

ปัญญาประดิษฐ์ ARTIFICIAL INTELLIGENCE

เอกสารคำสอนวิชา 2110654

บุญเสริม กิจศิริกุล
ภาควิชาวิศวกรรมคอมพิวเตอร์
จุฬาลงกรณ์มหาวิทยาลัย



ปัญญาประดิษฐ์

Artificial Intelligence

เอกสารคำสอนวิชา 2110654

บุญเสริม กิจศิริกุล
ภาควิชาวิศวกรรมคอมพิวเตอร์
จุฬาลงกรณ์มหาวิทยาลัย

แต่... นายหลี่ กิจศิริกุลและนางพเยาว์ แซ่เตียว

เตี้ยและแม่ผู้เป็นสุดที่รัก

คำนำ

ทพ.อรรถพร ลิ้มปัญญาเลิศ นิสิตปริญญาโทเมื่อหลายปีก่อนที่ลงเรียนวิชาปัญญาประดิษฐ์ (เดิมเปิดสอนในรายวิชาชื่อ Directed Studies in CS.) หัวเครื่องคอมพิวเตอร์โน้ตบุ๊กเข้ามาเรียนวิชานี้ทุกสัปดาห์ แล้วจบบันทึกคำสอนลงในเครื่องได้อย่างรวดเร็วไม่น่าเชื่อ การบันทึกคำสอนของอรรถพรเป็นจุดเริ่มต้นของการเขียนเอกสารคำสอนเล่มนี้

เอกสารคำสอนนี้ใช้ในการเรียนการสอนวิชาปัญญาประดิษฐ์ (2110654) เนื้อหาครอบคลุมปัญญาประดิษฐ์เบื้องต้น ปรัชมิมสถานะและการค้นหา การแทนความรู้โดยตรรกะเพรดิเคต โปรล็อกเบื้องต้น การประมวลผลภาษาธรรมชาติ และการเรียนรู้ของเครื่อง เมื่อเรียนแต่ละบทแล้ว นิสิตสามารถฝึกทำแบบฝึกหัดท้ายบท ซึ่งบางส่วนถูกนำมาใช้เป็นบ้านในรายวิชานี้ นอกจากนั้นนิสิตที่ต้องการเรียนรู้เพิ่มในหัวข้อที่สนใจ ก็สามารถศึกษาเพิ่มเติมในหนังสือหรือตำราที่แนะนำไว้ที่ท้ายบท

ไฟล์ของเอกสารคำสอนเล่มนี้อยู่ในรูปแบบของพีดีเอฟสามารถดาวน์โหลดไฟล์ได้ที่

["www.cp.eng.chula.ac.th/~boonserm/teaching/artificial.html"](http://www.cp.eng.chula.ac.th/~boonserm/teaching/artificial.html)

ในเอกสารนี้ใช้ตัวอักษรสีน้ำเงินเช่น "... **ดูรูปที่ 2-8**" แสดงไฮเปอร์ลิงค์ไปยังตำแหน่งที่อ้างถึง เพื่อให้ผู้อ่านสามารถกระโดดไปยังตำแหน่งที่อ้างได้อย่างสะดวก และใช้ตัวอักษรสีแดงเพื่อแสดงถึงคำศัพท์ใหม่ หากผู้อ่านพบข้อผิดพลาดใดๆ หรือมีข้อเสนอแนะใดๆ เกี่ยวกับเอกสารเล่มนี้ กรุณาส่งข้อความทางอีเมลมาได้ที่ boonserm.k@chula.ac.th

เอกสารคำสอนเล่มนี้สำเร็จลุล่วงไปได้ก็ด้วยความเหนื่อยยาก ความอดทน ความเสียสละและความรักจากภรรยาและลูกๆ ที่รักทุกคน ภรรยาต้องช่วยดูแลลูกที่ยังเล็ก โดยให้สามีมีเวลาทำงานเขียนเอกสาร ต้องอดทนต่อความเหน็ดเหนื่อยที่ต้องทำงานอยู่เวรจนค่ำ และยังมีภาระดูแลสมาชิกทุกคนในครอบครัว และขบใจลูกทั้งสองที่ต้อง (จำ) ยอมให้พ่อมีเวลาทำงาน ไม่ได้เล่นสนุกด้วยกัน ไม่ได้รับการสอนหนังสืออย่างเต็มที่ในเวลาที่เหมาะสมยิ่ง

บุญเสริม กิจศิริกุล

ภาควิชาวิศวกรรมคอมพิวเตอร์

จุฬาลงกรณ์มหาวิทยาลัย

12 ธันวาคม 2546

ประมวลรายวิชา

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

เนื้อหารายวิชา

นิยามของปัญหาประติษฐ์ การประยุกต์ทางปัญหาประติษฐ์ ปริภูมิสถานะและการค้นหา การแทนความรู้โดยตรรกะเพรดิเคต โปรล็อกเบื้องต้น การประมวลผลภาษาธรรมชาติ การเรียนรู้ของเครื่อง

ประมวลการเรียนรู้รายวิชา

วัตถุประสงค์ทั่วไป

- เพื่อให้นิสิตสามารถเข้าใจและอธิบายถึงปัญหาประติษฐ์และการประยุกต์ใช้งาน
- เพื่อให้นิสิตสามารถอธิบายการแทนความรู้และนำไปประยุกต์ใช้งานได้
- เพื่อให้นิสิตสามารถเขียนโปรแกรมภาษาโปรล็อกได้
- เพื่อให้นิสิตสามารถอธิบายถึงเทคนิคการประมวลผลภาษาธรรมชาติและสามารถนำไปประยุกต์ใช้งานได้
- เพื่อให้นิสิตสามารถอธิบายถึงเทคนิคการเรียนรู้ของเครื่องและนำไปประยุกต์ใช้งานได้

เนื้อหารายวิชาโดยละเอียด

สัปดาห์ที่ 1: นิยามของปัญญาประดิษฐ์ การทดสอบทัวริง ห้องเงิน งานประยุกต์ทางปัญญาประดิษฐ์

สัปดาห์ที่ 2-3: ปริภูมิสถานะและการค้นหา การนิยามปัญหาในรูปของการค้นหาในปริภูมิสถานะ เทคนิคการค้นหาในปัญญาประดิษฐ์ การค้นหาแนวกว้างก่อน การค้นหาแนวลึกก่อน การค้นหาแบบฮิวริสติก อัลกอริทึมปีนเขา อัลกอริทึมอบเหนียว จำลอง อัลกอริทึมดีสุดก่อน อัลกอริทึม A* การค้นหาตาปู

สัปดาห์ที่ 4-5: การแทนความรู้โดยตรรกะเพรดิเคต การแปลความหมายของสูตรอะตอม ตัวเชื่อมและตัวบ่งปริมาณ กฎการอนุมาน การแทนค่าและการทำให้เท่ากัน รีโซลูชัน การปฏิเสธแบบรีโซลูชัน

สัปดาห์ที่ 6: ภาษาโปรล็อก องค์ประกอบของภาษาโปรล็อก การโปรแกรมแบบเรียกซ้ำ นิเสธและตัวตัด

สัปดาห์ที่ 7: สอบกลางภาค

สัปดาห์ที่ 8-9: การประมวลผลภาษาธรรมชาติ ขั้นตอนในการเข้าใจภาษาธรรมชาติ การวิเคราะห์ทางวากยสัมพันธ์ ตัวแ่งส่วนแบบบนลงล่าง ตัวแ่งส่วนตารางไวยากรณ์ย้ายงานเปลี่ยนสถานะ

สัปดาห์ที่ 10: การเรียนรู้ของเครื่องเบื้องต้น ขั้นตอนวิธีเชิงพันธุกรรม

สัปดาห์ที่ 11: การเรียนรู้โดยการจำ การเรียนรู้โดยการวิเคราะห์ความต่าง

สัปดาห์ที่ 12: เวอร์ชันสเปซ การเรียนรู้ต้นไม้ตัดสินใจ

สัปดาห์ที่ 13: การเรียนรู้โดยการอธิบาย ข่ายงานประสาทเทียม

สัปดาห์ที่ 14: การเรียนรู้แบบเบย์

สัปดาห์ที่ 15: เทคนิคการเรียนรู้ของเครื่องอื่นๆ

สัปดาห์ที่ 16: สอบปลายภาค

วิธีการจัดการเรียนการสอน บรรยาย

สื่อการสอน แผ่นใส / กระดาน / เครื่องฉาย

การวัดผล รายงาน 40% สอบกลางภาค 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. ปัญหาประดิษฐ์เบื้องต้น	1
1.1 นิยามของปัญหาประดิษฐ์	1
1.2 การทดสอบทัวริง	2
1.3 ห้องจีน	3
1.4 งานประยุกต์ทางปัญหาประดิษฐ์	4
เอกสารอ่านเพิ่มเติม	5
2. ปริภูมิสถานะและการค้นหา	7
2.1 การนิยามปัญหาในรูปของการค้นหาในปริภูมิสถานะ	7
2.2 เทคนิคการค้นหาในปัญหาประดิษฐ์	10
2.3 การค้นหาแบบบอด	11
2.3.1 การค้นหาแนวกว้างก่อน	12
2.3.2 การค้นหาแนวลึกก่อน	13
2.4 การค้นหาแบบฮิวริสติก	16
2.4.1 ตัวอย่างของฟังก์ชันฮิวริสติก	17
ฟังก์ชันฮิวริสติก h1 สำหรับปัญหาโลกของบล็อก	17
ฟังก์ชันฮิวริสติก h2 สำหรับปัญหาโลกของบล็อก	19
ฟังก์ชันฮิวริสติกสำหรับปัญหา 8-Puzzle	20
2.4.2 อัลกอริทึมปิ่นเขา	22
2.4.3 อัลกอริทึมมอบเหนี่ยวจำลอง	25
2.4.4 อัลกอริทึมดีสุดก่อน	28
2.4.5 อัลกอริทึม A*	30

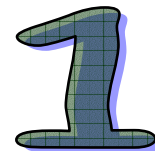
2.4.6 การค้นหาตาม	31
หน่วยความจำระยะสั้น	33
ตัวอย่างปัญหาต้นไม้ k กิ่งน้อยสุด	34
การเลือกคุณสมบัติที่ใช้กำหนดสถานะภาพต้องห้ามในการสร้างต้นไม้	35
เกณฑ์แห่งความหวัง	37
หน่วยความจำระยะยาว	38
เอกสารอ่านเพิ่มเติมและแบบฝึกหัด	41
3. การแทนความรู้โดยตรรกะเพรดิเคต	45
3.1 ตรรกะเพรดิเคต	45
การแปลความหมายของสูตรอะตอม	46
ตัวเชื่อมและตัวบ่งปริมาณ	47
3.2 กฎการอนุมาน	49
3.3 การแทนค่าและการทำให้เท่ากัน	50
3.4 รีโซลูชัน	54
รีโซลูชันของอนุประโยคพื้นฐาน	56
รีโซลูชันทั่วไป	57
การปฏิเสธแบบรีโซลูชัน	58
ตัวอย่างการทำรีโซลูชัน	59
เอกสารอ่านเพิ่มเติมและแบบฝึกหัด	60
4. โปรล็อกเบื้องต้น	63
4.1 องค์ประกอบของโปรล็อก	63
ข้อเท็จจริง	64
ค่าคงที่	66
ข้อคำถาม	66
ตัวแปร	66
พจน์และการแทนค่า	67
ตัวเชื่อม 'และ'	67

กฎ	67
ความหมายของกฎ 'P :- Q, R.'	68
การจับคู่และการย้อนรอย	69
ตัวอย่างปัญหาลิ่งกินกล้วย	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 การวิเคราะห์ทางวากยสัมพันธ์	96
5.3 ตัวแ่งส่วนแบบบนลงล่าง	97
อัลกอริทึมของตัวแ่งส่วนแบบบนลงล่างอย่างง่าย	99
5.4 ตัวแ่งส่วนตาราง	102
ตัวอย่างของการแ่งส่วนตาราง	105
5.5 ไวยากรณ์ข่ายงานเปลี่ยนสถานะ	110

อัลกอริทึมแจงส่วนบนลงล่างสำหรับอาร์ทีเอ็น	111
ตัวอย่างการแจงส่วนด้วยอาร์ทีเอ็น	113
เอกสารอ่านเพิ่มเติมและแบบฝึกหัด	116
6. การเรียนรู้ของเครื่อง	119
6.1 ขั้นตอนวิธีเชิงพันธุกรรม	119
โครโมโซมกำหนดลักษณะพิเศษที่สืบทอดได้	120
6.1.1 การออกแบบขั้นตอนวิธีเชิงพันธุกรรม	120
6.1.2 ค่าความเหมาะสมมาตรฐาน	122
6.1.3 การจำลองการคัดเลือกโดยธรรมชาติ	123
6.1.4 การไขว้เปลี่ยนเพื่อเอาชนะจุดดีสุดเฉพาะที่	126
6.1.5 ปรับปรุงจีโอด้วยฟังก์ชันความเหมาะสมแบบลำดับและการใช้ความหลากหลาย	128
ปรับปรุงจีโอด้วยฟังก์ชันความเหมาะสมแบบลำดับ	128
เพิ่มประสิทธิภาพจีโอให้สูงขึ้นโดยความหลากหลาย	129
6.2 การเรียนรู้โดยการจำ	136
6.3 การเรียนรู้โดยการวิเคราะห์ความแตกต่าง	141
6.4 เวอร์ชันสเปซ	147
6.4.1 ตัวอย่างการเรียนรู้โนทัศน์ car	150
6.4.2 ข้อจำกัดของเวอร์ชันสเปซ	152
6.5 การเรียนรู้ต้นไม้ตัดสินใจ	153
6.5.1 ฟังก์ชันเกณฑ์สำหรับการเลือกบัพทดสอบ	157
6.5.2 ฟังก์ชันเกณฑ์	160
6.5.3 การเปลี่ยนต้นไม้เป็นกฎ	162
6.6 การเรียนรู้โดยการอธิบาย	164
ตัวอย่างการเรียนรู้โนทัศน์ cup	166
6.7 ข่ายงานประสาทเทียม	169
6.7.1 เพอร์เซปตรอน	169
6.7.2 ตัวอย่างการเรียนรู้ฟังก์ชัน AND และ XOR ด้วยกฎการเรียนรู้เพอร์เซปตรอน	174
6.7.3 ข่ายงานหลายชั้นและการแพร่กระจายย้อนกลับ	181

6.8 การเรียนรู้แบบเบสส์	186
6.8.1 ทฤษฎีของเบสส์	186
6.8.2 สูตรพื้นฐานของความน่าจะเป็น	189
6.8.3 การจำแนกประเภทที่น่าจะเป็นที่สุดสำหรับตัวอย่าง	189
6.8.4 ตัวจำแนกประเภทเบสส์อย่างง่าย	190
การเรียนรู้เพื่อจำแนกประเภทข้อความโดยเบสส์อย่างง่าย	193
6.8.5 ข่ายงานความเชื่อเบสส์	195
6.8.6 การเรียนรู้ข่ายงานเบสส์	200
การเรียนรู้ข่ายงานเบสส์ในกรณีที่รู้โครงสร้างและข้อมูลครบ	200
การเรียนรู้ข่ายงานเบสส์ในกรณีที่โครงสร้างรู้และข้อมูลมีค่าหาย	201
เอกสารอ่านเพิ่มเติมและแบบฝึกหัด	206
ดัชนีศัพท์	211
Index	214

ปัญญาประดิษฐ์เบื้องต้น



1.1 นิยามของปัญญาประดิษฐ์

ปัญญาประดิษฐ์
คืออะไร?

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

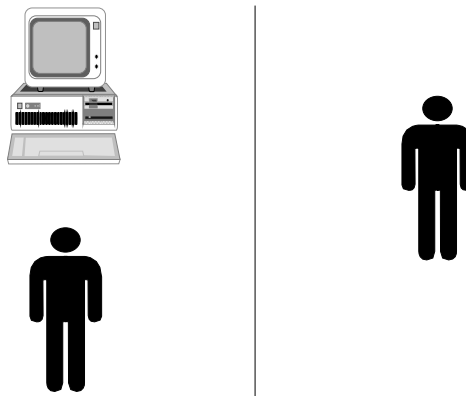
อย่างไรก็ดีงานบางอย่างที่ปัจจุบันเครื่องคอมพิวเตอร์ทำได้ดีกว่าอยู่แล้ว อย่างเช่นการคำนวณทางตัวเลข บวกหรือลบเลข ฯลฯ จะไม่อยู่ในความสนใจของสาขานี้ โดยจะเน้นที่งานที่มนุษย์ทำได้ดีกว่าและงานนั้นๆ เกี่ยวข้องกับความฉลาด (intelligence) ด้วย พฤติกรรมที่แสดงความสามารถของมนุษย์มีมากมาย ยกตัวอย่างดังด้านล่างนี้

- การเรียนรู้และเข้าใจจากประสบการณ์
- การตอบสนองต่อข้อความที่คลุมเครือหรือขัดแย้งกัน
- ความสามารถที่จะตอบสนองต่อสถานการณ์ใหม่ๆ ได้อย่างรวดเร็วและประสบความสำเร็จ
- ความสามารถให้เหตุผลในการแก้ไขปัญหาได้อย่างมีประสิทธิภาพ
- ความสามารถจัดการและแก้ไขสถานการณ์ที่ซับซ้อนได้
- ความสามารถที่จะเข้าใจและทำงานได้ในทิศทางที่ถูกต้อง
- ความสามารถที่จะใช้ความรู้เพื่อจัดการกับสภาพแวดล้อมได้
- ความสามารถที่จะหาความรู้และใช้ความรู้นั้นได้
- ความสามารถที่จะคิดและใช้เหตุผล

ระบบเอไอที่เราต้องการพัฒนาขึ้นนี้ เรามีจุดมุ่งหมายว่าต้องทำงานได้ดีกว่ามนุษย์เราแล้วยังต้องทำงานที่เกี่ยวข้องกับความฉลาดด้านบ่นด้วย คำถามคือว่าเมื่อเราสร้างระบบขึ้นมาระบบหนึ่งแล้ว จะทราบได้อย่างไรว่าระบบนี้สามารถเรียกว่าระบบเอไอได้หรือไม่ หรือเป็นระบบเอไอหรือไม่?

1.2 การทดสอบทัวริง

ในปี 2493 อลัน ทัวริง (Alan Turing) ได้เสนอวิธีการในการทดสอบสิ่งที่เรียกว่าปัญญาประดิษฐ์ขึ้น วิธีการทดสอบนี้มีชื่อเรียกว่า**การทดสอบทัวริง (Turing test)** มีจุดมุ่งหมายเพื่อแยกแยะระหว่างเครื่องคอมพิวเตอร์กับคน ถ้าหากว่าเราแยกแยะระหว่างเครื่องกับคนไม่ได้ก็ถือว่าระบบนั้นเป็นระบบปัญญาประดิษฐ์ วิธีการนี้อาศัยคนสองคนและระบบที่จะทดสอบ โดยที่คนหนึ่งจะทำหน้าที่เป็นผู้ซักถาม และผู้ซักถามนี้จะถูกแยกอยู่คนละห้องกับคนอีกคนหนึ่งและระบบที่จะทดสอบดังแสดงในรูปที่ 1-1

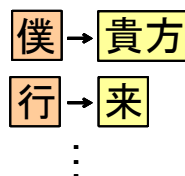


รูปที่ 1-1 การทดสอบทัวริง

ผู้ซักถามจะต้องตั้งคำถามเพื่อถามระบบหรือคนก็ได้ โดยที่คนถามจะไม่ทราบว่าห้องใดเป็นระบบหรือคน อาจจะทราบเพียงแต่ว่าเป็น A กับ B เท่านั้น หน้าที่ของผู้ซักถามคือจะต้องถามทั้ง A และ B เพื่อจำแนกให้ได้ว่า A และ B ใครเป็นระบบ ใครเป็นคน ถ้าระบบที่นำเข้ามาทำการทดสอบสามารถทำให้ผู้ซักถามเข้าใจว่าเป็นคน ระบบนั้นก็จะถูกพิจารณาได้ว่ามีความสามารถในการพูดคุยเข้าใจข้อซักถาม คิดหาเหตุผลและเป็นระบบเอไอ

1.3 ห้องจีน

ห้องจีน (Chinese room) เป็นปัญหาทางปรัชญาในเอไอปัญหาหนึ่งโดยปัญหามีอยู่ว่า มีห้องๆ หนึ่งตั้งอยู่ในประเทศจีนและมีคนนั่งอยู่ในห้องนั้น 1 คน คนนี้พูดภาษาอังกฤษได้เพียงภาษาเดียว รอบๆ ฝาผนังในห้องจะมีอักษรจีนเป็นคู่ๆ แปะไว้โดยรอบดังรูปที่ 1-2



รูปที่ 1-2 ชุดคู่ตัวอักษรในห้องจีน

ถ้ามีคนจีนที่อยู่นอกห้องยกแผ่นป้ายแผ่นหนึ่งมา คนอังกฤษคนนั้นจะต้องเทียบแผ่นป้ายนั้นกับแผ่นป้ายด้านซ้ายของคู่ตัวอักษรในห้อง แล้วยกแผ่นป้ายด้านขวาของคู่อักษรที่สอดคล้องที่แปะไว้ข้างฝาออกมายื่นให้คนจีนที่อยู่นอกห้อง การทำงานของคนอังกฤษนี้บอกถึงการแก้ไขปัญหาและตอบคำถามของคนจีน คนจีนข้างนอกอาจถามคำถามในสิ่งที่ตนไม่รู้ คนอังกฤษเมื่อเทียบแผ่นป้ายแล้วอาจให้ความรู้กับคนจีนไปได้ การพูดคุยกันโดยผ่านแผ่นป้ายถ้าสามารถดำเนินไปได้ด้วยดี ถามว่าคนอังกฤษที่นั่งอยู่ในห้อง ฉลาดหรือไม่? หรือเขารู้ภาษาจีนหรือไม่? อาจไม่ทั้งสองอย่าง แต่อย่างไรก็ตามเขาก็สามารถที่จะให้คำตอบกลับไปได้และบางครั้งอาจให้ความรู้ใหม่ๆ กับคนถามก็ได้ ระบบทางเอไอก็มีลักษณะคล้ายกับห้องจีนนี้เช่นกันโดยตัวระบบเองอาจไม่ได้เข้าใจสิ่งที่พูดคุยกัน แต่พฤติกรรมที่ระบบแสดงออกมาอาจเป็นพฤติกรรมที่แสดงถึงความฉลาดคล้ายกับมนุษย์ได้

1.4 งานประยุกต์ทางปัญญาประดิษฐ์

เทคนิคทางปัญญาประดิษฐ์ได้ถูกนำไปใช้ในงานประยุกต์ต่างๆ มากมาย ดังต่อไปนี้

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

ฐานข้อมูล ตัวอย่างเช่นธนาคารแห่งหนึ่งเก็บข้อมูลลูกค้าจำนวนมากไว้ในฐานข้อมูลเพื่อช่วยตัดสินใจในการออกบัตรเครดิตให้กับลูกค้า ซึ่งฐานข้อมูลดิบนั้นทำความเข้าใจยากมากกว่าลูกค้าดีกับไม่ดีต่างกันอย่างไร การทำเหมืองข้อมูลสามารถขุดค้นความสัมพันธ์ที่แฝงในฐานข้อมูลและสรุปคุณสมบัติของลูกค้าที่ดีที่ต่างจากลูกค้าไม่ดีออกมาได้ และใช้ประโยชน์ต่อการอนุมัติบัตรให้กับลูกค้ารายอื่นๆ ในอนาคตเพื่อให้ผลกำไรของธนาคารสูงสุด เป็นต้น

- ระบบผู้เชี่ยวชาญ (expert system) เป็นระบบเอไอที่ทำหน้าที่เหมือนผู้เชี่ยวชาญเฉพาะด้าน เช่นผู้เชี่ยวชาญในการประกอบเครื่องคอมพิวเตอร์ตามที่ลูกค้าต้องการ ตัวอย่างที่มีชื่อเสียงของระบบนี้ก็คือ MYCIN [Shortliffe, 1976] ซึ่งเป็นระบบผู้เชี่ยวชาญเหมือนแพทย์ทำหน้าที่วินิจฉัยโรคที่ติดเชื้อจากแบคทีเรียพร้อมทั้งบอกยาที่สอดคล้องกับการติดเชื้อด้วย พัฒนาขึ้นโดยเอ็ดเวิร์ด ซอร์ลลิฟฟ์ (E. Shortliffe) สามารถวินิจฉัยโรคทางด้านอายุรกรรม (โรคที่รักษาทางยา) ได้อย่างมีประสิทธิภาพ เนื่องจากระบบดังกล่าวนี้ได้บรรจุ “ความรู้” ในด้านนี้ไว้มากกว่านายแพทย์ที่เป็นมนุษย์จะจดจำได้หมดและให้ผลที่ถูกต้องแม่นยำมาก
- การพิสูจน์ทฤษฎี (theorem proving) ซึ่งนับเป็นเรื่องที่ยากมาก ถ้าให้มนุษย์ทำแต่เราสามารถใช้อีไอทำได้ค่อนข้างดี
- วิทยาการหุ่นยนต์ (robotics) เป็นการทำให้หุ่นยนต์ให้ทำงานได้มีความฉลาดสามารถตัดสินใจได้
- การโปรแกรมอัตโนมัติ (automatic programming) เป็นการเขียนโปรแกรมโดยอัตโนมัติ เช่นเราป้อนคำสั่งดับอินพุตกับเอาต์พุตของโปรแกรมที่ต้องการ เพื่อแสดงว่าอินพุตแบบนี้เรามุ่งหวังว่าจะได้เอาต์พุตอย่างไร โดยคำสั่งที่ป้อนเข้าไปมีจำนวนมากพอ แล้วระบบเอไอจะเขียนโปรแกรมที่ตรงกับคำสั่งดับอินพุตเอาต์พุตให้โดยอัตโนมัติ
- ปัญหาการจัดตาราง (scheduling problem) เทคนิคทางปัญญาประดิษฐ์ได้รับการประยุกต์ใช้กับปัญหาการจัดตารางเวลา เช่นการจัดตารางเวลาในสายการผลิตว่าจะทำอย่างไรให้เกิดประสิทธิภาพสูงสุด หรือจะจัดตารางการขึ้นลงของเครื่องบินอย่างไรให้เกิดประโยชน์สูงสุด เป็นต้น
- ปัญหาทางมโนทัศน์ (perception problem) เทคนิคทางปัญญาประดิษฐ์ได้ถูกนำไปใช้แก้ปัญหาเกี่ยวกับการมองเห็น การฟัง การได้ยิน เช่นเมื่อหุ่นยนต์มอง จะทราบได้อย่างไรว่าอันนี้เป็นกล่อง อันนี้เป็นสิ่งกีดขวาง ฯลฯ

เอกสารอ่านเพิ่มเติม

หนังสือของ Russell และ Norvig [Russell & Norvig, 1995] ให้รายละเอียดเกี่ยวกับประวัติของปัญญาประดิษฐ์ไว้อย่างละเอียด คำบรรยายเกี่ยวกับการศึกษาวิจัยเรื่องความฉลาดและปัญหาทางปัญญาประดิษฐ์แสดงไว้อย่างดีใน [Pfeifer & Scheier, 1999] นอกจากนั้นหนังสือของ 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 อย่างในปัญญาประดิษฐ์ซึ่งจะกล่าวในบทต่อไป เช่นการเรียนรู้ของเครื่อง การประมวลผลภาษาธรรมชาติ ฯลฯ ใช้พื้นฐานของการค้นหาทั้งสิ้น ในการแก้ปัญหาโดยหลักการนี้ เราจะสร้างปริภูมิ (space) ขึ้นมาหนึ่งปริภูมิ สมาชิกแต่ละตัวในปริภูมินี้แทนตัวเลือกของคำตอบ จากนั้นก็ทำการค้นหาด้วยวิธีการค้นหาอย่างใดอย่างหนึ่งซึ่งจะกล่าวในบทนี้ เพื่อให้ได้คำตอบที่ต้องการ

สิ่งที่ต้องทำในการแก้ปัญหาหนึ่งๆ คือ

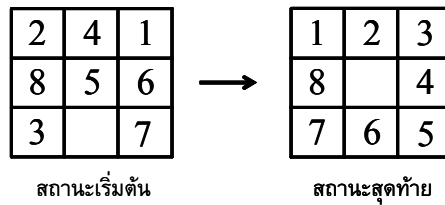
1. นิยามปัญหาอย่างชัดเจน หาสถานการณ์ปัจจุบัน (initial situation หรือ initial state) สถานการณ์สุดท้าย (final situation) ซึ่งเป็นคำตอบของปัญหา
2. วิเคราะห์ปัญหา
3. หาความรู้ที่ใช้ในการแก้ปัญหามีอะไรบ้าง
4. เลือกเทคนิคแก้ปัญหาที่เหมาะสม

2.1 การนิยามปัญหาในรูปของการค้นหาในปริภูมิสถานะ

ปัญหา
8-Puzzle

การค้นหาใน
ปริภูมิสถานะ

สมมติว่าเราต้องการแก้ปัญหา 8-Puzzle ซึ่งประกอบด้วยตารางขนาด 3x3 หน่วย ภายในตารางบรรจุแผ่นป้ายขนาด 1x1 หน่วยจำนวน 8 แผ่น ในแผ่นป้ายแต่ละแผ่นจะมีหมายเลขกำกับอยู่ และมีช่องว่างอยู่ 1 ช่องที่แผ่นป้ายสามารถเคลื่อนเข้ามาแทนที่ได้ ปัญหาคือให้เลื่อนแผ่นป้ายเหล่านี้จากตำแหน่งเริ่มต้นให้เป็นตำแหน่งสุดท้ายซึ่งเป็นคำตอบ (ดูรูปที่ 2-1 ประกอบ) ในการนิยามปัญหาในรูปของการค้นหาในปริภูมิสถานะ (state space search) นั้นสามารถทำได้ดังนี้



