค่าของ ?? เป็นตัวแปรไดๆ ได้เลย เช่นให้เป็น Z จะได้ ‘plus(s(X),Y,?) :- plus(X,Y,Z).’ และสมมติว่าถ้า Z ที่เป็นค่าเอาต์พุตเป็นจริงแล้ว ? ควรเป็นเท่าไร ซึ่งเราจะได้ว่าถ้า X บวก Y ได้ Z (plus(X,Y,Z)) และแนอนว่าจาก การคำนวณทางคณิตศาสตร์ s(X) บวก Y ก็ต้องได้ s(Z) ทำให้เราได้ก្នុងข้อที่สองดังในตารางที่ 4-5

โปรแกรมคุณจำนวนธรรมชาติ

โปรแกรมสุดท้ายในส่วนของการเขียนโปรแกรมเรียกช้ากับตัวเลขคือโปรแกรมคุณจำนวนธรรมชาติ ‘times(X,Y,Z)’ มีความหมายว่า X คูณ Y เท่ากับ Z โปรแกรมเป็นดังตารางที่ 4-6 นี้

ตารางที่ 4-6 โปรแกรมคุณจำนวนธรรมชาติ

1: times(0,X,0). 2: times(s(X),Y,Z) :- times(X,Y,W), plus(W,Y,Z).
--

อนุประโยคฐานในก្នុងข้อแรกของโปรแกรมนี้มีอาร์กิวเม้นต์ตัวแรกเป็น ‘0’ และอนุประโยคเรียกช้ามีอาร์กิวเม้นต์ตัวแรกที่เมื่อเรียกช้าแล้วลำดับลดลงทีละหนึ่ง ซึ่งเมื่อการเรียกช้าดำเนินไปจนกระทั่งอาร์กิวเม้นต์แรกเป็น ‘0’ ก็จะหยุดได้ที่ก្នុងข้อนี้ ความหมายของโปรแกรมนี้คือ 0 คูณตัวเลขใดก็ได้ 0 ตามก្នុងข้อแรก ส่วนก្នុងข้อที่สองหมายความว่าถ้า X คูณ Y ได้ W และ W บวก Y ได้ Z และ s(X) คูณ Y จะได้ Z

วิธีการเขียนโปรแกรมนี้คล้ายกับโปรแกรมบวกเลข โดยเรากำหนดว่าจะลดลำดับที่อาร์กิวเม้นต์ตัวแรก ดังนั้นเราได้ ‘times(0,X,?)’ และใช้ความรู้ทางคณิตศาสตร์ทำให้รู้ว่า

? เท่ากับ 0 (เพราะ 0 คูณอะไรก็ได้ 0) ส่วนอนุประโยคเรียกช้า้นนี้ เราเริ่มเขียนจากส่วนหัว ได้ 'times(s(X),Y,?)' และเพิ่มเพรดิเคต 'times' เข้าที่ส่วนลำดับได้เป็น 'times(s(X),Y,?) :- times(X,Y,??)' กำหนดให้ ?? เขียนแทนด้วย W และคำนวณว่า ? เป็นเท่าไรของ W ณ จุดนี้เราได้ 'times(s(X),Y,?) :- times(X,Y,W)' หา ? ว่าเท่ากับเท่าไร ตรงนี้เราคำนวณ คณิตศาสตร์กันเล็กน้อย

ถ้า $X*Y = W$ แล้วจะได้ว่า $s(X)*Y = (X+1)*Y = X*Y+Y = W+Y$ ดังนั้นแสดงว่า ?
เท่ากับ $W+Y$ และเนื่องจากเรามีโปรแกรมบวกเลขแล้วเรามีคำสั่ง ‘plus’ มา
ต่อเข้าที่ส่วนลำดับของกฎ แล้วเขียน ? แทนด้วย Z เราจะได้ ‘times($s(X)$, Y , Z) :-
times(X , Y , W), plus(W , Y , Z).’ เป็นโปรแกรมในตารางที่ 4-6

4.2.2 โปรแกรมเรียกซ้ำกันรายการ

รายการ (list) เป็นโครงสร้างข้อมูลสำคัญในภาษาโปรแกรม ลักษณะของรายการข้อมูลแสดงในตารางที่ 4-7

ตารางที่ 4-7 ตัวอย่างของรายการข้อมูลในໂປຣລູກ

รายการ (รูปแบบทั่วไป)	ส่วนหัว	ส่วนหาง	รายการ (รูปแบบในเชิงพังก์ชัน)
[a]	a	[]	.(a,[])
[a,b,c]	a	[b,c]	.(a,.(b,.(c,[])))
[]	—	—	[]
[[a,b],c]	[a,b]	[c]	.(.(a,.(b,[])),.(c,[]))
[X Y]	X	Y	.(X,Y)
[X,Y Z]	X	[Y Z]	.(X,.(Y,Z))

รายการประกอบด้วยสมาชิกตั้งแต่ศูนย์ตัวขึ้นไปเรียงอยู่ภายในวงลีบเพื่อแสดงสมาชิกในรายการเรียงลำดับตั้งแต่ตัวแรก (ถ้ามี) เป็นต้นไป รายการในโปรดีล็อกแบ่งเป็นสองส่วนคือ ส่วนหัว (*head*) และส่วนหาง (*tail*) ส่วนหัวคือสมาชิกตัวแรกในรายการ ส่วนหางคือรายการที่เหลือเมื่อลบสมาชิกตัวแรกออกไปแล้ว ตั้งที่ได้กล่าวแล้วข้างต้นว่าพจน์ในโปรดีล็อกคือตัวแปร *ค่าคงที่* และพจน์ประกอบ รายการข้อมูลก็เป็นพจน์ประกอบแบบหนึ่งในโปรดีล็อกที่มีเครื่องหมายพังก์ชันเป็น ‘.’ และมีอาร์กิวเมนต์สองตัว ตัวแรกเป็นส่วนหัว ตัวที่สองเป็นส่วนหาง แต่รูปแบบในเชิงพังก์ชันมักไม่เป็นที่นิยม เพราะอ่านเข้าใจยาก โดยมากเราจะใช้ในรูปแบบทั่วไป

รายการว่าง (empty list) แสดงด้วยสัญลักษณ์ ‘[]’ เพื่อแทนรายการที่ไม่มีสมาชิก รายการสามารถประกอบด้วยสมาชิกที่เป็นค่าคงที่ ตัวแปร หรือพจน์ประกอบก็ได้ ตัวอย่างที่ 4 ใน [ตารางที่ 4-7](#) แสดงรายการที่มีสมาชิกตัวแรกเป็นรายการ ‘[a,b]’ ตัวอย่างของรายการที่มีสมาชิกเป็นตัวแปรก็ เช่น ‘[X,Y]’ ซึ่งหมายถึงรายการที่มีสมาชิก 2 ตัว ตัวแรกคือ X ตัวที่สองคือ Y ตัวอย่างตัวที่ 5 ในตารางแสดงรายการ ‘[X|Y]’ (ซึ่งไม่เท่ากับ ‘[X,Y]’) เป็นรูปแบบการเขียนรายการในໂປຣລູກທີ່ມີຄວາມໝາຍໝາວ່າສ່ວນຫັກີ່ X ສ່ວນຫາງີ່ Y (ສັງເກດວ່າ Y เป็นรายการ) X ສາມາຄົມຈັບຄູ່ໄດ້ກັບພົນໃດໆ ສ່ວນ Y ຈັບຄູ່ໄດ້ກັບรายการທີ່ມີສາມາຊີກດັ່ງແຕ່ຫຼຸ່ມຍົດຕັ້ງໄປ ດັ່ງນັ້ນ ‘[X|Y]’ ສາມາຄົມຈັບຄູ່ໄດ້ກັບรายการທີ່ມີສາມາຊີກດັ່ງແຕ່ຫຼຸ່ມຍົດຕັ້ງໄປ ເຊັ່ນ ‘[1]’, ‘[a,b]’, ‘[s(0),s(s(0)),s(s(s(0)))]’, ‘[1,a,2,b]’ ເປັນດັ່ນ

สามารถอழນໄດ້ວ່າ ‘[]’ เป็นຂ້ອມໜຸລືທີ່ມີລຳດັບຕໍ່ສຸດໃນໂຄງສ້າງຂ້ອມໜຸລືປະເທນີ່ รายการທີ່ມີສາມາຊີກຕົວເດືອນວ່າເປັນຂ້ອມໜຸລືທີ່ມີລຳດັບຄັດໄປ ເປັນຕົ້ນ ໃນຫຼັກຂັ້ນໜີ່ເຈົ້າຈົ່າກັບໂປຣແກຣມທີ່ຈັດກັບรายการດັ່ງຕ້ອງໄປນີ້

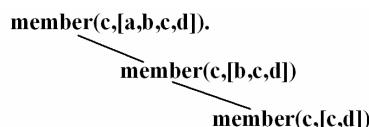
ໂປຣແກຣມກວະສາມາຊີກ

ໂປຣແກຣມແຮກໃນຫຼັກຂັ້ນໜີ່ເປັນໂປຣແກຣມຕຽບສອນກວະສາມາຊີກຂອງรายการ (membership of a list) ທຳນາ້ນທີ່ຕຽບສອນວ່າພົນທີ່ໆ ເປັນສາມາຊີກຂອງรายการທີ່ກຳຫັດໄທ້ຫຼື້ວ່າມີ ເຊິ່ນແກນດ້ວຍ ‘member(X, Ys)’ ມີຄວາມໝາຍໝາວ່າ X ເປັນສາມາຊີກຂອງรายการ Ys ມີໂປຣແກຣມດັ່ງ [ตารางที่ 4-8](#) ດ້ວຍລ່າງນີ້

ตารางที่ 4-8 ໂປຣແກຣມກວະສາມາຊີກ

1: member(X, [X _]).
2: member(X, [_ Y]) :- member(X, Y).

ກົງຂ້ອແຮກເປັນອຸປະໂພຄຈູາມີຄວາມໝາຍໝາວ່າ X ເປັນສາມາຊີກຂອງรายการທີ່ມີສ່ວນຫັກີ່ X ສ່ວນຫາງເປັນรายการໃດໆ (_) ກົງຂ້ອທີ່ສອງເປັນອຸປະໂພຄເຮີຍຂໍ້າໝາຍຄວາມວ່າ ຄ້າ X ເປັນສາມາຊີກຂອງรายการ Y ໄດ້ ແລ້ວ X ຈະເປັນສາມາຊີກຂອງรายการນັ້ນທີ່ມີພົນທີ່ຕົວເພີ່ມເຂົ້າທີ່ສ່ວນຫັກີ່ (|Y|) ດ້ວຍ ຕ້ອງຍ່າງການທຳການຂອງໂປຣແກຣມກັບຂ້ອຄໍາຄາມ ‘?- member(c,[a,b,c,d])’ ແສດໃນ [ຮູບທີ 4-3](#)



ຮູບທີ 4-3 ຕາມຮອຍການທຳການຂອງໂປຣແກຣມກວະສາມາຊີກ

จากข้อคำถาม ‘?- member(c,[a,b,c,d]).’ ซึ่งຈັບຄູ່ໄດ້ກັບສ່ວນຫວ່າອອກງົງຂໍ້ອໍ 2 ໃນຕາຮາງທີ 4-8 ໂດຍກາຣແທນຄ່າ { $X = c$, $_ = a$, $Y = [b,c,d]$ } ແລະພິສູງນີ້ສ່ວນລຳຕົວຂອງກົງ ‘member(c,[b,c,d])’ ໃນກາຣົພິສູງນີ້ສ່ວນລຳຕົວນີ້ໃນໂປຣແກຣມເຮົາກ້າວອົບຜົດມາຈະຈັບຄູ່ກັບສ່ວນຫວ່າອອກງົງຂໍ້ອໍ 2 ໂດຍກາຣແທນຄ່າ { $X = c$, $_ = b$, $Y = [c,d]$ } ພິສູງນີ້ສ່ວນລຳຕົວ ‘member(c,[c,d])’ ແລະໃນຄັ້ງນີ້ຈັບຄູ່ໄດ້ກັບກົງຂໍ້ອໍ 1 ທ່ານໃຫ້ຂໍ້ອຄາດເປັນຈິງ

ດັ່ງທີ່ເຮົາເຫັນໃນຕ້ອຍຢ່າງນີ້ຮ່າຍກາຣ ‘[a,b,c,d]’ ໃນອົບກົມເນົດທີ່ສອງຂອງຂໍ້ອຄາດຈະຄູກລົດລຳດັບລົງທີ່ລະໜຶ່ງ (ຮ່າຍກາຣສັ່ນລົງທີ່ລະໜຶ່ງ) ຈະກະທັ້ງສ່ວນຫວ່າຂອງອົບກົມເນົດຕົວນີ້ເທົກກັບ c ແລະຈະຫຼຸດໄດ້ວ່າຍອນຸປະໂຍຄູ້ານ

ໂປຣແກຣມຕ່ອງຮ່າຍກາຣ

ໂປຣແກຣມຕ່ອງຮ່າຍກາຣ ‘append(X,Y,Z)’ ທ່ານ້າທີ່ຕ່ອງຮ່າຍກາຣ X ເຂົ້າກັບຮ່າຍກາຣ Y ໄດ້ພັລັນພົ້ງເປັນຮ່າຍກາຣ Z ໂດຍກາຣຕ່ອງກື່ອການນຳສາມາຊີກຂອງ Y ຖຸກຕົວມາຕ່ອງເຂົ້າຂ້າງທ້າຍຂອງສາມາຊີກຂອງ X ຕ້ອຍຢ່າງເຊື່ອ ‘append([a,b],[1,2],Z)’ ຈະໄດ້ Z ເປັນ [a,b,1,2] ໂປຣແກຣມຕ່ອງຮ່າຍກາຣເປັນດັ່ງຕາຮາງທີ 4-9 ດ້ວຍລັດນີ້

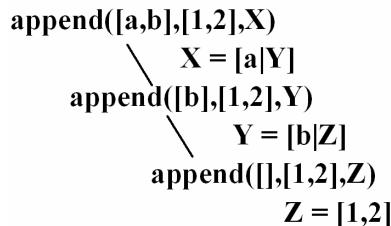
ຕາຮາງທີ 4-9 ໂປຣແກຣມຕ່ອງຮ່າຍກາຣ

```
1: append([],L,L).
2: append([A|L1],L2,[A|L3]) :- append(L1,L2,L3).
```

ວິທີເຂົ້ານໂປຣແກຣມເຮົາກ້າວສໍາຫັບຈັດກາຣໂຄຮ່າຮ້າງຂອ້ມູນແບນຮ່າຍກາຣມັກມີລັກຂະນະຄລ້າຍກັບໂປຣແກຣມເຮົາກ້າວສໍາຫັບຈັດກາຣກັບຕົວເລີນ ກລ່າວຄືເຮົາກວຣີມເຂົ້ານອນຸປະໂຍຄູ້ານກ່ອນໂດຍກໍານົດວ່າຕ້ອງການລັດລຳດັບຂອງຂອ້ມູນ ທີ່ອົບກົມເນົດຕົວໄດ້ ໃນທີ່ນີ້ຈະກຳທີ່ອົບກົມເນົດຕົວແຮກ ດັ່ງນັ້ນເຮົາຈະໄດ້ຂອ້ມູນທີ່ມີລຳດັບຕໍ່ສຸດຂອງຮ່າຍກາຣເປັນຮ່າຍກາຣວ່າ ທ່ານໃຫ້ເຮົາເຂົ້າກົງຂໍ້ອໍ 1 ໄດ້ເປັນ ‘append([],L,?)’ ໂດຍສົມມືວ່າອົບກົມເນົດສອງຕົວແຮກເປັນອິນພຸດ ຕົວທີ່ສາມເປັນເອົາຕົ້ນພຸດ (ດັ່ງທີ່ໄດ້ລ່າວແລ້ວວ່າອົບກົມເນົດໃນໂປຣແກຣມໂປຣລູກມັກເປັນໄດ້ທັງອິນພຸດແລະເອົາຕົ້ນພຸດ ແຕ່ເພື່ອຄວາມສະດວກໃນກາຣເຂົ້ານໂປຣແກຣມ ແລະ ຈຸດນີ້ເຮົາຈະສົມມືວ່າອົບກົມເນົດສອງຕົວແຮກເປັນອິນພຸດ ແລະ ຕົວທີ່ສາມເປັນເອົາຕົ້ນພຸດ) ແລະຈະໄດ້ຄ່າຂອງ ? ເປັນ L ເພະວ່າຮ່າຍກາຣວ່າງຕ່ອກກັບຮ່າຍກາຣໄດ້ ຈະໄດ້ຮ່າຍກາຣນັ້ນໆ ໃນ L ເພະວ່າຮ່າຍກາຣວ່າງຕ່ອກກັບຮ່າຍກາຣໄດ້ ຈະໄດ້ຮ່າຍກາຣນັ້ນໆ

ຈາກນັ້ນເຮົາຈະເຂົ້ານອນຸປະໂຍຄູ້ານພິຈາລະນາເພື່ອລັດລຳດັບຂອງອົບກົມເນົດຕົວທີ່ໜຶ່ງ ໄດ້ເປັນ ‘append([A|L1],L2,?) :- append(L1,L2,??)’ ໃນການອອນເດີຍກັບທີ່ເຮົາເຂົ້ານໂປຣແກຣມສໍາຫັບຕົວເລີນ ໃຫ້ເຮົາສົມມືໄດ້ເລີຍວ່າ ?? ເປັນຕົວແປຣໜຶ່ງ ເຊັ່ນໃຫ້ເປັນ L3 ແລ້ວຈະໄດ້ວ່າ ຄ້າ L1 ຕ່ອກັບ L2 ໄດ້ L3 ແລ້ວ [A|L1] (L1 ທີ່ມີສາມາຊີກ A ເພີ່ມທີ່ຂັ້ງໜ້າໜຶ່ງຕົວ) ຕ່ອກັບ L2 ຈະໄດ້ອະໄຮສິ່ງທ່ານໃຫ້ເຮົາວ່າ ? ກີ່ຄື່ອ L3 ທີ່ມີສາມາຊີກ A ເຕີມເຂົ້າທີ່ຂັ້ງໜ້າຮ້ອງ [A|L3] ນັ້ນເອງ ທ່ານໃຫ້ເຮົາໄດ້ກົງຂໍ້ອໍສອງເປັນ ‘append([A|L1],L2,[A|L3]) :- append(L1,L2,L3).’

เมื่อเราตั้งข้อคำถาม เช่น '?- append([a,b],[1,2],X).' การตามรอยแสดงในรูปที่ 4-4



รูปที่ 4-4 ตามรอยการทำงานของโปรแกรมต่อรายการ

4.3 นิเสธและตัวตัด

หัวข้อนี้อธิบาย **นิเสธ (negation)** และ **ตัวตัด (cut)** ดังต่อไปนี้

4.3.1 นิเสธ

ข้อเท็จจริงหรือกฎในโปรแกรมจะเขียนแสดงความสัมพันธ์ที่เป็นจริง แต่เมื่อเราต้องการเขียนความสัมพันธ์ที่ไม่เป็นจริงในโปรแกรม เราเขียนได้โดยใช้ **นิเสธ (negation)** แทนด้วย 'not' (ในตัวแปลภาษาโปรแกรมบางตัวใช้ '+') และตามด้วยความสัมพันธ์ที่คุณต้องการ เช่น 'not(G)' โดยที่ G เป็นความสัมพันธ์ใดๆ

'not(G)' จะเป็นจริงถ้า G ไม่เป็นจริงตามโปรแกรม และจะไม่เป็นจริงถ้า G เป็นจริงตามโปรแกรม ตัวอย่างเช่นถ้าเราต้องการเขียนโปรแกรมเพื่อธิบายว่า

X แต่งงานกับ Y ได้ถ้า X กับ Y ไม่ใช่พี่น้องกัน และ X ชอบ Y

เราจะเขียนโปรแกรมโดยใช้ not ได้ดังตารางที่ 4-10 ด้านล่างนี้

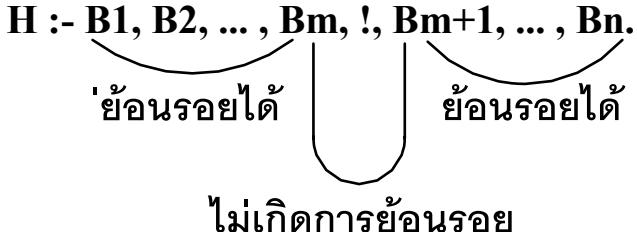
ตารางที่ 4-10 โปรแกรมแต่งงานกันได้

1: can_marry(X,Y) :- not(sibling(X,Y)), like(X,Y).
2: sibling(X,Y) :- parent(Z,X), parent(Z,Y), X \= Y.
3: like(arthur,margaret).

สมมติว่าเราใช้ฐานความรู้เดิมเกี่ยวกับต้นไม้ครอบครัวในตารางที่ 4-1 และสมมติว่า 'arthur' ชอบ 'margaret' เราเพิ่มข้อเท็จจริง 'like(arthur,margaret).' ไว้ในบรรทัดที่สามในตารางที่ 4-10 และเขียนกฎในบรรทัดที่ 1 แสดงความสัมพันธ์แต่งงานกันได้ 'can_marry(X,Y)' ส่วนความสัมพันธ์ 'sibling(X,Y)' ไว้ตรวจสอบว่า X กับ Y เป็นพี่น้องกัน ดังนั้นจะได้โปรแกรมของ 'can_marry' ดังในตาราง คำนิยามของ 'sibling(X,Y)' ดังในตารางหมายถึง X เป็นพี่น้องของ Y ถ้ามีพ่อแม่คนเดียวกันและ X ไม่เท่ากับ Y ('X \= Y' มี

ความສໍາຄັນໃນກົງນີ້ ເພົ່າຄ້າເຮົາໄມ່ໄສເຈື່ອນໄຂນີ້ເຮົາຈະໄດ້ວ່າ ‘sibling(arthur,arthur)’ ເປັນຈົງ
– ດັນຄົນເດືອຍກັນເປັນພື້ນໜັງກັນເອງ)

4.3.2 ຕັ້ວຕັດ

ຕັ້ວຕັດ (*cut*) ເຊິ່ນແທນດ້ວຍ ‘!’ ຕັ້ວຕັດຈະມີລັກຂະນະການໃຊ້ງານທີ່ສັບສົນ ແຕ່ເປັນສ່ວນສໍາຄັນໃນໂປຣລູກເພື່ອເຂົ້າໃຫ້ໄດ້ໂປຣແກຣມທີ່ມີປະສິທິພາພ ດັ່ງນັ້ນຈໍາເປັນຕົ້ນທີ່ການເຂົ້າໃຈຢ່າງຄ່ອງແກ້ ພ້າທີ່ຫລັກຂອງຕັ້ວຕັດຄືການເປັ້ນແປງລຳດັບການທຳການຂອງໂປຣແກຣມໂດຍຈະປັບປຸງກັນການເກີດການຍ້ອນຮອຍ ໂດຍປັດໃປໂປຣລູກຈະພິສູນນີ້ຂ້ອງຄໍາຖາມໄດ້ອັດໃນມັດລະຈະຄັ້ນຫາເສັ້ນທາງທຸກເສັ້ນທາງທີ່ເປັນໄປໄດ້ດ້ວຍການຄັ້ນຫາແບບແນວລຶກກ່ອນ ແຕ່ນາງຄັ້ງເຮາຈທຽບລ່ວງໜ້າວ່າເສັ້ນທາງໃນການພິສູນນີ້ນາງເສັ້ນທາງໄມ່ຈໍາເປັນຕົ້ນຄັ້ນຫາກີ່ໄດ້ເພົ່າຈະໄນ້ມີຄຳຕອບໃນເສັ້ນທາງນັ້ນໂຮງເຮົາໄມ່ຕົ້ນການໃຫ້ໂປຣລູກຄັ້ນຫາໃນເສັ້ນທາງນັ້ນ ການໃຊ້ຕັດຕັກີ່ຈະຊ່ວຍໃຫ້ເຮາມາຮັດຄຸມການທຳການຂອງໂປຣລູກໄດ້ທຳໃຫ້ການທາຄຳຕອບຮຽວແລ້ວແມ່ນປະສິທິພາພມາກີ່ນີ້ ພ້າການທຳການຂອງຕັ້ວຕັດສາມາດແສດງໄຫ້ເຫັນໄດ້ດັ່ງ  ຮູບທີ່ 4-5 ການທຳການຂອງຕັ້ວຕັດ

ເຮາມາຮັດຄຸມການທຳການຂອງກົງ ‘ $H :- B_1, B_2, \dots, B_m, !, B_{m+1}, \dots, B_n$ ’ ອອກໄດ້ເປັນ 2 ສ່ວນຫລັກຄືອ່າວັນດ້ານໜ້າຂອງເຄື່ອງໝາຍ ! ແລະ ສ່ວນດ້ານຫລັງຂອງເຄື່ອງໝາຍ ການຍ້ອນຮອຍສາມາດເກີດຂຶ້ນໄດ້ກາຍໃນສ່ວນທັງສອງນີ້ ແຕ່ໄມ່ສາມາດຂັ້ນຝາກຕັດຕັດໄປໄດ້ ກລ່ວຄືອ້າມມີເປົ້າໝາຍ B_i ໄດ້ ຮະຫວ່າງ B_1, B_2, \dots, B_m ທຳໄມ່ສໍາເລື່ອດ້ວຍການແທນຄ່າຊຸດທີ່ຈະເກີດການຍ້ອນຮອຍເພື່ອພຍາຍາມທາການແທນຄ່າຊຸດໃໝ່ໄດ້ ແຕ່ເມື່ອການທຳການຜ່ານຕັດຕັດໄປແລ້ວແລ້ວມີເປົ້າໝາຍ B_j ໄດ້ ຮະຫວ່າງ B_{m+1}, \dots, B_n ທຳໄມ່ສໍາເລື່ອ ຈະໄມ່ສາມາດຍ້ອນກັບໄປໃຫ້ໃນ B_1, B_2, \dots, B_m ໄດ້ອີກ ອຍ່າງໄຮັດເມື່ອຜ່ານເດືອນຕັດມາແລ້ວການຍ້ອນຮອຍສາມາດເກີດຂຶ້ນໄດ້ກາຍໃນ B_{m+1}, \dots, B_n

ตัวตัดที่อธิบายด้านบนนี้ป้องกันการเกิดการย้อนรอยภายในกฎ ซึ่งทำให้การย้อนรอยไม่สามารถข้ามตัวตัดได้ นอกจากผลที่เกิดขึ้นภายในกฎแล้ว ตัวตัดยังส่งผลไม่ให้เกิดการย้อนรอยข้ามกฎด้วย กล่าวคือถ้าหากมีกฎอื่นๆ ของ H เช่นกฎข้อที่ 2 ของ H ด้านล่างนี้

$H :- C1, C2, \dots, Cp.$

เมื่อตัดตัดกฎเรียกใช้งาน (การทำงานผ่านตัวตัดแล้ว) และพบว่ากฎข้อที่ 1 ด้านบนทำไม่สำเร็จ การย้อนรอยก็ไม่สามารถมาทำกฎข้อที่ 2 ของ H นี้ได้ เปรียบเสมือนเกิดกำแพงกั้นระหว่างกฎ อย่างไรก็ได้กำแพงกันนี้ (ทั้งกำแพงภายในกฎและกำแพงระหว่างกฎ) จะเกิดขึ้นก็ต่อเมื่อตัวตัดกฎเรียกใช้งานแล้วเท่านั้น

โปรแกรมตัวอย่างในตารางที่ 4-11 ด้านล่างนี้แสดงผลที่เกิดขึ้นเมื่อเราใช้ตัวตัด

ตารางที่ 4-11 โปรแกรมตัวอย่างแสดงการทำงานของตัวตัด

```

01: a(X,W) :- p(X,Y), q(Y,Z), !, r(Z,W), s(W).
02: a(X,W) :- t(X,W).
03: p(1,2).
04: q(2,4).
05: q(2,5).
06: r(4,6).
07: r(4,8).
08: r(5,7).
09: s(6).
10: s(7).
11: t(1,9).

```

โปรแกรมด้านบนจะให้คำตอบเดียวคือ $W = 6$ สำหรับข้อคำถาม '?- a(1,W).' พิจารณาการทำงานของโปรแกรมตามรูปที่ 4-6 ด้านล่างนี้

```

a(1,W) :-
  p(1,2)
  q(2,4)
  !
  r(4,6) ✗ r(4,8)
  s(6)
W=6;

```

รูปที่ 4-6 ตามรอยการทำงานโปรแกรมกรณีมีตัวตัด

จากตัวอย่างในรูปจะเห็นได้ว่าเมื่อการทำงานผ่านตัวตัดมาแล้ว ได้ $W = 6$ และเมื่อໂປຣລູກນັບໃຫຍ່ອນຮອຍด້ວຍຄຳສັ່ງໃຫ້ຫາຄຳຕອນອື່ນ ‘;’ ຈຶ່ງຍັນຮອຍມາທີ່ ‘t(4,W)’ ໄດ້ ‘t(4,8)’ ແຕ່ ‘t(8)’ ເປັນເທົ່າແລ້ວມີສາມາດຍັນຮອຍໄດ້ອັກແລ້ວ ຈຶ່ງມີຄຳຕອນອື່ນ

ເຮົາໄດ້ພິຈາລະນາທຳຄວາມເຂົາໃຈເຮືອງການທຳການຂອງຕົວຕັດດ້ານນັນແລ້ວ ອ່າງໆໄກກີດກາຈະເຂີຍໂປຣແກຣມທີ່ມີຕົວຕັດໄດ້ອິ່ນມີປະສິທິກາພຫຼືກໍາມກີມກົດກົນໂປຣແກຣມໜຶ່ງໆ ເພື່ອໃຫ້ເຂົາໃຈໄດ້ອ່າງໆດີໃນກຣີນີ້ທີ່ໂປຣແກຣມນັ້ນປະກອບດ້ວຍຕົວຕັດ ເຮົາຈໍາເປັນຕ້ອງເຮີຍຮູ່ຮູ່ປະບັບການໃຊ້ຕົວຕັດ ຕລອດຈົນຄວາມໝາຍຂອງຕົວຕັດໃນຮູ່ຮູ່ປະບັບນັ້ນໆ ຕ່ອໄປຈະກຳລ່າວສິ່ງຮູ່ຮູ່ປະບັບການເຂີຍໂປຣແກຣມ ໂດຍໃຊ້ຕົວຕັດດັ່ງຕ້ອງໄປນີ້

ການໃຊ້ຕົວຕັດຮ່ວມກັບເພຣດີເຄີຕ ‘fail’

ຮູ່ຮູ່ປະບັບແຮກຂອງການໃຊ້ຕົວຕັດເປັນການໃຊ້ຮ່ວມກັບເພຣດີເຄີຕ fail ທີ່ເປັນເພຣດີເຄີຕໃນຕົວໂປຣລູກມີຄວາມໝາຍວ່າ ‘ເປັນເທົ່າ’ ຖຸກຄໍ່ງທີ່ສູກເຮີຍກ ດັ່ງນັ້ນເມື່ອເຮີຍກ fail ຈະເກີດກາຍັນຮອຍທັນທີ່ fail ເປັນເພຣດີເຄີຕທີ່ໄມ້ມີອົບກິວເມີນຕໍ່

ສມມືວ່າເຮົາຕ້ອງການເຂີຍໂປຣແກຣມອີ້ນບາຍຄວາມສັນພັນນີ້ວ່າ ‘mary ຂອບສັຕ່ວົງທຸກຕົວທີ່ໄມ້ໃໝ່’ ເຮົາຈະເຂີຍໂປຣແກຣມທີ່ເປັນການໃຊ້ຈາກຮ່ວມກັນຮ່ວມກັນຕົວຕັດກັບ fail ໄດ້ດັ່ງຕາງໆທີ່ 4–12 ດ້ວນລັງນີ້

ຕາງໆທີ່ 4–12 ໂປຣແກຣມຕົວຍ່າງແສດງການໃຊ້ຈາກຂອງຕົວຕັດຮ່ວມກັບ fail

```
1: like(mary,X) :- snake(X), !, fail.
2: like(mary,X) :- animal(X).
3: snake(small_snake).
4: snake(big_snake).
5: animal(dog).
```

ກົງຂ້ອແຮກແສດງວ່າຄໍາ X ເປັນງູ້ແລ້ວ ‘like(mary,X)’ ຈະເປັນເທົ່າທັນທີ່ແລ້ວມີສາມາດຍັນຮອຍໄປຢັງກົງຂ້ອ່ນໆ ໄດ້ອັກ ດັ່ງນັ້ນຂ້ອຄໍາຄາມ ‘?- like(mary,small_snake).’ ຈະໄດ້ຄຳຕອນເປັນ No ແຕ່ຄໍາ X ເປັນສັຕ່ວົງ່ນໆ ທີ່ໄມ້ໃໝ່ ຈະໄດ້ວ່າກົງຂ້ອ່ທີ່ 1 ໄມ່ເປັນຈິງຈາກເຈິ່ງເວັບໄວ້ ‘snake(X)’ ທຳໄໝຍັນຮອຍມາທີ່ກົງທີ່ 2 ໄດ້ແລ້ວທຳສໍາເຮົາດ້ວຍເຈິ່ງເວັບໄວ້ ‘animal(X)’ ເຊັ່ນຂ້ອຄໍາ ‘?- like(mary,dog).’ ຈະໄໝຄຳຕອນເປັນ Yes

ຕົວຕັດເຂີຍ

ຕົວຕັດເຂີຍ (green cut) ເປັນຮູ່ຮູ່ປະບັບການໃຊ້ຕົວຕັດຮູ່ຮູ່ປະບັບນີ້ ໂປຣແກຣມທີ່ມີຕົວຕັດເຂີຍ ສາມາດລົບຕົວຕັດເຂີຍວ່າອັກໄດ້ໂດຍໄມ້ກະທບຕ່ອງຄວາມໝາຍຂອງໂປຣແກຣມ (ຄຳຕອນທີ່ໄດ້ຈາກໂປຣແກຣມເໜີອັນເດີມທຸກປະກາດ) ແຕ່ປະສິທິກາພຂອງໂປຣແກຣມທີ່ມີຕົວຕັດເຂີຍຈະດີກວ່າ ຕົວຕັດທີ່ມີຄຸນສົມບັດຕິດຮ່າມກັບຕົວຕັດເຂີຍກີ່ອື່ນຕົວຕັດແດງ (red cut) ດັ່ງຈະກຳລ່າວຕ້ອງໄປ

ตัวตัดเชี่ยวใช้กับโปรแกรมซึ่งประกอบด้วยเพรดิเกตที่มีการทดสอบเชิงกำหนด (deterministic test) ตัวอย่างของการทดสอบแบบนี้ เช่น ‘ $X < Y$ ’ หรือ ‘ $X = Y$ ’ หรือ ‘ $X > Y$ ’ ซึ่งจะเป็นจริงแค่กรณีใดกรณีหนึ่งเท่านั้น คือถ้า X น้อยกว่า Y แล้ว X จะไม่เท่ากับ Y และ X จะไม่มากกว่า Y

ตัวอย่างในตารางที่ 4-13 ด้านล่างนี้ เป็นโปรแกรมหาค่าต่ำสุดระหว่างตัวเลข 2 ตัวโดยใช้ตัวตัดเชี่ยว ‘minimum(X,Y,Z)’ มีความหมายว่า Z เป็นค่าต่ำสุดระหว่างตัวเลข 2 ตัวของ X กับ Y

ตารางที่ 4-13 โปรแกรมหาค่าต่ำสุดของตัวเลข 2 ตัว

1: <code>minimum(X,Y,Z) :- X < Y, !, Z = X.</code>
2: <code>minimum(X,Y,Z) :- X = Y, !, Z = X.</code>
3: <code>minimum(X,Y,Z) :- X > Y, Z = Y.</code>

กฎข้อที่หนึ่งมีความหมายว่าถ้า X น้อยกว่า Y และ Z จะมีค่าเท่ากับ X กฎข้อที่ 2 คือถ้า X เท่ากับ Y และ Z เท่ากับ X และกฎข้อที่ 3 คือถ้า X มากกว่า Y และ Z เท่ากับ Y การใส่ตัวตัดเข้าที่ด้านหลังของการทดสอบเชิงกำหนด ($X < Y$, $X = Y$ และ $X > Y$) จะไม่เปลี่ยนความหมายของโปรแกรม เนื่องจากว่าถ้า $X < Y$ และตัวตัดจะถูกเรียกทำให้การย้อนรอยไม่สามารถทำได้ และในกรณีที่โปรแกรมนี้ไม่ใส่ตัวตัดก็จะมีความหมายเหมือนกัน เพราะถ้า $X < Y$ และไม่มีตัวตัด การย้อนรอยจะเลือกกฎข้อที่ 2 และ 3 ได้แต่ก็ไม่สามารถได้คำตอบอื่น เพราะว่า X จะไม่เท่ากับ Y และ X ก็จะไม่มากกว่า Y ดังนั้นจะเห็นได้ว่าการใส่ตัวตัดไม่ทำให้ความหมายของโปรแกรมเปลี่ยนไป

แม้ว่าความหมายของโปรแกรมที่ใส่ตัวตัดกับไม่ใส่จะเหมือนกัน แต่สิ่งที่เราได้เพิ่มขึ้นจากการใส่ตัวตัดคือโปรแกรมจะทำงานเร็วขึ้น สมมติว่าข้อคำถามคือ ‘?- `minimum(4,5,Z)`.’ ข้อคำถามนี้จับคู่ได้กับส่วนหัวของกฎที่ 1 และทดสอบส่วนลำตัวของกฎ พบว่า ‘ $4 < 5$ ’ เป็นจริง การทำงานจึงผ่านตัวตัดและได้ $Z = 4$ เป็นคำตอบ ณ จุดนี้ถ้ามีการย้อนรอยเกิดขึ้นอาจเนื่องมาจากผู้ใช้ต้องการคำตอบอื่น หรืออาจเกิดจากโปรแกรมนี้เป็นโปรแกรมย่ออยู่ที่ถูกเรียกใช้ด้วยโปรแกรมอื่นและที่โปรแกรมอื่นนั้นมีเพรดิเกตเป้าหมายบางตัวทำไม่สำเร็จทำให้เกิดการย้อนรอยซึ่งส่งผลให้โปรแกรมนี้เกิดการย้อนรอยด้วย เมื่อเกิดการย้อนรอยขึ้นโปรแกรมนี้จะตอบได้ทันทีว่าไม่มีคำตอบอื่น เพราะว่าตัวตัดถูกสั่งทำงานแล้วทำให้การย้อนรอยในโปรแกรมนี้ไม่สามารถเลือกกฎข้ออื่นได้อีก

สมมติว่าโปรแกรมนี้ไม่มีตัวตัดอยู่เลย และให้ข้อคำถามเหมือนเดิมคือ ‘?- `minimum(4,5,Z)`.’ ข้อคำถามนี้จับคู่ได้กับส่วนหัวของกฎที่ 1 และทดสอบส่วนลำตัวของกฎพบว่า ‘ $4 < 5$ ’ เป็นจริงและได้ $Z = 4$ เป็นคำตอบ ณ จุดนี้ถ้ามีการย้อนรอยเกิดขึ้นโปรแกรมจะย้อนรอยมายังกฎข้อที่ 2 ได้ เพราะไม่มีตัวตัดและจับคู่กับส่วนหัวของข้อที่ 2 ได้

จากนั้นทดสอบส่วนลำตัวของกฎโดยทดสอบว่า ‘ $4 = 5$ ’ หรือไม่และพบว่าไม่เป็นจริง จึงย้อนรอยอีกไปยังกฎที่ 3 จับคู่กับส่วนหัวของกฎที่ 3 ได้ ทดสอบว่า ‘ $4 > 5$ ’ และพบว่าไม่เป็นจริง จึงได้ผลว่าไม่มีคำตอบอื่นเหมือนกับกรณีที่มีตัวตัด แต่จะทำงานซ้ำกันว่าแล้วไปทดสอบในส่วนที่ไม่จำเป็น กล่าวคือถ้า ‘ $4 < 5$ ’ เป็นจริงก็ไม่จำเป็นต้องทดสอบในกฎข้อที่ 2 และ 3

ตัวอย่างการเขียนโปรแกรมโดยใช้ตัวตัดเขียนอีกด้วยคือโปรแกรม ‘`insert(X, Ys, Zs)`’ กำหนดให้แทรกรสماชิก X เข้าไปในรายการของตัวเลข Ys ที่เรียงจากน้อยไปมาก ได้เป็นรายการของตัวเลข Zs ที่เรียงลำดับจากน้อยไปมาก เช่น ‘`insert(2,[0,1,4,6],[0,1,2,4,6])`’ เป็นต้น โปรแกรมเป็นดังตารางที่ 4-14 ด้านล่างนี้

ตารางที่ 4-14 โปรแกรมแทรกรตัวเลข

```

1: insert(X, [], [X]).  

2: insert(X, [Y|Ys], [Y|Zs]) :- X > Y, !,  

   insert(X, Ys, Zs).  

3: insert(X, [Y|Ys], [X,Y|Ys]) :- X <= Y.

```

โปรแกรมนี้ประกอบด้วยอนุประโยคฐาน 2 ข้อ (ข้อที่ 1 กับข้อที่ 3) และอนุประโยคเรียกช้า 1 ข้อ การเขียนโปรแกรมแทรกรสماชิกนี้มูลก็เช่นเดียวกับโปรแกรมเรียกช้าอื่นๆ คือเราเริ่มเขียนจากอนุประโยคฐานก่อน โดยสมมติว่าจะลดลำดับของอาร์กิวเม้นต์ตัวที่ 2 ดังนั้นจะได้ว่ารายการที่มีลำดับต่ำสุดคือ `[]` ได้เป็น ‘`insert(X, [], [X])`’ คือแทรกร X เข้าไปในรายการว่างจะได้รายการที่มีสมาชิกตัวเดียวคือ X จากนั้นเราก็เขียนอนุประโยคเรียกช้าโดยลดลำดับของอาร์กิวเม้นต์ตัวที่ 2 จะได้ว่า ‘`insert(X, [Y|Ys], ?) :- insert(X, Ys, Zs)`’ อย่างไรก็ได้การแทรกรสماชิกไว้ในรายการที่ต้องคำนึงถึงลำดับนั้น เราต้องใช้การเปรียบเทียบด้วย ดังนั้นเราเพิ่มการเปรียบเทียบเข้าไปในกฎข้อที่ 2 ได้เป็น ‘`insert(X, [Y|Ys], ?) :- X > Y, insert(X, Ys, Zs)`’ ณ จุดนี้เราพบว่าถ้า $X > Y$ และแทรกร X ใน Ys ได้ Zs แล้ว การแทรกร X ในรายการ Ys ตัวเดิมและมี Y บวกอยู่ข้างหน้า (คือรายการ `[Y|Ys]`) ก็ต้องได้รายการ Zs ตัวเดิมและมี Y ปะไว้ข้างหน้า (คือรายการ `[Y|Zs]`) ด้วยอย่างเช่นให้การแทนค่าคือ $\{X = 2, Y = 0, Ys = [1,4,6]\}$ เราจะได้ว่า `insert(2,[0,1,4,6],?) :- 2 > 0, insert(2,[1,4,6],Zs)` ที่จุดนี้ให้สมมติค่า Zs ที่เป็นค่าถูกต้องแล้วหาว่า ? ควรเป็นเท่าไรของ Zs กรณีนี้ได้ $Zs = [1,2,4,6]$ ดังนั้น ? (ซึ่งควรเป็น `[0,1,2,4,6]`) จึงเท่ากับ `[Y|Zs]`

สุดท้ายพบว่าการทดสอบ $X > Y$ ในกฎข้อที่ 2 ยังทดสอบไม่ครบถ้วน ต้องมีกรณีที่ $X <= Y$ (X น้อยกว่าหรือเท่ากับ Y) ด้วย เราจึงเพิ่มกฎข้อที่ 3 คือ ‘`insert(X, [Y|Ys], ?) :- X <= Y`’ กรณีนี้หา ? ได้อย่าง่ายว่าถ้า $X <= Y$ ดังนั้นแทรกร X ไว้ในรายการที่มีส่วนหัวคือ Y ที่มากกว่าหรือเท่ากับ X ก็จะต้องนำ X ไว้หน้า Y ได้ ? เป็น `[X,Y|Ys]`

โปรแกรม ณ จุดนี้คือ

`insert(X,[],[X]).`

`insert(X,[Y|Ys],[Y|Zs]) :- X > Y, insert(X,Ys,Zs).`

`insert(X,[Y|Ys],[X,Y|Ys]) :- X <= Y.`

และเราพบว่ากฎที่ 2 และ 3 มีการทดสอบเชิงกำหนด เราจึงใส่ตัวตัดไว้หลังการทดสอบของกฎข้อที่ 2 ได้โปรแกรมดัง [ตารางที่ 4-14](#)

ตัวตัดแดง

ตัวตัดแดง (red cut) ใช้ในกรณีที่เราต้องการจะเงื่อนไขบางตัวออกจากโปรแกรมแล้วแทนที่ด้วยตัวตัด การใช้ตัวตัดประเภทนี้ต้องระวังเนื่องจากว่าหากเราไปนำตัวตัดออกจากโปรแกรมความหมายของโปรแกรมจะเปลี่ยนไป ถ้าต้องการให้ความหมายคงเดิมจะต้องนำเงื่อนไขที่ละไว้กลับเข้าที่เดิม ดังนั้นเมื่อเราไปอ่านโปรแกรมหนึ่งๆ ถ้าหากไม่ระวัง ไปลบตัวตัดแล้วลบตัวตัดออก โปรแกรมจะมีความหมายผิดไปจากเดิมได้ถ้าตัวตัดนั้นเป็นตัวตัดแดง ตัวอย่างของตัวตัดประเภทนี้ เช่นถ้าเราต้องการเขียนโปรแกรม 'if_then_else(P,Q,R)' โดยที่ P, Q และ R แทนความสัมพันธ์ใดๆ ซึ่งมีความหมายว่า ถ้า P เป็นจริงสั่งทำ Q ถ้าไม่จริงทำ R สามารถเขียนโปรแกรมได้ดัง [ตารางที่ 4-16](#)

ตารางที่ 4-15 โปรแกรม 'ถ้าแล้ว' กรณีใช้ตัวตัด

1: <code>if_then_else(P,Q,R) :- P, !, Q.</code>
2: <code>if_then_else(P,Q,R) :- R.</code>

กฎข้อแรกหมายถึง P จริงแล้วเรียกตัวตัดทำงาน ทำให้การย้อนรอยไม่เกิดหลังจากนี้แล้วทำ Q และผลของการทำ Q ไม่ว่าจะจริงหรือไม่ก็ตาม จะไม่มาทำ R เพราะตัวตัดได้ตัดทางเลือกนี้ทิ้งแล้ว แต่ถ้า P ไม่จริงโปรแกรมจะย้อนรอยมาทำกฎข้อที่ 2 ได้ เพราะตัวตัดยังไม่ถูกเรียกใช้ ดังนั้นจะทำ R ซึ่งตรงกับความหมายของ 'if_then_else' ที่ต้องการ

แต่ถ้าเราลบตัวตัดออกจากความหมายจะผิดเพี้ยนไป กล่าวคือไม่ว่า P จะจริงหรือไม่ก็จะทำ R เสมอ ซึ่งไม่ใช่ความหมายที่ต้องการ เช่นถ้า P เป็นจริงแล้วเกิด Q เป็นเท็จหรืออาจเกิดการย้อนรอยจากสาเหตุอื่นๆ (เช่นผู้ใช้สั่งหาคำตอบอื่นหรือเกิดการย้อนรอยจากโปรแกรมที่เรียกใช้โปรแกรมนี้ ฯลฯ) โปรแกรมจะย้อนรอยมาทำกฎข้อที่ 2 ซึ่งไม่ควรเป็นเช่นนั้น เพราะ P เป็นจริงไม่ควรทำ R

โปรแกรมที่สมมูลกับโปรแกรมที่ไม่ใช้ตัวตัดต้องเขียนดัง [ตารางที่ 4-16](#) ต่อไปนี้

ตารางที่ 4-16 ໂປຣແກຣມຄໍາແລ້ວການນີ້ໃຫ້ຕັ້ງຕັດ

```
1: if_then_else(P,Q,R) :- P, Q.
2: if_then_else(P,Q,R) :- not(P), R.
```

ເມື່ອພິຈາລານກູ້ຂ້ອທີ່ 1 ໃນໂປຣແກຣມແບບນີ້ໃຫ້ຕັ້ງຕັດນີ້ ຈະເຫັນວ່າໃນການນີ້ທີ່ P ເປັນຈິງຈະທຳ Q ແລະ ອື່ງແນ່ຈະເກີດກາຍັນຮອຍກີ່ຈະໄໝ່ທຳ R ເພະກູ້ຂ້ອທີ່ 2 ມີການຕຽບສອນວ່າຈະທຳ R ໄດ້ກີ່ຕ່ອນເມື່ອ not(P) ຕ້ອງເປັນຈິງ (ຫົວໜ້າ P ເປັນເທົ່ານີ້ເອງ) ອ່າງໄປກີ່ໂປຣແກຣມນີ້ມີປະສິທິກິພາພົກພະກວດກາຍັນຮອຍຂຶ້ນໂປຣແກຣມຈະລົງມາກູ້ຂ້ອທີ່ 2 (ເພະໄມ້ມີຕັ້ງຕັດ) ແລະ ທົດສອນວ່າ not(P) ເປັນຈິງຫົວໜ້າ ການທົດສອນນີ້ຕ້ອງທຳ P ດູກກຸນແລະເມື່ອທຳ P ດູກຈະໄດ້ວ່າ P ເປັນຈິງ ດັ່ງນັ້ນ not(P) ເປັນເທິງ ຈະເຫັນໄດ້ວ່າ P ດູກທົດສອນຄື່ງ 2 ຄົງ (ທີ່ກູ້ຂ້ອທີ່ 1 ກັບກູ້ຂ້ອທີ່ 2) ເກີດການທຳກັນຫຼຸດຂຶ້ນຕ່າງຈາກໂປຣແກຣມແບບນີ້ໃຫ້ຕັ້ງຕັດທີ່ທຳ P ຄົງເດືອນ

ຕ້ວອຍ່າງໂປຣແກຣມທີ່ໃຫ້ຕັ້ງຕັດແດງອີກຕ້ວອຍ່າງທີ່ຈະຍົກໃຫ້ດູນເປັນໂປຣແກຣມ ‘delete(Xs,X,Ys)’ ທໍາທັນທີລົບສາມາຝຶກທຸກຕັ້ງທີ່ເທົກກັນ X ອອກຈາກຮາຍການ Xs ໄດ້ເປັນຮາຍການ Ys ເຊັ່ນ ‘delete([1,a,2,3,a],a,[1,2,3])’ ເປັນຕັ້ນ ໂປຣແກຣມເປັນດັ່ງ [ตารางที่ 4-17](#) ຕ່ອໄປນີ້

ตารางที่ 4-17 ໂປຣແກຣມລົບສາມາຝຶກ

```
1: delete([X|Ys],X,Zs) :- !, delete(Ys,X,Zs).
2: delete([Y|Ys],X,[Y|Zs]) :- !, delete(Ys,X,Zs).
3: delete([],X,[]).
```

ເຮົາເຮີມຈາກການເຂົ້ານອນຫຼຸດໂປຣແກຣມກ່ອນ ໂດຍສົມຜົນວ່າອົບກິວເມນຕົວແກ່ເປັນອີນຝຸດ ຕັ້ງທີ່ສາມເປັນເອົາຕຸພູດ ທໍາການລົດລຳດັບຂອງອົບກິວເມນຕົວທີ່ໜຶ່ງໄດ້ວ່າ ‘delete([],X,?)’ ແລະພວກຫຼຸດ X ອອກຈາກຮາຍການວ່າກ່ອງໄດ້ຮາຍການວ່າ ດັ່ງນັ້ນ ? = [] ໄດ້ກູ້ຂ້ອທີ່ 3 (ທີ່ໜ້ານຸ່ມປະໂຍດຈຸານໄວ້ເປັນກູ້ຂ້ອທີ່ 3 ເນື່ອຈາກເຫຼຸດຜລທາງດ້ານປະສິທິກິພາພອງໂປຣແກຣມ) ຈາກນັ້ນເຂົ້ານອນຫຼຸດໂປຣແກຣມເຮົາຕຸພູດໄດ້ວ່າ ‘delete([X|Ys],X,?) :- delete(Ys,X,Zs)’ ເປັນການນີ້ທີ່ຕັ້ງທີ່ຕ້ອງການລົບອອກເປັນຕັ້ງແກ່ໃນຮາຍການ (X = X) ເມື່ອເຮົາສົມຜົນວ່າລົບ X ອອກຈາກ Ys ໄດ້ Zs (‘delete(Ys,X,Zs)’ ທີ່ສ່ວນລຳຕັ້ງ) ແລ້ວຫວ່າ ? ເປັນເທົ່າໄວ່ຂອງ Xs ເຮົາຈະໄດ້ວ່າ ? ເທົກກັນ Zs ເພະວ່າຕັ້ງການລົບ X ອອກຈາກ Ys ໄດ້ Zs ແລ້ວ ການລົບ X ອອກຈາກ Ys ຕັ້ງເດີມທີ່ມີ X ເໜື່ອນກັນມາປະໄວ້ຂ້າງໜ້າ (ຫຼັງໝາຍຄື່ງ [X|Ys]) ກົດ້ອງໄດ້ຜລັພົບເດີມຄື່ອ Zs ເຊັ່ນຕັ້ງ ‘delete([2,3,a],a,[2,3])’ ເປັນຈິງແລ້ວ ‘delete([a,2,3,a],a,?)’ ກົດ້ອງໄດ້ ? ເທົ່າເດີມຄື່ອ [2,3] ເປັນຕັ້ນ

กรณีของอนุประโยคฐานด้านบนเป็นกรณีที่ X เหมือนกับตัวแรกของรายการที่จะลบ ซึ่งยังไม่ครอบคลุมกรณีที่ไม่เหมือนกัน ดังนั้นเราเพิ่มกฎอีกข้อที่เป็นข้อที่ 2 ในตาราง ได้เป็น 'delete([Y|Ys],X,?) :- X \= Y, delete(Ys,X,Zs)' แล้วหาว่า ? คืออะไร กรณีนี้จะได้ว่าถ้าลบ X ออกจาก Ys ได้ Zs แล้วลบ X ออกจาก Ys ตัวเดิมแต่มี Y ปะไว้ข้างหน้า (และ Y ไม่เท่ากับ X) ก็จะได้ Zs ที่มี Y ปะไว้หน้า ดังนั้นได้ ? เป็น $[Y|Zs]$

ณ จุดนี้โปรแกรมที่เราได้เป็น

```
delete([X|Ys],X,Zs) :- delete(Ys,X,Zs).
delete([Y|Ys],X,[Y|Zs]) :- X \= Y, delete(Ys,X,Zs).
delete([ ],X,[ ]).
```

เราจะเห็นว่า มีการตรวจสอบเงื่อนไข $X \= Y$ ที่กฎข้อที่ 2 ดังนั้นถ้าเราจะละเงื่อนไขนี้ เรายังไส่ตัวตัดໄว่ที่กฎข้อแรกได้เป็น

```
delete([X|Ys],X,Zs) :- !, delete(Ys,X,Zs).
delete([Y|Ys],X,[Y|Zs]) :- delete(Ys,X,Zs).
delete([ ],X,[ ]).
```

ด้วยเหตุผลดังที่อธิบายไว้ข้างต้น โปรแกรมนี้จะทำงานมีประสิทธิภาพดีกว่าโปรแกรมที่ไม่มีตัวตัด ส่วนเครื่องหมาย ! ที่ใส่ไว้ที่กฎข้อที่ 2 ตามตารางที่ 4-17 ก็ เพราะว่าหากกฎข้อที่ 2 ถูกเรียกใช้งานแล้ว เราไม่จำเป็นต้องทดลองจับคู่ข้อความกับกฎข้อที่ 3 เพราะจับคู่ไปก็ทำไม่สำเร็จ (กฎข้อที่ 3 มีอาร์กิวเมนต์ตัวแรกเป็นรายการว่างต่างจากของกฎข้อที่ 1 และ 2) เราจึงยุกอ่อนแล้วว่าจับคู่ไปก็ทำไม่สำเร็จ เราจึงใส่ตัวตัดเพื่อให้โปรแกรมไม่ต้องทดลองทำกฎข้อที่ 3 ในกรณีที่กฎข้อที่ 2 จับคู่สำเร็จซึ่งจะช่วยให้โปรแกรมทำงานรวดเร็วขึ้นอีก

เอกสารอ่านเพิ่มเติมและแบบฝึกหัด

ตำราของ Bratko [Bratko, 1990] มีเนื้อหาของภาษาโปรแกรมภาษาลอกิวัค่อนข้างสมบูรณ์ โดยเฉพาะอย่างยิ่งได้เน้นเรื่องการเขียนโปรแกรมโปรแกรมลอกิวัคหับงานทางปัญญาประดิษฐ์ซึ่งมีตัวอย่างให้ดูจำนวนมาก สำหรับผู้ที่ต้องการศึกษาอย่างจริงจังเกี่ยวกับการเขียนโปรแกรมภาษาที่นี้ ควรอ่าน [Sterling & Shapiro, 1994] ซึ่งเป็นหนังสือเล่มที่เยี่ยมที่สุดเล่มหนึ่งของโปรแกรมมีเทคนิคการเขียนโปรแกรมต่างๆ ให้ศึกษาจำนวนมาก

บรรณานุกรม

- Bratko, I. (1990) *Prolog: Programming for Artificial Intelligence*. Second Edition. Addison Wesley.
- Sterling, L. and Shapiro, E. (1994) *The Art of Prolog: Advanced Programming Techniques*. Second Edition. The MIT Press.