รูปที่ 2-1 8-Puzzle

สถานะเริ่มต้น
และ
สถานะสุดท้าย

ตัวกระทำ

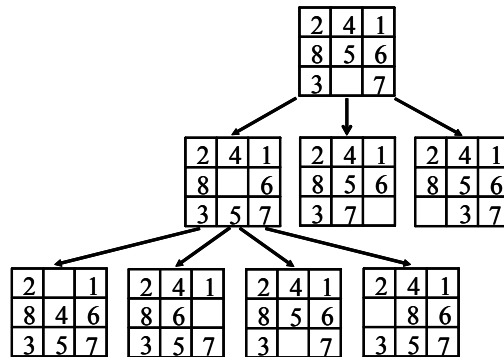
- นิยามปริภูมิสถานะโดยแสดงวัตถุทั้งหมดที่เกี่ยวข้องกับปัญหา ตัวอย่างเช่นในปัญหา 8-Puzzle เราอาจใช้รายการเพื่อแทนวัตถุเหล่านี้
- ให้**สถานะเริ่มต้น (initial state)** แทนตำแหน่งเริ่มต้นของวัตถุทั้งหมดที่เกี่ยวข้องกับปัญหาและ**สถานะสุดท้ายหรือสถานะเป้าหมายหรือคำตอบ (final state or goal state or solution)** 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 ซึ่งเราเรียกกฎเหล่านี้ว่า**ตัวกระทำ (operator)** ในที่นี้ตัวกระทำก็คือการเลื่อนแผ่นป้าย
- ใช้เทคนิคของการค้นหาในการเปลี่ยนจากสถานะเริ่มต้นไปยังสถานะเป้าหมาย

ตัวอย่างตัวกระทำในกรณีของ 8-Puzzle เป็นดังนี้

- แผ่นป้ายหนึ่งๆ จะเลื่อนมาที่ช่องว่างได้ถ้าอยู่ติดกับช่องว่าง และสถานะจะเปลี่ยนจากสถานะเดิมไปยังสถานะใหม่ซึ่งตำแหน่งของแผ่นป้ายสลับที่กับตำแหน่งของช่องว่าง ตัวกระทำมีทั้งหมด 4 ตัวดังนี้
 1. ด้านบนของช่องว่างมีแผ่นป้าย → สลับตำแหน่งของช่องว่างกับแผ่นป้าย
 2. ด้านขวาของช่องว่างมีแผ่นป้าย → สลับตำแหน่งของช่องว่างกับแผ่นป้าย
 3. ด้านล่างของช่องว่างมีแผ่นป้าย → สลับตำแหน่งของช่องว่างกับแผ่นป้าย
 4. ด้านซ้ายของช่องว่างมีแผ่นป้าย → สลับตำแหน่งของช่องว่างกับแผ่นป้าย

ตัวกระทำที่ 1. มีความหมายว่า ถ้าด้านบนของช่องว่างมีแผ่นป้าย ให้สลับตำแหน่งของช่องว่างกับแผ่นป้าย และจะเกิดสถานะใหม่ซึ่งมีตำแหน่งของช่องว่างกับแผ่นป้ายสลับที่กัน

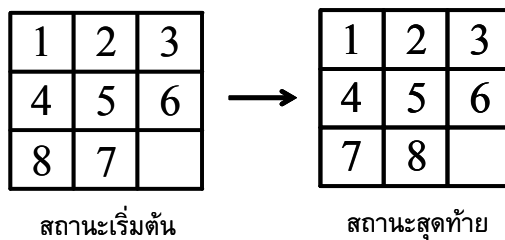
เมื่อเรานิยามตัวกระทำเหล่านี้แล้ว ปริภูมิสถานะก็จะเกิดขึ้นโดยปริยาย และเราจะเห็นปริภูมิสถานะว่าสถานะใดเชื่อมต่อกับสถานะใด ก็เมื่อเราเริ่มจากสถานะเริ่มต้นหนึ่งตัวแล้วใช้ตัวกระทำเหล่านี้สร้างการเชื่อมต่อของสถานะ ในกรณีของปัญหา 8-Puzzle เมื่อใช้ตัวกระทำด้านบน เริ่มจากสถานะเริ่มต้นในรูปที่ 2-1 จะสร้างปริภูมิสถานะบางส่วนดังแสดงในรูปที่ 2-2



รูปที่ 2-2 บางส่วนของปริภูมิสถานะในปัญหา 8-Puzzle

จากรูปจะเห็นว่าการเชื่อมต่อของสถานะถูกกำหนดโดยตัวกระทำ การ ดังนั้นในปัญหาหนึ่งๆ เมื่อเรานิยามตัวกระทำได้แล้ว เราก็จะได้ปริภูมิของสถานะสำหรับปัญหานั้นๆ อย่างไรก็ดี เราจะพบว่าปริภูมิสถานะโดยทั่วไปจะมีลักษณะเป็นโครงสร้างแบบกราฟมากกว่าจะเป็นแบบต้นไม้ เนื่องจากบางสถานะอาจเกิดซ้ำกันได้ เช่นในกรณีของตัวอย่างในรูปที่ 2-2 จะเห็นได้ว่าสถานะที่ 3 (นับจากซ้าย) ที่แถวล่างสุดซ้ำกับสถานะเริ่มต้น ซึ่งการเป็นกราฟนี้เองทำให้การค้นหาในปริภูมิสถานะมีความยุ่งยากมากขึ้นกว่าปริภูมิที่เป็นแบบต้นไม้

นอกจากนั้นสถานะเริ่มต้นตัวหนึ่งที่กำหนดให้ อาจไม่สามารถนำไปสู่สถานะเป้าหมายบางตัวได้ เนื่องจากสถานะเป้าหมายนั้นไม่มีเส้นทางที่เชื่อมต่อจากสถานะเริ่มต้น ตัวอย่างเช่นในรูปที่ 2-3



รูปที่ 2-3 ตัวอย่างคำตอบที่เข้าถึงไม่ได้ด้วยสถานะเริ่มต้น

จากที่กล่าวข้างต้นจะเห็นได้ว่า การแก้ปัญหาพื้นฐานทางปัญญาประดิษฐ์แบบหนึ่งคือการมองปัญหาในรูปแบบของการค้นหาในปริภูมิสถานะ โดยเริ่มจากการนิยามสถานะ นิยามโครงสร้างข้อมูลที่เก็บวัตถุที่เกี่ยวข้องกับปัญหา กำหนดสถานะเริ่มต้นและสถานะ

เป้าหมาย รวมทั้งหาตัวกระทำการว่าจะต้องใช้ตัวกระทำอะไรบ้างในการนิยามสถานะ จากนั้นก็เป็นการเลือกเทคนิคการค้นหาที่เหมาะสมกับปัญหาที่เราต้องการ หัวข้อต่อไปนี้จะกล่าวถึงเทคนิคการค้นหาแบบต่างๆ ที่สามารถนำมาใช้ในการค้นหาในปริภูมิสถานะ

2.2 เทคนิคการค้นหาในปัญญาประดิษฐ์

เทคนิคการค้นหาสามารถแบ่งประเภทได้ดังต่อไปนี้

1. การค้นหาแบบบอด (blind search)
 - 1.1 การค้นหาทั้งหมด (exhaustive search)
 - 1.2 การค้นหาบางส่วน (partial search)
 - 1.2.1 การค้นหาแนวกว้างก่อน (breadth-first search)
 - 1.2.2 การค้นหาแนวลึกก่อน (depth-first search)
2. การค้นหาแบบฮิวริสติก (heuristic search)
 - 2.1 อัลกอริทึมปีนเขา (hill-climbing algorithm)
 - 2.2 อัลกอริทึมอบเหนียวจำลอง (simulated annealing algorithm)
 - 2.3 การค้นหาดีที่สุดก่อน (best-first search)
 - 2.4 การค้นหา A* (A* search)
 - 2.5 อื่นๆ

เทคนิคการค้นหาสามารถแบ่งได้เป็นสองประเภทหลักๆ คือ **การค้นหาแบบบอด (blind search)** ซึ่งเป็นเทคนิคการค้นหาที่ไม่มีตัวช่วยในการค้นหา แต่จะมีรูปแบบการค้นหาที่แน่นอนตายตัว เช่นจะค้นหาสถานะจากบนลงล่างในปริภูมิค้นหา เป็นต้น ส่วนเทคนิคการค้นหาอีกประเภทหนึ่งคือ **การค้นหาแบบฮิวริสติก (heuristic search)** ซึ่งจะใช้ความรู้รูปแบบหนึ่งที่เรียกว่าฮิวริสติกมาช่วยทำให้การค้นหามีประสิทธิภาพยิ่งขึ้น

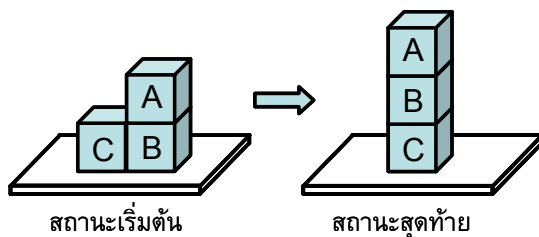
การค้นหาแบบบอดสามารถแบ่งย่อยได้ดังนี้คือ **การค้นหาทั้งหมด (exhaustive search)** หมายถึงการค้นหาทั้งหมดทั่วทั้งปริภูมิสถานะและ**การค้นหาบางส่วน (partial search)** เป็นการค้นหาเพียงบางส่วนของปริภูมิสถานะ ปัญหาส่วนใหญ่ทางปัญญาประดิษฐ์มีปริภูมิสถานะที่มีขนาดใหญ่มาก ทำให้เราไม่สามารถค้นหาได้ทั่วทั้งปริภูมิ จำเป็นต้องค้นหาเพียงบางส่วนของปริภูมิเท่านั้น ดังนั้นจึงมีความเป็นไปได้ว่าคำตอบที่ได้อาจไม่ใช่คำตอบที่ดีที่สุด การค้นหาเพียงบางส่วนโดยการค้นหาแบบบอดนั้นสามารถแบ่งได้เป็นสองประเภทคือ **การค้นหาแนวกว้างก่อน (breadth-first search)** ซึ่งเป็นการค้นหาในแนวกว้างก่อนเมื่อ

พิจารณาจากโครงสร้างต้นไม้ของปริภูมิสถานะ และ *การค้นหาแนวลึกก่อน (depth-first search)* คือหาแนวลึกก่อนเมื่อพิจารณาจากโครงสร้างต้นไม้ ส่วนการค้นหาแบบฮิวริสติก (heuristic search) มีหลายเทคนิคด้วยกัน เช่น *อัลกอริทึมปีนเขา (hill-climbing search)* *อัลกอริทึมอบเหนียวจำลอง (simulated annealing algorithm)* *การค้นหาที่ดีที่สุดก่อน (best-first search)* *การค้นหา A* (A* search)* เป็นต้น ในหัวข้อนี้จะกล่าวถึงเทคนิคค้นหาในแต่ละวิธีโดยเริ่มจากการค้นหาแบบบอดและตามด้วยการค้นหาแบบฮิวริสติก

2.3 การค้นหาแบบบอด

ส่วนนี้อธิบายอัลกอริทึมการค้นหาแบบบอด (blind search) โดยจะเริ่มจากการค้นหาแนวกว้างก่อน แล้วต่อด้วยการค้นหาแนวลึกก่อน โดยตัวอย่างที่ใช้เพื่ออธิบายอัลกอริทึมคือ *ปัญหาโลกของบล็อก (block world problem)* ซึ่งแสดงในรูปที่ 2-4 ด้านล่างนี้

ปัญหา
โลกของบล็อก



รูปที่ 2-4 ปัญหาโลกของบล็อก

ปัญหาคือกำหนดสถานะเริ่มต้นของการจัดเรียงตัวของบล็อกให้ ต้องการจัดเรียงใหม่ให้ได้ตามสถานะสุดท้าย โดยมีตัวกระทำดังนี้

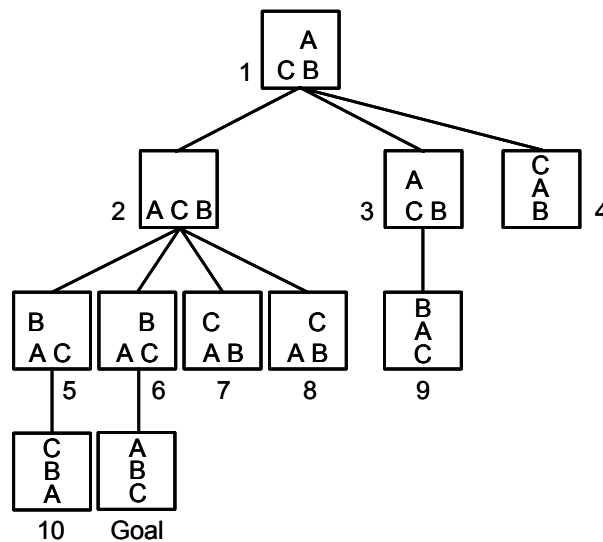
- (1) บล็อก X ไม่มีบล็อกอื่นทับ → วาง X บนโต๊ะ
- (2) บล็อก X และ Y ไม่มีบล็อกอื่นทับ → วาง X บน Y

โดยที่ X และ Y เป็นตัวแปรและสามารถแทนที่ด้วย 'A', 'B' หรือ 'C' เช่นเมื่อใช้ตัวกระทำ (1) กับสถานะเริ่มต้น จะได้ว่าถ้าบล็อก 'A' ไม่มีบล็อกอื่นทับ แล้ววาง 'A' บนโต๊ะได้ เป็นต้น

2.3.1 การค้นหาแนวกว้างก่อน

ในการค้นหาแนวกว้างก่อน (breadth-first search) นี้ สถานะทุกตัว (ซึ่งบางทีเรียกว่า **โหนด** (node)) เมื่อมองปริภูมิก้นหาอยู่ในรูปของต้นไม้ ที่อยู่ในระดับเดียวกันของต้นไม้จะถูกตรวจสอบก่อนสถานะที่อยู่ในระดับถัดไป วิธีการของการค้นหาแบบนี้ทำโดยเริ่มจากการสร้างสถานะลูกของสถานะเริ่มต้นก่อน แล้วตรวจสอบว่ามีสถานะใดที่เป็นสถานะสุดท้ายหรือไม่ ถ้าหากว่ามีก็เป็นอันว่าการค้นหาสิ้นสุด ถ้าไม่มีก็จะสร้างสถานะลูกของสถานะเหล่านั้น แล้วทำการตรวจสอบสถานะลูกทุกตัวของสถานะเหล่านั้น ถ้าพบสถานะสุดท้าย การค้นหาก็สิ้นสุด ถ้าไม่พบก็สร้างสถานะลูกของสถานะเหล่านั้นต่อไปอีก ทำเช่นนี้ไปเรื่อยๆ จนกว่าจะพบสถานะสุดท้ายหรือจนไม่สามารถสร้างสถานะลูกใหม่ได้อีก

ในกรณีของปัญหาโลกของบล็อก เริ่มจากสถานะเริ่มต้น เมื่อเราค้นหาด้วยการค้นหาแบบแนวกว้างก่อน ก็จะได้สถานะที่เกิดขึ้นดังรูปที่ 2-5 ในตัวอย่างนี้กำหนดว่าลำดับของแถวไม่มีความสำคัญ กล่าวคือสถานะ 'A C B' เท่ากับสถานะ 'C B A' เป็นต้น และในการสร้างสถานะจะไม่สร้างสถานะซ้ำเดิม เช่นจากสถานะที่ 2 เราสามารถสร้างสถานะลูกของมันตัวหนึ่งซึ่งเท่ากับสถานะที่ 1 แต่จะไม่นำสถานะนี้ใส่ลงเป็นสถานะลูกของสถานะที่ 2 เนื่องจากไปซ้ำกับสถานะที่ 1



รูปที่ 2-5 การค้นหาแบบแนวกว้างก่อนในปัญหาโลกของบล็อก

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

กระทำการ (1) และตัวกระทำ (2) และแทนที่ตัวแปร X และ Y ในตัวกระทำทั้งสองให้เป็น 'A', 'B' หรือ 'C' ตามลำดับจะได้สถานะลูกเป็นสถานะที่ 2 สถานะที่ 3 และ สถานะที่ 4 ตามลำดับ เมื่อสร้างสถานะลูกของสถานะที่ 1 ครบทุกตัวแล้ว ก็จะลงมาในระดับถัดไปเพื่อสร้างสถานะลูกของสถานะที่ 2, 3 และ 4 เป็นเช่นนี้ไปจนกว่าจะได้สถานะสุดท้ายที่ต้องการ ซึ่งจะเห็นได้ว่าการค้นหาจะทำในแนวกว้างที่ระดับตั้งแต่บนลงล่าง จากตัวอย่างเราได้สถานะสุดท้ายเป็นสถานะที่ 11 และไม่ต้องทำการค้นหาต่อเพราะการค้นหาแบบนี้เป็นการค้นหาเพียงบางส่วนเท่านั้น แม้ว่าอาจจะมีเส้นทางอื่นอีกที่สามารถนำไปสู่สถานะสุดท้ายได้ อัลกอริทึมของการค้นหาแนวกว้างก่อนแสดงได้ในตารางที่ 2-1 ด้านล่างนี้

ตารางที่ 2-1 อัลกอริทึมการค้นหาแนวกว้างก่อน

Algorithm: Breadth-First Search

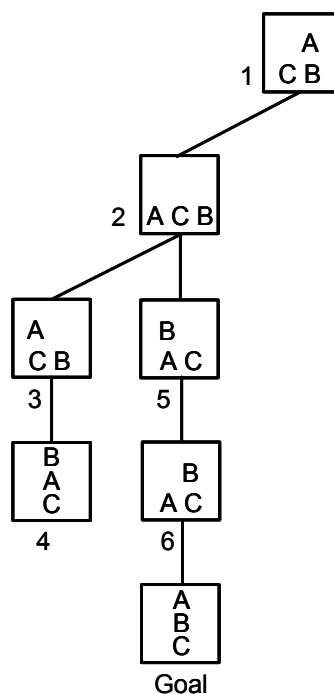
1. Node-list := {initial state}
2. UNTIL a goal state is found or Node-list is empty DO
 - 2.1 Remove the first element from Node-list and call it E.
 - 2.2 For Each operator matching E DO
 - 2.2.1 Apply the operator to generate a new state.
 - 2.2.2 IF the new state is a goal state THEN quit and return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

หมายเหตุ อัลกอริทึมด้านบนนี้ไม่ได้ตรวจสอบสถานะซ้ำ

2.3.2 การค้นหาแนวลึกก่อน

การค้นหาแนวลึกก่อน (depth-frist search) จะสร้างสถานะในแนวลึกทางด้านมุมซ้ายล่างก่อน (ดูรูปที่ 2-6 ประกอบ) ถ้าสถานะตามแนวดิ่งถูกสร้างหรือกระจายจนหมดและยังไม่ได้คำตอบ ก็จะไล่กลับขึ้นด้านบนเพื่อหาเส้นทางอื่นที่จะเป็นไปได้

การค้นหาแบบแนวกว้างก่อนกับการค้นหาแนวลึกก่อนนั้น ไม่สามารถบอกได้ว่าการค้นหาแบบไหนจะได้คำตอบเร็วกว่ากัน ขึ้นอยู่กับว่าคำตอบของเราอยู่บริเวณไหนในปริภูมิสถานะ (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 พอไปเส้นทางอื่น เส้นทางที่ผ่านมาก็ไม่จำเป็นต้องเก็บไว้อีก)	1. ใช้หน่วยความจำมากกว่า เพราะต้องเก็บสถานะไว้ทุกตัว เพื่อหาเส้นทางจากสถานะเริ่มต้นไปคำตอบ
2. อาจติดเส้นทางที่ลึกมาก ๆ โดยไม่พบคำตอบ เช่นในกรณีเส้นทางนั้นไม่มีคำตอบ และเป็นเส้นทางที่ยาวไม่สิ้นสุด ซึ่งการค้นหาแบบนี้ จะไปเส้นทางอื่นไม่ได้ (เช่นกรณีของ Prolog ซึ่งใช้วิธีค้นหาแบบนี้ จะทำการค้นหาไปเรื่อยๆ จนกว่าสถานะที่สร้างขึ้น ใช้หน่วยความจำเกิน ซึ่งจะเกิดข้อผิดพลาดขึ้น วิธีแก้ไขที่ใช้ใน Prolog คือให้ผู้กำหนดความลึกในการค้นหาคำตอบ จะได้ไม่เสียเวลา ในการค้นหานานเกินจำเป็น เช่นกำหนดให้ความลึกในการค้นหาในแต่ละเส้นทางไม่มากกว่า 100 ขั้นตอน เป็นต้น หรืออาจจะกำหนดที่ขนาดของหน่วยความจำก็ได้)	2. จะไม่ติดเส้นทางที่ลึกมาก ๆ โดยไม่พบคำตอบ
3. ถ้าคำตอบอยู่ที่ระดับ $n+1$ สถานะอื่นทุกตัวที่อยู่ระดับ 1 ถึงระดับ n ไม่จำเป็นต้องถูกกระจายจนหมด	3. ถ้าคำตอบอยู่ที่ระดับ $n+1$ สถานะทุกตัวที่ระดับ 1 ถึงระดับ n จะต้องถูกกระจายจนหมด ทำให้มีสถานะที่ไม่จำเป็นในเส้นทางที่จะไปสู่คำตอบถูกกระจายออกด้วย
4. เมื่อพบคำตอบ ไม่สามารถรับประกันได้ว่าเส้นทางที่ได้เป็นเส้นทางสั้นสุดหรือไม่	4. ถ้ามีคำตอบจะรับประกันได้ว่าจะพบคำตอบแน่ๆ และจะได้เส้นทางสั้นสุดด้วย (สมมติว่าระยะห่างหรือต้นทุนระหว่างสถานะ 2 ตัวใดๆ มีค่าเท่ากันหมด)

2.4 การค้นหาแบบฮิวริสติก

ปัญหา
การเดินทาง
ของ
พนักงานขาย

การค้นหาประเภทฮิวริสติกนี้จะใช้ความรู้แบบหนึ่งที่เรียกว่าฮิวริสติกมาช่วยในการค้นหาให้มีประสิทธิภาพมากขึ้น โดยฮิวริสติกตัวนี้จะช่วยชี้แนะว่ากระบวนการค้นหาควรที่จะเลือกเส้นทางใดหรือสถานะใดเพื่อทำการค้นหาต่อไปให้ได้คำตอบอย่างมีประสิทธิภาพ พิจารณาปัญหาการเดินทางของพนักงานขาย (traveling salesman problem) ซึ่งแสดงในรูปที่ 2-7



รูปที่ 2-7 ปัญหาการเดินทางของพนักงานขาย

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

ฮิวริสติก
คืออะไร?

ถ้าเราใช้สามัญสำนึกโดยคาดเดาอย่างมีเหตุผลว่า เมื่อเราต้องการระยะทางโดยรวมสั้นที่สุด เราก็น่าจะเลือกเมืองที่อยู่ใกล้มากที่สุดกับเมืองที่เราอยู่ในปัจจุบัน แล้วเดินทางไปเมืองนั้นก่อน เมื่อไปถึงเมืองนั้นแล้วค่อยทำในทำนองเดียวกันอีกว่า จะเดินไปยังเมืองที่ใกล้ที่สุดเมืองถัดไป ทำเช่นนี้จนกระทั่งเดินทางครบทุกเมือง ก็น่าจะได้ระยะทางโดยรวมสั้นที่สุด แม้ว่าวิธีการเช่นนี้จะทำงานได้อย่างมีประสิทธิภาพ และคำตอบที่ได้มีแนวโน้มว่าจะดี แต่อย่างไรก็ดี คำตอบที่ได้โดยวิธีนี้อาจไม่เป็นเส้นทางที่สั้นที่สุดก็ได้ วิธีการเช่นนี้ก็คือการนำความรู้แบบหนึ่งมาแก้ไขปัญห ความรู้แบบนี้อาจไม่ใช่ความรู้ที่สมบูรณ์ แต่ก็พอที่จะนำมาแก้ไขปัญหให้เราได้ และช่วยแนะให้เราทราบว่าควรที่จะค้นหาเส้นทางอย่างไร เราเรียกว่าความรู้ที่ไม่สมบูรณ์หรือการคาดเดาอย่างมีเหตุผลแบบนี้ว่า **ฮิวริสติก**

การค้นหาแบบฮิวริสติกคือ การค้นหาที่นำความรู้ประเภทนี้มาใช้ช่วยชี้แนะเส้นทางในการค้นหาคำตอบ โดยมีลักษณะเด่นดังนี้

ฟังก์ชัน
ฮิวริสติก

- เป็นเทคนิคที่ใช้เพิ่มประสิทธิภาพของกระบวนการค้นหา โดยอาจจะต้องยอมให้ขาดความสมบูรณ์ไปบ้าง คืออาจไม่พบคำตอบที่ถูกต้อง แม้ว่าในปริภูมิสถานะจะมีคำตอบนี้อยู่
- การนำฮิวริสติกมาใช้จะต้องนำมาใช้ในรูปแบบที่วัดค่าได้อย่างง่าย ซึ่งมักทำโดยนิยามฮิวริสติกให้อยู่ในรูปแบบของฟังก์ชัน เราเรียกว่า **ฟังก์ชันฮิวริสติก (heuristic function)** ซึ่งเป็นฟังก์ชันที่คำนวณค่าจากสถานะไปยังตัวเลขที่ชี้ว่าสถานะนั้นเข้าใกล้สถานะเป้าหมายมากเท่าไร (ยิ่งมากเท่าไร ยิ่งมีโอกาที่จะเปลี่ยนเป็นสถานะเป้าหมายมากเท่านั้น) การค้นหาจะมุ่งไปเส้นทางที่มีค่าฟังก์ชันฮิวริสติกที่ดีกว่า
- ฟังก์ชันฮิวริสติกนี้เป็นสิ่งที่ชี้แนะกระบวนการค้นหาว่าควรจะค้นหาไปในทิศทางใด ซึ่งกระบวนการค้นหาที่ใช้ฟังก์ชันฮิวริสติกสามารถออกแบบได้หลายชนิด ดังจะกล่าวต่อไป
- ในบางกรณีที่เราสามารถนิยามฟังก์ชันฮิวริสติกได้อย่างสมบูรณ์แบบ การค้นหาจะสามารถมุ่งตรงไปยังสถานะเป้าหมายโดยไม่ผิดเส้นทางเลย แต่ถ้าฟังก์ชันฮิวริสติกไม่ดีก็อาจทำให้กระบวนการค้นหาหลงไปในทิศทางที่ผิดได้ ทำให้คำตอบที่ได้เมื่อใช้ฮิวริสติกไม่ใช่คำตอบที่ดีที่สุด

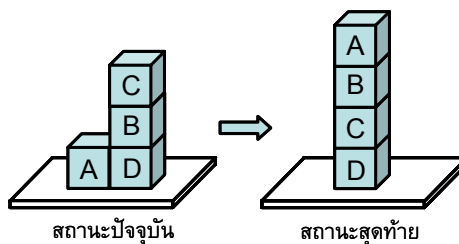
2.4.1 ตัวอย่างของฟังก์ชันฮิวริสติก

ในส่วนนี้จะขอยกตัวอย่างการนิยามฟังก์ชันฮิวริสติกสัก 3 ฟังก์ชันดังนี้

ฟังก์ชันฮิวริสติก h1 สำหรับปัญหาโลกของบล็อก

ฟังก์ชันฮิวริสติกตัวแรกที่จะยกมาให้ดูเป็นฟังก์ชันสำหรับคำนวณ **ค่าฮิวริสติก (heuristic value)** สำหรับสถานะใดๆ ของปัญหาโลกของบล็อก ขอเรียกฟังก์ชันนี้ว่า h1 ซึ่งนิยามดังนี้

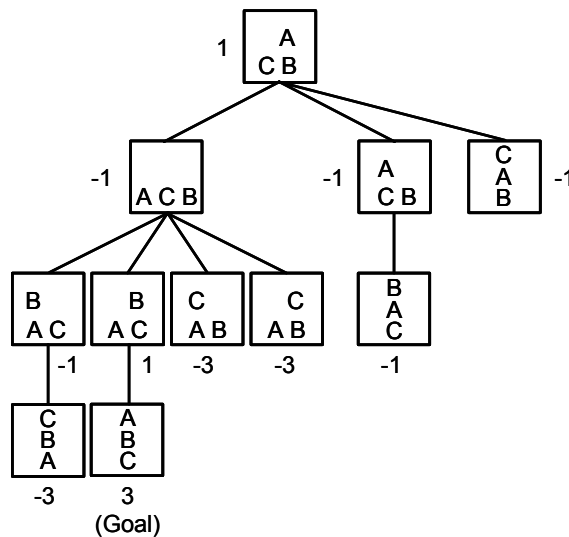
$h1$: บวกหนึ่งให้กับทุกบล็อกที่วางบนสิ่ง (บล็อกหรือโต๊ะ) ที่มันควรอยู่ และลบหนึ่งถ้าไม่ใช่



รูปที่ 2-8 ตัวอย่างฟังก์ชันฮิวริสติก h1

‘สิ่งที่มันควรอยู่’ ในที่นี้คือสถานะสุดท้ายหรือคำตอบ พิจารณา **รูปที่ 2-8** เราจะได้ว่าค่าฟังก์ชันจะเท่ากับค่าที่บล็อกแต่ละบล็อกได้รับเมื่อเทียบกับสถานะสุดท้ายว่า มันวางอยู่บนสิ่งเดียวกันหรือไม่ ซึ่งจะได้ว่า A ได้ -1 เพราะว่าในสถานะสุดท้าย A วางอยู่บน B แต่ในสถานะปัจจุบัน A วางอยู่บนโต๊ะ เมื่อตรวจสอบบล็อกทุกบล็อกจะได้ดังนี้คือ $A = -1$, $B = -1$, $C = -1$, $D = 1$ ซึ่งทำให้ได้ค่ารวมของฟังก์ชันเท่ากับ -2 หน่วย

ด้านล่างนี้แสดงค่าฮิวริสติกสำหรับสถานะต่างๆ ใน **รูปที่ 2-5**



รูปที่ 2-9 ค่าฮิวริสติก h1 สำหรับสถานะต่างๆ ใน **รูปที่ 2-5**

ใน **รูปที่ 2-9** นี้ ตัวเลขที่กำกับที่แต่ละสถานะเป็นค่าฮิวริสติกที่คำนวณได้โดยฟังก์ชัน h1 จะ

เห็นว่าสถานะ $\begin{matrix} B \\ A C \end{matrix}$ ที่เข้าใกล้คำตอบมีค่าฮิวริสติกเท่ากับ 1 ส่วนสถานะ $\begin{matrix} C B \\ A \end{matrix}$ ที่ต่างจากคำตอบมากก็มีค่าฮิวริสติกน้อยสุดเท่ากับ -3 ซึ่งแสดงให้เห็นว่าฟังก์ชัน h1 สามารถวัดความใกล้เคียงคำตอบได้ค่อนข้างดี อย่างไรก็ตามเมื่อพิจารณาที่สถานะลูกของสถานะเริ่มต้นในระดับแรก จะพบว่าสถานะทุกตัวมีค่าเท่ากับ -1 ซึ่งหมายความว่าฟังก์ชัน h1 ไม่สามารถ

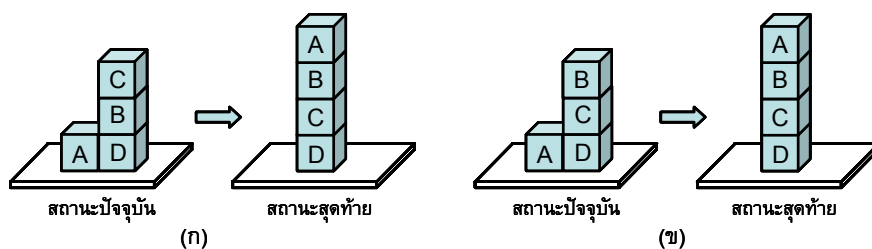
แยกความแตกต่างของสถานะทั้งสามนี้ได้ แม้ว่าสถานะ $\begin{matrix} A C B \end{matrix}$ จะนำไปสู่คำตอบได้เร็ว

กว่าสถานะ $\begin{matrix} A \\ C B \end{matrix}$ และสถานะ $\begin{matrix} C \\ A B \end{matrix}$

ฟังก์ชันฮิวริสติก h2 สำหรับปัญหาโลกของบล็อก



ดังจะเห็นได้ในตัวอย่างของฟังก์ชันฮิวริสติก h1 ที่กล่าวข้างต้นว่าไม่สามารถแยกความแตกต่างของบางสถานะได้ ในที่นี้จึงขอยกตัวอย่างฟังก์ชันฮิวริสติก h2 ที่มีประสิทธิภาพในการวัดความเข้าใจลำดับค่าตอบหรือความดีของสถานะได้อย่างสมบูรณ์แบบและมีประสิทธิภาพดีกว่า h1 ซึ่ง h2 มีนิยามดังนี้

h2: สำหรับบล็อกแต่ละก้อนที่อยู่บนโครงสร้างที่ถูก บวก 1 แต้มให้กับบล็อกทุกก้อนที่อยู่ในโครงสร้างนั้นเพื่อเป็นคะแนนสำหรับบล็อกที่อยู่บนโครงสร้างที่ถูกนั้น และลบ 1 แต้มสำหรับบล็อกทุกก้อนที่อยู่บนโครงสร้างที่ผิด เพื่อเป็นคะแนนสำหรับบล็อกที่อยู่บนโครงสร้างที่ผิดนั้น ค่าฮิวริสติกสำหรับสถานะคือคะแนนรวมของบล็อกทุกก้อนที่พิจารณาโดยที่โครงสร้างคือบล็อกที่เรียงตัวต่อกันตามแนวดิ่ง(ไม่นับโต๊ะ)

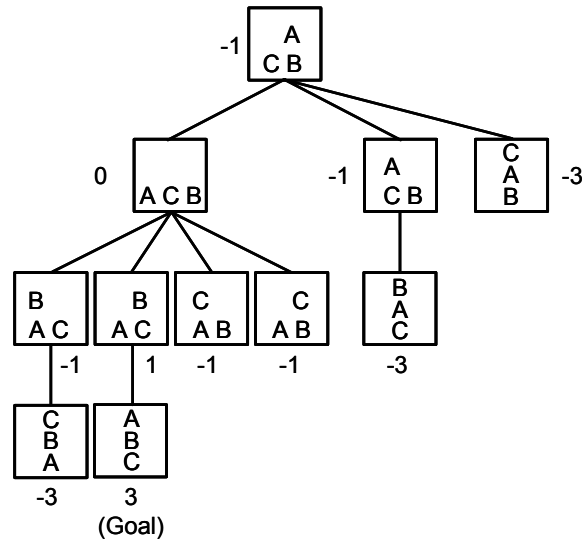


รูปที่ 2-10 ตัวอย่างฟังก์ชันฮิวริสติก h2

ฟังก์ชันฮิวริสติก h2 นี้จะพิจารณาถึงค่าความดีของสถานะโดยดูที่โครงสร้างที่รองรับบล็อกใดๆ ว่าเป็นโครงสร้างที่ตรงกับโครงสร้างที่รองรับบล็อกเดียวกันในคำตอบ ทำให้การวัดค่าฮิวริสติกมีความแม่นยำยิ่งขึ้น พิจารณาตัวอย่างในรูปที่ 2-10 (ก) โครงสร้างที่รองรับ

บล็อก 'C' ในสถานะปัจจุบันคือโครงสร้าง  ซึ่งต่างจากในคำตอบที่โครงสร้าง  เป็นโครงสร้างที่รองรับบล็อก 'C' ดังนั้นสำหรับบล็อก 'C' ที่สถานะปัจจุบันที่มีบล็อก 2 ก้อนอยู่ในโครงสร้างรองรับที่ผิดจึงได้แต้มเท่ากับ -2 เมื่อคำนวณคะแนนสำหรับบล็อกทุกก้อนก็จะได้คะแนนตามนี้คือ $A = 0, B = -1, C = -2, D = 0$ ซึ่งทำให้ได้ค่ารวมของฟังก์ชันเท่ากับ -3 หน่วย (บล็อก A ได้ศูนย์แต้มเพราะว่าไม่มีโครงสร้างที่รองรับมันอยู่เลย เนื่องจากโครงสร้างไม่รวมถึงโต๊ะ) ในกรณีของสถานะปัจจุบันในรูปที่ 2-10 (ข) จะได้คะแนนตามนี้คือ $A = 0, B = 2, C = 1, D = 0$ ซึ่งทำให้ได้ค่ารวมของฟังก์ชันเท่ากับ 3 หน่วย

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



รูปที่ 2-11 ค่าฮิวริสติก h_2 สำหรับสถานะต่างๆ ในรูปที่ 2-5

จะเห็นว่าฟังก์ชันฮิวริสติก h_2 สามารถแยกความแตกต่างระหว่างสถานะลูกทุกตัวของสถานะเริ่มต้นได้อย่างถูกต้อง นอกจากนั้นฟังก์ชันนี้ยังบอกความใกล้กับคำตอบของสถานะทุกตัวได้อย่างสมบูรณ์แบบ เมื่อดูเส้นทางจากสถานะเริ่มต้นจนถึงคำตอบ จะพบว่าในแต่ละระดับ สถานะที่มีค่าฮิวริสติกสูงกว่าสถานะอื่นๆ ในระดับเดียวกันล้วนเป็นสถานะที่นำไปสู่คำตอบได้โดยตรงทั้งสิ้น

ฟังก์ชันฮิวริสติกสำหรับปัญหา 8-Puzzle

ตัวอย่างของฟังก์ชันฮิวริสติกสำหรับปัญหา 8-Puzzle แสดงในสมการด้านล่างนี้

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

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

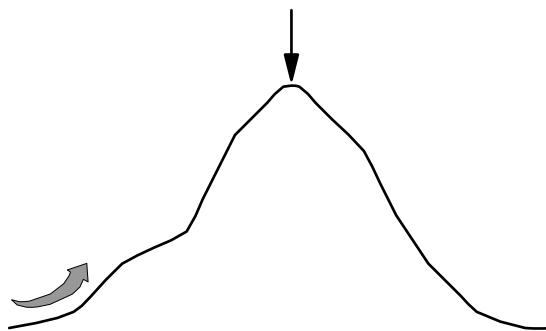
ฟังก์ชัน
แมนฮัตตัน

ฟังก์ชันนี้เรียกว่าฟังก์ชันแมนฮัตตัน ฟังก์ชันนี้สามารถแปลความหมายอย่างง่ายคือ การหาค่าจำนวนครั้งที่ต้องขยับแผ่นป้ายตั้งแต่แผ่นที่ 1 ถึงแผ่นที่ 8 จากตำแหน่งปัจจุบันตาม

จากตัวอย่างของการนิยามฟังก์ชันฮิวริสติกทั้งสามด้านบน จะเห็นแนวทางการนิยามฟังก์ชันฮิวริสติกเหล่านี้ ซึ่งโดยมากเรามักนิยามฟังก์ชันเพื่อคำนวณความคล้าย (ความต่าง) กับคำตอบ และวัดเป็นตัวเลขไปใช้ในการค้นหาต่อไป ฟังก์ชันฮิวริสติกที่ดีต้องคำนวณง่าย ไม่ยุ่งยากซับซ้อนมากจนกระทั่งเสียเวลาคำนวณมหาศาล ข้อสังเกตอีกอย่างก็คือการสร้างฟังก์ชันฮิวริสติกอาจทำได้โดยตัดเงื่อนไขของการใช้ตัวกระทำการออก อย่างเช่นในตัวอย่างของ 8-Puzzle นั้น ตัวกระทำเพื่อเลื่อนแผ่นป้ายมีเงื่อนไขว่า แผ่นป้ายจะเลื่อนได้ก็ต่อเมื่อแผ่นป้ายนั้นติดกับช่องว่าง การนิยามฟังก์ชันแมนฮัตตันนั้นเสมือนกับว่าแผ่นป้ายเลื่อนไปยังตำแหน่งต่างๆ ได้โดยไม่ต้องติดเงื่อนไขว่า แผ่นป้ายนั้นอยู่ติดกับช่องว่างหรือไม่ แล้วนับการเลื่อนแผ่นป้ายทั้งหมดเป็นการนิยามฟังก์ชัน เป็นต้น

2.4.2 อัลกอริทึมปีนเขา

ฟังก์ชันฮิวริสติกที่นิยามขึ้นในข้างต้นนั้น สามารถนำมาช่วยกระบวนการค้นหาเพื่อให้ได้คำตอบอย่างรวดเร็วและมีประสิทธิภาพ วิธีการที่จะนำฟังก์ชันฮิวริสติกมาใช้มีหลายวิธีด้วยกันขึ้นอยู่กับว่าจะใช้ในลักษณะใด เช่นเลือกสถานะที่มีค่าฮิวริสติกดีขึ้น แล้วเดินไปยังสถานะนั้นเลยโดยไม่ต้องสนใจสถานะที่มีค่าฮิวริสติกต่ำกว่าสถานะปัจจุบัน หรือว่าจะเก็บสถานะทุกตัวไว้แม้ว่าค่าฮิวริสติกจะแย่ลง แล้วพิจารณาสถานะเหล่านี้ทีหลัง เป็นต้น ในส่วนต่อไปนี้จะกล่าวถึงอัลกอริทึมต่างๆ ที่นำฟังก์ชันฮิวริสติกมาช่วยในการค้นหาคำตอบ โดยเริ่มจากอัลกอริทึมปีนเขา (hill-climbing algorithm)



รูปที่ 2-14 อัลกอริทึมปีนเขา

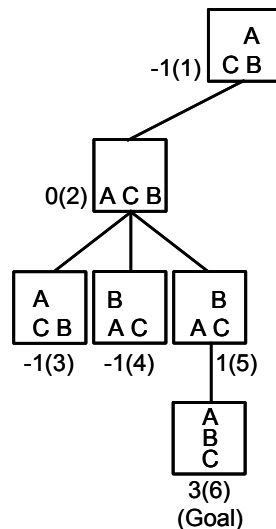
การค้นหาแบบนี้เปรียบเสมือนการปีนไปสู่อยอดเขาดังแสดงในรูปที่ 2-14 ซึ่งอัลกอริทึมจะขึ้นในแนวตั้งตลอด ถ้าเจอทางแยกเราก็จะไปแยกที่ตรงดิ่งขึ้นไปจนกระทั่งถึงยอดเขา ความสูงจากฐานภูเขาจนถึงตำแหน่งที่อยู่ในปัจจุบันก็จะเปรียบเหมือนค่าฮิวริสติก (ในกรณีนี้เราต้องการหาค่าสูงสุด) ซึ่งในที่นี้ยิ่งมากยิ่งดี อัลกอริทึมแสดงในตารางที่ 2-4

ตารางที่ 2-4 อัลกอริทึมปีนเขาอย่างง่าย

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.

ตัวอย่างการใช้ฟังก์ชันฮิวริสติก h_2 โดยอัลกอริทึมปีนเขาอย่างง่ายในตารางที่ 2-4 กับปัญหาโลกของบล็อกแสดงในรูปที่ 2-15



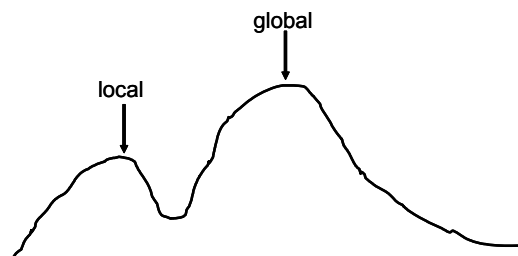
รูปที่ 2-15 ตัวอย่างอัลกอริทึมปีนเขากับปัญหาโลกของบล็อก

ตัวเลข $h(i)$ ในรูปแสดงว่า สถานะที่ i มีค่าฮิวริสติกเท่ากับ h จากรูปจะเห็นได้ว่า เริ่มต้นจากสถานะที่ 1 ที่มีค่าฮิวริสติกเท่ากับ -1 อัลกอริทึมป็นเขาใช้ตัวกระทำการเพื่อสร้างสถานะลูกตัวแรกของสถานะที่ 1 แล้ววัดค่าฮิวริสติกได้ 0 ซึ่งมีค่าดีขึ้น ถ้าสังเกตจากรูปที่ 2-5 จะพบว่าสถานะที่ 1 มีสถานะลูกทั้งหมด 3 ตัว แต่ในกรณีของอัลกอริทึมป็นเขานี้ เมื่อได้สถานะลูกตัวแรกซึ่งมีค่าฮิวริสติกดีขึ้น อัลกอริทึมจะไม่สร้างสถานะลูกที่เหลืออีก 2 ตัว และจะไม่มีการย้อนกลับมาที่สถานะลูกทั้งสองนี้ แม้ว่าหลังจากนี้อัลกอริทึมจะค้นไม่พบคำตอบ กล่าวคือเป็นการตัดทางเลือกทิ้งไปเลย ซึ่งการทำเช่นนี้แม้ว่าจะมีโอกาสไม่พบคำตอบ แต่ก็ก็มีข้อดีที่เป็นการช่วยลดเวลาและปริภูมิที่ทำการค้นหาจะลดลงอย่างมาก

จากนั้นอัลกอริทึมมาที่สถานะที่ 2 แล้วเริ่มสร้างสถานะลูก ได้สถานะที่ 3 ที่มีค่าฮิวริสติก -1 ซึ่งแยกลง ในกรณีที่แยกลงเช่นนี้ อัลกอริทึมจะไม่ไปยังสถานะลูกตัวนี้ และสร้างสถานะลูกตัวต่อไป โดยใช้ตัวกระทำที่เหลือ ได้สถานะที่ 4 มีค่าฮิวริสติกเท่ากับ -1 ไม่ดีขึ้นเช่นกัน จึงสร้างสถานะลูกตัวถัดไป เป็นสถานะที่ 5 มีค่าฮิวริสติกเท่ากับ 1 เป็นค่าที่ดีขึ้น อัลกอริทึมจะมายังสถานะนี้และค้นพบคำตอบในที่สุด

อัลกอริทึมป็นเขานี้จะมีประสิทธิภาพมากดังเช่นแสดงในตัวอย่างนี้ซึ่งกระจายสถานะทั้งสิ้นเพียง 6 ตัวแล้วพบคำตอบ เปรียบเทียบกับอัลกอริทึมการค้นหาแนวกว้างก่อนซึ่งใช้สถานะทั้งสิ้นถึง 11 ตัว อย่างไรก็ตามอัลกอริทึมนี้จะมีประสิทธิภาพมาก ถ้าใช้ฟังก์ชันฮิวริสติกที่ดีมากๆ ดังเช่นฟังก์ชัน h_2 ซึ่งเป็นฟังก์ชันที่สมบูรณ์มาก ในกรณีที่ฟังก์ชันฮิวริสติกไม่ดีนัก อัลกอริทึมนี้ก็อาจหลงเส้นทางได้ และอาจไม่พบคำตอบแม้ว่าปริภูมิที่กำลังค้นหามีคำตอบอยู่ด้วยก็ตาม สาเหตุของการหลงเส้นทางประการหนึ่งมาจากการเลือกสถานะลูก ซึ่งอัลกอริทึมจะไม่ได้พิจารณาสถานะลูกทุกตัว โดยเมื่อพบสถานะลูกตัวใดตัวหนึ่งที่ดีขึ้น ก็จะเลือกเส้นทางนั้นในทันที อัลกอริทึมนี้สามารถดัดแปลงเล็กน้อยให้พิจารณาสถานะลูกทุกตัวให้ครบก่อน แล้วเลือกสถานะลูกตัวที่มีค่าฮิวริสติกสูงสุด เมื่อทำเช่นนี้ก็จะทำให้อัลกอริทึมได้พิจารณาเส้นทางที่ดีที่สุด ณ ขณะหนึ่งๆ ได้ดีขึ้น เราเรียกอัลกอริทึมที่ดัดแปลงนี้ว่าอัลกอริทึมป็นเขาชันสุด (steepest ascent hill-climbing)

2.4.3 อัลกอริทึมอบเหนียวจำลอง

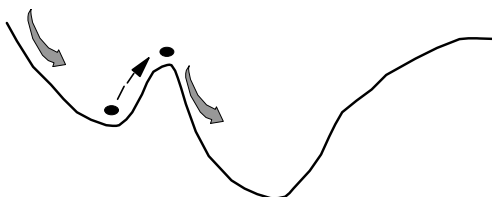


รูปที่ 2-16 ตัวอย่างปัญหาที่เกิดขึ้นกับอัลกอริทึมป็นเขา

ค่าดีที่สุดเฉพาะที่
และ
ค่าดีสุดวงกว้าง

พิจารณาตัวอย่างในรูปที่ 2-16 ซึ่งแสดงค่าฮิวริสติกด้วยความสูงจากฐานที่เกิดขึ้นระหว่างการค้นหาด้วยอัลกอริทึมป็นเขา ปัญหาที่เกิดขึ้นก็คือ อัลกอริทึมป็นเขาจะค้นพบสถานะที่มีค่าดีที่สุดเฉพาะที่ (local optimum) เท่านั้น ไม่สามารถค้นพบค่าดีสุดวงกว้าง (global optimum) ได้ เนื่องจากการค้นหาที่ติดอยู่ที่สถานะดีที่สุดเฉพาะที่แล้ว พบว่าสถานะใหม่ที่เกิดขึ้นจะมีค่าฮิวริสติกแย่งลงทั้งหมด ทำให้การค้นหาหยุดที่สถานะนั้น ปริภูมิสถานะจำนวนมากที่มีลักษณะของค่าฮิวริสติกดังรูปด้านบนนี้ และในปัญหาจำนวนมาก เรามักพบปริภูมิสถานะที่มีสถานะดีที่สุดเฉพาะที่มากกว่าหนึ่งสถานะ ทำให้การค้นหาประสบความสำเร็จลำบากในการค้นหาค่าดีที่สุดสุดวงกว้าง

อัลกอริทึมการอบเหนียวจำลอง (simulated annealing algorithm) ถูกออกแบบมาเพื่อให้สามารถหลุดออกจากสถานะดีที่สุดเฉพาะที่ โดยใช้แนวคิดอุณหพลศาสตร์ของกระบวนการอบเหนียว ซึ่งเป็นขั้นตอนการลดอุณหภูมิลงอย่างช้าๆ ระหว่างการหลอมเพื่อให้ได้โลหะที่อยู่ในสภาวะที่เหมาะสมที่สุด เป็นโลหะเหนียว ไม่เปราะ แนวคิดอธิบายให้เข้าใจได้ดีโดยใช้รูปที่ 2-17



รูปที่ 2-17 อัลกอริทึมการอบเหนียวจำลอง

เนื่องจากแนวคิดนี้เป็นการหาค่าต่ำสุด จึงใช้รูปที่กลับหัวกลับหางกับรูปที่ 2-16 ปัญหาของค่าดีที่สุดเฉพาะที่ (ในรูปนี้คือค่าต่ำสุดเฉพาะที่) ก็คือเมื่อการค้นหาแบบป็นเขา (ในกรณีนี้

คือการค้นหาแบบลงเหว) มาตกที่สถานะนี้ การค้นหาที่จะหยุด วิธีที่จะแก้ปัญหาการตกที่สถานะนี้ได้ก็คือ ต้องตั้งการค้นหาให้ขึ้นมาให้ได้เพื่อจะได้ค้นต่อไปจนพบค่าที่ดีที่สุดกว้าง

อัลกอริทึมการอบเหนียวจำลองจะยอมให้การค้นหาวิ่งไปในทิศทางที่ไม่ดีได้ในช่วงเริ่มต้นของกระบวนการค้นหา เพื่อเป็นการสำรวจทั่วๆ แบบหยาบก่อน แล้วจึงค่อยๆ ค้นหาอย่างละเอียดเมื่อเวลาผ่านไป ด้วยแนวคิดเช่นนี้ทำให้เราคาดหวังว่า ผลลัพธ์สุดท้ายที่ได้จะไม่ขึ้นกับสถานะเริ่มต้นมากนัก เพื่อให้เข้าใจถึงอัลกอริทึมนี้ ขออธิบายการอบเหนียวของโลหะโดยย่อดังนี้

การอบเหนียวของโลหะเป็นการหลอมโลหะจนละลาย (ทำให้โลหะอยู่ในสถานะที่มีพลังงานสูง) แล้วค่อยๆ ลดอุณหภูมิลงทีละน้อยจนโลหะเปลี่ยนกลับมาอยู่ในสถานะของแข็ง จุดมุ่งหมายคือพยายามทำให้โลหะกลับมาเป็นของแข็งในสถานะสุดท้ายที่มีพลังงานต่ำสุด ซึ่งโดยทั่วไปตามธรรมชาติ สสารจะพยายามเปลี่ยนตัวเองจากสถานะที่มีพลังงานสูงไปสู่สถานะพลังงานต่ำ แต่ก็มีแนวโน้มจะเป็นที่สสารเปลี่ยนจากพลังงานต่ำไปพลังงานสูงอยู่บ้าง ความน่าจะเป็นนี้สามารถคำนวณได้โดยสมการด้านล่างนี้

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

โดยที่ ΔE เป็นระดับพลังงานที่เปลี่ยนไป (เป็นค่าบวก) T เป็นอุณหภูมิ และ k เป็นค่าคงที่ของโบลต์ซมันน์ (Boltzmann) การเปลี่ยนแปลงจากระดับพลังงานสูงไปต่ำนั้น มีความน่าจะเป็นที่จะเกิดในช่วงเริ่มต้นมากกว่าในช่วงปลายของกระบวนการอบเหนียว อัตราการลดอุณหภูมิในการอบเหนียวเรียกว่า *หมายเหตุกำหนดการอบเหนียว (annealing schedule)* ถ้าหมายเหตุกำหนดการอบเหนียวเร็ว กล่าวคือลดอุณหภูมิลงอย่างรวดเร็ว โลหะก็มีโอกาสเข้าสู่สถานะสุดท้ายที่มีพลังงานสูงอยู่ เราจึงต้องพิจารณาหมายเหตุกำหนดการอบเหนียวไม่ให้เร็วเกินไป แต่ถ้าช้าไปก็ทำให้เสียเวลาโดยไม่จำเป็นเช่นกัน

แนวคิดของการอบเหนียวจำลองก็ได้จากการล้อเลียนการอบเหนียวของโลหะ โดยเป็นการค้นหาแบบปีนเขา (ลงเหว) แบบหนึ่ง ซึ่งการค้นหาสามารถไปในทิศทางที่ไม่ดีได้ โดยเฉพาะในช่วงต้นของกระบวนการค้นหา เพื่อสำรวจบริเวณที่น่าจะนำไปสู่คำตอบที่ดีที่สุด เราได้สมการความน่าจะเป็นสำหรับการอบเหนียวจำลองดังนี้

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

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

สมการนี้เมื่อนำไปใช้ร่วมกับการค้นหาแบบป็นเขา ก็จะช่วยให้กระบวนการค้นหาสามารถหลุดออกจากค่าที่ดีที่สุดเฉพาะที่ได้ตรงกับความต้องการของเรา ตัวอย่างเช่นเมื่อเราอยู่ที่สถานะปัจจุบันมีค่าฮิวริสติกเท่ากับ A และเมื่อสร้างสถานะลูกที่มีค่าฮิวริสติกเท่ากับ B ซึ่งแย่ง การค้นหาอาจไปยังสถานะลูกนี้ได้ โดยคำนวณค่าความน่าจะเป็น (p) ตามสมการด้านบน ได้เป็น $p = e^{-|A-B|/T}$ ค่าที่ได้นี้จะอยู่ระหว่าง 0 ถึง 1 จากนั้นเราจะสุ่มตัวเลข (random(0, 1)) ขึ้นมาหนึ่งตัว ถ้าค่า p มีค่ามากกว่า ก็จะได้รับสถานะลูกเป็นสถานะต่อไปได้ ค่า T เป็นพารามิเตอร์ของอัลกอริทึมนี้ที่เราสามารถปรับแต่งให้เหมาะสมสำหรับปัญหาหนึ่งๆ ที่เราสนใจ โดยช่วงเริ่มต้นกำหนดให้เป็นค่ามากแล้วค่อยๆ ลดลงเมื่อการค้นหาดำเนินต่อไป

เมื่อพิจารณาสมการนี้ เราจะได้คุณสมบัติที่เราต้องการดังนี้คือ เมื่อ T มาก (ในช่วงต้นของกระบวนการค้นหา) จะได้ p มีค่ามาก คือเรายอมให้การค้นหาไปในทิศทางที่ไม่ดีได้ง่ายหน่อยในช่วงต้นการค้นหา แต่เมื่อ T น้อย (ในช่วงปลายของกระบวนการค้นหา) จะได้ p มีค่าน้อยคือการค้นหาเริ่มเข้าสู่คำตอบ ก็ไม่ควรให้การค้นหากระโดดไปยังสถานะที่แย่ง เมื่อพิจารณา ΔE จะพบว่า เมื่อ ΔE มีค่ามากจะได้ว่า p มีค่าน้อย กล่าวคือการก้าวกระโดดจากสถานะที่ดีไปยังสถานะที่ไม่ดีที่การก้าวกระโดดนั้นเป็นก้าวใหญ่ (มีการเปลี่ยนแปลงค่าฮิวริสติกมาก) จะเกิดขึ้นได้ยากกว่าการก้าวกระโดดที่เป็นก้าวเล็ก (มีการเปลี่ยนแปลงค่าฮิวริสติกน้อย) ซึ่งเป็นคุณสมบัติที่น่าสนใจ เนื่องจากเราเชื่อว่าสถานะที่เป็นคำตอบมักจะเป็นหลุมลึกๆ ถ้าจะหลุดออกจากหลุมลึกได้ต้องก้าวใหญ่ ซึ่งไม่ควรให้เกิดขึ้นได้ง่าย ส่วนสถานะที่ดีที่สุดเฉพาะที่มักเป็นหลุมตื้นๆ ดังนั้นจึงให้อโอกาสหลุดลอดได้ง่ายหน่อย อัลกอริทึมการอบเหนียวจำลองแสดงในตารางที่ 2-5

อัลกอริทึมจะมีลักษณะคล้ายกับอัลกอริทึมป็นเขา แต่สามารถเลือกสถานะที่มีค่าฮิวริสติกแย่งได้ ดังนั้นเมื่อสิ้นสุดกระบวนการค้นหา สถานะตัวสุดท้ายที่พิจารณาอยู่อาจไม่ใช่สถานะที่มีค่าฮิวริสติกดีที่สุด ดังนั้นเราจึงจำเป็นต้องจำสถานะที่มีค่าฮิวริสติกดีที่สุดที่ผ่านมาไว้ โดยเก็บค่าไว้ในตัวแปรชื่อ BEST-SO-FAR ในตอนเริ่มกระบวนการค้นหา จะกำหนดค่าคงที่ตัวหนึ่งเป็นอุณหภูมิการอบเหนียว (T) และลดอุณหภูมิลงตามความเหมาะสม (ในขั้นตอนที่ 5.3 ในอัลกอริทึม) ในกรณีที่เรากำลังพิจารณาสถานะใหม่ (new state) ตัวหนึ่งอยู่ ถ้าพบว่าสถานะนี้มีค่าฮิวริสติกดีขึ้น ก็จะค้นหาต่อไปยังสถานะใหม่นี้ และปรับค่าตัวแปร BEST-SO-FAR แต่ถ้าหากว่าค่าฮิวริสติกแย่ง เราจะคำนวณค่าความน่าจะเป็น ($e^{-\Delta E/T}$) ที่จะไปยังสถานะใหม่นี้ การประยุกต์ใช้อัลกอริทึมนี้ ผู้ใช้จำเป็นต้องกำหนดพารามิเตอร์ในอัลกอริทึมให้เหมาะสมสำหรับปัญหาที่กำลังพิจารณาอยู่ด้วย พารามิเตอร์นี้ได้แก่ ค่าอุณหภูมิเริ่มต้นและปริมาณการลดอุณหภูมิ