ແບບຝຶກທັດ

1. ຈົນເຂີຍນ `reverse(Xs, Ys)` ທີ່ທຳຫັນທີ່ສັບລຳດັບສາມາດໃນຮາຍການ `Xs` ເປັນຮາຍການ `Ys` (ມີສາມາດກົດຢູ່ໃນລຳດັບຕຽບກັນຂັ້ນກັບຂອງ `Xs`) ໂດຍໄມ້ໃຊ້ `append` (ໂປຣແກຣມໃນຕາງໆທີ່ 4-9)

ຕ້ວອຍ່າງ :
`reverse([a, b, c], Ys) ໄດ້ Ys = [c, b, a]`
`reverse([1, 2, 3, 4, 5], Ys) ໄດ້ Ys = [5, 4, 3, 2, 1]`
`reverse(Xs, [m, y, o, b]) ໄດ້ Xs = [b, o, y, m]`
2. ຈົນເຂີຍໂປຣແກຣມ `last(As,L)` ທີ່ທຳຫັນທີ່ຫາ `L` ຜູ້ເປັນສາມາດຕັ້ງສຸດທ້າຍໃນຮາຍການ `As`

ຕ້ວອຍ່າງ :
`last([1, 2, 3], L) ໄດ້ L = 3`
`last([a,b,c,d], L) ໄດ້ L = d`
3. ຈົນເຂີຍໂປຣແກຣມ `minus(X,Y,Z)` ທີ່ທຳຫັນທີ່ຫາ `Z` ຜູ້ເປັນມີຄ່າເທົ່າກັນ `X` ລົບ `Y` ໂດຍທີ່ `X, Y, Z` ເປັນຈຳນວນຮຽມชาຕີ

ຕ້ວອຍ່າງ :
`minus(s(s(s(s(0)))),s(s(s(0))),Z) ໄດ້ Z = s(0)`
`minus(s(s(s(s(0)))),s(s(0)),Z) ໄດ້ Z = s(s(0))`
4. ຈົນເຂີຍໂປຣແກຣມ `power(X,Y,Z)` ທີ່ທຳຫັນທີ່ຫາ `Z` ຜູ້ເປັນມີຄ່າເທົ່າກັນ `X` ຍາກກຳລັງ `Y` ໂດຍທີ່ `X, Y, Z` ເປັນຈຳນວນຮຽມชาຕີ

ຕ້ວອຍ່າງ :
`power(s(s(0)),s(s(s(0))),Z) ໄດ້ Z = s(s(s(s(s(s(0))))))`
`power(s(s(0)),0,Z) ໄດ້ Z = s(0)`
5. ພິຈາລະນາໂປຣແກຣມໃນຕາງໆທີ່ 4-11 ສມຜົນວ່າໂປຣແກຣມນີ້ໄມ້ມີຕົວຕັດເລີຍ ຂ້ອຄໍາຄາມ ‘?- `a(1,W)`.’ ຈະມີຄໍາຕອບກີ່ຕ້ວອຍ່າໄປບ້າງ
6. ພິຈາລະນາໂປຣແກຣມດ້ານລ່າງນີ້

```
whoami([X|Xs],Y,[X|Ls],Bs) :- X <= Y, whoami(Xs,Y,Ls,Bs).
```

```
whoami([X|Xs],Y,Ls,[X|Bs]) :- X > Y, whoami(Xs,Y,Ls,Bs).
```

```
whoami([],Y,[],[]).
```

ໜ້າຍເຫດຸ: ກໍາຫັດໄຫ້ອັບກິວເມັນທີ່ 2 ຕ້ວແຮກເປັນອິນພຸດ (ຄູກແທນຄ່າແລ້ວເວລາເຮີຍກ)

- ຈົນອີ້ນບາຍວ່າໂປຣແກຣມນີ້ໃຊ້ທຳອະໄຣ

- จงเขียนโปรแกรมนี้ใหม่โดยใช้ cut (!)

7. พิจารณาปริมาณกระต่อไปนี้

กาลครั้งหนึ่งนานมาแล้ว มีชาย 5 คนอาศัยอยู่ในบ้านคนละหลังซึ่งแต่ละหลังมีสีแตกต่างจากบ้านอื่น ชายทั้ง 5 มีเชื้อชาติแตกต่างกัน เลี้ยงสัตว์คนละชนิด ชอบเครื่องดื่มและบุหรี่คนละยี่ห้อ เงื่อนไขของปัญหามีดังนี้

- (1) คนอินเดียอาศัยอยู่ในบ้านสีแดง
- (2) คนไทยเลี้ยงสุนัข
- (3) กาแฟเป็นเครื่องดื่มในบ้านสีเขียว
- (4) คนลาวดื่มชา
- (5) บ้านสีเขียวอยู่ด้านขวามือของบ้านสีดำ
- (6) คนสูบกรุงทองเลี้ยงทาก
- (7) สายฟันถูกสูบในบ้านสีเหลือง
- (8) نمเป็นเครื่องดื่มในบ้านที่อยู่ตรงกลาง
- (9) คนพม่าอยู่ที่บ้านริมสุดด้านซ้ายมือ
- (10) คนที่สูบเกล็ดทองอาศัยอยู่ในบ้านที่ติดกับบ้านที่เลี้ยงแมว
- (11) สายฟันถูกสูบในบ้านที่ติดกับบ้านที่เลี้ยงม้า
- (12) คนสูบมาร์ลboro ดื่มน้ำส้ม
- (13) คนญี่ปุ่นสูบเชวนสตาร์
- (14) คนพม่าอาศัยอยู่ในบ้านที่ติดกับบ้านสีฟ้า

คำถามคือ ใครเป็นเจ้าของกบและใครดื่มกระทิงแดง ?

จงเขียนโปรแกรมโปรแกรมเพื่อแก้ปัญหานี้ พร้อมทั้งแสดงผลลัพธ์ของคำถามว่าคืออะไรและเป็นไปได้ทั้งหมดกี่ทางเลือก

การประมวลผลภาษาธรรมชาติ

5

การประมวลผลภาษาธรรมชาติ – เอ็นแอลพี (*Natural Language Processing – NLP*) เป็นการกระบวนการที่จะทำให้คอมพิวเตอร์เข้าใจภาษาตามนุษย์โดยรับอินพุตเป็นข้อความในภาษาหนึ่งๆ และทำความเข้าใจว่าผู้ป้อนข้อความกล่าวถึงอะไร แม้ว่าภาษาตามนุษย์หรือภาษาธรรมชาตินั้นมีคุณสมบัติต่างๆ ที่ทำให้เราสามารถสื่อสารและเข้าใจกันได้ แต่การที่จะทำให้ระบบเอ็นแอลพีเข้าใจภาษาธรรมชาตินั้นทำได้ยากลำบาก อันเนื่องมาจากคุณสมบัติทางภาษาที่ทำให้เกิดปัญหาและความยุ่งยากในพัฒนาระบบเอ็นแอลพี อย่างเช่นด้านล่างนี้

- ประโยชน์เป็นคำอธิบายสารสนเทศที่ผู้พูดตั้งใจถ่ายทอดไปยังผู้ฟัง เช่นประโยชน์

Some dogs are outside.

อาจสื่อความหมายได้หลากหลาย เช่น

Some dogs are on the lawn.

Three dogs are on the lawn.

Rover, Tripp and Spot are on the lawn.

ข้อดี: ภาษาธรรมชาติให้ผู้พูดสื่อสารด้วยข้อความที่คลุ่มเครื่อหรือชัดเจนเท่าที่ผู้พูดต้องการได้โดยผู้พูดอาจละเอียดข้อความบางอย่างที่ผู้ฟังรู้อยู่แล้ว

- ประโยชน์เดียวกันอาจหมายถึงสิ่งของหลายอย่างโดยขึ้นอยู่กับสถานการณ์ที่พูด เช่น

Where's the water?

ถ้าประโยชน์นี้ถูกพูดในห้องทดลองเคมี “weter” ก็อาจหมายถึงน้ำบริสุทธิ์ ถ้าพูดตอนหิว ก็อาจหมายถึงน้ำที่เรายินดีมีได้ หรือพูดในขณะที่กำลังซ้อมหลังคาบ้านที่ร้าว ก็อาจจะหมายถึงน้ำที่เป็นน้ำรั่วลงในบ้าน

ข้อดี: ภาษาช่วยให้เราสามารถอธิบายถึงสิ่งที่มีมากมายไม่จำกัดโดยใช้คำพูดที่จำกัด

- ภาษามีการเปลี่ยนแปลงอยู่ตลอดเวลา มีศัพท์ใหม่ๆ เกิดขึ้นเสมอ เช่น

I'll fax it to you.

คำว่า fax ดังเดิมเป็นคำนามแต่มีการเปลี่ยนแปลงมาใช้อยู่ในรูปของกริยา

ข้อดี: ภาษามีการวิวัฒนาการตามที่เราต้องการจะสื่อสาร

- มีวิธีการพูดได้หลายแบบสำหรับสิ่งเดียวกัน เช่น

Mary was born on October 11.

Mary's birthday is October 11.

ทั้งสองประโยคนี้สื่อความหมายถึงวันเดียวกัน

ข้อดี: เวลาที่เรารู้สึ้งต่างๆ มากมาย ข้อเท็จจริงหนึ่งจะทำให้รู้สึกข้อเท็จจริงอีกเรื่องหนึ่งได้ ภาษาไม่ไว้ให้ผู้ที่รู้สึ้งต่างๆ มากมายใช้ได้อย่างสะดวก

5.1 ขั้นตอนในการเข้าใจภาษาธรรมชาติ

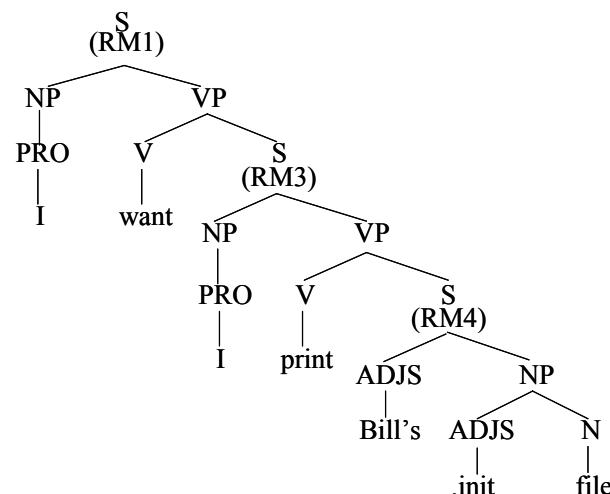
การเข้าใจภาษาธรรมชาติ (*Natural Language Understanding*) มีขั้นตอนดังต่อไปนี้

- การวิเคราะห์ทางองค์ประกอบ (*morphological analysis*) เป็นการวิเคราะห์ในหน่วยคำว่าคำๆ หนึ่งสามารถแยกออกมานะนี่อย่างไร เช่น 'friendly' มาจาก 'friend' กับ 'ly' ก็สามารถนับได้ว่าคำนี้มีหน้าที่อะไร หรือในภาษาไทยเช่น 'การทำงาน' ก็จะแยกเป็น 'การ' กับ 'ทำงาน' เป็นต้น
- การวิเคราะห์ทางภาษาลัมพันธ์ (*syntactic analysis*) เป็นการวิเคราะห์ทางไวยากรณ์ จุดมุ่งหมายก็เพื่อต้องการถูกรับเข้าซึ่งประกอบด้วยคำหลายๆ คำเรียงต่อกันนั้นเป็นโครงสร้างเชิงภาษาลัมพันธ์เป็นอย่างไร คำไหนทำหน้าที่เป็นประธาน กริยา กรรมหรือส่วนใดเป็นวารี เป็นต้น โดยจะวิเคราะห์ลำดับของคำให้อยู่ในโครงสร้างบางอย่าง เช่นต้นไม้เพื่อบอกความสัมพันธ์ของคำต่างๆ
- การวิเคราะห์ทางความหมาย (*semantic analysis*) เป็นการวิเคราะห์ความหมาย เมื่อได้โครงสร้างโดยการวิเคราะห์ทางภาษาลัมพันธ์มาแล้ว ก็จะกำหนดค่าของคำแต่ละคำว่าหมายถึงสิ่งใด
- มุณากการทางวันนีพันธ์ (*discourse integration*) เป็นการพิจารณาความหมายของประโยคโดยดูจากประโยคข้างเคียง เนื่องจากคำบางคำในประโยคนี้ๆ จะเข้าใจความหมายได้ต้องดูประโยคก่อนหน้าหรือประโยคตามด้วย
- การวิเคราะห์ทางปฏิบัติ (*pragmatic analysis*) เป็นการแปลความหมายของประโยคใหม่อีกรั้งว่าที่จริงแล้วผู้พูดตั้งใจหมายความว่าอย่างไรหรือต้องการสื่อความหมายอะไร

ตัวอย่างการประมวลผลภาษาธรรมชาติ

ตัวอย่างนี้เป็นการประยุกต์ใช้การประมวลผลภาษาธรรมชาติกับการแปลงข้อความภาษาธรรมชาติเป็นคำสั่งของระบบยูนิกิร์ [Rich & Knight, 1991] เช่นเมื่อพิมพ์ว่า “I want to print Bill's .init file.” ระบบจะทำการแปลงเป็นคำสั่งของระบบยูนิกิร์แล้วไปสั่งให้ระบบยูนิกิร์ทำงาน ระบบการประมวลผลภาษาธรรมชาติจะทำเป็นขั้นตอนดังต่อไปนี้ โดยสมมติว่าประโยคภาษาธรรมชาติที่ป้อนเข้าไปคือประโยคด้านบน

- เริ่มจากการวิเคราะห์ทางองค์ประกอบโดยวิเคราะห์ในระดับคำ ในกรณีนี้วิเคราะห์ได้ว่า Bill's ประกอบด้วยหน่วยย่อของสองตัวคือ Bill และ 's และผลการวิเคราะห์คือ 's ทำหน้าที่เปลี่ยนคำนาม Bill ให้อยู่ในรูปของคำคุณศัพท์ (adjective) เช่นเดียวกับ .init ในเดิมของคำสั่งในยูนิกิร์นี้ทำหน้าที่เป็นคำคุณศัพท์เช่นกัน
 - จากนั้นจะทำการวิเคราะห์ทางวิเคราะห์ที่ต้องการสัมพันธ์โดยการแปลงประโยคนี้ให้อยู่ในรูปของ **ต้นไม้แจงส่วน (parse tree)** เพื่อแสดงความสัมพันธ์ของคำแต่ละคำในประโยค เช่น คำใดทำหน้าที่ขยายคำอื่นอยู่ ตัวใดเป็นประธานหรือกรรมของกริยาที่สนใจเป็นเด่น ผลที่ได้จากการวิเคราะห์ทางวิเคราะห์ที่ต้องการสัมพันธ์ของประโยคด้านบนแสดงดังรูปที่ 2-8
- ต้นไม้แจงส่วน



รูปที่ 5-1 ผลการวิเคราะห์ทางวิเคราะห์ที่ต้องการสัมพันธ์ของประโยค “I want to print Bill's .init file.”

โครงสร้างต้นไม้ด้านบนทำให้เราทราบความสัมพันธ์ของคำในประโยค ตัวใดทำหน้าที่เป็นประธาน กริยา กรรม หรือตัวขยายต่างๆ ตัวบทสุดในรูปเป็นแทนด้วยสัญลักษณ์ S แสดงถึงประโยค (sentence) ส่วน RM1 เป็นเครื่องหมายอ้างอิงที่จะบอกถึงประโยคนี้และจะถูกนำไปใช้ในการวิเคราะห์ทางความหมายต่อไป ต้นไม้แจงส่วนนี้แสดงโครงสร้างของ

ประโยคเช่นแสดงให้รู้ว่าประโยคแบ่งออกเป็นสองส่วนคือ นามวลี (noun phrase) ซึ่งแทนด้วย NP ในรูปที่ 2-8 และกริยาลี (verb phrase) ซึ่งแทนด้วย VP ในรูปที่ 2-8 นามวลีมีตัวเดียวคือสรรพนาม (pronoun) ส่วนกริยาลีประกอบด้วยกริยา (verb) ซึ่งแทนด้วย V และอนุประโยคย่อย S (RM4) เป็นต้น และเมื่อเราดูจนครบทั้งต้นไม้ก็จะรู้ว่าโครงสร้างทางไวยากรณ์ของประโยคนี้

เมื่อวิเคราะห์ทางไวยากรณ์แล้ว ในส่วนต่อไปเราต้องวิเคราะห์ทางความหมายเพื่อทำความเข้าใจว่าคำแต่ละคำหมายถึงสิ่งใดในโดเมนที่พิจารณาอยู่ อย่างเช่น Bill หมายถึงใคร '.init file' คือวัตถุใดในฐานความรู้เกี่ยวกับโดเมนของระบบยูนิกซ์ที่กำลังพิจารณาอยู่นี้ เป็นต้น ดังนั้นการวิเคราะห์ทางความหมายนี้จะทำหน้าที่ผูกโครงสร้างที่ได้จากการวิเคราะห์ทางไวยากรณ์เข้ากับวัตถุในฐานความรู้ พิจารณาฐานความรู้ในโดเมนนี้ดังแสดงในรูปที่ 5-2 ต่อไปนี้ที่ใช้การแทนความรู้แบบกรอบ

User		F1	
isa:	Person		
*login-name:	must be <string>	instance:	File-Struct
User068		name:	stuff
instance:	User	extension:	.init
login-name:	Susan-Black	owner:	User073
		in-directory:	/wsmith/
User073		File-Struct	
instance:	User	isa:	Information-Object
login-name:	Bill-Smith		
Printing			
isa:	Physical-Event		
*agent:	must be <animate or program>		
*object:	must be <information-object>		
Wanting			
isa:	Mental-Event		
*agent:	must be <animate>		
*object:	must be <state or event>		
Commanding			
isa:	Mental-Event		
*agent:	must be <animate>		
*performer:	must be <animate or program>		
*object:	must be <or event>		
This-System			
instance:	Program		

รูปที่ 5-2 ฐานความรู้บางส่วน

การวิเคราะห์ทางความหมายจะผูกโครงสร้างต้นไม้ใน **รูปที่ 2-8** เข้ากับวัตถุในฐานความรู้ใน **รูปที่ 5-2** ผลการวิเคราะห์ทางความหมายแสดงใน **รูปที่ 5-3**

RM1		{the whole sentence}
instance:	Wanting	
agent:	RM2	{I}
object:	RM3	{a printing event}
RM2		{I}
RM3		{a printing event}
instance:	Printing	
agent:	RM2	{I}
object:	RM4	{Bill's .init file}
RM4		{Bill's .init file}
instance:	File-Struct	
extension:	.init	
owner:	RM5	{Bill}
RM5		{Bill}
instance:	Person	
first-name:	Bill	

รูปที่ 5-3 ผลของการวิเคราะห์ทางความหมายของประโยค “I want to print Bill's .init file.”

หลังจากนั้นทำบูรณาการฐานนิพนธ์ได้ผลดังรูปที่ 5-4

Meaning	
instance:	Commanding
agent:	User068
performer:	This-System
object:	P27
P27	
instance:	Printing
agent:	This-System
object:	F1

รูปที่ 5-4 ผลของการวิเคราะห์ทางปฏิบัติ

ผลลัพธ์สุดท้ายที่ได้จากการวิเคราะห์ทางปฏิบัติคือคำสั่งในยูนิเกิร์ทที่ใช้สั่งยูนิเกิร์พิมพ์ไฟล์ที่ต้องการดังด้านล่างนี้

lpr /wsmith/stuff.init

ในที่นี้เราจะกล่าวถึงขั้นตอนที่สำคัญอย่างหนึ่งในการประมวลผลภาษาธรรมชาติคือการวิเคราะห์ทางภาษาสมพันธ์ดังหัวข้อต่อไปนี้

5.2 การวิเคราะห์ทางภาษาสัมพันธ์

การวิเคราะห์ทางภาษาสัมพันธ์ (syntactic processing) เป็นการวิเคราะห์ประโยคทางไวยากรณ์เพื่อดูโครงสร้างและความสัมพันธ์ของคำในประโยค ช่วยให้การประมวลผลภาษาธรรมชาติในขั้นตอนต่อไปทำได้ง่ายขึ้น เช่น ช่วยให้การประมวลผลทางความหมายทำได้ง่ายขึ้น ซึ่งโดยปกติการประมวลผลทางความหมายมักใช้เวลานาน การประมวลผลทางภาษาสัมพันธ์จะช่วยกรองให้มีตัวเลือกเกิดขึ้นน้อยลง

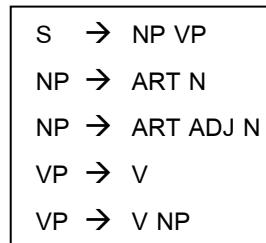
การประมวลผลทางภาษาสัมพันธ์มีส่วนประกอบของกระบวนการ 2 ส่วนหลักๆ ได้แก่

ไวยากรณ์
คลังศัพท์
และ
ตัวแจงส่วน

ไวยากรณ์
ไม่พึงบริบท

- ไวยากรณ์ (grammar) + คลังศัพท์ (lexicon)
- ตัวแจงส่วน (parser)

ไวยากรณ์เป็นตัวกำหนดระเบียบในการประกอบคำให้เป็นประโยค ใช้สำหรับการวิเคราะห์ประโยคว่าประโยคที่รับเข้ามานั้นถูกต้องตามไวยากรณ์หรือไม่ ถ้าถูกต้องมีโครงสร้างของประโยคเป็นอย่างไร เช่น ตัวไหนเป็นประธาน กริยา กรรม ฯลฯ ด้านล่างนี้แสดงด้วยร่างของไวยากรณ์สำหรับประโยคในภาษาอังกฤษแบบง่าย ซึ่งเขียนอยู่ในรูปของ **ไวยากรณ์ไม่พึงบริบท (context-free grammar)**



รูปที่ 5-5 ไวยากรณ์ไม่พึงบริบท 1

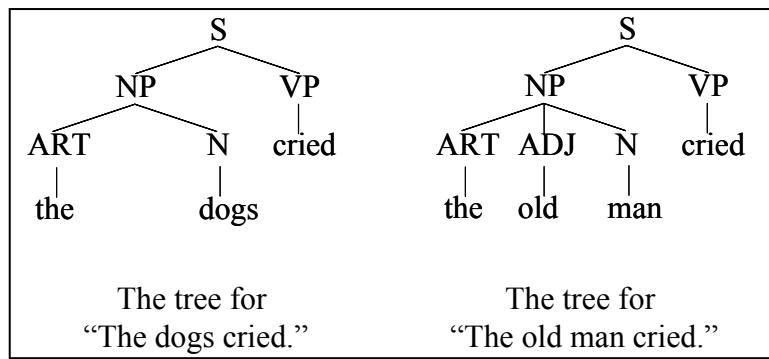
ไวยากรณ์นี้แสดงด้วยกฎทั้งหมด 5 ข้อ กฎแต่ละข้อจะอธิบายลักษณะการเขียนประโยคที่ถูกต้องสำหรับโดยเม้นต์ อย่างเช่นกฎข้อแรกของ S หมายความว่าประโยค (แทนด้วยสัญลักษณ์ S) ประกอบด้วยคำนามวิ (แทนด้วย NP) และต่อด้วยกริยา (แทนด้วย VP) ส่วนกฎของคำนามวิมีอยู่ด้วยกัน 2 ข้อ กฎข้อแรกของคำนามวิหมายถึงคำนามวิประกอบด้วยคำนำหน้า (article) และต่อด้วยคำนาม เป็นต้น หมวดคำบางตัว (เช่น NP ในไวยากรณ์ด้านบน) อาจประกอบด้วยกฎมากกว่าหนึ่งข้อก็ได้

นอกจากไวยากรณ์แล้วเราจำเป็นต้องมีคลังศัพท์ (lexicon) ด้วยเพื่อบอก **หมวดคำ (category)** หรือหน้าที่ของคำที่เป็นไปได้ของคำแต่ละคำ เช่น

cried: V
dogs: N, V
the: ART
old: ADJ, N
man: N, V

ในคลังศัพท์ด้านบนนี้ คำว่า “cried” มีหมวดคำเป็น “V” (กริยา) คำบางคำอาจมีหมวดคำได้มากกว่าหนึ่งอย่าง เช่น “old” เป็นได้ทั้งคำคุณศัพท์ (ADJ) และคำนาม (N) จากคลังศัพท์นี้ทำให้เรารู้ว่าแต่ละคำมีหมวดคำเป็นอะไรได้บ้าง

ตัวแจงส่วนจะทำหน้าที่รับประโภคอินพุต แล้วใช้วิวยากรณ์และคลังศัพท์เพื่อวิเคราะห์โครงสร้างของประโยคที่รับเข้ามา ตัวอย่างเช่นถ้าให้ประโภคอินพุตเป็น “The dogs cried.” หรือ “The old man cried.” จะใช้ผลการวิเคราะห์ในรูปของต้นไม้แจงส่วนเดิมแสดงในรูปที่ 5–6 ต่อไปนี้



The tree for “The dogs cried.”

The tree for “The old man cried.”

5.3 ตัวแจงส่วนแบบลงล่าง

ตัวแจงส่วนแบบบูลงล่าง (top-down parser) เป็นวิธีการหนึ่งสำหรับการวิเคราะห์ทางภาษาอังกฤษ ซึ่งใช้กฎไนไวยากรณ์และคลังศัพท์เพื่อค้นหาวิธีการทุกแบบที่สามารถสร้างต้นไม้แจงส่วน ต้นไม้จะมีคุณสมบัติคือสามารถอธิบายโครงสร้างของประโยคที่เข้ามา

ตัวแจงส่วนแบบบันลงล่างจะสร้างตันไม้จากบันลงล่างโดยเริ่มจาก S จากนั้นก็จะแตก S ออกเรื่อยๆ จนกระทั่งพบรคำที่มาจากประโยคที่ผู้ใช้ป้อนเข้าไป หรือเขียน S เสียใหม่โดยใช้ด้านขวาเมื่อของ S เข้าไปแทนที่ S จากนั้นก็จะแทนที่ด้วยกฎเหล่านี้ไปเรื่อยๆ จนกระทั่งพบร

สัญลักษณ์ปลาย

สัญลักษณ์ปลาย (terminal symbol) ซึ่งหมายถึงสัญลักษณ์ที่เป็นตัวสุดท้ายที่อยู่ที่บัพใบซึ่งตรงกับคำในประโยค

เรามองกระบวนการแจงส่วนว่าคือการค้นหา ในที่นี่เราแทนสถานะในการค้นหาให้อยู่ในรูปของคู่ลำดับที่ประกอบด้วยข้อมูลสองส่วนดังนี้

- รายการสัญลักษณ์: แสดงตัวกระทำการที่ใช้จันถึงปัจจุบัน
- ตัวเลข: แสดงตัวແහນปัจจุบันในประโยคที่จะทำการแจงส่วนต่อไป เช่นกำหนดไว้กรณีให้ดังข้างต้น และให้ประโยค

“₁The ₂ dogs ₃ cried₄”

โดยที่ตัวเลขแสดงตำแหน่งในประโยค สมมติว่าเมื่อเราแจงส่วนไปได้ถึงตำแหน่งหนึ่งแล้วได้สถานะดังนี้

((N VP) 2)

รายการสัญลักษณ์ (N VP) แสดงว่าคำต่อไปคือ N แล้วต่อด้วย VP ที่ตำแหน่งที่ 2 (แสดงด้วยตัวเลขในคู่ลำดับของสถานะด้านบน) สถานะต่อไปที่ได้จากการแจงส่วนกิດจากการตรวจสอบ “N” กับคำในตำแหน่งที่ 2 และเนื่องจาก “dogs” เป็นคำนาม (N) ดังนั้นสถานะต่อไปคือ

((VP) 3)

สัญลักษณ์ไม่ปลาย

ในกรณีนี้ “dogs” เป็นคำที่มีหมวดคำเป็นสัญลักษณ์ปลาย ในบางกรณีที่สัญลักษณ์ตัวแรกในรายการเป็นสัญลักษณ์ไม่ปลาย (non-terminal symbol) เราจะนำกฎที่มีด้านซ้ายมือตรงกับสัญลักษณ์นั้นในไว้กรณีมาสร้างสถานะใหม่โดยแทนที่สัญลักษณ์ตัวแรกนั้นด้วยสัญลักษณ์ด้านขวาเมื่อของกฎนั้น เช่นสถานะ ((VP) 3) ด้านบนนี้ เราจะนำกฎข้อที่ 4 หรือข้อที่ 5 ในรูปที่ 5-5 มาสร้างสถานะใหม่ได้ดังนี้

- ((V) 3) – ในการนี้ที่ใช้กฎข้อที่ 4
- ((V NP) 3) – ในการนี้ที่ใช้กฎข้อที่ 5

อัลกอริทึมจะเก็บรายการสถานะที่เป็นไปได้ (possible state list) ซึ่งประกอบด้วยสองส่วนหลักได้แก่

- สถานะปัจจุบัน: สถานะแรกในรายการสถานะที่เป็นไปได้
 - สถานะสำรอง: สถานะอื่นที่เหลือในรายการสถานะที่เป็นไปได้
- ตัวอย่างของรายการสถานะที่เป็นไปได้ เช่น

((V) 3) ((V NP) 3) ((ART ADJ N VP) 1))

ประกอบด้วยสถานะ 3 ตัว ในกรณีสถานะปัจจุบันคือ ((V) 3) ซึ่งจะถูกเลือกมาทำก่อน ถ้าทำสำเร็จก็หยุด แต่ถ้าไม่สำเร็จจะเลือกสถานะสำรองอีก 2 ตัวที่เหลือมาทำต่อ

อัลกอริทึมของตัวแจงส่วนแบบบูลงล่างอย่างง่าย

อัลกอริทึมของตัวแจงส่วนแบบบูลงล่างอย่างง่ายแสดงใน [ตารางที่ 2-1](#)

ตารางที่ 5-1 อัลกอริทึมของการแจงส่วนแบบบูลงล่างอย่างง่าย

Algorithm: Simple Top-Down Parsing

The algorithm starts with the initial state ((S) 1) and no backup states.

3. Take the first state off the possibilities list and call it C.
IF the list is empty, **THEN** fails.
4. **IF** C consists of an empty symbol list and the word position is at the end of the sentence, **THEN** succeeds.
5. **OTHERWISE**, generate the next possible states.
 - 3.1 **IF** the first symbol of C is a lexical symbol, **AND** the next word in the sentence can be in that class,
THEN
 - create a new state by removing the first symbol
 - updating the word position
 - add it to the possibilities list.
 - 3.2 **OTHERWISE**, **IF** the first symbol of C is a non-terminal
THEN
 - generate a new state for each rule that can rewrite that non-terminal symbol
 - add them all to the possibilities list.

อัลกอริทึมเริ่มจากสถานะเริ่มต้น ((S) 1) และไม่มีสถานะสำรอง จากนั้นดึงสถานะแรกออกจากรายการสถานะที่เป็นไปได้ เรียกสถานะนี้ว่า C ถ้ารายการนั้นว่างแสดงว่าการ

แจงส่วนทำไม่สำเร็จ ถ้า C ประกอบด้วยรายการว่างและตำแหน่งคำ (word position) อญจีในตำแหน่งสุดท้ายของประโยคแสดงว่าการแจงส่วนสำเร็จ ในกรณีอื่นๆ ให้สร้างสถานะใหม่ที่เป็นไปได้โดยทำตามข้อ 3.1 หรือ 3.2 แล้วแต่กรณีดังนี้

- ถ้าสัญลักษณ์ตัวแรกของ C เป็นสัญลักษณ์ในคลังศัพท์ (สัญลักษณ์ปลาย) และคำต่อไปของประโยคที่จะทำการแจงส่วนต่อไปสามารถมีหมวดคำตามสัญลักษณ์นั้นให้สร้างสถานะใหม่โดยตัดคำๆ แรกออกจาก C และปรับตำแหน่งคำใหม่ให้ไปยังตำแหน่งถัดไป แล้วนำสถานะใหม่ที่ได้มาเพิ่มเข้าไปในรายการสถานะที่เป็นไปได้
- ถ้าสัญลักษณ์ตัวแรกของ C เป็นสัญลักษณ์ไม่ปลายให้สร้างสถานะใหม่โดยใช้กฎทุกข้อที่สามารถใช้กระทำกับสัญลักษณ์นั้นได้ แล้วเพิ่มสถานะใหม่ที่ได้ทุกตัวไว้ในรายการสถานะที่เป็นไปได้

จากนั้นให้วนทำไปจนกระทั่งการแจงส่วนเสร็จสิ้น

ตัวอย่างที่ 1

การแจงส่วนของประโยคตัวอย่าง “₁The ₂dogs ₃cried₄” โดยวิธีการแจงส่วนแบบบันลุ่งล่างอย่างง่ายแสดงในตารางที่ 5-2

ตารางที่ 5-2 ตัวอย่างการแจงส่วนของประโยค “₁The ₂dogs ₃cried₄”

ขั้นตอนที่	สถานะปัจจุบัน	สถานะสำรอง	คำอธิบาย
1	((S) 1)		สถานะเริ่มต้น
2	((NP VP) 1)		เขียน S ใหม่ด้วยกฎข้อที่ 1 ของไวยากรณ์ในรูปที่ 5-5
3	((ART N VP) 1)	((ART ADJ N VP) 1)	เขียน NP ใหม่ด้วยกฎข้อที่ 2 และ 3
4	((N VP) 2)	((ART ADJ N VP) 1)	จับคู่ ART กับ the
5	((VP) 3)	((ART ADJ N VP) 1)	จับคู่ N กับ dogs
6	((V) 3)	((V NP) 3) ((ART ADJ N VP) 1)	เขียน VP ใหม่ด้วยกฎข้อที่ 4 และ 5
7	()		การแจงส่วนสำเร็จโดยจับคู่ V กับ cried

ตัวอย่างที่ 2

ตัวอย่างนี้เป็นตัวอย่างที่มีขั้นตอนในการแจงส่วนที่ต้องใช้สถานะสำรอง ประโยคที่จะทำการแจงส่วนคือ “₁The ₂old ₃ man ₄cried₅” ขั้นตอนการแจงส่วนแสดงในตารางที่ 5-3

ตารางที่ 5-3 ตัวอย่างการแจงส่วนของประโยค “₁The ₂old ₃ man ₄cried₅”

ขั้นตอนที่	สถานะปัจจุบัน	สถานะสำรอง	คำอธิบาย
1	((S) 1)		สถานะเริ่มต้น
2	((NP VP) 1)		เขียน S ใหม่ด้วย NP VP
3	((ART N VP) 1)	((ART ADJ N VP) 1)	เขียน NP ใหม่ด้วยกฎข้อที่ 2 และ 3
4	((N VP) 2)	((ART ADJ N VP) 1)	
5	((VP) 3)	((ART ADJ N VP) 1)	
6	((V) 3)	((V NP) 3) ((ART ADJ N VP) 1)	เขียน VP ใหม่ด้วยกฎข้อที่ 4 และ 5
7	(() 4)	((V NP) 3) ((ART ADJ N VP) 1)	
8	((V NP) 3)	((ART ADJ N VP) 1)	เลือกสถานะสำรองแรกมาทำ
9	((NP) 4)	(ART ADJ N VP) 1)	
10	((ART N) 4)	((ART ADJ N) 4) ((ART ADJ N VP) 1)	ไม่พบ ART ในประโยค การแจงส่วนไม่สำเร็จ
11	((ART ADJ N) 4)	((ART ADJ N VP) 1)	การแจงส่วนไม่สำเร็จอีก
12	((ART ADJ N VP) 1)		เลือกสถานะสำรองที่เก็บไว้ที่ขั้นตอนที่ 3 มาทำ
13	((ADJ N VP) 2)		
14	((N VP) 3)		
15	((VP) 4)		
16	((V) 4)	((V NP) 4)	
17	(() 5)		การแจงส่วนสำเร็จ!

5.4 ตัวแจงส่วนตาราง

ตัวแจงส่วนตาราง (*chart parser*) เป็นตัวแจงส่วนที่ทำงานแบบล่างขึ้นบน โดยเริ่มจากคำก่อนแล้วนำคำหล่ายๆ คำรามกันเป็นวงลีหรือหน่วยที่ใหญ่ขึ้น ถ้ามองจากต้นไม้ก็จะเป็นการสร้างต้นไม้จากล่างขึ้นบน วิธีการแจงส่วนจากล่างขึ้นบนจะนำคำมาใส่หมวดคำของมันเข้าไปแล้วบุนรวมจากด้านขวาของกฎไปเป็นด้านซ้ายของกฎ เช่น เมื่อพบ the จะแทนด้วย ART พบ dogs แทนด้วย N จากนั้นก็เขียนแทน ART N ด้วย NP เป็นต้น

ตัวแจงส่วนตารางจะใช้ตารางมาช่วยให้การแจงส่วนทำงานได้อย่างรวดเร็ว มีประสิทธิภาพมากกว่าตัวแจงส่วนแบบบูลดลงล่าง ตัวแจงส่วนนี้ใช้แนวคิดเช่นเดียวกับการโปรแกรมแบบพลวัต (dynamic programming) ตารางนี้จะเก็บผลบางส่วน (partial result) ของการจับคู่ไว้ แล้วใช้กูญแจ (key) เพื่อหากฎที่ขึ้นต้นด้วยกูญแจนั้น หรือหากกฎที่ได้ใช้ไปก่อนหน้านี้และต้องการกูญแจตัวนี้เพื่อขยายกฎหรือทำกฎให้สมบูรณ์

พิจารณาไวยากรณ์ดังต่อไปนี้

S → NP VP
NP → ART ADJ N
NP → ART N
NP → ADJ N
VP → AUX VP
VP → V NP

รูปที่ 5-7 ไวยากรณ์ไม่พึงบริบท 2

สมมติว่ามีประโยคหนึ่งเข้ามาและเริ่มต้นประโยคด้วยคำนำหน้านาม (ART) ตัวแจงส่วนจะใช้ ART เป็นกูญแจและจับคู่กับกฎข้อที่ 2 และ 3 (ดูไวยากรณ์ประกอบ) การจับคู่นี้จะทำเป็นบางส่วนไม่ได้ทำทั้งหมด ซึ่งอาจจะเป็นกฎข้อที่ 2 หรือข้อที่ 3 ก็ได้ ต้องดูคำถัดไปประกอบ แต่ในขณะนี้เรายังไม่เห็นคำถัดไปจึงต้องสร้างการจับคู่เพียงบางส่วนขึ้นมาก่อน ในกรณีนี้คือ

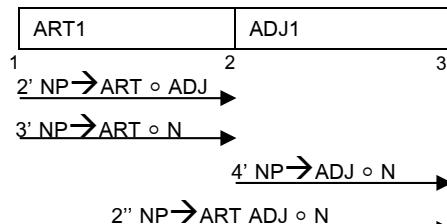
$$2' \text{ NP } \rightarrow \text{ART} \circ \text{ADJ N}$$

$$3' \text{ NP } \rightarrow \text{ART} \circ \text{N}$$

การจับคู่เพียงบางส่วนที่สร้างนี้แสดงให้เห็นว่า เมื่อเราพบ ART อย่างเดียวอาจจะเป็นไปได้ทั้งกฎข้อที่ 2 และ 3 เราจะใช้ \circ เป็นตัวทำเครื่องหมายว่ากฎข้อนี้ได้ถูกใช้ไปถึงจุดไหนแล้วในกรณีของกฎข้อ 2 (2' ด้านบน) แสดงถึงการแจงส่วนมากถึง ART และ แล้ว และถ้าสมมติว่าคำตัดไปมีหมวดคำแสดงด้วยสัญลักษณ์ ADJ เราจะทราบได้ทันทีว่า ADJ จะจับคู่ได้กับกฎข้อที่ 2 และจะสร้างการจับคู่บางส่วนขึ้นมาใหม่เป็น

$$2'' \text{ NP} \rightarrow \text{ART ADJ} \circ \text{ N}$$

เราเรียกการกระทำเช่นนี้ว่าการขยายกฎข้อที่สองออกไปและเราจะนำสัญลักษณ์ที่ได้ไปใส่ไว้ในตาราง สัญลักษณ์นี้เราระบุว่า **องค์ประกอบ (constituent)** ซึ่งเราจะเก็บองค์ประกอบเหล่านี้ไว้ในตารางทั้งหมด แล้วก็สร้างเส้นเชื่อมกัมมันต์ (active arc) ซึ่งก็คือกฎที่สร้างไปแล้วบางส่วนแต่ยังไม่สมบูรณ์ (เช่น 2' 2'' 3' เหล่านี้คือเส้นเชื่อมกัมมันต์ทั้งสิ้น) ดังนั้นตารางจะประกอบด้วยสองส่วนคือเส้นเชื่อมกัมมันต์และองค์ประกอบ รูปที่ 5-8 ด้านล่างนี้แสดงการแจงส่วนเมื่อทำไปจนถึงคำในตำแหน่งที่ 2



รูปที่ 5-8 ตัวอย่างของเส้นเชื่อมกัมมันต์และองค์ประกอบ

ตัวกระทำการของการแจงส่วนตารางนี้จะทำการรวมเส้นเชื่อมกัมมันต์เข้ากับองค์ประกอบที่สมบูรณ์ ผลจากการรวมจะได้ (1) องค์ประกอบที่สมบูรณ์ตัวใหม่ หรือ (2) เส้นเชื่อมกัมมันต์ใหม่ที่เป็นการขยายจากเส้นเชื่อมเดิม และทุกครั้งที่เราได้องค์ประกอบตัวใหม่เราจะนำเส้นเชื่อมกัมมันต์มาขยายจากจุดนั้นไปยังองค์ประกอบที่สมบูรณ์ (สามารถเขียนแทนด้านขวาเมื่อของกฎด้วยองค์ประกอบในด้านซ้ายเมื่อของกฎได้) ถ้าได้องค์ประกอบที่สมบูรณ์ องค์ประกอบนี้จะถูกใส่ไว้ในรายการตัวหนึ่งที่เรียกว่า **อาเจนดา (agenda)** จากนั้นก็นำข้อมูลในอาเจนดาที่ได้ใส่เข้าไปในตาราง อัลกอริทึมของตัวแจงส่วนตารางแสดงในตารางที่ 5-4

ตารางที่ 5-4 อัลกอริทึมของตัวแจงส่วนตาราง

Algorithm: Chart Parsing

UNTIL there is no input left **DO**

- IF** the agenda is empty
THEN look up the interpretations for the next word in the input and add them to the agenda.
- Select a constituent C from the agenda.
Let C is from position p_1 to p_2 .
- FOR EACH** rule in the grammar of form **DO**

$$X \leftarrow C X_1, \dots, X_n$$
add an active arc of the form
$$X \leftarrow C \circ X_1, \dots, X_n$$
from position p_1 to p_2 .
- Add C to the chart using the arc extension algorithm.

อัลกอริทึมจะวนจนกระทั่งไม่มีอินพุตเหลืออยู่ โดยอินพุตจะเป็นคำในประโยคที่ต้องการแจงส่วน อัลกอริทึมจะเริ่มจากตรวจสอบว่าอาเจนดาว่างหรือไม่ ถ้าว่างให้ไปดูหมวดคำสำหรับคำถัดไปในประโยคที่ป้อนเข้ามา นำหมวดคำที่ได้ใส่เข้าไปในอาเจนดา (เช่น man เป็นได้ทั้ง V และ N ก็จะนำทั้ง V และ N ใส่เข้าไปในอาเจนดา ต่อไปเราจะดึงองค์ประกอบนี้เหล่านี้ออกมาระมวลผลทีละตัว) ให้ C เป็นองค์ประกอบที่เลือกออกมาราจากอาเจนดา ซึ่งอาจเป็น N หรือ V เป็นต้น กำหนดให้ C เริ่มจากตำแหน่ง p_1 ไป p_2 จากนั้นให้ไปดูกฎในไวยากรณ์ที่ด้านขวาเมื่อที่ตัวแรกสุดว่าขึ้นต้นด้วย C มีหรือไม่ ($X \leftarrow C X_1, \dots, X_n$) ถ้ามีให้สร้างเส้นเชื่อมกับมันต์ที่แสดงว่าได้ประมวลผล C ไปเรียบร้อยแล้ว (จับคู่บางส่วนกับ C แล้ว) โดยใส่เครื่องหมาย \circ ไว้หลัง C เส้นเชื่อมนี้จะเริ่มจากตำแหน่ง p_1 ไป p_2 ตามคุณสมบัติของ C และนำ C ที่ได้ใส่เข้าไปในตารางโดยใช้อัลกอริทึมขยายเส้นเชื่อม (arc extension algorithm) จากนั้นอัลกอริทึมจะวนกลับมาตรวจสอบว่ามีอินพุตตัวต่อไปหลงเหลือหรือไม่ ถ้าหลงเหลืออยู่ก็จะทำต่อจนหมด

อัลกอริทึมขยายเส้นเชื่อมแสดงในตารางที่ 5-5 ต่อไปนี้

ตารางที่ 5-5 อัลกอริทึมขยายเส้นเชื่อม

Algorithm: Arc Extension

To add a constituent C from position p_1 to p_2 :

1. Insert C into the chart from position p_1 to p_2 .
2. For any active arc of the form

$X \leftarrow X_1 \dots \circ C \dots X_n$
from position p_0 to p_1 , add a new active arc
 $X \leftarrow X_1 \dots C \circ \dots X_n$

from position p_0 to p_2 .

3. For any active arc of the form

$X \leftarrow X_1 \dots X_n \circ C$
from position p_0 to p_1 , add a new constituent of type
 x from p_0 to p_2 to the agenda.

ถ้าต้องการเพิ่มองค์ประกอบ C จากตำแหน่ง p_1 ไป p_2 จะต้องทำตามขั้นตอน 3 ขั้นตอนดังต่อไปนี้คือ

- (1) ใส่ C เข้าไปในตารางจากตำแหน่งที่ p_1 ถึง p_2
- (2) สำหรับเส้นเชื่อมกัมมันต์ใดๆ ที่กำลังรอรับ C อยู่ให้เลื่อนเครื่องหมาย \circ ไปอยู่หลัง C เพื่อแสดงว่าได้ประมวลผล C ไปแล้ว และเมื่อเลื่อน \circ ไปหลัง C จะได้เส้นเชื่อมใหม่ที่ยาวกว่าเดิมหนึ่งหน่วย
- (3) สำหรับเส้นเชื่อมกัมมันต์ใดๆ ที่กำลังรอรับ C เป็นตัวสุดท้าย เช่น $X \leftarrow X_1 \dots X_n \circ C$ เมื่อขยายเส้นเชื่อมนี้จะทำให้ได้องค์ประกอบที่สมบูรณ์ และจะเขียนเส้นเชื่อมนี้ใหม่ด้วยองค์ประกอบที่ด้านซ้ายของกัณฑ์ (X) และนำไปใส่ไว้ในอาเจนดาเพื่อประมวลผลต่อไป

ตัวอย่างของการแจงส่วนตาราง

พิจารณาประโยค “The large can can hold the water” โดยใช้ไวยากรณ์ต่อไปนี้

- 1 S → NP VP
- 2 NP → ART ADJ N
- 3 NP → ART N
- 4 NP → ADJ N
- 5 VP → AUX VP
- 6 VP → V NP

และใช้คลังศัพท์ดังนี้

the: ART

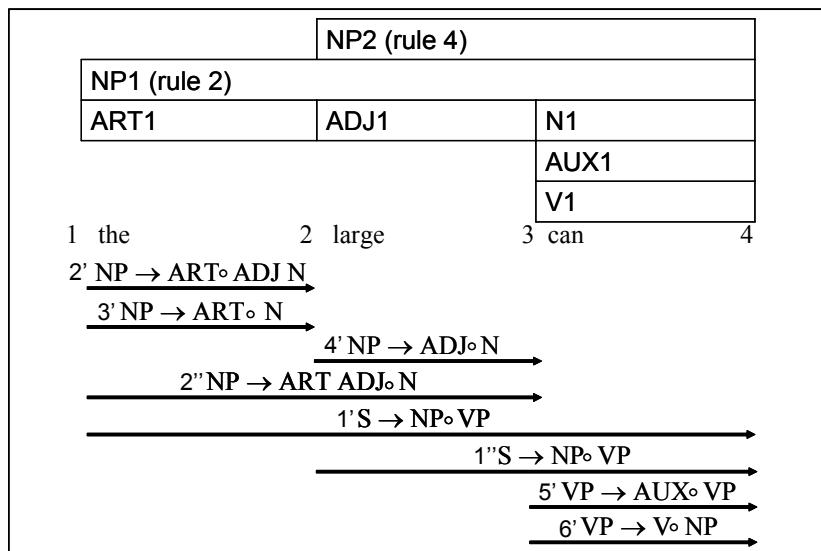
large: ADJ

can: N, AUX, V

hold: N, V

water: N, V

ผลการแจงส่วนเมื่อทำการแจงส่วนจนถึงตำแหน่งที่ 3 แสดงในรูปที่ 5-9



รูปที่ 5-9 การแจงส่วนตารางจนถึง “the large can”

ขั้นตอนแรกอาเจนดายังคงว่างอยู่ (ไม่ได้แสดงอาเจนด้าไว้ในรูป) เรายังคงแยกจากประโยคขึ้นมาแล้วดูหมวดคำ ในที่นี่คือ the เป็น ART จึงนำ ART ใส่ไว้ในอาเจนดา ขั้นตอนที่สองถึง C ออกจากอาเจนดา (ในที่นี่ C คือ ART เพราะตอนนี้มีอยู่ตัวเดียวและอาเจนดาจะว่าง) จากนั้นไปดูกฎที่อยู่ในไวยากรณ์ว่ามีกฎข้อใดที่ด้านขวา มีของกฎขึ้นต้นด้วย ART หรือไม่ พบว่ามีกฎอยู่สองข้อคือกฎที่ 2 และ 3 เรายังสร้างเส้นเชื่อมกัมมันต์โดยทำการประมวลผล ART และได้เส้นเชื่อม 2' และ 3' (ดูรูปที่ 5-9 ประกอบ) สังเกตว่าเส้นเชื่อมที่สร้างขึ้นนี้จะมี 0 เลื่อนไปอยู่หลัง ART

หลังจากนั้นนำ ART ใส่เข้าไปในตารางโดยใช้อัลกอริทึมขยายเส้นเชื่อม วิธีการจะเริ่มจากเส้น ART เข้าไปในตารางจากตำแหน่งที่ 1 ถึง 2 (ART1 ในรูปที่ 5-9) และดูว่ามีเส้นเชื่อมกัมมันต์อื่นหรือไม่ที่กำลังรอรับ ART อยู่ ถ้ามีก็ให้ประมวลผล ART ณ ขณะนี้พบว่ายังไม่มีเส้นเชื่อมกัมมันต์ใดที่รอรับ ART อยู่ ขั้นตอนนี้จึงสิ้นสุดเท่านี้

ณ จุดนี้อาเจนดาว่างอยู่ เราจึงดึงคำตัดไปคือ large เข้ามาและดูว่า large เป็นอะไรได้บ้างจากคลังศัพท์ พบว่าเป็น ADJ จึงนำ ADJ ใส่เข้าไปในอาเจนดา หลังจากนั้นก็ดึง ADJ ออกจากอาเจนดาและดูว่าในไวยากรณ์มี ADJ ปรากฏเป็นตัวแรกสุดทางขวาเมื่อของกฎหรือไม่ พบว่ามีปรากฏในกฎข้อที่ 4 เราจึงนำกฎข้อที่ 4 มาสร้างเป็นเส้นเชื่อมกับมันต่อโดยประมวลผล ADJ (นำ 0 ไปไว้หลัง ADJ) ก็จะได้กฎข้อที่ 4' ดังในรูปที่ 5-9

หลังจากนั้นก็นำ ADJ ใส่เข้าไปในตารางโดยใช้อัลกอริทึมขยายเส้นเชื่อม โดยใส่เข้าไปในตำแหน่งที่ 2 และดูว่ามีเส้นเชื่อมกับมันต่อไปที่กำลังรอรับ ADJ อยู่บ้าง พบว่ามีเส้นเชื่อม 2' เราจึงขยายเส้นเชื่อม 2' ให้ยาวขึ้นโดยเลื่อน 0' ไปอยู่หลัง ADJ ในกฎ 2' เราจะได้กฎ 2" เป็น $NP \rightarrow ART\ ADJ\circ\ N$ ดังรูปที่ 5-9

ต่อมานำ can ไปหาหมวดคำในคลังศัพท์พบว่าเป็นได้ทั้งหมด 3 แบบคือ N, AUX (auxiliary verb (กริยาอนุเคราะห์)) และ V เรานำค่าทั้งสามใส่ไว้ในอาเจนดา ณ จุดนี้เราจะเห็นประโยชน์ของอาเจนดาที่ทำหน้าที่เป็นหน่วยความจำชั่วคราวในการเก็บข้อมูลไว้ประมวลผลทีละตัว เราดึง N ออกจากอาเจนดาแล้วก็ไปดูในไวยากรณ์ว่ามีกฎข้อไหนที่ด้านขวาเมื่อขึ้นต้นด้วย N ก็จะพบว่าไม่มี จึงไม่ต้องสร้างเส้นเชื่อมกับมันต่อสำหรับ N ต่อมาเรานำ N เข้าไปใส่ไว้ในตาราง และใช้อัลกอริทึมขยายเส้นเชื่อมตรวจสอบว่ามีเส้นเชื่อมกับมันต่อไปที่กำลังรอรับ N อยู่หรือไม่ พบว่ากฎข้อ 4' กำลังรอรับ N อยู่พอดี (4' $NP \rightarrow ADJ\circ\ N$) กรณีนี้เราไม่ต้องสร้างเส้นเชื่อมกับมันต่อเส้นใหม่ขึ้นอีก เพราะว่า N เป็นตัวสุดท้ายซึ่งเมื่อประมวลผลแล้วจะทำให้เส้นเชื่อมเป็นองค์ประกอบที่สมบูรณ์และมีตำแหน่งตั้งแต่ 2 จนถึง 4 ทำให้เราได้ NP และนำ NP ใส่เข้าไปในอาเจนดาเพื่อที่จะนำมันมาประมวลผลต่อไปกระบวนการนี้ยังไม่สิ้นสุด เพราะยังมีเส้นเชื่อม 2" ที่ยังรอ N อยู่ เช่นเดียวกัน ดังนั้นเมื่อเส้นเชื่อม 2" รับ N เข้าไปก็จะเกิดองค์ประกอบสมบูรณ์ ซึ่งจะมีบริเวณตั้งแต่ 1 ถึง 4 จากนี้ก็นำ NP ใส่เข้าไปในอาเจนดาเช่นเดียวกัน (เรานำ NP ที่ได้จากเส้นเชื่อม 4' และที่ได้จากเส้นเชื่อม 2" ไปเก็บไว้ในอาเจนดา)

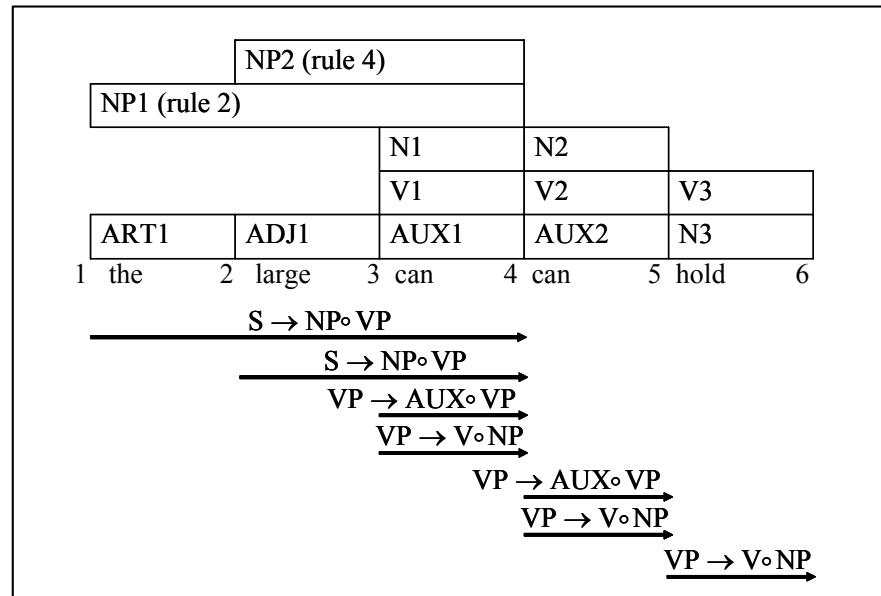
จากนั้นก็ดึง NP2 (ที่ได้จากกฎที่ 2") มาประมวลผลและพบว่ามีความยาวตั้งแต่ตำแหน่งที่ 1 ถึง 4 จากนั้นก็ไปตรวจสอบว่ามีกฎข้อใดบ้างที่ขึ้นต้นด้วย NP ที่ด้านขวาเมื่อพบว่ามีกฎข้อที่ 1 จึงนำมาสร้างเส้นเชื่อมกับมันต่อของกฎข้อที่ 1 มีความยาวตั้งแต่ตำแหน่งที่ 1 ถึง 4 เพราะมาจาก NP2" ที่ยาวตั้งแต่ตำแหน่งที่ 1-4 เราจึงได้เส้นเชื่อมกับมันต่อ 1' $S \rightarrow NP\circ\ VP$ (ประมวลผล NP แล้ว กำลังรอรับ VP) ต่อมา ก็นำ NP ใส่เข้าไปในตารางได้เป็น NP1 (NP ที่มาจากกฎข้อที่ 2) ส่วนอัลกอริทึมขยายเส้นเชื่อมไม่สามารถทำต่อได้อีก จึงสิ้นสุดเท่านี้

ต่อมาดึง NP ตัวต่อมาออกจากอาเจนดา (NP ที่มาจากการข้อที่ 4) และใส่เข้าไปในตารางแล้วดูว่ามีกฎข้อใดที่ด้านขวามีขึ้นต้นด้วย NP หรือไม่ ก็พบกฎข้อที่ 1 เหมือนเดิม “ได้เส้นเชื่อมกัมมันต์ 1” $S \rightarrow NP \circ VP$ ดูว่า ณ ตำแหน่งนี้มีเส้นเชื่อมได้ที่รอรับ NP หรือไม่ พบว่าไม่มี จึงหยุดเท่านี้

ต่อมาดึง AUX ออกจากอาเจนดา ขั้นตอนใหม่มีอนเดิมคือดูในไวยากรณ์ว่าตรงกับตัวแรกที่ด้านขวามีของกฎข้อใดหรือไม่และพบว่าตรงกับกฎข้อที่ 5 ในไวยากรณ์ จึงได้เส้นเชื่อมกัมมันต์ 5’ $VP \rightarrow AUX \circ VP$ และนำ AUX ใส่เข้าไปในตาราง ณ ตำแหน่งนี้ไม่มีเส้นเชื่อมอื่นที่รอรับ AUX จึงสิ้นสุดเท่านี้

ดึง V ออกจากอาเจนดา ไปดูในไวยากรณ์เช่นเดิม พบร่วมกับกฎข้อ 6 ที่ด้านขวามีขึ้นต้นด้วย V จึงได้เส้นเชื่อมกัมมันต์ 6’ $VP \rightarrow V \circ NP$ จากนั้นนำ V ไปใส่ในตารางและไม่มีเส้นเชื่อมใดที่รอรับ V ในตำแหน่งนี้จึงสิ้นสุด