ตารางที่ 2-5 อัลกอริทึมการอบเหนียวจำลอง

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 อัลกอริทึมที่ดีที่สุดก่อน

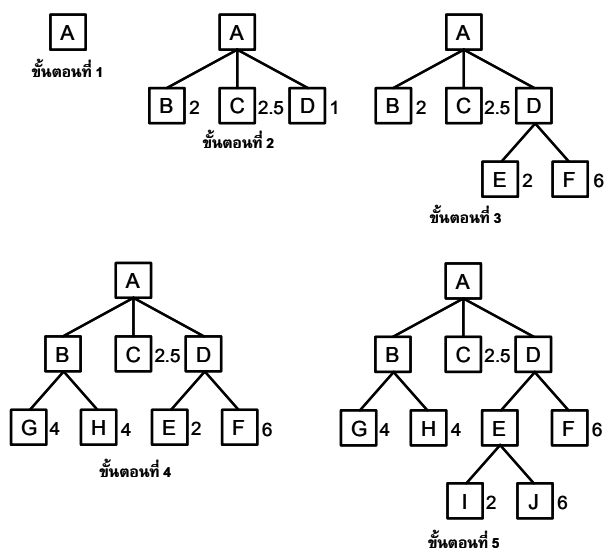
อัลกอริทึมที่ดีที่สุดก่อน (best-first search) จะเก็บสถานะทุกตัวโดยไม่มีการตัดทิ้งไป ต่างจากอัลกอริทึมปิ่นเขาที่เมื่อเลือกเส้นทางหนึ่งแล้ว ตัวเลือกอื่นที่เป็นลูกของสถานะปัจจุบันจะถูกตัดทิ้งไป การไม่ตัดทิ้งจึงทำให้อัลกอริทึมนี้ไม่พลาดเส้นทางที่นำไปสู่คำตอบ โดยในแต่ละขั้นตอนจะเลือกสถานะที่มีค่าฮิวริสติกดีที่สุด โดยพิจารณาสถานะทุกตัวที่ยังไม่ถูกกระจาย (สถานะที่ยังไม่ได้สร้างสถานะลูก) อัลกอริทึมแสดงในตารางที่ 2-6

ตารางที่ 2-6 อัลกอริทึมที่ดีที่สุดก่อน

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) มีไว้เพื่อปรับเส้นทางและต้นทุนใหม่ เนื่องจากว่าสถานะหนึ่งๆ อาจเข้าถึงจากหลายเส้นทาง (ดังที่ได้กล่าวข้างต้นแล้วว่าปริภูมิสถานะโดยทั่วไปเป็นกราฟ) ในกรณีที่เราพบเส้นทางใหม่ที่น่ามาสู่สถานะที่เคยสร้างแล้ว และเส้นทางใหม่ดีกว่าหรือมีต้นทุนหรือจำนวนครั้งจากสถานะเริ่มต้นมายังสถานะนี้น้อยกว่าเส้นทางเดิม ก็ให้แก้ไขเส้นทางให้ถูกต้อง ตัวอย่างของการค้นหาแบบอัลกอริทึมที่ดีที่สุดก่อนแสดงในรูปที่ 2-18



รูปที่ 2-18 อัลกอริทึมที่ดีที่สุดก่อน

สมมติว่า 'A' เป็นสถานะเริ่มต้น สถานะลูกทุกตัวของ 'A' (คือ 'B', 'C' และ 'D') ถูกสร้างขึ้นในขั้นตอนที่ 2 จากนั้นเมื่อวัดค่าฮิวริสติกของสถานะทุกตัวได้ว่า 'D' มีค่าดีที่สุด (ในที่นี้ยิ่งน้อยยิ่งดี) จึงนำ 'D' มากระจายสถานะลูก ได้สถานะ 'E' และ 'F' ซึ่งมีค่าฮิวริสติก 2 และ 6 ตามลำดับ เมื่อเปรียบเทียบค่าฮิวริสติกของสถานะทุกตัวที่ยังไม่ได้กระจาย (เก็บไว้ใน OPEN ในตารางที่ 2-6) จะได้ว่า 'B' มีค่าดีที่สุด (เท่ากับ 'E' แต่ในที่นี้กำหนดให้กรณีค่าเท่ากันให้นำสถานะที่สร้างก่อนมาทำก่อน) จึงนำ 'B' มากระจายต่อ และต่อจากนั้นในขั้นตอนที่ 5 สถานะ 'E' ถูกกระจายต่อไปตามลำดับ เป็นเช่นนั้นจนกระทั่งพบคำตอบหรือไม่สามารถสร้างสถานะใหม่ได้อีก

2.4.5 อัลกอริทึม A*

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

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

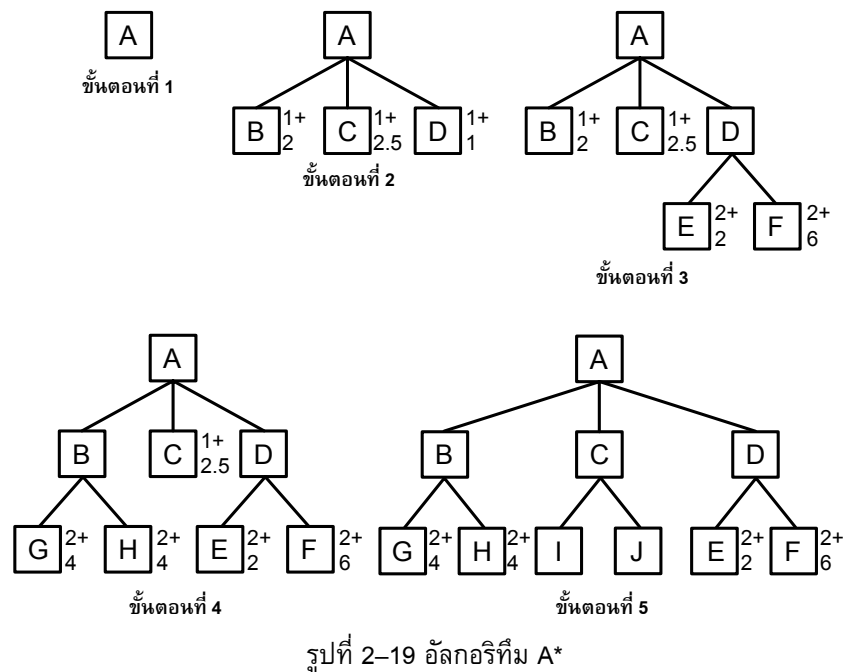
โดยที่ g คือฟังก์ชันที่คำนวณต้นทุนจากสถานะเริ่มต้นมายังสถานะปัจจุบัน h' คือฟังก์ชันที่ประมาณต้นทุนจากสถานะปัจจุบันไปยังคำตอบ ดังนั้น f จึงเป็นฟังก์ชันที่ประมาณต้นทุนจากสถานะเริ่มต้นไปยังคำตอบ (ยิ่งน้อยยิ่งดี)

เรามองได้ว่าฟังก์ชัน h' คือฟังก์ชันฮิวริสติกที่เราเคยใช้ในการค้นหาอื่นๆ ก่อนหน้านี้ เช่นอัลกอริทึมปีนเขา อัลกอริทึมที่สุดก่อน เป็นต้น ในที่นี้เราใส่เครื่องหมาย ' เพื่อแสดงว่าฟังก์ชันนี้เป็นฟังก์ชันประมาณของฟังก์ชันจริงที่ไม่รู้ (เราทำได้แค่ประมาณว่า h' คือต้นทุนจากสถานะปัจจุบันไปยังคำตอบ เราจะรู้ต้นทุนจริงก็ต่อเมื่อเราได้ทำการค้นหาจริงจนไปถึงคำตอบแล้ว) ส่วน g เป็นฟังก์ชันที่คำนวณต้นทุนจริงจากสถานะเริ่มต้นมายังสถานะปัจจุบัน (จึงไม่ได้ใส่เครื่องหมาย ') เพราะเราสามารถหาต้นทุนจริงได้เนื่องจากได้ค้นหาจากสถานะเริ่มต้นจนมาถึงสถานะปัจจุบันแล้ว ส่วน f ก็เป็นเพียงแค่ฟังก์ชันประมาณโดยการรวมต้นทุนทั้งสองคือ h' กับ g

อัลกอริทึม A* จะทำการค้นหาโดยวิธีเดียวกันกับอัลกอริทึมที่สุดก่อนทุกประการ ยกเว้นฟังก์ชันฮิวริสติกที่ใช้เปลี่ยนมาเป็น f (ต่างจากอัลกอริทึมที่สุดก่อนที่ใช้ h') และด้วยการใช้ f อัลกอริทึม A* จึงให้ความสำคัญกับสถานะหนึ่งๆ 2 ประการคือ (1) สถานะที่ดีต้องมี h' ดีคือต้นทุนเพื่อจะนำไปสู่คำตอบหลังจากนี้ต้องน้อย และ (2) ต้นทุนที่จ่ายไปแล้วกว่าจะถึงสถานะนี้ (g) ต้องน้อยด้วย เราจึงได้ว่า A* จะค้นหาเส้นทางที่ให้ต้นทุนโดยรวมน้อยสุดตาม

ค่า f ซึ่งต่างจากอัลกอริทึมที่ดีที่สุดก่อนที่เน้นความสำคัญของสถานะที่ต้นทุนหลังจากนี้จะนำไปสู่ค่าตอบต่อน้อย โดยไม่สนใจว่าต้นทุนที่จ่ายไปแล้วกว่าจะนำมาถึงสถานะนี้ต้องเสียไปเท่าไร

รูปที่ 2-19 แสดงการค้นหาด้วยอัลกอริทึม A^* กับสถานะในรูปที่ 2-18 โดยสมมติให้ต้นทุนหรือระยะห่างระหว่างสถานะพ่อแม่ไปยังสถานะลูกเท่ากับ 1 หน่วย เช่น ต้นทุนจริง (g) จาก 'A' ไปยัง 'B', 'C' หรือ 'D' มีค่าเท่ากับ 1 หน่วย



จากรูปจะเห็นว่าในขั้นตอนที่ 4 สถานะ 'C' จะถูกเลือกมากระจายโดยอัลกอริทึม A^* เนื่องจากมีค่า f น้อยสุดเท่ากับ 3.5 ซึ่งน้อยกว่า 'E' ที่มีค่าเท่ากับ 4 แม้ว่าค่า h ของ E จะน้อยกว่า ซึ่งต่างจากการสร้างสถานะของอัลกอริทึมที่ดีที่สุดก่อน

2.4.6 การค้นหาตาบ

ตาบ (tabu, taboo) แปลว่า ต้องห้าม (เช่น สิ่งที่เป็นของศักดิ์สิทธิ์ จึงห้ามแตะต้อง) การค้นหาตาบเป็นเทคนิคการค้นหาที่ค่อนข้างใหม่ มักไม่ถูกอธิบายไว้ในหนังสือเกี่ยวกับปัญญาประดิษฐ์ ในหัวข้อนี้เราจะใช้เนื้อที่ค่อนข้างมากเพื่อบรรยายถึงวิธีการนี้

ในกระบวนการค้นหาตามนั้น จะทำเครื่องหมายบนเส้นทางบางเส้นทางที่ไม่สนใจจะค้นหา การทำเครื่องหมายนี้อาจทำในระดับของตัวกระทำ หรือหน่วยย่อยของตัวกระทำก็ได้ หน่วยย่อยใดที่ถูกทำเครื่องหมายไว้จะเปลี่ยน**สถานภาพต้องห้าม (tabu status)**ให้อยู่ใน**ภาวะต้องห้าม (tabu active)** กล่าวคือหน่วยย่อยนี้จะไม่ถูกนำมาใช้เพื่อสร้างเส้นทางในการค้นหา อาจเป็นเพราะเส้นทางนี้คงไม่นำไปสู่คำตอบหรืออาจเป็นเส้นทางที่เคยค้นหามาแล้ว เป็นต้น

การค้นหาตามนี้จะกำหนดสถานภาพของหน่วยย่อยโดยขึ้นกับหน่วยความจำ ซึ่งหน่วยความจำนี้จะเปลี่ยนแปลงตามเวลาและสภาพแวดล้อมในระหว่างกระบวนการค้นหา การค้นหาตามนี้มีแนวคิดที่ว่า การค้นหาที่ฉลาดจะต้องพิจารณาถึงสิ่งเหล่านี้คือ

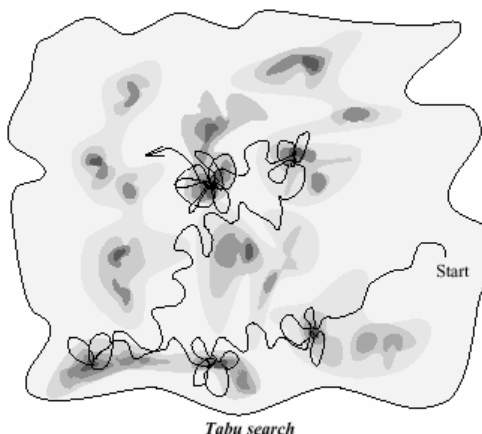
1. หน่วยความจำปรับตัว (adaptive memory) กล่าวคือหน่วยความจำจะต้องมีการปรับตัวเพื่อให้การค้นหาเป็นไปอย่างมีประสิทธิภาพตามสภาพการค้นหา ณ ตำแหน่งนั้นๆ
2. การสำรวจแบบตอบสนอง (responsive exploration) ซึ่งการค้นหานี้ เราจะต้องทำการสำรวจอย่างถี่ถ้วนในเส้นทางที่ดี เพื่อหวังว่าอาจจะได้เส้นทางที่ดีกว่าเดิม นอกจากนั้นแล้ว เรายังต้องค้นหาในเส้นทางที่ไม่ดีด้วย เพราะบางครั้งเส้นทางที่เลวอาจจะให้ข้อมูลมากกว่าเส้นทางที่ดีเพื่อหาเส้นทางใหม่ที่ดีขึ้นได้

หน่วยความจำที่ใช้ในการค้นหานี้มี 2 ชนิดคือ

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

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

¹ 'ผลเฉลย' ในบริบทของการค้นหาตามนี้เทียบได้กับ 'สถานะ' ของการค้นหาในปริภูมิสถานะ ซึ่งผลเฉลยนี้อาจไม่เป็นผลเฉลยที่ดีที่สุด



รูปที่ 2-20 การค้นหาตาบู่

รูปที่ 2-20 แสดงแนวคิดการค้นหาตาบู่ ในรูปบริเวณที่มีสีเข้มแสดงถึงบริเวณที่มีค่าฮิวริสติกที่ดี จากรูปจะเห็นว่าจากจุดเริ่มต้น (Start ในรูป) การค้นหาจะพยายามไปสู่อบริเวณที่มีค่าฮิวริสติกสูง (สีเข้ม) ซึ่งก็จะพบจุดสูงสุดเฉพาะที่ จากนั้นก็จะสำรวจบริเวณรอบๆ และค่อยเปลี่ยนไปยังเส้นทางใหม่ที่ต่างไปจากเดิม ซึ่งอาจต้องไปในเส้นทางใหม่ที่มีค่าฮิวริสติกไม่ดีด้วย ในที่สุดก็คาดหวังว่าการค้นหาจะสามารถไปสู่จุดสูงสุดดวงกว้างได้

หน่วยความจำระยะสั้น

ปัญหาที่เราสนใจคือ หาค่า x ที่ทำให้ฟังก์ชัน $f(x)$ มีค่าต่ำสุด โดยกำหนดให้ $N(x)$ เป็นสถานะข้างเคียง (neighborhood) ของ x ในปริภูมิสถานะ X กล่าวคือ $N(x)$ คือสถานะลูกที่สามารถสร้างได้จากตัวกระทำ

การค้นหาตาบู่ใช้หน่วยความจำ 2 ชนิดเพื่อเปลี่ยนค่า $N(x)$ คือหน่วยความจำระยะสั้น (short term memory) เพื่อใช้สำหรับเป็นหน่วยความจำตามเวลา และหน่วยความจำระยะยาว (long term memory) ใช้เป็นหน่วยความจำตามความถี่ โดยให้ $N^*(x)$ แทนจุดข้างเคียงของ x ที่เปลี่ยนไป เมื่อใช้หน่วยความจำเหล่านี้มาช่วยสร้างสถานะข้างเคียง

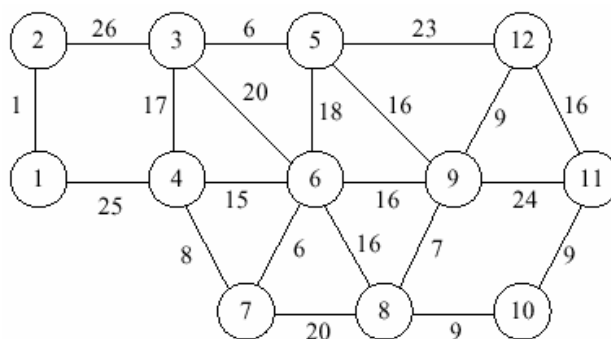
ในกรณีของหน่วยความจำระยะสั้น หน่วยความจำนี้ใช้เป็นหน่วยความจำตามเวลาเพื่อเก็บผลเฉลยหรือคุณสมบัติของผลเฉลย (solution attribute) ในการค้นหาที่เพิ่งจะผ่านมา และคุณสมบัติที่ปรากฏในผลเฉลยที่เพิ่งจะค้นหาไปจะถูกกำหนดให้มีสถานะภาพเป็น “ภาวะต้องห้าม” ซึ่งภาวะต้องห้ามก็คือการทำเครื่องหมายไว้ว่าเราไม่ต้องค้นหาเส้นทางหรือผลเฉลยอื่นๆ ถ้าเส้นทางหรือผลเฉลยนั้นๆ มีคุณสมบัติเหมือนกับผลเฉลยที่มีเพิ่งค้นหาไปเมื่อเร็วๆ นี้ เพราะจะได้ผลเฉลยที่ใกล้เคียงกันนั่นเอง และผลเฉลยอื่นๆ ที่จะพบในอนาคตที่มีคุณสมบัติเป็นภาวะต้องห้ามก็จะมีสถานะภาพเป็นภาวะต้องห้ามด้วย (ก็เพราะว่ามันมี

คุณสมบัติเหมือนกัน จึงไม่จำเป็นจะต้องไปเสียเวลาค้นหามันอีก) ดังนั้นหน้าที่ของหน่วยความจำระยะสั้นก็คือ การป้องกันการสร้างผลเฉลยบางตัวไม่ให้อยู่ใน $N^*(x)$ กล่าวคือ $N^*(x)$ จะเป็นเซตย่อยของ $N(x)$ โดยมีสถานะบางตัวที่มีสถานะภาพเป็นภาวะต้องห้ามถูกตัดออกไป (ในกรณีของหน่วยความจำระยะยาว $N^*(x)$ อาจเป็นซูเปอร์เซตของ $N(x)$) $N^*(x)$ นี้จะถูกกำหนดโดยหน่วยความจำระยะสั้น ซึ่งจะเปลี่ยนแปลงในแต่ละครั้งของการค้นหา

ตัวอย่างปัญหาต้นไม้ k กิ่งน้อยสุด

ในที่นี้จะยกตัวอย่างปัญหาต้นไม้ k กิ่งน้อยสุด เพื่อแสดงกระบวนการค้นหาตามด้วยหน่วยความจำระยะสั้น

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



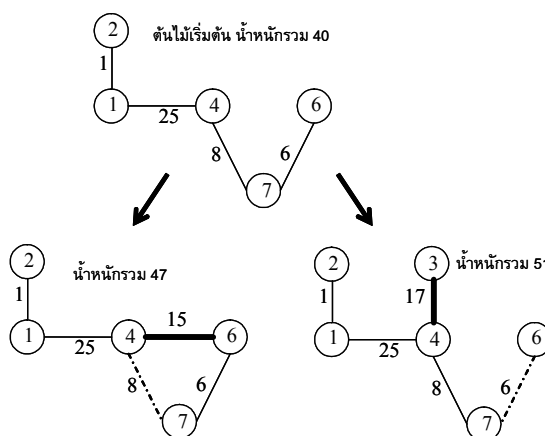
รูปที่ 2-21 ตัวอย่างปัญหาต้นไม้ k กิ่งน้อยสุด

จากกราฟข้างบน เราจะสร้างต้นไม้ที่มี 4 กิ่งโดยใช้ **อัลกอริทึมตะกราม (greedy algorithm)** ซึ่งเริ่มจากการหากิ่งแรกในกราฟที่มีน้ำหนักน้อยสุด (กิ่ง (1,2)) จากนั้นหากิ่งที่เชื่อมกับกิ่งนี้ที่มีน้ำหนักน้อยสุด ทำเช่นนี้ไปจนครบ 4 กิ่ง จะได้ผลดังตารางข้างล่างนี้

ตารางที่ 2-7 การสร้างผลเฉลยเริ่มต้นด้วยอัลกอริทึมตะกราม

ขั้นตอนที่	ตัวเลือก	กิ่งที่เลือก	น้ำหนักรวม
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 พบว่าน้ำหนักรวมที่หาได้คือ 40 เมื่อได้ผลเฉลยเริ่มต้นแล้ว ต่อไปเราก็จะสร้างผลเฉลยข้างเคียงใหม่ โดยแทนที่กิ่ง 1 กิ่งในต้นไม้ด้วยกิ่งใหม่ โดยที่ผลที่ได้ต้องเป็นต้นไม้ **รูปที่ 2-22** แสดงการสร้างผลเฉลยใหม่ 2 ตัวโดยการลบกิ่งหนึ่งกิ่ง (แสดงด้วยเส้นประในรูป) ออกจากต้นไม้เดิม และเพิ่มกิ่งหนึ่งกิ่ง (แสดงด้วยเส้นทึบในรูป) เข้าไป



รูปที่ 2-22 การสร้างผลเฉลยข้างเคียงโดยลบกิ่งหนึ่งกิ่งและเพิ่มกิ่งหนึ่งกิ่ง

การเลือกคุณสมบัติที่ใช้กำหนดสถานะภาพต้องห้ามในการสร้างต้นไม้

ในปัญหานี้กำหนดให้ กิ่งเพิ่มเข้า (added edge) และ กิ่งลบออก (dropped edge) เป็นตัวกำหนดสถานะภาพต้องห้าม โดยที่

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

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

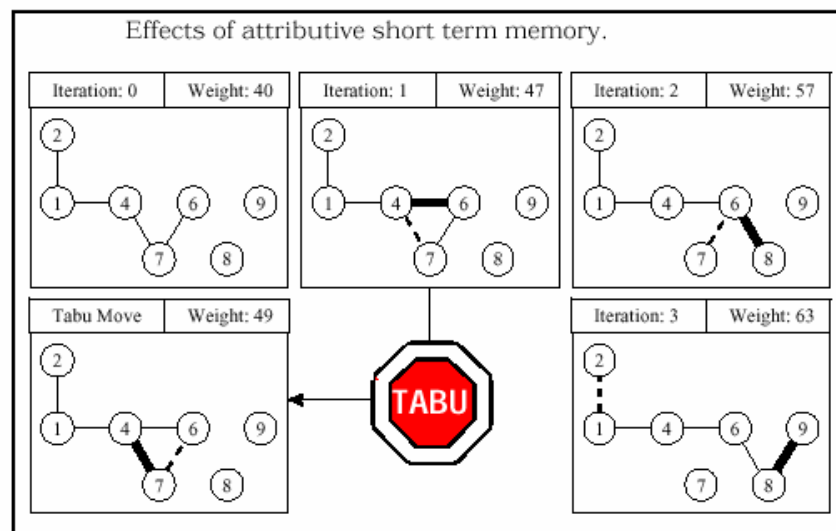
- ให้ระยะเวลาต้องห้ามของกิ่งลบออกเท่ากับ 2 หมายความว่าห้ามเพิ่มกิ่งนี้เข้าไปในอีก 2 รอบ
- ให้ระยะเวลาต้องห้ามของกิ่งเพิ่มเข้าเท่ากับ 1 หมายความว่าห้ามลบกิ่งนี้ออกไปในอีก 1 รอบ

ดังนั้นเมื่อผ่านไป 3 รอบ จะได้ผลเฉลยดัง **ตารางที่ 2-8** ต่อไปนี้

ตารางที่ 2-8 ผลเฉลยที่ได้เมื่อผ่านไป 3 รอบ

รอบที่	ระยะเวลาต้องห้าม		กึ่งเพิ่ม เข้า	กึ่งลบ ออก	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 เก็บไว้ในช่อง 'กึ่งเพิ่มเข้า' จะเก็บค่ากึ่ง (4,6) และ 'กึ่งลบออก' เก็บค่า (4,7) เพื่อให้ทราบว่าขณะนี้ได้เพิ่มและลบกึ่งใด และน้ำหนักรวมเป็นเท่าไร ในรอบที่สองจะมีการเก็บค่าระยะเวลาต้องห้ามแล้ว ในช่อง 'กึ่งเพิ่มเข้า' ก็จะเพิ่มเข้าที่หมายเลข 1 ซึ่งตรงตามข้อตกลงข้างต้นว่า ถ้าเป็น 'กึ่งเพิ่มเข้า' จะให้ระยะเวลาต้องห้ามมีค่าเป็น 1 หมายความว่าห้ามไปยุ่งกับกึ่งที่ (4,6) หนึ่งรอบและห้ามยุ่งกับกึ่ง (4,7) สองรอบ ในขณะที่รอบนี้มีการเพิ่มกึ่ง (6,8) และลบกึ่ง (6,7) ดูรูปที่ 2-23 ประกอบ ในรอบที่สามจะเห็นว่า (4,6) หายไปจากช่อง 1 ในระยะเวลาต้องห้าม และ (4,7) จะย้ายมาอยู่ในช่อง 1 แทนเพราะมันถูกห้ามสองรอบ แต่ผ่านไปหนึ่งรอบจึงยังคงถูกห้ามอีกเพียง 1 รอบเท่านั้น นอกจากนั้นในช่อง 1 ยังมีกึ่ง (6,8) เพิ่มเข้ามาอีกตัว ทำเช่นนี้ไปเรื่อยๆ

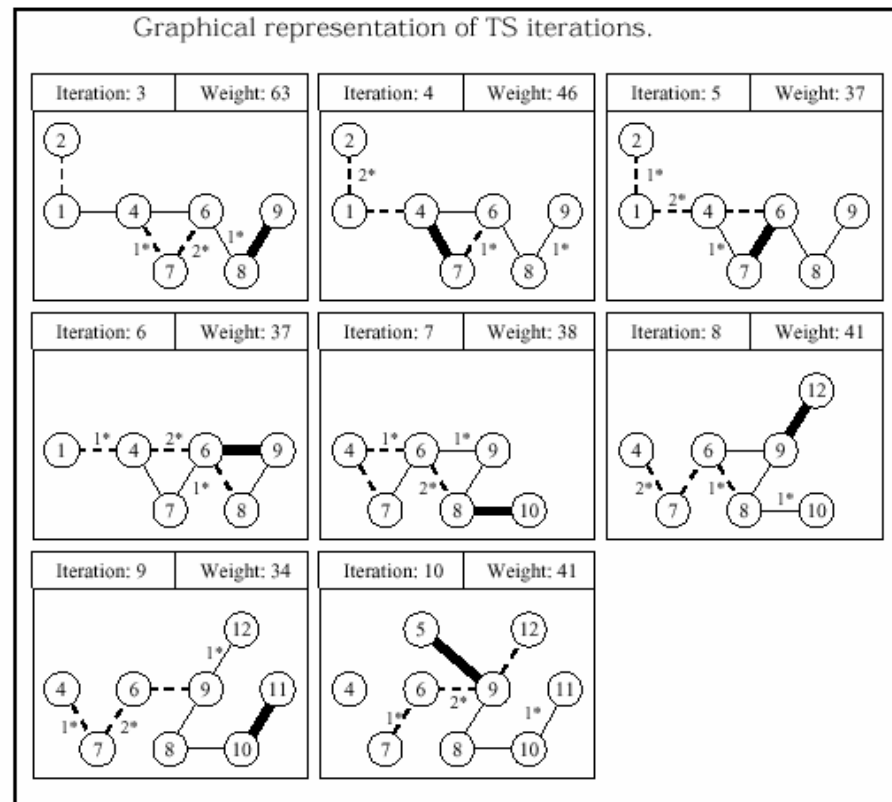


รูปที่ 2-23 ผลของการใช้หน่วยความจำระยะสั้น

เกณฑ์แห่งความหวัง

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

การค้นหาคำตอบจะทำการค้นหาไปจนกระทั่งจำนวนรอบเกินกว่าค่าขีดแบ่งที่กำหนดไว้ (ในตัวอย่างด้านบนกำหนดให้เป็น 10 รอบ) กระบวนการค้นหาแสดงในรูปที่ 2-24 และตารางที่ 2-9 ซึ่งจะพบว่าการใช้ภาวะต้องห้ามจะทำให้ประหยัดเวลาในการไปในเส้นทางต่างๆ ได้มากพอควร ตัวเลข 1* และ 2* ในรูปที่ 2-24 แสดงระยะเวลาต้องห้ามของกิ่ง



รูปที่ 2-24 ต้นไม้ที่ได้ในรอบที่ 3 ถึงรอบที่ 10 ของการค้นหาคำตอบ

ตารางที่ 2-9 ผลเฉลยที่ได้เมื่อผ่านไป 10 รอบ

รอบที่	ระยะเวลาต้องห้าม		กึ่งเพิ่ม เข้า	กึ่งลบ 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