ณ จุดนี้ได้การแจงส่วนจนถึง “The large can” ดังแสดงในรูปที่ 5-9 เมื่อทำต่อด้วยวิธีการแบบเดียวกันนี้จนถึงตำแหน่งที่ 5 ของคำจะได้ผลการแจงส่วนดังรูปที่ 5-10



รูปที่ 5-10 การแจงส่วนตารางจนถึง “the large can can hold”

รูปที่ 5-10 ข้างต้นไม่ได้เขียนเส้นเชื่อมกัมมันต์ทุกตัวโดยเขียนเฉพาะเส้นเชื่อมที่เกี่ยวข้องเท่านั้น ขั้นตอนต่อไปคือการรับคำ can ตัวที่สองเพื่อมาประมวลผลต่อ ซึ่งจะเหมือนกับ can ตัวแรกคือมีหมวดคำเป็นไปได้ 3 แบบคือ N, AUX และ V นำทั้งหมดเก็บไว้

ในอาเจนดาแล้วตึง N (N2 ในรูป) ขึ้นมาทำก่อน นำ N ไปตรวจว่ามีกฎใดที่ทางข้ามีขึ้นต้นด้วย N หรือไม่ พบว่าไม่มี นำ N ใส่เข้าไปในตารางและดูต่อว่ามีเส้นเชื่อมใดรองรับ N หรือไม่ ก็ไม่มีอีก จบการประมวลผลสำหรับ N

นำ AUX (AUX2 ในรูป) มาทำต่อโดยดูว่าตรงกับกฎข้อใด พบว่ามีกฎข้อ 5 ดังนี้สร้าง 5" VP → AUX o VP จากนั้นนำ AUX ใส่เข้าไปในตารางและดูว่ามีเส้นเชื่อมได้รองรับ AUX หรือไม่ ปรากฏว่าไม่มี สิ้นสุด

ตัวต่อไปคือ V ดึง V ออกจากอาเจนดาใส่เข้าไปในตารางแล้วไปหากฎ พบกฎข้อ 6 จึงสร้าง 6" VP → V o NP ดูว่ามีเส้นเชื่อมได้รองรับ V อยู่หรือไม่ พบว่าไม่มี สิ้นสุด

การประมวลผลตัวถัดไปคือ hold ก็ทำเช่นเดิม ดูในคลังศัพท์พบว่า hold เป็นได้ทั้ง N และ V นำทั้งคู่ไปใส่ไว้ในอาเจนดาแล้วตึง N ออกมาทำก่อน นำ N ไปใส่ไว้ในตารางแล้วเทียบกับกฎ พบว่าไม่มีกฎข้อใดที่ด้านขวาขึ้นต้นด้วย N จากนั้นทำการขยายเส้นเชื่อมและไม่พบเส้นเชื่อมกัมมันต์ใดที่รองรับ N สิ้นสุด

ต่อไปดึง V ออกมา นำ V ใส่ในตารางแล้วไปหากรณีพบว่าตรงกับกฎข้อ 6 จึงสร้าง 6" VP → V o NP และดูว่าตำแหน่งนี้มีเส้นเชื่อมได้รองรับ V หรือไม่ ไม่มี สิ้นสุด

ถึงจุดนี้ได้การแจงส่วนดังรูปที่ 5-10 เมื่อทำต่อจนครบทุกคำด้วยวิธีการเช่นเดิมจะได้ดังรูปที่ 5-11 ซึ่งแสดงให้เห็นโครงสร้างทางไวยากรณ์ของประโยค (S1) ที่ประกอบด้วยนามวลี (NP1) และกริยาวลี (VP2) ส่วนนามวลี (NP1) ประกอบด้วยคำนำหน้านาม คำคุณศัพท์ และคำนาม (ART1 ADJ1 N1) ส่วนกริยาวลี (VP2) ประกอบด้วยกริยานุเคราะห์ กริยา คำนำหน้านามและคำนาม (AUX2 V3 ART2 N4)

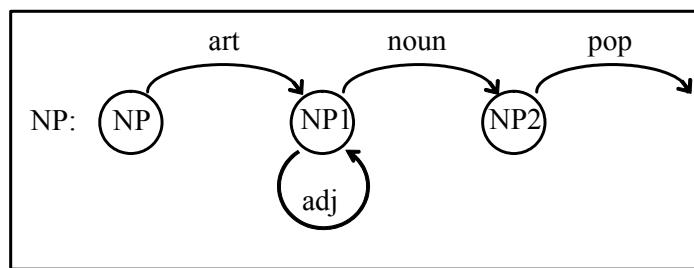
S1 (rule 1 with NP1 and VP2)						
S2 (rule 1 with NP2 and VP2)						
VP3 (rule 5 with AUX1 and VP2)						
NP1 (rule 2)		VP2 (rule 5)				
ART1	N1	N2			NP3 (rule 3)	
	V1	V2	V3		V4	
ADJ1	AUX1	AUX2	N3	ART2	N4	

รูปที่ 5-11 ผลการแจงส่วนตารางจนครบประโยค “the large can can hold the water”

5.5 ไวยากรณ์ข่ายงานเปลี่ยนสถานะ

หัวข้อนี้กล่าวถึงไวยากรณ์อิกรูปแบบหนึ่งที่เรียกว่า **ไวยากรณ์ข่ายงานเปลี่ยนสถานะ (Transition Network Grammar)** ซึ่งเป็นไวยากรณ์ในรูปของข่ายงานประกอบด้วยเส้นเชื่อม (arc) และบัพ (node)

บัพแต่ละบัพแสดงสถานะของการแจงส่วนของประโยคและเส้นเชื่อมแต่ละเส้นแสดงองค์ประกอบหรือคำต่างๆ ข่ายงานนี้จะมีบัพอยู่หนึ่งบัพที่เป็นบัพเริ่มต้น จากนั้นจะรับอินพุตเข้ามาและท่องไปตามเส้นเชื่อม โดยเริ่มจากบัพเริ่มต้นและท่องไปที่ปลายทางของเส้นเชื่อมสำหรับคำนั้นไปยังบัพถัดไป ณ ขณะหนึ่งถ้ามาถึงสถานะสุดท้ายที่มีเส้นเชื่อม “pop” ก็แสดงว่าการแจงส่วนสมบูรณ์ [รูปที่ 5-12](#) แสดงตัวอย่างของไวยากรณ์ชนิดนี้และให้ชื่อว่า “ข่ายงานเปลี่ยนสถานะ 1”



รูปที่ 5-12 ข่ายงานเปลี่ยนสถานะ 1

ข่ายงานเปลี่ยนสถานะ 1 ข้างต้นนี้เป็นข่ายงานของนามวลี (NP) หรือว่า NP คืออะไร จากจุดเริ่มต้นที่สถานะ NP ถ้าพบ art ก็จะท่องไปที่สถานะ NP1 และถ้าพบ adj ก็ท่องมาอยู่ที่สถานะเดิม แต่ถ้าพบ noun ก็จะไปยังสถานะ NP2 และเมื่อเส้นเชื่อม pop ก็จะไปที่ไวยากรณ์ข่ายงานสถานะ 1 นี้สมมูลกับไวยากรณ์ไม่พึงบวบกทต่อไปนี้

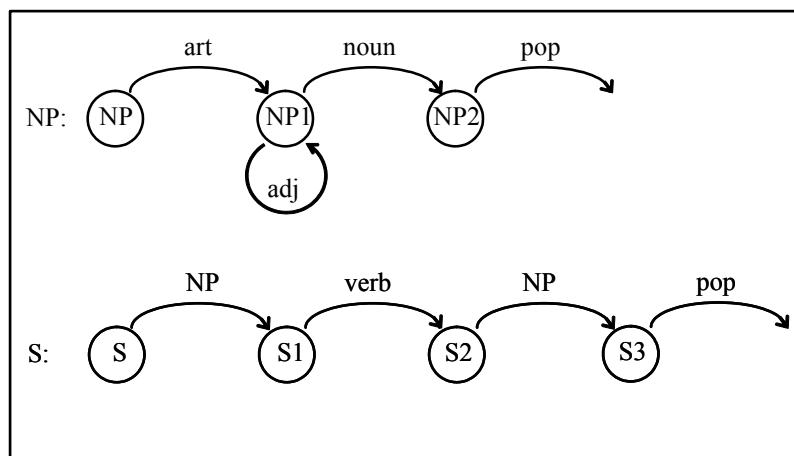
NP → ART NP1

NP1 → ADJ NP1

NP1 → N

เราสามารถนำข่ายงานเปลี่ยนสถานะ 1 ข้างต้นไปแจงส่วนสำหรับนามวลีได้ เช่น “a purple cow” โดยเริ่มจากสถานะเริ่มต้น NP นำ “a” ไปตรวจสอบในคลังศัพท์พบว่า “a” เป็น art ก็จะท่องไปที่สถานะ NP1 จากนั้นรับคำตัดไปคือ “purple” ซึ่งเป็น adj ก็ท่องกลับมาที่สถานะเดิม และรับคำต่อไปคือ “cow” ซึ่งเป็น noun ก็เปลี่ยนไปที่สถานะ NP2 และพบเส้นเชื่อม pop ก็แสดงว่าการแจงส่วนสิ้นสุด

อย่างไรก็ดีไวยากรณ์ข่ายงานเปลี่ยนสถานะนี้มีข้อจำกัดเมื่อนำไปแจงส่วนประโยคที่ซับซ้อนขึ้น จึงได้มีการขยายข่ายงานนี้ให้สามารถใช้งานได้กว้างขวางขึ้นเป็นไวยากรณ์ใหม่ที่เรียกว่า **ข่ายงานเปลี่ยนสถานะเรียกซ้ำ – อาร์ทีอี็น (Recursive Transition Network – RTN)** อาร์ทีอี็นนี้สามารถอ้างถึงข่ายงานอื่นได้ เส้นเชื่อมในอาร์ทีอี็นมีสองชนิดคือ (1) เส้นเชื่อมแบบเดิมที่เป็นหมวดคำ เช่น article, noun เป็นต้น และ (2) เส้นเชื่อมอ้างถึงข่ายงานอื่น และเพื่อแยกความแตกต่างระหว่างเส้นเชื่อมทั้งสองชนิดให้เห็นอย่างชัดเจน จึงใช้อักษรตัวเล็กแทนเส้นเชื่อมแบบเดิมและใช้ตัวอักษรใหญ่เขียนเส้นเชื่อมที่อ้างถึงข่ายงานอื่น ดัวอย่างของอาร์ทีอี็นแสดงในรูปที่ 5-13



รูปที่ 5-13 ข่ายงานเปลี่ยนสถานะ 2 (อาร์ทีอี็น)

ข่ายงานเปลี่ยนสถานะ 2 ในรูปด้านบนนี้ประกอบด้วยข่ายงานย่อย 2 ข่ายงานคือ ข่ายงานส่วนบนที่เป็นข่ายงานของ NP หรือ กับตัวอย่างที่แล้ว ส่วนข่ายงานล่างเป็นของ S ซึ่งจะมีเส้นเชื่อมบางเส้นที่อ้างข่ายงาน NP (สังเกตจากตัวอักษรใหญ่) ไวยากรณ์นี้มีความหมายดังนี้คือ เริ่มต้นจากสถานะ S เรายจะท่องไปยัง S1 ได้นั้น คำที่รับเข้ามายังต้องประกอบด้วย NP ก่อน ซึ่ง NP คืออะไรนั้นก็ต้องท่องไปที่ข่ายงาน NP ให้เรียบร้อยเสียก่อน เราสามารถใช้อาร์ทีอี็นนี้แจงส่วนของประโยคที่ซับซ้อนขึ้นได้ เช่นประโยค “The purple cow ate the grass” เป็นต้น

อัลกอริทึมแจงส่วนบนลงล่างสำหรับอาร์ทีอี็น

อัลกอริทึมที่จะกล่าวต่อไปนี้เป็นอัลกอริทึมสำหรับแจงส่วนแบบบนลงล่างโดยใช้ไวยากรณ์แบบอาร์ทีอี็น อัลกอริทึมนี้จะเก็บสถานะของการแจงส่วนที่ประกอบด้วยข้อมูลต่อไปนี้

- บัพปั้จจุบัน (current node): บัพที่การแจงส่วนอยู่ ณ ปัจจุบัน

- ตำแหน่งปัจจุบัน (current position): ตำแหน่งของคำถัดไปที่จะประมวลผล
 - จุดกลับ (return points): กองซ้อน (stack) ของสถานะที่จะย้อนกลับมาเมื่อมีการอ้างและท่องไปยังข่ายงานอื่นๆ และพับเส้นเชื่อม pop

อัลกอริทึมแสดงในตารางที่ 5-6

ตารางที่ 5-6 อัลกอริทึมแจงส่วนแบบบันลั่งสำหรับอาร์ทีอีเอ็น

Algorithm: RTN Parsing

At each node, you can leave the current node and traverse an arc in the following cases:

Case 1: **IF** arc is word category and next word in the sentence is in that category

THEN

- (1) update current position to start at the next word
 - (2) update current node to the destination of the arc.

Case 2: **IF** arc is a push arc to a network N

THEN

- (1) add the destination of the arc onto
return points
 - (2) update current node to the starting node
in the network N .

Case 3: IF arc is a pop arc and return points list is not empty

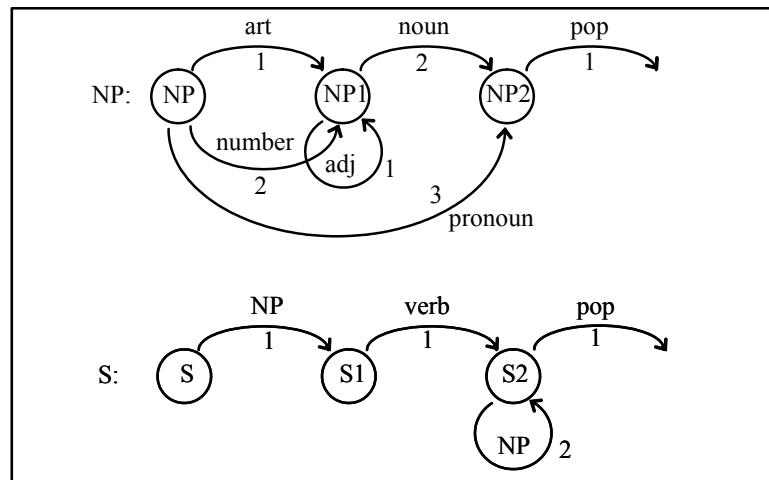
THEN remove first return point and make it current node

Case 4: **IF** arc is a pop arc, return points list is empty and there are no words left

and there are no words left
THEN parse completes successfully

อัลกอริทึมเริ่มจากสถานะเริ่มต้น เมื่อออยู่ที่สถานะปัจจุบันหนึ่งๆ อัลกอริทึมจะอ่านจากสถานะนั้นและท่องเส้นเชื่อมตามกรณิตต่างๆ ดังนี้ (1) กรณิตที่เส้นเชื่อมเป็นหมวดคำและคำถัดไปอยู่ในหมวดคำนั้น ก็ทำการปรับค่าของตำแหน่งปัจจุบันและบัพปัจจุบัน (2) กรณิตที่เส้นเชื่อมอ้างถึงข่ายงานอื่น ก็ให้ไปเริ่มที่สถานะเริ่มต้นในข่ายงานนั้น (3) กรณิตที่เส้นเชื่อมเป็นเส้นเชื่อม pop และรายการจุดกลับไม่ว่าง ให้ไปยังสถานะที่กำหนดโดยจุดกลับตัวแรกในรายการ และ (4) กรณิตที่เส้นเชื่อมเป็นเส้นเชื่อม pop และรายการจุดกลับว่างและไม่มีคำเหลือในประโยค ก็แสดงว่าการแจงส่วนสมบูรณ์

ตัวอย่างการแจงส่วนด้วยอาร์ทีอีน

พิจารณาข่ายงานเปลี่ยนสถานะ 3 ใน [รูปที่ 5-14](#) ด้านล่างนี้

รูปที่ 5-14 ข่ายงานเปลี่ยนสถานะ 3 (อาร์ทีอีน)

ผลการแจงส่วนของประโยค “₁The ₂old ₃man ₄cried₅” แสดงใน [ตารางที่ 5-7](#)ตารางที่ 5-7 ผลการแจงส่วนของประโยค “₁The ₂old ₃man ₄cried₅”

ขั้นตอน ที่	บัพ ปัจจุบัน	ตำแหน่ง ปัจจุบัน	จุด กลับ	เส้นเชื่อม ที่ใช้	คำอธิบาย
1	(S,	1,	NIL)	S/1	สถานะเริ่มต้น
2	(NP,	1,	(S1))	NP/1	ท่องตามเส้นเชื่อมที่อ้างถึง ข่ายงาน NP และมีจุดกลับที่ S1
3	(NP1,	2,	(S1))	NP1/1	ท่องเส้นเชื่อม NP1/1(the)
4	(NP1,	3,	(S1))	NP1/2	ท่องเส้นเชื่อม NP1/1(old)
5	(NP2,	4,	(S1))	NP2/2	ท่องเส้นเชื่อม NP1/2(man) เนื่องจาก NP1/1 ใช้ไม่ได้
6	(S1,	4,	NIL)	S1/1	เส้นเชื่อม pop ทำให้กลับมาที่ S1
7	(S2,	5,	NIL)	S2/1	ท่องเส้นเชื่อม S1/1(cried)
8	แจงส่วน สมบูรณ์				การแจงส่วนเสร็จสมบูรณ์ด้วย เส้นเชื่อม pop จาก S2

บัพแรกในข่ายงาน NP มีเส้นเชื่อม 3 เส้น ถ้าเป็น art จะท่องไปยัง NP1 ถ้าเป็น number จะไปที่ NP1 และถ้าเป็น pronoun จะไปที่ NP2 คำอธิบายในตารางที่ 5-7 จะใช้ตัวเลขกำกับไว้ที่ด้านหลังของสถานะเพื่อแสดงการแจงส่วนว่าใช้เส้นเชื่อมใด ตัวอย่างเช่น NP/1 หมายถึงการใช้เส้นเชื่อมที่ 1 ของสถานะ NP

กำหนดให้คลังศัพท์เป็นดังต่อไปนี้

the: ART

old: ADJ, N

man: N

cried: V

การแจงส่วนเริ่มจากบัพแรกคือ S ตำแหน่งบัพจุบันคือตำแหน่งที่ 1 และจุดกลับไม่มี เราท่องตามเส้นเชื่อมโดยเริ่มจาก NP ก่อน สังเกตว่า NP เป็น push arc หมายความว่าให้ “ไปยังตำแหน่งแรกในข่ายงาน NP โดยให้จำไว้ว่าจุดกลับคือปลายเส้นเชื่อมที่ 1 ของ NP ซึ่ง ก็คือบัพ S1 ขณะนี้เรออยู่ในข่ายงาน NP แล้วดูต่อว่า the เป็น art ได้หรือไม่ พบร่วมได้ ก็ ท่องไปที่ NP1 และปรับค่าของตำแหน่งคำจาก 1 เป็น 2 บัพบัพจุบันอยู่ที่ NP1 และจุดกลับอยู่ที่ S1 เมื่อันเดิม เตรียมรับคำต่อไป จากนั้นดูว่าที่ตำแหน่งบัพจุบันคือ NP1 สามารถรับ adj ได้หรือไม่ พบร่วมได้ ก็ประมวลผล old และกลับมายัง NP1 เมื่อันเดิม (ตามรูป) พร้อมทั้งเลื่อนตำแหน่งบัพจุบันไปยังตำแหน่งที่ 3 จุดกลับยังเป็น S1 เช่นเดิม

คำต่อไปคือ man ซึ่งเป็น noun และที่บัพ NP1 สามารถรับ noun ได้ เราจึงท่องไปที่ NP2 เปลี่ยนตำแหน่งบัพจุบันไปยังตำแหน่งที่ 4 และจุดกลับคือ S1 เมื่อันเดิม ขณะนี้เราอยู่ที่ NP2 ซึ่งมี pop arc ก็ท่องไปที่ pop arc และออกจากข่ายงาน NP กลับมายังจุดกลับ S1 ได้ตามขั้นตอนที่ 6 ในตารางและรับ V จากนั้นก็ดูว่าคำต่อไปคือ cried สามารถเป็น V “ได้หรือไม่ พบร่วม cried เป็น V ได้ก็ท่องไปที่ S2 ซึ่งเป็น pop arc และตำแหน่งบัพจุบันเป็น ตำแหน่งสุดท้าย ได้ว่าแจงส่วนสำเร็จ

พิจารณาตัวอย่างอีกตัวอย่างหนึ่งของการแจงส่วนของประโยค “One saw the man.” ดังแสดงในตารางที่ 5-8 ต่อไปนี้ โดยกำหนดให้คลังศัพท์เป็นดังต่อไปนี้

one: number, pronoun

saw: V, N

the: art

man: N

ตารางที่ 5-8 ผลการแจงส่วนของประโยค “₁One ₂saw ₃the ₄man ₅”

ขั้นตอน ที่	สถานะปัจจุบัน	เส้นเชื่อมที่ใช้	สถานะสำรอง
1	(S,1,NIL)	S/1	NIL
2	(NP,1,(S1))	NP/2 (เก็บ NP/3 ไว้สำรอง)	NIL
3	(NP1, 2,(S1))	NP1/2	(NP2,2,(S1))
4	(NP2,3,(S1))	NP2/1	(NP2,2,(S1))
5	(S1,3,NIL)	ไม่มีเส้นเชื่อมที่ไปได้	(NP2,2,(S1))
6	(NP2,2,(S1))	NP2/1	NIL
7	(S1,2,NIL)	S1/1	NIL
8	(S2,3,NIL)	S2/2	NIL
9	(NP,3,(S2))	NP/1	NIL
10	(NP1,4,(S2))	NP1/2	NIL
11	(NP2,5,(S2))	NP2/1	NIL
12	(S2,5,NIL)	S2/1	NIL
13	แจงส่วนสมบูรณ์		

การแจงส่วนของตัวอย่างที่แล้วไม่มีการทำข้อเสนออย่างสามารถแจงส่วนรวมเดียวจบ ตัวอย่างนี้จะแสดงให้เห็นถึงการข้อนรอย

เริ่มต้นจาก S เมื่อตอนเดิม และห้องโดยใช้ push arc ไปยังข่ายงาน NP เริ่มจากบัพ NP เพื่อรับคำต่อไป คำที่รับเข้าคือ one เป็นได้ทั้ง number และ pronoun ถ้าเลือก number ก็จะห้องไปตามเส้นเชื่อมที่ 2 ถ้าเลือก pronoun ก็จะห้องตามเส้นเชื่อมที่ 3 ครั้งแรกจะเลือก number ก่อนและห้องไปที่ NP1 และปรับค่าตำแหน่งปัจจุบันเป็นตำแหน่งที่ 2 แต่ถ้าเลือก pronoun จะห้องไปยัง NP2 (สถานะสำรองในตาราง) สถานะสำรองจะถูกเรียกมาทำก็ต่อเมื่อสถานะปัจจุบันไม่สามารถทำให้การแจงส่วนสมบูรณ์

ที่บัพ NP1 สามารถรับได้ทั้ง adj กับ noun เรากับ saw เป็นได้ทั้ง verb และ noun จึงห้องไปที่ NP2 และตำแหน่งปัจจุบันเปลี่ยนเป็นตำแหน่งที่ 3 ที่ NP2 เรากับ pop arc จึงกลับมาที่จุดกลับคือ S1 ณ จุดนี้คำในตำแหน่งที่ 3 ต้องเป็น verb และเรากับคำในตำแหน่งที่ 3 คือ the ซึ่งเป็น art ไม่ใช่ verb ทำให้การแจงส่วนไม่สำเร็จ เราจึงต้องดึงสถานะสำรองมาเป็นสถานะปัจจุบันเพื่อทำการแจงส่วนต่อไป

ที่ NP2 ซึ่งไป pop arc กลับมาที่ S1 และตำแหน่งปัจจุบันเป็นตำแหน่งที่ 2 คำที่จะรับเข้ามาคือ saw ซึ่งเป็น verb ได้ เราจึงห้องต่อไปยัง S2 รอบคำที่ 3 ซึ่งยังไม่จบประโยค ดังนั้นเราห้องตามเส้นเชื่อมที่ 2 เพื่อกลับไปยังข่ายงาน NP คำที่ 3 คือ the ดังนั้นไปที่ NP1 รอบคำในตำแหน่งที่ 4 ซึ่งเป็น noun ห้องไปยัง NP2 พน pop arc กลับมาที่จุดกลับ S2 ซึ่งจบประโยคพอดี จึงห้องไปยัง pop arc สุดท้าย แสดงว่าการแจงส่วนสำเร็จ

เอกสารอ่านเพิ่มเติมและแบบฝึกหัด

หนังสือที่น่าศึกษาสำหรับการประมวลภาษาธรรมชาติ คือ [Allen, 1995] ซึ่งอธิบาย 'ไวยากรณ์ชนิดต่างๆ การแจงส่วนวิธีการต่างๆ รวมถึงการวิเคราะห์ทางความหมาย ตลอดจนวิธีการอื่นๆ ในการประมวลผลภาษาธรรมชาติ' วิทยากร Manning และ Schütze เขียนหนังสือ [Manning & Schütze, 1999] ที่เกี่ยวกับการประมวลผลภาษาธรรมชาติโดยเทคนิคทางสถิติ ซึ่งอธิบายวิธีการใช้ประโยชน์จากคลังข้อความ (corpus) มาช่วยให้การประมวลผลภาษาธรรมชาติมีประสิทธิภาพ หนังสือของ Jurafsky และ Martin [Jurafsky & Martin, 2000] อธิบายถึงเทคนิคในการประมวลผลภาษาธรรมชาติ และการประมวลผลทางเสียงด้วย

บรรณานุกรม

- Allen, J. (1995) *Natural Language Understanding*. Second Edition. Pearson Addison Wesley.
- Jurafsky, D. and Martin, J. (2000) *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall.
- Manning, C. and Schütze, H. (1999) *Foundations of Statistical Natural Language Processing*. The MIT Press.
- Rich, E. and Knight, K. (1991) *Artificial Intelligence*. (International Edition), McGraw-Hill.

แบบฝึกหัด

- กำหนดไวยากรณ์และคลังศัพท์ให้ดังต่อไปนี้
ไวยากรณ์

$S \rightarrow NP\ VP$

$NP \rightarrow ART\ ADJ\ N$

$NP \rightarrow ART\ N$

NP → ADJ N

VP → AUX VP

VP → V NP

VP → V

คลังศัพท์

the : ART

old : ADJ, N

man : N, V

can : N, AUX, V

do : V, AUX

จะแสดงขั้นตอนการแจงส่วนโดยใช้ตัวแจงส่วนตารางสำหรับประโยชน์ “The old man can do.”

2. จงเขียนอาร์ทีอีนที่สมมูลยกับไวยากรณ์ในข้อ 1.
 3. จงใช้อาร์ทีอีนที่ได้จากข้อ 2. และทำการแจงส่วนของประโยชน์ “The old man can do.”
ด้วยอัลกอริทึมแจงส่วนแบบบูลจังสำหรับอาร์ทีอีน
-
-

การเรียนรู้ของเครื่อง

6

บทนี้กล่าวถึง **การเรียนรู้ของเครื่อง** (*machine learning*) ซึ่งเทคนิคการเรียนรู้ส่วนมากเป็น **การเรียนรู้เชิงอุปนัย** (*inductive learning*) และมีบางเทคนิคเป็น **การเรียนรู้เชิงวิเคราะห์** (*analytical learning*) การเรียนรู้เชิงอุปนัยคือการเรียนรู้ที่หากฎเกณฑ์หรือความรู้ที่ແง່ງอยู่ในชุดตัวอย่างสอน (*training example set*) เพื่อเรียนรู้ให้ได้ความรู้ใหม่ที่สอดคล้องกับชุดตัวอย่างสอน ส่วนการเรียนรู้เชิงวิเคราะห์เป็นการจัดรูปแบบของความรู้ใหม่เพื่อให้ใช้งานได้อย่างมีประสิทธิภาพมากขึ้น ทำงานได้เร็วขึ้น

6.1 ขั้นตอนวิธีเชิงพันธุกรรม

ขั้นตอนวิธีเชิงพันธุกรรม – จีเอ (*Genetic Algorithm – GA*) [Goldberg, 1989; Mitchell, 1996] เป็นการเรียนรู้ที่จำลองการวิวัฒนาการ เราอาจมองได้ว่าจีเอเป็นกระบวนการค้นหาประชาบทหนึ่งหรืออาจมองว่าจีเอเป็นการเรียนรู้ของเครื่องประชาบทหนึ่งก็ได้ จีเอได้ถูกขยายขึ้นเป็น **การโปรแกรมเชิงพันธุกรรม – จีพี** (*Genetic Programming – GP*) [Koza, 1992] ข้อแตกต่างที่สำคัญอย่างหนึ่งระหว่างจีเอกับจีพีคือในจีเอสิ่งที่เรียนรู้ได้เป็นสายอักขระความยาวคงที่ (*fixed-length string*) ส่วนในจีพีจะได้สายอักขระความยาวแปรไป (*variable-length string*) ซึ่งมักแสดงในรูปของโปรแกรมภาษา LISP

แนวคิดของจีเอมาจากการของสิ่งมีชีวิต เช่นการไขว้เปลี่ยนของโครโมโซม (*chromosome crossover*) การกลายพันธุ์ของยีน (*gene mutation*) การวิวัฒนาการของสิ่งมีชีวิต เป็นต้น จีอสามารถจัดการกับปัญหาค่าดีสุดเฉพาะที่ (*local optimum*) ในการค้นหาได้ การค้นหาทั่วไปจะมองว่าจุดดีสุดเฉพาะที่เป็นกับดักและจะหลีกเลี่ยงกับดักโดยใช้วิธีต่างๆ เช่น การย้อนรอย (*backtracking*) หรือการค้นหาแบบขนาน (*parallel search*) โดยใช้สถานะเริ่มต้นที่ต่างๆ กัน เป็นต้น แต่เทคนิคการค้นหาด้วยจีเอจะใช้วิธีการที่แตกต่างไปดังจะกล่าวต่อไป

การไขว้เปลี่ยน
การกลายพันธุ์

โครโนซมกำหนดลักษณะพิเศษที่สืบทอดได้

เซลล์แต่ละเซลล์ในพืชชั้นสูงและสัตว์ประกอบด้วยนิวเคลียส 1 นิวเคลียส และนิวเคลียสหนึ่งๆ ประกอบด้วยโครโนซมจำนวนมาก โครโนซมจะอยู่กันเป็นคู่โดยได้รับมาจากพ่อและแม่อย่างละ 1 เส้น โครโนซมแต่ละเส้นจะมี Yin เป็นตัวกำหนดลักษณะพิเศษของสิ่งมีชีวิต ในขณะที่มีการจับคู่กันของโครโนซมอาจเกิด **การไขว้เปลี่ยน (crossover)** ซึ่งเป็นการที่ยึนจากโครโนซมพ่อแม่สลับเปลี่ยนกันทำให้เกิดโครโนซมใหม่ขึ้น 2 คู่ และในขณะที่เซลล์แบ่งตัวจะเกิดกระบวนการ **คัดลอกโครโนซม (chromosome copying)** ซึ่งบางครั้งจะมีการเปลี่ยนแปลงของยีนที่มาจากการยึนพ่อและแม่เกิดเป็นยีนที่ไม่เคยมีมาก่อน เราเรียกการเกิดยึนลักษณะนี้ว่า **การกลายพันธุ์ (mutation)**

ชา尔斯 ดาร์วิน (Charles Darwin) ได้อธิบายการสืบทอดของสิ่งมีชีวิตด้วยกฎที่เรียกว่า **การวิวัฒนาการโดยผ่านการคัดเลือกตามธรรมชาติ (evolution through natural selection)** ไว้ว่าสิ่งมีชีวิตมีแนวโน้มที่จะสืบทอดลักษณะพิเศษให้ลูกหลานและธรรมชาติจะผลิตสิ่งมีชีวิตที่มีลักษณะพิเศษแตกต่างไปจากเดิม สิ่งมีชีวิตที่เหมาะสมที่สุด (fittest) ก็คือสิ่งมีชีวิตที่มีลักษณะพิเศษที่ธรรมชาติพ่อใจมากที่สุดจะมีแนวโน้มที่มีลูกหลานมากกว่าตัวที่ไม่เหมาะสม ดังนั้นประชากรจะโน้มเอียงไปทางตัวที่เหมาะสม เมื่อช่วงเวลาผ่านไปนานๆ การเปลี่ยนแปลงจะสะสมไปเรื่อยๆ และเกิดสปีชีส์ (species) ใหม่ที่เหมาะสมกับสภาพแวดล้อม ดังนั้นเรารายจากล่าสุดได้ว่าการคัดเลือกโดยธรรมชาติเกิดจากการเปลี่ยนแปลงที่เป็นผลของการไขว้เปลี่ยนและการกลายพันธุ์

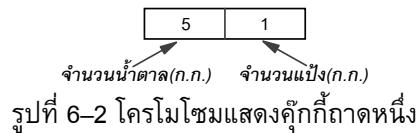
6.1.1 การออกแบบชั้นตอนให้มีเชิงพันธุกรรม

จะยกตัวอย่างปัญหาการทำคุ๊กกี้เพื่ออธิบายการออกแบบจีเอ [Winston, 1992] สมมติว่าเราต้องการหาส่วนผสมที่ดีที่สุดเพื่อทำคุ๊กกี้โดยที่คุ๊กกี้มีส่วนผสมสองอย่างคือแป้งและน้ำตาล และสมมติว่าคุณภาพของคุ๊กกี้เป็นพังก์ชันแสดงใน **รูปที่ 2-8** ด้านล่างนี้

9	1	2	3	4	5	4	3	2	1
8	2	3	4	5	6	5	4	3	2
7	3	4	5	6	7	6	5	4	3
6	4	5	6	7	8	7	6	5	4
5	5	6	7	8	9	8	7	6	5
4	4	5	6	7	8	7	6	5	4
3	3	4	5	6	7	6	5	4	3
2	2	3	4	5	6	5	4	3	2
1	1	2	3	4	5	4	3	2	1
แป้ง	1	2	3	4	5	6	7	8	9

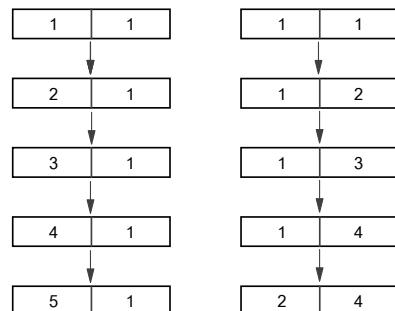
รูปที่ 6-1 พังก์ชันภูเขาระบบทองคุณภาพคุ๊กกี้

แนวตั้งและแนวนอนแสดงจำนวนกิโลกรัมของส่วนผสมน้ำตาลและแป้งตามลำดับ เช่น น้ำตาล 1 กก. กับแป้ง 1 กก. ผลิตได้คุกกี้มีคุณภาพ 1 หน่วย พังก์ชันนี้จะมีค่าสูงสุดอยู่ที่ 5-5 (น้ำตาล 5 กก. กับแป้ง 5 กก. ผลิตได้คุกกี้คุณภาพ 9 หน่วย) เราออกแบบให้แต่ละถาด ของคุกกี้ถูกแทนด้วยโครโน่ซึมเส้นหนึ่งตั้งแสดงในรูปที่ 1-1



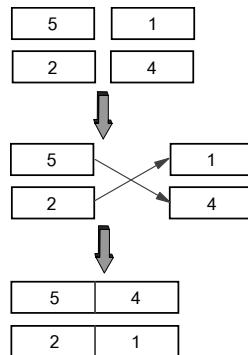
ในการออกแบบครั้งนี้กำหนดให้โครโน่ซึมมีค่า 2 ตัว ยืนด้านข้างแทนจำนวนกิโลกรัม ของน้ำตาลและยืนด้านขวาแทนจำนวนกิโลกรัมของแป้ง กำหนดให้ตัวเลขแสดงจำนวน กิโลกรัมของทั้งน้ำตาลและแป้งมีค่าตั้งแต่ 1 ถึง 9 โครโน่ซึมแทนถาดของคุกกี้นี้กำหนด ความเหมาะสม (fitness) กับธรรมชาติของคุกกี้ โครโน่ซึมสามารถสร้างขึ้นจากยืนน้ำตาลและ แป้ง สร้างขึ้นจากการไขว้เปลี่ยนของโครโน่ซึมพ่อแม่คุณหนึ่ง หรือสร้างได้จากการกลยุทธ์ ของยืนในโครโน่ซึมตัวหนึ่งที่มีอยู่ และถ้าหากเรามีโครโน่ซึม 1 เส้น เราสามารถตัดแป้ง เอเยี่ยนของน้ำตาลหรือยืนของแป้งได้

ในการทำจีโครั่งนี้ กำหนดให้ประชากรรุ่นหนึ่ง ๆ มีโครโน่ซึมที่เหมือนกันเพียงเส้นเดียว เราจำลองการเกิดการกลยุทธ์ของโครโน่ซึมโดยการเลือกยืนตัวหนึ่งแบบสุ่มแล้ว เปลี่ยนค่าของยืนโดยบวกหนึ่งหรือลบหนึ่งแบบสุ่มและยอมรับค่าที่ได้ถ้าค่าน้อยกว่า 1 ถึง 9 **รูปที่ 6-3** แสดงวิวัฒนาการของโครโน่ซึมโดยการกลยุทธ์ในรูปแสดงการกลยุทธ์ พันธ์สองรูปแบบซึ่งในแต่ละแบบแสดงการกลยุทธ์เมื่อผ่านไป 4 ครั้ง ในแต่ละครั้งยืนที่เลือกและค่าที่เปลี่ยนไปเกิดจากการสุ่มในครั้งนั้น ๆ เราเห็นได้ว่าเมื่อผ่านการกลยุทธ์ไป 4 ครั้งโครโน่ซึมที่ได้มีความต่างกันค่อนข้างมาก โครโน่ซึมเส้นที่ดีเหมาะสมกับธรรมชาติคือ ถูกคัดเลือกซึ่งจะกล่าวต่อไป



รูปที่ 6-3 การจำลองการกลยุทธ์ของโครโน่ซึมคุกกี้

เราจำลองการไขว้เปลี่ยนของโครโนซมโดยตัดที่กึ่งกลางของโครโนซมพ่อแม่ 2 เส้นแล้วนำแต่ละส่วนมาต่อ กัน ดังรูปที่ 4-1



รูปที่ 6-4 การจำลองการไขว้เปลี่ยนของโครโนซมคุกคิ้ว

จากรูปเราจะเห็นได้ว่าโครโนซมพ่อแม่ 5-1 และ 2-4 ผลิตได้โครโนซมลูกสองเส้นคือ 5-4 กับ 2-1 ในกรณีที่ว่าไปที่โครโนซมประกอบด้วยยีนมากกว่า 2 ตัว การตัดและการต่อจะซับซ้อนยิ่งขึ้น

เมื่อพิจารณาปริภูมิค้นหาในบทที่ 2 จะพบว่าการกalyพันธ์มีลักษณะเทียบเคียงได้กับตัวกระทำการในปริภูมิค้นหา มีหน้าที่สร้างสถานะ (โครโนซม) ลูกของสถานะปัจจุบันอย่างไรก็ได้การกalyพันธ์มีความแตกต่างอยู่ที่ลักษณะสำคัญของวิธีการจีเอชีงใช้ความน่าจะเป็นในกระบวนการค้นหา กล่าวคือการกalyพันธ์จะสร้างโครโนซมโดยการสุ่มและเมื่อเราพิจารณาการไขว้เปลี่ยนจะไม่พบตัวกระทำการในปริภูมิค้นหาที่มีการทำงานในลักษณะเช่นนี้ กล่าวได้ว่าการไขว้เปลี่ยนเป็นคุณสมบัติเฉพาะของจีเอ เปรียบเสมือนการกระโดดไปยังสถานะใหม่ 2 ตัวจากสถานะพ่อแม่ 1 คู่ ซึ่งข้อดีของการไขว้เปลี่ยนจะได้กล่าวต่อไป

6.1.2 ค่าความเหมาะสมมาตรฐาน

ค่าความเหมาะสม (fitness) ของโครโนซมคือความน่าจะเป็นที่โครโนซมจะอยู่รอดในรุ่น (generation) ถัดไป ค่าความเหมาะสมมาตรฐานสามารถนิยามได้ดังนี้

$$f_i = \frac{q_i}{\sum_j q_j} \quad (6.1)$$

โดยที่ f_i คือค่าความเหมาะสมของโครโนซมเส้นที่ i ซึ่งมีค่าระหว่าง 0 ถึง 1 และ q_i คือคุณภาพของคุกคิ้วที่ถูกกำหนดโดยโครโนซมเส้นที่ i

ตัวอย่างเช่น สมมติว่าประชากรประกอบด้วยโครโนซม 4 เส้นคือ 1-4, 3-1, 1-2, 1-1 ค่าความเหณะของโครโนซมแต่ละเส้นแสดงได้ในตารางที่ 6-1

ตารางที่ 6-1 ตัวอย่างค่าความเหณะมาตราฐานของโครโนซมคุกคัก

โครโนซม	คุณภาพ	ค่าความเหณะมาตราฐาน
1-4	4	0.40
3-1	3	0.30
1-2	2	0.20
1-1	1	0.10

ค่าความเหณะมาตราฐานที่คำนวณได้ในตารางนี้ (เช่นค่าความเหณะของโครโนซม 1-4 จะเท่ากับ $4/(4+3+2+1)=0.40$) เป็นความน่าจะเป็นที่โครโนซมจะอยู่รอด (ถูกเลือก) ในรุ่นถัดไป ดังนั้นโครโนซม 1-4 จะมีโอกาสอยู่รอดมากกว่าโครโนซมเส้นอื่นๆ และมีโอกาสอยู่รอดมากกว่าโครโนซม 1-1 ถึง 4 เท่า แต่ก็ไม่ได้หมายความว่าถ้าให้เลือกโครโนซมได้แค่เส้นเดียวแล้วโครโนซม 1-4 ที่มีค่าความเหณะสูงสุดจะถูกเลือกทุกครั้งไป แต่จะขึ้นอยู่กับการสุ่มค่า แน่นอนว่า 1-4 มีโอกาสมากที่สุด และถ้าการสุ่มทำได้อย่างไม่โน้มเอียงในการสุ่ม 100 ครั้ง 1-4 น่าจะมีโอกาสถูกเลือกสัก 40 ครั้ง

6.1.3 การจำลองการคัดเลือกโดยธรรมชาติ

เราได้ออกแบบโครโนซมสำหรับปัญหาที่เรานำไป การกalyพันธ์ การไขว้เปลี่ยน ค่าความเหณะแล้ว หัวข้อนี้จะกล่าวถึงการจำลองการคัดเลือกโดยธรรมชาติซึ่งสามารถทำได้โดยใช้ขั้นตอนทั่วไปดังต่อไปนี้

- กำหนดประชากรเริ่มต้น อาจมีโครโนซม 1 เส้นหรือหลายเส้นก็ได้ เราอาจสุ่มโครโนซมเหล่านี้หรือกำหนดขึ้นเองก็ได้
- ทำการกalyพันธ์ยืนในโครโนซมในรุ่นปัจจุบันและผลิตลูก
- ทำการไขว้เปลี่ยนโครโนซม (พ่อแม่) ในรุ่นปัจจุบันและผลิตลูก
- เพิ่มลูกที่เกิดใหม่ในประชากร
- สร้างประชากรรุ่นใหม่โดยเลือกโครโนซมตามค่าความเหณะอย่างสุ่ม

ในการแก้ปัญหานี้ๆ ที่เรานำใจด้วยก็เง้นนี้ เราจำเป็นต้องกำหนดพารามิเตอร์ต่างๆ ใน การจำลองการคัดเลือกโดยธรรมชาติ อย่างเช่นในประชากรรุ่นหนึ่งๆ ควรมีโครโนซมจำนวนเท่าไร? ถ้า้อยไปก็มีแนวโน้มว่าโครโนซมในประชากรรุ่นหนึ่งๆ จะมีลักษณะ

คล้ายกันหรือเหมือนกันเกือบทั้งหมดและการทำการไข้วัวเปลี่ยนก็จะไม่มีผลมากนัก แต่ถ้ามากไปเราก็จะเสียเวลาคำนวณมาก อัตราการกลยุทธ์เป็นเท่าไร? ถ้าต่ำไปลักษณะใหม่จะเกิดข้า ถ้าสูงไปประชารุ่นใหม่จะไม่เกี่ยวเนื่องกับรุ่นเดิม จะทำการไข้วัวเปลี่ยนด้วยหรือไม่? ถ้าทำจะเลือกคู่ผสมอย่างไร? และการไข้วัวเปลี่ยนกำหนดอย่างไร? ตัดครึ่งตรงกึ่งกลางหรือส่วนจุดตัด เป็นต้น โครโน่โอมเหมือนกันจะยอมให้มีหลายเส้นหรือไม่?

ในปัญหาการหาส่วนผสมดีสุดของคุณก็เป็น เราจะจำลองการคัดเลือกโดยธรรมชาติดังนี้

- เริ่มจากโครโน่โอม 1-1 เพียงเส้นเดียว
- โครโน่โอมที่เหมือนกันจะมีแค่เส้นเดียวในประชารุ่นหนึ่งๆ
- โครโน่โอม 4 เส้นหรือน้อยกว่าจะอยู่รอดไปถึงรุ่นใหม่
- สำหรับโครโน่โอมแต่ละเส้นที่อยู่รอด เลือกยืนตัวหนึ่งแบบสุ่มเพื่อทำการกลยุทธ์ ถ้าโครโน่โอมที่ได้จากการกลยุทธ์ยังไม่เคยมีมาเลยให้เพิ่มเข้าไปในประชารุ่น
- ไม่ทำการไข้วัวเปลี่ยน
- โครโน่โอมที่อยู่รอดจะแข่งขันกับโครโน่โอมใหม่เพื่อกำหนดโครโน่โอมที่จะอยู่ในรุ่นถัดไป โครโน่โอมที่มีค่าความเหมาะสมสูงสุดจะถูกเลือกเสมอให้อยู่รอดไปถึงรุ่นถัดไป ส่วนเส้นที่อยู่รอดที่เหลือจะถูกเลือกจากโครโน่โอมที่เหลือแบบสุ่มตามค่าความเหมาะสม

จากการทดลอง 1,000 ครั้งโดยใช้การคัดเลือกโดยธรรมชาติด้านบน พบว่าส่วนผสมที่ทำให้คุณภาพของคุณก็ดีที่สุดถูกผลิตในรุ่นที่ 16 โดยเฉลี่ย และในจำนวนนี้การทดลองที่ใช้ค่าที่สุดผลิตโครโน่โอมที่ดีที่สุดในรุ่นที่ 8 ดังแสดงในตารางที่ 6-2 ด้านล่างนี้

ตารางที่ 6-2 ผลการทดลองดีสุดผลิตโครโน่โอมดีสุดได้ในรุ่นที่ 8 โดยค่าความเหมาะสมมาตรฐาน

รุ่นที่ 0:			• 1-1 กลยุทธ์เป็น 1-2
โครโน่โอม	คุณภาพ		
1-1	1		
รุ่นที่ 1:			• 1-2 กลยุทธ์เป็น 1-3 และ 1-1 เป็น 1-2
โครโน่โอม	คุณภาพ	ซึ่งมีอยู่แล้ว	
1-2	2		
1-1	1		

รุ่นที่ 2:				• 1-3 กล้ายพันธุ์เป็น 1-4, 1-2 เป็น 2-2, 1-1 เป็น 2-1 โครโน่จะหมดมี 6 เส้น และ 4 เส้นถูกเลือกดังแสดงในรุ่นที่ 3 (1-4 ที่มีค่าความหมายสูงสุดถูกเลือกเลย ส่วนอีกสามเส้นที่เหลือได้จากการสุ่มตาม ค่าความหมาย สังเกตว่าแม้ว่า 2-2 จะมี ค่าความหมายดีกว่า 1-2 และ 2-1 แต่ไม่ ถูกเลือกในรุ่นนี้)
รุ่นที่ 3:				• การกล้ายพันธุ์ผลิตได้โครโน่ใหม่ 3 เส้นดังต่อไปนี้
โครโน่	คุณภาพ	โครโน่	คุณภาพ	
1-4	4	2-4	5	
1-3	3	2-3	4	
1-2	2	3-1	3	
2-1	2			
รุ่นที่ 4:				• โครโน่ทุกเส้นกล้ายพันธุ์และผลิตถูก
โครโน่	คุณภาพ			
2-4	5			
1-4	4			
1-3	3			
2-1	2			
รุ่นที่ 5:				• โครโน่ทุกเส้นกล้ายพันธุ์และผลิตถูก
โครโน่	คุณภาพ			
2-5	6			
1-5	5			
2-3	4			
2-2	3			
รุ่นที่ 6:				• 3-5 กล้ายพันธุ์เป็น 4-5, 3-2 เป็น 3-1, 1-4 เป็น 1-5, 1-5 เป็น 1-4 จะเห็นได้ ว่าการผลิตถูกมักมีการซ้ำซ้อนของ โครโน่ เช่น ไปซ้ำเดิมกับพ่อแม่เป็น ต้น
โครโน่	คุณภาพ			
3-5	7			
1-5	5			
3-2	4			
1-4	4			

รุ่นที่ 7:		• ที่จุดนี้ 4-5 กลายพันธุ์เป็น 5-5 ซึ่งเป็น	
โครงไมโครซม	คุณภาพ	คำตอบในที่สุด	
4-5	8		
1-5	5		
1-4	4		
3-1	3		

รุ่นที่ 8:	
โครงไมโครซม	คุณภาพ
5-5	9
4-5	8
2-5	6
2-1	2

6.1.4 การไขว้เปลี่ยนเพื่อเอาชนะจุดดีสุดเฉพาะที่

หัวข้อนี้เราจะดูผลของการไขว้เปลี่ยนที่มีต่อจีเอ โดยทำการทดลองเมื่อทำการทดลองที่แล้ว แต่เพิ่มการไขว้เปลี่ยนเพื่อสร้างโครงไมโครซมใหม่ด้วย การไขว้เปลี่ยนทำดังต่อไปนี้

- ทำการไขว้เปลี่ยนโดยใช้โครงไมโครซมที่อยู่รอดจากรุ่นที่แล้ว (อย่างมากสุด 4 เส้น)
- สำหรับโครงไมโครซมที่จะทำการไขว้เปลี่ยนเส้นหนึ่งๆ ให้เลือกคู่ทำการไขว้เปลี่ยนแบบสุ่ม
- สลับยืนของโครงไมโครซมพ่อแม่และผลิตโครงไมโครซมลูก 2 เส้น ถ้าโครงไมโครซมลูกยังไม่เคยมีมาเลยให้เพิ่มเข้าไปในประชากรเพื่อแข่งขันที่จะอยู่รอดในรุ่นถัดไป

ผลจากการทดลอง 1,000 ครั้ง ส่วนผสมที่ดีที่สุดถูกผลิตในรุ่นที่ 14 โดยเฉลี่ย ใช้จำนวนรุ่นน้อยกว่ากรณีไม่ใช้การไขว้เปลี่ยน 2 รุ่น แม้ว่าการไขว้เปลี่ยนจะช่วยให้เราพบส่วนผสมดีสุดโดยใช้จำนวนรุ่นน้อยกว่าเดิม แต่เราต้องใช้การคำนวณในแต่ละรุ่นมากขึ้น กว่าเดิมเนื่องจากจำนวนโครงไมโครซมที่มากขึ้นและการคำนวณค่าความเหมาะสมที่เพิ่มขึ้น ดังนั้นเวลาโดยรวมจะเพิ่มขึ้นกว่าเดิม สำหรับปัญหานี้เป็นปัญหาที่ไม่มีจุดดีสุดเฉพาะที่ มีแค่จุดดีสุดวงกว้างจุดเดียว ประสิทธิภาพของการไขว้เปลี่ยนจึงไม่เห็นอย่างชัดเจน ปัญหาที่เราจะพิจารณาต่อไปเป็นปัญหาที่มีจุดดีสุดเฉพาะที่ซึ่งจะสร้างความยากลำบาก สำหรับวิธีการค้นหาทั่วไป แต่จีเอสามารถจัดการกับปัญหาลักษณะนี้ได้ ปัญหานี้เป็นการหาส่วนผสมดีสุดของคึกคักที่เหมือนเดิมแต่ใช้ฟังก์ชันใหม่ดังรูปที่ 6-5 ต่อไปนี้

9	1	2	3	4	5	4	3	2	1
8	2	0	0	0	0	0	0	0	2
7	3	0	0	0	0	0	0	0	3
6	4	0	0	7	8	7	0	0	4
5	5	0	0	8	9	8	0	0	5
4	4	0	0	7	8	7	0	0	4
3	3	0	0	0	0	0	0	0	3
2	2	0	0	0	0	0	0	0	2
1	1	2	3	4	5	4	3	2	1
1 2 3 4 5 6 7 8 9 ແປ້ງ									

รูปที่ 6-5 พังก์ชันกฎเขามีคุณลักษณะของคุณภาพคุ้กคูก

เริ่มต้นจากໂຄຣໂມໂຄຣມ 1-1 ເຊັ່ນເດີມ ເຮັດວຽກວ່າໃນກຣານີນີ້ກຣາລາຍພັນຊີເພີຍງອຍ່າງເດືອຍໄວ່ໄໝສາມາດກຳທຳໃຫ້ໂຄຣໂມໂຄຣມໃນຮຸນທີ່ອູ່ກາຍນອກຄູ້ນໍາ (ບຣິເວນທີ່ມີຄ່າເປັນ 0) ພລິຕໂຄຣໂມໂຄຣມທະລຸເຂົ້າໄປອູ່ພື້ນທີ່ກາຍໃນຄູ້ນໍາໄດ້ ເນື່ອງຈາກໂຄຣໂມໂຄຣມທຽງກລາງມີຄ່າຄວາມເໝາະເປັນ 0 ຜຶ້ງໄໝສາມາດຄອງຢູ່ຮອດໃນຮຸນຄັດໄປໄດ້ (ຄ່າຄວາມເໝາະຂອງໂຄຣໂມໂຄຣມເປັນ 0 ທຳໃຫ້ຄວາມນໍາຈະເປັນທີ່ຈະອູ່ຮອດໄມໜີເລຍ) ອຍ່າງໄຣກີດການໄຂວ້ເປັນກີນທີ່ຈັບຄູ້ໂຄຣໂມໂຄຣມພ່ອແມ່ທີ່ເໝາະສົມເຊັ່ນ 1-5 ແລະ 5-1 ຈະສາມາດພລິຕລຸກທີ່ຂັ້ມຄູ້ນໍາໄປໄດ້ ຈາກກາຮທດລອງ 1,000 ຄຮັ້ງພບວ່າສ່ວນຜສມທີ່ດີທີ່ສຸດຄູກພລິຕໃນຮຸນທີ່ 155 ໂດຍຈະເລີ່ມ!! ເປັນຜລທີ່ໄໝຈີ ດ້ວຍຄໍາວານຄູກຈະກວາມທັນທີວ່າໂຄຣໂມໂຄຣມທີ່ແຕກຕ່າງກັນທີ່ເປັນໄປໄດ້ທັງໝາດມີແຕ່ $9 \times 9 = 81$ ເສັ້ນເທົ່ານັ້ນ ຜລທີ່ໄໝຈີອຸ່ນທີ່ 155 ແລະ ແຕ່ລະຮຸນມີໂຄຣໂມໂຄຣມທີ່ເຮົາດສອນນາກກວ່າໜຶ່ງເສັ້ນ (ແນວ່າຈະມີໂຄຣໂມໂຄຣມມາກມາຍທີ່ຂໍ້ກັນໃນຮຸນຕ່າງໆ)

ສາເຫຼຸ່າຫຼືນີ້ທີ່ຜລໄມ້ດີກີພຣະວ່າກ່ອນທີ່ໂຄຣໂມໂຄຣມຈະກຣາຍພັນຊີເປັນໂຄຣໂມໂຄຣມທີ່ອູ່ບຣິເວນ 1-5 ທີ່ຮູ້ 5-1 ນັ້ນ ໂດຍມາກຕາຍໄປກ່ອນທີ່ຈະໄປສູບຮິເວນນັ້ນສໍາເລົງ ແລະ ໂອກາສທີ່ຄູ້ທີ່ເໝາະສົມຂອງໂຄຣໂມໂຄຣມຈະເກີດການໄຂວ້ເປັນກີນມີໂອກາສນ້ອຍມາກ ຜຶ້ງທີ່ຈິງແລ້ວຄູ້ທີ່ເໝາະສົມຂອງການໄຂວ້ເປັນມີຈຳນວນນາກອຍ່າງເຊັ່ນ 2-6 ກັບ 4-2, 4-8 ກັບ 2-5, 6-8 ກັບ 2-4 ເປັນຕົ້ນ ແລະ ໂຄຣໂມໂຄຣມໃນຄູ້ທັງໝາດນີ້ລ້ວນມີຄວາມນໍາຈະເປັນທີ່ຈະອູ່ຮອດເປັນ 0 ທັງສິ້ນ ທີ່ເປັນເຊັ່ນນີ້ເກີດຂຶ້ນຈາກພັງກົນຄວາມເໝາະມາຕຣານທີ່ຈະກຳທັນໃຫ້ໂຄຣໂມໂຄຣມເຫັນນີ້ມີຄວາມນໍາຈະເປັນທີ່ຈະອູ່ຮອດເປັນ 0 ກ່າວເປັນແກ້ພັງກົນຄວາມເໝາະໃຫ້ໂຄຣໂມໂຄຣມເຫັນນີ້ມີໂອກາສອູ່ຮອດບ້ານແນ້ຈະນ້ອຍ ກົ້າຈະຂ່າຍໃຫ້ການຄັນຫາສ່ວນຜສມດີສຸດທຳໄດ້ຂຶ້ນ ດັ່ງຈະກຳລ່າວໃນຫຼວຂ້ອຕ່ອໄປ

6.1.5 ปรับปรุงจีเอด้วยฟังก์ชันความเหมาะสมแบบลำดับและการใช้ความหลากหลาย

ปรับปรุงจีเอด้วยฟังก์ชันความเหมาะสมแบบลำดับ

ฟังก์ชันความเหมาะสมใหม่ที่เราจะพิจารณา กันนี้เรียกว่า **ค่าความเหมาะสมแบบลำดับ (rank fitness)** เป็นวิธีที่ใช้ความคุณภาพในการเลือกโครโนซ์โดยไม่สนใจคุณภาพของโครโนซ์ว่ามีค่าเท่าไร จะเพียงแค่จัดลำดับเรียงโครโนซ์ตามคุณภาพที่มีค่าสูงสุดจนถึงต่ำสุด จากนั้นกำหนดให้ p ค่าคงที่ค่าหนึ่งเป็นความน่าจะเป็นที่โครโนซ์มีลำดับที่ 1 จะถูกเลือก และเป็นความน่าจะเป็นที่โครโนซ์มีลำดับที่ 2 จะถูกเลือกเมื่อลำดับที่ 1 ไม่ถูกเลือก และเป็นความน่าจะเป็นที่ลำดับที่ 3 จะถูกเลือกเมื่อลำดับที่ 1 และ 2 ไม่ถูกเลือก เป็นเช่นนี้ไปจนกระทั่งถึงลำดับสุดท้ายซึ่งจะถูกเลือกเมื่อลำดับก่อนหน้ามันไม่ถูกเลือกเลย

ตัวอย่างเช่นสมมติว่า $p=2/3$ และโครโนซ์ที่เราสนใจอยู่คือ 1-4, 3-1, 1-2, 1-1 และ 7-5 (ในกรณีของภูเขามีคูน้ำล้อม) จะได้ค่าความเหมาะสมของโครโนซ์ดังตารางที่ 6-3 ซึ่งเปรียบเทียบค่าความเหมาะสมแบบลำดับกับค่าความเหมาะสมมาตรฐาน

ตารางที่ 6-3 เปรียบเทียบค่าความเหมาะสมแบบลำดับกับค่าความเหมาะสมมาตรฐาน

โครโนซ์	คุณภาพ	ลำดับ	ค่าความเหมาะสมมาตรฐาน	ค่าความเหมาะสมแบบลำดับ
1-4	4	1	0.40	0.667
1-3	3	2	0.30	0.222
1-2	2	3	0.20	0.074
1-1	1	4	0.10	0.025
7-5	0	5	0.00	0.012

ดังแสดงในตารางที่ 6-3 ค่าความเหมาะสมแบบลำดับของโครโนซ์ 1-4 เท่ากับ $p = 2/3$ (ประมาณ 0.667) ส่วนโครโนซ์ 1-3 มีค่าความน่าจะเป็นเท่ากับ $p(1-p)$ (ความน่าจะเป็นที่ตัวเองจะถูกเลือกเมื่อโครโนซ์มีลำดับที่ 1 ไม่ถูกเลือก) ซึ่งมีค่าประมาณ 0.222 ส่วนลำดับที่ 3 จะถูกเลือกเมื่อเส้นที่ 1 และ 2 ไม่ถูกเลือกด้วยความน่าจะเป็นเท่ากับ $p(1-p)(1-p) \approx 0.074$ ส่วนเส้นที่ 4 ก็เท่ากับ $p(1-p)(1-p)(1-p) \approx 0.025$ และเส้นสุดท้ายมีค่าความน่าจะเป็นเท่ากับ $1 - (0.667 + 0.222 + 0.074 + 0.025 + 0.012) = 0.012$

จากผลการทดลอง 1,000 ครั้งโดยใช้ค่าความเหมาะสมแบบลำดับและจำลองการคัดเลือกโดยธรรมชาติเหมือนเดิมทุกประการ พบว่าส่วนผสมที่ดีที่สุดถูกผลิตในรุ่นที่ 75 โดยเฉลี่ยเร็วขึ้นกว่าเดิม (ส่วนผสมที่สุดถูกผลิตในรุ่นที่ 155) ประมาณ 2 เท่า ซึ่งแสดงให้เห็นว่าค่าความเหมาะสมแบบลำดับดีกว่าค่าความเหมาะสมมาตรฐาน และจากการใช้ค่าความเหมาะสมแบบลำดับนี้ทำให้โครโนซ์ที่อยู่ตรงกลางในคูน้ำสามารถอยู่รอดถึงรุ่นถัดไปและวิวัฒนาการเป็น

โครโน่โชมที่อยู่ภายใต้ชื่อคุณภาพสูงต่อไปได้ อย่างไรก็ได้มีว่าค่าความหมายแบบลำดับจะทำให้เร็วขึ้นกว่าเดิมประมาณ 2 เท่า แต่ยังคงเป็นผลที่ไม่ดีนักดังเช่นที่ได้กล่าวแล้วว่า 75 รุ่นแต่ละรุ่นเราตรวจสอบโครโน่โชมมากกว่า 1 เส้น

เพิ่มประสิทธิภาพจีเอให้สูงขึ้นโดยความหลากหลาย

หัวข้อนี้แสดงการใช้ **ความหลากหลาย (diversity)** เพื่อเพิ่มประสิทธิภาพของจีเอให้สูงขึ้นอีกชั้นได้แนวคิดจากการวิพากษากิจกรรมของสิ่งมีชีวิต ที่เรามักพบว่าบ่อยครั้งในธรรมชาติที่สเปชีส์ชั้นลักษณะแตกต่างไปจากสเปชีส์ที่หมายกับธรรมชาติสามารถอยู่รอดได้ดี ซึ่งความหลากหลายนี้จะช่วยให้โครโน่โชมที่มีข้อผิดพลาดทางพากพ้องถูกคัดเลือกได้ง่ายขึ้น

การจะนำความต่างเข้าไปช่วยเลือกโครโน่โชมนั้น อย่างแรกที่ต้องทำก็คือนิยามความต่างให้ชัดเจน ในการนี้เราจะดูความต่างของโครโน่โชมเส้นหนึ่งๆ โดยคำนวณค่าของ “ผลรวมของ $1/\text{ระยะห่างกำลังสองระหว่างโครโน่โชมนั้นกับโครโน่โชมอื่นที่ถูกเลือกแล้วว่าให้อยู่รอดในรุ่นถัดไป}$ ” เนื่องจากเราต้องการโครโน่โชมเส้นที่ต่างจากโครโน่โชมที่หมายกับธรรมชาติ ดังนั้นการวัดความต่างหรือความหลากหลายจึงเทียบกับโครโน่โชมเส้นที่หมายกับธรรมชาติ ส่วนระยะห่างหมายถึงระยะห่างตามระยะบุคคล (Euclidian distance) เช่น 5-2 กับ 1-4 มีระยะห่างกำลังสองเท่ากับ $(5-1)^2 + (2-4)^2 = 20$ เป็นต้น

พิจารณาโครโน่โชม 5-1, 1-4, 3-1, 1-2, 1-1 และ 7-5 โครโน่โชมที่มีคุณภาพสูงสุดคือ 5-1 (ซึ่งเราจะเลือกเลยให้อยู่ในรุ่นถัดไปเป็นเส้นแรก) **ตารางที่ 6-4** ด้านล่างแสดงลำดับของ 5 เส้นที่เหลือโดยเรียงตามคุณภาพและผลรวม 1/ระยะห่างกำลังสองจาก 5-1

ตารางที่ 6-4 ลำดับของโครโน่โชมเรียงตามลำดับความหลากหลายและลำดับคุณภาพ

โครโน่โชม	คุณภาพ	$1/d^2$	ลำดับความหลากหลาย	ลำดับคุณภาพ
1-4	4	0.040	1	1
3-1	3	0.250	5	2
1-2	2	0.059	3	3
1-1	1	0.062	4	4
7-5	0	0.050	2	5

$1/d^2$ แสดง 1/ระยะห่างกำลังสองระหว่างโครโน่โชมที่พิจารณา กับ 5-1 ด้วยอย่างเช่น 1-4 กับ 5-1 มีค่าเท่ากับ $1/((5-1)^2 + (1-4)^2) = 0.040$ เป็นต้น จากตารางจะพบว่าโครโน่โชม 7-5 ซึ่งมีคุณภาพเป็น 0 และจะไม่เคยถูกเลือกเลยโดยค่าความหมายมาตรฐาน แต่เมื่อคำนวณค่าความหมายแบบลำดับความหลากหลายจะอยู่ในลำดับที่ 2 ซึ่งในกรณีนี้เมื่อถูกจาก

รูปที่ 6-5 จะเห็นว่า 7-5 เป็นโครโนซมที่ดีเส้นหนึ่งและมีโอกาสสกัดรายพันธุ์เข้าสู่บริเวณด้านในของคุณ้ำเพื่อเป็นคำตอบต่อไป

เมื่อเราได้ลำดับความหลากหลายแล้ว เราจำเป็นต้องนำลำดับนี้ผ่านกับค่าความหมายเดิม เราไม่อาจใช้ลำดับความหลากหลายอย่างเดียวได้ เพราะเป็นแค่ปัจจัยหนึ่งในการเลือกโครโนซม ลำดับคุณภาพเดิมซึ่งค่อนข้างดีอยู่แล้วก็ไม่อาจตัดทิ้งได้ ดังนั้นวิธีผ่านกับลำดับความหลากหลายเข้าใช้ร่วมกับลำดับคุณภาพสามารถทำได้โดยนำลำดับทั้งสองนั้นกันแล้วจัดเรียงลำดับใหม่อีกรัง เราเรียกลำดับที่ได้ใหม่นี้ว่า **ลำดับรวม (combined rank)** เมื่อได้ลำดับรวมซึ่งคิดทั้งคุณภาพและความหลากหลายแล้ว การเลือกจะทำได้เหมือนเดิมโดยกำหนดความน่าจะเป็นของลำดับแรกเป็น $p = 2/3$ (ดูตารางที่ 6-5)

ตารางที่ 6-5 ลำดับรวมที่พิจารณาทั้งคุณภาพและความหลากหลาย

โครโนซม	ผลรวมของลำดับคุณภาพและลำดับความหลากหลาย	ลำดับผลรวม	ค่าความหมาย
1-4	2	1	0.667
3-1	7	4	0.025
1-2	6	2	0.222
1-1	8	5	0.012
7-5	7	3	0.074

ลำดับผลรวมในตารางได้จากการเรียงลำดับผลในสุดมภที่สองใหม่ ในการนี้ที่มีค่าเท่ากันอย่างเช่น 3-1 กับ 7-5 มีค่าเท่ากันเท่ากับ 7 ก็ใช้การสุ่มเลือก ในที่นี้ 7-5 ถูกสุ่มให้มีลำดับผลรวมเป็นลำดับสาม จากตารางสมมติว่าเราเลือกโครโนซมตามค่าความหมายได้เป็น 1-4 และเป็นเส้นที่สองต่อจาก 5-1 หลังจากนี้เราจะเลือกเส้นที่ 3 ในครั้งนี้เราต้องคำนวณหา 1/ระยะห่างกำลังสอง โดยคิดทั้ง 5-1 และ 1-4 (ดูตารางถัดไป)

ตารางที่ 6-6 การเลือกโครโนซมเส้นที่ 3 ต่อจาก 5-1 และ 1-4

โครโนซม	$\sum_i \frac{1}{d_i^2}$	ลำดับความหลากหลาย	ลำดับคุณภาพ	ลำดับรวม	ค่าความหมาย
3-1	0.327	4	1	4	0.037
1-2	0.309	3	2	3	0.074
1-1	0.173	2	3	2	0.222
7-5	0.077	1	4	1	0.667

ตัวอย่างการคำนวณค่าของ $\sum_i \frac{1}{d_i^2}$ อย่างเช่นในกรณีของโครโนซม 3-1 จะได้ค่าเป็น

$$\frac{1}{(5-3)^2+(1-1)^2} + \frac{1}{(1-3)^2+(4-1)^2} = 0.327$$
 เป็นต้น สมมติว่าโครโนซมที่ถูกเลือกตามค่าความหมายเส้นต่อไปคือ 7-5 และโครโนซมเส้นสุดท้ายเรียกว่าสามารถทำได้ในลักษณะเดียวกัน และเลือกได้เป็น 1-1 ดังแสดงตารางที่ 6-7 ต่อไปนี้

ตารางที่ 6-7 การเลือกโครโนซมเส้นที่ 3 ต่อจาก 5-1, 1-4 และ 7-5

โครโนซม	$\sum_i \frac{1}{d_i^2}$	ลำดับความ หลากหลาย	ลำดับ คุณภาพ	ลำดับรวม	ค่า ความหมาย
3-1	0.358	3	1	3	0.111
1-2	0.331	2	2	2	0.222
1-1	0.190	1	3	1	0.667

ค่าความหมายที่คำนวณตามลำดับรวมมีความแตกต่างจากค่าความหมายมาตรฐานที่โครโนซม 7-5 ซึ่งเป็นโครโนซมที่ดีเส้นหนึ่งและไม่เคยถูกเลือกเลยด้วยค่าความหมายมาตรฐาน แต่สามารถจะถูกเลือกได้ด้วยค่าความหมายตัวใหม่นี้

จากการทดลอง 1,000 ครั้งโดยใช้ลำดับรวมค่า $p = 2/3$ เริ่มจากโครโนซม 1-1 คำตอบที่ดีที่สุดถูกผลิตได้ในรุ่นที่ 15 โดยเฉลี่ย!!! เร็วกว่าลำดับคุณภาพถึง 5 เท่า นอกจากนั้นค่าความหมายแบบลำดับรวมนี้ไม่ได้ถูกพัฒนาขึ้นโดยเฉพาะสำหรับแก้ปัญหาภูเขาเมื่อถูกน้ำล้อมอย่างเดียวเท่านั้น ยังสามารถทำงานได้ดีสำหรับปัญหาภูเขาเรียบด้วย ซึ่งดูสรุปการเปรียบเทียบค่าความหมายได้ในตารางที่ 6-8 ด้านล่างนี้ (ค่าในตารางได้จากการใช้การกลยุทธ์และการไขว้เปลี่ยนเหมือนกันหมด)

ตารางที่ 6-8 เปรียบเทียบค่าความหมาย 3 วิธี: มาตรฐาน ลำดับคุณภาพ และลำดับรวม

พั้งก์ชัน	ค่าความหมาย มาตรฐาน	ค่าความหมายแบบ ลำดับคุณภาพ		ค่าความหมายแบบ ลำดับรวม
		ลำดับคุณภาพ	ลำดับรวม	
ภูเขาเรียบ	14	12	12	
ภูเขามีคุน้ำล้อม	155	75	15	

ในจำนวนการทดลอง 1,000 ครั้งโดยใช้ลำดับรวมนั้น ครั้งที่ดีที่สุดโครโนซม 5-5 ถูกผลิตในรุ่นที่ 7 ดังแสดงในตารางที่ 6-9 ต่อไปนี้

ตารางที่ 6-9 ผลการทดลองดีสุดที่ผลิตโครโนซมดีสุดได้ในรุ่นที่ 7 โดยลำดับรวม

รุ่นที่ 0:			• 1-1 กล้ายพันธ์เป็น 2-1
โครโนซม	คุณภาพ		
1-1	1		
รุ่นที่ 1:			• การกล้ายพันธ์ผลิตได้ 3-1 ส่วนการไขว้เปลี่ยนไม่ได้ลูกตัวใหม่เพราะยืนตัวที่สองเหมือนกัน
โครโนซม	คุณภาพ		
2-1	2		
1-1	1		
รุ่นที่ 2:			• การกล้ายพันธ์ผลิตได้ 4-1 และ 2-2 ส่วนการไขว้เปลี่ยนยังคงไม่เกิดผล
โครโนซม	คุณภาพ		
3-1	3		
2-1	2		
1-1	1		
รุ่นที่ 3:			• การกล้ายพันธ์ผลิตได้โครโนซมใหม่ 3 เส้นคือ 5-1, 1-2, 2-3 ส่วนการไขว้เปลี่ยนของ 2-2 กับ 4-1 ผลิตได้ 2-1 กับ 4-2 การไขว้เปลี่ยนของคู่อื่นซ้ำกับโครโนซมที่ผลิตได้ก่อนมัน
โครโนซม	คุณภาพ		
4-1	4		
3-1	3		
1-1	1		
2-2	0		
โครโนซม	คุณภาพ		
		5	
		2	
		0	
		2	
		0	
รุ่นที่ 4:			• การกล้ายพันธ์ผลิตได้ 6-1, 3-2, 2-2, 2-4 ส่วนการไขว้เปลี่ยนผลิต 2-1, 1-1, 5-2, 3-2, 5-3
โครโนซม	คุณภาพ		
5-1	5		
3-1	4		
1-2	2		
2-3	0		
โครโนซม	คุณภาพ		
		4	
		0	
		0	

2-4	0
2-1	2
1-1	1
5-2	0
3-2	0
5-3	0
<hr/>	
รุ่นที่ 5: โครงไม้chrom	คุณภาพ
5-1	5
3-1	3
1-2	2
2-4	0
<hr/>	
รุ่นที่ 6: โครงไม้chrom	คุณภาพ
5-4	8
1-4	4
3-1	3
1-2	2
<hr/>	
รุ่นที่ 7: โครงไม้chrom	คุณภาพ
5-5	9
1-4	4
1-2	2
5-2	0

- ที่จุดนี้เกิดการไขว้เปลี่ยนของ 5-1 กับ 2-4 ได้ 5-4 ซึ่งเป็นโครงไม้chromที่ดีในรุ่นหน้า

- และในท้ายที่สุด 5-4 กลายพันธุ์เป็น 5-5

ด้านล่างนี้แสดงการค้นหาคำตอบโดยจีเอ โดยแสดงเฉพาะโครงโน้มที่ถูกเลือกในแต่ละรุ่น

(0)	1 2 3 4 5 4 3 2 1 2 0 0 0 0 0 0 0 2 3 0 0 0 0 0 0 0 3 4 0 0 7 8 7 0 0 4 5 0 0 8 9 8 0 0 5 4 0 0 7 8 7 0 0 4 3 0 0 0 0 0 0 0 3 2 0 0 0 0 0 0 0 2 1 2 3 4 5 4 3 2 1
-----	---

(2)	1 2 3 4 5 4 3 2 1 2 0 0 0 0 0 0 0 2 3 0 0 0 0 0 0 0 3 4 0 0 7 8 7 0 0 4 5 0 0 8 9 8 0 0 5 4 0 0 7 8 7 0 0 4 3 0 0 0 0 0 0 0 3 2 0 0 0 0 0 0 0 2 1 2 3 4 5 4 3 2 1
-----	---

(4)	1 2 3 4 5 4 3 2 1 2 0 0 0 0 0 0 0 2 3 0 0 0 0 0 0 0 3 4 0 0 7 8 7 0 0 4 5 0 0 8 9 8 0 0 5 4 0 0 7 8 7 0 0 4 3 0 0 0 0 0 0 0 3 2 0 0 0 0 0 0 0 2 1 2 3 4 5 4 3 2 1
-----	---

(6)	1 2 3 4 5 4 3 2 1 2 0 0 0 0 0 0 0 2 3 0 0 0 0 0 0 0 3 4 0 0 7 8 7 0 0 4 5 0 0 8 9 8 0 0 5 4 0 0 7 8 7 0 0 4 3 0 0 0 0 0 0 0 3 2 0 0 0 0 0 0 0 2 1 2 3 4 5 4 3 2 1
-----	---

(1)	1 2 3 4 5 4 3 2 1 2 0 0 0 0 0 0 0 2 3 0 0 0 0 0 0 0 3 4 0 0 7 8 7 0 0 4 5 0 0 8 9 8 0 0 5 4 0 0 7 8 7 0 0 4 3 0 0 0 0 0 0 0 3 2 0 0 0 0 0 0 0 2 1 2 3 4 5 4 3 2 1
-----	---

(3)	1 2 3 4 5 4 3 2 1 2 0 0 0 0 0 0 0 2 3 0 0 0 0 0 0 0 3 4 0 0 7 8 7 0 0 4 5 0 0 8 9 8 0 0 5 4 0 0 7 8 7 0 0 4 3 0 0 0 0 0 0 0 3 2 0 0 0 0 0 0 0 2 1 2 3 4 5 4 3 2 1
-----	---

(5)	1 2 3 4 5 4 3 2 1 2 0 0 0 0 0 0 0 2 3 0 0 0 0 0 0 0 3 4 0 0 7 8 7 0 0 4 5 0 0 8 9 8 0 0 5 4 0 0 7 8 7 0 0 4 3 0 0 0 0 0 0 0 3 2 0 0 0 0 0 0 0 2 1 2 3 4 5 4 3 2 1
-----	---

(7)	1 2 3 4 5 4 3 2 1 2 0 0 0 0 0 0 0 2 3 0 0 0 0 0 0 0 3 4 0 0 7 8 7 0 0 4 5 0 0 8 9 8 0 0 5 4 0 0 7 8 7 0 0 4 3 0 0 0 0 0 0 0 3 2 0 0 0 0 0 0 0 2 1 2 3 4 5 4 3 2 1
-----	---

รูปที่ 6-6 การค้นหาโดยจีเอในปัญญาเวชนาคูณล้อม

จากรูปจะเห็นได้ว่าในรุ่นที่ 3 โครงโน้มที่คุณภาพเป็น 0 สามารถถูกเลือกได้โดยค่าความหมายแบบลำดับรวมและจะเห็นการเคลื่อนที่ของโครงโน้มจากรุ่นที่ 1 ถึง 4 ว่า โครงโน้มค่อยๆ ขยายตัวไปยังจุดสูงสุดเฉพาะที่ซึ่งมีคุณภาพเท่ากับ 5 และจะเห็นการเคลื่อนที่ของโครงโน้มที่มีคุณภาพเท่ากับ 0 ที่ค่อยๆ ขยายออกจากจุดสูงสุดเฉพาะที่ที่ล

น้อย จนกระทั่งในรุ่นที่ 5 เมื่อออยู่ในตำแหน่งที่เหมาะสมและเกิดการไขว้เปลี่ยนกับจุดสูงสุด เฉพาะที่แล้วสามารถทະลุฝ่านคุณ้ำเข้าไปยังภัยในคุ้ดได้ แล้วเปลี่ยนเป็นจุดสูงสุดในที่สุด

จากรูปแสดงการทำงานของจีเอ เราสามารถเห็นได้ว่าการค้นหาโดยทั่วไปมักจะพยายาม หลีกเลี่ยงจุดดีสุดเฉพาะที่ แต่การทำงานของจีเอใช้วิธีการที่ต่างไป โดยการผลิตโครโนซมที่ เป็นค่าดีสุดเฉพาะที่จากนั้นจึงใช้ความหลากหลายเพื่อเป็นส่วนประกอบของค่าความเหมาะสม แล้วผลิตโครโนซมที่อยู่ห่างออกจากค่าดีสุดเฉพาะที่ หากมีโครโนซมอยู่ในจุดดีสุดเฉพาะที่ ทุกจุดแล้ว ก็มีโอกาสที่โครโนซมเหล่านี้จะหาทางไปยังจุดดีสุดของภัย (global optimum) ได้ในที่สุด

6.2 การเรียนรู้โดยการจำ

การเรียนรู้โดยการจำ (rote learning) เป็นการเรียนรู้แบบที่ง่ายที่สุดของกระบวนการเรียนรู้ ทั้งหลาย โดยเมื่อพบความรู้หรือข้อเท็จจริงใหม่ๆ ก็เก็บไว้ในหน่วยความจำ เวลาที่ต้องการใช้ก็เพียงแค่ดึงความรู้นั้นมาใช้ ถ้าเรามองว่าระบบปัญญาประดิษฐ์มีหน้าที่รับอินพุต (X_1, \dots, X_n) และทำการหาเอาต์พุต (Y_1, \dots, Y_n) = $f(X_1, \dots, X_n)$ โดยที่ f เป็นฟังก์ชันใดๆ ใน การคำนวนเอาต์พุตหรืออาจเป็นการอนุมานหาค่าเอาต์พุตจากอินพุตก็ได้ ดังนั้นการเรียนรู้โดยการจำคือการเก็บคู่ลำดับ $[(X_1, \dots, X_n), (Y_1, \dots, Y_n)]$ ไว้ในหน่วยความจำ หลังจากนั้น เมื่อเราต้องการหา $f(X_1, \dots, X_n)$ ใหม่ก็ทำโดยการดึง (Y_1, \dots, Y_m) จากคู่ลำดับนี้เท่านั้นโดย ไม่ต้องคำนวนหรืออนุมานซ้ำอีกรังซึ่งโดยมากจะเสียต้นทุนและเวลาสูง

จะเห็นได้ว่าแนวคิดนี้ง่ายแต่ไม่ได้หมายความว่าการเรียนรู้นี้จะไม่มีประสิทธิภาพ มนุษย์ แรกเรียนรู้โดยการจำด้วยเช่นกันหรือซอฟต์แวร์ในปัจจุบันหลายตัวก็สามารถจำข้อมูลที่ ผู้ใช้งานครั้งล่าสุดได้และช่วยให้การเปิดไฟล์ทำได้ง่ายขึ้นมีประโยชน์ในการใช้งานจริง

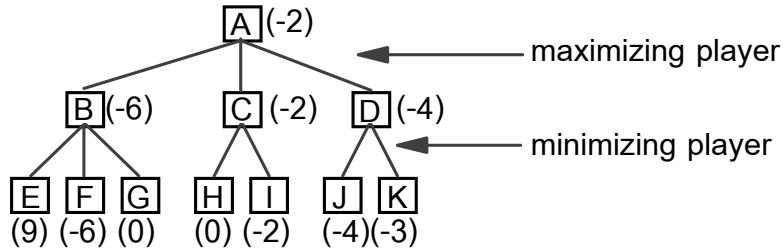
ในการเรียนรู้โดยการจำนี้ สิ่งที่เราต้องพิจารณาเพิ่มเติมได้แก่ (1) การจัดการ หน่วยความจำ (memory organization) ที่ต้องมีประสิทธิภาพสามารถดึงความรู้ที่เก็บไว้ได้ อย่างรวดเร็ว (2) ความเสถียรภาพของสภาพแวดล้อมต้องไม่เปลี่ยนแปลงอย่างรวดเร็วจน ส่งผลให้ความรู้ที่เก็บไว้ไม่ถูกต้องเมื่อเวลาเปลี่ยนไป (3) ความสมดุลย์ระหว่างการคำนวน ใหม่กับการจัดเก็บ ต้องมีสมดุลย์ที่ดีไม่จัดเก็บมากไปจนทำให้การค้นคืนคู่ลำดับที่จัดเก็บมี ประสิทธิภาพต่ำ ส่งผลให้ประสิทธิภาพโดยรวมลดลง เพราะเสียเวลามากไปเพื่อตรวจสอบว่า เป็นความรู้ที่อยู่ในหน่วยความจำหรือไม่ ดังนั้นควรเลือกจำเฉพาะความรู้ที่ใช้บ่อย

แม้ว่าแนวคิดนี้จะง่ายแต่หากใช้ได้ตระกับงานประยุกต์หนึ่งๆ ก็จะส่งผลให้ประสิทธิภาพ ของระบบเพิ่มขึ้นได้อย่างดี ดังเช่นที่จะแสดงในการเรียนรู้โดยการจำของโปรแกรม เชคเกอร์ (checkers) ของ Samuel เชคเกอร์² เป็นเกมที่เล่นให้เก่งยากและการพัฒนา โปรแกรมเชคเกอร์ให้เล่นแข่งชนะมนุษย์ก็ไม่ง่าย ตาเดินที่เป็นไปได้ทั้งหมดของเชคเกอร์ มีประมาณ 10^{40} ตาเดิน ทำให้การสร้างตาเดินทั้งหมดโดยโปรแกรมก่อนที่จะตัดสินใจว่าจะ เลือกตาเดินต่อไปอย่างไรไม่สามารถทำได้อย่างรวดเร็วโดยคอมพิวเตอร์ประสิทธิภาพไม่สูง นัก ดังนั้นโปรแกรมเล่นเกมประเภทนี้จะสร้างตาเดินได้เท่าที่เวลาอำนวย เช่นถ้าต้องเดิน มากกว่าใน 1 นาที โปรแกรมสร้างตาเดินที่เป็นไปได้เท่าไรก็เท่านั้น และเลือกจากตาเดิน

² เป็นเกมคล้ายกับหมากอสังหาริมทรัพย์ แต่มีจำนวนบิ๊กตี้และฝ่าย 12 ตัว

การค้นหา
ต้นไม้เกมน้อย
สุดมากสุด

ที่สร้างได้ ลักษณะของอัลกอริทึมประเกทนี้เป็นการค้นหาแบบหนึ่งซึ่งมีผู้เล่นสองฝ่ายคือ โปรแกรมกับฝ่ายตรงข้าม อัลกอริทึมที่นิยมใช้ในเกมประเกทนี้คือ **การค้นหาต้นไม้เกมน้อยสุดมากสุด (minimax game-tree search)** ตั้งแสดงในรูปที่ 6-7



รูปที่ 6-7 ต้นไม้เกมน้อยสุดมากสุด

การค้นหาต้นไม้เกมน้อยสุดมากสุดแตกต่างจากการค้นหาในปรัชญาสถานะทั่วไป ที่ต้นไม้เกมมีผู้สร้างสถานะในต้นไม้ 2 คนคือ **ผู้เล่นฝ่ายทำมากสุด (maximizing player)** โดยทั่วไปคือโปรแกรมและ **ผู้เล่นฝ่ายทำน้อยสุด (minimizing player)** หรือฝ่ายตรงข้าม ในรูปสถานะ A เป็นสถานะเริ่มต้น (แทนการจัดเรียงตัวหมากบนกระดานหนึ่งๆ) สมมติว่า A มีสถานะลูกคือ B, C และ D การสร้างสถานะลูกทำโดยการเดินหมากทุกรูปแบบที่เป็นไปได้ และผู้เล่นที่ทำหน้าที่สร้างสถานะลูกคือผู้เล่นฝ่ายทำมากสุด จากสถานะ B, C และ D ผู้เล่นฝ่ายตรงข้ามหรือผู้เล่นฝ่ายทำน้อยสุดจะสร้างสถานะลูกทั้งหมดของ B, C และ D ได้เป็น E, F, ..., K ในทางปฏิบัติโปรแกรมจะทำหน้าที่คำนวณสถานะทั้งหมดด้วยตัวเอง

ตัวเลขที่สถานะแต่ละตัวแสดงค่าความดีของสถานะนั้นๆ ค่าเหล่านี้เป็นค่าของผู้เล่นฝ่ายทำมากสุด ถ้าค่ามากแสดงว่าโอกาสชนะของผู้เล่นฝ่ายทำมากสุดมีมาก แต่ถ้าน้อยแสดงว่าผู้เล่นฝ่ายทำมากสุดมีโอกาสชนะน้อย ดังนั้นหน้าที่ของผู้เล่นฝ่ายทำมากสุดคือพยายามทำให้ตัวเลขเหล่านี้มีค่ามากโดยเลือกเส้นทางที่จะทำให้ค่าสูงสุด ตัวเลขเหล่านี้แบ่งเป็น 2 จำพวก คือ (1) ตัวเลขที่สถานะปลายต้นไม้ (ใบ) (9, -6, 0, ..., -3) และ (2) ตัวเลขที่สถานะเริ่มต้น และสถานะภายนอกต้นไม้ เราเรียกว่าตัวเลขที่ปลายต้นไม้ว่า **ค่าประเมินสถิติ (static evaluation value)** ค่าเหล่านี้เป็นค่าอิวิสติกที่วัดค่าความดีของการจัดเรียงตัวหมากบนกระดานว่าโอกาสชนะของผู้เล่นฝ่ายทำมากสุดมีมากแค่ไหน ค่าประเมินสถิตินี้วัดจากจำนวนเบี้ยของเราว่ามากกว่าของฝ่ายตรงข้ามมากน้อยแค่ไหน จำนวนกุน (king) ของเรามีมากกว่าฝ่ายตรงข้ามและมากน้อยแค่ไหน ตำแหน่งของตัวหมากของเรารอยู่ในตำแหน่งที่ได้เปรียบฝ่ายตรงข้ามมากน้อยแค่ไหน เป็นต้น

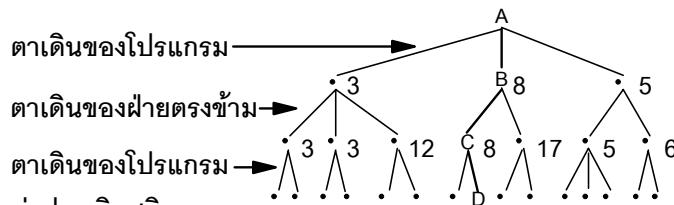
ตัวเลขที่สถานะเริ่มต้นและสถานะภายในต้นไม้เรียกว่า **ค่าแบ็คอัพ (backup value)** เป็นค่าที่ได้จากการส่งค่าประมินสติจากด้านล่างย้อนกลับขึ้นไปทางด้านบนที่ระดับ ในการคำนวณค่าแบ็คอัพนั้นจะพิจารณาเป็น 2 กรณีคือ (1) กรณีที่ผู้เล่นฝ่ายทำน้อยสุดเป็นผู้สร้างสถานะลูก ค่าแบ็คอัพของสถานะพ่อแม่จะเป็นค่าต่ำสุดในจำนวนค่าทั้งหมดของสถานะลูก (2) กรณีที่ผู้เล่นฝ่ายทำมากสุดเป็นผู้สร้างสถานะลูก ค่าแบ็คอัพของสถานะพ่อแม่จะเป็นค่าสูงสุดในจำนวนค่าทั้งหมดของลูก เช่นกรณีการคำนวณค่าแบ็คอัพของสถานะ B ซึ่งเป็นกรณีที่ (1) นั้น ค่าของ B จะเท่ากับ $\min\{9, -6, 0\} = -6$ กรณีการคำนวณค่าแบ็คอัพของสถานะ A ซึ่งเป็นกรณีที่ (2) นั้น ค่าของ A จะเท่ากับ $\max\{-6, -2, -4\} = -2$ เนื่องจากค่าในต้นไม้เป็นค่าที่แสดงโอกาสที่ผู้เล่นฝ่ายทำมากสุดมีโอกาสชนะ ดังนั้นผู้เล่นฝ่ายทำมากสุดจึงต้องพยายามทำให้ค่าที่ได้มีค่ามากสุด ส่วนผู้เล่นฝ่ายทำน้อยสุดมีหน้าที่สกัดกั้นไม่ให้ผู้เล่นฝ่ายทำมากสุดมีโอกาสชนะ ดังนั้นจึงต้องพยายามทำให้ค่าที่ได้มีค่าน้อยสุด และเป็นที่มาของชื่ออัลกอริทึมนี้

จากตัวอย่างในรูปด้านบน เมื่อผู้เล่นฝ่ายทำมากสุดจะเลือกตัดเดินก็ควรเลือกเส้นทางตามค่าแบ็คอัพ กล่าวคือเมื่อออยู่ที่สถานะ A ควรเลือกตัดเดินไปยังสถานะ C ซึ่งคาดว่าหลังจากนั้นฝ่ายผู้เล่นทำน้อยสุดน่าจะเดินไปยัง I สังเกตว่าในจำนวนสถานะทั้งหมดค่าที่มากสุดคือ 9 ของสถานะ E แต่อย่างไรก็ดีผู้เล่นฝ่ายทำมากสุดไม่มีโอกาสที่จะได้ค่าแบ็คอัพเป็น 9 ได้ แม้ว่าตนเองจะเดินจากสถานะ A ไปยัง B เพราะว่ามีผู้เล่นฝ่ายทำน้อยสุดพยายามขัดขวางให้ค่าที่ได้มีค่าน้อยสุด ถ้าผู้เล่นฝ่ายทำมากสุดเดินมายังสถานะ B ผู้เล่นฝ่ายทำน้อยสุดก็จะเดินไปยัง F ทำให้โอกาสชนะของผู้เล่นฝ่ายทำมากสุดเหลือ -6 (อย่าลืมว่าตัวเลขในต้นไม้เป็นค่าที่แสดงโอกาสชนะของผู้เล่นฝ่ายทำมากสุดเท่านั้น) ในรูปที่ 6-7 นั้นแสดงการค้นหาที่มองล่วงหน้า 2 ก้าวเดิน (2 moves look-ahead)

ค่าแบ็คอัพที่คำนวณได้ของสถานะ A นี้จะไม่เท่ากับค่าประมินสติของ A เนื่องจากว่าถ้าเราวัดค่าประมินสติก็คือการคำนวณค่าอิหริสติกของ A โดยตรงโดยดูที่ตัวมาก ณ สถานะ A แต่ค่าแบ็คอัพของ A คือการมองล่วงหน้าต่อจากนี้อีก 2 ก้าวเดินในทุกเส้นทางแล้วคำนวณเส้นทางที่น่าจะเป็นที่สุด (เส้นทางที่ผู้เล่นทั้งสองเลือกตัดเดินได้ดีที่สุด) และส่งค่าประมินสติที่ปลายต้นไม้ย้อนกลับมาที่สถานะ A ดังนั้นค่าแบ็คอัพจะมีความถูกต้องแม่นยำมากกว่าค่าประมินสติโดยตรงของ A

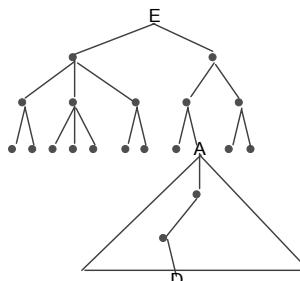
ค่าแบ็คอัพที่ได้จากการมองล่วงหน้า 2 ก้าวเดินมีความแม่นยำมากกว่าค่าประมินสติในทำนองเดียวกันค่าแบ็คอัพที่ได้จากการมองล่วงหน้า 3 ก้าวเดินก็ย่อมมีความแม่นยำมากกว่ามองล่วงหน้า 2 ก้าวเดิน ยิ่งเราเพิ่มการมองล่วงหน้าได้ลึกเท่าไร ความแม่นยำของค่าแบ็คอัพที่คำนวณได้ก็ยิ่งสูงขึ้นเท่านั้น และถ้าเราระบุมองล่วงหน้าจนถึงสถานะที่จบ

เกม ค่าที่ได้ก็จะถูกต้องสมบูรณ์ อย่างไรก็ได้ในทางปฏิบัติเราไม่สามารถมองล่วงหน้าจนจบเกมได้เนื่องจากข้อจำกัดด้านเวลา โปรแกรมจะเดินตัวหามากได้เก่งถ้าสามารถมองล่วงหน้าได้ลึกมากๆ



รูปที่ 6-8 ตัวอย่างต้นไม้เกมในการนิ่งมองล่วงหน้า 3 ก้าวเดิน

การเพิ่มความสามารถของโปรแกรมสามารถทำได้โดยการเพิ่มจำนวนก้าวเดินที่จะมองล่วงหน้า เพราะยิ่งมองล่วงหน้าได้ลึกค่าเบิกอัพก็จะถูกต้องมากขึ้น และดังเช่นที่กล่าวแล้ว ว่าจากข้อจำกัดเรื่องเวลาที่โปรแกรมสามารถใช้ได้ การกระจายสถานะเพิ่มขึ้นจึงไม่สามารถทำได้ แต่อย่างไรก็ได้โดยการใช้การเรียนรู้โดยการจำจะสามารถเพิ่มความสามารถของโปรแกรมให้เสมือนกับว่าโปรแกรมมองล่วงหน้าได้มากขึ้น เราใช้การเรียนรู้โดยการจำเพื่อที่จะเก็บคู่ลำดับค่าเบิกอัพของสถานะเริ่มต้น เช่นในรูปที่ 6-8 หลังจากที่เราได้ค้นหาล่วงหน้า 3 ก้าวเดินและพบว่าค่าเบิกอัพของ A เท่ากับ 8 และ เราจะจำคู่ลำดับ [A,8] ไว้ในหน่วยความจำ เมื่อ A ถูกพบอีกครั้งที่ปลายของต้นไม้เกมต้นนี้ในการเล่นครั้งใหม่ เราจะไม่ต้องหาค่าประเมินสกิดของ A แต่จะนำค่าเบิกอัพของ A มาใช้แทน การนำเอาค่าเบิกอัพมาใช้แทนที่จะคำนวนค่าประเมินสกิดนั้น นอกจากจะมีข้อดีที่รวดเร็วขึ้นเนื่องจากการคำนวนค่าประเมินสกิดจะใช้เวลานานกว่าแล้ว ยังส่งผลดีอีกประการที่สำคัญดังแสดงในรูปที่ 6-9 ซึ่งแสดงต้นไม้เกมต้นหนึ่งที่มี E เป็นสถานะแรกและมีสถานะที่ปลายต้นไม้สีสถานะหนึ่งคือ A



รูปที่ 6-9 การเรียนรู้โดยการจำเพิ่มประสิทธิภาพของการค้นหา

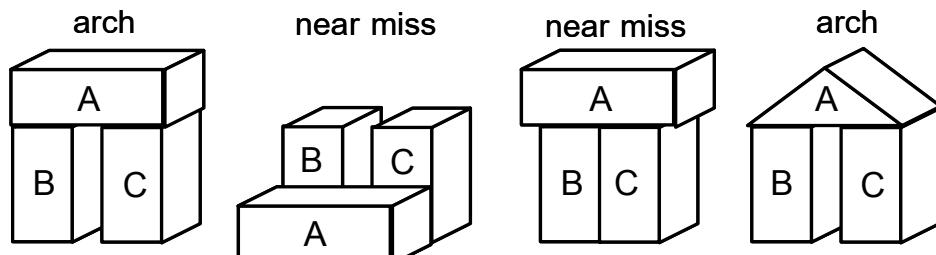
ด้วยการใช้ค่าเบ็คอัพของ A แทนที่จะใช้ค่าประมีนสกิตก์สมือนกับว่าที่จุด A นี้ได้ร่วมการคันหาอีก 3 ก้าวเดินล่วงหน้าเข้าไว้ด้วย ดังนั้นที่ E แม้ว่าด้วยข้อจำกัดทางเวลาทำให้เราคันหาได้เพียง 3 ก้าวเดินล่วงหน้า แต่ก็สมือนกับว่าในเส้นทางที่รวม A จะเป็นการคันหาล่วงหน้าถึง 6 ก้าวเดินล่วงหน้า และด้วยการจำคู่ลำดับระหว่างสถานะกับค่าเบ็คอัพไว้จำนวนมากก็จะทำให้การคันหาเพิ่มจำนวนก้าวเดินล่วงหน้าเป็น 3, 6, 9, ... ตามลำดับ ซึ่งส่งผลให้ประสิทธิภาพของโปรแกรมเพิ่มขึ้น

ตัวอย่างนี้แสดงให้เห็นการประยุกต์ใช้การเรียนรู้โดยการจำที่ส่งผลให้ประสิทธิภาพของงานที่กระทำได้ขึ้นอย่างชัดเจน และโปรแกรมเรียนรู้การเล่นเกมเชกเกอร์นี้ยังมีการจัดการหน่วยความจำอย่างประยุกต์โดยจัดเก็บเฉพาะตำแหน่งตัวมากบนกระดานของผู้เล่นฝ่ายที่มากสุดฝ่ายเดียว และเมื่อจะใช้กับผู้เล่นฝ่ายท่านอื่นอยู่สุดกีลับตำแหน่งของตัวมากกลับด้านกันเท่านั้น นอกจากนั้นยังมีการทำดัชนีเพื่อdingตำแหน่งตัวมากบนกระดานให้ได้อย่างรวดเร็วโดยใช้คุณสมบัติของกระดาน เช่น จำนวนตัวมาก การมีหรือไม่มีชุน เพื่อใช้เป็นดัชนี และยังได้จัดการปัญหาความสมดุลย์ระหว่างการจัดเก็บกับการคำนวณใหม่โดยใช้วิธีที่เรียกว่าการแทนที่ตัวที่ถูกใช้น้อยสุด (least recently used replacement) วิธีนี้พยายามจะไม่จัดเก็บคู่ลำดับให้มากเกินไปเพื่อจะทำให้การคันคืนคู่ลำดับใช้เวลามาก โดยกำหนดจำนวนคู่ลำดับที่จะจำเป็นค่าคงที่ค่าหนึ่ง เช่น 100,000 คู่ลำดับ จากนั้นคู่ลำดับใดที่ถูกใช้น้อยสุด (เมื่อจำไว้แล้วถูกลบในตันไม้เกมอื่นน้อยสุด) จะถูกลบออกจากหน่วยความจำแล้วแทนที่ด้วยคู่ลำดับใหม่ตัวอื่น วิธีนี้ทำโดยกำหนดอายุให้กับคู่ลำดับแต่ละคู่และทุกครั้งที่คู่ลำดับถูกเรียกมาใช้อยู่ของมันจะลดลงครึ่งหนึ่ง และทุกครั้งที่มีการจำคู่ลำดับใหม่ อายุของคู่ลำดับอื่นทุกตัวในหน่วยความจำจะถูกบวกเพิ่ม 1 หน่วย จากนั้นตัวที่มีอายุมากสุดจะถูกลบออกจากหน่วยความจำ

6.3 การเรียนรู้โดยการวิเคราะห์ความแตกต่าง

การเรียนรู้โดยการวิเคราะห์ความแตกต่าง (*learning by analyzing differences*) ถูกพัฒนาโดย Winston ในปีค. 1975 [Winston, 1992] แม้ว่าจะเป็นวิธีการเรียนรู้ที่ค่อนข้างเก่ามากแล้วก็ตาม แต่ว่าแนวคิดต่างๆ สามารถนำไปใช้ในการเรียนรู้แบบใหม่ๆ ได้อย่างดี ในที่นี้จึงยกวิธีการเรียนรู้แบบนี้มาเพื่อศึกษาแนวคิดของการเรียนรู้เชิงอุปนัย การเรียนรู้โดยการวิเคราะห์ความแตกต่างนี้ใช้เรียนรู้ในทัศน์ทางโครงสร้าง (*structural concept*) ในโดเมนปัญหาโลกของบล็อก เช่น arch, tent หรือ house เป็นต้น วิธีการเรียนรู้นี้จะวิเคราะห์ความแตกต่างที่ปรากฏในลำดับของตัวอย่างที่ผู้สอนป้อนให้ โดยตัวอย่างสอน (*training example*) มี 2 ประเภทคือ **ตัวอย่างบวก (positive example)** และ **ตัวอย่างลบ (negative example)** ตัวอย่างบวกคือตัวอย่างที่ถูกต้องของมโนทัศน์ (*concept*) ที่สอน เช่นจะสอน house ตัวอย่างบวกก็จะเป็นบ้านหลังที่หนึ่ง บ้านหลังที่สอง เป็นต้น ตัวอย่างลบคือตัวอย่างที่ไม่ถูกต้อง เช่นจะสอน house ตัวอย่างลบก็จะเป็นเดันท์หลังที่หนึ่ง โรงเรียนหลังที่หนึ่งเหล่านี้เป็นต้น

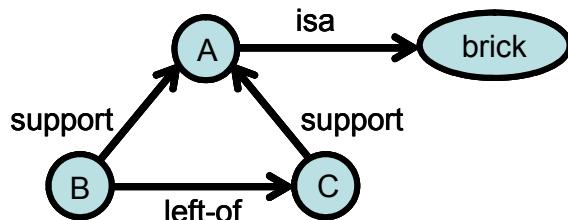
สำหรับการเรียนรู้โดยการวิเคราะห์ความแตกต่างนี้ ตัวอย่างลบที่ผู้สอนให้จะต้องเป็นตัวอย่างลบแบบที่เรียกว่า **พลาดน้อย (near miss)** กล่าวคือตัวอย่างลบแบบพลาดน้อยนี้จะต่างจากตัวอย่างบวกเพียงเล็กน้อย เช่นจะสอน house ตัวอย่างลบแบบพลาดน้อยก็จะเป็นบ้านที่ขาดประตู หรือบ้านที่ไม่มีหลังคา เป็นต้น การเรียนรู้แบบนี้ผู้สอนจะจัดเตรียมลำดับของตัวอย่างไว้ค่อนข้างดีเพื่อให้โปรแกรมเรียนรู้สามารถวิเคราะห์ความต่างของตัวอย่างบวกกับตัวอย่างลบแบบพลาดน้อย ด้านล่างนี้ยกตัวอย่างการเรียนรู้แบบ arch ซึ่งมีลำดับของตัวอย่างที่จะสอนดังแสดงในรูปที่ 6-10



รูปที่ 6-10 ตัวอย่างบวกและตัวอย่างลบแบบพลาดน้อยของ arch

ในรูป 'arch' และ 'near miss' หมายถึงตัวอย่างบวกและตัวอย่างลบแบบพลาดนโยบาย ตามลำดับ จากตัวอย่างที่ให้ทั้งสี่ตัวนี้ โปรแกรมจะเรียนรู้ว่าอะไรคือ arch เมื่อเราดูตัวอย่างข้างต้น เราจะพอยเข้าใจได้ว่าตัวอย่างบวกตัวแรกบอกว่า arch คือสิ่งที่ประกอบด้วยอิฐ (brick) แนวตั้ง 2 ก้อนและอิฐแนวอน 1 ก้อนที่ถูกองรับด้วยอิฐแนวตั้ง ตัวอย่างที่สอง อธิบายสิ่งที่ไม่ใช่ arch ว่าคือสิ่งที่ประกอบด้วยอิฐแนวตั้ง 2 ก้อนและอิฐแนวอนซึ่งไม่ถูกองรับด้วยอิฐแนวตั้ง ตัวอย่างที่ 3 และ 4 แสดงตัวอย่างของ arch และสิ่งที่ไม่ใช่ตามลำดับ โปรแกรมเรียนรู้นี้ใช้การแทนความรู้เพื่อแสดงมโนทัศน์ในรูปของ **ช่ายงานความหมาย (semantic network)** การแทนความรู้แบบนี้จะประกอบด้วยบัพ (node) และเส้นเชื่อม (link) บัพแสดงวัตถุและเส้นเชื่อมแทนความสัมพันธ์ระหว่างวัตถุในโดเมนนั้น

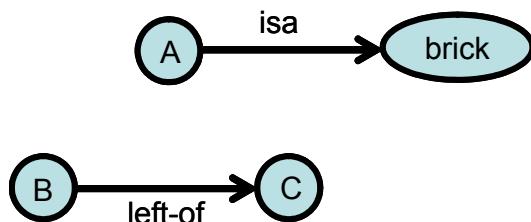
จากตัวอย่างบวกตัวที่หนึ่ง โปรแกรมจะสร้างคำอธิบายเริ่มต้น (initial description) ของ มโนทัศน์ดังรูปที่ 6-11



รูปที่ 6-11 คำอธิบายเริ่มต้น

บัพ A มีเส้นเชื่อม isa แสดงความสัมพันธ์ว่า A เป็น brick และเส้นเชื่อมจาก B และ C ไป A คือ support แสดงความสัมพันธ์ว่า B และ C รองรับ A และมีเส้นเชื่อม left-of แสดงว่า B อยู่ด้านซ้ายของ C ส่วนเส้นเชื่อมอื่นๆ ที่ไม่เกี่ยวข้องโดยตรงกับ concept ของไว้ในที่นี่ เช่นเส้นเชื่อม isa จาก B ไปยังบัพ brick เป็นต้น

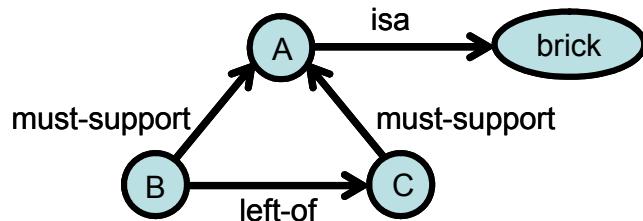
จากตัวอย่างลบแบบพลาดนโยบายตัวที่สอง โปรแกรมสร้างคำอธิบายของตัวอย่างลบได้ดังรูปที่ 6-12



รูปที่ 6-12 คำอธิบายของตัวอย่างตัวที่สอง

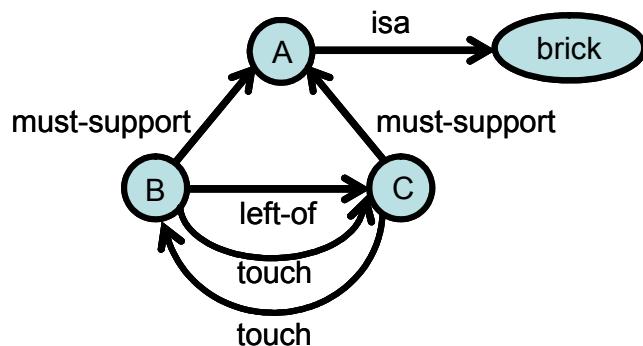
การแจง
จำเพาะของ
มโนทัศน์

ที่จุดนี้โปรแกรมจะวิเคราะห์หาความแตกต่างของตัวอย่างที่ถูกกับที่ผิดโดยการจับคู่บันทึกและเส้นเชื่อม และพบว่าเส้นเชื่อม support ซึ่งต่างกันในตัวอย่างทั้งสองจำเป็นสำหรับมโนทัศน์ arch โปรแกรมจึงใส่เงื่อนไขเพิ่มเข้าไปในคำอธิบายในรูปที่ 6-11 โดยใช้เส้นเชื่อมใหม่ชื่อ must-support แทนที่เส้นเชื่อมเดิมดังแสดงในรูปที่ 6-13 เราเรียกคำอธิบายใหม่ที่ได้นี้ว่า **โมเดลระหว่างวิวัฒนาการ (evolving model)** ในกรณีนี้ตัวอย่างลบให้ข้อมูลสำหรับการใช้อิริสติกเส้นเชื่อมจำเป็น (require-link heuristic) ที่ใส่เงื่อนไขที่มากขึ้นในเส้นเชื่อมเดิม เราเรียกการทำเช่นนี้ว่าเป็นอิริสติกแบบหนึ่งเนื่องจากว่าเป็นการคาดคะเนจากเหตุผลของความแตกต่างระหว่างตัวอย่างที่น่าจะเป็น แต่ก็อาจไม่ถูกต้องเสมอไปก็เป็นได้ ในกรณีนี้ตัวอย่างลบแบบพลาดน้อยเป็นตัวอย่างที่ให้ข้อมูลสำหรับ **การแจงจำเพาะของมโนทัศน์ (specialization of concept)** ซึ่งหมายถึงว่าคำอธิบายของมโนทัศน์จะถูกทำให้แคบลง มีเงื่อนไขมากขึ้น ตรงกับตัวอย่างจำนวนน้อยลง



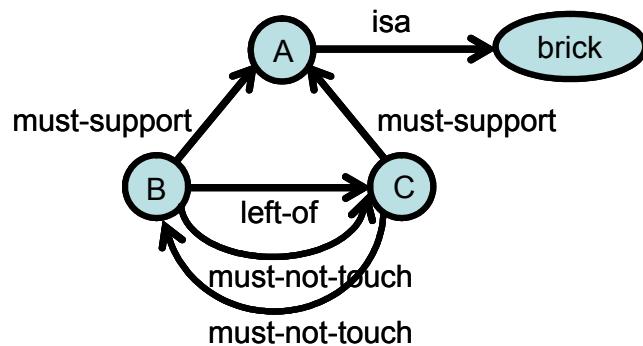
รูปที่ 6-13 โมเดลระหว่างวิวัฒนาการ

จากตัวอย่างลบตัวที่สาม โปรแกรมสร้างคำอธิบายของตัวอย่างลบได้ดังรูปที่ 6-14



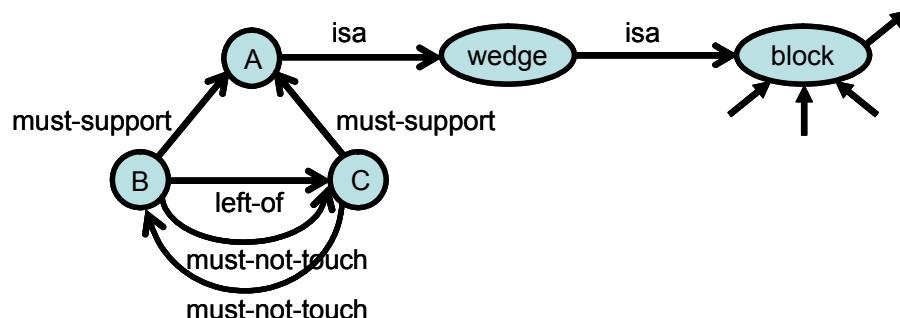
รูปที่ 6-14 คำอธิบายของตัวอย่างลบตัวที่สาม

โปรแกรมหาความแตกต่างระหว่างคำอธิบายของตัวอย่างที่สามกับโมเดล พบว่ามีเส้นเชื่อม touch อยู่ในตัวอย่างลงซึ่งไม่มีในโมเดล ดังนั้นโปรแกรมจึงเพิ่มเส้นเชื่อมเข้าไปในโมเดลและปรับโมเดลใหม่ได้ดัง [รูปที่ 6-15](#) ในการนี้ตัวอย่างลงให้ข้อมูลสำหรับ **อิวาริสติก เส้นเชื่อมห้าม (forbid-link heuristic)**



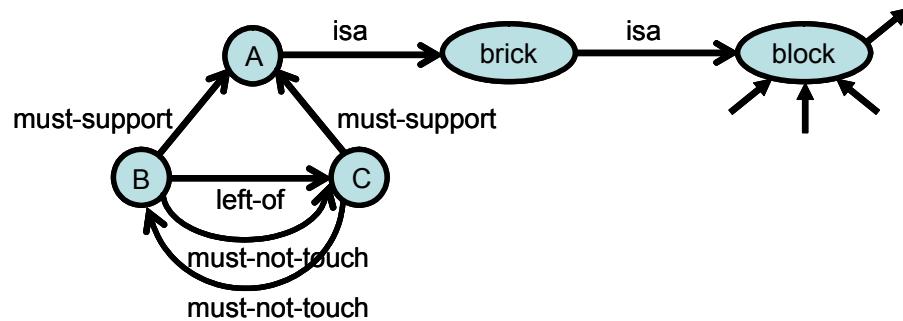
รูปที่ 6-15 โมเดลหลังรับตัวอย่างตัวที่สาม

จากตัวอย่างบวกตัวที่สี่ โปรแกรมสร้างคำอธิบายของตัวอย่างได้ดัง [รูปที่ 6-16](#)



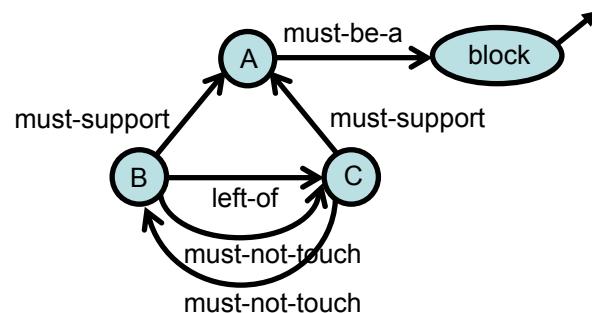
รูปที่ 6-16 คำอธิบายของตัวอย่างบวกตัวที่สี่

สมมติว่าเรามีความสัมพันธ์ของต้นไม้จำแนกประเภท (classification tree) ว่าวัตถุหนึ่งๆ จดอยู่ในประเภทอะไรในฐานความรู้ของเราด้วย เช่นในที่นี่ wedge จัดเป็นวัตถุหนึ่งในประเภทของ block ในท่านองเดียวกัน brick ก็จัดเป็นวัตถุหนึ่งในประเภทของ block ด้วย โมเดลของเรานำไปดูใน [รูปที่ 6-15](#) เมื่อนำมาเขียนใหม่ให้รวมความสัมพันธ์ของต้นไม้จำแนกประเภทเข้าไปด้วยก็จะได้ดัง [รูปที่ 6-17](#)



รูปที่ 6-17 โมเดลหลังรับตัวอย่างตัวที่สามที่รวมตันไม่จำแนกประเภทด้วย

เมื่อนำโมเดลเบรี่ยนที่ยึดกับคำอธิบายตัวอย่างที่สี่ข้างต้นจะพบว่ามีความแตกต่างกันที่ brick กับ wedge และทั้งคู่ต่างก็เป็นวัตถุในประเภทของ block ดังนั้นเราจึงแทนที่ brick ด้วยประเภทที่สูง (กว้าง) กว่าคือ block ได้เป็นโมเดลใน [รูปที่ 6-18](#) เราเรียกอิริสติกแบบนี้ว่า **อิริสติกปีนต้นไม้** (*climb-tree heuristic*)



รูปที่ 6-18 โมเดลเมื่อรับตัวอย่างครบถ้วนทุกตัว

ในกรณีที่เราไม่มีตันไม่จำแนกประเภท โปรแกรมจะสร้างประเภทใหม่คือ “brick-or-wedge” ขึ้นมาเพื่อใช้แทนบัพ block ใน [รูปที่ 6-18](#) เราเรียกอิริสติกแบบนี้ว่า **อิริสติกขยายเซต** (*enlarge-set heuristic*) และถ้าหากว่าในกรณีที่เรามีวัตถุอื่นอยู่ในโดเมนนี้อีกเลยที่นอกเหนือจาก brick และ wedge เราถึงสามารถตัดเส้นเชื่อม isa ออกได้เลยเพื่อเป็นการลดเงื่อนไข และในกรณีนี้เราระบุว่า **อิริสติกตัดเส้นเชื่อม** (*drop-link heuristic*) ในกรณีเหล่านี้ตัวอย่างบางทำหน้าที่สำหรับการวางแผนทั่วไปของมโนทัศน์ (*generalization of concept*) ซึ่งหมายถึงว่าคำอธิบายของมโนทัศน์จะถูกทำให้กว้างขึ้น มีเงื่อนไขน้อยลง ตรงกับตัวอย่างจำนวนมากขึ้น

อัลกอริทึมของโปรแกรมเรียนรู้โดยวิเคราะห์ความแตกต่างแสดงใน [ตารางที่ 6-10](#)

ตารางที่ 6–10 อัลกอริทึมการเรียนรู้โดยวิเคราะห์ความแตกต่าง

Algorithm: Learning by Analyzing Differences

- Near-miss is for specialize model by using
 - require-link heuristic
 - forbid-link heuristic
- Positive example is for generalize mode by using
 - climb-tree heuristic
 - enlarge-set heuristic
 - drop-link heuristic

Specialization algorithm

Specialization to make a model more restrictive by:

- (1) Match the evolving model to the example to establish correspondences among parts.
- (2) Determine whether there is a single, most important difference between the evolving model and the near miss.
 - If there is a single, most important difference,
 - (a) If the evolving model has a link that is not in the near miss, use the require-link heuristic
 - (b) If the near miss has a link that is not in the model, use the forbid-link heuristic
 - otherwise, ignore the example.