การค้นหากล่าวมาข้างต้นเป็นการค้นหาในบริเวณใกล้ๆ กับผลเฉลยที่ดี (ผลเฉลยที่ได้จากอัลกอริทึมตะกราม) กล่าวคือเมื่อพบผลเฉลยที่ดีเฉพาะที่แล้ว ก็将继续ค้นหาให้ทั่วๆ ในบริเวณนั้นโดยใช้หน่วยความจำระยะสั้น

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 ดังนั้นเราต้องการเริ่มการค้นหาครั้งใหม่ก่อนรอบนี้โดยไม่นำรอบที่ 9 นี้มาพิจารณา

ในกรณีนี้เหตุการณ์สำคัญคือเหตุการณ์ที่เกิดในรอบที่ 0, 5, 6 เราจะรวบรวมข้อมูลทั้งหมดของทั้งสามรอบไว้ในหน่วยความจำระยะยาว ซึ่งก็คือกึ่ง (1,2), (1,4), (4,7), (6,7), (6,8), (8,9) และ (6,9) และในกรณีที่ใช้หน่วยความจำตามความถี่ (frequency-based memory) เป็นหน่วยความจำระยะยาว เราจะใช้จำนวนครั้งเพื่อกำหนดความสำคัญของแต่ละกึ่งด้วย 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)

จากนั้นจะให้กึ่งเหล่านี้มีสถานะภาพเป็นภาวะต้องห้าม เพื่อใช้เป็นตัวป้องกันการสร้างจุดเริ่มต้นใหม่ที่มีกึ่งเหมือนกับกึ่งในหน่วยความจำนี้ อย่างไรก็ตามในแต่ละขั้นตอนของการสร้างจุดเริ่มต้นใหม่นั้น (ในกรณีของตัวอย่างเราใช้อัลกอริทึมตะกราม ซึ่งจะทำการทั้งหมด 4 ขั้นตอน – มี 4 กึ่ง) จะทำการป้องกันมากถ้าเป็นขั้นตอนต้นๆ และป้องกันน้อยถ้าเป็นขั้นตอนท้ายๆ ของการสร้างจุดเริ่มต้นใหม่ (เพราะหากป้องกันมากไปอาจทำให้ไม่สามารถสร้างจุดใหม่ได้เลย)

ในกรณีของตัวอย่าง กำหนดให้การป้องกันเป็นดังต่อไปนี้

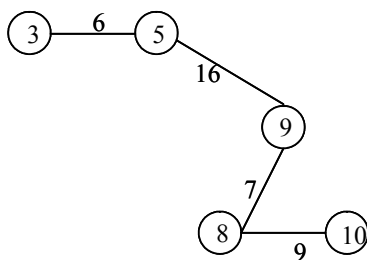
- ใน 2 ขั้นตอนแรก ห้ามมีกึ่งที่เราเก็บไว้ในหน่วยความจำเลย (คือห้ามมีกึ่งที่เราจำไว้ข้างบนของรอบ 0, 5, 6 ซึ่งก็คือกึ่ง (1,2), (1,4), (4,7), (6,7), (6,8), (8,9) และ (6,9) ดังนั้นเราจะเริ่มจาก (3,5) เป็นกึ่งแรก (ดูตารางที่ 2–10 ประกอบ)
- หลังจากนั้นยอมให้มีกึ่งในหน่วยความจำได้

ตารางที่ 2–10 กระบวนการหาจุดเริ่มต้นใหม่โดยใช้หน่วยความจำระยะยาว

ขั้นตอนที่	ตัวเลือก	กึ่งที่เลือก	น้ำหนักรวม
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) ก็เพราะว่ามีน้ำหนักน้อยที่สุดในทางเลือกทั้งหมดที่เป็นไปได้ ส่วนขั้นตอนที่ 3 และ 4 ก็เช่นเดียวกัน เราจะใช้ต้นไม้ที่ได้ (ในรูปที่ 2–25) เป็นจุดเริ่มต้น

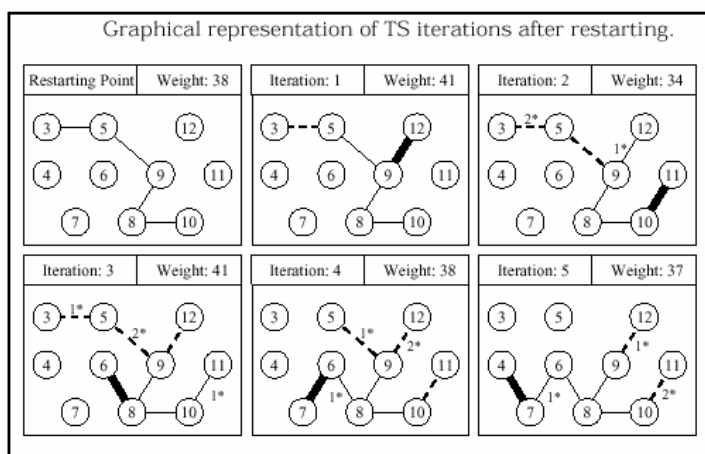
ใหม่ และใช้กระบวนการค้นหาด้วยหน่วยความจำระยะสั้นเช่นเดียวกับที่ผ่านมา ซึ่งจะได้ผลดังตารางที่ 2-11 และรูปที่ 2-26



รูปที่ 2-25 ต้นไม้ที่ได้สำหรับเริ่มการค้นหาใหม่

ตารางที่ 2-11 กระบวนการค้นหาตามเมื่อเริ่มจากผลเฉลยตัวใหม่

รอบที่	ระยะเวลาต้องห้าม		กิ่งเพิ่ม เข้า	กิ่งลบ ออก	ค่าที่ เปลี่ยน	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

เราสามารถเขียนอัลกอริทึมของการค้นหาตามโดยใช้แนวคิดด้านบนได้ดังตารางที่ 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

การศึกษาเรื่องฮิวริสติกดูเพิ่มเติมได้ใน [Polya, 1971] อัลกอริทึม A^* มีคุณสมบัติที่จะรับประกันได้ว่าเส้นทางที่ได้จากอัลกอริทึมจะเป็นเส้นทางสั้นสุด (ใช้ต้นทุนน้อยสุด) ถ้าใช้ฟังก์ชันฮิวริสติกที่ประเมินต่ำกว่าความจริง รายละเอียดและการพิสูจน์ดูได้จาก [Russell & Norvig, 1995] การค้นหาตามดูรายละเอียดเพิ่มเติมได้ใน [Glover & Laguna, 2002] นอกจากค้นหาในปฏิภูมิสถานะแล้ว 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

- จงอธิบายว่าทำไมเส้นทางไปสู่คำตอบที่ได้จากอัลกอริทึมการค้นหาแนวกว้างก่อน จึงเป็นเส้นทางสั้นสุด โดยกำหนดให้ระยะห่างหรือค่าใช้จ่ายระหว่างสถานะสองตัวใดๆ มีค่าเท่ากันหมด
- จงเปรียบเทียบข้อดีข้อเสียระหว่างอัลกอริทึมค้นหา 3 ตัวนี้คือ อัลกอริทึมค้นหาแนวกว้างก่อน อัลกอริทึมป็นเขา และอัลกอริทึมตามู
- กำหนดให้สถานะเริ่มต้นและสถานะสุดท้ายของปัญหา 8-puzzle เป็นดังต่อไปนี้

1	2	3
8	5	6
4	7	

1	2	3
4	5	6
7	8	

จงออกแบบฟังก์ชันฮิวริสติกที่ใช้กับการค้นหาป็นเขาให้สามารถค้นหาจากสถานะเริ่มต้นและค้นพบสถานะสุดท้ายด้านบนนี้ได้

- 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 $s(2,4)$, $s(4,4)$, $s(3,5)$, $s(3,3)$
- ฟังก์ชันฮิวริสติก h ของสถานะ s ใดๆ แสดงโดยรูปด้านบน เช่น $h(s(1,1))=1$, $h(s(1,3))=3$, $h(s(3,3))=0$, $h(s(5,5))=9$ เป็นต้น

จงเขียนโปรแกรมอัลกอริทึมการอบเหนียวจำลอง เพื่อค้นหาสถานะสุดท้ายจากสถานะเริ่มต้น

5. จงเขียนโปรแกรมเพื่อแก้ปัญหาการคำนวณค่าใช้จ่ายที่น้อยที่สุดในการจัดการแข่งขันเล่นหมากรุกเก็บชิงแชมป์โลก

- ในการแข่งขันเล่นหมากรุกเก็บชิงแชมป์โลกนี้ มีผู้สมัครแข่งขันทั้งหมด n คนจากคนละประเทศ ค่าใช้จ่ายในการจัดการแข่งขันสำหรับผู้สมัครจากประเทศ i กับประเทศ j เสียค่าใช้จ่ายเป็น c_{ij} โดยที่ $c_{ij} > 0$ จงเขียนโปรแกรมเพื่อคำนวณค่าใช้จ่ายที่น้อยที่สุดสำหรับการจัดการแข่งขันในรอบแรก (เฉพาะรอบแรกซึ่งมีการแข่งทั้งหมด $n/2$ ครั้ง ไม่ต้องคำนวณของรอบอื่นๆ)
- กำหนดให้ n เป็นเลขคู่ และโปรแกรมรับอินพุตจากไฟล์มีรูปแบบดังตัวอย่างต่อไปนี้

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 (หน่วยเป็นล้านบาท) ในการจัดการแข่งขันสำหรับผู้สมัครคนหนึ่งกับคนอื่นที่เหลือ 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
-

การแทนความรู้โดยตรรกะเพรดิเคต



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

3.1 ตรรกะเพรดิเคต

สูตรรูปดี

ตรรกะเพรดิเคตเป็นภาษาหนึ่งซึ่งมีวากยสัมพันธ์ (syntax) และความหมาย (semantics) ของภาษาที่กำหนดขึ้นอย่างชัดเจน เราเรียกสูตรที่ถูกต้องตามวากยสัมพันธ์ของภาษานี้ว่า *สูตรรูปดี (well form formular)* ภาษานี้ประกอบด้วยสัญลักษณ์พื้นฐานดังนี้คือ

- สัญลักษณ์แสดงเพรดิเคต (predicate symbol) ใช้สายอักขระตั้งแต่ 1 ตัวขึ้นไป และขึ้นต้นด้วยตัวอักษรใหญ่ เช่น P, Q, R เป็นต้น
- สัญลักษณ์แสดงตัวแปร (variable symbol) ใช้สายอักขระตั้งแต่ 1 ตัวขึ้นไปและขึ้นต้นด้วยตัวอักษรเล็ก เช่น x, y, z เป็นต้น
- สัญลักษณ์แสดงฟังก์ชัน (function symbol) ใช้สายอักขระตั้งแต่ 1 ตัวขึ้นไปและขึ้นต้นด้วยตัวอักษรเล็ก เช่น f, g, h เป็นต้น
- สัญลักษณ์แสดงค่าคงที่ (constant symbol) ใช้สายอักขระตั้งแต่ 1 ตัวขึ้นไปและขึ้นต้นด้วยตัวอักษรใหญ่ เช่น A, B, C เป็นต้น
- เครื่องหมายวงเล็บ เช่น [], { }, () เป็นต้น

เครื่องหมายวงเล็บทั้งสามประเภทรูปนี้ไม่มีความแตกต่างกันทางความหมาย เพรดิเคตเป็นสัญลักษณ์ที่ใช้แสดงความสัมพันธ์ในโดเมนที่กล่าวถึง เช่น

สูตรอะตอม

FATHER(SOMCHAI, SOMSRI)

(3.1)

สูตร (3.1) เรียกว่า **สูตรอะตอม (atomic formula)** ซึ่งเป็นสูตรรูปดีที่เล็กที่สุดที่ต้องตามวากยสัมพันธ์ของภาษานี้ สูตรอะตอมประกอบด้วยสัญลักษณ์เพรดิเคตในที่นี้คือ 'FATHER' ซึ่งแสดงความสัมพันธ์ 'พ่อ' ความสัมพันธ์นี้จะรับ **อาร์กิวเมนต์ (argument)** 2 ตัว คือ 'SOMCHAI' และ 'SOMSRI' ซึ่งอาร์กิวเมนต์ทั้งสองนี้เป็นค่าคงที่ สูตรอะตอมที่ต้องตามวากยสัมพันธ์จะต้องประกอบด้วยเพรดิเคตและอาร์กิวเมนต์ตั้งแต่ 0 ตัวขึ้นไป โดยที่อาร์กิวเมนต์แต่ละตัวคั่นด้วยเครื่องหมาย ',' และอาร์กิวเมนต์ทั้งหมดต้องถูกคลุมด้วยเครื่องหมายวงเล็บ ลำดับของอาร์กิวเมนต์มีความสำคัญ ผู้เขียนต้องกำหนดลำดับเอง ในตัวอย่างนี้เราต้องการให้มีหมายความว่า คน 2 คนในโดเมนนี้ที่ชื่อ 'SOMCHAI' และ 'SOMSRI' มีความสัมพันธ์กันโดยที่ 'SOMCHAI' เป็นพ่อของ 'SOMSRI'

ภาษานี้สามารถใช้ตัวแปรเพื่อให้แทนถึงค่าคงที่ใดๆ ได้ และสามารถใช้ฟังก์ชันเพื่อระบุหาพจน์ (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' และมีฟังก์ชัน 'salary' ที่ระบุหาพจน์จาก 'SOMCHAI' ซึ่งพจน์นี้อาจเป็นค่าคงที่ค่าหนึ่ง เช่น 10850 เป็นต้น

การแปลความหมายของสูตรอะตอม

การแปลความหมาย (interpretation) คือการกำหนดค่าให้กับเพรดิเคต ค่าคงที่ ตัวแปร ฟังก์ชันในโดเมนนั้น ซึ่งการกำหนดค่าเหล่านี้จะเป็นตัวนิยามความหมายของสูตรอะตอม เช่นการแปลความหมายของสูตรที่ (3.3) คือการกำหนดให้ 'SOMCHAI' มีค่าคือคนที่ชื่อสมชายในโดเมนของเรา กำหนดให้ 'salary(SOMCHAI)' มีค่าเท่ากับ 10850 บาทและกำหนดให้ 'HAS-MONEY' มีค่าเป็นการที่สมชายมีเงินเท่ากับ 10850 บาท เป็นต้น เมื่อเรานิยามการแปลความหมายแล้ว เราสามารถบอกได้ว่าสูตรอะตอมตัวหนึ่งๆ มีค่าเป็น T (จริง) ถ้าความสัมพันธ์ที่ถูกแสดงด้วยสูตรนั้นเป็นจริงในโดเมนที่กล่าวถึง และจะมีค่าเป็น F (เท็จ) ถ้าไม่ใช่

ตัวเชื่อมและตัวบ่งปริมาณ

ภาษานี้มี**ตัวเชื่อม(connective)** และ**ตัวบ่งปริมาณ(quantifier)** เพื่อใช้เขียนความสัมพันธ์ได้ซับซ้อนยิ่งขึ้น ตรงกับความต้องการของเรามากขึ้น ตัวเชื่อมที่มีในภาษานี้ได้แก่

- และ แทนด้วยเครื่องหมาย \wedge
- หรือ แทนด้วยเครื่องหมาย \vee
- อิมพลี แทนด้วยเครื่องหมาย \Rightarrow
- นิเสธ แทนด้วยเครื่องหมาย \sim

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

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

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

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

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

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

เมื่อกำหนดการแปลความหมายของสูตรอะตอมแล้ว ค่าความจริงของสูตรที่ประกอบด้วยตัวเชื่อมสามารถหาได้โดยใช้ตารางค่าความจริงด้านล่างนี้ โดยที่ P และ Q แทนสูตรอะตอมหนึ่งๆ

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 ตัวบ่งปริมาณที่มีในภาษานี้ได้แก่

- **ตัวบ่งปริมาณเอกภาพ (universal quantifier)** แทนด้วยเครื่องหมาย \forall
- **ตัวบ่งปริมาณเมื่ออยู่ (existential quantifier)** แทนด้วยเครื่องหมาย \exists

ตัวอย่างเช่นต้องการเขียนประโยค '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

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

ปัญหาหนึ่งที่เกิดขึ้นในกรณีที่มีตัวบ่งปริมาณปรากฏในสูตรก็คือ เราอาจค่าความจริงของสูตรนั้นไม่ได้ เช่นกำหนดให้สูตรเป็น ' $\forall x \text{ (P}(x))$ ' และเมื่อให้การแปลความหมายของ 'P' และให้โดเมนของ x เป็นโดเมนอนันต์ เช่น x แทนเลขจำนวนจริง บางครั้งเราอาจหาค่าความจริงของสูตรนี้ไม่ได้ เนื่องจากเราไม่สามารถนำเลขจำนวนจริงมาทดสอบสูตรนี้ได้ทั้งหมด แต่ถ้าหากว่ามีจำนวนจริงตัวใดตัวหนึ่งที่ทำให้สูตรเป็นเท็จ ก็จะสรุปได้ว่าสูตรเป็นเท็จ แต่ถ้าหากว่าตรวจสอบเท่าที่จำนวนจริงที่นำมาทดสอบยังให้ค่าเป็นจริงอยู่ก็จะยังคงสรุปไม่ได้

ตรรกะเพรดิเคต
อันดับที่หนึ่ง

ตรรกะเพรดิเคตอันดับที่หนึ่ง (first-order predicate logic) คือตรรกะเพรดิเคตที่ไม่มีตัวบ่งปริมาณของสัญลักษณ์แสดงเพรดิเคตหรือของสัญลักษณ์แสดงฟังก์ชัน ในบทนี้เราจะสนใจเฉพาะตรรกะเพรดิเคตอันดับที่หนึ่งเท่านั้น ไม่สนใจตรรกะเพรดิเคตอันดับสูง (high order predicate logic) เนื่องจากตรรกะเพรดิเคตอันดับที่หนึ่งก็สามารถใช้เขียนความสัมพันธ์แทนความรู้ต่างๆ ได้อย่างกว้างขวางมาก และการคำนวณเชิงตรรกะของการแทนความรู้ประเภทนี้ก็ไม่สูงเกินไปนัก ตัวอย่างของสูตรที่เป็นตรรกะเพรดิเคตอันดับที่หนึ่งได้แก่ สูตรที่ (3.7) สูตรที่ (3.8) เป็นต้น ส่วนตัวอย่างของสูตรที่ไม่เป็นตรรกะเพรดิเคตอันดับที่หนึ่งก็เช่น $\forall x \forall y \text{ (} y(x) \Rightarrow \text{COLOR}(x, \text{GRAY}))$ เป็นต้น สังเกตว่าในกรณีนี้ 'y' เป็นตัวแปรแสดงเพรดิเคต ซึ่งในกรณีเช่นนี้การแปลความหมายของสูตรนี้จะมีความยุ่งยากซับซ้อนมากขึ้น คือเราต้องหาทุกความสัมพันธ์ 'y' เพื่อนำมาตรวจสอบหาค่าความจริงของสูตร ' $y(x) \Rightarrow \text{COLOR}(x, \text{GRAY})$ '

3.2 กฎการอนุมาน

เมื่อเราไปสังเกตปรากฏการณ์ สิ่งต่างๆ หรือความสัมพันธ์ต่างๆ ในโดเมนที่เราสนใจและนำมาเขียนให้อยู่ในรูปของตรรกะเพรดิเคตได้แล้ว เราสามารถมองได้ว่าสิ่งที่เราเขียนในรูปของตรรกะเพรดิเคตก็คือ ความรู้ที่เราทราบในโดเมนนั้นๆ ขั้นตอนต่อไปซึ่งเป็นข้อดีของการแทนความรู้ก็คือ เราสามารถอนุมาน (inference) เพื่อหาข้อเท็จจริงใหม่ๆ หรือผลสรุปที่แฝงอยู่ในความรู้นั้นได้ เราเรียกกฎที่ใช้อนุมานเพื่อหาความรู้ใหม่ว่า **กฎการอนุมาน (rule of inference)** ซึ่งจะทำหน้าที่สร้างสูตรใหม่จากสูตรหลายๆ ตัวที่มีอยู่ กฎการอนุมานมีอยู่หลากหลายชนิด ตัวอย่างของกฎการอนุมาน เช่น

กฎโมดัสโปเนนส์

และ

กฎเจาะจงตัวแปร

กฎโมดัสโปเนนส์ (Modus Ponens):

$$W1 \Rightarrow W2$$

$$\underline{W1}$$

$$W2$$

กฎเจาะจงตัวแปรเอกภาพ (Universal Specialization):

$$\underline{\forall x (W(x))}$$

$$W(A)$$

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

ตัวอย่างเช่นจากสูตร 2 ตัวคือ ' $\forall x (W1(x) \Rightarrow W2(x))$ ' และ ' $W1(A)$ ' เราสามารถอนุมานได้ว่า ' $W2(A)$ ' เป็นจริงถ้าสูตร 2 ตัวบนเป็นจริง ' $W2(A)$ ' เป็นทฤษฎี ส่วนการพิสูจน์ก็คือลำดับของกฎการอนุมานด้านล่างนี้

ใช้กฎเจาะจงตัวแปรเอกภาพ

$$\underline{\forall x (W1(x) \Rightarrow W2(x))}$$

ได้ว่า

$$W1(A) \Rightarrow W2(A)$$

จากนั้นใช้กฎโมดัสโปเนนส์

$$\underline{W1(A)}$$

ได้ว่า

$$W2(A)$$

□

ปัญหาหนึ่งของการพิสูจน์ก็ดังเช่นที่แสดงในตัวอย่างข้างต้นนี้คือ จะรู้ได้อย่างไรว่าเวลาที่ใช้กฎเจาะจงตัวแปรเอกภาพ จะต้องแทนตัวแปรด้วยค่าคงที่ตัวใด 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 ดังนั้นส่วนต่อไปที่เราจะศึกษาก็คือการแทนค่าและการทำให้เท่ากัน

3.3 การแทนค่าและการทำให้เท่ากัน

การแทนค่า (substitution) คือการแทน **พจน์ (term)** ให้กับตัวแปร พจน์หมายถึงรวมถึงค่าคงที่ ฟังก์ชัน 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 s_1 และ s_2

$$s_1 = \{z/x, w/y\} \quad (3.13)$$

$$s_2 = \{C/x, A/y\} \quad (3.14)$$

กระทำกับสูตรที่ (3.9) จะได้สูตรที่ (3.10) และสูตรที่ (3.11) ตามลำดับ

เราเขียนสูตรที่ได้จากการแทนค่า s กับสูตร E ด้วย Es จากตัวอย่างที่แล้วจะได้ว่า

$$P(z, f(w), B) = P(x, f(y), B)s_1 \quad (3.15)$$

และ

$$P(C, f(A), B) = P(x, f(y), B)s_2 \quad (3.16)$$

จุดประสงค์หนึ่งของการแทนความรู้ในรูปของตรรกะเพรดิเคตก็คือ เมื่อเราเขียนความรู้ในรูปตรรกะเพรดิเคตแล้ว ทำให้เราสามารถอนุมานหาความรู้ใหม่ๆ ที่แฝงอยู่ในความรู้ที่มีอยู่ได้ และดังเช่นที่แสดงในกฎการอนุมานว่า ปัญหาหนึ่งที่เราพบในการอนุมานหาความรู้ใหม่ก็คือ การแทนค่าให้กับตัวแปรว่าจะต้องใช้ค่าคงที่ใดแทนค่าให้กับตัวแปรใดเพื่อที่จะทำให้พจน์บางตัวเท่ากัน ซึ่งส่งผลให้การอนุมานสำเร็จ เราได้กล่าวถึงการแทนค่าไปแล้ว ส่วนต่อไปที่ก็จะกล่าวคือ *การทำให้เท่ากัน (unification)*

เรากล่าวว่า สูตร E_1 และ E_2 สามารถ*ทำให้เท่ากัน (unify)* ถ้ามีการแทนค่า s ที่ทำให้ $E_1s = E_2s$ และในกรณีนี้เราเรียก s ว่าเป็น*ตัวทำให้เท่ากัน (unifier)* ของ E_1 และ E_2 ตัวอย่างเช่น $P[x, f(y), B]$ และ $P[x, f(B), B]$ ทำให้เท่ากันได้โดยมีตัวทำให้เท่ากันคือ $s = \{A/x, B/y\}$ และผลของการทำให้เท่ากันคือ $P[A, f(B), B]$ สูตรสองตัวใดๆ มักจะมีตัวทำให้เท่ากันมากกว่าหนึ่งตัว แต่ตัวที่เราสนใจคือตัวทำให้เท่ากันที่ใช้การแทนค่าไม่มากเกินไปจนเกินไป เราเรียกตัวทำให้เท่ากันแบบนี้ว่า *ตัวทำให้เท่ากันกว้างสุด – เอ็มจียู (most general unifier – mgu)* นิยามได้ดังนี้

g เป็นตัวทำให้เท่ากันกว้างสุดของ E_1 และ E_2 ก็ต่อเมื่อ ถ้ามี s ที่เป็นตัวทำให้เท่ากันอื่นของ E_1 และ E_2 แล้ว จะต้องมีความเท่ากัน s' ที่ทำให้ $E_1s = E_1gs'$ และ $E_2s = E_2gs'$

จากตัวอย่างด้านบน เอ็มจียูของ $P[x, f(y), B]$ และ $P[x, f(B), B]$ คือ $\{B/y\}$ ซึ่งจะเห็นได้ว่าการแทนค่าไม่มากเกินไป 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 \neq NIL THEN
 - 5.3.1 Apply S to the remainder of both L1 and L2
 - 5.3.2 SUBST := SUBST \cup 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 Unify(L1,L2) ซึ่งจะได้ขั้นตอนการทำงานดังนี้

- L1 และ L2 ไม่ใช่ตัวแปรหรือค่าคงที่ ดังนั้นจึงตรวจสอบด้วยขั้นตอนที่ 2 และ 3 ในอัลกอริทึมว่า L1 กับ L2 มีเพรดิเคตตัวเดียวกัน (= P) และมีจำนวนอาร์กิวเมนต์เท่ากัน (= 3) จึงจะทำในขั้นตอนที่ 4 และ 5 ต่อไป ถ้าหากว่าสูตรสองตัวใดที่มีเพรดิเคตไม่เหมือนกันหรือมีจำนวนอาร์กิวเมนต์ไม่เท่ากัน ก็จะไม่สามารถทำให้เท่ากันได้
- ขั้นตอนที่ 4 กำหนดค่าเริ่มต้นให้กับผลลัพธ์ของการแทนค่าที่จะทำให้ L1 เท่ากับ L2 โดยเริ่มต้นให้เท่ากับเซตว่าง (แทนด้วย NIL ในอัลกอริทึม)
- ขั้นตอนที่ 5 เป็นขั้นตอนที่พยายามทำอาร์กิวเมนต์ในตำแหน่งที่ตรงกันของ L1 และ L2 ให้เท่ากัน โดยเริ่มจากอาร์กิวเมนต์ตัวที่ 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 ไปกระทำกับ $L1$ 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 ไปกระทำกับ $L1$ 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 $h(y)$ เป็นฟังก์ชันซึ่งตรวจสอบด้วยขั้นตอนที่ 2 และพบว่ามีจำนวนอาร์กิวเมนต์เท่ากันเท่ากับ 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 ไปกระทำกับ $L1$ 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

$$x \text{ กับ } f(x)$$

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

$$f(A) \text{ กับ } f(f(A))$$

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

$$f(f(A)) \text{ กับ } f(f(f(A)))$$

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

3.4 รีโซลูชัน

วิธีการอนุมานทางตรรกเพรดิเคตได้กล่าวไปแล้วสองวิธีคือกฎโมดัสโปเนนส์และกฎเจาะจงตัวแปรเอกภาพ ซึ่งกฎทั้งสองข้อนี้มีข้อจำกัดอยู่ เช่นกฎโมดัสโปเนนส์จะใช้ได้กับสูตรที่ต้องอยู่ในรูปแบบที่กำหนดเท่านั้นคือ ตัวแรกอยู่ในรูป $W1 \Rightarrow W2$ และตัวที่สองเป็น $W1$ ซึ่งเหมือนกันกับด้านซ้ายมือของสูตรแรก หรือกฎเจาะจงตัวแปรเอกภาพก็ใช้ได้กับสูตรที่มีตัวแปรแล้วแทนที่ด้วยค่าคงที่เท่านั้น การใช้งานจึงจำกัดเช่นกัน ในขณะที่ความรู้ที่เขียนอยู่ในตรรกเพรดิเคตโดยทั่วไปมีหลากหลายรูปแบบ ทำให้การใช้กฎดังกล่าวไม่กว้างขวางเพียงพอหรือมีโอกาสพบรูปแบบที่กฎใช้ได้้น้อยมาก

หัวข้อนี้จะกล่าวถึงกฎการอนุมานอีกวิธีหนึ่งซึ่งสามารถใช้ได้อย่างกว้างขวางเรียกว่า **รีโซลูชัน (resolution)** ที่สามารถใช้กับสูตรทุกตัวที่เป็น **อนุประโยค (clause)**

อนุประโยค

อนุประโยคคือสูตรที่เป็น **การหรือของสัญลักษณ์ (disjunction of literals)** สัญลักษณ์ก็คือสูตรอะตอมซึ่งอาจมีเครื่องหมายนิเสธอยู่หน้าหรือไม่ก็ได้ กล่าวคืออนุประโยคก็คือสูตรอะตอมหลายตัวมาเชื่อมกันด้วยเครื่องหมาย \vee และสูตรอะตอมบางตัวอาจมีเครื่องหมายนิเสธอยู่บางตัวอาจไม่มี ตัวอย่างของอนุประโยคก็เช่น $P(x) \vee Q(x,y) \vee \sim R(A)$ เป็นต้น ในทางปฏิบัติสูตรที่เราเขียนแทนความรู้ อาจไม่อยู่ในรูปของอนุประโยค ดังนั้นก่อนที่เราจะสามารถใช้รีโซลูชันเพื่ออนุมานได้นั้น เราจำเป็นต้องแปลงสูตรให้อยู่ในรูปของอนุประโยคก่อน โดยใช้ขั้นตอนพร้อมทั้งยกตัวอย่างการแปลงสูตรให้เป็นอนุประโยคดังต่อไปนี้