Generalization algorithm

Generalization to make a model more permissive by:

- (1) Match the evolving model to the example to establish correspondences among parts.
- (2) For each difference, determine the difference type:
 - If a link points to a class in the evolving model different from the class to which the link points in the example,
 - (a) If the classes are part of a classification tree, use the climb-tree heuristic
 - (b) If the classes form an exhaustive set, use the drop-link heuristic
 - (c) Otherwise, use the enlarge-set heuristic
 - (3) If a link is missing in the example, use the drop-link heuristic
 - (4) Otherwise, ignore the difference.

6.4 เวอร์ชันสเปช

เวอร์ชันสเปช (*version space*) [Mitchell, 1977] เรียนรู้คำอธิบายที่อธิบายตัวอย่างบวกและไม่อธิบายตัวอย่างลบ รูปที่ 6-19 ด้านล่างแสดงตัวอย่างของการเรียนรู้ในทัศน์ car ซึ่งใช้การแทนความรู้แบบกรอบ (*frame*)

Car023
origin: Japan
manufacturer: Honda
color: Blue
decade: 1970
type: Economy

รูปที่ 6-19 ตัวอย่างบวกของในทัศน์ car

กรอบประกอบด้วยชื่อกรอบ ในที่นี่คือ Car023 และสล็อต (slot) ในที่นี่สล็อตมี 5 ตัวคือ origin, manufacturer, color, decade และ type ซึ่งแสดงคุณสมบัติทั้งห้าอย่างของรถยนต์ สมมติว่าสล็อตแต่ละตัวมีค่าที่เป็นไปได้ตามตารางที่ 6-11 ด้านล่างนี้

ตารางที่ 6-11 ค่าที่เป็นไปได้ของสล็อตแต่ละตัว

origin	= {Japan, USA, Britain, Germany, Italy}
manufacturer	= {Honda, Toyota, Ford, Chrysler, Jaguar, BMW, Fiat}
color	= {Blue, Green, Red, White}
decade	= {1950, 1960, 1970, 1980, 1990, 2000}
type	= {Economy, Luxury, Sports}

การเรียนรู้โดยเวอร์ชันสเปชจะแสดงคำอธิบายในรูปของสล็อตและค่าของสล็อต เช่นถ้าเป็นมโนทัศน์ “Japanese economy car” จะแสดงได้ดังรูปที่ 6-20 โดยที่ x1, x2 และ x3 เป็นตัวแปรสามารถถูกแทนด้วยค่าคงที่ได้ๆ

origin:	Japan
manufacturer:	x1
color:	x2
decade:	x3
type:	Economy

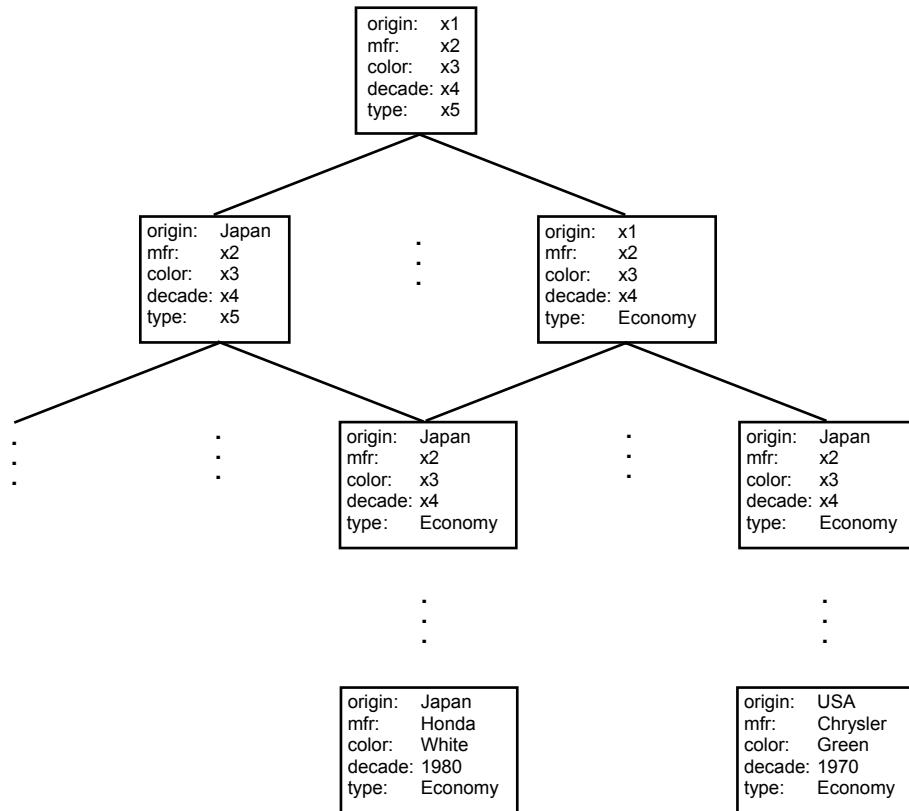
รูปที่ 6-20 มโนทัศน์ “Japanese economy car”

สอดคล้องกับ

มีนัยทั่วไปกว่า
และ
จำเพาะกว่า

ปัญหาการเรียนรู้ที่เราสนใจคือ กำหนดค่าที่เป็นไปได้ของสิ่งตัวอย่างบางและตัวอย่างลบให้ จงหาคำอธิบายมโนทัศน์ที่ **สอดคล้องกับ (consistent with)** ตัวอย่าง (อธิบายตัวอย่างบางและไม่อธิบายตัวอย่างลบ)

วิธีการเรียนรู้เวอร์ชันสเปชน์มองว่าการเรียนรู้คือการค้นหาในปริภูมิค้นหาที่เรียกว่า **ปริภูมิมโนทัศน์ (concept space)** ซึ่งเป็นปริภูมิที่มีสมาชิกแต่ละตัวเป็นคำอธิบายในรูปของกรอบโดยที่สมาชิกเหล่านี้มีลำดับบางส่วน (partial ordering) ในลำดับนี้สมาชิกตัวที่ **มีนัยทั่วไปกว่า (more general)** จะอยู่ด้านบนของสมาชิกตัวที่ **จำเพาะกว่า (more specific)** ดังแสดงในรูปที่ 6-21 โดยที่ตัวอักษรเล็ก (x_1, x_2, x_3, x_4 และ x_5) แสดงตัวแปรซึ่งสามารถแทนที่ด้วยค่าคงที่ได้ ส่วนตัวอักษรใหญ่และตัวเลข (เช่น Japan, Economy, 1980) แสดงค่าคงที่



รูปที่ 6-21 ปริภูมิมโนทัศน์

ตัวที่อยู่บนสุดในรูปแสดง **มโนทัศน์มีนัยทั่วไปสุด (most general concept)** ส่วนตัวที่อยู่ล่างสุดแสดง **มโนทัศน์จำเพาะสุด (most specific concept)** ซึ่งเป็นตัวอย่างหนึ่งๆ และตัวที่

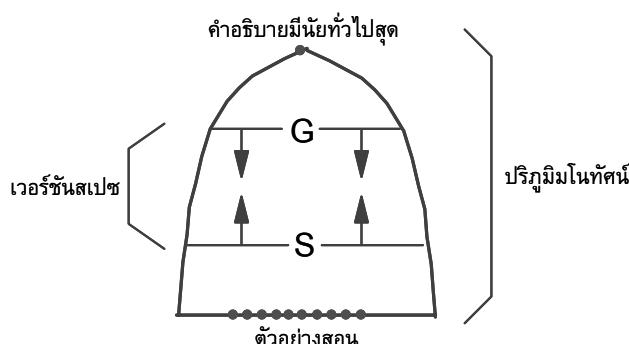
เป็นคำอธิบายในทัศน์ป่าหมาย (*target concept description*) จะอยู่ระหว่างบนสุดกับล่างสุด วิธีการเรียนรู้เวอร์ชันสเปชคือการสร้างเซตย่อยประกอบด้วย **สมมติฐาน** (*hypothesis*) ที่อยู่ในปริภูมิมโนทัศน์ที่สอดคล้องกับตัวอย่างสอน และเรียกเซตย่อยนี้ว่า **เวอร์ชันสเปช** (*version space*)

เวอร์ชันสเปชที่สร้างขึ้นนี้จะต้องประกอบด้วยสมมติฐาน (คำอธิบาย) ที่สอดคล้องกับตัวอย่างที่เคยพบมากทั้งหมด วิธีการสร้างเวอร์ชันสเปชที่ทำได้คือการแจงสมาชิกทุกตัวในปริภูมิมโนทัศน์ แล้วตรวจสอบกับตัวอย่างสอนทุกตัวที่รับเข้ามา หากสมาชิกตัวใดไม่สอดคล้องกับตัวอย่างก็ตัดทิ้งไป คงไว้เฉพาะตัวที่สอดคล้องเท่านั้น อย่างไรก็ตามปริภูมิมโนทัศน์มีขนาดใหญ่มาก วิธีการนี้จึงไม่มีประสิทธิภาพ ตัวอย่างเช่นในกรณีของปัญหาใน **รูปที่ 6-21** เมื่อพิจารณาค่าที่เป็นไปได้ในสล็อตแต่ละตัวตาม **ตารางที่ 6-11** จะเห็นว่าปริภูมิมโนทัศน์มีขนาดเท่ากับ $((5+1)(7+1)(4+1)(6+1)(3+1)) = 6,720$ และในกรณีที่จำนวนสล็อตมีมากขึ้นเช่น 10 ตัว และสล็อตแต่ละตัวมีค่าที่เป็นไปได้มากขึ้นเช่น 10 ค่า จะได้ว่าปริภูมิมโนทัศน์จะยิ่งมีขนาดใหญ่ขึ้นมาก ($\approx 2.6 \times 10^{10}$)

วิธีการเรียนรู้เวอร์ชันสเปชจะใช้วิธีการแทนสเปชด้วยวิธีที่ประยุกต์และมีประสิทธิภาพในการค้นหากลายโดยจะใช้เซตย่อย 2 เซตเรียกว่าเซต G และเซต S

- เซต G ประกอบด้วยคำอธิบายมีนัยทั่วไปสุดที่ยังสอดคล้องกับตัวอย่างที่เคยพบมากทั้งหมด
- เซต S ประกอบด้วยคำอธิบายจำเพาะสุดที่ยังสอดคล้องกับตัวอย่างที่เคยพบมากทั้งหมด

เวอร์ชันสเปชจะอยู่ระหว่างเซต G กับ S ดังแสดงใน **รูปที่ 6-22**



รูปที่ 6-22 เวอร์ชันสเปช

หลักการของเวอร์ชันสเปชคือทุกครั้งที่เราได้รับตัวอย่างบางตัวใหม่เราจะทำให้ S มีนัยทั่วไป (general) มากขึ้นและทุกครั้งที่ได้รับตัวอย่างลบเราจะทำให้ G จำเพาะ

(specific) มากขึ้น จนในที่สุด S และ G ลู่เข้าสู่ค่าเดียวกันที่เป็นคำอธิบายในทัศน์เป้าหมาย อัลกอริทึมการเรียนรู้ของเวอร์ชันสเปชเป็นดังตารางที่ 6-12 นี้

ตารางที่ 6-12 อัลกอริทึมการเรียนรู้เวอร์ชันสเปช

Algorithm: Version-Space-Candidate-Elimination

```

1. G := {most general description}
2. S := {first positive example}
3. Accept a new example E
   IF E is positive THEN
     • Remove from G any descriptions that do not cover
       the example.
     • Update S to contain the most specific set of
       descriptions in the version space that cover the
       example and the current elements of S.
   ELSE IF E is negative THEN
     • Remove from S any descriptions that cover the
       example.
     • Update G to contain the most general set of
       descriptions in the version space that do not
       cover the example.
4. IF S and G are both singleton sets and S = G THEN
   Output the element
 ELSE IF S and G are both singleton sets and S<>G THEN
   examples were inconsistent
 ELSE goto 3.

```

6.4.1 ตัวอย่างการเรียนรู้มโนทัศน์ car

กำหนดเซตตัวอย่างสอนที่ประกอบด้วยตัวอย่างบวกและตัวอย่างลบดังรูปที่ 6-23
อัลกอริทึมในตารางที่ 6-12 จะเรียนรู้ดังต่อไปนี้

origin:	Japan
mfr:	Honda
color:	Blue
decade:	1980
type:	Economy

(+)

origin:	Japan
mfr:	Toyota
color:	Green
decade:	1970
type:	Sports

(-)

origin:	Japan
mfr:	Toyota
color:	Blue
decade:	1990
type:	Economy

(+)

origin:	USA
mfr:	Chrysler
color:	Red
decade:	1980
type:	Economy

(-)

origin:	Japan
mfr:	Honda
color:	White
decade:	1980
type:	Economy

(+)

รูปที่ 6-23 ตัวอย่างสอนของมโนทัศน์ car

- จากตัวอย่างบวก 3 ตัวและตัวอย่างลบ 2 ตัวตามรูปด้านบน เราเริ่มด้วยการสร้าง G และ S ตามตัวอย่างแรกได้

$$G = \{(x1, x2, x3, x4, x5)\}$$

$$S = \{(Japan, Honda, Blue, 1980, Economy)\}$$

โดยที่ $(x1, x2, x3, x4, x5)$ เป็นค่าของสิ้นสุดที่ 1, 2, 3, 4, 5 ตามลำดับ

- ตัวอย่างที่ 2 เป็นตัวอย่างลบ ดังนั้นเราทำการแจงจำเพาะของ G เพื่อไม่ให้เวอร์ชันคู่มุ่ง

สเปชอธิบายหรือ **คดุณ (cover)** ตัวอย่างลบนี้โดยการเปลี่ยนตัวแปรให้เป็นค่าคงที่

$$G = \{(x1, Honda, x3, x4, x5), (x1, x2, Blue, x4, x5),$$

$$(x1, x2, x3, 1980, x5), (x1, x2, x3, x4, Economy)\}$$

$$S \text{ ไม่เปลี่ยนแปลง} = \{(Japan, Honda, Blue, 1980, Economy)\}$$

- ตัวอย่างที่ 3 เป็นบวก = $(Japan, Toyota, Blue, 1990, Economy)$ เราจำจัดคำอธิบายใน G ที่ไม่สอดคล้องกับตัวอย่างนี้

$$G = \{(x1, x2, Blue, x4, x5), (x1, x2, x3, x4, Economy)\}$$

และทำการวางแผนทั่วไปของ S ให้รวมตัวอย่างนี้

$$S = \{(Japan, x2, Blue, x4, Economy)\}$$

ที่จุดนี้เราได้เวอร์ชันสเปชที่แสดง "Japanese blue economy car", "blue car" หรือ "Economy car"

- ตัวอย่างที่ 4 เป็นลบ = $(USA, Chrysler, Red, 1980, Economy)$

$$G = \{(x1, x2, Blue, x4, x5), (x1, x2, Blue, x4, Economy),$$

$$(Japan, x2, x3, x4, Economy)\}$$

$$S = \{(Japan, x2, Blue, x4, Economy)\}$$

- ตัวอย่างที่ 5 เป็นบวก = $(Japan, Honda, White, 1980, Economy)$

$$G = \{(Japan, x2, x3, x4, Economy)\}$$

$$S = \{(Japan, x2, x3, x4, Economy)\}$$

ที่จุดนี้ได้คำตอบ $S=G$ และแสดง "Japanese economy car"

6.4.2 ข้อจำกัดของเวอร์ชันสเปช

ดังที่แสดงในตัวอย่างด้านบนนี้ เวอร์ชันสเปชสามารถเรียนรู้ได้จากตัวอย่างที่สอน อย่างไรก็ได้เวอร์ชันสเปชก็ยังมีข้อจำกัดดังต่อไปนี้

- อัลกอริทึมเรียนรู้นี้เป็นแบบทำน้อยสุด (least-commitment algorithm) กล่าวคือในแต่ละขั้นตอนเวอร์ชันสเปชจะถูกตัดเลิมให้เล็กลงน้อยที่สุดเท่าที่เป็นไปได้ ดังนั้นถึงแม้ว่าตัวอย่างบางทุกตัวเป็น Japanese cars ก็ตาม อัลกอริทึมก็จะไม่ตัดความน่าจะเป็นที่มีโน้ทศน์อ้างจะรวม car อื่นๆ ทั้งจะกระทั้งพบตัวอย่างลง ซึ่งหมายถึง เวอร์ชันสเปชจะเรียนรู้ไม่สำเร็จถ้าไม่มีตัวอย่างลงเลย
- กระบวนการค้นหาเป็นการค้นหาแนววิวัังแบบทั่วหมด (exhaustive breadth-first search) ซึ่งเห็นได้จากการปรับค่าของเซ็ต G ที่จะทดลองทำที่สล็อตทุกตัวให้ได้ทุกแบบที่เป็นไปได้ ดังนั้นทำให้อัลกอริทึมมีประสิทธิภาพต่ำในการนี้ที่ปริภูมิใหญ่มากๆ ซึ่งอาจทำให้ได้ขึ้นโดยใช้อิริสติกเข้าช่วยในการค้นหาโดยลองเปลี่ยนตัวแปรเป็นค่าคงที่ในบางสล็อตที่น่าจะนำไปสู่คำตอบก่อน เป็นต้น
- เซ็ต S ประกอบด้วยสมาชิกเพียงตัวเดียว เพราะว่าตัวอย่างมาก 2 ตัวใดๆ มีการวางแผนที่ไม่เพียงหนึ่งเดียว ดังนั้นเวอร์ชันสเปชจึงไม่สามารถเรียนรู้โน้ทศน์แบบ 'หรือ' (disjunctive concept) ซึ่งเป็นมโน้ทศน์ที่อยู่ในรูปของ or เช่น "Japanese economy car or Japanese sport car"
- ข้อจำกัดอีกอย่างของเวอร์ชันสเปชคือไม่สามารถจัดการกับตัวอย่างมีสัญญาณรบกวน (noisy example) ซึ่งเป็นตัวอย่างที่มีข้อมูลบางส่วนผิดพลาด เช่นถ้าตัวอย่างตัวที่ 3 ในรูปที่ 6-23 (Japan Toyota Blue 1990 Economy) เราให้ประเภทผิดเป็นตัวอย่างลง (-) อัลกอริทึมจะไม่สามารถเรียนรู้โน้ทศน์ "Japanese economy car" ได้ถูกต้อง

6.5 การเรียนรู้ต้นไม้ตัดสินใจ

การเรียนรู้ต้นไม้ตัดสินใจ (decision tree learning) [Quinlan, 1986; Quinlan, 1993] เป็นการเรียนรู้ที่ใช้การแทนความรู้อยู่ในรูปของต้นไม้ตัดสินใจ ใช้สำหรับจำแนกประเภทของตัวอย่าง วิธีการเรียนรู้คล้ายกับการเรียนรู้เวอร์ชันสเปชโดยเริ่มจากการป้อนตัวอย่างเข้าไปในระบบ ซึ่งตัวอย่างที่ป้อนให้เป็นตัวอย่างบวกกับตัวอย่างลบก็ได้และนอกจากนั้นเรายังสามารถป้อนตัวอย่างที่มากกว่า 2 ประเภท (class) ได้ กล่าวคือแทนที่จะมีแต่บวกกับลบ ก็สามารถมีได้หลายประเภท เช่นในการรู้จำตัวอักษร จะมีตัวอย่างมาจากหลายประเภทที่แตกต่างกันคือประเภท 'ก', ประเภท 'ข', ประเภท 'ค', ประเภท 'ง' ฯลฯ แต่เพื่อให้ง่ายต่อการอธิบาย ตัวอย่างที่จะยกให้ดูต่อไปนี้จะมีเพียง 2 ประเภทเท่านั้น โดยเราจะใช้ปัญหาการฝึกัดเป็นตัวอย่างอธิบาย

ปัญหาการฝึกัด : เราไปเที่ยวที่ชายทะเลและพบว่าคนที่ไปฝึกัดตามชายทะเล บางคน ก็จะมีผิวเปลี่ยนเป็นสีแทน แต่บางคนต้องได้รับความทรมานจากผิวใหม่ เราต้องการหาว่า อะไรคือปัจจัยที่ทำให้คนที่ไปฝึกัดตามชายทะเลแล้วผิวใหม่หรือไม่ใหม่ โดยข้อมูลที่สังเกตได้ประกอบด้วยความแตกต่างของสีผิว หน้าหนัง ลักษณะของผู้ที่ไปฝึกัด และการใช้โลชัน ซึ่งบางคนก็ใช้โลชัน บางคนก็ไม่ใช้

สมมติว่าเรารับข้อมูลของตัวอย่างสอนได้ตามตารางที่ 6-13 เพื่อใช้สร้างต้นไม้ตัดสินใจ

ตารางที่ 6-13 ตัวอย่างสอนที่สังเกตได้

↓
ประเภท

คุณสมบัติ →

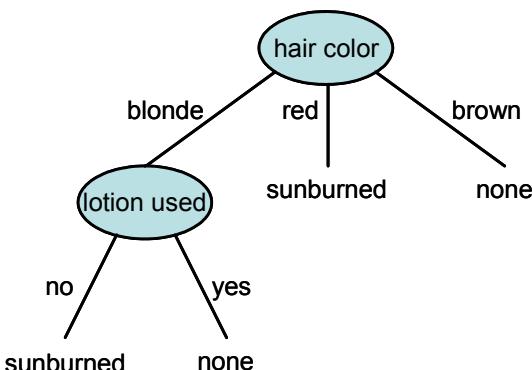
ค่า

Name	Hair	Height	Weight	Lotion	Result
Sarah	blonde	average	light	no	sunburned
Dana	blonde	tall	average	yes	none
Alex	brown	short	average	yes	none
Annie	blonde	short	average	no	sunburned
Emily	red	average	heavy	no	sunburned
Pete	brown	tall	heavy	no	none
John	brown	average	heavy	no	none
Katie	blonde	short	light	Yes	none

แຄוแรกสุดในตารางแสดง **คุณสมบัติ (attribute)** ของข้อมูลซึ่งประกอบด้วยชื่อ (Name) สีผม (Hair) ส่วนสูง (Height) น้ำหนัก (Weight) และการใช้โลชัน (Lotion) ส่วนสมบัติสุดท้ายแทนประเภทของตัวอย่าง คุณสมบัติ Name ไว้สำหรับบังอิงตัวอย่างและไม่มีผลต่อการจำแนกข้อมูล เราจึงจะไม่ใช้ Name ในการเรียนรู้ด้านล่างนี้ แต่ละแຄวในตารางนอกเหนือจากแຄวแรกแทนตัวอย่างหนึ่งตัว เช่นแຄวที่สองแสดงตัวอย่างของคนที่ชื่อ Sarah ซึ่งมีสีผม ส่วนสูง น้ำหนัก และการใช้โลชัน เป็น blond, average, light และ no ตามลำดับ ตัวอย่างนี้อยู่ในประเภท sunburned เป็นต้น

เมื่อเราได้ข้อมูลตัวอย่างทั้ง 8 ตัวแล้ว ลิสที่เราต้องการทำก็คือทำการวางแผนทั่วไปของตัวอย่างเพื่อสร้างเป็นโมเดลสำหรับทำนายประเภทของข้อมูลของคนอื่นที่ไม่ได้บันทึกไว้ วิธีที่ง่ายสุดก็คือการเรียนรู้โดยการจำ และเมื่อมีตัวอย่างในอนาคตที่เรายังไม่ทราบประเภทและต้องการทำการทำนาย เรา ก็นำตัวอย่างนั้นมาเปรียบเทียบกับตัวอย่างสอนในตาราง ถ้าตัวอย่างที่นำมาเปรียบเทียบมีคุณสมบัติตรงกับข้อมูลในตาราง เรา ก็นำประเภทของตัวอย่างสอนที่ตรงกันทำนายให้กับตัวอย่างนั้น อย่างไรก็ได้วิธีการนี้ทำงานได้ไม่ดีนักเนื่องจากว่าโอกาสที่เราจะพบตัวอย่างทดสอบที่ตรงกับตัวอย่างสอนมีน้อย สมมติว่าสีผมมีค่าที่เป็นไปได้ทั้งหมด 3 ค่าคือ blonde, brown, red ส่วนสูงมีได้ 3 ค่าคือ tall, average, short น้ำหนักมีได้ 3 ค่าคือ heavy, average, light และการใช้โลชันมีได้ 2 ค่าคือ yes, no เราจะพบว่าความน่าจะเป็นที่ตัวอย่างทดสอบจะตรงกับตัวอย่างสอนมีค่าเท่ากับ $8/(3 \times 3 \times 3 \times 2) = 15\%$ (สมมติว่าความน่าจะเป็นที่ค่าแต่ละค่าสำหรับคุณสมบัติหนึ่ง ๆ มีความน่าจะเป็นที่จะเกิดขึ้นเท่ากัน)

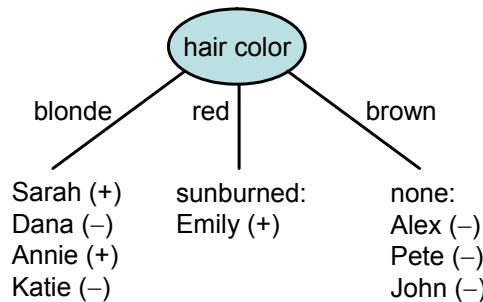
การเรียนรู้ต้นไม้ตัดสินใจจะทำการวางแผนทั่วไปของข้อมูลโดยสร้างเป็นโมเดลอยู่ในรูปต้นไม้ตัดสินใจ ตัวอย่างของต้นไม้ตัดสินใจแสดงใน [รูปที่ 6-24](#)



รูปที่ 6-24 ตัวอย่างของต้นไม้ตัดสินใจ

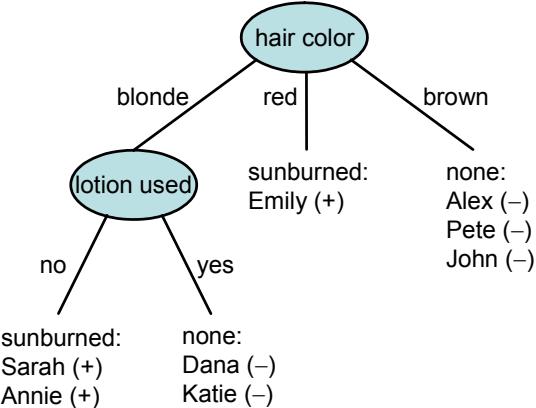
ต้นไม้ตัดสินใจประกอบด้วย **บัพ (node)** และ **กิ้ง (link)** ที่ต่อ กับ บัพ บัพที่ปลายสุดเรียกว่า **บัพใบ (leaf node)** หรือเรียกย่อๆ ว่า ใบ บัพแสดงคุณสมบัติและกิ้งแสดงค่าของคุณสมบัตินั้น ใบ (leaf) แสดงประเภท การสร้างต้นไม้ตัดสินใจทำโดยสร้างบัพที่จะบัพเพื่อตรวจสอบคุณสมบัติของตัวอย่าง แล้วแยกตัวอย่างลงตามค่าของกิ้ง ทำงานจะหั่งตัวอย่างในใบแต่ละใบอยู่ในประเภทเดียวกันทั้งหมด

สมมติว่า เราเลือกคุณสมบัติ hair color เป็นบัพแรกหรือบัพแรกของต้นไม้ เราจะแยกตัวอย่างลงตามกิ่งของบัพ hair color ตัวอย่างใดที่มีค่าของ hair color เป็น blonde ก็แยกลงตามกิ่งซ้าย ถ้าเป็น red ก็แยกลงตามกิ่งกลาง และถ้าเป็น brown ก็แยกลงตามกิ่งขวา ผลที่ได้แสดงใน [รูปที่ 6-25](#) เครื่องหมาย + และ - แสดงประเภท sunburned และ none ตามลำดับ



รูปที่ 6-25 ผลการแยกตัวอย่างลงตามกิ่งของบัพ 'hair color'

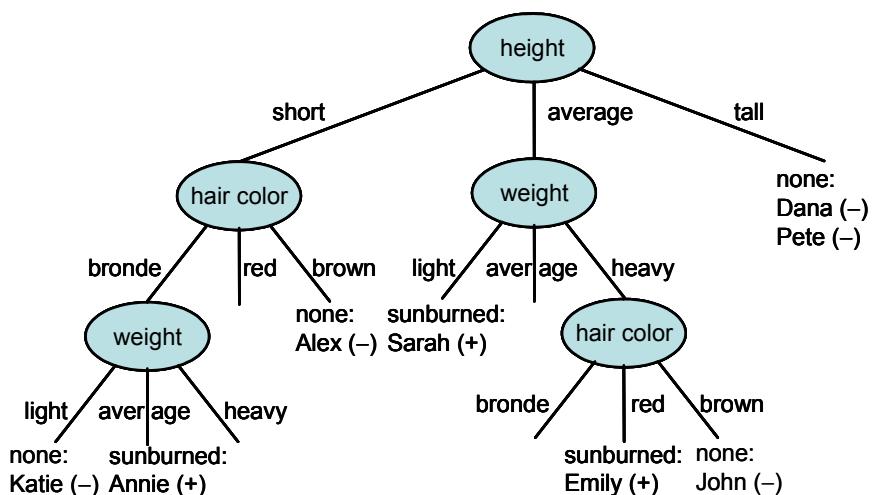
ต้นไม้ที่สร้างขึ้นนี้แยกตัวอย่างได้ในกรณีที่ hair color เป็น red (ตัวอย่างทุกตัวมีประเภทเป็น sunburned) และ brown (ตัวอย่างทุกตัวมีประเภทเป็น none) แต่ในกรณีที่ hair color เป็น blonde ยังแยกตัวอย่างไม่ได้ กล่าวคือมีตัวอย่างที่เป็นหั่ง sunburned และ none ปะปนกันอยู่ ในกรณีที่ hair color มีค่าเป็น brown เราสรุปได้ว่า ตัวอย่างทุกตัวจะมีประเภทเป็น none หมดเนื่องจากตัวอย่างสอนทุกตัวที่ลักษณะเป็นน้ำ หรือในกรณีที่ hair color มีค่าเป็น red เรา ก็สรุปได้ในทำนองเดียวกันว่า ตัวอย่างจะมีประเภทเป็น sunburned แต่ในกรณีของ hair color เป็น blonde เราต้องการคุณสมบัติอื่นเข้าช่วยจำแนกประเภทตัวอย่าง ต่อไป ที่จุดนี้ สมมติว่า เราใช้คุณสมบัติ lotion เพื่อแยกข้อมูลในกิ่งของ blonde ต่อไป ผลที่ได้แสดงใน [รูปที่ 6-26](#)



รูปที่ 6-26 ต้นไม้ตัดสินใจที่สอดคล้องกับตัวอย่างสอน

ต้นไม้ตัดสินใจใน [รูปที่ 6-26](#) ด้านบนนี้สอดคล้องกับตัวอย่างสอนทุกด้าน หมายความว่า ถ้านำตัวอย่างสอนมาตรวจสอบด้วยต้นไม้ตัดสินใจ ต้นไม้จะทำนายประเภทได้ถูกต้องทุกด้าน การตรวจสอบทำโดยดูว่าตัวอย่างมี hair color เป็นค่าอะไร ถ้าเป็น brown จะทำนายประเภทเป็น none ถ้าเป็น red จะทำนายประเภทเป็น sunburned แต่ถ้าเป็น blonde จะดู lotion used ด้วยว่าถ้าเป็น no แสดงว่าประเภทเป็น sunburned แต่ถ้าเป็น yes แสดงว่าประเภทเป็น none

โดยทั่วไปต้นไม้ตัดสินใจที่สอดคล้องกับตัวอย่างสอนมีได้มากกว่า 1 ต้น เช่น ต้นไม้ใน [รูปที่ 6-27](#) ก็เป็นต้นไม้อีกดันหนึ่งที่สอดคล้องกับตัวอย่าง



รูปที่ 6-27 ต้นไม้ตัดสินใจอีกดันหนึ่งที่มีความซับซ้อนมากกว่าต้นแรก

เมื่อเราพิจารณาต้นไม้ต้นแรกในรูปที่ 6-26 และต้นที่สองในรูปที่ 6-27 เรายังว่าต้นไม้ต้นแรกน่าจะถูกต้องมากกว่าต้นที่สอง เนื่องจากว่าในต้นแรกนั้นใช้คุณสมบัติสีผุ้มและใช้โลชันในการจำแนกข้อมูล ซึ่งน่าจะเป็นไปได้ เพราะสีผุ้มมีความสัมพันธ์อย่างมากกับความแข็งแรงของผิวเรา คนที่มีผุ้มสีน้ำตาลน่าจะมีผิวที่แข็งแรงไปดีกว่าคนที่มีผุ้มสีขาว แล้วก็จะไม่เป็นอะไรส่วนผุ้มสีแดงมีผิวอบอุ่น และผุ้มสีบรอนซ์มีผิวปานกลางซึ่งจะขึ้นกับการใช้โลชันหรือไม่ใช้ ถ้าใช้ไปดีกว่าเดดกีดจะไม่เป็นอะไร ถ้าไม่ใช้ไปดีกว่าเดดแล้วผิวจะใหม่ ส่วนต้นไม้ต้นที่สองเรายังไม่สามารถอธิบายได้ว่าทำไส่ส่วนสูงที่ใช้เป็นบัพراكหรือหนังที่บัพในระดับถัดมาจึงมีความสำคัญต่อการที่ผิวจะใหม่หรือไม่ใหม่

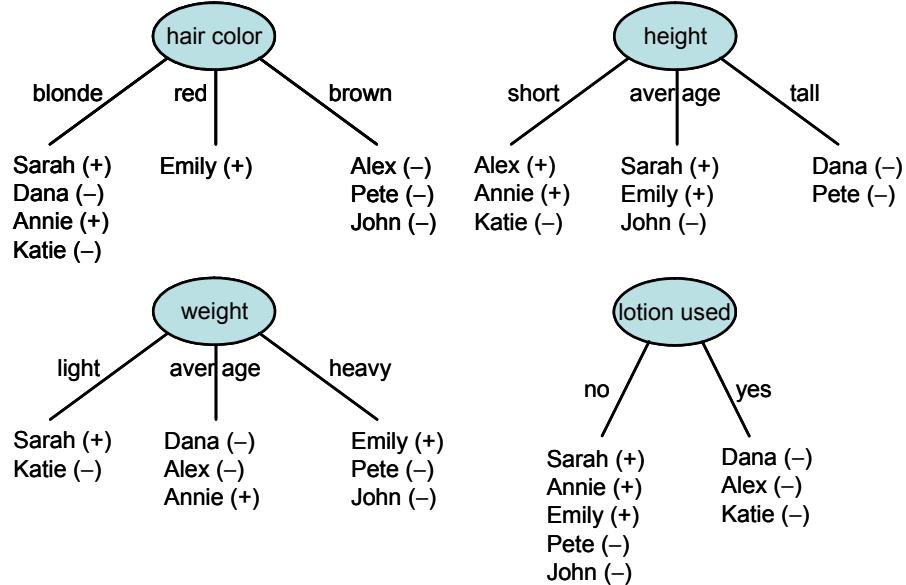
ความแตกต่างที่เห็นได้เด่นชัดอีกประการของต้นไม้ทั้งสองคือจำนวนบัพภายในต้นไม้จะเห็นได้ว่าจำนวนบัพของต้นไม้ต้นที่สองมีจำนวนมากกว่า หรือกล่าวอีกนัยหนึ่งคือต้นไม้ต้นที่สองมีความซับซ้อนมากกว่า หรือกล่าวได้ว่าต้นไม้ต้นแรกมีขนาดเล็กกว่าต้นไม้ต้นที่สองโดยที่ขนาดตัวจากจำนวนบัพภายในต้นไม้

มีดโกนของ
อ็อกแคม

ในการเรียนรู้ของเครื่องนั้น เรายังมีอิทธิสติกตัวหนึ่งที่นิยมใช้กันและพบว่าทำงานได้ดีกว่าดีในหลายกรณีเรียกว่า **มีดโกนของอ็อกแคม (occam's razor)** เมื่อเรานำมีดโกนของอ็อกแคมมาใช้ในการเลือกต้นไม้ตัดสินใจ เรายังจะได้ว่า “ต้นไม้ตัดสินใจที่ดีที่สุด” อย่างไรก็ได้ถ้าเราจะหาต้นไม้ตัดสินใจที่มีขนาดเล็กที่สุดที่สอดคล้องกับตัวอย่างสอนคือต้นไม้ตัดสินใจที่ดีที่สุด” อย่างไรก็ได้ถ้าเราจะหาต้นไม้ตัดสินใจที่มีขนาดเล็กที่สุดที่สอดคล้องกับตัวอย่างสอนก็ไม่สามารถทำได้โดยง่าย เราต้องสร้างต้นไม้ตัดสินใจจำนวนมาก โดยเริ่มจากต้นไม้ที่มีจำนวนบัพ 1 บัพทุกต้นที่เป็นไปได้แล้วดูว่ามีต้นไหนหรือไม่ที่สอดคล้องกับตัวอย่างสอน ถ้าไม่มีก็เพิ่มจำนวนบัพเป็น 2 บัพ ทำอย่างนี้ไปจนกระทั่งพบต้นไม้ตัดสินใจที่สอดคล้องกับตัวอย่าง เราพบว่าวิธีการนี้จะมีจำนวนต้นไม้ที่ต้องสร้างเป็นพังก์ชันเลขยกกำลังของจำนวนคุณสมบัติซึ่งไม่เหมาะสมกับการใช้งานจริง

6.5.1 พังก์ชันเกนสำหรับการเลือกบัพทดสอบ

ส่วนนี้จะกล่าวถึงวิธีการเลือกบัพเพื่อสร้างต้นไม้ตัดสินใจโดยใช้หลักการว่า เนื่องจากจุดมุ่งหมายของการสร้างต้นไม้คือเพื่อจำแนกประเภทของข้อมูลเพื่อให้ตัวอย่างในแต่ละบัพใบอยู่ในประเภทเดียวกันทั้งหมด ดังนั้นบัพที่ดีควรเป็นบัพที่แยกตัวอย่างออกเป็นชุดย่อยตามกิ่งของบัพนั้นและเซตย่อยในแต่ละกิ่งประกอบด้วยสมาชิกที่ส่วนใหญ่เป็นประเภทเดียวกันมากที่สุด ตัวอย่างในรูปที่ 6-28 แสดงผลของบัพทดสอบแต่ละบัพ



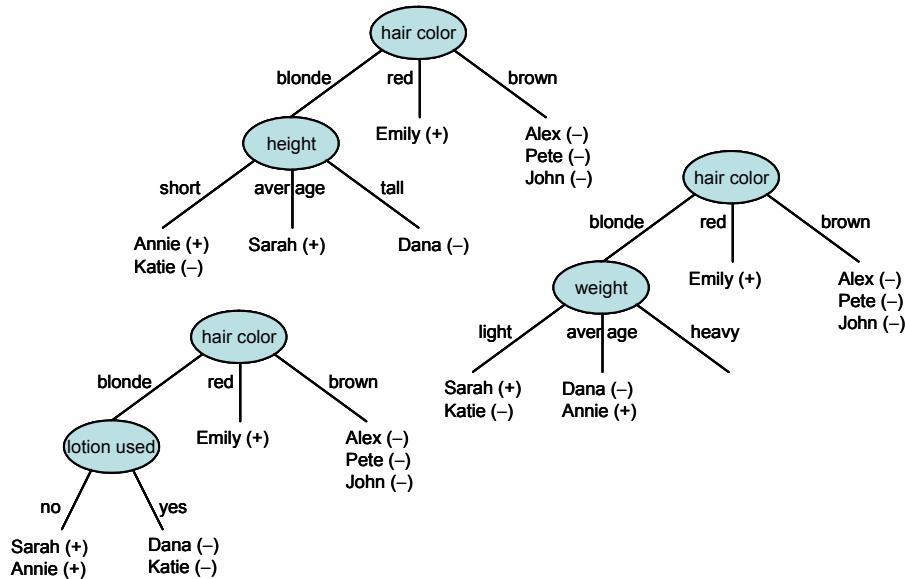
รูปที่ 6-28 ผลของบัพทดสอบแต่ละบัพในการแยกตัวอย่างเพื่อเลือกบัพแรก

ตั้งแสดงในรูปด้านบน บัพแต่ละบัพแยกตัวอย่างได้ต่างกันดังนี้

- ในกรณีของบัพทดสอบเป็น hair color สามารถแยกตัวอย่างเป็น 3 เชตย้อย เชตย้อยแรก (blonde) มีตัวอย่างของ 2 ประเภทปนกันอยู่ ส่วนเชตย้อยที่ 2 (red) และ 3 (brown) มีตัวอย่างของประเภท sunburned และ none อยู่อย่างเดียว ตามลำดับ ซึ่งกรณีนี้ hair color แยกตัวอย่างได้ดีเมื่อเทียบกับบัพอื่นด้านล่างนี้
- ในกรณีของบัพทดสอบเป็น height สามารถแยกตัวอย่างเป็น 3 เชตย้อย เชตย้อยแรก (short) และเชตย้อยที่ 2 (average) มีตัวอย่าง 2 ประเภทปนกันอยู่ในแต่ละ เชต ส่วนเชตย้อยที่ 3 (tall) มีตัวอย่างของ none อยู่อย่างเดียว จะเห็นว่ากรณีนี้ แยกตัวอย่างไม่ดีเท่ากรณีของ hair color
- ในกรณีของบัพทดสอบเป็น weight สามารถแยกตัวอย่างเป็น 3 เชตย้อย เชตย้อยทั้งสามเชต (light, average, heavy) ต่างก็มีตัวอย่าง 2 ประเภทปนกันอยู่ ซึ่งกรณีนี้เป็นกรณีที่แยกที่สุด
- ในกรณีของบัพทดสอบเป็น lotion used สามารถแยกตัวอย่างเป็น 2 เชตย้อย เชตย้อยแรก (no) มีตัวอย่างของ 2 ประเภทปนกัน ส่วนเชตย้อยที่ 2 (yes) มีตัวอย่างของ none อยู่อย่างเดียว และในเชตย้อยแรกสามารถส่วนใหญ่ของเชตนี้เป็น sunburned (+) เกือบทั้งหมด ซึ่งเมื่อเทียบกับกรณีแรกของบัพ hair color ถือได้ว่ามีความสามารถในการแยกตัวอย่างได้ใกล้เคียงกัน

ดังจะเห็นได้ในตัวอย่างด้านบนนี้ เรายังคงเปรียบเทียบได้ว่าบัพหนึ่งๆ มีความสามารถในการแยกตัวอย่างเด็กว่าบัพอีกบัพหนึ่งหรือไม่ แต่ในบางกรณี เช่น hair color กับ lotion used เราอาจพบว่าความแตกต่างไม่ได้ ดังนั้นเราจำเป็นต้องทำการวัดที่สามารถบอกความต่างได้อย่างชัดเจนโดยการนิยามพังก์ชันเพื่อวัดประสิทธิภาพของบัพ ออกเป็นค่าที่วัดได้อย่างละเอียด ซึ่งจะกล่าวต่อไป

ณ จุดนี้ เพื่อให้เข้าใจถึงการสร้างต้นไม้ตัดสินใจของสมมติว่าเรามีฟังก์ชันน้อยๆ และสมมติว่าระหว่างบัพ hair color กับ lotion used ค่าฟังก์ชันของ hair color ดีกว่า และได้รับเลือกเป็นบัพแรก ในขั้นตอนต่อไปก็คือเราต้องพิจารณาต่อว่าในแต่ละกิ่งของบัพแรก มีกี่ไดหรือไม่ที่ยังมีตัวอย่างจากหอยประเทกประปนกันอยู่ ถ้ามีเราต้องเพิ่มบัพของคุณสมบัติอื่นเพื่อช่วยแยกตัวอย่างที่ยังประปนกันอยู่ต่อไป ในกรณีของบัพ hair color ในรูปที่ 6-28 กิ่ง blonde เท่านั้นที่ยังมีตัวอย่างจากหอยประเทกปนกัน เรายังจำเป็นต้องเพิ่มบัพต่อไปโดยทดลงเพิ่มคุณสมบัติที่เหลือทั้งสาม (height, weight และ lotion used) ผลที่ได้แสดงในรูปที่ 6-29



รูปที่ 6-29 ผลของบัพทดสอบแต่ละบัพในการแยกตัวอย่างเพื่อเลือกบัพต่อจากกิง blonde

จากรูปด้านบนจะเห็นได้ว่าบัพ lotion used เป็นบัพที่แยกตัวอย่างออกเป็นเซตย่อยโดยที่แต่ละเซตย่อยมีสมาชิกอยู่ในประเภทเดียวกัน ดังนั้นบัพ lotion used ถูกเลือกในขั้นตอนนี้

6.5.2 พังก์ชันเกน

พังก์ชันที่ใช้วัดความสามารถในการแยกตัวอย่างของบัพทดสอบที่มีประสิทธิภาพมาก พังก์ชันหนึ่งคือ **พังก์ชันเกน (Gain function)** พังก์ชันเกนนี้ใช้ในการตัดสินใจเลือกคุณสมบัติ ที่จะใช้เป็นรากหรือบัพในต้นไม้โดยการคำนวณค่าเกนของคุณสมบัติแต่ละตัวเมื่อทดลองใช้ คุณสมบัตินั้นแบ่งตัวอย่าง แล้วเลือกคุณสมบัติที่มีค่าเกนสูงที่สุดมาเป็นรากหรือบัพ ค่าเกนนี้คำนวณได้โดยใช้ความรู้จาก**ทฤษฎีสารสนเทศ (information theory)** ซึ่งมีสาระสำคัญคือ ค่าสารสนเทศหรือของข้อมูลขึ้นอยู่กับค่าความน่าจะเป็นของข้อมูลซึ่งสามารถวัดอยู่ในรูป ของบิต (bits) จากสูตร

$$\text{ค่าสารสนเทศของข้อมูล} = - \log_2 (\text{ความน่าจะเป็นของข้อมูล}) \quad (6.2)$$

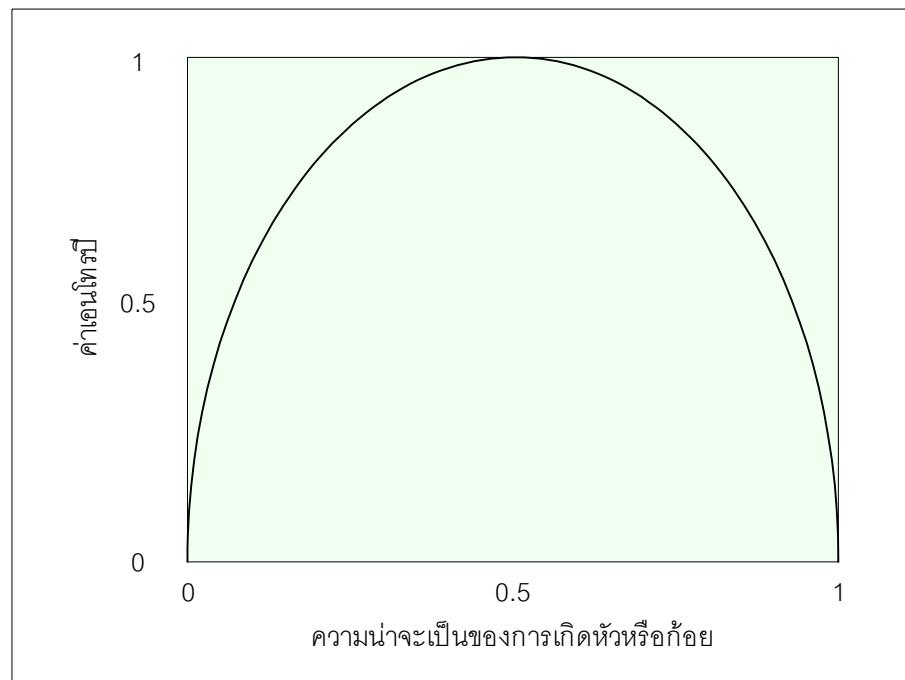
ถ้าให้ชุดของข้อมูล M ประกอบด้วยค่าที่เป็นไปได้ คือ $\{m_1, m_2, \dots, m_n\}$ และให้ความน่าจะเป็นที่จะเกิดค่า m_i มีค่าเท่ากับ $P(m_i)$ จะได้ว่าค่า**เอนโทรปี (entropy)** ของ M ซึ่งใช้วัดค่าสารสนเทศโดยเฉลี่ยเพื่อระบุประเภทของข้อมูลสามารถเขียนแทนด้วย $I(M)$ คำนวณได้จากสูตร

$$I(M) = \sum_i^n -P(m_i) \log_2 P(m_i) \quad (6.3)$$

ตัวอย่างเช่นในการโยนหัวโยนก้อย ชุดข้อมูล M จะประกอบด้วยค่าที่เป็นไปได้คือ {หัว, ก้อย} และถ้าให้ความน่าจะเป็นที่ออกหัวเท่ากับ $P(\text{หัว})$ และความน่าจะเป็นที่ออกก้อยเท่ากับ $P(\text{ก้อย})$ ดังนั้นค่าเอนโทรปีของการโยนหัวโยนก้อย จะคำนวณได้จากสูตร

$$I(\text{การโยนหัวโยนก้อย}) = -P(\text{หัว}) \log_2 (P(\text{หัว})) - P(\text{ก้อย}) \log_2 (P(\text{ก้อย})) \quad (6.4)$$

เมื่อความน่าจะเป็นของการเกิดหัวหรือก้อยมีค่าต่าง ๆ กันจะสามารถคำนวณค่าเอนโทรปีของการโยนหัวโยนก้อยได้ต่าง ๆ กันดังรูปที่ 6-30 จะเห็นได้ว่าเมื่อออกหัวหมดหรือก้อยหมด ค่าเอนโทรปีจะเป็น 0 และค่าเอนโทรปีจะค่อย ๆ เพิ่มขึ้นจนสูงที่สุดเมื่อความน่าจะเป็นของการเกิดหัวเท่ากับความน่าจะเป็นของการเกิดก้อย แสดงให้เห็นว่าค่าเอนโทรปีที่น้อยจะบ่งบอกว่าข้อมูลชุดนั้นมีความแตกต่างกันน้อยหรือเกือบจะเป็นพวกลดียกัน และถ้าค่าเอนโทรปีสูงจะบ่งบอกว่าข้อมูลชุดนั้นมีความแตกต่างกันมากหรือประกอบด้วยตัวอย่างหลายพวกที่มีจำนวนใกล้เคียงกัน



รูปที่ 6-30 ค่าเงินโทรศัพท์ของการโยนหัวโยนก้อย

ในการเลือกคุณสมบัติที่จะมาเป็นบัพรากจะอาศัยค่าเกณ ซึ่งคำนวณจากค่าเออนໂໂຣປ ทั้งหมดของชุดข้อมูลนั้นแลบด้วยค่าเออนໂໂຣປหลังจากเลือกคุณสมบัติได้คุณสมบัติหนึ่งเป็น แรก ค่าเออนໂໂຣປหลังจากแบ่งตามคุณสมบัติที่เลือกแล้วคำนวณได้จาก ค่าผลรวมของผล คุณระหว่างค่าเออนໂໂຣປของแต่ละบัพกับอัตราส่วนของตัวอย่างในแต่ละกิ่ง ต่อตัวอย่าง ทั้งหมดที่บันทึก

ถ้าให้ข้อมูลสอนคือ T และคุณสมบัติที่เป็นบัพคือ X และมีค่าทั้งหมดที่เป็นไปได้ n ค่าบัพปุจจุบันจะแบ่งตัวอย่าง T ออกตามกิ่งเป็น $\{t_1, t_2, \dots, t_n\}$ ตามค่าที่เป็นไปได้ของ X ดังนั้นจึงสามารถคำนวณค่าเออนໂໂทรเป็นหลังจากแบ่งตามคุณสมบัติ X ดังนี้

$$I_x(T) = \sum_{i=1}^n \frac{|t_i|}{|T|} I(t_i) \quad (6.5)$$

ค่าเงินของคุณสมบัติ X ที่ใช้แบ่งข้อมูลที่บันทึกไว้ สามารถคำนวณได้จากการลบค่าเงินโทรศัพท์ทั้งหมดที่บันทึกกับค่าเงินโทรศัพท์ที่ได้หลังจากแบ่งด้วยคุณสมบัติ X ดังนี้

$$Gain(X) = I(T) - I_x(T) \quad (6.6)$$

พิจารณาคุณสมบัติ hair color ซึ่งแบ่งแยกข้อมูลได้ดัง [รูปที่ 6-25](#) ในการนี้ที่ใช้ hair color เป็นบัพراك เราคาคำนวณหาค่าเกนได้ดังนี้

$$\begin{aligned} Gain(hair color) &= \left[-\left(\frac{3}{8}\right) \log_2\left(\frac{3}{8}\right) - \left(\frac{5}{8}\right) \log_2\left(\frac{5}{8}\right) \right] - \\ &\quad \left[\frac{4}{8} \left(-\left(\frac{2}{4}\right) \log_2\left(\frac{2}{4}\right) - \left(\frac{2}{4}\right) \log_2\left(\frac{2}{4}\right) \right) + \frac{1}{8} \left(-\left(\frac{1}{1}\right) \log_2\left(\frac{1}{1}\right) \right) + \frac{3}{8} \left(-\left(\frac{3}{3}\right) \log_2\left(\frac{3}{3}\right) \right) \right] \\ &= 0.45 \end{aligned}$$

ในทำนองเดียวกัน คุณสมบัติอื่นจะมีค่ามาตรฐานเกนเป็นดังต่อไปนี้

$$Gain(\text{height}) = 0.26 \quad Gain(\text{weight}) = 0.01 \quad Gain(\text{lotion}) = 0.34$$

จึงเลือกคุณสมบัติ hair color มาเป็นบัพแรกของต้นไม้ตัดสินใจ แต่คุณสมบัตินี้เพียงอย่างเดียวไม่สามารถแยกตัวอย่างบวกและลบออกจากกันได้ในกึ่งของค่าคุณสมบัติ blonde จึงต้องพิจารณาคุณสมบัติอื่นเพื่อแบ่งแยกข้อมูลที่ตกลงมายังกึ่งนี้ ([ดูรูปที่ 6-29](#) ประกอบ) โดยค่าพังก์ชันแกนของคุณสมบัติแต่ละตัวมีค่าดังนี้

$$Gain(\text{height}) = 0.5 \quad Gain(\text{weight}) = 0.0 \quad Gain(\text{lotion}) = 1.0$$

เราใช้คุณสมบัติ lotion ซึ่งมีค่าเกนมากสุดมาแบ่งแยกข้อมูลต่อไป ซึ่งพบว่าเมื่อแบ่งแยกแล้วข้อมูลที่ผ่านการแบ่งแยกมีกลุ่มเดียวกัน จึงได้ต้นไม้ตัดสินใจดัง [รูปที่ 6-24](#)

6.5.3 การเปลี่ยนต้นไม้เป็นกฎ

ระบบปัญญาประดิษฐ์ส่วนใหญ่ใช้การแทนความรู้ในรูปของกฎ ดังนั้นเมื่อเราสร้างต้นไม้ตัดสินใจแล้วเราสามารถเปลี่ยนต้นไม้ให้อยู่ในรูปของกฎเพื่อใช้กับในกรณีที่ระบบของเรายังการแทนความรู้ของกฎเป็นหลัก วิธีการแปลงต้นไม้เป็นกฎ "IF THEN" ทำได้โดยแสดงทุกเส้นทางเริ่มต้นจากบัพراكไปยังบัพใบและทุกรุ้งที่พบบัพทดสอบก็ให้เพิ่มบัพทดสอบกับค่าของทดสอบไว้ในส่วนของ IF และเมื่อพบบัพใบก็ให้สีประเภทไว้ในส่วนของ THEN จากต้นไม้ [รูปที่ 6-24](#) เราเปลี่ยนเป็นกฎได้ดังนี้

- (1) IF the person's hair color is blonde AND
the person uses lotion
THEN nothing happens
- (2) IF the person's hair color is blonde AND
the person uses no lotion
THEN the person turns red
- (3) IF the person's hair color is red

THEN the person turns red

(4) IF the person's hair color is brown

THEN nothing happens

ตารางด้านล่างแสดงการเปรียบเทียบการใช้เครื่องมือการเรียนรู้ (learning tools) และไม่ใช้เครื่องมือในการพัฒนาระบบผู้เชี่ยวชาญ (expert system) GASOIL และ BMT เป็นระบบผู้เชี่ยวชาญที่สร้างโดยเครื่องมือการเรียนรู้แบบต้นไม้ตัดสินใจซึ่งพัฒนาโดยใช้แนวคิดของการเรียนรู้ต้นไม้ตัดสินใจ

ตารางที่ 6-14 เปรียบเทียบระบบผู้เชี่ยวชาญที่ใช้และไม่ใช้การเรียนรู้ของเครื่องเพื่อช่วยในการพัฒนาระบบ

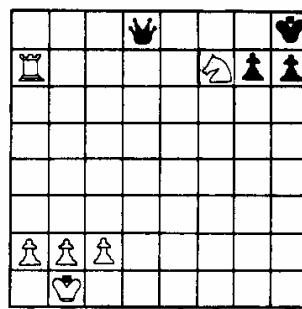
	Application	No. of Rules	Develop (Man Ys)	Maintain (Man Ys)	Learning Tools
MYCIN	Medical Diagnosis	400	100	N/A	N/A
XCON	VAX computer configuration	8,000	180	30	N/A
GASOIL	Hydrocarbon separation system configuration	2,800	1	0.1	ExpertEase and Extran7
BMT	Configuration of fire-protection equipment in buildings	30,000	9	2.0	1st Class and Rulemaster

จากตารางจะเห็นได้ว่าระบบผู้เชี่ยวชาญที่ไม่ใช้เครื่องมือการเรียนรู้ (MYCIN และ XCON) ใช้แรงงานในการพัฒนาและดูแลระบบมากกว่าระบบผู้เชี่ยวชาญที่ใช้เครื่องมือเรียนรู้ (GASOIL และ BMT) หลายเท่าเมื่อเทียบโดยจำนวนกฎที่ใช้ในระบบ ตัวอย่างนี้แสดงให้เห็นถึงประโยชน์ของการเรียนรู้ของเครื่องได้อย่างชัดเจน

6.6 การเรียนรู้โดยการอธิบาย

การเรียนรู้โดยการอธิบาย — อีบีแอล (*Explanation Based Learning — EBL*) [DeJong & Mooney, 1986; Mitchell, et al., 1986] เป็นการเรียนรู้ที่มีลักษณะเด่นคือสามารถเรียนรู้ได้จากตัวอย่างบวกเพียงอย่างเดียวไม่จำเป็นต้องใช้ตัวอย่างลบ และจำนวนตัวอย่างบวกที่ใช้ก็ใช้เพียงตัวเดียวที่สามารถทำการเรียนรู้ได้ โดยมีแนวคิดว่าการเรียนรู้สามารถทำได้โดยการให้ความรู้พื้นฐานของโดเมนที่เกี่ยวข้อง จากนั้นจะให้ตัวอย่างบวกที่เป็นตัวอย่างของมโนทัศน์ที่จะสอน กระบวนการเรียนรู้ก็คือการใช้ความรู้ในโดเมนนั้นมาอธิบายให้ได้ว่าทำในตัวอย่างที่สอนจึงเป็นตัวอย่างของมโนทัศน์แล้วจึงทำการวางแผนทั่วไปให้ครอบคลุมกรณีอื่นๆ

ยกตัวอย่างการเรียนรู้มโนทัศน์ fork ในการเล่นหมากรุกสากล (chess) โดยให้ตัวอย่างบวกของ fork ดังด้านล่างนี้



รูปที่ 6-31 ตัวอย่างบวกของ fork

ตัวอย่างด้านบนนี้แสดงสถานการณ์ที่ “ม้าขาวโจมตีกิงตำแหน่งควีนดำพร้อมกัน” ในกรณีนี้ฝ่ายดำต้องยอมเสียควีน ไม่เช่นนั้นจะแพ้ จากตัวอย่างบวกด้วยตัวเดียวด้านบน อีบีแอลจะเรียนได้ก้าวต่อไป “ถ้าตัวหมาก x โจมตีกิงกับตัวหมาก y ของฝ่ายตรงข้ามพร้อมกันแล้ว ฝ่ายตรงข้ามจะเสีย y” ซึ่งวิธีการเรียนรู้ก็จะกล่าวต่อไป กฎที่เรียนรู้ได้นี้สามารถใช้กับสถานการณ์อื่นๆ นอกเหนือจากตัวอย่างสอนอีกด้วย กล่าวคือ x ไม่จำเป็นต้องเป็นม้าหรือ y ไม่จำเป็นต้องเป็นควีน นอกเหนือจากนั้นตำแหน่งของตัวหมากอื่นๆ ที่ไม่เกี่ยวข้องกับมโนทัศน์นี้ก็จะไม่ปรากฏในกฎ หมายความว่าตำแหน่งของตัวหมากอื่นๆ จะอยู่ที่ได้ก็ได้ ทราบเท่าที่ตัวหมากเรากำลังโจมตีกิงและตัวหมากอีกตัวของฝ่ายตรงข้ามพร้อมกัน

จะเห็นได้ว่าประสิทธิภาพของอีบีแอลสูงมาก เพราะใช้ตัวอย่างแค่ตัวเดียวที่สามารถทำการวางแผนทั่วไปได้ สาเหตุที่สามารถทำได้เช่นนี้เนื่องจากว่าในอีบีแอลนี้เราต้องให้ความรู้ใน

โดเมนกับระบบเรียนรู้แบบนี้ด้วย ความรู้ในโดเมนของหมากรุกสากลก็อย่างเช่นกฎการเล่นหมากรุก ตัวมากแต่ละตัวเดินอย่างไร ม้า คิง ควิน เดินอย่างไร การกินกันเกิดขึ้นได้เมื่อไร เกมจบเมื่อไร เป็นต้น ซึ่งกฎเหล่านี้เราสามารถให้ได้ไม่ยากนัก เพราะมีเชียนไว้ในหนังสือ อธิบายวิธีเล่นหมากรุกอยู่แล้ว อย่างไรก็ได้แม้ว่าเราจะให้ความรู้ในโดเมนแล้วก็ไม่ได้หมายความว่าเราไม่ต้องสอนอีบีแอล เปรียบเสมือนการเรียนรู้ของนักเรียนมัธยม แม้ว่าเราจะยกทฤษฎีเกี่ยวกับการเท่ากันของสามเหลี่ยมไปครอบทุกทฤษฎีบท ก็ไม่ได้หมายความว่า นักเรียนจะพิสูจน์การเท่ากันของสามเหลี่ยมสองรูปได้ ได้ทันที ครูก็ยังคงต้องยกตัวอย่าง การพิสูจน์แสดงสามเหลี่ยม 2 รูปคู่หนึ่งๆ แล้วอธิบายว่าต้องใช้ทฤษฎีบทใดบ้างเพื่อการพิสูจน์และทำให้ไม่ทฤษฎีบทเหล่านี้จึงพิสูจน์การเท่ากันของสามเหลี่ยมที่ยกตัวอย่างให้ดูได้ ซึ่งจะช่วยให้นักเรียนเข้าใจได้ดีขึ้น และเมื่อทำโจทย์การพิสูจน์สามเหลี่ยม 2 รูปที่ใช้ทฤษฎีบทซึ่งเหมือนกับครุยกตัวอย่างก็จะทำโจทย์ได้

กระบวนการเรียนรู้ของอีบีแอลประกอบด้วย 2 ขั้นตอนหลักคือ

- ใช้ความรู้ในโดเมนอธิบายให้ได้ว่าทำไม่ตัวอย่างจึงเป็นตัวอย่างของมโนทัศน์ในรูปของกฎ
 - ทำการวางแผนที่ไปของกฎที่ได้เพื่อให้เข้ากับกรณีนี้ได้
- อินพุตและเอาต์พุตของอีบีแอลเป็นดังตารางที่ 6-15 ต่อไปนี้

ตารางที่ 6-15 อินพุตและเอาต์พุตของอีบีแอล

อินพุต:	<ul style="list-style-type: none"> ตัวอย่างสอน (training example) – ตัวอย่างบางข้อมโนทัศน์ที่จะสอน เช่นในกรณีของ fork ตัวอย่างสอนคือตำแหน่งตัวหมากบนกระดานที่เกิด fork มโนทัศน์เป้าหมาย (goal concept) – มโนทัศน์ที่จะสอนเช่นมโนทัศน์ fork เกณฑ์ดำเนินการ (operational criterion) – คำอธิบายที่สามารถนำไปใช้ได้ทันที เช่นในกรณีของ fork นั้น เพื่อเดิน attack-both(WKn,BK,BQ) ไม่สามารถนำไปใช้ได้ทันที ที่ต้องแสดงในรูปของตำแหน่งตัวหมากบนกระดาน เช่น position(WKn,f7), position(BK,h8), position(BQ,d8) เป็นต้น ความรู้ในโดเมน (domain knowledge) – กฎต่างๆ ที่ใช้แสดงความสัมพันธ์ของวัตถุและการกระทำต่างๆ ในโดเมนนั้น เช่น กฎการเล่นหมากรุกสากล เป็นต้น
เอาต์พุต:	<ul style="list-style-type: none"> การวางแผนที่ไปของตัวอย่างสอนซึ่งเพียงพอสำหรับอธิบายมโนทัศน์เป้าหมาย และสอดคล้องกับเกณฑ์ดำเนินการ

ตัวอย่างการเรียนรู้มโนทัศน์ cup

ตัวอย่างการเรียนรู้ที่จะยกมานี้เป็นตัวอย่างการเรียนรู้มโนทัศน์ cup (อะไรคือถ้วย) โดยมี อินพุตที่ให้ดังนี้

- ตัวอย่างบวก:

owner(object23,ralph), has-part(object23,concavity12),
 isa(concavity12,concavity), is(concavity12,upward-pointing),
 has-part(object23,handle16), isa(handle16,handle), is(object23,light),
 color(object23,brown), has-part(object23,bottom19), is(bottom19,bottom),
 is(bottom19,flat), ...

- ความรู้ในโดเมน:

liftable(X), stable(X), open-vessel(X) → cup(X)
 is(X,light), has-part(X,Y), isa(Y,handle) → liftable(X)
 small(X), made-from(X,Y), low-density(Y) → liftable(X)
 has-part(X,Y), isa(Y,bottom), is(Y,flat) → stable(X)
 has-part(X,Y), isa(Y,concavity), is(Y,upward-pointing) → open-vessel(X)

- มโนทัศน์เป้าหมาย: cup(X)

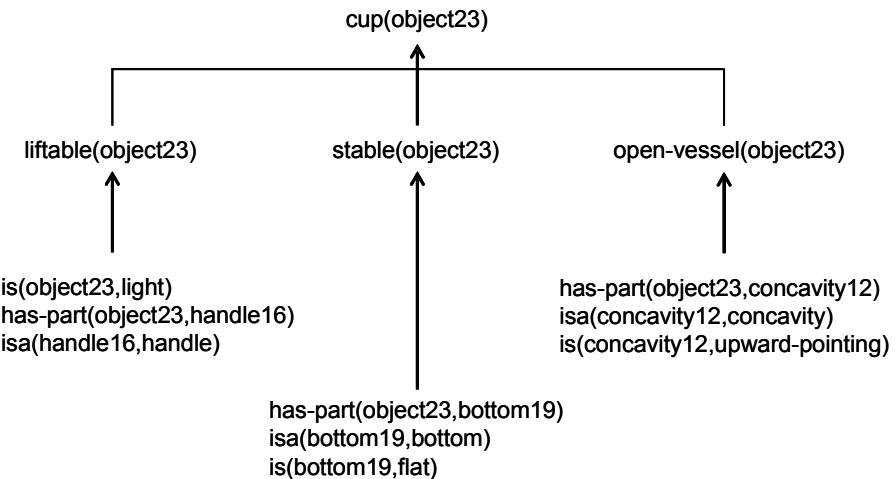
X เป็นถ้วยก็ต่อเมื่อ X ยกได้ (liftable) เสถียร (stable) และเป็นภาชนะเปิด (open-vessel)

- เกณฑ์ดำเนินการ: สิ่งที่แสดงลักษณะต่างๆ ของถ้วย เช่น isa, has-part, color, owner เป็นต้น

ขั้นตอนการเรียนรู้ประกอบด้วย 2 ขั้นตอนดังนี้

ต้นไม้พิสูจน์

(1) ใช้ความรู้ในโดเมนอธิบายว่าทำไม object23 จึงเป็น cup โดยการสร้างต้นไม้พิสูจน์ (proof tree) ของ object23 ดังแสดงในรูปที่ 6-32



รูปที่ 6-32 ต้นไม้พิสูจน์ของตัวอย่างบวก cup

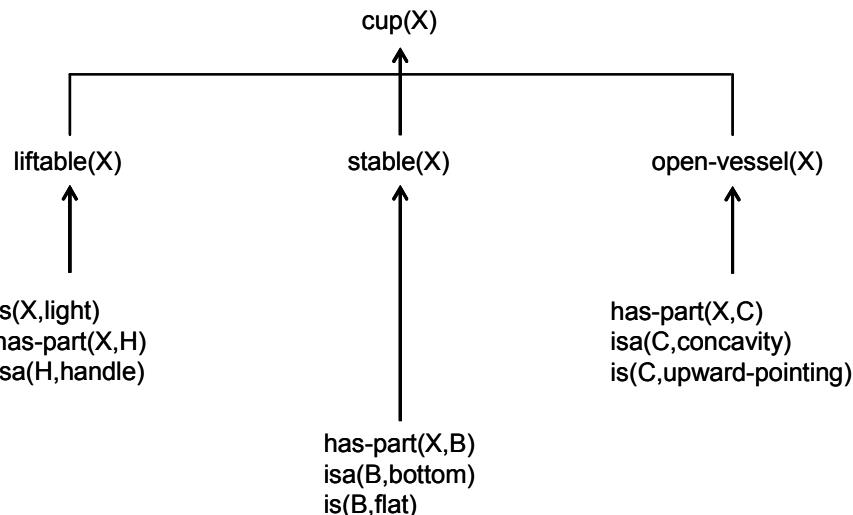
ต้นไม้พิสูจน์แสดงว่า object23 เป็น cup โดยมีคุณสมบัติ 3 อย่างคือยกได้ เสถียร และเป็นภาชนะเปิด เมื่อเราสังเกตความรู้ในโดเมนเรื่องถ้วยจะพบว่าการยกได้ของถ้วย มีกฎ 2 ข้อที่ใช้อธิบายได้และ object23 ตรงกับกฎข้อแรกของการยกได้ กล่าวคือเป็นถ้วยที่มีหูหิ้วและเป็น ออกจากนั้นในต้นไม้พิสูจน์จะไม่มีเพรดิคเตที่ไม่เกี่ยวข้องกับการเป็นถ้วย อย่างเช่น owner, color เป็นต้น ซึ่งสิ่งนี้เป็นการทำความนัยทั่วไปแบบหนึ่งที่ตัดเงื่อนไขไม่จำเป็นทั้งไป ดังนั้น ณ จุดนี้ถ้าเราสร้างกฎขึ้นเพื่ออธิบายการเป็นถ้วยของ object23 ก็จะได้กฎดังนี้

$\text{is}(\text{object23}, \text{light}), \text{has-part}(\text{object23}, \text{handle16}), \text{isa}(\text{handle16}, \text{handle}),$
 $\text{has-part}(\text{object23}, \text{bottom19}), \text{isa}(\text{bottom19}, \text{bottom}), \text{is}(\text{bottom19}, \text{flat}),$
 $\text{has-part}(\text{object23}, \text{concavity12}), \text{isa}(\text{concavity12}, \text{concavity}),$
 $\text{is}(\text{concavity12}, \text{upward-pointing}) \rightarrow \text{cup}(\text{object23})$

อย่างไรก็ได้ม้วกว่ากฎนี้จะไม่มีเพรดิคเตที่ไม่เกี่ยวข้อง แต่ว่ากฎนี้ยังคงอธิบายได้เฉพาะ object23 เท่านั้น เราจำเป็นต้องทำการวางแผนทั่วไปเพิ่มเติมขึ้นเพื่อให้ใช้กับถ้วยที่มีคุณสมบัติเหมือนกับ object23 ได้

- (2) การวางแผนทั่วไปและดึงเพรดิคเตทที่อยู่ในเกณฑ์ดำเนินการมาสร้างกฎ ขั้นตอนนี้ทำการวางแผนทั่วไปโดยตามความรู้ในโดเมน กล่าวคือถ้าอาร์กิวเมนต์ของเพรดิคเตทในต้นไม้พิสูจน์ที่ตรงกับอาร์กิวเมนต์ของเพรดิคเตทของความรู้ในโดเมนเป็นตัวแปรก็เปลี่ยนอาร์กิวเมนต์ที่เป็นค่าคงที่ให้เป็นตัวแปร แต่ถ้าอาร์กิวเมนต์ของเพรดิคเตทที่

ตรงกันกับความรู้ในโดเมนเป็นค่าคงที่ก็ไม่ต้องเปลี่ยน เช่น $\text{is}(\text{object23}, \text{light})$ เปลี่ยนเป็น $\text{is}(X, \text{light})$ เป็นต้น ผลที่ได้แสดงในรูปที่ 6-33



รูปที่ 6-33 การวางแผนที่ไปของตัวอย่างบวก cup

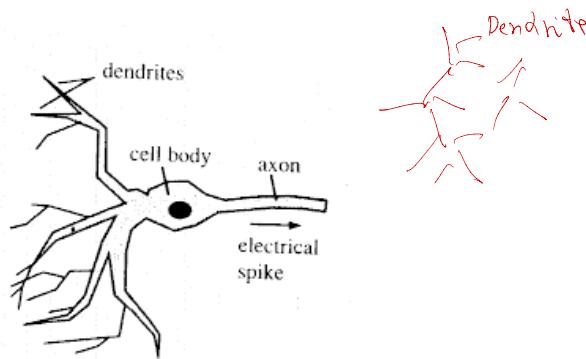
ดังนั้นเราจะได้กูดังนี้

$\text{is}(X, \text{light})$, $\text{has-part}(X, H)$, $\text{isa}(H, \text{handle})$, $\text{has-part}(X, B)$, $\text{isa}(B, \text{bottom})$, $\text{is}(B, \text{flat})$,
 $\text{has-part}(X, C)$, $\text{isa}(C, \text{concavity})$, $\text{is}(C, \text{upward-pointing}) \rightarrow \text{cup}(X)$

การเรียนรู้อีบีแอลนี้เป็นการเรียนรู้ประเภทที่เรียกว่า **การเรียนรู้เชิงวิเคราะห์ (analytical learning)** ก้าวคือการเรียนรู้ประเภทนี้จะเป็นการจัดความรู้ (ความรู้ในโดเมน) ในรูปแบบใหม่ให้ใช้งานได้อย่างมีประสิทธิภาพ ดังจะเห็นได้ว่ากูที่ได้โดยอีบีแอลประกอบด้วย เพรดิคเตทที่อยู่ในเกณฑ์ดำเนินการเท่านั้น ซึ่งเพรดิคเตทเหล่านี้จะใช้งานได้อย่างมีประสิทธิภาพสามารถจับคู่ (match) กับข้อมูลในตัวอย่างที่สอนแล้วทราบทันทีว่าตรงกัน หรือไม่ ต่างกับความรู้ในโดเมนเดิมที่ประกอบด้วยเพรดิคเตทบางตัว เช่น liftable ที่ต้องการการอธิบายโดยการพิสูจน์ต่อว่าเพรดิคเตทนี้ตรงกับตัวอย่างหรือไม่

6.7 ข่ายงานประสาทเทียม

ข่ายงานประสาทเทียม (Artificial Neural Network) เป็นการจำลองการทำงานบางส่วนของสมองมนุษย์ เชลล์ประสาท (neuron) ในสมองของคนเราประกอบด้วยนิวเคลียส (nucleus) ตัวเซลล์ (cell body) ใยประสาทนำเข้า (dendrite) แกนประสาทน้ำออก (axon) แสดงในรูปที่ 6-34

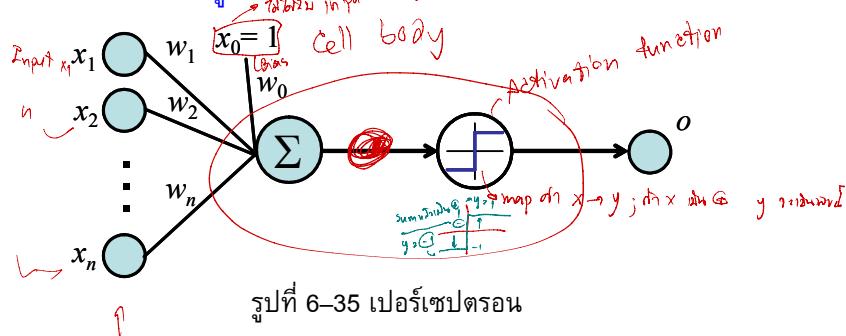


รูปที่ 6-34 เชลล์ประสาท

เดนไ/drท์ทำหน้าที่รับสัญญาณไฟฟ้าเคมีซึ่งส่งมาจากเซลล์ประสาทใกล้เคียง เชลล์ประสาทตัวหนึ่งๆ จะเชื่อมต่อกับเซลล์ตัวอื่นๆ ประมาณ 10,000 ตัว เมื่อสัญญาณไฟฟ้าเคมีที่รับเข้ามาเกินค่าค่าหนึ่ง เชลล์จะสูญเสียตุนและส่งสัญญาณไปทางแกนประสาทน้ำออกไปยังเซลล์อื่นๆ ต่อไป ประมาณกันว่าสมองของคนเรามีเซลล์ประสาทอยู่ทั้งสิ้นประมาณ 10^{11} ตัว

6.7.1 เพอร์เซปตรอน

เพอร์เซปตรอน (perceptron) เป็นข่ายงานประสาทเทียมแบบง่ายมีหน่วยเดียวที่จำลองลักษณะของเซลล์ประสาทดังรูปที่ 6-35



รูปที่ 6-35 เปอร์เซปตรอน

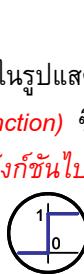
Input

$$x_0w_0 + x_1w_1 + x_2w_2 + \dots + x_nw_n$$

เพอร์เซปตรอนรับอินพุตเป็นเวกเตอร์จำนวนจริงแล้วคำนวณหาผลรวมเชิงเส้น (linear combination) แบบถ่วงน้ำหนักของอินพุต (x_1, x_2, \dots, x_n) โดยที่ค่า w_1, w_2, \dots, w_n ในรูปเป็นค่าถ่วงน้ำหนักของอินพุตและให้เอาต์พุต (o) เป็น 1 ถ้าผลรวมที่ได้มีค่าเกินค่าขีดแบ่ง (θ) และเป็น -1 ถ้าไม่เกิน ส่วน w_0 ในรูปเป็นค่าลบของค่าขีดแบ่งดังจะได้อธิบายต่อไป และ x_0 เป็นอินพุตเที่ยมกำหนดให้มีค่าเป็น 1 เสมอ



พังก์ชันกระตุ้น



ในรูปแสดงพังก์ชันกระตุ้น (activation function) ชนิดที่เรียกว่า **พังก์ชันสองขั้ว (bipolar function)** ซึ่งแสดงผลของเอาต์พุตเป็น 1 กับ -1 พังก์ชันกระตุ้นอื่นๆ ที่นิยมใช้กันอย่างเช่น **พังก์ชันไบนารี (binary function)** ซึ่งแสดงผลของเอาต์พุตเป็น 1 กับ 0 และเขียนแทนด้วยรูป

เราสามารถแสดงเอาต์พุต (o) ในรูปของพังก์ชันของอินพุต (x_1, x_2, \dots, x_n) "ได้ดังนี้"

$$o(x_1, x_2, \dots, x_n) = \begin{cases} 1 & \text{if } w_1 x_1 + w_2 x_2 + \dots + w_n x_n > \theta \\ -1 & \text{if } w_1 x_1 + w_2 x_2 + \dots + w_n x_n < \theta \end{cases} \quad (6.7)$$

เอาต์พุตเป็นพังก์ชันของอินพุตในรูปของผลรวมเชิงเส้นแบบถ่วงน้ำหนัก น้ำหนักจะเป็นตัวกำหนดว่าในจำนวนอินพุตนั้น อินพุต (x_i) ตัวใดมีความสำคัญต่อการกำหนดค่าเอาต์พุต ตัวที่มีความสำคัญมากจะมีค่าสัมบูรณ์ของน้ำหนักมาก ส่วนตัวที่มีความสำคัญน้อยจะมีค่าใกล้ศูนย์ ในกรณีที่ผลรวมเท่ากับค่าขีดแบ่งค่าเอาต์พุตไม่นิยม (จะเป็น 1 หรือ -1 ก็ได้)

จากพังก์ชันในสูตรที่ (6.7) เราจัดรูปใหม่โดยย้าย θ ไปรวมกับผลรวมเชิงเส้นแล้วแทน $-\theta$ ด้วย w_0 เราจะได้พังก์ชันของเอาต์พุตดังด้านล่างนี้

$$o(x_1, x_2, \dots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + w_2 x_2 + \dots + w_n x_n > 0 \\ -1 & \text{if } w_0 + w_1 x_1 + w_2 x_2 + \dots + w_n x_n < 0 \end{cases} \quad (6.8)$$

กำหนดให้ $g(\vec{x}) = \sum_{i=0}^n w_i x_i = \vec{w} \cdot \vec{x}$ โดยที่ \vec{x} แทนเวกเตอร์อินพุต เราสามารถเขียน

พังก์ชันของเอาต์พุตได้ใหม่ดังนี้

$$o(x_1, x_2, \dots, x_n) = \begin{cases} 1 & \text{if } g(\vec{x}) > 0 \\ -1 & \text{if } g(\vec{x}) < 0 \end{cases} \quad (6.9)$$

สมมติว่าเรามีอินพุตสองตัวคือ x_1 และ x_2 ซึ่งแสดงค่าส่วนสูงและน้ำหนักของเด็กนักเรียน ประсимและหลังจากที่แพทย์ตรวจร่างกายของเด็กโดยละเอียดแล้วได้จำแนกนักเรียน

ออกเป็นสองกลุ่มคือเด็กอ้วนและเด็กไม่อ้วน เราให้เอาต์พุตเป็นค่าที่แสดงเด็กอ้วนแทนด้วย +1 กับไม่อ้วนแทนด้วย -1 ดังตารางที่ 6-16

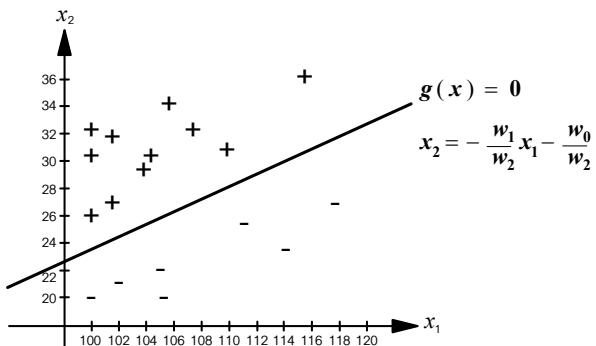
ตารางที่ 6-16 ข้อมูลเด็กอ้วนและเด็กไม่อ้วน

เด็กคนที่	ส่วนสูง (ซม.)	น้ำหนัก (กг.)	อ้วน/ไม่อ้วน
1	100.0	20.0	-1
2	100.0	26.0	1
3	100.0	30.4	1
4	100.0	32.4	1
5	101.6	27.0	1
6	101.6	32.0	1
7	102.0	21.0	-1
8	103.6	29.6	1
9	104.4	30.4	1
10	104.9	22.0	-1
11	105.2	20.0	-1
12	105.6	34.4	1
13	107.2	32.4	1
14	109.9	34.9	1
15	111.0	25.4	-1
16	114.2	23.5	-1
17	115.5	36.3	1
18	117.8	26.9	-1

ในการนี้ที่มีอินพุต 2 ตัว (ไม่รวม x_0) เราจะได้ $g(\vec{x}) = w_0 + w_1x_1 + w_2x_2$ ซึ่งถ้าเราให้ $g(\vec{x}) = 0$ จะได้ว่า $w_0 + w_1x_1 + w_2x_2 = 0$ ซึ่งแทนสมการเส้นตรงในระบบสองมิติ x_1 ,

x_2 สมการนี้มีจุดตัดแกนอยู่ที่ $-\frac{w_0}{w_2}$ และมีความชันเท่ากับ $-\frac{w_1}{w_2}$ เมื่อนำสมการนี้ไปวาดใน

ระบบสองมิติร่วมกับตัวอย่างสอนในตารางที่ 6-16 โดยกำหนดค่า w_0 , w_1 , w_2 ที่เหมาะสมจะได้ดังรูปที่ 6-36

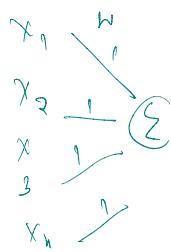


รูปที่ 6-36 สมการเส้นตัดร่างโดยเพอร์เซปตรอน

เครื่องหมาย + และ - ในรูปแทนตัวอย่างบวก (เด็กอ้วน) และตัวอย่างลบ (เด็กไม่อ้วน) ตามลำดับ ดังจะเห็นได้ในรูปว่าเส้นตัดร่างนี้เมื่อกำหนดจุดตัดแกนและความชันที่เหมาะสมซึ่ง กำหนดโดย w_0 , w_1 , w_2 เส้นตัดร่างนี้จะแบ่งตัวอย่างออกเป็นสองกลุ่มซึ่งอยู่คันละด้านของ เส้นตัดร่าง และเมื่อมีข้อมูลส่วนสูงและน้ำหนักของเด็กคนอื่นที่เราต้องการทำนายว่าจะเป็นเด็ก อ้วนหรือไม่ ก็ใช้เส้นตัดร่างนี้โดยดูว่าข้อมูลใหม่นี้อยู่ด้านใดของเส้นตัด ถ้าด้านบนก็ทำนายว่า เป็นเด็กอ้วน (+) ถ้าด้านล่างก็ทำนายว่าเด็กไม่อ้วน (-)

ตัวอย่างด้านบนแสดงกรณีของอินพุตในสองมิติ จะเห็นได้ว่าเพอร์เซปตรอนจะเป็น เส้นตัดร่าง ในการกรณีที่อินพุตมากกว่าสองมิติเพอร์เซปตรอนจะเป็น **ระนาบตัดสินใจหลายมิติ (hyperplane decision surface)** ปัญหาการเรียนรู้เพอร์เซปตรอนก็คือการหาค่าเวกเตอร์ น้ำหนัก (\vec{w}) ที่เหมาะสมในการจำแนกประเภทของข้อมูลสอนเพื่อให้เพอร์เซปตรอนแสดง เอ้าต์พุตได้ตรงกับค่าที่สอน **กฎการเรียนรู้เพอร์เซปตรอน (perceptron learning rule)** ใช้ สำหรับสอนเพอร์เซปตรอนโดยจะหาค่าเวกเตอร์น้ำหนักดังแสดงในตารางที่ 6-17

อัลกอริทึมเริ่มต้นจากสุ่มค่าเวกเตอร์น้ำหนัก ซึ่งโดยมากค่าที่สุ่มมาจะไม่ได้ระนาบ หลายมิติที่แบ่งตัวอย่างได้ถูกต้องทุกตัวดังนั้นจึงต้องมีการแก้ไขน้ำหนักโดยเทียบเพอร์เซปตรอนกับตัวอย่างที่สอน หมายถึงว่าเมื่อเราป้อนตัวอย่างสอนเข้าไปในเพอร์เซปตรอน เราจะคำนวนค่าเอ้าต์พุตได้ นำค่าเอ้าต์พุตที่คำนวนได้โดยเพอร์เซปตรอนเทียบกับ เอ้าต์พุตเป้าหมาย ถ้าตรงกันแสดงว่าจำแนกตัวอย่างได้ถูกต้อง ไม่ต้องปรับน้ำหนักสำหรับ ตัวอย่างนั้น แต่ถ้าไม่ตรงกันจะทำการปรับน้ำหนักตามสมการในอัลกอริทึม สำหรับการ เรียนรู้เป็นตัวเลขบวกจำนวนน้อยๆ เช่น 0.01, 0.005 เป็นต้น อัตราการเรียนรู้นี้จะส่งผลต่อ การลู่เข้าของเพอร์เซปตรอน ถ้าอัตราการเรียนรู้มีค่ามากเพอร์เซปตรอนก็จะเรียนรู้ได้เร็ว แต่ก็อาจเรียนรู้ไม่สำเร็จเนื่องจากการปรับค่ามีความหมายเกินไป อัตราการเรียนรู้ที่มีค่า น้อยก็จะทำให้การปรับน้ำหนักทำได้อย่างละเอียดแต่ก็อาจเสียเวลาในการเรียนรู้นาน



ตารางที่ 6-17 อัลกอริทึมภารกิจการเรียนรู้เพอร์เซปตรอน

Stopping condition

1. σ in Epoch
 2. Weight Y_{output}

~~Algorithm: Perceptron-Learning-Rule~~

1. Initialize weights w_i of the perceptron.
 2. UNTIL the termination condition is met DO *show data (training)*
 - 2.1 FOR EACH training example DO
 - Input the example and compute the output.
 - Change the weights if the output from the perceptron is not equal to the target output using the following rule.

$$w_i \leftarrow w_i + \Delta w_i$$

↑
lth
↑
Update

← learning rate
Δw_i ← α(t-o)x_i ← input feature.

actual value
target

where t , o and α are the target output, the output from the perceptron and the learning rate, respectively.

Predict, Act

การปรับน้ำหนักตามกฎการเรียนรู้เพอร์เซปตรอนโดยใช้อัตราการเรียนรู้ที่มีค่าน้อยเพียงพอ จะได้รับ nab หลายมิติที่จะสู่เข้าสู่ระบบหนึ่งที่สามารถแบ่งข้อมูลออกเป็นสองส่วน (ในกรณีที่ข้อมูลสามารถแบ่งได้) เพื่อวิเคราะห์ผลที่เกิดจากการปรับค่าน้ำหนัก เราจะลองพิจารณาพฤติกรรมของกฎการเรียนรู้นี้ดูว่าทำไงการปรับน้ำหนัก เช่นนี้จึงสู่เข้าสู่ระบบหนึ่งที่แบ่งข้อมูลได้อย่างถูกต้อง

- พิจารณากรณีแรกที่ $\text{เพอร์เซปตอรอน} \text{แยกตัวอย่าง} \text{สองตัวหนึ่ง} \text{ที่รับเข้ามา} \text{ได้ถูกต้อง}$ กรณีนี้จะพบว่า $(t=0)$ จะมีค่าเป็น 0 ดังนั้น Δw_i $\text{ไม่เปลี่ยนแปลง} \text{ เพราะ} \Delta w_i = \alpha(t=0)x_i$
 - พิจารณาในกรณีที่ $\text{เพอร์เซปตอรอน} \text{ให้} \text{เอ้าต์พุต} \text{เป็น} -1$ $\text{แต่} \text{เอ้าต์พุต} \text{เป้าหมาย} \text{หรือ} \text{ค่า} \text{ที่} \text{แท้จริง} \text{เท่ากับ} 1$ $\text{ในกรณี} \text{นี้} \text{หมายความ} \text{ว่า} \text{ค่า} \text{ที่} \text{เรา} \text{ต้องการ} \text{คือ} 1$ $\text{แต่} \text{ค่า} \text{นี้} \text{หนัก} \text{ไม่} \text{เหมาะสม} \text{ ดังนั้น} \text{เพื่อ} \text{ที่} \text{จะ} \text{ทำ} \text{ให้} \text{เพอร์เซปตอรอน} \text{ให้} \text{เอ้าต์พุต} \text{เป็น} 1$ $\text{น้ำหนัก} \text{ต้อง} \text{ถูก} \text{ปรับ} \text{ให้} \text{สามารถ} \text{เพิ่ม} \text{ค่า} \text{ของ} \vec{w} \cdot \vec{x}$ $\text{ในกรณี} \text{นี้} \text{หมายความ} \text{ว่า} \text{ผลรวม} \text{เชิงเส้น} \text{น้อย} \text{เกิน} \text{ไป} \text{และ} \text{น้อย} \text{กว่า} 0$ $\text{จึง} \text{ได้} \text{เอ้าต์พุต} \text{เป็น} -1$ $\text{ดังนั้น} \text{สิ่ง} \text{ที่} \text{เรา} \text{ต้องการ} \text{คือ} \text{การ} \text{เพิ่ม} \text{ค่า} \text{ผลรวม} \text{เชิงเส้น} \text{เพิ่ม} \text{ค่า} \text{เรา} \text{เพิ่ม} \text{ค่า} \text{ได้} \text{เรื่อยๆ} \text{ จน} \text{มาก} \text{กว่า} 0$ $\text{เพอร์เซปตอรอน} \text{จะ} \text{ให้} \text{เอ้าต์พุต} \text{เป็น} 1$ $\text{ซึ่ง} \text{ตรงกับ} \text{ที่} \text{เรา} \text{ต้องการ}$ พิจารณาดูดูดังต่อไปนี้ว่าการปรับค่าโดยกฎเรียนรู้ทำให้ผลรวมเชิงเส้นเพิ่มขึ้นได้อย่างไร กรณีนี้เราจะได้ว่า $(t=0)$ เท่ากับ $(1-(1))$ มีค่าเป็น 2 และลองพิจารณาค่าของอินพุต x_i แยกกรณีดังนี้

- ถ้า $x_i > 0$ จะได้ว่า Δw_i มากกว่า 0 เพราะว่า $\Delta w_i \leftarrow \alpha(t-o)x_i$ และ α มากกว่า 0, $(t-o) = 2$ และ $x_i > 0$ จากสมการการปรับน้ำหนัก $w_i \leftarrow w_i + \Delta w_i$ เมื่อ Δw_i มากกว่า 0 จะทำให้ w_i มีค่าเพิ่มขึ้นและ $\sum w_i x_i$ ก็จะมีค่าเพิ่มขึ้น เมื่อผลรวมมีค่ามากขึ้นแสดงว่าการปรับไปในทิศทางที่ถูกต้องคือเมื่อปรับไปจนกระทั่งได้ผลรวมมากกว่า 0 จะทำให้เพอร์เซปตรอนเอาร์พุตได้ถูกต้องยิ่งขึ้น
- ถ้า $x_i < 0$ เราจะได้ว่า $\alpha(t-o)x_i$ จะมีค่าน้อยกว่า 0 แสดงว่า w_i ตัวที่คุณกับ x_i ที่น้อยกว่า 0 จะลดลงทำให้ $\sum w_i x_i$ เพิ่มขึ้นเหมือนเดิม เพราะ x_i เป็นค่าลบและ w_i มีค่าลดลง ในที่สุดก็จะทำให้เพอร์เซปตรอนให้เอาร์พุตได้ถูกต้องยิ่งขึ้น
- ในการนี้ที่เพอร์เซปตรอนให้เอาร์พุตเป็น 1 แต่เอาร์พุตเป้าหมายหรือค่าที่แท้จริงเท่ากับ -1 จะได้ว่า w_i ของ x_i ที่เป็นค่าบวกจะลดลง ส่วน w_i ของ x_i ที่เป็นค่าลบจะเพิ่มขึ้นและทำให้การปรับเป็นไปในทิศทางที่ถูกต้องเช่นเดียวกับในกรณีแรก

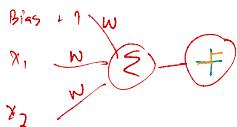
6.7.2 ตัวอย่างการเรียนฟังก์ชัน AND และ XOR ด้วยกฎเรียนรู้เพอร์เซปตรอน

พิจารณาตัวอย่างการเรียนรู้ของเพอร์เซปตรอนโดยจะให้เรียนรู้ฟังก์ชัน 2 ฟังก์ชัน ฟังก์ชัน แรกคือฟังก์ชัน AND และในตารางที่ 6-18 ในกรณีนี้เราใช้ฟังก์ชันไบนารีเป็นฟังก์ชันกระดับ

ตารางที่ 6-18 ฟังก์ชัน AND(x_1, x_2)

x_1	x_2	เอาร์พุต เป้าหมาย
0	0	0
0	1	0
1	0	0
1	1	1

ฟังก์ชัน AND ตามตารางด้านบนนี้จะให้ค่าที่เป็นจริงก็ต่อเมื่อ x_1 และ x_2 เป็นจริงทั้งคู่ (ดูที่สมมติเอาร์พุตเป้าหมาย) ผลการใช้กฎการเรียนรู้เพอร์เซปตรอนกับฟังก์ชัน AND และในตารางที่ 6-19



ตารางที่ 6-19 ผลการเรียนรู้พังก์ชัน AND โดยกฎการเรียนรู้เพอร์เซปตรอน

Perceptron Learning Example - Function AND $\alpha(t-o) \times_i$											
Input	Bias Input $x_0 = +1$		Net Sum		Target	Actual	α	$\alpha(t-o) \times_i$	Weight Values		
	x_1	x_2	$1.0 * w_0$	$x_1 * w_1$					w_0	w_1	w_2
	0	0	0.10	0.00	0.10	0	1	-0.50	-0.40	0.10	0.10
	0	1	-0.40	0.00	0.10	-0.30	0	0.00	-0.40	0.10	0.10
	1	0	-0.40	0.10	0.00	-0.30	0	0.00	-0.40	0.10	0.10
	1	1	-0.40	0.10	0.10	-0.20	1	0	0.50	0.10	0.60
	0	0	0.10	0.00	0.10	0	1	-0.50	-0.40	0.60	0.60
	0	1	-0.40	0.00	0.60	0.20	0	1	-0.50	-0.90	0.60
	1	0	-0.90	0.60	0.00	-0.30	0	0	0.00	-0.90	0.60
	1	1	-0.90	0.60	0.10	-0.20	1	0	0.50	-0.40	1.10
	0	0	-0.40	0.00	0.00	-0.40	0	0	0.00	-0.40	1.10
	0	1	-0.40	0.00	0.60	0.20	0	1	-0.50	-0.90	1.10
	1	0	-0.90	1.10	0.00	0.20	0	1	-0.50	-1.40	0.60
	1	1	-1.40	0.60	0.10	-0.70	1	0	0.50	-0.90	1.10
	0	0	-0.90	0.00	0.00	-0.90	0	0	0.00	-0.90	1.10
	0	1	-0.90	0.00	0.60	-0.30	0	0	0.00	-0.90	1.10
	1	0	-0.90	1.10	0.00	0.20	0	1	-0.50	-1.40	0.60
	1	1	-1.40	0.60	0.60	-0.20	1	0	0.50	-0.90	1.10
	0	0	-0.90	0.00	0.00	-0.90	0	0	0.00	-0.90	1.10
	0	1	-0.90	0.00	1.10	0.20	0	1	-0.50	-1.40	1.10
	1	0	-1.40	1.10	0.00	-0.30	0	0	0.00	-1.40	1.10
	1	1	-1.40	1.10	0.60	0.30	1	1	0.00	-1.40	1.10
	0	0	-1.40	0.00	0.00	-1.40	0	0	0.00	-1.40	1.10
	0	1	-1.40	0.00	0.60	-0.80	0	0	0.00	-1.40	1.10
	1	0	-1.40	1.10	0.00	-0.30	0	0	0.00	-1.40	1.10
	1	1	-1.40	1.10	0.60	0.30	1	1	0.00	-1.40	1.10

x_1, x_2 0 0, 1 1 \rightarrow 0.2 1

ขั้นตอนแรกเริ่มจากการสุ่มค่า w_0 จนถึง w_2 ในที่นี่กำหนดให้เป็น 0.1 ทั้งสามตัว จากนั้นก็เริ่มป้อนตัวอย่างเข้าไป (ทีละแบบ) ตัวอย่างแรกได้ผลรวมเชิงเส้น (Net Sum) เป็น 0.10 ซึ่งมากกว่า 0 ดังนั้น perceptron จะให้อาตต์พุตจริง (Actual Output) ออกมาระหว่าง 0 ถึง 1 ซึ่งผิดเพระ เอาต์พุตเป็นหมาย (Target Output) จะต้องได้เป็น 0 ทำให้อัตราการเรียนรู้คูณค่า ผิดพลาด (Alpha x Error) ได้ -0.50 หลังจากนั้นนำไปปรับหนักตาม $w_i \leftarrow w_i + \Delta w_i$ และ $\Delta w_i \leftarrow \alpha(t-o)x_i$ ดังนั้นจะได้เป็น $w_0 \leftarrow w_0 + \alpha(t-o)x_0 = w_0 + 0.50(-1) \times 1 = 0.10 + (-0.5)$ $= -0.4$ ต่อไปก็ปรับค่า w_1 ในทำนองเดียวกัน $w_1 \leftarrow w_1 + \alpha(t-o)x_1 = w_1 + 0.50(-1) \times 0$ ดังนั้น w_1 จะเท่ากับ 0.10 คือไม่เปลี่ยนแปลง เช่นเดียวกับ w_2 ที่ไม่เปลี่ยนแปลง จะเห็นได้ว่าแม้มีค่าผิดพลาดแต่ไม่มีการปรับค่า w_1 และ w_2 เนื่องจากอินพุตที่ใส่เข้าไปเป็น 0 ทำ

ให้ผลคุณเป็น 0 จึงไม่ได้ปรับ และเป็นข้อเสียของฟังก์ชันกระตุ้นแบบไบนาเรีซึ่งถ้าผลออกมานี้เป็น 0 จะไม่มีการปรับค่าให้ (ถ้าเราเปลี่ยน 0 เป็น -1 การปรับค่าจะดีขึ้น w_i จะถูกปรับทันทีตั้งแต่รอบแรก)

ตัวอย่างที่สองจะถือว่าตัวอย่างที่สักทำเช่นเดียวกัน และเมื่อทำการสอน (epoch) แล้วจะต้องทำการสอนซ้ำด้วยข้อมูลชุดเดิม นี่คือวิธีการสอนของข่ายงานประสาทเทียมซึ่งต่างจากวิธีอื่นๆ ที่ต้องใช้ข้อมูลชุดเดิมสอนซ้ำไปจนกระทั่งค่าพิดพลาดลดลงจนถึงจุดที่เราต้องการ ในที่นี่คือ 0 เนื่องจากเราต้องการให้มีการแบ่งข้อมูลอย่างเด็ดขาด และสมการเส้นตรงที่ได้จะมีค่า $w_0 = -1.40$, $w_1 = 1.10$ และ $w_2 = 0.60$

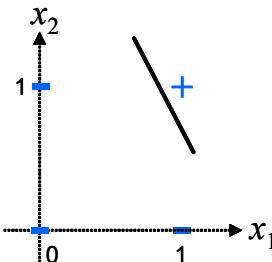
ฟังก์ชันที่สองที่จะทดลองเรียนรู้ด้วยกฎการเรียนรู้เพอร์เซปตรอนคือฟังก์ชัน XOR แสดงในตารางที่ 6-20

ตารางที่ 6-20 ฟังก์ชัน XOR(x_1, x_2)

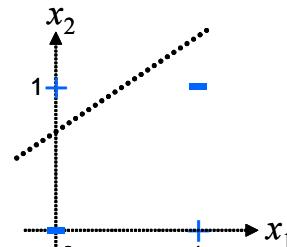
x_1	x_2	เอาต์พุตเป้าหมาย
0	0	0
0	1	1
1	0	1
1	1	0

ฟังก์ชัน XOR ตามตารางด้านบนนี้จะให้ค่าที่เป็นจริงก็ต่อเมื่อ x_1 หรือ x_2 ตัวใดตัวหนึ่งเพียงตัวเดียวเป็นจริง (ดูที่สมมติเอาต์พุตเป้าหมาย) ผลการใช้กฎการเรียนรู้เพอร์เซปตรอนกับฟังก์ชัน XOR แสดงในตารางที่ 6-21

ในกรณีของฟังก์ชัน XOR นี้พบว่าค่าพิดพลาดไม่ลดลง และค่าน้ำหนักจะแกงงไปมาโดยไม่ลู่เข้าแม้ว่าจะสอนต่อจากนี้ไปอีกกี่รอบการสอนก็ตาม จึงสรุปว่าฟังก์ชัน XOR เรียนไม่สำเร็จด้วยเพอร์เซปตรอน เมื่อเราทำฟังก์ชัน AND และ XOR ไปวัดกราฟในสองมิติจะได้กราฟดังรูปที่ 6-37



(ก) ฟังก์ชันแยกเชิงเส้นได้ (AND)



(ข) ฟังก์ชันแยกเชิงเส้นไม่ได้ (XOR)

รูปที่ 6-37 ฟังก์ชันแยกเชิงเส้นได้และไม่ได้

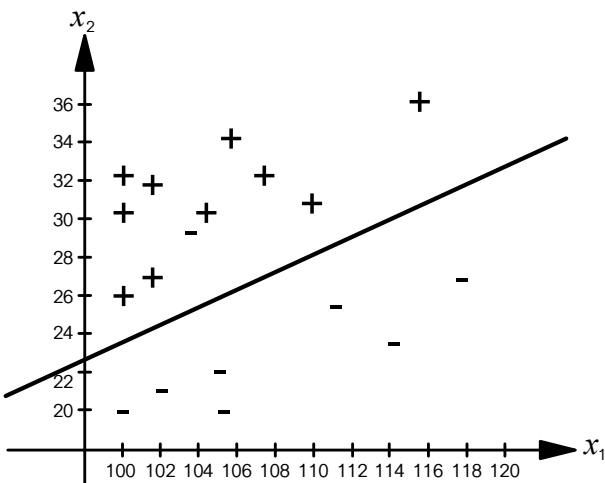
ตารางที่ 6-21 ผลการเรียนรู้พังก์ชัน XOR โดยภูมิการเรียนรู้เพอร์เซปตรอน

Perceptron Learning Example XOR												
		Bias Input X0=+1				Alpha = 0.5						
Input	Input			Net Sum	Target	Actual	Alpha*	Weight Values				
x1	x2	1.0*w0	x1*w1	x2*w2	Input	Output	Output	w0	w1	w2		
0	0	0.10	0.00	0.10	0	1	-0.50	-0.40	0.10	0.10		
0	1	-0.40	0.00	0.10	-0.30	1	0	0.50	0.10	0.10		
1	0	0.10	0.10	0.00	0.20	1	1	0.00	0.10	0.10		
1	1	0.10	0.10	0.60	0.80	0	1	-0.50	-0.40	-0.40		
0	0	-0.40	0.00	0.00	-0.40	0	0	0.00	-0.40	-0.40		
0	1	-0.40	0.00	0.10	-0.30	1	0	0.50	0.10	-0.40		
1	0	0.10	-0.40	0.00	-0.30	1	0	0.50	0.60	0.10		
1	1	0.60	0.10	0.60	1.30	0	1	-0.50	0.10	-0.40		
0	0	0.10	0.00	0.00	0.10	0	1	-0.50	-0.40	-0.40		
0	1	-0.40	0.00	0.10	-0.30	1	0	0.50	0.10	-0.40		
1	0	0.10	-0.40	0.00	-0.30	1	0	0.50	0.60	0.10		
1	1	0.60	0.10	0.60	1.30	0	1	-0.50	0.10	-0.40		
0	0	0.10	0.00	0.00	0.10	0	1	-0.50	-0.40	-0.40		
0	1	-0.40	0.00	0.10	-0.30	1	0	0.50	0.10	-0.40		
1	0	0.10	-0.40	0.00	-0.30	1	0	0.50	0.60	0.10		
1	1	0.60	0.10	0.60	1.30	0	1	-0.50	0.10	-0.40		
0	0	0.10	0.00	0.00	0.10	0	1	-0.50	-0.40	-0.40		
0	1	-0.40	0.00	0.10	-0.30	1	0	0.50	0.10	-0.40		
1	0	0.10	-0.40	0.00	-0.30	1	0	0.50	0.60	0.10		
1	1	0.60	0.10	0.60	1.30	0	1	-0.50	0.10	-0.40		
0	0	0.10	0.00	0.00	0.10	0	1	-0.50	-0.40	-0.40		
0	1	-0.40	0.00	0.10	-0.30	1	0	0.50	0.10	-0.40		
1	0	0.10	-0.40	0.00	-0.30	1	0	0.50	0.60	0.10		
1	1	0.60	0.10	0.60	1.30	0	1	-0.50	0.10	-0.40		

พังก์ชันแยก
เชิงเส้นไม่ได้

จากรูปจะเห็นได้ว่าพังก์ชัน AND เป็นพังก์ชันที่แยก (ระหว่างตัวอย่างบวกกับตัวอย่างลบ) ได้ด้วยเส้นตรง ส่วนพังก์ชัน XOR เราไม่สามารถหาเส้นตรงที่มาแบ่งตัวอย่างบวกและลบออกจากกัน (ไม่สามารถหาเส้นตรงให้ตัวอย่างบวกและลบให้อยู่คันละด้านของเส้น) ตัวอย่างการเรียนรู้พังก์ชัน XOR ข้างต้นได้แสดงให้เห็นว่า เพอร์เซปตรอนเรียนรู้บ้างพังก์ชันไม่ได้ พังก์ชันเหล่านี้เรียกว่า **พังก์ชันแยกเชิงเส้นไม่ได้ (linearly non-separable function)** ส่วนพังก์ชันที่แยกได้เรียกว่า **พังก์ชันแยกเชิงเส้นได้ (linearly separable function)** ซึ่งเป็นข้อจำกัดของเพอร์เซปตรอน เมื่อเราย้อนกลับไปดูตารางที่ 6-21 จะพบว่า นอกจากการเรียนรู้พังก์ชัน XOR "ไม่สำเร็จแล้ว การเรียนรู้ก็จะไม่ลู่เข้าสู่เส้นตรงได้เส้นตรงหนึ่งอีกด้วย ดังจะเห็นได้จากการที่เวกเตอร์น้ำหนักจะแกว่งไปมา การไม่ลู่เข้าก่อให้เกิดปัญหาในการเรียนรู้เพราเราจะไม่รู้ว่าเมื่อไรจะหยุดอัลกอริทึม พิจารณาตัวอย่างใน

รูปที่ 6-38 ซึ่งมีตัวอย่างสอนเหมือนกับในรูปที่ 6-36 ยกเว้นว่าตัวอย่างตัวที่แปดในตารางที่ 6-16 มีการบันทึกค่าของประเภทผิดจาก 1 เป็น -1 ทำให้เกิดตัวอย่างลบປะปນไปในกลุ่มของตัวอย่างบวกดังแสดงในรูป ในการนีเซ็นนีเส้นตรงที่ตีที่สุดก็ยังคงเป็นเส้นตรงเดิมเหมือนกับในรูปที่ 6-36 แต่ว่ากฏการเรียนรู้เพอร์เซปตรอนจะไม่ให้คำตอบเป็นเส้นตรงนี้เนื่องจากอัลกอริทึมไม่สู่เข้าสู่เส้นตรงเดียว แต่จะแก่วงไปมา

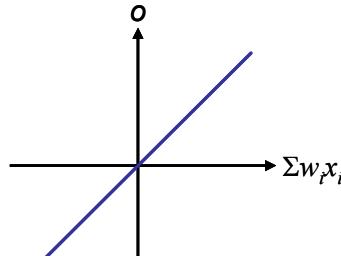


รูปที่ 6-38 เส้นตรงที่ให้ค่าผิดพลาดน้อยสุด

กฏเดลต้า (*delta rule*) เป็นกฏการเรียนรู้สำหรับหาค่าเวกเตอร์น้ำหนักของเพอร์เซปตรอนอีกกฎหนึ่งและมีข้อดีที่การเรียนรู้จะสู่เข้าสู่ระบบหลาบมิติที่ให้ค่าผิดพลาดน้อยสุด แม้ว่าตัวอย่างจะเป็นฟังก์ชันแบบแยกเชิงเส้นไม่ได้ กฏนี้ใช้หลักการของ **การเคลื่อนลงตามความชัน (gradient descent)** เพื่อหาคำตอบจากปริภูมิของเวกเตอร์น้ำหนักที่เป็นไปได้ ซึ่งกฏนี้เป็นพื้นฐานของอัลกอริทึมการแพร่กระจายย้อนกลับ (back-propagation) ดังจะกล่าวต่อไป

กฏเดลต้านี้จะหาเวกเตอร์น้ำหนักที่ให้ค่าผิดพลาดของตัวอย่างสอนน้อยสุดโดยใช้การหาอนุพันธ์ทางคณิตศาสตร์ ซึ่งในการหาค่าน้อยสุดด้วยอนุพันธ์นั้นจำเป็นต้องใช้ฟังก์ชันกราฟตุ้นที่หาอนุพันธ์ได้ ฟังก์ชันที่เราเคยใช้ก่อนหน้านี้ เช่น ฟังก์ชันสองขั้วและฟังก์ชันไบนาเรียเป็นฟังก์ชันที่หาอนุพันธ์ไม่ได้ในบางจุด ดังนั้นในกฏเดลต้านี้เราจะใช้ฟังก์ชันกราฟตุ้นแบบ **ฟังก์ชันเชิงเส้น (linear function)** ดังแสดงในรูปที่ 6-39 ซึ่งค่าเอาร์พุต (o) แสดงโดย $o(\vec{x}) = \vec{w} \cdot \vec{x} = \sum w_i x_i$ กล่าวคือเอาร์พุตจะเท่ากับผลรวมเชิงเส้น แม้ว่าตัวอย่างสอนจะแปรเปลี่ยนสองกลุ่ม (เช่น 1 กับ -1) ก็ไม่เกิดปัญหาเมื่อเราใช้ฟังก์ชันเชิงเส้นโดยจะทำนาย

ประเภทของตัวอย่างได้โดยรูปที่ เครื่องหมาย เช่น ฟังก์ชันเชิงเส้นให้อาต์พุตเป็น -0.21 ก็ให้กำหนดประเภทเป็น -1 เป็นต้น



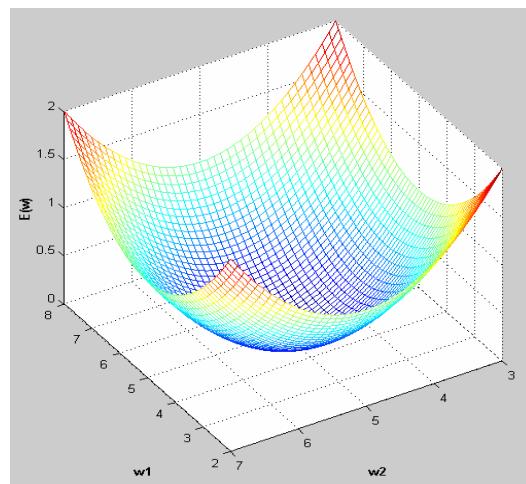
รูปที่ 6-39 ฟังก์ชันเชิงเส้น

ดังที่กล่าวข้างต้น กฎเดลต้าจะหาค่าเวกเตอร์น้ำหนักที่ให้ค่าผิดพลาดต่ำสุด ดังนั้นเรา尼ยามฟังก์ชันค่าผิดพลาดการสอน (training error function) $E(\vec{w})$ ดังนี้

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 \quad (6.10)$$

โดยที่ D เป็นเซตของตัวอย่างสอน t_d เป็นอาต์พุตเป้าหมายของตัวอย่าง d และ o_d เป็นอาต์พุตของเพอร์เซปตรอนสำหรับตัวอย่าง d

ฟังก์ชันค่าผิดพลาดการสอน $E(\vec{w})$ เป็นฟังก์ชันของ \vec{w} จะมีค่า \vec{w} บางตัวที่ทำให้ฟังก์ชันมีค่าต่ำสุด และพบว่าจะมี \vec{w} เช่นนั้นแค่ตัวเดียว เพราะ $E(\vec{w})$ เป็นฟังก์ชันพาราโบลาของ \vec{w} ในการนี้ที่ \vec{w} ประกอบด้วยน้ำหนัก 2 ค่าคือ w_1 และ w_2 เราจะได้ฟังก์ชันดังรูปที่ 6-40



รูปที่ 6-40 ฟังก์ชันค่าผิดพลาดการสอน $E(\vec{w})$

คุณสมบัติของพังก์ชันพาราโบล่าคือจะมีค่าต่ำสุดเพียงค่าเดียว ในการหาค่าต่ำสุดเราสามารถทำได้โดยกำหนดจุด (w_1, w_2) เริ่มต้น สมมติว่าเป็น (w_{10}, w_{20}) จากนั้นหาเวกเตอร์สัมผัสพาราโบล่า ณ ตำแหน่ง $E(w_{10}, w_{20})$ และเราจะวิ่งลงตามความชันของเวกเตอร์ที่สัมผัสถกับผิวค่าผิดพลาด (error surface) ถ้าชันมากก็ปรับค่าเวกเตอร์น้ำหนักมาก ถ้าชันน้อยก็ปรับค่าน้อยจนกระทั่งมาถึงจุดต่ำสุด ซึ่ง ณ จุดนี้ความชันจะเท่ากับศูนย์และไม่ต้องปรับค่าเวกเตอร์น้ำหนักอีกต่อไป ดังนั้นการใช้หลักการนี้ต้องการการหาอนุพันธ์ของผิวค่าผิดพลาด ซึ่งจะได้เป็นความชันของผิวสัมผัสกับผิวค่าผิดพลาด $E(\vec{w})$ นี้ (เขียนแทนด้วย $\nabla E(\vec{w})$) ดังแสดงต่อไปนี้

$$\nabla E(\vec{w}) = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right] \quad (6.11)$$

เนื่องจากเวกเตอร์สัมผัสนี้มีทิศในแนวขึ้น แต่เราต้องการวิ่งลงดังนั้นเวกเตอร์ในแนวลงจึงเป็น $-\nabla E(\vec{w})$ เราจะได้ว่ากฏการปรับค่าเวกเตอร์น้ำหนักเป็น $\vec{w} \leftarrow \vec{w} + \Delta\vec{w}$ โดยที่ $\Delta\vec{w} = -\eta \nabla E(\vec{w})$ และ η คืออัตราการเรียนรู้เป็นค่าคงที่ตัวเลขบวก กฏเดลต้านี้สามารถเขียนให้อยู่ในรูปของสมाचิกแต่ละตัวของเวกเตอร์น้ำหนักได้เป็น $w_i \leftarrow w_i + \Delta w_i$ โดยที่

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

$\frac{\partial E}{\partial w_i}$ สามารถคำนวณได้ดังต่อไปนี้

$$\begin{aligned} \frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 \\ &= \frac{1}{2} \sum_{d \in D} \frac{\partial}{\partial w_i} (t_d - o_d)^2 \\ &= \frac{1}{2} \sum_{d \in D} 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d) \\ &= \sum_{d \in D} (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - \vec{w} \cdot \vec{x}_d) \end{aligned}$$

$$\frac{\partial E}{\partial w_i} = \sum_{d \in D} (t_d - o_d) (-x_{id})$$

โดยที่ $-x_{id}$ คือสมाचิก x_i ของตัวอย่าง d

$$\therefore \Delta w_i = \eta \sum_{d \in D} (t_d - o_d) x_{id} \quad (6.12)$$

อัลกอริทึมของกฎเดลต้าเป็นดัง [ตารางที่ 6-22](#) ต่อไปนี้

ตารางที่ 6-22 อัลกอริทึมกฎเดลต้า

Algorithm: Delta-Rule(training-examples, η)

Each training example is a pair $\langle \vec{x}, t \rangle$, where \vec{x} is the vector of input values, and t is the target output value. η is the learning rate.