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

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

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

2. ลดขอบเขตของเครื่องหมายนิเสธ: นิเสธแต่ละตัวจะมีขอบเขตไม่เท่ากัน ถ้าพบนิเสธที่คลุมบริเวณกว้าง ก็ให้ลดขอบเขตให้แคบสุดโดยกระจายนิเสธเข้าไปข้างในบริเวณที่มันคลุมอยู่ และเปลี่ยนเครื่องหมายอื่นๆ เป็นตรงข้ามให้หมด เช่นนิเสธของ ' \wedge ' เป็น ' \vee ' นิเสธของ ' \vee ' เป็น ' \wedge ' 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. ทำตัวแปรเป็นมาตรฐาน: เปลี่ยนชื่อตัวแปรตามขอบเขตของตัวบ่งปริมาณ เช่นถ้าเราพบว่ามี y ซ้ำกันสองที่ ก็ให้เปลี่ยนชื่อตัวใดตัวหนึ่ง 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. กำจัดตัวบ่งปริมาณเมื่ออยู่: แทนค่าตัวแปรด้วยฟังก์ชันสคอเล็ม (Skolem function) ซึ่งเป็นฟังก์ชันที่แทนค่าตัวแปรหนึ่งด้วยฟังก์ชันของตัวแปรอื่นๆ ที่ตัวแปรตัวนั้นขึ้นอยู่กับมัน ในกรณีของตัวอย่างสูตรในขั้นตอนที่ 3 ที่ตำแหน่งของ $(\forall x) \{ \dots (\exists w) [Q(x, w) \wedge \sim P(w)]$ 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 เป็นฟังก์ชันสคอเล็ม

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

5. แปลงให้อยู่ในรูปแบบพรีเน็กซ์ (prenex form): ย้ายตัวบ่งปริมาณเอกภพทุกตัวมาอยู่หน้าสุด และรูปแบบที่ได้นี้เรียกว่ารูปแบบพรีเน็กซ์

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

6. จัดรูปของพรีเน็กซ์ใหม่ให้อยู่ในรูปทั่วไปแบบและ (conjunctive normal form): รูปที่สูตรทุกตัวเชื่อมกันด้วยเครื่องหมาย ' \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. ละเครื่องหมายตัวบ่งปริมาณเอกภพ: เนื่องจากว่า ณ จุดนี้ตัวแปรทุกตัวจะมีตัวบ่งปริมาณเป็นแบบเอกภพเท่านั้น
8. ที่จุดนี้เราจะได้อนุประโยคตั้งแต่ 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

$$(1) \sim P(x1) \vee \sim P(y) \vee P(f(x1, y))$$

$$(2) \sim P(x2) \vee Q(x2, g(x2))$$

$$(3) \sim P(x3) \vee \sim P(g(x3))$$

รีโซลูชันของอนุประโยคพื้นฐาน

ก่อนที่จะอธิบายถึงรีโซลูชันของอนุประโยคทั่วไป ขอกล่าวถึงรีโซลูชันของ **อนุประโยคพื้นฐาน (ground clause)** ก่อน ซึ่งจะง่ายกว่าของอนุประโยคทั่วไป อนุประโยคพื้นฐานคืออนุประโยคที่ไม่มีตัวแปร การทำรีโซลูชันสำหรับอนุประโยคพื้นฐานจะรับอินพุตเป็นอนุประโยคพ่อแม่ (parent clause) 2 ประโยค และจะให้อนุประโยคเป็นเอาต์พุต 1 ประโยค สามารถแสดงได้ดังต่อไปนี้

$$\frac{P1 \vee P2 \vee \dots \vee Pn \quad \sim P1 \vee Q2 \vee \dots \vee Qm}{P2 \vee \dots \vee Pn \vee Q2 \vee \dots \vee Qm} \quad (3.17)$$

เราเรียกอนุประโยค 2 ประโยคบนว่า **อนุประโยคพ่อแม่ (parent clause)** ส่วนอนุประโยคที่เป็นผลลัพธ์เรียกว่า **รีโซเวนต์ (resolvent)** การทำรีโซลูชันนี้ทำได้ดังแสดงใน (3.17) ด้านบนนี้ ซึ่งหมายความว่าถ้าเรามีอนุประโยค 2 ประโยค และพบว่าอนุประโยคทั้งสองมีสัจพจน์อยู่ 1 ตัวที่เหมือนกันทุกประการยกเว้นเครื่องหมายนิเสธที่ต่างกัน (สัจพจน์แรกไม่มีนิเสธแต่สัจพจน์ที่สองมีนิเสธ ($P1$ กับ $\sim P1$) หรือกลับกัน) ผลลัพธ์ที่ได้จะเป็นอนุประโยคที่นำสัจพจน์อื่นๆทั้งหมดของทั้งสองอนุประโยคมารวมกัน ยกเว้นสัจพจน์ที่เหมือนกัน (ต่างเฉพาะเครื่องหมาย) นั้น ผลที่ได้จากการทำรีโซลูชันจะเป็นจริงถ้าอนุประโยคพ่อแม่ทั้งสองเป็นจริง ตัวอย่างของการใช้รีโซลูชันแสดงใน **ตารางที่ 3-3** ด้านล่างนี้

ตารางที่ 3-3 ตัวอย่างของการทำรีโซลูชัน

อนุประโยคพ่อแม่

รีโซเวนท์

P และ $\sim P \vee Q$

Q

$P \vee Q$ ແລະ $\sim P \vee Q$

Q

$P \vee Q$ และ $\sim P \vee \sim Q$

$Q \vee \sim Q$ และ $P \vee \sim P$

$\sim P$ ແລະ P

NIL

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

ดังนั้นจะเห็นได้จากตัวอย่างทั้งห้าด้านบน วิธีโซลูชันสามารถใช้ได้อย่างกว้างขวางมาก ดังเช่นตัวอย่างแรกในตารางก็เป็นกรณีหนึ่งของวิธีโซลูชัน ซึ่งเท่ากับกฎโมดัสโปเนนส์

ร้อยละ 100

กรณีของริโซลูชันทั่วไปที่กระทำกับอนุประโยคที่มีตัวแปรด้วยนั้น ขั้นตอนจะซับซ้อนกว่าเดิม ซึ่งเราต้องให้การทำให้เท่ากันร่วมด้วยเพื่อให้อนุประโยคพ่อแม่ประกอบด้วย **สัญญาณ์เติมเต็ม (complimentary literals)** สัญญาณ์เติมเต็มคือสัญญาณ์ที่เหมือนกันทุกประการเว้นแต่ว่าตัวหนึ่งมีเครื่องหมายนิเสธส่วนอีกตัวไม่มี เช่น $\sim Q(z)$ กับ $Q(z)$ เป็นต้น ในการอธิบายการทำให้ริโซลูชันทั่วไป จะเขียนแทนอนุประโยคด้วยเซตของสัญญาณ์ เช่น $\sim P(z, f(A)) \vee \sim Q(z)$ เขียนแทนด้วย $\{\sim P(z, f(A)), \sim Q(z)\}$ เป็นต้น ขั้นตอนของริโซลูชันทั่วไปทำตามขั้นตอนต่อไปนี้

- กำหนดให้อนุประโยคพ่อแม่เป็น $\{L_i\}$ และ $\{M_i\}$
- เลือก $\{l_i\}$ และ $\{m_i\}$ ที่เป็นเซตย่อยของ $\{L_i\}$ และ $\{M_i\}$ ตามลำดับ ซึ่งมี s ที่เป็นเอ็มจีของ $\{l_i\}$ และ $\{\sim m_i\}$ (หมายความว่าเซตย่อยทั้งสองจะต้องสามารถทำให้เป็นสัจพจน์เต็มเต็มได้)
- จะได้ตัวรหัสเวกเตอร์ของอนุประโยคพ่อแม่ $\{L_i\}$ กับ $\{M_i\}$ คือ $\{\{L_i\} - \{l_i\}\}s \cup \{\{M_i\} - \{m_i\}\}s$

โดยทั่วไปไรโซเนนซ์จากกริโซลูชันทั่วไปอาจมีได้มากกว่าหนึ่งประโยชน์ขึ้นกับการเลือกเซตย่อยที่จะมาทำให้เป็นสัญญาณเติมเต็ม

ตัวอย่างของการทำรีไซเคิลขั้นทั่วไปแสดงดังด้านล่างนี้

- กำหนดให้ $\{L_i\} = \{P[x, f(A)], P[x, f(y)], Q(y)\}$ $\{M_i\} = \{\sim P[z, f(A)], \sim Q(z)\}$
(กล่าวคือต้องการทำให้สูตรชั้นระหว่างอนุประโยค $P[x, f(A)] \vee P[x, f(y)] \vee Q(y)$ 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 $s = \{z/x\}$

- ได้ว่าริโซเวนท์เท่ากับ $\{ \{P[x, f(A)], P[x, f(y)], Q(y)\} - \{ \{p(x, f(A))\} \} \{z/x\} \cup \{ \{ \sim P[z, f(A)], \sim Q(z) \} - \{ \sim P[z, f(A)] \} \} \{z/x\} = \{ P[z, f(y)], Q(y), \sim Q(z) \}$
- แต่ถ้าเราเลือก $\{l\} = \{ \{p(x, f(A)), P[x, f(y)] \} \quad \{m\} = \{ \sim P[z, f(A)] \}$ จะได้ริโซเวนท์เท่ากับ $\{Q(A), \sim Q(z)\}$

การปฏิเสธแบบปริโซลูชัน

จากที่ได้แสดงให้เห็นข้างต้นว่าการทำริโซลูชันจะทำให้เราหาความรู้ที่แฝงอยู่ในความรู้ที่มีอยู่ได้ และในหลายๆ กรณีเรามีความรู้ที่แสดงอยู่ในรูปตรรกะเพรดิเคตและเราต้องการพิสูจน์อนุประโยคตัวใหม่ใดๆ ว่าเป็นผลสรุปของความรู้ที่มีอยู่หรือไม่ วิธีการพิสูจน์ก็อาจทำได้โดยการเลือกอนุประโยคพ่อแม่ 2 ประโยคแล้วหาริโซเวนท์ ถ้าริโซเวนท์เป็นความรู้ใหม่ที่เราต้องการพิสูจน์ก็แสดงว่าเราพิสูจน์สำเร็จ ถ้าไม่ใช่เราก็อาจเลือกอนุประโยคพ่อแม่ 2 ประโยคอื่นๆ แล้วลองทำริโซลูชันดู หรืออาจนำริโซเวนท์ที่ได้ก่อนหน้านี้มาจับคู่กับอนุประโยคอื่นเพื่อเป็นอนุประโยคพ่อแม่แล้วทำริโซลูชันต่อไป อย่างไรก็ตามวิธีการพิสูจน์แบบนี้ อาจไม่มีเป้าหมายที่ชัดเจน ทำให้การพิสูจน์เสียเวลาในการคำนวณมาก เรามีวิธีการซึ่งเน้นที่เป้าหมายในการพิสูจน์เพื่อทำให้การพิสูจน์ทำได้ดีขึ้น เราเรียกวิธีการที่จะนำเสนอนี้ว่า

ความขัดแย้ง

การปฏิเสธแบบปริโซลูชัน (resolution refutation) ก่อนอื่นขอกล่าวถึง**ความขัดแย้ง (contradictory)** ที่ใช้ในการพิสูจน์โดยการปฏิเสธแบบปริโซลูชันดังนี้

สัจพจน์ 2 ตัวใดๆ จะขัดแย้งกัน ถ้าตัวหนึ่งสามารถทำให้เท่ากับนิเสธของอีกตัวหนึ่งได้ เช่น $MAN(x)$ กับ $\sim MAN(Spot)$ ขัดแย้งกัน เนื่องจาก $MAN(x)$ ทำให้เท่ากับ $MAN(Spot)$ ได้

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

การพิสูจน์ในลักษณะนี้จะมีเป้าหมายในการพิสูจน์ กล่าวคือ $\sim W$ จะเป็นตัวที่เราเลือกเพื่อไปการทำริโซลูชันกับอนุประโยคอื่น แล้วนำริโซเวนท์ที่ได้ไปกระทำริโซลูชันกับ

อนุประโยคอื่นๆ ต่อไปตามลำดับจนกระทั่งพบความขัดแย้ง ตารางด้านล่างนี้แสดง
อัลกอริทึมการปฏิเสธแบบรีโซลูชัน

ตารางที่ 3-4 อัลกอริทึมการปฏิเสธแบบรีโซลูชัน

Algorithm: Resolution Refutation

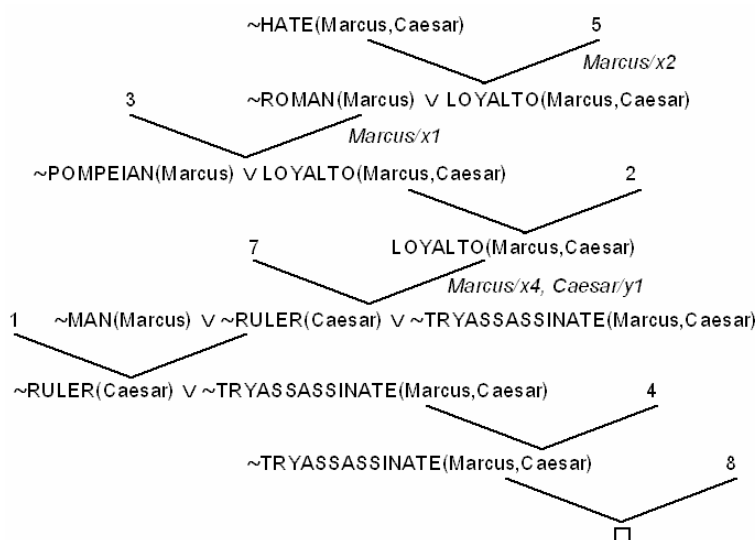
1. Convert all the statements of F to clause form.
2. Negate P and convert the result to clause form. Add it to the set of clauses obtained in 1.
3. Let $Clauses$ be the set of clauses. /* $F \cup \{\sim P\}$ */
4. UNTIL (NIL is a member of $Clauses$) or
($Clauses$ do not change) DO
 - 4.1 Select C_i 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 ทุกครั้งที่ได้รีโซเวนต์ตัวใหม่ อย่างไรก็ตามก็ตีหากพบว่าไม่มีรีโซเวนต์ใหม่เกิดขึ้นอีกหรือไม่สามารถทำรีโซลูชันได้อีก $Clauses$ จะหยุดขยายตัว ซึ่งเป็นกรณีอีกหนึ่งกรณีที่อัลกอริทึมนี้จะหยุด และในกรณีนี้ได้ว่า P ไม่ใช่ผลสรุปของ F อย่างไรก็ตามบางครั้งเราอาจพบกรณีที่ $Clauses$ ยังคงขยายตัวต่อเรื่อยๆ ซึ่งในกรณีเช่นนี้เราไม่สามารถสรุปอะไรได้ กล่าวคือ P อาจเป็นผลสรุปของ F หรือไม่ก็ได้

ตัวอย่างการทำรีโซลูชัน

- กำหนดอนุประโยคด้านล่างนี้ให้
 1. $MAN(Marcus)$
 2. $POMPEIAN(Marcus)$
 3. $\sim POMPEIAN(x1) \vee ROMAN(x1)$
 4. $RULER(Caesar)$
 5. $\sim ROMAN(x2) \vee LOYALTO(x2, Caesar) \vee HATE(x2, Caesar)$
 6. $LOYALTO(x3, f1(x3))$
 7. $\sim MAN(x4) \vee \sim RULER(y1) \vee \sim TRYASSASSINATE(x4, y1) \vee \sim LOYALTO(x4, y1)$
 8. $TRYASSASSINATE(Marcus, Caesar)$
- ต้องการพิสูจน์ $HATE(Marcus, Caesar)$

- เติม $\sim\text{HATE}(\text{Marcus}, \text{Caesar})$ เข้าไปในฐานความรู้แล้วทำรีโซลูชันดังแสดงในรูปที่ 2-14 ต่อไปนี้



รูปที่ 3-1 ตัวอย่างการทำการปฏิเสธแบบรีโซลูชัน

หมายเลขในรูปด้านบนแสดงหมายเลขอนุประโยคที่เลือกมาทำรีโซลูชัน และเครื่องหมาย 'V' แสดงการทำรีโซลูชันของอนุประโยค 2 ประโยคที่อยู่ด้านบนของเครื่องหมาย ส่วนรีโซเวนท์คืออนุประโยคด้านล่างของเครื่องหมาย ที่ด้านข้างของเครื่องหมายแสดงการแทนค่าที่ทำให้อนุประโยคพ่อแม่สามารถทำรีโซลูชันกันได้ จากตัวอย่างแสดงให้เห็นว่า $\text{HATE}(\text{Marcus}, \text{Caesar})$ เป็นผลสรุปของฐานความรู้ที่มีอยู่

เอกสารอ่านเพิ่มเติมและแบบฝึกหัด

หนังสือของ Lloyd [Lloyd, 1984] แม้ว่าจะตีพิมพ์ไว้นานแล้ว แต่ก็ยังเป็นหนังสือที่อธิบายเรื่องตรรกะเพรดิเคตไว้อย่างดีมาก นอกจากนั้นหนังสือ [Genesereth & Nilsson, 1987] ได้แสดงการใช้ตรรกะเพรดิเคตในการแก้ปัญหาต่างๆ 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. เรานิยามตัวเชื่อมตัวใหม่ในตรรกะเพรดิเคตเรียกว่า exclusive-or เขียนแทนด้วยสัญลักษณ์ \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. จงบอกว่าคู่ของสูตรในแต่ละข้อต่อไปนี้สามารถทำให้เท่ากันได้หรือไม่ ถ้าได้ให้เขียนแสดงเอ็มจียูของสูตรทั้งสอง (ตัวอักษรเล็กแสดงตัวแปรหรือฟังก์ชัน ตัวอักษรใหญ่แทนชื่อเพรดิเคตหรือค่าคงที่)

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.

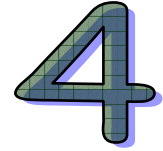
If x is a greyhound then x is a dog.

HaHa is a horse.

RaRa is a rabbit.

- (a) จงเขียนประโยคด้านบนทั้งหมดให้อยู่ในรูปของสูตรทางตรรกะเพรดิเคต
 (b) จงแปลงสูตรที่ได้ให้อยู่ในรูปของอนุประโยค
 (c) จงพิสูจน์ว่า "HaHa is faster than RaRa." โดยใช้การปฏิเสธแบบรีโซลูชัน

โปรล็อกเบื้องต้น



ภาษาโปรล็อกมาจากคำว่าโปรแกรมในตรรกศาสตร์ (PROLOG – PROgramming in LOGic) เป็นภาษาคอมพิวเตอร์ที่ใช้ในการแก้ปัญหาทางด้านสัญลักษณ์ (symbol) โดยใช้พื้นฐานของตรรกะเพรดิเคต โปรล็อกเป็นภาษาคอมพิวเตอร์ขั้นสูงที่มีจุดเด่นเหนือภาษาอื่น ๆ เกี่ยวกับการจัดการเรื่องความสัมพันธ์ของสัญลักษณ์ต่างๆ นอกจากนี้โปรล็อกยังมีลักษณะเป็นเอกลักษณ์พิเศษของตัวเองอีกคือเป็นภาษาเชิงพรรณนา (descriptive language) ซึ่งแตกต่างจากภาษาขั้นสูงทั่วไปประเภทภาษาเชิงกระบวนการคำสั่ง (procedural language) อย่างสิ้นเชิง ดังจะกล่าวต่อไปในบทนี้

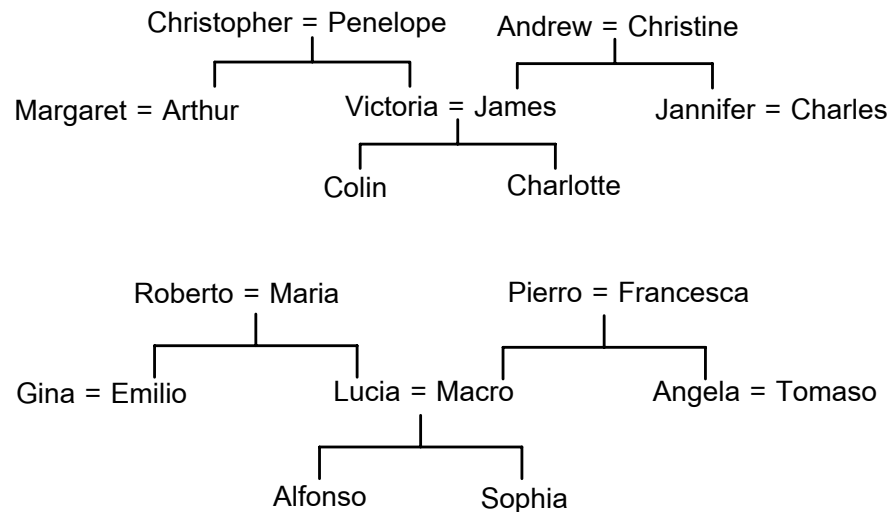
4.1 องค์ประกอบของโปรล็อก

องค์ประกอบที่สำคัญของภาษาโปรล็อกประกอบด้วย **ข้อเท็จจริง (fact)** **ข้อคำถาม (query)** **ตัวแปร (variable)** **สันธาน (conjunction)** และ **กฎ (rule)** ใช้อธิบายวัตถุ (object) และความสัมพันธ์ (relation) ระหว่างวัตถุ เราใช้โปรล็อกในการจัดการกับตรรกะเพรดิเคตที่กล่าวไปในบทที่ 3 โปรล็อกเหมาะสำหรับเขียนโปรแกรมที่อธิบายความสัมพันธ์และหาข้อสรุปเกี่ยวกับความสัมพันธ์นั้น แต่ไม่เหมาะกับการนำมาคำนวณทางคณิตศาสตร์ โปรล็อกประกอบด้วยสามส่วนหลักคือ

ข้อเท็จจริง
กฎ
และ
ข้อคำถาม

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

พิจารณาแผนภาพต้นไม้ครอบครัวในรูปที่ 4-1



รูปที่ 4-1 ต้นไม้ครอบครัว

เครื่องหมาย 'A = B' ในรูปหมายถึง A แต่งงานกับ B และต้นไม้แสดงความสัมพันธ์ภายในครอบครัว 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 แล้วให้โปรแกรมเมอร์คำตอบให้ สมมติว่าเราต้องการเขียนความสัมพันธ์ภายในครอบครัวด้วยภาษาโปรแกรมเมอร์จะสามารถเขียนได้ดังจะกล่าวต่อไป

ข้อเท็จจริง

จากตัวอย่างในรูปที่ 4-1 สมมติว่าเราเลือกความสัมพันธ์พ่อแม่และความสัมพันธ์เพศมาเป็นความสัมพันธ์หลัก 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, jannifer).
07: parent(christine, james).
08: parent(christine, jannifer).
09: parent(victoria, colin).
10: parent(victoria, chlotte).
11: parent(james, colin).
12: parent(james, chlotte).
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(jannifer).
42: female(charlotte).
43: female(maria).
44: female(francesca).
45: female(gina).
46: female(lucia).
47: female(angela).
48: female(sophia).
```

ตัวเลข01...48:ในตารางที่ 4-1 แสดงหมายเลขบรรทัดและไม่ต้องเขียนลงในโปรแกรม แต่ละบรรทัดในตารางคือข้อเท็จจริง 1 ข้อ เช่น 'parent(christopher,arthur).' โดยมี 'parent' เป็นชื่อ*เพรดิเคต(predicate)* และมี 'christopher' กับ 'arthur' เป็น*อาร์กิวเมนต์(argument)* ภายในเครื่องหมายวงเล็บ ข้อเท็จจริงทุกข้อต้องจบด้วยเครื่องหมายมหัพภาค '.' ในตารางมี

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

ค่าคงที่

ค่าคงที่ (*constant*) ในโปรล็อกมี 2 ประเภทหลักๆ คือ

อะตอม

และ

ตัวเลข

- *อะตอม (atom)* เช่น christopher, arthur
- *ตัวเลข (number)* เช่น 123, 99.99

ข้อคำถาม

เมื่อเราอยู่ใน*ตัวแปลภาษาโปรล็อก (Prolog interpreter)* จะเห็นสัญลักษณ์ '?' ซึ่งแสดงให้ป้อนข้อคำถาม เราสามารถป้อนข้อคำถามได้ เช่นถ้าต้องการรู้ว่า 'christopher' เป็นพ่อแม่ของ 'arthur' หรือไม่ก็ป้อนดังนี้ (โดยเราต้องบรรจุ (load) ข้อเท็จจริงที่เขียนไว้ใน*ตารางที่ 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

เมื่อเราตั้งข้อความถาม 'parent(christopher,X)' เพื่อให้โปรล็อกหาคำตอบสำหรับตัวแปร X โปรล็อกจะค้นหาข้อเท็จจริงที่จับคู่ได้กับ X ในฐานความรู้ (โปรแกรม) และจะให้คำตอบเป็น 'arthur' ถ้าเราต้องการหาคำตอบอื่น เราสามารถถามต่อได้ว่า มีคำตอบอื่นอีกหรือไม่ โดยใช้เครื่องหมายอัมภาค ';' ดังแสดงในตัวอย่างด้านบน ซึ่งโปรล็อกจะให้คำตอบที่สองคือ 'victoria' และถ้าถามต่อด้วย ';' ซึ่งจะไม่พบค่าคงที่ตัวอื่นอีกที่เมื่อแทนใน X แล้วทำให้ 'parent(christopher,X)' เป็นจริง ดังนั้นโปรล็อกจะตอบ 'No'

พจน์และการแทนค่า

พจน์ (term) คือสิ่งต่อไปนี้

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

การแทนค่า (substitution) คือเซตของคู่ลำดับ $\{X = t\}$ 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) เขียนแทนด้วย ';' เป็นเครื่องหมายแสดง 'และ' ทางตรรกศาสตร์ ตัวอย่างเช่น

?- parent(penelope,X), parent(X,Y).

คือข้อความถามให้หา X และ Y ซึ่ง X มีพ่อแม่เป็น penelope และ X เป็นพ่อแม่ของ Y ข้อความนี้เป็น 1 ข้อความแต่มี **เป้าหมาย (goal)** 2 ตัวคือ 'parent(penelope,X)' และ 'parent(X,Y)'

กฎ

กฎ (rule) หรือที่เราเรียกว่า **อนุประโยค (clause)** ในตรรกะเพรดิคัตนั้น ในโปรล็อกจะเขียนอยู่ในรูปด้านล่างนี้

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

โดยที่ ‘:-’ แทนคำว่า ‘ถ้า’ ส่วนเครื่องหมาย ‘,’ ที่อยู่ระหว่าง **สัจพจน์** B_i ก็คือตัวเชื่อม ‘และ’ ดังที่กล่าวแล้วข้างต้น ดังนั้นกฎนี้อ่านว่า ถ้า B_1 และ B_2 และจนกระทั่งถึง B_n ทุกตัวเป็นจริงแล้ว สัจพจน์ H ก็จะเป็นจริงด้วย

กฎหรืออนุประโยคในภาษาโปรแกรมมิ่งจะมีสัจพจน์อยู่ตัวเดียวที่ไม่มีนิเสธคือ H กฎด้านบนสามารถเขียนอยู่ในรูปอนุประโยคในตรรกะเพรดิเคตได้เป็น

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

อนุประโยค
ของฮอร์น

เราเรียกอนุประโยคที่มีสัจพจน์ตัวเดียวที่ไม่มีนิเสธว่า **อนุประโยคของฮอร์น (Horn clause)** ดังนั้นกฎทุกกฎในโปรแกรมมิ่งจะเป็นอนุประโยคของฮอร์น ซึ่งต้องมีสัจพจน์หนึ่งตัวที่ไม่มีนิเสธ ส่วนสัจพจน์อื่นๆ ที่เหลือทั้งหมด (ตั้งแต่ 0 ตัวเป็นต้นไป – กรณีที่เป็น 0 ตัวคือข้อเท็จจริง) ต้องมีนิเสธ และเราเรียก H ว่าเป็น **ส่วนหัว (head)** ของกฎและเรียก ‘ B_1, B_2, \dots, B_n ’ รวมกันว่าเป็น **ลำตัว (body)** ของกฎ ตัวอย่างของกฎเช่น

$\text{grandparent}(X, Y) \text{ :- } \text{parent}(X, Z), \text{parent}(Z, Y).$

มีส่วนหัวคือ ‘ $\text{grandparent}(X, Y)$ ’ และส่วนลำตัวคือ ‘ $\text{parent}(X, Z), \text{parent}(Z, Y)$ ’ กฎนี้อ่านว่า ถ้า ‘ $\text{parent}(X, Z), \text{parent}(Z, Y)$ ’ เป็นจริงแล้ว ‘ $\text{grandparent}(X, Y)$ ’ เป็นจริงด้วย

ความหมายของกฎ ‘ $P \text{ :- } Q, R.$ ’

เราสามารถมองกฎที่เขียนในโปรแกรมมิ่งได้สองลักษณะคือ

1. ในลักษณะของความหมายเชิงประกาศ (declarative meaning): แปลความหมายของกฎในลักษณะของความจริงทางตรรกศาสตร์ ตัวอย่างเช่น
 - ‘ $P \text{ :- } Q, R.$ ’ คือ P จะเป็นจริงถ้า Q และ R เป็นจริง
 - ‘ $\text{grandparent}(X, Y) \text{ :- } \text{parent}(X, Z), \text{parent}(Z, Y).$ ’ คือ ‘ $\text{grandfather}(X, Y)$ ’ เป็นจริงถ้า ‘ $\text{parent}(X, Z)$ ’ และ ‘ $\text{parent}(Z, Y)$ ’ เป็นจริง หรือสำหรับ X, Y และ Z ทุกตัว X เป็นปู่ย่าของ Y ถ้า X เป็นพ่อแม่ของ Z และ Z เป็นพ่อแม่ของ Y
2. ในลักษณะของความหมายเชิงกระบวนการคำสั่ง (procedural meaning) มองในลักษณะเชิงโปรแกรมที่สามารถทำงานได้ ตัวอย่างเช่น
 - ‘ $P \text{ :- } Q, R.$ ’ คือ ถ้าจะแก้ปัญหา P จะต้องแก้ปัญหาย่อย Q ตามด้วยปัญหาย่อย R
 - ‘ $\text{grandparent}(X, Y) \text{ :- } \text{parent}(X, Z), \text{parent}(Z, Y).$ ’ คือ ในการตอบข้อคำถามว่า X เป็นปู่ย่าของ Y หรือไม่ ให้ตอบข้อคำถามว่า X เป็นพ่อแม่ของ Z หรือไม่ และ Z เป็นพ่อแม่ของ Y หรือไม่

การจับคู่และการย่อหรรอย

การเท่ากันและการจับคู่ (equality and matching)

พิจารณาข้อความถามในโปรล็อก

?- $X = Y$.

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 การจับคู่ในโปรล็อกก็คือ **การทำให้เท่ากัน (unify)** ในตรรกะเพรดิเคต

กฎสำหรับจับคู่ X กับ Y

กฎในการจับคู่ X กับ Y มีดังนี้

- ถ้า X เป็นตัวแปรที่ยังไม่ได้แทนค่าและ Y ถูกแทนค่าด้วยพจน์ใดๆ แล้ว X กับ Y เท่ากันและ Y จะถูกแทนค่าด้วยพจน์นั้นด้วย
- ตัวเลขและอะตอมเท่ากับตัวมันเองเท่านั้น
- พจน์ประกอบ 2 ตัวใดๆ จะเท่ากันถ้าทั้งคู่มีฟังก์เตอร์เดียวกัน มีจำนวนอาร์กิวเมนต์เท่ากันและอาร์กิวเมนต์ในตำแหน่งที่ตรงกันเท่ากัน

ตัวอย่างเช่น

?- triangle(point(1,1),A,point(2,3)) = triangle(X,point(4,Y),point(2,Z)).

จับคู่กันได้และเท่ากันด้วยการแทนค่า $\{X = \text{point}(1,1), A = \text{point}(4,Y), Z = 3\}$

การย่อหรรอย

โปรล็อกมีกระบวนการที่เรียกว่า **การย่อหรรอย (backtracking)** เพื่อช่วยในการค้นหาคำตอบเปรียบเสมือนการค้นหาแนวลึกก่อนในการค้นหาของปริภูมิสถานะ พิจารณากฎด้านล่างนี้

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

กฎข้อนี้กล่าวว่า A จะเป็นจริง ถ้า B_1 และ B_2 จนกระทั่งถึง B_n เป็นจริง สัญพจน์ A, B_1, B_2, \dots, B_n อาจประกอบด้วยตัวแปร ถ้าเราต้องการพิสูจน์ว่า A เป็นจริงหรือไม่ เราก็จะเริ่มพิสูจน์ 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 โปรล็อกจะพยายามทำการพิสูจน์ต่อโดยตัดการแทนค่าของตัวแปรที่ผ่านมาสำหรับตัวแปรที่ปรากฏใน B_i แล้วย้อนกลับไปยัง B_{i-1} เพื่อหาเส้นทางพิสูจน์ใหม่ ซึ่งหมายถึงการแทนค่าใหม่ การย้อน

รอยสามารถกลับไปจนถึง B_1 และถ้ามีการแทนค่าบางตัวที่ทำให้ B_1 ถึง B_i เป็นจริงก็จะพิสูจน์ต่อไปได้ ตัวอย่างเช่นถ้าเราต้องการหาคนที่เป็ลูกของ christopher และเป็นพ่อแม่ของ colin เราใช้ข้อความดังนี้

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

โปรล็อกจะพิสูจน์เป้าหมายตัวแรกในข้อความถามด้านบน โดยพยายามหาการแทนค่าของ X ที่ทำให้ 'parent(christopher,X)' เป็นจริง และพบว่าข้อเท็จจริงตัวที่ 1 ในตารางที่ 4-1 'parent(christopher,arthur)' ด้วยการแทนค่า $\{X = arthur\}$ ทำให้เป้าหมายแรกเป็นจริงได้ จึงส่งการแทนค่านี้ไปยังเป้าหมายถัดไป ในการพิสูจน์เป้าหมายตัวที่สอง 'parent(X,colin)' เนื่องจาก X ถูกแทนค่าด้วย arthur ดังนั้นโปรล็อกจะค้นหามี 'parent(arthur,colin)' ในฐานความรู้หรือไม่ ซึ่งพบว่าไม่มี ทำให้เป้าหมายตัวนี้ไม่เป็นจริง ณ จุดนี้จะเกิดการย้อนรอยกลับไปยังเป้าหมายตัวแรกพร้อมกับตัดการแทนค่า $\{X = arthur\}$ ออก เพื่อหาการแทนค่าอื่นที่ทำให้เป้าหมายตัวแรกเป็นจริงและพบว่า $\{X = victoria\}$ ทำให้เป้าหมายตัวแรกเป็นจริงได้ พร้อมกับส่งการแทนค่าใหม่นี้เพื่อไปพิสูจน์ต่อสำหรับเป้าหมายตัวที่สอง ซึ่งครั้งนี้จะเป็น 'parent(victoria,colin)' และพบว่าเป็จริงด้วยข้อเท็จจริงข้อที่ 9 ในตารางที่ 4-1 จึงได้การแทนค่า $\{X = victoria\}$ ทำให้ข้อความถามเป็นจริง

กระบวนการ
ทำงานของ
โปรล็อก

กระบวนการทำงานของโปรล็อกเพื่อตอบข้อความแสดงในตารางที่ 4-2

ตารางที่ 4-2 กระบวนการทำงานของโปรล็อก

<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 $H\theta = G1$. Replace G1 in the goal list with $B1\theta,B2\theta,...,Bm\theta$, obtaining the new goal list $B1\theta,...,Bm\theta,G2\theta,...,Gn\theta$ 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.
--

ตัวอย่างปัญหาลิงกินกล้วย

ลิงตัวหนึ่งอยู่ที่ประตูในห้อง กลางห้องมีกล้วยแขวนอยู่ที่เพดาน ลิงหิวมากและอยากไปหยิบกล้วยแต่มันสูงไม่พอที่จะเอื้อมถึง ที่หน้าต่างในห้องมีกล่องๆ หนึ่ง ลิงสามารถเดินบนพื้น ปีนกล่อง ผลักกล่องไปมารอบห้อง และหยิบกล้วยถ้ากล่องอยู่ใต้กล้วย คำถามคือลิงหยิบกล้วยได้หรือไม่

ในการเขียนโปรแกรมโปรล็อกนั้นเราต้องเลือกเพรดิเคตที่จะแทนความสัมพันธ์ในปัญหาที่เราสนใจ ในที่นี้จะใช้เพรดิเคต 'state' เพื่อแสดงสถานะต่างๆ ที่แสดงข้อมูล 4 ตัวคือ (1) ตำแหน่งที่ลิงอยู่ (2) ลิงยืนบนพื้นหรือกล่อง (3) ตำแหน่งที่กล่องตั้งอยู่และ (4) ลิงมีกล้วยหรือไม่ เราแทนสถานะเริ่มต้นได้ดังนี้

```
state(atdoor,onfloor,atwindow,hasnot)
```

และสถานะสุดท้ายคือ

```
state(.,.,.,has)
```

โดยที่ '.' คือ *ตัวแปรไม่สนใจ (don't care variable)* หมายถึงตัวแปรที่เราไม่สนใจค่าที่ได้หลังพบคำตอบ ดังนั้น state(.,.,.,has) เท่ากับ state(X,Y,Z,has) แต่กรณีแรกเราไม่สนใจการแทนค่าที่ได้

เราจะใช้แนวคิดของการค้นหาในปริภูมิสถานะเพื่อหาคำตอบของปัญหานี้ โดยพิจารณาว่าเรามีสถานะเริ่มต้น มีสถานะสุดท้าย และถ้าเรามีตัวกระทำที่ใช้เปลี่ยนสถานะ เราก็จะค้นหาคำตอบได้เพราะว่าโปรล็อกจะใช้กระบวนการย้อนรอยเพื่อค้นหาเส้นทางต่างๆ ในลักษณะของการค้นหาแนวลึก ดังนั้น ณ จุดนี้เราให้เพรดิเคต 'move' แสดงการเปลี่ยนจากสถานะหนึ่งไปอีกสถานะหนึ่ง เพรดิเคตนี้มีรูปแบบดังนี้

```
move(State1,MoveOp,State2)
```

โดยที่ MoveOp เป็นตัวกระทำและอาจเป็น (1) หยิบกล้วย – grasp (2) ปีนกล่อง – climb (3) ผลักกล่อง – push (4) เดินไปมา – walk ตัวอย่างเช่น

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

แสดงการเปลี่ยนสถานะโดยตัวกระทำตัวที่หนึ่ง grasp เพื่อเปลี่ยนสถานะจากสถานะที่ 'ลิงอยู่กลางห้อง ลิงอยู่บนกล่อง กล่องอยู่กลางห้อง ลิงไม่มีกล้วย' ไปยังสถานะที่ 'ลิงอยู่กลางห้อง ลิงอยู่บนกล่อง กล่องอยู่กลางห้อง ลิงมีกล้วย'

เมื่อเราเขียนเพรดิเคต 'move' สำหรับตัวกระทำที่เหลือเรียบร้อยแล้ว (ดูใน [ตารางที่ 4-3](#)) เพรดิเคตตัวสุดท้ายที่เราต้องนิยามก็คือ 'canget' เพื่อใช้สำหรับพิสูจน์ว่าที่สถานะหนึ่งๆ ลิงจะหยิบกล้วยได้หรือไม่

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

เพรดิเคตนี้ประกอบด้วยกฎสองข้อ ข้อแรกเป็นกรณีของสถานะสุดท้ายคือกรณีที่ลิงมีกล้วยแล้วแสดงว่าลิงหยิบกล้วยได้ ส่วนกฎข้อที่สองเป็นกรณีของสถานะอื่นๆ (S1) ที่ยังไม่มีกล้วย ซึ่งนิยามว่าที่สถานะ S1 ลิงหยิบกล้วยได้ถ้ามีตัวกระทำ M บางตัวที่เปลี่ยนสถานะจาก S1 ไปเป็น S2 และพบว่าที่ S2 ลิงหยิบกล้วยได้ กฎข้อที่สองนี้มีนิยามแบบเรียกซ้ำ (ดูรายละเอียดของโปรแกรมเรียกซ้ำใน [หัวข้อที่ 4.2](#))

เรานำเพรดิเคตทั้งหมดเขียนเป็นโปรแกรมได้ใน [ตารางที่ 4-3](#) ด้านล่างนี้

ตารางที่ 4-3 โปรแกรมลิงกินกล้วย

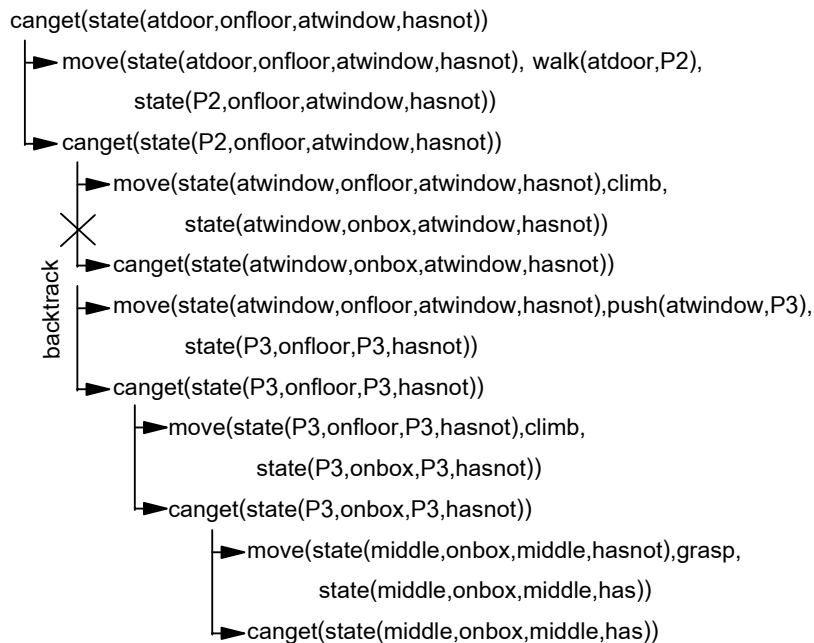
```
01: canget(state(._._,has)).
02: canget(S1) :- move(S1,M,S2), canget(S2).
03: move(state(middle,onbox,middle,hasnot), grasp,
04:      state(middle,onbox,middle,has)).
05: move(state(P,onfloor,P,H), climb,
06:      state(P,onbox,P,H)).
07: move(state(P1,onfloor,P1,H), push(P1,P2),
08:      state(P2,onfloor,P2,H)).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

เมื่อบรรจุโปรแกรมนี้เข้าไปในตัวแปลภาษาโปรล็อก เราสามารถตั้งคำถามที่เราต้องการหาคำตอบได้ดังนี้

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

Yes

ซึ่งแสดงให้เห็นว่าจากสถานะเริ่มต้นลิงจะหยิบกล้วยได้ ด้านล่างนี้แสดง *การตามรอย (trace)* การทำงานของโปรล็อก



รูปที่ 4-2 ตามรอยการทำงานของโปรล็อกสำหรับปัญหาลิงกีนกล้วย

4.2 การโปรแกรมแบบเรียกซ้ำ

การโปรแกรมแบบเรียกซ้ำ (recursive programming) เป็นเครื่องมือที่สำคัญของการใช้โปรล็อกเนื่องจากในภาษาโปรล็อกไม่มีคำสั่งวนซ้ำ (loop) เหมือนในภาษาอื่นๆ หลายภาษาดังนั้นการโปรแกรมเรียกซ้ำจึงเป็นเครื่องมือหลักอีกตัวหนึ่งในการเขียนโปรแกรมโปรล็อก เช่นถ้าเราต้องการหาว่าใครเป็นบรรพบุรุษของใคร เราอาจเขียนโปรแกรมโปรล็อกที่ไม่เป็นโปรแกรมแบบเรียกซ้ำได้ดังนี้

```

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).
...
```

จะเห็นได้ว่าโปรแกรมด้านบนนี้ไม่มีประสิทธิภาพเพราะต้องเขียนกฎจำนวนมาก และก็ไม่สามารถเขียนได้มากพอที่จะครอบคลุมทุกกรณี แต่ถ้าใช้โปรแกรมแบบเรียกซ้ำก็จะนิยามความสัมพันธ์นี้ได้ครอบคลุมทั้งหมดตั้งด้านล่างนี้

$\text{predessor}(X,Y) :- \text{parent}(X,Y).$

$\text{predessor}(X,Y) :- \text{parent}(X,Z), \text{predessor}(Z,Y).$

เมื่อพิจารณาโปรแกรม 'predessor' ด้านบน เราจะเห็นลักษณะการเขียนโปรแกรมแบบเรียกซ้ำในโปรล็อกที่จะประกอบด้วย 2 ส่วนหลักคือ

- **อนุประโยคฐาน (base clause)** คือกฎที่ไม่มีการเรียกตัวมันเองและมีตั้งแต่ 1 ข้อขึ้นไป อนุประโยคฐานนี้ไว้สำหรับการหยุดการเรียกซ้ำ
- **อนุประโยคเรียกซ้ำ (recursive clause)** คือกฎที่มีการเรียกตัวของมันเอง โดยจะมีเพรดิเคตในส่วนลำตัวของกฎบางเพรดิเคตที่เหมือนกับเพรดิเคตในส่วนหัวโปรแกรมของเพรดิเคตหนึ่งๆ มีกฎเรียกซ้ำแบบนี้ตั้งแต่ 1 ข้อขึ้นไป ในอนุประโยคเรียกซ้ำนี้เพรดิเคตที่เรียกซ้ำต้องมีอาร์กิวเมนต์ที่แตกต่างกับเพรดิเคตในส่วนหัวไม่เช่นนั้นการเรียกซ้ำจะไม่จบ

เพื่อให้เข้าใจถึงการเขียนโปรแกรมเรียกซ้ำในโปรล็อก ในส่วนต่อไปนี้จะยกตัวอย่างการเขียนโปรแกรมเรียกซ้ำที่ใช้จัดการกับตัวเลขและรายการ (list) ตามลำดับ

4.2.1 โปรแกรมเรียกซ้ำกับตัวเลข

โปรแกรมจำนวนธรรมชาติ

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

$0, s(0), s(s(0)), s(s(s(0))), s(s(s(s(0)))) , \dots$

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

ในการเขียนโปรแกรมเรียกซ้ำโดยทั่วไป สิ่งที่เราต้องพิจารณาก็คือ อนุประโยคฐานเขียนได้อย่างไรและอนุประโยคเรียกซ้ำเขียนได้อย่างไร ซึ่งมีข้อสังเกตว่าอนุประโยคเรียกซ้ำมักจะทำหน้าที่เพิ่มหรือลดลำดับของข้อมูลในอาร์กิวเมนต์ของเพรดิเคตที่เรียกซ้ำและอนุประโยคฐานมักเป็นตัวที่มีลำดับน้อยสุดหรือมากที่สุดของข้อมูล 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 และพบว่าในอนุประโยคเรียกซ้ำข้อมูลถูกลดลำดับลงหนึ่งหน่วยจาก s(X) เป็น X เช่นถ้าเราตั้งข้อคำถามว่า '?- natural_number(s(s(s(0))))' (3 เป็นจำนวนธรรมชาติหรือไม่?) ข้อคำถามนี้จับคู่ได้กับส่วนหัวของกฎข้อ 2 โดยพยายามทำให้อาร์กิวเมนต์ที่ส่วนหัวของกฎคือ 's(X)' เท่ากับ 's(s(s(0)))' ได้การแทนค่า {X = s(s(0))} จากนั้นโปรล็อกจะพิสูจน์ต่อว่าส่วนลำตัวเป็นจริงหรือไม่ นั่นคือ natural_number(s(s(0))) จะเห็นได้ว่าการเรียกซ้ำเกิดขึ้นแต่ละครั้ง ลำดับของข้อมูลจะน้อยลงหนึ่งหน่วย ดังนั้นเมื่อเรียกซ้ำหลายรอบจะถึงจุดที่อาร์กิวเมนต์เป็น '0' ซึ่งจะหยุดได้ตัวอนุประโยคฐาน

โปรแกรมบวกจำนวนธรรมชาติ

โปรแกรมต่อไปคือโปรแกรมบวกจำนวนธรรมชาติ ให้โปรแกรมบวกเลขใช้เพรดิเคตชื่อ 'plus' มีรูปแบบคือ 'plus(X,Y,Z)' มีความหมายว่า Z เป็นผลบวกของ X กับ Y เช่นเมื่อตั้งข้อคำถาม '?-plus(s(s(0)),s(s(0))),Z)' แล้วต้องได้ Z = s(s(s(s(0)))) เป็นต้น โปรแกรมที่ได้เป็นดังตารางที่ 4-5 นี้

ตารางที่ 4-5 โปรแกรมบวกเลข

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

สมมติว่าอาร์กิวเมนต์สองตัวแรกเป็นอินพุตและตัวที่สามเป็นเอาต์พุต อย่างไรก็ตามโดยทั่วไปแล้วโปรแกรมโปรล็อกส่วนมากดังเช่นโปรแกรมนี้ อาร์กิวเมนต์แต่ละตัวเป็นได้ทั้งอินพุตและเอาต์พุต ในโปรแกรมนี้จะเห็นว่ากฎข้อแรกซึ่งเป็นอนุประโยคฐานมีอาร์กิวเมนต์ตัวแรกเป็นข้อมูลที่มีลำดับต่ำสุดคือ '0' และเมื่อสังเกตอาร์กิวเมนต์ตัวเดียวกัน (ตัวแรก) ในกฎข้อที่สองจะเห็นได้ว่า อาร์กิวเมนต์ตัวแรกที่ส่วนหัวมีลำดับมากกว่าอาร์กิวเมนต์ตัวแรกในส่วนลำตัว ซึ่งก็คือมีการลดลำดับลงทีละหนึ่ง ซึ่งโปรแกรมเรียกซ้ำจะมีลักษณะเช่นนี้คืออนุประโยคเรียกซ้ำจะถูกเรียกวนซ้ำ จนกระทั่งถึงจุดหนึ่งที่อาร์กิวเมนต์ตรงกับอนุประโยคฐานก็จะหยุด

กฎข้อแรกหมายความว่า '0' บวกกับตัวเลขใดๆ จะได้ตัวเลขนั้น ส่วนกฎข้อที่สองหมายความว่า ถ้า X บวก Y ได้ Z (plus(X,Y,Z)) แล้ว X+1 (s(X) ก็คือ X+1) บวก Y ต้องได้ Z+1

การเขียนโปรแกรมเรียกซ้ำนี้สามารถเขียนได้โดยเรากำหนดอนุประโยคฐานก่อน และต้องคำนึงว่าเราจะลดลำดับของอาร์กิวเมนต์ตัวใด สมมติว่าในกรณีนี้จะลดลำดับที่อาร์กิวเมนต์ตัวแรก (หรือเราจะลดที่อาร์กิวเมนต์ตัวที่สองก็ได้) ดังนั้นเราจะได้ในขั้นตอนแรกว่ากฎข้อแรกควรเป็น 'plus(0,X,?)' แล้วเราก็มาคำค่าของ ? ว่าควรเป็นอะไร ซึ่งจากคณิตศาสตร์ง่าย ๆ เราจะได้ว่า ? ก็คือ X (เพราะ 0 บวกตัวเลขใดก็ได้ตัวเลขนั้น) ดังนั้นกฎข้อแรกจึงเป็น 'plus(0,X,X) :- natural_number(X).' ส่วนลำดับของกฎนี้เพิ่มเข้าไปเพื่อตรวจสอบว่า X ต้องเป็นจำนวนธรรมชาติเท่านั้น กล่าวคือเราจะไม่ใช่โปรแกรมนี้สำหรับบวกเลขที่อยู่ในรูปจำนวนเต็มเช่น '?- plus(5,2,Z)' เป็นต้น

จากนั้นเราจึงมาเขียนอนุประโยคเรียกซ้ำ ดังที่กล่าวแล้วว่าเราต้องการลดลำดับของอาร์กิวเมนต์ตัวแรก ดังนั้นส่วนหัวของกฎจึงเป็น 'plus(s(X),Y,?)' ซึ่งเราต้องหาเอาต์พุต ? ต่อไปว่าควรเป็นอะไร ณ จุดนี้เรารู้ว่ากำลังเขียนโปรแกรมเรียกซ้ำ ดังนั้นส่วนลำดับของกฎนี้จึงต้องมีเพรดิเคต plus โดยที่มีการลดลำดับของอาร์กิวเมนต์ตัวแรก เราจึงได้กฎเป็น 'plus(s(X),Y,?) :- plus(X,Y,??).' ให้เรากำหนดค่าของ ?? เป็นตัวแปรใดๆ ได้เลย เช่นให้เป็น Z จะได้ 'plus(s(X),Y,?) :- plus(X,Y,Z).' แล้วสมมติว่าถ้า Z ที่เป็นค่าเอาต์พุตเป็นจริงแล้ว ? ควรเป็นเท่าไร ซึ่งเราจะได้ว่าถ้า X บวก Y ได้ Z (plus(X,Y,Z)) แล้วแน่นอนว่าจากการคำนวณทางคณิตศาสตร์ s(X) บวก Y ก็ต้องได้ s(Z) ทำให้เราได้กฎข้อที่สองดังในตารางที่ 4-5

โปรแกรมคูณจำนวนธรรมชาติ

โปรแกรมสุดท้ายในส่วนของเขียนโปรแกรมเรียกซ้ำกับตัวเลขคือโปรแกรมคูณจำนวนธรรมชาติ 'times(X,Y,Z)' มีความหมายว่า X คูณ Y เท่ากับ Z โปรแกรมเป็นดังตารางที่ 4-6 นี้

ตารางที่ 4-6 โปรแกรมคูณจำนวนธรรมชาติ

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

อนุประโยคฐานในกฎข้อแรกของโปรแกรมนี้มีอาร์กิวเมนต์ตัวแรกเป็น '0' และอนุประโยคเรียกซ้ำมีอาร์กิวเมนต์ตัวแรกที่เมื่อเรียกซ้ำแล้วลำดับลดลงทีละหนึ่ง ซึ่งเมื่อการเรียกซ้ำดำเนินไปจนกระทั่งอาร์กิวเมนต์แรกเป็น '0' ก็จะหยุดได้ที่กฎข้อหนึ่ง ความหมายของโปรแกรมนี้คือ 0 คูณตัวเลขใดก็ได้ 0 ตามกฎข้อแรก ส่วนกฎข้อที่สองหมายความว่าถ้า X คูณ Y ได้ W และ W บวก Y ได้ Z แล้ว s(X) คูณ Y จะได้ Z

วิธีการเขียนโปรแกรมนี้คล้ายกับโปรแกรมบวกเลข โดยเรากำหนดว่าจะลดลำดับที่อาร์กิวเมนต์ตัวแรก ดังนั้นเราได้ 'times(0,X,?)' และใช้ความรู้ทางคณิตศาสตร์ทำให้รู้ว่า

? เท่ากับ 0 (เพราะ 0 คูณอะไรก็ได้ 0) ส่วนอนุประโยคเรียกซ้ำนั้น เราเริ่มเขียนจากส่วนหัวได้ 'times(s(X),Y,?)' และเพิ่มเพรดิเคต 'times' เข้าที่ส่วนลำตัวได้เป็น 'times(s(X),Y,?) :- times(X,Y,??)' กำหนดให้ ?? เขียนแทนด้วย W แล้วคำนวณว่า ? เป็นเท่าไรของ W ณ จุดนี้เราได้ 'times(s(X),Y,?) :- times(X,Y,W)' หา ? ว่าเท่ากับเท่าไร ตรงนี้เราคำนวณคณิตศาสตร์กันเล็กน้อย

ถ้า $X*Y = W$ แล้วจะได้ว่า $s(X)*Y = (X+1)*Y = X*Y+Y = W+Y$ ดังนั้นแสดงว่า ? เท่ากับ $W+Y$ และเนื่องจากเรามีโปรแกรมบวกเลขแล้วเราจึงนำเพรดิเคตบวกเลข 'plus' มาต่อเข้าที่ส่วนลำตัวของกฎ แล้วเขียน ? แทนด้วย Z เราจะได้ 'times(s(X),Y,Z) :- times(X,Y,W), plus(W,Y,Z).' เป็นโปรแกรมในตารางที่ 4-6

4.2.2 โปรแกรมเรียกซ้ำกับรายการ

รายการ (list) เป็นโครงสร้างข้อมูลสำคัญในภาษาโปรล็อก ลักษณะของรายการข้อมูลแสดงในตารางที่ 4-7

ตารางที่ 4-7 ตัวอย่างของรายการข้อมูลในโปรล็อก

รายการ (รูปแบบทั่วไป)	ส่วนหัว	ส่วนหาง	รายการ (รูปแบบในเชิงฟังก์ชัน)
[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))

รายการประกอบด้วยสมาชิกตั้งแต่ศูนย์ตัวขึ้นไปเขียนอยู่ภายในวงเล็บเพื่อแสดงสมาชิกในรายการเรียงลำดับตั้งแต่ตัวแรก (ถ้ามี) เป็นต้นไป รายการในโปรล็อกแบ่งเป็นสองส่วนคือ **ส่วนหัว (head)** และ **ส่วนหาง (tail)** ส่วนหัวคือสมาชิกตัวแรกในรายการ ส่วนหางคือรายการที่เหลือเมื่อลบสมาชิกตัวแรกออกไปแล้ว ดังที่ได้กล่าวแล้วข้างต้นว่าพจน์ในโปรล็อกคือตัวแปร ค่าคงที่ และพจน์ประกอบ รายการข้อมูลก็เป็นพจน์ประกอบแบบหนึ่งในโปรล็อกที่มีเครื่องหมายฟังก์ชันเป็น '.' และมีอาร์กิวเมนต์สองตัว ตัวแรกเป็นส่วนหัว ตัวที่สองเป็นส่วนหาง แต่รูปแบบในเชิงฟังก์ชันมักไม่เป็นที่นิยมเพราะอ่านเข้าใจยาก โดยมากเราจะใช้ในรูปแบบทั่วไป

รายการว่าง (empty list) แสดงด้วยสัญลักษณ์ $[]$ เพื่อแทนรายการที่ไม่มีสมาชิก รายการสามารถประกอบด้วยสมาชิกที่เป็นค่าคงที่ ตัวแปร หรือพจน์ประกอบก็ได้ ตัวอย่างที่ 4 ในตารางที่ 4-7 แสดงรายการที่มีสมาชิกตัวแรกเป็นรายการ $[a,b]$ ตัวอย่างของรายการที่มีสมาชิกเป็นตัวแปรก็เช่น $[X,Y]$ ซึ่งหมายถึงรายการที่มีสมาชิก 2 ตัว ตัวแรกคือ X ตัวที่สองคือ Y ตัวอย่างตัวที่ 5 ในตารางแสดงรายการ $[X|Y]$ (ซึ่งไม่เท่ากับ $[X,Y]$) เป็นรูปแบบการเขียนรายการในโปรล็อกที่มีความหมายว่าส่วนหัวคือ X ส่วนหางคือ Y (สังเกตว่า Y เป็นรายการ) X สามารถจับคู่ได้กับพจน์ใดๆ ส่วน Y จับคู่ได้กับรายการที่มีสมาชิกตั้งแต่ศูนย์ตัวขึ้นไป ดังนั้น $[X|Y]$ สามารถจับคู่ได้กับรายการที่มีสมาชิกตั้งแต่หนึ่งตัวขึ้นไป เช่น $[1]$, $[a,b]$, $[s(0),s(s(0)),s(s(s(0)))]$, $[1,a,2,b]$ เป็นต้น