1. Initialize each w_i to some small random value.
2. **UNTIL** the termination condition is met **DO**
 - 2.1 Initialize each Δw_i to zero.
 - 2.2 **FOR EACH** $\langle \vec{x}, t \rangle$ in training-examples **DO**
 - Input the instance \vec{x} to the unit and compute the output o .
 - **FOR EACH** linear unit weight w_i **DO**

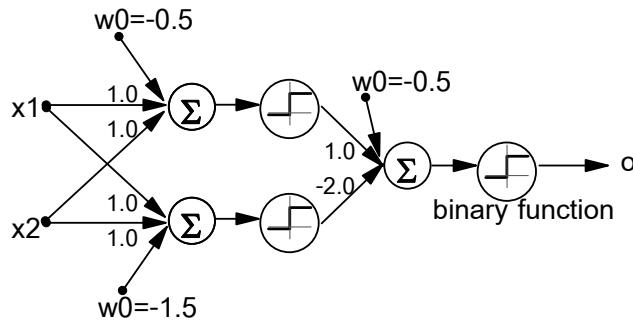
$$\Delta w_i \leftarrow \Delta w_i + \eta(t - o)x_i$$
 - 2.3 **FOR EACH** linear weight w_i **DO**

$$w_i \leftarrow w_i + \Delta w_i$$

อัลกอริทึมกฎเดลต้าข้างต้นนี้จะหาค่าเวกเตอร์น้ำหนักที่ให้ความผิดพลาดน้อยสุด ซึ่งมีข้อดีที่อัลกอริทึมจะลุ้นเข้า อย่างไรก็ได้พังก์ชันแยกเชิงเส้นไม่ได้ที่เรียนรู้ไม่ได้ด้วยกฎเรียนรู้ เพื่อเริ่มต้นก็ไม่สามารถแยกได้อย่างถูกต้องสมบูรณ์ด้วยกฎเดลต้าเช่นกัน ในหัวข้อต่อไปจะกล่าวถึงข่ายงานหลายชั้นที่สามารถแยกพังก์ชันประเภทนี้ได้

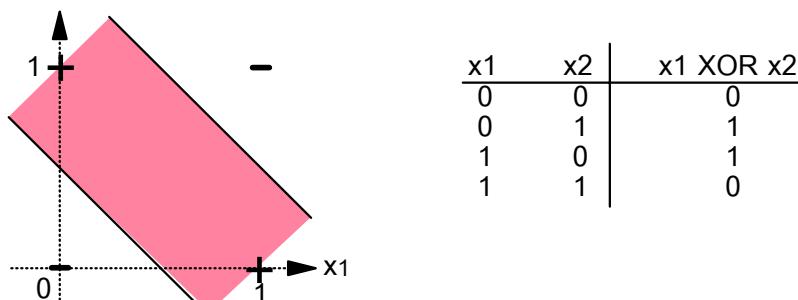
6.7.3 ข่ายงานหลายชั้นและการแพร่กระจายข้อมูล

จากข้างต้นจะเห็นว่าเพอร์เซปตรอนสามารถเรียนรู้พังก์ชันแยกได้เชิงเส้นเท่านั้น ในส่วนนี้จะอธิบายการนำเพอร์เซปตรอนหลายๆ ตัวมาเชื่อมต่อกัน เพื่อสร้างเป็นข่ายงานประสาทหลายชั้น (multilayer neural network) ที่สามารถแสดงผิวดั้ดสินใจไม่เชิงเส้น (non-linear decision surface) เพื่อให้เห็นถึงประสิทธิภาพของข่ายงานหลายชั้น จะยกตัวอย่างการต่อเพอร์เซปตรอน 3 ตัวเข้าด้วยกันเพื่อเรียนรู้พังก์ชัน XOR ดังแสดงใน [รูปที่ 6-41](#)



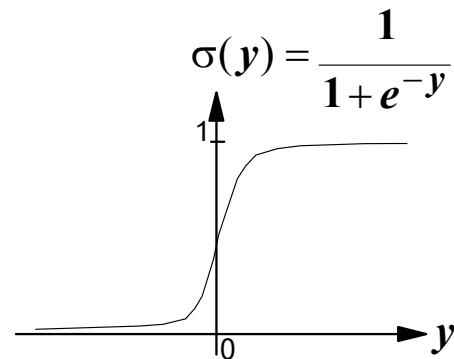
รูปที่ 6-41 ข่ายงานหลายชั้นสามารถเรียนรู้พังก์ชัน XOR

รูปที่ 6-41 แสดงการเชื่อมต่อเพอร์เซปตรอน 3 ตัวเข้าด้วยกัน เพอร์เซปตรอนสองตัว แรกรับอินพุตโดยตรงส่วนเพอร์เซปตรอนตัวที่สามรับอินพุตจากเอ้าต์พุตของ เพอร์เซปตรอนสองตัวแรก จะเห็นได้ว่าเพอร์เซปตรอนตัวแรกที่อยู่ด้านซ้ายบนของรูปนั้น แทนพังก์ชันเชิงเส้น $x_1 + x_2 = 0.5$ ส่วนเพอร์เซปตรอนตัวที่สองที่อยู่ด้านซ้ายล่างของรูปนั้น แทนพังก์ชันเชิงเส้น $x_1 + x_2 = 1.5$ พังก์ชันทั้งสองมีความซ้ำซ้อนเท่ากันเท่ากับ -1 แต่มี จุดตัดแกนต่างกันดังแสดงในรูปที่ 6-42 ส่วนเพอร์เซปตรอนตัวที่สามทำหน้าที่รวมผลลัพธ์ จากเพอร์เซปตรอนสองตัวแรก และโดยการกำหนดเวลาเตอร์หน้าหักที่หมายความของ เพอร์เซปตรอนตัวที่สามทำให้ได้ผิวตัดสินใจที่อยู่ระหว่างเส้นตรงทั้งสองเป็นตัวอย่างบวก และที่อยู่ด้านนอกเป็นตัวอย่างลบ



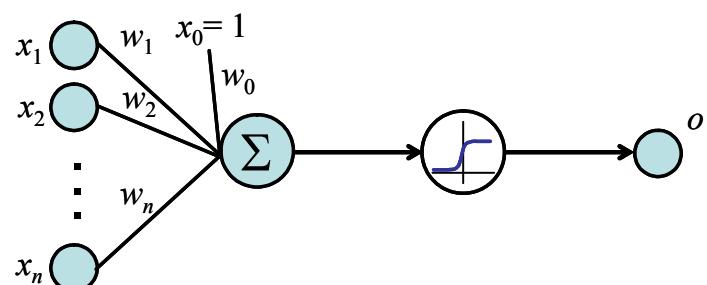
รูปที่ 6-42 ผิวตัดสินใจของข่ายงานในรูปที่ 6-41

ในการเชื่อมต่อครั้งนี้ใช้พังก์ชันกราฟตุนแบบไบโนารีเพื่อให้ง่ายต่อการทำความเข้าใจ แต่ การคำนวณหากภาระเรียนรู้สำหรับข่ายงานหลายชั้นต้องใช้พังก์ชันกราฟตุนที่หอนุพันธ์ได้ ดังนั้นเราจะไม่ใช้พังก์ชันไบโนารีกับข่ายงานหลายชั้น แต่จะใช้พังก์ชันเชิงเส้นหรืออาจใช้ พังก์ชันซิกมอยด์ (sigmoid function) ดังแสดงในรูปที่ 6-43



รูปที่ 6-43 พังก์ชันซิกมอยด์

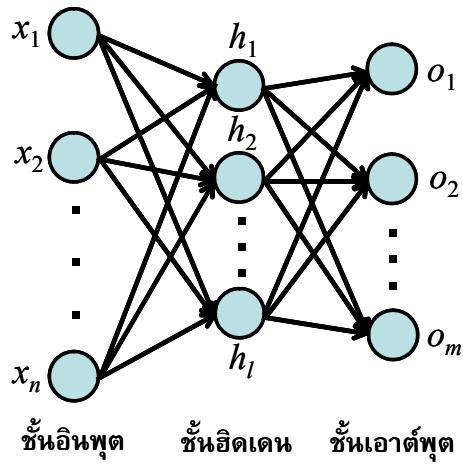
เพอร์เซปตรอนที่ใช้พังก์ชันกระตุ้นเป็นพังก์ชันซิกมอยด์แสดงในรูปที่ 6-44



รูปที่ 6-44 เพอร์เซปตรอนที่ใช้ฟังก์ชันซิกมอยด์

$$\frac{d\sigma(y)}{dy} = \sigma(y)(1 - \sigma(y)) \quad (6.13)$$

อัลกอริทึมการแพร่กระจายย้อนกลับ (*backpropagation algorithm*) [Rumelhart & McClelland, 1986] เรียนรู้ค่าเวกเตอร์น้ำหนักสำหรับข่ายงานป้อนไปหน้าแบบหลายชั้น (multilayer feedforward network) โดยใช้การเคลื่อนลงตามความชันเพื่อหาค่าต่ำสุดของค่าผิดพลาดระหว่างเอาต์พุตของข่ายงานกับเอาต์พุตเป้าหมาย ด้วยวิธีการขั้นตอนที่ 6-45



รูปที่ 6-45 ตัวอย่างข่ายงานป้อนไปหน้าแบบหลายชั้น

ตัวอย่างในรูปด้านบนแสดงข่ายงานป้อนไปหน้าแบบหลายชั้นซึ่งประกอบด้วยชั้นอินพุต ชั้นฮีดเดนหรือชั้นซ่อน และชั้นเอาต์พุต ในรูปแสดงชั้นฮีดเดนเพียงชั้นเดียวแต่อาจมีมากกว่าหนึ่งชั้นก็ได้ เส้นเชื่อมจะเชื่อมต่อเป็นชั้นๆ ไม่ข้ามชั้นจากชั้นอินพุตไปชั้นฮีดเดน ถ้ามีชั้นฮีดเดนมากกว่าหนึ่งชั้นก็เชื่อมต่อ กันไป และสุดท้ายจากชั้นฮีดเดนไปชั้นเอาต์พุต ข่ายงานป้อนไปหน้าแบบหลายชั้นนี้จะไม่มีเส้นเชื่อมย้อนกลับจะมีแต่เส้นเชื่อมไปข้างหน้าอย่างเดียวเช่นไม่มีเส้นเชื่อมจากบัพในชั้นเอาต์พุตส่งกลับมายังบัพในชั้นฮีดเดน หรือชั้นอินพุต เป็นต้น

ในการปรับค่าเวกเตอร์น้ำหนักโดยอัลกอริทึมการแพร่กระจายย้อนกลับนั้น เราต้องนิยามค่าผิดพลาดการสอนสำหรับข่ายงาน $E(\vec{w})$ จากนั้นจะหาค่าเวกเตอร์น้ำหนักที่ให้ค่าผิดพลาดต่ำสุด นิยามค่าผิดพลาดดังนี้

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in \text{outputs}} (t_{kd} - o_{kd})^2 \quad (6.14)$$

โดยที่ $outputs$ คือเซตของบัพเอาต์พุตในข่ายงาน t_{kd} และ o_{kd} เป็นค่าเอาต์พุตเป้าหมายและเอาต์พุตที่ได้จากข่ายงานตามลำดับของบัพเอาต์พุตที่ k ของตัวอย่างตัวที่ d อัลกอริทึมการแพร่กระจายย้อนกลับจะค้นหาเวกเตอร์น้ำหนักที่ให้ค่าผิดพลาดต่ำสุด แต่ในกรณีของข่ายงานป้อนไปหน้าแบบหลายชั้นนี้ค่าต่ำสุดมักมีมากกว่าหนึ่งจุด ดังนั้นคำตوبของ การแพร่กระจายย้อนกลับจึงเป็นค่าต่ำสุดเฉพาะที่ อัลกอริทึมแสดงใน [ตารางที่ 6-23](#)

ตารางที่ 6-23 อัลกอริทึมการแพร่กระจายข้อมูลลับ

Algorithm: Backpropagation(*training-examples, h, n_{in}, n_{out}, n_{hidden}*)

Each training example is a pair $\langle \vec{x}, \vec{t} \rangle$, where \vec{x} is the input vector, \vec{t} is the target output vector, η is the learning rate. $n_{in}, n_{out}, n_{hidden}$ are the number of network inputs, units in the hidden layer, output units, respectively. The input from unit i into unit j and the weight from unit i to unit j are denoted x_{ji} and w_{ji}

1. Initialize all network weights to small random numbers (e.g., $[-0.05..0.05]$)
2. **UNTIL** the termination condition is met **DO**
 - 2.1 **FOR EACH** $\langle \vec{x}, \vec{t} \rangle$ in training-examples **DO**
 - /*Propagate input forward through the network*/
 - Input the instance \vec{x} to the network, compute the output o_u of every unit u .
 - /*Propagate errors backward through the network*/
 - For each network output unit k , calculate its error term δ_k
$$\delta_k = o_k(1 - o_k)(t_k - o_k)$$
 - For each hidden unit h , calculate its error term δ_h
$$\delta_h = o_h(1 - o_h) \sum_{k \in outputs} w_{kh} \delta_k$$
 - Update each network weight w_{ji} : $w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$

where $\Delta w_{ji} = \eta \delta_j x_{ji}$

6.8 การเรียนรู้แบบเบส์

การเรียนรู้แบบเบส์ (*Bayesian learning*) เป็นวิธีการเรียนรู้ที่ใช้ทฤษฎีความน่าจะเป็นซึ่งมีพื้นฐานมาจาก **ทฤษฎีของเบส์** (*Bayes theorem*) เข้ามาช่วยในการเรียนรู้ จุดมุ่งหมายก็เพื่อต้องการสร้างโมเดลที่อยู่ในรูปของความน่าจะเป็น ซึ่งเป็นค่าที่บันทึกได้จากการสังเกตจากนั้นนำโมเดลมาหาว่าสมมติฐานใดถูกต้องที่สุดโดยใช้ความน่าจะเป็นเข้ามาช่วย ข้อดีก็คือเราสามารถใช้ข้อมูลและ **ความรู้ก่อนหน้า** (*prior knowledge*) เข้ามาช่วยในการเรียนรู้ได้ด้วย ความรู้ก่อนหน้าหมายถึงความรู้ที่เรามีเกี่ยวกับสมมติฐานแต่ละตัวก่อนที่เราจะเก็บข้อมูล เมื่อใช้งานเราจะนำความน่าจะเป็นของข้อมูลที่เก็บได้มาปรับสมมติฐานซ้ำอีกครั้ง ซึ่งพบว่าวิธีนี้ให้ประสิทธิภาพในการเรียนรู้ได้ดีไม่ต้องกว่าวิธีการเรียนรู้ประเภทอื่น

6.8.1 ทฤษฎีของเบส์

กำหนดให้ A และ B เป็นเหตุการณ์ใดๆ ความน่าจะเป็นของ A เมื่อรู้ B (ความน่าจะเป็นที่จะเกิดเหตุการณ์ A โดยมีเงื่อนไขว่าเหตุการณ์ B ได้เกิดขึ้นแล้ว) เขียนแทนด้วย $P(A|B)$ สามารถคำนวณได้ด้วยทฤษฎีของเบส์ดังนี้

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (6.15)$$

กล่าวคือความน่าจะเป็นของ A เมื่อรู้ B (โดยมีเงื่อนไขว่า B เกิดขึ้นแล้ว) สามารถคำนวณได้จากผลคูณของความน่าจะเป็นของ B เมื่อรู้ A กับความน่าจะเป็นของ A หารด้วยความน่าจะเป็นของ B เราเรียก $P(A)$ ว่า **ความน่าจะเป็นก่อน** (*prior probability*) และเรียก $P(A|B)$ ว่า **ความน่าจะเป็นภายหลัง** (*posterior probability*) ความน่าจะเป็นก่อนเป็นค่าที่ได้จากข้อมูลเบื้องต้น ส่วนความน่าจะเป็นภายหลังเป็นค่าความน่าจะเป็นก่อนที่ถูกปรับด้วยข้อมูลที่เพิ่มขึ้น

ในกรณีของการเรียนรู้ของเครื่องนั้น สิ่งที่เราสนใจคือเมื่อเรามีชุดข้อมูลหรือเซตของตัวอย่างสอน D เราต้องการหาค่าความน่าจะเป็นที่สมมติฐาน (h) ที่เราสนใจว่ามีโอกาสจะเกิดขึ้นเท่าไร เราจึงสามารถใช้ทฤษฎีของเบส์ในการคำนวณได้ดังนี้

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)} \quad (6.16)$$

ความน่าจะเป็นภายหลัง
และ
ความน่าจะเป็นก่อน

โดยที่ $P(h)$ คือความน่าจะเป็นก่อนซึ่งเป็นความน่าจะเป็นที่สมมติฐาน h จะเป็นจริงโดยที่เรายังไม่ได้ดูข้อมูลตัวอย่างสอน ส่วน $P(h|D)$ เป็นความน่าจะเป็นภายหลังซึ่งเป็นความน่าจะเป็นที่สมมติฐาน h จะเป็นจริงโดยมีเงื่อนไขว่า D เป็นจริง (เราเห็นข้อมูลตัวอย่างสอน D แล้ว) ในการเรียนรู้ของเครื่อง เราต้องการคำนวณความน่าจะเป็นภายหลังนี้ ซึ่งมักจะหาไม่ได้โดยตรง แต่ถ้าเราใช้ทฤษฎีของเบส์ดังข้างต้นความน่าจะเป็นนี้จะคำนวณได้ง่ายขึ้นโดยใช้นิพจน์ทางด้านความมือของสูตรที่ (6.16)

ยกตัวอย่าง เช่นถ้าเรามีต้นไม้ตัดสินใจหลายๆ ต้นและอยากรู้ว่าแต่ละต้นมีโอกาสเกิดขึ้นหรือมีความถูกต้องเท่าไร ก็คือเราต้องการหา $P(h|D)$ นั่นเอง โดยที่ h แทนต้นไม้ตัดสินใจต้นหนึ่งที่เรากำลังพิจารณา เราอาจจะมีความเชื่อว่าต้นไม้ต้นเล็กมีโอกาสที่จะเป็นจริงมากกว่าต้นใหญ่ (คล้ายกับกฎของอ็อกแคม) นั่นคือเรามีความน่าจะเป็นก่อน $P(h)$ ที่ต้นไม้จะเป็นจริงโดยยังไม่ได้ดูตัวอย่างสอน ซึ่งจะให้ความน่าจะเป็นของต้นไม้ต้นเล็กมีค่ามากกว่าของต้นไม้ต้นใหญ่ เมื่อเรารับตัวอย่างสอนแล้วนำมาปรับค่าความน่าจะเป็นก่อน ได้เป็นความน่าจะเป็นภายหลัง ส่วน $P(D|h)$ เป็นความน่าจะเป็นที่ D จะเป็นจริงเมื่อรู้ว่า h เป็นจริง ความน่าจะเป็นค่านี้สามารถวัดได้โดยนำตัวอย่างสอนมาตรวจสอบกับต้นไม้ h ว่าในจำนวนตัวอย่างสอนทั้งหมดนั้นมีอัตราส่วนของตัวอย่างที่ตรงหรือสอดคล้องกับต้นไม้เท่าไร ส่วน $P(D)$ เป็นความน่าจะเป็นที่เซตตัวอย่างสอนจะเป็นจริง ซึ่งในการหา h ที่ดีที่สุดนั้นโดยมากเรามักจะค้นที่โดยไม่ต้องคำนวณดังจะกล่าวต่อไป ดังนั้นจะเห็นได้ว่าการใช้ทฤษฎีของเบส์สามารถใช้คำนวณความน่าจะเป็นของสมมติฐานแต่ละตัว เมื่อรู้ว่าเซตตัวอย่างสอนเป็นจริงซึ่งจะช่วยให้เราเลือกสมมติฐานที่ดีที่สุดได้

สมมติฐานภาษาภายหลังมากสุด

เราระบุสมมติฐานที่ดีที่สุดว่า **สมมติฐานภาษาภายหลังมากสุด – เอ็มเอ็ป (Maximum A Posterior hypothesis – MAP)** ซึ่งนิยามให้เป็นดังนี้

$$\begin{aligned} h_{MAP} &= \arg \max_{h \in H} P(h | D) \\ &= \arg \max_{h \in H} \frac{P(D | h)P(h)}{P(D)} \end{aligned} \quad (6.17)$$

$$h_{MAP} = \arg \max_{h \in H} P(D | h)P(h) \quad (6.18)$$

โดยที่ H เป็นปริภูมิของสมมติฐานทั้งหมด $\arg \max f(x)$ เป็นฟังก์ชันที่คืนค่า x ที่ทำให้ $f(x)$ สูงสุด สมการที่ (6.17) ได้จากการใช้ทฤษฎีของเบส์และเนื่องจากว่าสำหรับ $h \in H$ ทุกตัวมี

ค่า $P(D)$ เท่ากันหมด ดังนั้นเราจึงสามารถถะ $P(D)$ ได้และได้สมการที่ (6.18) กล่าวคือ h ที่ดีที่สุดตามเอ็มเพรคิอ h ที่ทำให้ค่า $P(D|h)P(h)$ มีค่าสูงสุด

เทคนิคการเรียนรู้ของเครื่องหมายวิธีไม่ได้หาค่า h_{MAP} แต่มาหา h_{ML} (Maximum Likelihood hypothesis) ดังในสมการที่ (6.19) ด้านล่างนี้ ซึ่งหมายถึงสมมติฐานที่ตรงหรือสอดคล้องกับข้อมูลสอนมากสุดจะเป็นสมมติฐานที่ดีสุดโดยไม่ได้พิจารณาความน่าจะเป็นก่อน

$$h_{ML} = \arg \max_{h \in H} P(D | h) \quad (6.19)$$

ยกตัวอย่างการใช้ทฤษฎีของเบสเพื่อเลือกสมมติฐานที่น่าจะเป็นที่สุด สมมติว่าคนหนึ่งไปตรวจมะเร็งและผลการตรวจเป็นบวก อย่างไรก็ได้รวมค่าสถิติว่าผลการตรวจเมื่อเป็นบวกจะให้ความถูกต้อง 98% ของกรณีที่มีโรคนั้นอยู่จริง และผลการตรวจเมื่อเป็นลบจะให้ความถูกต้อง 97% ของกรณีที่ไม่มีโรคนั้น นอกจากนี้เรายังมีสถิติของการเป็นโรคมะเร็งว่า 0.008 ของประชากรทั้งหมดเป็นโรคมะเร็ง คำถามคือว่าคนไข้คนนี้มีโอกาสเป็นมะเร็งหรือไม่เป็นมะเร็งมากกว่ากัน?

เราใช้ทฤษฎีของเบสสำหรับปัญหานี้ โดยกำหนดให้ $H = \{\text{cancer}, \sim \text{cancer}\}$ กล่าวคือมีสมมติฐานที่เป็นไปได้สองข้อคือคนไข้คนนี้เป็นมะเร็งกับไม่เป็นมะเร็ง เชตัวอย่างสอนหรือข้อมูลของเรามีผลการตรวจเป็นบวก แทนด้วย + ดังนั้นเราแทนค่า H, h, D ในสมการที่ (6.18) จะได้ว่า

$$h_{MAP} = \arg \max_{h \in \{\text{cancer}, \sim \text{cancer}\}} P(+) | h) P(h) \quad (6.20)$$

จากข้อมูลทางสถิติทำให้ได้ว่า

$$P(\text{cancer}) = 0.008 \quad P(\sim \text{cancer}) = 0.992$$

$$P(+) | \text{cancer} = 0.98 \quad P(+) | \sim \text{cancer} = 0.03$$

ดังนั้นเราจะได้ว่าในกรณีของ

$$h=\text{cancer} \text{ ได้ด้านขวามือของสมการที่ (6.20) เป็น } 0.98 \times 0.008 = 0.00784$$

$$h=\sim \text{cancer} \text{ ได้ด้านขวามือของสมการที่ (6.20) เป็น } 0.03 \times 0.992 = 0.02976$$

เพราะฉะนั้นเราสรุปได้ว่า $h_{MAP} = \sim \text{cancer}$ กล่าวคือมีโอกาสไม่เป็นมะเร็งมากกว่า

6.8.2 สูตรพื้นฐานของความน่าจะเป็น

สูตรพื้นฐานเกี่ยวกับความน่าจะเป็น ที่จะใช้บ่อยครั้งในการเรียนรู้แบบเบสิก มีดังต่อไปนี้

1. กฎผลคูณ (product rule): ความน่าจะเป็น $P(A \wedge B)$ ที่สองเหตุการณ์ A และ B จะเกิดพร้อมกัน (หรือเขียนย่อเป็น $P(A, B)$) มีค่าเท่ากับ

$$P(A \wedge B) = P(A|B)P(B) = P(B|A)P(A)$$

2. กฎผลรวม (sum rule): ความน่าจะเป็น $P(A \vee B)$ ที่เหตุการณ์ A หรือ B เหตุการณ์ใดเหตุการณ์หนึ่งจะเกิดหรือเกิดพร้อมกันมีค่าเท่ากับ

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$

3. ทฤษฎีความน่าจะเป็นทั้งหมด (theorem of total probability) ถ้าเหตุการณ์

A_1, \dots, A_n ไม่เกิดร่วมกันและ $\sum_{i=1}^n P(A_i) = 1$ และ ความน่าจะเป็น $P(B)$ มีค่าเท่ากับ

$$P(B) = \sum_{i=1}^n P(B | A_i)P(A_i)$$

4. กฎลูกโซ่ (chain rule): A_1, \dots, A_n เป็นเหตุการณ์ n เหตุการณ์จะได้ว่าความน่าจะเป็นร่วม $P(A_1, \dots, A_n)$ มีค่าเท่ากับ

$$P(A_1, A_2, \dots, A_n) = \prod_{i=1}^n P(A_i | A_{i-1}, \dots, A_1)$$

6.8.3 การจำแนกประเภทที่น่าจะเป็นที่สุดสำหรับตัวอย่าง

ดังที่กล่าวข้างต้น ในกรณีที่กำหนดให้เราใช้สมมติฐานได้เพียงข้อเดียวในการจำแนกประเภทของตัวอย่าง จะได้ว่า h_{MAP} เป็นสมมติฐานที่ดีที่สุด แต่การจำแนกประเภทของตัวอย่างด้วย h_{MAP} ไม่ใช่ **การจำแนกประเภทที่น่าจะเป็นที่สุด (most probable classification)** สำหรับตัวอย่างนั้น ในบางกรณีที่เราสามารถใช้สมมติฐานหลายข้อเรารสามารถจำแนกประเภทของตัวอย่างได้ดีกว่าการใช้ h_{MAP} ตัวเดียว

สมมติว่าเรามีสมมติฐาน 3 ข้อ แต่ละข้อมีค่าความน่าจะเป็นภายหลังดังต่อไปนี้

$$P(h_1|D) = 0.4 \quad P(h_2|D) = 0.3 \quad P(h_3|D) = 0.3$$

และเมื่อให้ตัวอย่าง x ผลการจำแนกประเภทของสมมติฐานเป็นดังนี้

$$h_1(x) = + \quad h_2(x) = - \quad h_3(x) = -$$

ในกรณีนี้เรารู้ว่าจะจำแนกประเภทของ x เป็น哪ว่าหรือลบ? ซึ่งถ้าใช้ h_{MAP} ก็จะได้ว่า h_1 เป็นสมมติฐานที่ดีที่สุดเนื่องจาก h_1 มีค่าความน่าจะเป็นภายหลังมากที่สุด แต่เมื่อพิจารณาดูสมมติฐานอื่นในปริภูมิของสมมติฐาน เราพบว่า h_{MAP} ให้คำตอบเป็น + เพียงตัวเดียว แต่สมมติฐานอีกสองตัวให้คำตอบเป็น - เราจะได้ว่าการจำแนกประเภทที่น่าจะเป็นที่สุดในแบบของเบสเมสสูตรการคำนวณดังนี้

$$\arg \max_{v_j \in V} \sum_{h_i \in H} P(v_j | h_i) P(h_i | D) \quad (6.21)$$

โดยที่ V เป็นเซตของค่า (ประเภท) ของตัวอย่าง H เป็นปริภูมิของสมมติฐาน ในตัวอย่างด้านบนเราจะได้ว่า

$$\begin{array}{lll} P(h_1|D) = 0.4 & P(-|h_1) = 0.0 & P(+|h_1) = 1.0 \\ P(h_2|D) = 0.3 & P(-|h_1) = 1.0 & P(+|h_1) = 0.0 \\ P(h_3|D) = 0.3 & P(-|h_1) = 1.0 & P(+|h_1) = 0.0 \end{array}$$

ทำให้ได้ค่าความน่าจะเป็นของประเภท + และ - ดังนี้

$$\begin{aligned} \sum_{h_i \in H} P(+ | h_i) P(h_i | D) &= 0.4 \\ \sum_{h_i \in H} P(- | h_i) P(h_i | D) &= 0.6 \end{aligned}$$

ดังนั้น

$$\arg \max_{v_j \in V} \sum_{h_i \in H} P(v_j | h_i) P(h_i | D) = -$$

6.8.4 ตัวจำแนกประเภทเบสอย่างง่าย

ตัวจำแนกประเภทเบสอย่างง่าย (naive Bayes classifier) เป็นตัวจำแนกประเภทแบบหนึ่งที่ใช้งานได้ดี เนื่องจากมีข้อจำกัดตัวอย่างมีจำนวนมากและคุณสมบัติ (attribute) ของตัวอย่างไม่ขึ้นต่อ กัน มีการนำตัวจำแนกประเภทเบสอย่างง่ายไปประยุกต์ใช้งานในด้านการจำแนกประเภทข้อความ (text classification) การวินิจฉัย (diagnosis) และพบว่าใช้งานได้ดีไม่ต่างจากการจำแนกประเภทที่ใช้การเรียนรู้ด้วยตัวเอง ใช้งานง่ายและมีประสิทธิภาพ

สมมติให้ A_1, A_2, \dots, A_n เป็นคุณสมบัติของตัวอย่าง เราจะได้ว่าค่า (ประเภท) ที่น่าจะเป็นที่สุดของตัวอย่าง x คือ

$$v_{MAP} = \arg \max_{v_j \in V} P(v_j | a_1, a_2, \dots, a_n) \\ = \arg \max_{v_j \in V} \frac{P(a_1, a_2, \dots, a_n | v_j) P(v_j)}{P(a_1, a_2, \dots, a_n)} \quad (6.22)$$

$$v_{MAP} = \arg \max_{v_j \in V} P(a_1, a_2, \dots, a_n | v_j) P(v_j) \quad (6.23)$$

โดยที่ a_i ในสมการเป็นค่าของคุณสมบัติ A_i V เป็นเซตของประเภทหรือค่าที่เป็นไปได้ของ x สมการที่ (6.23) แสดงการหาประเภทที่ดีสุดของตัวอย่าง x แต่เราจะพบว่าสมการนี้ใช้งานไม่ได้อย่างมีประสิทธิภาพ เนื่องจากว่าการคำนวณค่าของ $P(a_1, a_2, \dots, a_n | v_j)$ ทำได้ยากลำบากมากเพื่อให้ได้ค่าที่น่าเชื่อถือในเชิงสถิติ ที่เป็นเช่นนี้ เพราะว่าถ้าให้คุณสมบัติ A_i แต่ละตัวของตัวอย่างมีค่าที่เป็นไปได้ 10 ค่า และคุณสมบัติทั้งหมดมี 10 ตัว เราจะได้ว่ามีลำดับ a_1, a_2, \dots, a_n ที่เป็นไปได้ทั้งสิ้นเท่ากับ 10^{10} รูปแบบ ซึ่งหมายถึงว่าเราต้องหาตัวอย่างทั้งสิ้น 10^{10} ตัว จึงจะมีโอกาสพบรูปแบบหนึ่งๆ ของ a_1, a_2, \dots, a_n มากที่สุด แต่เราต้องหาตัวอย่างมากกว่า 10^{10} ตัวหลายเท่า ซึ่งการที่จะหาตัวอย่างจำนวนมากขนาดนั้นแทบจะทำไม่ได้จริงในทางปฏิบัติ เราจึงต้องการโมเดลที่จะคำนวณ $P(a_1, a_2, \dots, a_n | v_j)$ ให้ได้ในเชิงปฏิบัติ

สมมติฐานของตัวจำแนกประเภทเบสอย่างง่ายคือ เรากำหนดให้คุณสมบัติแต่ละตัวไม่ขึ้น (เป็นอิสระ) กับคุณสมบัติอื่นๆ ซึ่งทำให้เราสามารถเขียนแทน $P(a_1, a_2, \dots, a_n | v_j)$ ด้วยผลคูณของค่าความน่าจะเป็นด้านล่างนี้ที่หาค่าได้ง่ายขึ้น

$$P(a_1, a_2, \dots, a_n | v_j) = \prod_{i=1}^n P(a_i | v_j) \quad (6.24)$$

โดยที่ \prod หมายถึงการนำค่า $P(a_i | v_j)$ ทั้งหมดมาคูณกัน สูตรนี้ถ้าใช้กับลูกโซ่มาคำนวณค่าความน่าจะเป็นที่ด้านซ้ายของสูตรจะได้เท่ากับ $P(a_1 | v_j) \times P(a_2 | a_1, v_j) \times P(a_3 | a_2, a_1, v_j) \times \dots \times P(a_n | a_{n-1}, a_{n-2}, \dots, a_1, v_j)$ ดังนั้นค่าความน่าจะเป็นทางด้านซ้ายของสมการจะเท่ากับผลคูณค่าความน่าจะเป็นทางด้านขวาที่ต่อเมื่อคุณสมบัติ a_1, a_2, \dots, a_n ไม่ขึ้นต่อกัน เช่นสิ่งไม่ขึ้นกับส่วนสูง ฯลฯ แต่ในความเป็นจริงแล้วคุณสมบัติส่วนใหญ่มักจะมีความสัมพันธ์กัน เช่นส่วนสูงกับน้ำหนัก เพราะถ้าตัวสูงน้ำหนักจะมากตามไปด้วย แต่อย่างไรก็ตามการใช้สมมติฐานความไม่ขึ้นต่อกัน (*conditional independence assumption*) นี้จะช่วยให้เราคำนวณค่าความน่าจะเป็นในสูตรที่ (6.24) ได้ง่ายขึ้น เพราะค่าความน่าจะเป็นของ a_i เมื่อรู้ v_j หาได้ง่ายกว่า เช่นถ้าจะหาคณิตมูลสี่เหลี่ยม ล่างส่วนสูงมาก น้ำหนักมาก และไม่ใช่โลหะไปด้วยกันแล้วผิวจะใหม่หรือไม่ เมื่อเวลาไปหาดูในฐานข้อมูลอาจจะมีโอกาส

สมมติฐาน

ความไม่ขึ้นต่อกัน

พบข้อมูลที่มีค่าครบทั้ง 4 ค่านี้น้อยมากๆ หรือต้องใช้จำนวนตัวอย่างมากมากมหาศาลถึงจะพบข้อมูลที่มีค่าครบตรงที่ต้องการ แต่ถ้าเราแยกคุณสมบัติออกจากกัน เช่น หาคนผอม สีน้ำตาลที่เป็นตัวอย่างบวก หรือหาคนไม่ใช่โลหันที่เป็นตัวอย่างบวก ทำให้ใช้ตัวอย่างไม่มาก และได้คำตอบ ถึงแม้ว่าคำตอบที่ได้อาจจะไม่ถูกต้องสมบูรณ์แต่ก็พบว่าทำงานได้ดีในทางปฏิบัติ

ดังนั้นเราจะได้ว่าตัวจำแนกประเภทแบบเบส์อย่างง่ายคือ

$$v_{NB} = \arg \max_{v_j \in V} P(v_j) \times \prod_{i=1}^n P(a_i | v_j) \quad (6.25)$$

จากสมการด้านบนนี้เราจะได้อัลกอริทึมการเรียนรู้เบส์อย่างง่ายดัง ตารางที่ 6-24

ตารางที่ 6-24 อัลกอริทึมการเรียนรู้เบส์อย่างง่าย

Algorithm: Naïve-Bayes

- **Naïve_Bayes_Learn(examples)**
FOR EACH target value v DO
 $\bar{P}(v_j) \leftarrow$ estimate $P(v_j)$
 FOR EACH attribute value a of each attribute DO
 $\bar{P}(a_i | v_j) \leftarrow$ estimate $P(a_i | v_j)$
- **Classify_New_Example(x)**

$$v_{NB} = \arg \max_{v_j \in V} \bar{P}(v_j) \times \prod_{i=1}^n \bar{P}(a_i | v_j)$$

ยกตัวอย่างการใช้อัลกอริทึมการเรียนรู้เบส์อย่างง่าย โดยใช้ชุดตัวอย่างสอนใน ตารางที่ 6-25 ต่อไปนี้

ตารางที่ 6-25 ตัวอย่างสอนสำหรับการเรียนรู้เบส์อย่างง่าย (เหมือนกับ ตารางที่ 6-13)

attribute →	class ↓					
value ↴	Name	Hair	Height	Weight	Lotion	Result
Sarah	blonde	average	light	no	Sunburned	
Dana	blonde	tall	average	yes	none	
Alex	brown	short	average	yes	none	
Annie	blonde	short	average	no	sunburned	
Emily	red	average	heavy	no	sunburned	
Pete	brown	tall	heavy	no	none	
John	brown	average	heavy	no	none	
Katie	blonde	short	light	yes	none	

สมมติว่าตัวอย่างที่ต้องการจำแนกประเภทคือ

Name	Hair	Height	Weight	Lotion	Result
Judy	blonde	average	heavy	no	?

คำนวณ $v_{NB} = \arg \max_{v_j \in V} P(v_j) \times \prod_{i=1}^n P(a_i | v_j)$ โดย $V = \{+, -\}$ เราจะได้ดังต่อไปนี้
กรณี $v_j = +$ ได้ว่า

$$P(+)P(\text{blonde}|+)P(\text{average}|+)P(\text{heavy}|+)P(\text{no}|+) = \frac{3}{8} \times \frac{2}{3} \times \frac{2}{3} \times \frac{1}{3} \times \frac{3}{3} = \frac{1}{18}$$

ส่วนกรณี $v_j = -$ ได้ว่า

$$P(-)P(\text{blonde}|-)P(\text{average}|-)P(\text{heavy}|-)P(\text{no}|-) = \frac{5}{8} \times \frac{2}{5} \times \frac{1}{5} \times \frac{2}{5} \times \frac{2}{5} = \frac{1}{125}$$

ดังนั้นได้ $v_{NB} = +$

การเรียนรู้เพื่อจำแนกประเภทข้อความโดยเบส์อย่างง่าย

ในการเรียนรู้เพื่อจำแนกประเภทข้อความโดยใช้เบส์อย่างง่ายนี้ สมมติว่าเรามีข้อความที่เราสนใจกับไม่สนใจ เมื่อทำการเรียนรู้แล้วเราต้องการท่านายว่าเอกสารหนึ่งๆ จะเป็นเอกสารที่เราสนใจหรือไม่ สามารถนำประยุกต์ใช้งาน เช่น การกรองข่าวสารเลือกเฉพาะข่าวที่สนใจ เป็นต้น

ก่อนอื่นเราให้เอกสารหนึ่งๆ คือตัวอย่างหนึ่งตัว และเราแทนเอกสารแต่ละฉบับด้วย เวกเตอร์ของคำโดยใช้คำที่ปรากฏในเอกสารเป็นคุณสมบัติของเอกสาร กล่าวคือคำที่หนึ่งในเอกสารเป็นคุณสมบัติตัวที่หนึ่ง คำที่สองในเอกสารเป็นคุณสมบัติตัวที่สอง ตามลำดับ ดังนั้นจะได้ว่า a_1 คือคำที่หนึ่ง a_2 คือคำที่สองตามลำดับ จากนั้นก็ทำการเรียนรู้โดยใช้ตัวอย่างสอนเพื่อประมาณค่าความน่าจะเป็นต่อไปนี้คือ

1. $P(+)$
2. $P(-)$
3. $P(\text{doc}|+)$
4. $P(\text{doc}|-)$

จากสมมติฐานเรื่องความไม่ชี้นต่อ กันของคุณสมบัติของเบส์อย่างง่ายทำให้เราได้ว่า

$$P(\text{doc} | v_j) = \prod_{i=1}^{\text{length}(\text{doc})} P(a_i = w_k | v_j) \quad (6.26)$$

เมื่อ a_i คือคุณสมบัติตัวที่ i ส่วนค่าของมันคือ w_k (คำที่ k ในรายการของคำที่เรามีอยู่) $P(a_i = w_k | v_j)$ คือความน่าจะเป็นที่คำในตำแหน่งที่ i เป็น w_k เมื่อรู้ v_j แต่พบว่าสูตรนี้ก็ยังนำไปคำนวณยากเนื่องจากเหตุผลในทำนองเดียวกันกับสมมติฐานความไม่ชี้นต่อ กันข้างต้น

จึงสร้างสมมติฐานเพิ่มเติมดังสมการที่ (6.27) เพื่อให้การคำนวณทำได้มีประสิทธิภาพในทางปฏิบัติ

$$P(a_i = w_k | v_j) = P(a_m = w_k | v_j), \forall i, m \quad (6.27)$$

หมายความว่าโอกาสที่เราจะเห็นคำที่หนึ่งไปปรากฏที่ตำแหน่งใดๆ มีค่าเท่ากันหมด ทำให้การคำนวณง่ายขึ้น เพราะไม่ต้องสนใจว่าคำหนึ่งๆ จะไปปรากฏในตำแหน่งใด หรือคำแต่ละคำจะไม่ขึ้นกับตำแหน่ง อัลกอริทึมสำหรับการจำแนกประเภทข้อความโดยใช้การเรียนรู้เบส์อย่างง่ายเป็นดังตารางที่ 6-26 ต่อไปนี้

ตารางที่ 6-26 อัลกอริทึมการเรียนรู้เบส์อย่างง่ายสำหรับจำแนกประเภทข้อความ

Algorithm: `Learn_naive_Bayes_text(Examples, V)`

1. Collect all words and other tokens that occur in *Examples*.
 - *Vocabulary* \leftarrow all distinct words and other tokens in *Examples*.
2. Calculate the required $P(v_j)$ and $P(w_k | v_j)$:
 - **FOR EACH** target value v_j in *V* **DO**
 - $docs_j \leftarrow$ subset of *Examples* for which the target value is v_j
 - $P(v_j) = \frac{|docs_j|}{Examples}$
 - $Text_j \leftarrow$ a single document created by concatenating all members of $docs_j$
 - $n \leftarrow$ total number of words in $Text_j$ (counting duplicate words multiple times)
 - **FOR EACH** word w_k in *Vocabulary* **DO**
 - $n \leftarrow$ number of times word w_k occurs in $Text_j$
 - $P(w_k | v_j) = \frac{n_k + 1}{n + |Vocabulary|}$

Algorithm: `classify_naive_Bayes_text(Doc)`

- $positions \leftarrow$ all word positions in *Doc* that contain tokens found in *Vocabulary*
- **Return** v_{NB}

$$v_{NB} = \arg \max_{v_j \in V} P(v_j) \times \prod_{i \in positions} P(a_i | v_j)$$

6.8.5 ข่ายงานความเชื่อเบส์

ข่ายงานความเชื่อเบส์ (*Bayesian belief network*) หรือเรียกโดยย่อว่า ข่ายงานเบส์ (*Bayes net*) เป็นวิธีการเรียนรู้ที่ลดข้อจำกัดของการเรียนรู้เบส์อย่างง่ายในสมมติฐานของความไม่ชัดต่อ กันระหว่างคุณสมบัติ ในวิธีการเรียนรู้เบส์อย่างง่ายในหัวข้อที่แล้วจะตั้งสมมติฐานว่า คุณสมบัติใดๆ ไม่ขึ้นต่อ กัน แต่ในความเป็นจริงเราพบว่า คุณสมบัติบางตัวจะขึ้นต่อ กัน บ้าง และควรที่จะนำความขึ้นต่อ กันนี้เข้ามาใส่ไว้ในโมเดลด้วย เราจึงใช้ ข่ายงานความเชื่อเบส์ ใน การอธิบายความไม่ชัดต่อ กันอย่างมีเงื่อนไข (condition independent) ระหว่างตัวแปร (ใน บริบทของ ข่ายงานความเชื่อเบส์นิยมใช้คำว่า 'ตัวแปร' (variable) แทนคำว่า 'คุณสมบัติ') และในโมเดลนี้เราสามารถใช้ (1) ความรู้ก่อน (prior knowledge) เกี่ยวกับความ('ไม่')ขึ้นต่อ กันระหว่างตัวแปร ร่วมกับ (2) ตัวอย่างสอน เพื่อทำให้กระบวนการเรียนรู้มีประสิทธิภาพ โดยเราสามารถใส่ความรู้ก่อนใน ข่ายงานความเชื่อเบส์ให้อยู่ในรูปของโครงสร้าง ข่ายงาน และตารางความน่าจะเป็นมีเงื่อนไข ดังจะกล่าวต่อไป

ก่อนอื่นเรานิยามความไม่ชัดต่อ กันอย่างมีเงื่อนไขดังนี้

ความไม่ชัดต่อ กัน
อย่างมีเงื่อนไข

นิยามที่ 5.1 ความไม่ชัดต่อ กันอย่างมีเงื่อนไข

X ไม่ขึ้นกับ Y อย่างมีเงื่อนไข เมื่อรู้ Z ถ้าความน่าจะเป็นของ X ไม่ขึ้นกับค่าของ Y เมื่อรู้ค่า ของ Z นั่นคือ

$$(\forall x_i, y_j, z_k) P(X=x_i | Y=y_j, Z=z_k) = P(X=x_i | Z=z_k)$$

หรือในรูปง่าย

$$P(X | Y, Z) = P(X | Z) \quad \square$$

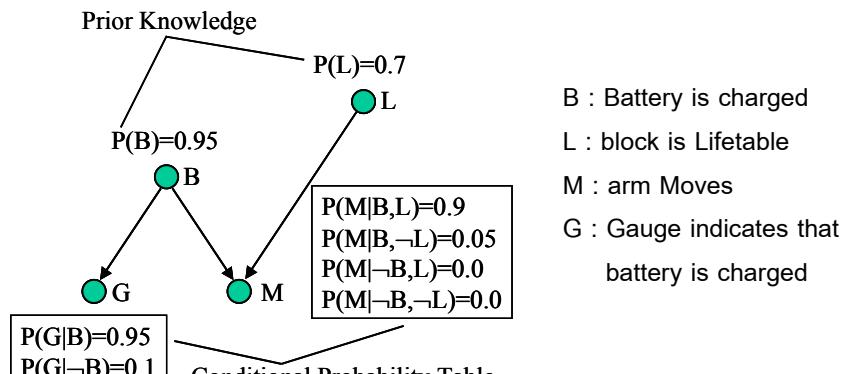
นิยามด้านบนนี้หมายความว่า สำหรับ x_i, y_j, z_k ใดๆ ความน่าจะเป็นที่ X จะมีค่าเป็น x_i (X เป็นตัวแปรส่วน x_i คือค่าของมัน) เมื่อรู้ว่า Y มีค่าเป็น y_j และ Z มีค่าเป็น z_k จะมีค่าเท่ากับ ความน่าจะเป็นของ X จะมีค่าเป็น x_i เมื่อรู้ว่า Z มีค่าเป็น z_k ในกรณีที่ความน่าจะเป็นทั้งสองเท่ากัน เช่นนี้ เราเรียกว่า ค่าของ X ไม่ขึ้นกับค่าของ Y อย่างมีเงื่อนไข เมื่อรู้ค่าของ Z เรา จึงสามารถตัด Y ทิ้งไปได้

ตัวอย่าง เช่น พาร้องไม่ขึ้นกับฝนตกถ้ารู้ว่า พาрапาน หรือ เรียกได้เป็น

$$P(\text{Thunder} | \text{Rain, Lighting}) = P(\text{Thunder} | \text{Lightning})$$

ดังนั้นถ้ามี พาрапาน และสามารถบอกได้เลยว่า จะต้องได้ยินเสียงพาร้องด้วยความน่าจะเป็นเท่าไร โดยไม่ต้องสนใจว่าเกิดฝนตกหรือไม่

จากความไม่ชัดต่อ กันอย่างมีเงื่อนไขข้างต้น เราสร้างข่ายงานของเบส์ได้ดังตัวอย่างในรูปที่ 6-46 ต่อไปนี้



รูปที่ 6-46 ตัวอย่างของข่ายงานเบส์

จากรูปจะเห็นได้ว่าข่ายงานประกอบด้วยบพหลายบพ บพแต่ละบพหมายถึงคุณสมบัติของข้อมูลหรือตัวแปร และบพแต่ละบพจะมี**ตารางความน่าจะเป็นเมื่อเงื่อนไข** – **ซีพีที (conditional probability table – CPT)** ติดอยู่ด้วย ข่ายงานเบส์นี้แสดงในรูปของกราฟมีทิศทางซึ่งสามารถอภิได้ว่ามีตัวแปรใดบ้างที่ขึ้นกับตัวแปรอื่น และตัวแปรตัวใดบ้างที่ไม่ขึ้นกับตัวอื่น ตัวอย่างเช่นบพ M ขึ้นกับบพ B และบพ L หรือถ้ามองเป็นลักษณะความสัมพันธ์ของบพฟ่อแม่กับบพลูกจะเห็นว่าบพฟ่อแม่ของ M คือ B และ L ส่วนบพฟ่อแม่ของ G คือ B และสามารถอภิได้ว่าบพ G จะไม่ขึ้นกับบพ L ถ้ารู้ B และได้ว่า G ไม่ขึ้นกับ M เมื่อรู้ B (สมมติว่าตัวแปรทั้งสี่คือ G, M, B และ L เป็นตัวแปรแบบบูล และเขียนแทนค่าของตัวแปรอย่างง่ายโดยใช้ตัวแปรนั้นแทนค่าจริงและใส่เครื่องหมาย — แทนค่าเท็จ เช่น G แทนค่าตัวแปร G เป็นจริง ส่วน —G แทนค่าตัวแปร G เป็นเท็จ)

บพใดๆ จะไม่ขึ้นกับบพอื่นถ้ารู้บพฟ่อแม่โดยตรงของมัน จึงได้ว่า G จะไม่ขึ้นกับบพอื่นถ้ารู้บพ B ส่วน L ไม่มีบพฟ่อแม่ แสดงว่า L ไม่ขึ้นกับบพอื่นๆ เช่นเดียวกับบพ B ก็ไม่ขึ้นกับบพอื่น ส่วน M ขึ้นกับ B และ L

จากข่ายงานเบส์ข้างต้น สมมติว่าเรากำลังจะเขียนข่ายงานที่อธิบายหุ่นยนต์ตัวหนึ่งที่กำลังจะบ่ายของในโดเมนโลกของบล็อก หุ่นยนต์ตัวนี้จะาร์จแบตเตอรี่และมีเกจ (G) อยู่ วัดว่าขณะนี้แบตเตอรี่เหลืออยู่หรือไม่ หุ่นยนต์ทำงานด้วยการเคลื่อนแขนไปยกบล็อก เมื่อเราจำลองเหตุการณ์นี้ในข่ายงานเบส์จะได้ว่าแบตเตอรี่ (B) จะส่งผลต่อเกจ G นอกจักนั้นยังส่งผลต่อ M (การเคลื่อนแขนของหุ่นยนต์) และเราได้ใส่ความรู้ก่อนหน้าเข้าไปในรูปของ

ตารางความน่าจะเป็นมีเงื่อนไขว่า 70% ของบล็อกทั้งหมดสามารถยกได้ ($P(L)=0.7$) และในเวลา 100 ชั่วโมงมี 95 ชั่วโมงที่แบบเตอร์มีไฟ ($P(B)=0.95$)

เมื่อคุณที่ซื้อพิมพ์ของ G พบว่า ถ้าแบบเตอร์มีไฟ เกจซึ่งมีความบกพร่องอยู่บ้างนี้จะแสดงผลว่ามีไฟด้วยความน่าจะเป็นเท่ากับ 0.95 ($P(G|B)=0.95$) และถ้าไฟหมดแต่เกจยังแสดงว่ามีไฟด้วยความน่าจะเป็นเท่ากับ 0.1 ($P(G|\neg B)=0.1$)

ในตารางซึ่พิมพ์ของบัพ M นั้น ตัวแรก $P(M|B,L)=0.9$ หมายความว่าหุ่นยนต์จะเคลื่อนแขวนถ้าแบบเตอร์มีไฟและบล็อกสามารถยกได้ และถ้ามีไฟแต่บล็อกไม่สามารถยกได้แขวนจะเคลื่อนด้วยความน่าจะเป็น 0.05 ($P(M|\neg B,L)=0.05$) ถ้าไม่มีไฟและบล็อกสามารถยกได้หุ่นยนต์ก็จะไม่เคลื่อนแขวน ($P(M|\neg B, \neg L)=0.0$) และสุดท้ายถ้าบล็อกยกไม่ได้และไฟไม่มีแขวนก็จะไม่เคลื่อนแขวนกัน ($P(M|\neg B, \neg L)=0.0$)

ทั้งหมดนี้คือความน่าจะเป็นทั้งหมดที่เราป้อนให้ระบบในรูปของซีพีที ผู้ที่ป้อนข้อมูลคือผู้เชี่ยวชาญที่ทำงานกีฬากับหุ่นยนต์ เมื่อเราทราบค่าต่างๆ ทั้งหมดเราจะสามารถที่จะคำนวณความน่าจะเป็นต่างๆ ที่จะเกิดขึ้นภายในระบบนี้ได้ เช่น ถ้าต้องการคำนวณหาว่าความน่าจะเป็นที่ แบบเตอร์มีไฟ บล็อกสามารถยกได้ เกจขึ้นและหุ่นยนต์เคลื่อนแขวน ทั้งสี่เหตุการณ์เกิดขึ้นพร้อมกันว่ามีค่าเท่าไรก็สามารถคำนวณได้จากข้างบนนี้

ความน่าจะเป็นร่วม (Joint probability) ระหว่างตัวแปรคือความน่าจะเป็นที่ตัวแปรหลายตัวจะมีค่าตามที่กำหนด เช่น $P(\text{Battery}, \text{Liftable}, \text{Gauge}, \text{Move})$ เป็นต้น เราเขียนความน่าจะเป็นร่วมให้อยู่ในรูปทั่วไปได้เป็น

$$P(y_1, \dots, y_n) = \prod_{i=1}^n P(y_i | \text{Parents}(Y_i)) \quad (6.28)$$

โดยที่ $\text{Parents}(Y_i)$ หมายถึง บัพพ่อแม่โดยตรงของบัพ Y_i ถ้าเราต้องการจะหาความน่าจะเป็นที่ y_1, \dots, y_n เกิดขึ้นพร้อมกันสามารถคำนวณได้จากความน่าจะเป็นของ y_1 คูณกับความน่าจะเป็นของ y_2 คูณไปเรื่อยๆ จนถึง y_n แต่ต้องดูว่าบัพแต่ละบัพขึ้นกับบัพพ่อแม่ตัวใดบ้าง เช่น y_1 ขึ้นกับบัพใด y_2 ขึ้นกับบัพใด เป็นต้น ยกตัวอย่างเช่นจากรูปที่ 6-46

$$\begin{aligned} P(G, M, B, L) &= P(G|B, M, L)P(M|B, L)P(B|L)P(L) \\ &= P(G|B)P(M|B, L)P(B|L)P(L) \\ &= (0.95)(0.9)(0.95)(0.7) \\ &= 0.57 \end{aligned}$$

สังเกตได้ว่าบรรทัดแรกใช้กฎโซนิคกระจาย $P(G, M, B, L)$ อกมาเป็นด้านขวามือ และเมื่อกระจายแล้วจะเห็นว่าตัวแปรบางตัวไม่ขึ้นกับตัวอื่น เช่นเราสังเกตได้ว่า G จะขึ้นกับ B ตัวเดียวไม่ขึ้นกับ M หรือ L ดังนั้น $P(G|B, M, L)$ จึงลดรูปลงมาเหลือเป็น $P(G|B)$ เท่านั้น และ B ไม่ขึ้นกับ L ดังนั้น $P(B|L)$ จึงเหลือแค่ $P(B)$ พอลดรูปคงทุกตัวก็นำมาใส่ไว้ในสมการแล้วหาผลลัพธ์อกมา จะสังเกตได้ว่าเมื่อลดรูปลงมาแล้วบัพที่เราสนใจจะขึ้นกับพ่อแม่ของมันเท่านั้น เช่น $P(G|B, M, L)$ ก็จะเหลือ $P(G|B)$ หรือหากว่าน่าจะเป็นของ G เมื่อรู้ B กรณีตัวอย่างที่ยกมาเป็นกรณีง่ายๆ เพราะเรารู้ค่าความน่าจะเป็นครบทั้งสี่ตัวแล้ว แต่ในบางกรณี เช่นเราทราบค่าของตัวแปรเพียงแค่ 2 ตัว หรือ 3 ตัว ไม่ใช้ทั้งหมดก็สามารถใช้เทคนิคในการอนุมานของข่ายงานเบสเพื่อหาความน่าจะเป็นร่วมได้เช่นกัน ดังจะได้อธิบายด้านล่างนี้ ซึ่งเป็นเทคนิคการอนุมานที่ใช้ทั่วไปสำหรับข่ายงานเบส 3 เทคนิคเพื่อหาความน่าจะเป็นของตัวแปรที่เราสนใจ

1. **การอนุมานจากเหตุ (causal reasoning):** เมื่อเราทราบเหตุ เราสามารถหาได้ว่าผลจะเกิดขึ้นด้วยความน่าจะเป็นเท่าไร เช่น $P(M|L)$ หรือความน่าจะเป็นที่แขนจะเคลื่อนเมื่อรู้ว่าบล็อกสามารถยกได้ (บล็อกยกได้เป็นสาเหตุหนึ่งของการที่หุ่นยนต์จะเคลื่อนแขน) แต่เราไม่ทราบค่า B (ไม่ทราบว่าขั้นตอนนี้แบบเดอร์มไฟหรือไม่) ถ้าเราย้อนกลับไปดูในข่ายงานเบสในรูปที่ 6-46 จะเห็นว่าเราไม่สามารถคำนวณ $P(M|L)$ ได้โดยตรง เพราะไม่มีค่าบวกไว้ในตาราง ในตารางมีค่าที่ไม่เกี่ยวกับ L ดังนั้นเราต้องพยายามกระจาย $P(M|L)$ ให้อยู่ในรูปที่เกี่ยวข้อง ในที่นี้จะใช้ทฤษฎีความน่าจะเป็นทั้งหมดที่กล่าวว่า ถ้าเหตุการณ์ A_1, \dots, A_n ไม่เกิดร่วมกันและ $\sum P(A_i) = 1$ และ $P(B) = \sum P(B|A_i)P(A_i) = \sum P(B, A_i)$ เราสามารถนำ A_1, \dots, A_n กระจายเข้ามาได้ ดังนั้นเราสามารถกระจาย $P(M|L)$ ให้อยู่ในรูปของผลรวมของความน่าจะเป็นร่วมระหว่าง M กับบัพพ่อแม่อื่นนอกจาก L (ซึ่งก็คือ B ที่เป็นบัพพ่อแม่ของ M ด้วย) ได้เป็น

$$P(M|L) = P(M, B|L) + P(M, \neg B|L)$$

เมื่อกระจายแล้วก็ยังพบว่าเรายังไม่ทราบค่าของ $P(M, B|L)$ อญี่ สิ่งที่เรารู้คือ $P(M|B, L)$ เราจึงต้องใช้กฎโซนิคกระจายแต่ละตัวได้เป็น

$$\begin{aligned} P(M|L) &= P(M|B, L)P(B|L) + P(M|\neg B, L)P(\neg B|L) \\ &= P(M|B, L)P(B) + P(M|\neg B, L)P(\neg B) \\ &= (0.9)(0.95) + (0.0)(0.05) \\ &= 0.855 \end{aligned}$$

2. **การอนุมานจากผล (diagnosis reasoning):** ข้อนี้จะตรงข้ามกับข้อแรก กล่าวคือเราทราบผลแล้วแต่อยากรู้ว่าสาเหตุจะเกิดขึ้นด้วยความน่าจะเป็นเท่าไร เช่นต้องการคำนวณ $P(\neg L | \neg M)$ หรือความน่าจะเป็นที่บล็อกยกไม้ได้เมื่อรู้ว่าแขนไม้ได้เคลื่อนและขาไม่ได้โดยตรง ในการนี้เราใช้ทฤษฎีของเบสตังที่กล่าวในตอนแรกว่า

$$P(A | B) = \frac{P(B | A)P(A)}{P(B)} \quad \text{ดังนั้น} \quad P(\neg L | \neg M) = \frac{P(\neg M | \neg L)P(\neg L)}{P(\neg M)}$$

ในส่วนของ $P(\neg M | \neg L)$ สามารถคำนวณได้โดยใช้การอนุมานจากเหตุในข้อที่แล้ว (ลองคำนวณดู) จะได้ $P(\neg M | \neg L) = 0.9525$ ส่วน $P(\neg L) = 0.3$ ดังนั้นจะได้ว่า $P(\neg L | \neg M) = \frac{0.9525 \times 0.3}{P(\neg M)}$ และพบยังมี $P(\neg M)$ ที่ยังไม่ทราบค่าอีก ซึ่งการหาค่า

โดยตรงค่อนข้างยุ่ง เราจึงไปหาค่า $P(L | \neg M)$ เนื่องจาก $P(\neg L | \neg M) + P(L | \neg M) = 1$

$$\text{ดังนั้นเราจะได้ว่า} \quad P(L | \neg M) = \frac{P(\neg M | L)P(L)}{P(\neg M)} = \frac{0.145 \times 0.7}{P(\neg M)} = \frac{0.1015}{P(\neg M)}$$

จาก $P(\neg L | \neg M) + P(L | \neg M) = 1$ ซึ่งจะทำให้เราหาค่าของ $P(\neg M)$ ได้แล้วก็นำไปแทนค่าได้ $P(\neg L | \neg M) = 0.88632$

3. **การอธิบายลดความเป็นไปได้ (explaining away):** เป็นการทำการอนุมานจากเหตุภัยในการอนุมานจากผล เป็นการสมมูลว่า วิธีการทั้งสองแบบข้างต้น เช่นถ้าเราทราบ $\neg M$ (แขนไม่เคลื่อน) เราสามารถคำนวณ $\neg L$ หรือความน่าจะเป็นที่บล็อกไม้สามารถยกได้ แต่ถ้าเรารู้ $\neg B$ และ $\neg L$ ควรจะมีความน่าจะเป็นน้อยลง ในการนี้เรียกว่า $\neg B$ อธิบาย $\neg M$ ทำให้ $\neg L$ มีความเป็นไปได้น้อยลง

$$\begin{aligned} P(\neg L | \neg B, \neg M) &= \frac{P(\neg B, \neg M | \neg L)P(\neg L)}{P(\neg B, \neg M)} \\ &= \frac{P(\neg M | \neg B, \neg L)P(\neg B | \neg L)P(\neg L)}{P(\neg B, \neg M)} \\ &= \frac{P(\neg M | \neg B, \neg L)P(\neg B)P(\neg L)}{P(\neg B, \neg M)} \end{aligned}$$

หลังคำนวณ $P(\neg B, \neg M)$ เราจะได้ $P(\neg L | \neg B, \neg M) = 0.03$

6.8.6 การเรียนรู้ข่ายงานเบส

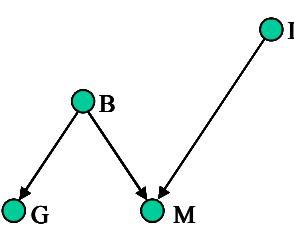
การเรียนรู้ข่ายงานเบสคือการหาโครงสร้างข่ายงานและ/หรือซีพีที่ที่สอดคล้องกับตัวอย่างสอนมากที่สุด ปัญหาการเรียนรู้ข่ายงานเบสแบ่งออกเป็นกรณีดังต่อไปนี้

1. โครงสร้างไม่รู้ (structure unknown)
2. โครงสร้างรู้ (structure known)
 - 2.1 ข้อมูลมีค่าครบ (no missing value data)
 - 2.2 ข้อมูลมีค่าหาย (missing value data)

กรณีที่ 1 เป็นกรณียากที่สุด เพราะเรามีรู้ว่าโครงสร้างของข่ายงานเบสมีรูปร่างเป็นอย่างไร มีการเชื่อมต่อระหว่างบัพอย่างไร และแน่นอนว่าเรามีรู้ค่าในซีพีที่อีกด้วย ดังนั้นการเรียนรู้ต้องคำนึงเหตุทั้งโครงสร้างข่ายงานและซีพีที่ ส่วนกรณีที่สองเป็นกรณีที่รู้โครงสร้างแล้ว ซึ่งบอยครั้งผู้เขียนข่ายงานเบสเป็นผู้เชี่ยวชาญในปัญหานั้นสามารถออกโครงสร้างได้อย่างชัดเจน รู้ความสัมพันธ์ระหว่างตัวแปรในปัญหานั้นแต่อาจไม่รู้ค่าที่ถูกต้องและแม่นยำในตารางซีพีที่ ดังนั้นกรณีนี้การเรียนรู้เป็นการหาค่าในซีพีที่โดยอาศัยตัวอย่างสอน กรณีที่สองนี้ยังแบ่งเป็นกรณีที่มีอยู่อีกสองกรณีคือ กรณีที่ข้อมูลหรือตัวอย่างสอนทุกตัวมีค่าครบถ้วน กับ กรณีที่ตัวอย่างสอนบางตัวหรือทุกตัวมีค่าบางส่วนหายไป เช่นไม่มีค่าของคุณสมบัติบางตัว เป็นต้น กรณีที่ 2.1 เป็นกรณีที่ง่ายสุดสามารถทำการเรียนรู้ได้ในลักษณะเดียวกับการเรียนรู้ของตัวจำแนกประเภทเบสอย่างง่าย โดยนับจำนวนครั้งที่เกิดขึ้นของข้อมูลเพื่อไปคำนวณซีพีของแต่ละบัพว่ามีค่าเท่าไรดังจะแสดงต่อไปนี้ ส่วนกรณีที่ 1 ไม่ขออธิบายในที่นี้

การเรียนรู้ข่ายงานเบสในกรณีที่รู้โครงสร้างและข้อมูลครบ

ดูตัวอย่างต่อไปนี้ เรานับความถี่ของการเกิดค่าต่างๆ ของ G, M, B, L ว่าเกิดขึ้นกี่ครั้ง ได้ดังรูปที่ 6-47 โดยที่สมมติว่าโครงสร้างถูกกำหนดแล้วดังรูปที่ 6-47



G	M	B	L	จำนวนตัวอย่าง
True	True	True	True	54
True	True	True	False	1
True	False	True	True	7
True	False	True	False	27
False	True	True	True	3
False	False	True	False	2
False	False	False	True	4
False	False	False	False	2

100

รูปที่ 6-47 ตัวอย่างสอนสำหรับเรียนรู้ซีพีที่ในกรณีข้อมูลครบ

$$\text{จาก } P(V_i=v_i | \text{Parents}(V_i)=P_i) = \frac{\text{จำนวนตัวอย่างที่มี } V_i = v_i}{\text{จำนวนตัวอย่างที่มี } \text{Parents}(V_i)=P_i}$$

ดังนั้นจะได้ค่าความน่าจะเป็นต่างๆ ดังนี้

$$P(B=\text{true}) = (54+1+7+27+3+2)/100 = 0.94$$

คือนับจำนวนตัวอย่างที่ B เป็นจริงในตารางหารด้วยจำนวนตัวอย่างทั้งหมด ค่าความน่าจะเป็นอื่นๆ ก็คำนวณในทำนองเดียวกัน

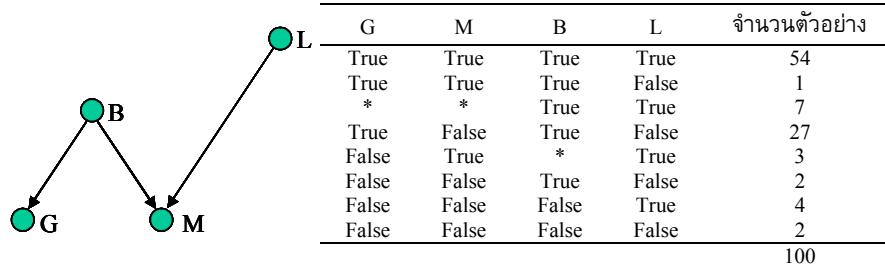
$$P(L=\text{true}) = (54+7+3+4)/100 = 0.68$$

$$P(M|B,L) \text{ เท่ากับอัตราส่วนที่ } M=\text{true} \text{ เมื่อ } B=\text{true}, L=\text{false} \text{ เท่ากับ } 1/(1+27+2) = 0.03$$

ด้วยวิธีนี้เราสามารถนำไปคำนวณหาความน่าจะเป็นของบัพ G ได้เช่นเดียวกัน

การเรียนรู้ข่ายงานเบส์ในกรณีที่โครงสร้างรู้และข้อมูลมีค่าหาย

กรณีต่อไปที่จะพิจารณาคือกรณีที่ข้อมูลบางตัวมีค่าบางค่าหายไปดังแสดงในรูปที่ 6-48 โดยที่สมมติว่ารู้โครงสร้างของข่ายงานเบส์แล้ว



รูปที่ 6-48 ตัวอย่างสอนสำหรับเรียนรู้ซีพีที่ในกรณีข้อมูลมีค่าหาย

‘*’ ในตารางหมายถึงค่าหายไป พิจารณาแล้วที่ห้าของข้อมูลในรูปซึ่งเป็นกรณีของตัวอย่าง 3 ตัวที่มีค่า $G=\text{false}$, $M=\text{true}$, $L=\text{true}$ ในกรณีนี้เราไม่รู้ค่าของ B แต่อาจคำนวณ $P(B|\neg G, M, L)$ หรือ $P(\neg B|\neg G, M, L)$ ได้ถ้าหากเรารู้ซีพีที่ (แต่รายั่งไม่รู้) สมมติว่าเรารู้ซีพีที่ซึ่งจะทำให้เราหาความน่าจะเป็นที่ B จะเป็นจริง (หรือเท็จ) ของตัวอย่างทั้ง 3 ตัวได้จากนั้นเราจะแทนที่ตัวอย่างทั้งสามนี้ด้วยตัวอย่างมีน้ำหนัก (weighted example) 2 ตัว ดังนี้

- ตัวแรกคือตัวอย่างที่ $B=\text{true}$ มีน้ำหนักเท่ากับ $P(B|\neg G, M, L)$
- ตัวที่สองคือตัวอย่างที่ $B=\text{false}$ มีน้ำหนักเท่ากับ $P(\neg B|\neg G, M, L)$

ในทำนองเดียวกัน กรณีของตัวอย่าง 7 ตัวในແກ່ที่สองที่มีค่า $B=\text{true}$, $L=\text{true}$ ส่วน G และ M ไม่รู้ค่า นั้น เราสามารถแทนที่ตัวอย่างทั้งเจ็ดตัวด้วยตัวอย่างมีน้ำหนัก 4 ตัว ดังนี้

- ตัวอย่างที่ 1 คือตัวอย่างที่ $G=true, M=true$ มีน้ำหนักเท่ากับ $P(G, M|B, L)$
- ตัวอย่างที่ 2 คือตัวอย่างที่ $G=true, M=false$ มีน้ำหนักเท่ากับ $P(G, \neg M|B, L)$
- ตัวอย่างที่ 3 คือตัวอย่างที่ $G=false, M=true$ มีน้ำหนักเท่ากับ $P(\neg G, M|B, L)$
- ตัวอย่างที่ 4 คือตัวอย่างที่ $G=false, M=false$ มีน้ำหนักเท่ากับ $P(\neg G, \neg M|B, L)$

ดังที่ได้กล่าวข้างต้นว่าเราสามารถหาค่าน้ำหนักทั้งสองค่าของตัวอย่าง 3 ตัวด้านบนกับค่าน้ำหนักทั้งสี่ค่าของตัวอย่าง 7 ตัวนี้ได้ถ้าเรารู้ค่าความน่าจะเป็นในซีพีที่ จากนั้นเราจะใช้ตัวอย่างมีน้ำหนักเหล่านี้ร่วมกับตัวอย่างที่เหลือใน [รูปที่ 6-48](#) เพื่อคำนวณซีพีที่ซึ่งเป็นสิ่งที่เราต้องการเรียน (ตัวอย่างที่ไม่รู้ค่าถูกแทนที่ด้วยตัวอย่างมีน้ำหนัก) แต่ต่ออย่างไรก็ต้องเราจะทำเช่นนี้ได้โดยมีเงื่อนไขว่าเราต้องรู้ค่าในซีพีที่ก่อน ซึ่งเรายังไม่รู้

วิธีการทำก็คือเราจะสมมติค่าความน่าจะเป็นในซีพีที่โดยสุ่มค่าเริ่มต้นเข้าไปในซีพีที่ ซึ่งก็จะสมมุนว่าเรามีค่าในซีพีที่แล้ว และเราจะสามารถหาค่าน้ำหนักของตัวอย่างไม่ทราบค่าได้ทุกตัว ก็จะทำให้เขตตัวอย่างเดิมที่มีตัวอย่างไม่รู้ค่าเป็นเขตตัวอย่างที่เรารู้ค่าทุกตัว การเรียนรู้ ก็จะเหมือนกับกรณีที่ตัวอย่างมีข้อมูลครบ แต่ตอนนี้การคำนวณค่าน้ำหนักจะไม่ได้ค่า น้ำหนักที่ถูกต้อง เพราะว่าเราสุ่มซีพีที่เริ่มต้นที่ไม่ใช่ซีพีที่ที่ถูก แต่เนื่องจากว่าเมื่อเราได้ น้ำหนักแล้วนำตัวอย่างไปรวมกับตัวอย่างที่เหลือที่เป็นตัวอย่างมีข้อมูลครบ ก็จะทำให้การ ประมาณค่าซีพีที่ครั้งใหม่มีความถูกต้องเพิ่มขึ้นกว่าซีพีที่เริ่มต้น เพราะว่าตัวอย่างส่วนใหญ่ ของเรามีตัวอย่างที่ถูกต้อง จะมีตัวอย่างมีน้ำหนักเท่านั้นที่ไม่ถูกต้องสมบูรณ์ แสดงว่าการ ปรับค่าซีพีที่ทำให้ได้ซีพีที่ใหม่ที่ดีขึ้น และถ้าเราทำซ้ำกระบวนการเดิมด้วยซีพีที่ที่ดีขึ้นก็จะ ทำให้การหาค่าน้ำหนักมีความแม่นยำขึ้น และส่งผลให้การปรับซีพีที่ในรอบต่อไปดีขึ้นอีก เมื่อวันซ้ำไปเรื่อยๆ ก็จะได้ซีพีที่ที่ดีขึ้นเรื่อยๆ จนกระทั่งซีพีที่ไม่เปลี่ยนแปลง เราก็หยุด กระบวนการเรียนรู้ได้ อัลกอริทึมการเรียนรู้แบบนี้เรียกว่า [อัลกอริทึมอีเมม \(EM – expectation maximization algorithm\)](#) [Dempster, et al., 1977; McLachlan & Krishnan, 1996] ซึ่งแสดงในตารางที่ 6-27 ต่อไปนี้

ตารางที่ 6-27 อัลกอริทึมอีเมมสำหรับคำนวณค่าน้ำหนักของตัวอย่างไม่รู้ค่า

Algorithm: EM

1. Initialize all entries in all CPTs to some random values.
2. UNTIL the termination condition is met DO
 1. Use the CPTs to calculate weights of the weighted examples.
 2. Use the calculated weighted to estimate new CPTs.

อัลกอริทึมอีเม้นโดยทั่วไปจะใช้เวลาในการลู่เข้าไม่นาน ดังจะได้แสดงในตัวอย่างการเรียนรู้ซึ่งที่ของตัวอย่างสอนในรูปที่ 6-48 ดังนี้

(1) สูมค่าสำหรับตารางซึ่งพีที

- $P(L) = 0.5$ $(P(\neg L) = 1 - P(L))$
- $P(B) = 0.5$ $(P(\neg B) = 1 - P(B))$
- $P(M|B,L) = 0.5$ $(P(\neg M|B,L) = 1 - P(M|B,L))$
 $P(M|B,\neg L) = 0.5$ $(P(\neg M|B,\neg L) = 1 - P(M|B,\neg L))$
 $P(M|\neg B,L) = 0.5$ $(P(\neg M|\neg B,L) = 1 - P(M|\neg B,L))$
 $P(M|\neg B,\neg L) = 0.5$ $(P(\neg M|\neg B,\neg L) = 1 - P(M|\neg B,\neg L))$
- $P(G|B) = 0.5$ $(P(\neg G|B) = 1 - P(G|B))$
 $P(G|\neg B) = 0.5$ $(P(\neg G|\neg B) = 1 - P(G|\neg B))$

(2) ใช้ซึ่พีทีที่สูมมาได้ในการหาหนักของตัวอย่างไม่รู้ค่าตัวอย่างไม่รู้ค่าคือ

G	M	B	L	จำนวนตัวอย่าง
*	*	True	True	7
False	True	*	True	3

ในกรณีของ 7 ตัวอย่างแรกเราต้องการหา $P(G,M|B,L)$, $P(G,\neg M|B,L)$, $P(\neg G,M|B,L)$ และ $P(\neg G,\neg M|B,L)$

- $P(G,M|B,L) = P(G|B) \times P(M|B,L) = 0.5 \times 0.5$
- $P(G,\neg M|B,L) = P(G|B) \times P(\neg M|B,L) = 0.5 \times 0.5$
- $P(\neg G,M|B,L) = P(\neg G|B) \times P(M|B,L) = 0.5 \times 0.5$
- $P(\neg G,\neg M|B,L) = P(\neg G|B) \times P(\neg M|B,L) = 0.5 \times 0.5$

ดังนั้นสำหรับตัวอย่าง 7 ตัวแรก เรายสามารถใส่หนักให้เป็นตัวอย่างมีหนักดังต่อไปนี้

G	M	B	L	จำนวนตัวอย่าง
True	True	True	True	$7 \times 0.5 \times 0.5 = 1.75$
True	False	True	True	$7 \times 0.5 \times 0.5 = 1.75$
False	True	True	True	$7 \times 0.5 \times 0.5 = 1.75$
False	False	True	True	$7 \times 0.5 \times 0.5 = 1.75$

ในกรณีของ 3 ตัวอย่าง

G	M	B	L	จำนวนตัวอย่าง
False	True	*	True	3

เราต้องหา $P(B|\neg G, M, L)$ และ $P(\neg B|\neg G, M, L)$ ซึ่งทำได้ดังนี้

$$\begin{aligned}
 \bullet P(B|\neg G, M, L) &= \frac{P(B, \neg G, M, L)}{P(\neg G, M, L)} \\
 &= \frac{P(\neg G|B, M, L)P(M|B, L)P(B|L)P(L)}{P(\neg G, M, L, B) + P(\neg G, M, L, \neg B)} \\
 &= \frac{P(\neg G|B, M, L)P(M|B, L)P(B|L)P(L)}{P(\neg G|B, M, L)P(M|B, L)P(B|L)P(L) + P(\neg G|\neg B, M, L)P(M|\neg B, L)P(\neg B|L)P(L)} \\
 &= \frac{P(\neg G|B)P(M|B, L)P(B)}{P(\neg G|B)P(M|B, L)P(B) + P(\neg G|\neg B)P(M|\neg B, L)P(\neg B)} \\
 \text{ดังนั้น } P(B|\neg G, M, L) &= \frac{0.5 \times 0.5 \times 0.5}{0.5 \times 0.5 \times 0.5 + 0.5 \times 0.5 \times 0.5} = 0.5 \\
 P(\neg B|\neg G, M, L) &= 0.5
 \end{aligned}$$

ดังนั้นสำหรับตัวอย่าง 3 ตัว เราสามารถใส่น้ำหนักให้เป็นตัวอย่างมีน้ำหนักดังต่อไปนี้

G	M	B	L	จำนวนตัวอย่าง
False	True	True	True	$3 \times 0.5 = 1.5$
False	True	False	True	$3 \times 0.5 = 1.5$

จะได้ว่าตัวอย่างทั้งหมดเป็นดังนี้

G	M	B	L	จำนวนตัวอย่าง
True	True	True	True	54
True	True	True	False	1
True	True	True	True	1.75
True	False	True	True	1.75
False	True	True	True	1.75
False	False	True	True	1.75
True	False	True	False	27
False	True	True	True	1.5
False	True	False	True	1.5
False	False	True	False	2
False	False	False	True	4
False	False	False	False	2

(3) ใช้ตัวอย่างมีน้ำหนักที่คำนวณได้ เพื่อประมาณซีพีทีใหม่

- $P(L) = 68/100 = 0.680 \quad (P(\neg L) = 1 - P(L))$
- $P(B) = 92.5/100 = 0.925 \quad (P(\neg B) = 1 - P(B))$
- $P(M|B, L) = 59/62.5 = 0.944 \quad (P(\neg M|B, L) = 1 - P(M|B, L))$
 $P(M|B, \neg L) = 1/30 = 0.033 \quad (P(\neg M|B, \neg L) = 1 - P(M|B, \neg L))$
 $P(M|\neg B, L) = 1.5/5.5 = 0.273 \quad (P(\neg M|\neg B, L) = 1 - P(M|\neg B, L))$
 $P(M|\neg B, \neg L) = 0/2 = 0.000 \quad (P(\neg M|\neg B, \neg L) = 1 - P(M|\neg B, \neg L))$
- $P(G|B) = 85.5/92.5 = 0.924 \quad (P(\neg G|B) = 1 - P(G|B))$
 $P(G|\neg B) = 0/7.5 = 0.000 \quad (P(\neg G|\neg B) = 1 - P(G|\neg B))$

(2) ใช้ชีพีที่เพื่อคำนวนน้ำหนักของตัวอย่างไม่รู้ค่าใหม่

ในกรณีของ 7 ตัวอย่างแรก

- $P(G, M|B, L) = P(G|B) \times P(M|B, L) = 0.924 \times 0.944 = 0.872$
- $P(G, \neg M|B, L) = P(G|B) \times P(\neg M|B, L) = 0.924 \times 0.056 = 0.052$
- $P(\neg G, M|B, L) = P(\neg G|B) \times P(M|B, L) = 0.076 \times 0.944 = 0.072$
- $P(\neg G, \neg M|B, L) = P(\neg G|B) \times P(\neg M|B, L) = 0.076 \times 0.056 = 0.004$

ได้ตัวอย่างมีน้ำหนักเป็น

G	M	B	L	จำนวนตัวอย่าง
True	True	True	True	7×0.872=6.11
True	False	True	True	7×0.052=0.36
False	True	True	True	7×0.072=0.50
False	False	True	True	7×0.004=0.03

ในกรณีของ 3 ตัวอย่าง

- $P(B|\neg G, M, L) = \frac{0.076 \times 0.944 \times 0.925}{0.076 \times 0.944 \times 0.925 + 1.000 \times 0.273 \times 0.075} = 0.764$
- $P(\neg B|\neg G, M, L) = 1 - P(B|\neg G, M, L) = 0.236$

ได้ตัวอย่างมีน้ำหนักเป็น

G	M	B	L	จำนวนตัวอย่าง
False	True	True	True	3× 0.764=2.29
False	True	False	True	3× 0.236=0.71

เมื่อทำขั้นตอน (2), (3) จนครบ 20 รอบชีพีที่ลู่เข้าดังนี้ (ค่าความน่าจะเป็นทุกตัวในทุกตารางชีพีที่มีค่าเปลี่ยนแปลงน้อยกว่า 0.001)

- $P(L) = 0.680$
- $P(B) = 0.940$
- $P(M|B, L) = 1.000$
- $P(M|B, \neg L) = 0.033$
- $P(M|\neg B, L) = 0.005$
- $P(M|\neg B, \neg L) = 0.000$
- $P(G|B) = 0.943$
- $P(G|\neg B) = 0.000$

เอกสารอ่านเพิ่มเติมและแบบฝึกหัด

หนังสือของ Mitchell [Mitchell, 1997] ได้ถูกใช้เป็นตำราเรียนของวิชาการเรียนรู้ของเครื่องในมหาวิทยาลัยจำนวนมาก มีคำอธิบายครอบคลุมเทคโนโลยีของการเรียนรู้ของเครื่องไว้ค่อนข้างครบถ้วน แต่ถ้าต้องการศึกษาอย่างละเอียดเฉพาะเทคนิคหนึ่งๆ เช่นถ้าเกี่ยวกับอัลกอริทึมเชิงพันธุกรรมแนะนำให้ดูหนังสือของ Mitchell [Mitchell, 1996] และ Goldberg [Goldberg, 1989] ถ้าเป็นการเรียนรู้ต้นไม้ตัดสินใจให้ดูหนังสือของ Quinlan [Quinlan, 1993] ถ้าเกี่ยวกับข่ายงานประสาทเทียมก็แนะนำให้อ่านหนังสือ [Hassoun, 1995] ส่วนหนังสือของ Pearl [Pearl, 1988] เป็นหนังสือเกี่ยวกับข่ายงานเบสท์ที่น่าศึกษาอย่างยิ่ง

บรรณานุกรม

- DeJong, G. and Mooney, R. (1986) Explanation-based learning: An alternative view. *Machine Learning*, 1(2), 145-176.
- Dempster, A. P., Laird, N. M. and Rubin D. B. (1977) Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society, Series B*, 39 (1), 1-38.
- Goldberg, D. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.
- Hassoun, M. H. (1995) *Fundamentals of Artificial Neural Networks*. The MIT Press.
- Koza, J. (1992) *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press.
- McLachlan, G. J. and Krishman, T. (1996) *The EM Algorithm and Extensions*. Wiley Interscience.
- Mitchell, M. (1996) *An Introduction to Genetic Algorithms*. The MIT Press.
- Mitchell, T. (1977) Version space: A candidate elimination approach to rule learning. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI-77)*.
- Mitchell, T., Keller, R. and Kedar-Cabelli, S. (1986) Explanation-based generalization: A unifying view, *Machine Learning*, 1(1), 47-80.
- Mitchell, T. (1997) *Machine Learning*, McGraw-Hill.
- Pearl, J. (1998) *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.
- Quinlan, J. R. (1986) Induction of decision trees. *Machine Learning*, 1(1), 81-106.
- Quinlan, J. R. (1993) *C4.5: Programs for Machine Learning*. Morgan Kaufmann.
- Rumelhart, D. E., and McClelland, J. L. (1986) *Parallel Distributed Processing: Exploration in the Microstructure of Cognition* (Vol. 1&2) The MIT Press.
- Winston, P. H. (1992) *Artificial Intelligence*. Thrid Edition. Addison Wesley.

แบบฝึกหัด

1. กำหนดให้ความรู้ในโดเมนและตัวอย่างสอนเป็นดังต่อไปนี้
- ความรู้ในโดเมน:

$$\begin{aligned} a(X, X, Y), b(red, Z) &\rightarrow c(W, X, Y, Z) \\ d(Z, Z), d(Y, X), e(X, Y) &\rightarrow a(X, Y, Z) \\ f(Y, X) &\rightarrow b(X, Y) \\ g(X, X) &\rightarrow d(X, Y) \end{aligned}$$

ตัวอย่างสอน:

$$\begin{array}{ll} e(eyes, eyes) & e(eyes, ears) \\ e(eyes, nose) & f(fire, red) \\ f(tree, green) & f(snow, white) \\ g(2, 2) & g(2, 1) \\ g(3, 2) & g(eyes, eyes) \\ g(eyes, ears) & g(eyes, nose) \end{array}$$

- จงแสดงให้เห็นว่า $c(white, eyes, 2, fire)$ เป็นตัวอย่างที่ถูกโดยใช้ต้นไม้พิสูจน์
 - กำหนดให้เกณฑ์ดำเนินการประกอบด้วยเพรดิคเตต 3 ตัวคือ e , f และ g จงเขียนกฎที่เรียนได้จากตัวอย่างด้านบน
2. ในการเรียนมโนทัศน์ของ “EnjoySport” เราสังเกตว่าเพื่อนของเรานั่นจะสนุกับการเล่นกีฬาทางน้ำหรือไม่ โดยได้พิจารณาถึงปัจจัย 6 อย่างคือ Sky (ท้องฟ้า), AirTemp (อุณหภูมิอากาศ), Humidity (ความชื้น), Wind (ลม), Water (น้ำ), Forecast (คำพยากรณ์) และได้บันทึกตัวอย่างบวก (3 ตัว) และตัวอย่างลบ (1 ตัว) ดังแสดงด้านล่าง

$$\begin{aligned} &(Sunny, Warm, Normal, Strong, Warm, Same) + \\ &(Sunny, Warm, High, Strong, Warm, Same) + \\ &(Rainy, Cold, High, Strong, Warm, Change) - \\ &(Sunny, Warm, High, Strong, Cool, Change) + \end{aligned}$$

หมายเหตุ: ตัวอย่างแต่ละตัวแสดงอยู่ในรูป $(x_1, x_2, x_3, x_4, x_5, x_6)$ โดยที่ x_1 เป็นค่าของ Sky, x_2 เป็นค่าของ AirTemp, x_3 เป็นค่าของ Humidity, x_4 เป็นค่าของ Wind, x_5 เป็นค่าของ Water, x_6 เป็นค่าของ Forecast และเครื่องหมายบวกแสดงตัวอย่างบวก เครื่องหมายลบแสดงตัวอย่างลบ กำหนดภาษาที่ใช้แสดงเป็นดังต่อไปนี้

- ค่าของ Sky ที่เป็นไปได้คือ Sunny, Cloudy, Rainy
- ค่าของ AirTemp ที่เป็นไปได้คือ Warm, Cold
- ค่าของ Humidity ที่เป็นไปได้คือ Normal, High
- ค่าของ Wind ที่เป็นไปได้คือ Strong, Weak
- ค่าของ Water ที่เป็นไปได้คือ Warm, Cool
- ค่าของ Forecast ที่เป็นไปได้คือ Same, Change

จงตอบคำถามต่อไปนี้

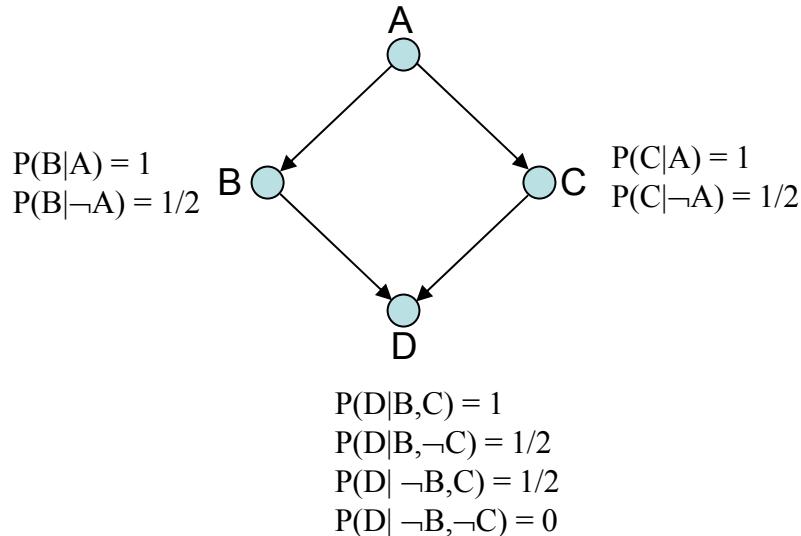
- ปริภูมิมโนทัศน์ในกรณีนี้มีขนาดเท่าไร
- จงแสดงเซต S และ G เมื่อรับตัวอย่างเข้าไปทีละตัวตามลำดับ
- เมื่อรับตัวอย่างทั้ง 4 ตัวเข้าไปหมวดแล้ว เวอร์ชันสเปซจะประกอบด้วย สมมติฐานที่เป็นไปได้ทั้งหมดกี่ตัว อะไรบ้าง (สมมติฐานทั้งหมดที่อยู่ระหว่าง S และ G (รวม S และ G ด้วย))

3. ตารางด้านล่างนี้เป็นข้อมูลของผู้ที่มาขอทำบัตรเครดิตจากธนาคารแห่งหนึ่ง ข้อมูลของคนหนึ่งๆ ประกอบด้วย account, employed, cash ธนาคารใช้ข้อมูลเหล่านี้สั่งรับกำหนดว่าจะทำบัตรให้หรือไม่ ถ้าทำให้จะมีประเภท (class) เป็น accept ถ้าไม่ทำให้จะมีประเภทเป็น reject จงสร้างต้นไม้ตัดสินใจเพื่อจำแนกประเภทข้อมูลของประเภททั้งสองนี้

Attribute			
account	employed	cash	class
bank	yes	3000	accept
bank	no	3000	accept
bank	no	40000	accept
none	yes	40000	accept
none	yes	3000	reject
none	no	40000	reject
none	no	3000	reject
other	yes	3000	reject
other	no	3000	reject
other	no	40000	accept

4. คณะกรรมการสอบคัดเลือกนิสิตที่สมัครเรียนต่อปริญญาโทของภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย ต้องการทราบว่าผู้ที่สอบผ่านจริงๆ แล้วมีคุณสมบัติใดจริงหรือไม่ จึงได้สร้างข่ายงานเบส์ตังรูปต่อไปนี้

$$P(A)=1/2$$



A = applicant is qualified.

B = applicant has high grade point average.

C = applicant has excellent recommendations.

D = applicant is admitted.

กำหนดให้ตัวแปร A, B, C, D เป็นตัวแปรแบบบูล็อกีมีค่าได้ 2 ค่าคือจริงกับเท็จ และการเขียนค่าตัวแปรในข่ายงานเป็นการเขียนแบบบูล็อกี

X แทนตัวแปร X มีค่าความจริงเป็นจริง

$\neg X$ แทนตัวแปร X มีค่าความจริงเป็นเท็จ

เช่น $P(D|\neg B,C)$ คือความน่าจะเป็นที่ D มีค่าเป็นจริงเมื่อ B มีค่าเป็นเท็จและ C มีค่าเป็นจริง เป็นต้น

จะแสดงวิธีการคำนวณหาค่า $P(A|D)$ ว่ามีค่าเท่ากับเท่าไร

5. พิจารณาเพอร์เซปตรอนยูนิตเดี่ยวที่มีอินพุต 3 บัพคือ x, y, z และมีเอาต์พุต 1 บัพคือ f ถ้าให้ตัวอย่าง 4 ตัวต่อไปนี้ จงแสดงให้เห็นว่าเพอร์เซปตรอนนี้จะเรียนรู้ได้สำเร็จ หรือไม่

อินพุต			เอาต์พุต
x	y	z	f
0	1	0	0
1	0	0	1
1	1	1	0
0	0	1	1

ดัชนีศัพท์

- 8-Puzzle, 7
- กกฎ, 63, 67
- กฎเจาะจงตัวแปรเอกพ, 49
- กฎการอนุมาน, 49
- กฎผลคุณ, 189
- กฎผลรวม, 189
- กฎลูกโซ่, 189
- กรอบ, 147
- การเข้าใจภาษาธรรมชาติ, 92
- การเคลื่อนลงตามความชัน, 178
- การเรียนรู้ของเครื่อง, 119
- การเรียนรู้เชิงวิเคราะห์, 173
- การเรียนรู้โดยการจำ, 136
- การเรียนรู้โดยการวิเคราะห์ความแตกต่าง, 141
- การเรียนรู้โดยการอธิบาย, 164
- การเรียนรู้ด้วยไม้ตัดสินใจ, 153
- การเรียนรู้แบบเบส, 186
- การแจงจำเพาะ, 143
- การแทนความรู้, 45
- การแทนค่า, 50, 67
- การแปลความหมาย, 46
- การแพร่กระจายย้อนกลับ, 183
- การโปรแกรมแบบเรียกซ้ำ, 73
- การโปรแกรมอัตโนมัติ, 4
- การไขว้เปลี่ยน, 120
- การกลยุทธ์พันธ์, 120
- การค้นหา
- การค้นหาตาม, 31
- การค้นหาต้นไม้กemen้อยสุดมากสุด, 137
- การค้นหาทั้งหมด, 10
- การค้นหาแนวกว้างก่อน, 10, 12
- การค้นหาแนวลึกก่อน, 13
- การค้นหาในปริภูมิสถานะ, 7
- การค้นหาบางส่วน, 10
- การค้นหาแบบบอต, 10, 11
- การค้นหาแบบอิวาริสติก, 10, 16
- การจับคู่, 69
- การจำแนกประเภทที่น่าจะเป็นที่สุด, 189
- การทดสอบทั่วเริ่ง, 2
- การทำเหมืองข้อมูล, 3
- การทำให้เท่ากัน, 51, 70
- การปฏิเสธแบบรีโซลูชัน, 58
- การประมวลผลภาษาธรรมชาติ, 3, 91
- การพิสูจน์, 49
- การพิสูจน์ทฤษฎี, 4
- การย้อนรอย, 69
- การวางแผนทั่วไปของโน้ตคัล, 145
- การวิเคราะห์ทางความหมาย, 92
- การวิเคราะห์ทางปฏิบัติ, 92
- การวิเคราะห์ทางวากยสัมพันธ์, 92, 96
- การวิเคราะห์ทางองค์ประกอบ, 92
- การวิพนាឌการโดยผ่านการคัดเลือกตามธรรมชาติ, 120
- การหรือของสัญญาณ, 54
- การอธิบายลดความเป็นไปได้, 199
- การอนุมานจากเหตุ, 198
- การอนุมานจากผล, 199
- ข้อเท็จจริง, 63, 65
- ข้อคิดเห็น, 63, 66
- ขั้นตอนวิธีเชิงพันธุกรรม, 119
- ข่ายงานเบส, 195
- ข่ายงานเบลี่ยนสถานะเรียกซ้ำ, 111

- ข่ายงานความเชื่อเบส์, 195
 ข่ายงานความหมาย, 142
 ข่ายงานประสาทเทียม, 169
 ข่ายงานหลาຍชັນ, 181
 คลังศัพท์, 96
 ความไม่เข้าต่อ กันอย่างมีเงื่อนไข, 195
 ความขัดแย้ง, 58
 ความน่าจะเป็นก่อน, 186
 ความน่าจะเป็นภายหลัง, 186
 ความหลากหลาย, 129
 คำคงที่, 45, 67
 คำความหมาย, 122
 คำดีสุดเฉพาะที่, 25
 คำดีสุด枉กว้าง, 25
 คำอิวิสติก, 17
 คุณสมบัติ, 154
 งานประยุกต์ทางปัญญาประดิษฐ์, 3
 จำเพาะกว่า, 148
 ต้นไม้แจงส่วน, 93
 ต้นไม้ตัดสินใจ, 153
 ต้นไม้พิสูจน์, 166
 ตระรักษ์พรดิเกต, 45
 ตระรักษ์พรดิเกตอันดับที่หนึ่ง, 48
 ตัวเชื่อม, 47
 ตัวแจงส่วน, 96
 ตัวแจงส่วนแบบบูลลงล่าง, 97
 ตัวแจงส่วนตาราง, 102
 ตัวแปร, 45, 63, 66
 ตัวแปรไม่สนใจ, 71
 ตัวแปลภาษาโปรดล็อก, 66
 ตัวกระทำการ, 8
 ตัวจำแนกประเภทเบส์อย่างง่าย, 190
 ตัวตัด, 81
 ตัวตัดเขี้ยว, 83
 ตัวตัดแดง, 86
 ตัวทำให้เท่ากัน, 51
 ตัวทำให้เท่ากันกว้างสุด, 51
 ตัวบ่งบริมาณ, 47
 ตัวบ่งบริมาณเอกสาร, 48
 ตัวบ่งบริมาณเมื่อยี่, 48
 ตัวอย่างการแทนค่า, 50
 ตัวอย่างบวก, 141
 ตัวอย่างลบ, 141
 ตามรอย, 72
 ตารางความน่าจะเป็นมีเงื่อนไข, 196
 ทฤษฎี, 49
 ทฤษฎีของเบส์, 186
 ทฤษฎีความน่าจะเป็นทั้งหมด, 189
 ทฤษฎีสารสนเทศ, 160
 ทำให้เท่ากัน, 51
 นิเสธ, 80
 นิยามของปัญญาประดิษฐ์, 1
 บูรณาการทางวิจิพนธ์, 92
 ประเภท, 153
 ปริภูมิโนทัศน์, 148
 ปัญหาโลกของบล็อก, 11
 ปัญหาการเดินทางของพนักงานขาย, 16
 ปัญหาการจัดตาราง, 4
 ปัญหาต้นไม้ K กิ่งน้อยสุด, 34
 ปัญหาทางโน้ตราชน์, 4
 ปัญหาลิงกิบกิลวัย, 71
 โปรแกรมแทรกตัวเลข, 85
 โปรแกรมคุณจำนวนธรรมชาติ, 76
 โปรแกรมจำนวนธรรมชาติ, 74
 โปรแกรมต่อรายการ, 79
 โปรแกรมบากเลข, 75
 โปรแกรมภาวะสมาร์ท, 78
 โปรแกรมลบสมาชิก, 87
 เป้าหมาย, 67
 พจน์, 50, 67
 พลัดน้อย, 141
 เพรดิเกต, 45, 65

- เพอร์เซปตรอน, 169
 พังก์ชัน, 45
 พังก์ชันเกน, 160
 พังก์ชันแม่นอัตตัน, 20
 พังก์ชันแยกเชิงเส้นได้, 177
 พังก์ชันแยกเชิงเส้นไม่ได้, 177
 พังก์ชันกระดุน, 170
 พังก์ชันสกอเต็ม, 55
 พังก์ชันอิวิสติก, 17
 ภาวะต้องห้าม, 32
 ภาษาโปรแกรม, 63
 มโนทัศน์เป้าหมาย, 149
 มีดโกนของอ็อคแคม, 157
 มีนัยทั่วไปกว่า, 148
 โมดัลเพนน์, 49
 ระบบตัดสินใจหลายมิติ, 172
 ระบบผู้เชี่ยวชาญ, 4, 163
 ระยะเวลาต้องห้าม, 35
 รายการ, 77
 รีโซลูชัน, 54
 ลำตัว, 68
 วิทยาการหุ่นยนต์, 4
 เวอร์ชันสเปซ, 147
 ไวยากรณ์, 96
 ไวยากรณ์ไม่พึงบริษท, 96
 ไวยากรณ์ข่ายงานเปลี่ยนสถานะ, 110
 สถานภาพต้องห้าม, 32
 สถานะเป้าหมาย, 8
 สถานะเริ่มต้น, 8
 สถานะสุดท้าย, 8
 สมมติฐาน, 149
 สมมติฐานความไม่ชื่นตอกัน, 191
 สมมติฐานภาษาหลังมากสุด, 187
 ส่วนหัว, 68, 77
 สอดคล้องกับ, 148
 สัญญาณ, 68
 สัญญาณเติมเต็ม, 57
 สัญลักษณ์ไม่ปลาย, 98
 สัญลักษณ์ปลาย, 98
 สูตรรูปดี, 45
 สูตรอะตอม, 46
 เส้นเชื่อมกัมมันต์, 103
 หน่วยความจำระยะยาว, 38
 หน่วยความจำระยะสั้น, 33
 หมวดคำ, 97
 หมายกำหนดการอุบหนี่ยา, 26
 ห้องจีน, 3
 องค์ประกอบ, 103
 อนุประโยชน์, 54, 67
 อนุประโยชน์ของออร์น, 68
 อนุประโยชน์ฐาน, 74
 อนุประโยชน์พื้นฐาน, 56
 อนุประโยชน์เรียกช้า, 74
 อนุมาน, 49
 อะตอม, 66
 อัลกอริทึม A*, 30
 อัลกอริทึมกฎเดลต้า, 181
 อัลกอริทึมกฎการเรียนรู้เพอร์เซปตรอน, 173
 อัลกอริทึมการค้นหาแนวว่างก่อน, 13
 อัลกอริทึมการค้นหาแนวลึกก่อน, 14
 อัลกอริทึมการค้นหาตาม, 41
 อัลกอริทึมการทำให้เท่ากัน, 52
 อัลกอริทึมการปฏิเสธแบบรีโซลูชัน, 59
 อัลกอริทึมการแพร่กระจายยั่นกลับ, 183, 185
 อัลกอริทึมการเรียนรู้เบสส์อย่างง่าย, 192
 อัลกอริทึมการเรียนรู้เวอร์ชันสเปซ, 150
 อัลกอริทึมการเรียนรู้โดยวิเคราะห์ความแตกต่าง, 146
 อัลกอริทึมการอุบหนี่ยาจำลอง, 25, 28
 อัลกอริทึมของการแจงส่วนแบบบันลงล่างอย่างง่าย, 99

-
- อัลกอริทึมของด้วยแจงส่วนตาราง, 104
อัลกอริทึมแจงส่วนแบบบูลจั่งสำหรับอาร์ทีอี็น, 112
อัลกอริทึมดีสุดก่อน, 28, 29
อัลกอริทึมต่างๆ, 34
อัลกอริทึมปีนเขา, 22
อัลกอริทึมปีนเขาชั้นสุด, 24
อัลกอริทึมปีนเขาอย่างง่าย, 23
อัลกอริทึมปีนเขาหนึ่งจั่งล่อง, 25
อัลกอริทึมอีอีม, 202
อาเจนดา, 103
อาร์กิวเมนต์, 46, 65
เอนไทรปี, 160
เอ็มจียู, 51
ชิริสติก, 16
-

Index

- 8-Puzzle, 7
A* Search, 30
activation function, 170
active arc, 103
agenda, 103
algorithm: Backpropagation, 185
algorithm: Best-First Search, 29
algorithm: Breadth-First Search, 13
algorithm: Chart Parsing, 104
algorithm: Delta-Rule, 181
algorithm: Depth-First Search, 14
algorithm: EM, 202
algorithm: Learning by Analyzing Differences, 146
algorithm: Naive-Bayes, 192
algorithm: Perceptron-Learning-Rule, 173
algorithm: Resolution Refutation, 59
algorithm: RTN Parsing, 112
algorithm: Simple Hill-Climbing Search, 23
algorithm: Simple Top-Down Parsing, 99
algorithm: Simulated Annealing Search, 28
algorithm: Tabu Search, 41
algorithm: Unify, 52
algorithm: Version-Space-Candidate-Elimination, 150
analytical learning, 168
annealing schedule, 26
argument, 46, 65
artificial neural network, 169
atom, 66
atomic formula, 46
attribute, 154
automatic programming, 4
backpropagation algorithm, 183
backtracking, 69
base clause, 74
Bayes net, 195
Bayesian belief network, 195
Bayesian learning, 186
best-first search, 28
blind search, 10
block world problem, 11
body, 68
breadth-first search, 12
category, 96
causal reasoning, 198
chain rule, 189

- chart parser, 102
Chinese room, 3
class, 153
clause, 54, 67
complimentary literals, 57
concept space, 148
conditional independence assumption, 191
conditional probability table, 196
conjunction, 67
connective, 47
consistent with, 148
constant, 67
constant symbol, 45
constituent, 103
context-free grammar, 96
contradictory, 58
crossover, 120
cut, 81
decision tree learning, 153
definition of artificial intelligence, 1
depth-first search, 13
diagnosis reasoning, 199
discourse integration, 92
disjunction of literals, 54
diversity, 129
don't care variable, 71
entropy, 160
equality, 69
evolution through natural selection, 120
exhaustive search, 10
existential quantifier, 48
expectation maximization algorithm, 202
expert system, 4, 163
explaining away, 199
explanation based learning, 164
fact, 63
fail, 83
final state, 8
first-order predicate logic, 48
fitness, 122
frame, 147
function symbol, 45
Gain function, 160
generalization of concept, 145
genetic algorithm, 119
global optimum, 25
goal, 67
goal state, 8
gradient descent, 178
grammar, 96
greedy algorithm, 34
green cut, 83
ground clause, 56
head, 69, 77
heuristic function, 17
heuristic search, 10
heuristic value, 17
hill-climbing, 22
Horn clause, 68
hyperplane decision surface, 172
hypothesis, 149
inference, 49
information theory, 160
interpretation, 46
knowledge representation, 45
learning by analyzing differences, 141
lexicon, 96
linearly non-separable function, 177
linearly separable function, 177
list, 77
local optimum, 25
long term memory, 38

- machine learning, 119
matching, 69
maximum a posterior hypothesis, 187
maximum likelihood hypothesis, 188
minimax game-tree search, 137
minimum k-tree problem, 34
Modus Ponens, 49
more general, 148
more specific, 148
morphological analysis, 92
most general unifier, 51
most probable classification, 189
multilayer neural network, 181
mutation, 120
MYCIN, 4, 163
naive Bayes classifier, 190
natural language processing, 3, 91
natural language understanding, 92
near miss, 141
negation, 80
negative example, 141
non-terminal symbol, 98
occam's razor, 157
operator, 8
parse tree, 93
parser, 96
partial search, 10
perception problem, 4
perceptron, 169
positive example, 141
posterior probability, 186
pragmatic analysis, 92
predicate, 65
predicate logic, 45
predicate symbol, 45
prior probability, 186
product rule, 189
PROLOG, 63
Prolog interpreter, 66
proof, 49
proof tree, 166
quantifier, 47
query, 63
recursive clause, 74
recursive programming, 73
recursive transition network, 111
red cut, 86
resolution, 54
resolution refutation, 58
robotics, 4
rote learning, 136
rule, 63, 67
rule of inference, 49
scheduling problem, 4
semantic analysis, 92
semantic network, 142
short term memory, 33
simulated annealing algorithm, 25
Skolem function, 55
specialization of concept, 143
state space search, 7
steepest ascent hill-climbing, 24
substitution, 50, 67
substitution instance, 50
sum rule, 189
syntactic analysis, 92
syntactic processing, 96
tabu, 31
tabu active, 32
tabu status, 32
tabu-tenure, 35

tail, 77	Turing test, 2
target concept, 149	unification, 51
term, 50, 67	unifier, 51
terminal symbol, 98	unify, 51, 69
theorem, 49	universal quantifier, 48
theorem of total probability, 189	universal specialization, 49
theorem proving, 4	variable, 63
top-down parser, 97	variable symbol, 45
trace, 72	version space, 147
transition network grammar, 110	well form formula, 45
traveling salesman problem, 16	