เราสามารถมองได้ว่า $[]$ เป็นข้อมูลที่มีลำดับต่ำสุดในโครงสร้างข้อมูลประเภทนี้ รายการที่มีสมาชิกตัวเดียวเป็นข้อมูลที่มีลำดับถัดไป เป็นต้น ในหัวข้อนี้เราจะพิจารณาโปรแกรมที่จัดการกับรายการดังต่อไปนี้

โปรแกรมภาวะสมาชิก

โปรแกรมแรกในหัวข้อนี้เป็นโปรแกรมตรวจสอบภาวะสมาชิกของรายการ (membership of a list) ทำหน้าที่ตรวจสอบว่าพจน์หนึ่งๆ เป็นสมาชิกของรายการที่กำหนดให้หรือไม่ เขียนแทนด้วย $\text{member}(X,Ys)$ มีความหมายว่า X เป็นสมาชิกของรายการ Ys มีโปรแกรมดังตารางที่ 4-8 ด้านล่างนี้

ตารางที่ 4-8 โปรแกรมภาวะสมาชิก

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

กฎข้อแรกเป็นอนุประโยคฐานมีความหมายว่า X เป็นสมาชิกของรายการที่มีส่วนหัวคือ X ส่วนหางเป็นรายการใดๆ ($_$) กฎข้อที่สองเป็นอนุประโยคเรียกซ้ำหมายความว่า ถ้า X เป็นสมาชิกของรายการ Y ใดๆ แล้ว X จะเป็นสมาชิกของรายการนั้นที่มีพจน์หนึ่งตัวเพิ่มเข้าที่ส่วนหัว $([_|Y])$ ด้วย ตัวอย่างการทำงานของโปรแกรมกับข้อคำถาม $\text{'?- member(c,[a,b,c,d])'}$ แสดงในรูปที่ 4-3

```
member(c,[a,b,c,d]).
      |
      | member(c,[b,c,d])
      |
      | member(c,[c,d])
      |
      |
      |
```

รูปที่ 4-3 ตามรอยการทำงานของโปรแกรมภาวะสมาชิก

จากข้อคำถาม ‘?- member(c,[a,b,c,d]).’ ซึ่งจับคู่ได้กับส่วนหัวของกฎข้อที่ 2 ใน ตารางที่ 4-8 โดยการแทนค่า $\{X = c, _ = a, Y = [b,c,d]\}$ และพิสูจน์ส่วนลำตัวของกฎ ‘member(c,[b,c,d])’ ในการพิสูจน์ส่วนลำตัวนี้โปรแกรมเรียกซ้ำรอบถัดมาจะจับคู่กับส่วนหัวของกฎข้อที่ 2 โดยการแทนค่า $\{X = c, _ = b, Y = [c,d]\}$ พิสูจน์ส่วนลำตัว ‘member(c,[c,d])’ และในครั้งนี้นี้จับคู่ได้กับกฎข้อที่ 1 ทำให้ข้อคำถามเป็นจริง

ดังที่เราเห็นในตัวอย่างนี้รายการ ‘[a,b,c,d]’ ในอาร์กิวเมนต์ที่สองของข้อคำถามจะถูกลดลำดับลงทีละหนึ่ง (รายการสั้นลงทีละหนึ่ง) จนกระทั่งส่วนหัวของอาร์กิวเมนต์ตัวนี้เท่ากับ c และจะหยุดได้ด้วยอนุประโยคฐาน

โปรแกรมต่อรายการ

โปรแกรมต่อรายการ ‘append(X,Y,Z)’ ทำหน้าที่ต่อรายการ X เข้ากับรายการ Y ได้ผลลัพธ์เป็นรายการ Z โดยการต่อคือการนำสมาชิกของ Y ทุกตัวมาต่อเข้าข้างท้ายของสมาชิกของ X ตัวอย่างเช่น ‘append([a,b],[1,2],Z)’ จะได้ Z เป็น [a,b,1,2] โปรแกรมต่อรายการเป็นดัง ตารางที่ 4-9 ด้านล่างนี้

ตารางที่ 4-9 โปรแกรมต่อรายการ

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

วิธีเขียนโปรแกรมเรียกซ้ำสำหรับจัดการโครงสร้างข้อมูลแบบรายการมักมีลักษณะคล้ายกับโปรแกรมเรียกซ้ำสำหรับจัดการกับตัวเลข กล่าวคือเราควรเริ่มเขียนอนุประโยคฐานก่อน โดยกำหนดว่าต้องการลดลำดับของข้อมูลที่อาร์กิวเมนต์ตัวใด ในที่นี้จะทำให้อาร์กิวเมนต์ตัวแรก ดังนั้นเราจะได้ข้อมูลที่มีลำดับต่ำสุดของรายการเป็นรายการว่าง ทำให้เราเขียนกฎข้อที่ 1 ได้เป็น ‘append([],L,?)’ โดยสมมติว่าอาร์กิวเมนต์สองตัวแรกเป็นอินพุต ตัวที่สามเป็นเอาต์พุต (ดังที่ได้กล่าวแล้วว่าอาร์กิวเมนต์ในโปรแกรมโปรล็อกมักเป็นได้ทั้งอินพุตและเอาต์พุต แต่เพื่อความสะดวกในการเขียนโปรแกรม ณ จุดนี้เราจะสมมติว่าอาร์กิวเมนต์สองตัวแรกเป็นอินพุต และตัวที่สามเป็นเอาต์พุต) และจะได้ค่าของ ? เป็น L เพราะว่ารายการว่างต่อกับรายการใดๆ จะได้รายการนั้นๆ

จากนั้นเราจะเขียนอนุประโยคเรียกซ้ำเพื่อลดลำดับของอาร์กิวเมนต์ตัวที่หนึ่ง ได้เป็น ‘append([A|L1],L2,?) :- append(L1,L2,??)’ ในทำนองเดียวกับที่เราเขียนโปรแกรมสำหรับตัวเลข ให้เราสมมติได้เลยว่า ?? เป็นตัวแปรหนึ่งๆ เช่นให้เป็น L3 แล้วจะได้ว่า ถ้า L1 ต่อกับ L2 ได้ L3 แล้ว [A|L1] (L1 ที่มีสมาชิก A เพิ่มที่ข้างหน้าหนึ่งตัว) ต่อกับ L2 จะได้อะไร ซึ่งทำให้เรารู้ว่า ? ก็คือ L3 ที่มีสมาชิก A เดิมเข้าที่ข้างหน้าหรือ [A|L3] นั่นเอง ทำให้เราได้กฎข้อที่สองเป็น ‘append([A|L1],L2,[A|L3]) :- append(L1,L2,L3).’

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

```

append([a,b],[1,2],X)
  \
   X = [a|Y]
append([b],[1,2],Y)
  \
   Y = [b|Z]
append([], [1,2], Z)
  \
   Z = [1,2]

```

รูปที่ 4-4 ตามรอยการทำงานของโปรแกรมต่อรายการ

4.3 นิเสธและตัด

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

4.3.1 นิเสธ

ข้อเท็จจริงหรือกฎในโปรล็อกเขียนแสดงความสัมพันธ์ที่เป็นจริง แต่เมื่อเราต้องการเขียนความสัมพันธ์ที่ไม่เป็นจริงในโปรล็อก เราเขียนได้โดยใช้ **นิเสธ (negation)** แทนด้วย 'not' (ในตัวแปลภาษาโปรล็อกบางตัวใช้ '+') แล้วตามด้วยความสัมพันธ์ที่คลุมด้วยวงเล็บ 'not(G)' โดยที่ G เป็นความสัมพันธ์ใดๆ

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

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)' ดังในตาราง หมายถึง X เป็นพี่น้องของ Y ถ้ามีพ่อแม่คนเดียวกันและ X ไม่เท่ากับ Y ('X \= Y' มี

ความสำคัญในกฎนี้ เพราะถ้าเราไม่ใส่เงื่อนไขนี้เราจะได้ว่า 'sibling(arthur,arthur)' เป็นจริง – คนคนเดียวกันเป็นพี่น้องกันเอง)

4.3.2 ตัวตัด

ตัวตัด (cut) เขียนแทนด้วย '!' ตัวตัดจะมีลักษณะการใช้งานที่ซับซ้อน แต่เป็นส่วนสำคัญในโปรล็อกเพื่อเขียนให้ได้โปรแกรมที่มีประสิทธิภาพ ดังนั้นจำเป็นต้องทำความเข้าใจอย่างถ่องแท้ หน้าที่หลักของตัวตัดคือการเปลี่ยนแปลงลำดับการทำงานของโปรแกรมโดยจะป้องกันการเกิดการย้อนรอย โดยปกติโปรล็อกจะพิสูจน์ข้อคำถามโดยอัตโนมัติและจะค้นหาเส้นทางทุกเส้นทางที่เป็นไปได้ด้วยการค้นหาแบบแนวลึกก่อน แต่บางครั้งเราอาจทราบล่วงหน้าว่าเส้นทางในการพิสูจน์บางเส้นทางไม่จำเป็นต้องค้นหาก็ได้เพราะจะไม่มีคำตอบในเส้นทางนั้นหรือเราไม่ต้องการให้โปรล็อกค้นหาในเส้นทางนั้น การใช้ตัวตัดก็จะช่วยให้เราสามารถควบคุมการทำงานของโปรล็อกได้ทำให้การหาคำตอบรวดเร็วและมีประสิทธิภาพมากขึ้น

หลักการทำงานของตัวตัดสามารถแสดงให้เห็นได้ดังรูปที่ 4-5 ซึ่งเป็นกฎข้อที่ 1 ของ H



รูปที่ 4-5 การทำงานของตัวตัด

เราสามารถแบ่งส่วนลำตัวของกฎ 'H :- B₁, B₂, ..., B_m!, B_{m+1}, ..., B_n.' ออกได้เป็น 2 ส่วนหลักคือส่วนด้านหน้าของเครื่องหมาย ! และส่วนด้านหลังของเครื่องหมาย การย้อนรอยสามารถเกิดขึ้นได้ภายในส่วนทั้งสองนี้ แต่ไม่สามารถข้ามผ่านตัวตัดไปได้ กล่าวคือถ้ามีเป้าหมาย B_i ใดๆ ระหว่าง B₁, B₂, ..., B_m ทำไม่สำเร็จด้วยการแทนค่าชุดหนึ่งจะเกิดการย้อนรอยเพื่อพยายามหาการแทนค่าชุดใหม่ได้ แต่เมื่อการทำงานผ่านตัวตัดไปแล้วและมีเป้าหมาย B_j ใดๆ ระหว่าง B_{m+1}, ..., B_n ทำไม่สำเร็จ จะไม่สามารถย้อนกลับไปทำใน B₁, B₂, ..., B_m ได้อีก อย่างไรก็ตามเมื่อผ่านตัวตัดมาแล้วการย้อนรอยสามารถเกิดขึ้นได้ภายใน B_{m+1}, ..., B_n

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

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

เมื่อตัดตัดถูกเรียกใช้งาน (การทำงานผ่านตัวตัดแล้ว) และพบว่ากฎข้อที่ 1 ด้านบนทำไม่สำเร็จ การย้อนรอยก็ไม่สามารถมาทำกฎข้อที่ 2 ของ H นี้ได้ เปรียบเสมือนเกิดกำแพงกันระหว่างกฎ อย่างไรก็ตามกำแพงกันนี้ (ทั้งกำแพงภายในกฎและกำแพงระหว่างกฎ) จะเกิดขึ้นก็ต่อเมื่อตัวตัดถูกเรียกใช้งานแล้วเท่านั้น

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 $W = 6$ สำหรับข้อความถาม '?- 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 ตามรอยการทำงานของโปรแกรมกรณีมีตัวตัด

จากตัวอย่างในรูปจะเห็นได้ว่าการทำงานผ่านตัวตัดมาแล้ว ได้ $W = 6$ และเมื่อโปรล็อกถูกบังคับให้ย้อนรอยด้วยคำสั่งให้หาคำตอบอื่น ‘;’ จึงย้อนรอยมาที่ ‘ $r(4,W)$ ’ ได้ ‘ $r(4,8)$ ’ แต่ ‘ $s(8)$ ’ เป็นเท็จและไม่สามารถย้อนรอยได้อีกแล้ว จึงไม่มีคำตอบอื่น

เราได้พิจารณาทำความเข้าใจเรื่องการทำงานของตัวตัดด้านบนแล้ว อย่างไรก็ตามการจะเขียนโปรแกรมที่มีตัวตัดได้อย่างมีประสิทธิภาพหรืออ่านโปรแกรมหนึ่งๆ เพื่อให้เข้าใจได้อย่างดีในกรณีที่โปรแกรมนั้นประกอบด้วยตัวตัด เราจำเป็นต้องเรียนรู้รูปแบบการใช้ตัวตัดตลอดจนความหมายของตัวตัดในรูปแบบนั้นๆ ต่อไปจะกล่าวถึงรูปแบบการเขียนโปรแกรมโดยใช้ตัวตัดดังต่อไปนี้

การใช้ตัวตัดร่วมกับเพรดิเคต ‘fail’

รูปแบบแรกของการใช้ตัวตัดเป็นการใช้ร่วมกับเพรดิเคต fail ซึ่งเพรดิเคตนี้เป็นเพรดิเคตในตัวโปรล็อกมีความหมายว่า ‘เป็นเท็จ’ ทุกครั้งที่ถูกเรียก ดังนั้นเมื่อเราเรียก fail จะเกิดการย้อนรอยทันทีที่ fail เป็นเพรดิเคตที่ไม่มีอาร์กิวเมนต์

สมมติว่าเราต้องการเขียนโปรแกรมอธิบายความสัมพันธ์ว่า ‘mary ชอบสัตว์ทุกตัวที่ไม่ใช่’ เราจะเขียนโปรแกรมที่เป็นการใช้งานร่วมกันระหว่างตัวตัดกับ fail ได้ดังตารางที่ 4-12 ด้านล่างนี้

ตารางที่ 4-12 โปรแกรมตัวอย่างแสดงการใช้งานของตัวตัดร่วมกับ 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).
```

กฎข้อแรกแสดงว่าถ้า X เป็นงูแล้ว ‘like(mary,X)’ จะเป็นเท็จทันทีและไม่สามารถย้อนรอยไปยังกฎข้ออื่นๆ ได้อีก ดังนั้นข้อคำถาม ‘?- like(mary,small_snake).’ จะได้คำตอบเป็น No แต่ถ้า X เป็นสัตว์อื่นๆ ที่ไม่ใช่งู จะได้ว่ากฎข้อที่ 1 ไม่เป็นจริงจากเงื่อนไข ‘snake(X)’ ทำให้ย้อนรอยมาที่กฎที่ 2 ได้และทำสำเร็จด้วยเงื่อนไข ‘animal(X)’ เช่นข้อคำถาม ‘?- like(mary,dog).’ จะให้คำตอบเป็น Yes

ตัวตัดเขียว

ตัวตัดเขียว (green cut) เป็นรูปแบบการใช้ตัวตัดรูปแบบหนึ่ง โปรแกรมที่มีตัวตัดเขียวสามารถลบตัวตัดเขียวออกได้โดยไม่กระทบต่อความหมายของโปรแกรม (คำตอบที่ได้จากโปรแกรมเหมือนเดิมทุกประการ) แต่ประสิทธิภาพของโปรแกรมที่มีตัวตัดเขียวจะดีกว่า ตัวตัดที่มีคุณสมบัติตรงข้ามกับตัวตัดเขียวคือ **ตัวตัดแดง (red cut)** ดังจะกล่าวต่อไป

ตัวตัดเขียวใช้กับโปรแกรมซึ่งประกอบด้วยเพรดิเคตที่มีการทดสอบเชิงกำหนด (deterministic test) ตัวอย่างของการทดสอบแบบนี้เช่น ' $X < Y$ ' หรือ ' $X = Y$ ' หรือ ' $X > Y$ ' ซึ่งจะ เป็นจริงแค่กรณีใดกรณีหนึ่งเท่านั้น คือถ้า X น้อยกว่า Y แล้ว X จะไม่เท่ากับ Y และ X จะ ไม่มากกว่า Y

ตัวอย่างในตารางที่ 4-13 ด้านล่างนี้เป็นโปรแกรมหาค่าต่ำสุดระหว่างตัวเลข 2 ตัว โดย ใช้ตัวตัดเขียว ' $\text{minimum}(X,Y,Z)$ ' มีความหมายว่า Z เป็นค่าต่ำสุดระหว่างตัวเลข 2 ตัวของ X กับ Y

ตารางที่ 4-13 โปรแกรมหาค่าต่ำสุดของตัวเลข 2 ตัว

1: $\text{minimum}(X,Y,Z)$:- $X < Y, !, Z = X.$
2: $\text{minimum}(X,Y,Z)$:- $X = Y, !, Z = X.$
3: $\text{minimum}(X,Y,Z)$:- $X > Y, Z = Y.$

กฎข้อที่หนึ่งมีความหมายว่าถ้า X น้อยกว่า Y แล้ว Z จะมีค่าเท่ากับ X กฎข้อที่ 2 คือ ถ้า X เท่ากับ Y แล้ว Z เท่ากับ X และกฎข้อที่ 3 คือถ้า X มากกว่า Y แล้ว Z เท่ากับ Y การ ใส่ตัวตัดเข้าที่ด้านหลังของการทดสอบเชิงกำหนด ($X < Y$, $X = Y$ และ $X > Y$) จะไม่เปลี่ยน ความหมายของโปรแกรม เนื่องจากว่าถ้า $X < Y$ แล้วตัวตัดจะถูกเรียกทำให้การย้อนรอยไม่สามารถทำได้ และในกรณีที่โปรแกรมนี้ไม่ใส่ตัวตัดก็มีความหมายเหมือนกัน เพราะว่าถ้า $X < Y$ แล้วไม่มีตัวตัด การย้อนรอยจะเลือกกฎข้อที่ 2 และ 3 ได้แต่ก็ไม่สามารถได้คำตอบอื่น เพราะว่า X จะไม่เท่ากับ Y และ X ก็จะไม่มากกว่า Y ดังนั้นจะเห็นได้ว่าการใส่ตัวตัดไม่ทำให้ ความหมายของโปรแกรมเปลี่ยนไป

แม้ว่าความหมายของโปรแกรมที่ใส่ตัวตัดกับไม่ใส่จะเหมือนกัน แต่สิ่งที่เราได้เพิ่มขึ้น จากการใส่ตัวตัดคือโปรแกรมจะทำงานเร็วขึ้น สมมติว่าข้อคำถามคือ '?- $\text{minimum}(4,5,Z).$ ' ข้อคำถามนี้จับคู่ได้กับส่วนหัวของกฎที่ 1 และทดสอบส่วนลำตัวของกฎ พบว่า ' $4 < 5$ ' เป็น จริง การทำงานจึงผ่านตัวตัดและได้ $Z = 4$ เป็นคำตอบ ณ จุดนี้ถ้ามีการย้อนรอยเกิดขึ้น อาจเนื่องมาจากผู้ใช้ต้องการคำตอบอื่น หรืออาจเกิดจากโปรแกรมนี้เป็นโปรแกรมย่อยที่ถูก เรียกใช้ด้วยโปรแกรมอื่นและที่โปรแกรมอื่นนั้นมีเพรดิเคตเป้าหมายบางตัวทำไม่สำเร็จทำให้ เกิดการย้อนรอยซึ่งส่งผลให้โปรแกรมนี้เกิดการย้อนรอยด้วย เมื่อเกิดการย้อนรอยขึ้น โปรแกรมนี้จะตอบได้ทันทีว่าไม่มีคำตอบอื่น เพราะว่าตัวตัดถูกสั่งทำงานแล้วทำให้การย้อน รอยในโปรแกรมนี้ไม่สามารถเลือกกฎข้ออื่นได้อีก

สมมติว่าโปรแกรมนี้ไม่มีตัวตัดอยู่เลยและให้ข้อคำถามเหมือนเดิมคือ '?- $\text{minimum}(4,5,Z).$ ' ข้อคำถามนี้จับคู่ได้กับส่วนหัวของกฎที่ 1 และทดสอบส่วนลำตัวของ กฎพบว่า ' $4 < 5$ ' เป็นจริงและได้ $Z = 4$ เป็นคำตอบ ณ จุดนี้ถ้ามีการย้อนรอยเกิดขึ้น โปรแกรมจะย้อนรอยมายังกฎข้อที่ 2 ได้เพราะไม่มีตัวตัดและจับคู่กับส่วนหัวของข้อที่ 2 ได้

จากนั้นทดสอบส่วนลำตัวของกฎโดยทดสอบว่า ' $4 = 5$ ' หรือไม่และพบว่าไม่เป็นจริง จึงย้อนรอยอีกไปยังกฎที่ 3 จับคู่กับส่วนหัวของกฎที่ 3 ได้ ทดสอบว่า ' $4 > 5$ ' และพบว่าไม่เป็นจริง จึงได้ผลว่าไม่มีคำตอบอื่นเหมือนกับกรณีที่มีตัวตัด แต่จะทำงานช้ากว่าและไปทดสอบในส่วนที่ไม่จำเป็น กล่าวคือถ้า ' $4 < 5$ ' เป็นจริงก็ไม่จำเป็นต้องทดสอบในกฎข้อที่ 2 และ 3

ตัวอย่างการเขียนโปรแกรมโดยใช้ตัวตัดเขี้ยวอีกตัวอย่างคือโปรแกรม ' $\text{insert}(X, Ys, Zs)$ ' ทำหน้าที่แทรกสมาชิก X เข้าไปในรายการของตัวเลข Ys ที่เรียงจากน้อยไปมาก ได้เป็นรายการของตัวเลข Zs ที่เรียงลำดับจากน้อยไปมาก เช่น ' $\text{insert}(2, [0, 1, 4, 6], [0, 1, 2, 4, 6])$ ' เป็นต้น โปรแกรมเป็นดังตารางที่ 4-14 ด้านล่างนี้

ตารางที่ 4-14 โปรแกรมแทรกตัวเลข

```
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) และอนุประโยคเรียกซ้ำ 1 ข้อ การเขียนโปรแกรมแทรกข้อมูลก็เช่นเดียวกับโปรแกรมเรียกซ้ำอื่นๆ คือเราเริ่มเขียนจากอนุประโยคฐานก่อน โดยสมมติว่าจะลดลำดับของอาร์กิวเมนต์ตัวที่ 2 ดังนั้นจะได้ว่ารายการที่มีลำดับต่ำสุดคือ $[]$ ได้เป็น ' $\text{insert}(X, [], [X])$ ' คือแทรก X เข้าไปในรายการว่าง จะได้รายการที่มีสมาชิกตัวเดียวคือ X จากนั้นเราก็เขียนอนุประโยคเรียกซ้ำโดยลดลำดับของอาร์กิวเมนต์ตัวที่ 2 จะได้ว่า ' $\text{insert}(X, [Y|Ys], ?) :- \text{insert}(X, Ys, Zs)$ ' อย่างไรก็ตามการแทรกสมาชิกไว้ในรายการที่ต้องคำนึงถึงลำดับนั้น เราต้องใช้การเปรียบเทียบด้วย ดังนั้นเราเพิ่มการเปรียบเทียบเข้าไปในกฎข้อที่ 2 ได้เป็น ' $\text{insert}(X, [Y|Ys], ?) :- X > Y, \text{insert}(X, Ys, Zs)$ ' ณ จุดนี้เราพบว่าถ้า $X > Y$ และแทรก X ใน Ys ได้ Zs แล้ว การแทรก X ในรายการ Ys ตัวเดิมและมี Y ปะอยู่ข้างหน้า (คือรายการ $[Y|Ys]$) ก็ต้องได้รายการ Zs ตัวเดิมและมี Y ปะไว้ข้างหน้า (คือรายการ $[Y|Zs]$) ตัวอย่างเช่นให้การแทนค่าคือ $\{X = 2, Y = 0, Ys = [1, 4, 6]\}$ เราจะได้ว่า $\text{insert}(2, [0, 1, 4, 6], ?) :- 2 > 0, \text{insert}(2, [1, 4, 6], Zs)$ 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 ' $\text{insert}(X, [Y|Ys], ?) :- X <= Y$ ' กรณีนี้หา ? ได้อย่างง่ายว่าถ้า $X <= Y$ ดังนั้นแทรก X ไว้ในรายการที่มีส่วนหัวคือ Y ที่มากกว่าหรือเท่ากับ X ก็จะต้องนำ X ไว้หน้า Y ได้ ? เป็น $[X, Y|Ys]$

โปรแกรม ณ จุดนี้คือ

$\text{insert}(X,[],[X]).$

$\text{insert}(X,[Y|Ys],[Y|Zs]) :- X > Y, \text{insert}(X,Ys,Zs).$

$\text{insert}(X,[Y|Ys],[X,Y|Ys]) :- X \leq Y.$

และเราพบว่ากฎที่ 2 และ 3 มีการทดสอบเชิงกำหนด เราจึงใส่ตัวตัดไว้หลังการทดสอบของกฎข้อที่ 2 ได้โปรแกรมดังตารางที่ 4-14

ตัวตัดแดง

ตัวตัดแดง (red cut) ใช้ในกรณีที่เรต้องการละเงื่อนไขบางตัวออกจากโปรแกรมแล้วแทนที่ด้วยตัวตัด การใช้ตัวตัดประเภทนี้ต้องระวังเนื่องจากว่าหากเราไปนำตัวตัดออกจากโปรแกรมความหมายของโปรแกรมจะเปลี่ยนไป ถ้าต้องการให้ความหมายคงเดิมจะต้องนำเงื่อนไขที่ละไว้กลับเข้าที่เดิม ดังนั้นเมื่อเราไปอ่านโปรแกรมหนึ่งๆ ถ้าหากไม่ระวัง ไปพบตัวตัดแล้วลบตัวตัดออก โปรแกรมจะมีความหมายผิดไปจากเดิมได้ถ้าตัวตัดนั้นเป็นตัวตัดแดง ตัวอย่างของตัวตัดประเภทนี้เช่นถ้าเรต้องการเขียนโปรแกรม 'if_then_else(P,Q,R)' โดยที่ P, Q และ R แทนความสัมพันธ์ใดๆ ซึ่งมีความหมายว่า ถ้า P เป็นจริงสั่งทำ Q ถ้าไม่จริงทำ R สามารถเขียนโปรแกรมได้ดังตารางที่ 4-16

ตารางที่ 4-15 โปรแกรม 'ถ้าแล้ว' กรณีใช้ตัวตัด

1: $\text{if_then_else}(P,Q,R) :- P, !, Q.$ 2: $\text{if_then_else}(P,Q,R) :- R.$
--

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

แต่ถ้าเราลบตัวตัดออกความหมายจะผิดเพี้ยนไป กล่าวคือไม่ว่า P จะจริงหรือไม่ก็จะทำ R เสมอ ซึ่งไม่ใช่ความหมายที่ต้องการ เช่นถ้า P เป็นจริงแล้วเกิด Q เป็นเท็จหรืออาจเกิดการย้อนรอยจากสาเหตุอื่นๆ (เช่นผู้ใช้สั่งหาคำตอบอื่นหรือเกิดการย้อนรอยจากโปรแกรมที่เรียกใช้โปรแกรมนี้ ฯลฯ) โปรแกรมจะย้อนรอยมาทำกฎข้อที่ 2 ซึ่งไม่ควรเป็นเช่นนั้นเพราะ P เป็นจริงไม่ควรทำ R

โปรแกรมที่สมมูลกับโปรแกรมที่ไม่ใช้ตัวตัดต้องเขียนดังตารางที่ 4-16 ต่อไปนี้

ตารางที่ 4-16 โปรแกรมถ้าแล้วกรณีไม่ใช่ตัวตัด

```

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

```

เมื่อพิจารณากฎข้อที่ 1 ในโปรแกรมแบบไม่ใช่ตัวตัดนี้ จะเห็นว่าในกรณีที่ P เป็นจริงจะทำ Q และถึงแม้จะเกิดการย้อนรอยก็ยังไม่ทำ R เพราะกฎข้อที่ 2 มีการตรวจสอบว่าจะทำ R ได้ก็ต่อเมื่อ not(P) ต้องเป็นจริง (หรือ P เป็นเท็จนั่นเอง) อย่างไรก็ตามโปรแกรมนี้มีประสิทธิภาพต่ำกว่าโปรแกรมแบบใช้ตัวตัด เพราะว่ามีสมมติ P เป็นจริง กฎข้อที่ 1 จะทำ Q หากพบว่าเกิดการย้อนรอยขึ้นโปรแกรมจะลงมาทำกฎข้อที่ 2 (เพราะไม่มีตัวตัด) และทดสอบว่า not(P) เป็นจริงหรือไม่ การทดสอบนี้ต้องทำ P ดูก่อนและเมื่อทำ P ดูก็จะได้ว่า P เป็นจริง ดังนั้น not(P) เป็นเท็จ จะเห็นได้ว่า P ถูกทดสอบถึง 2 ครั้ง (ที่กฎข้อที่ 1 กับกฎข้อที่ 2) เกิดการทำงานซ้ำซ้อนขึ้นต่างจากโปรแกรมแบบใช้ตัวตัดที่ทำ P ครั้งเดียว

ตัวอย่างโปรแกรมที่ใช้ตัวตัดแต่งอีกตัวอย่างที่จะยกให้ดูนี้เป็นโปรแกรม 'delete(Xs,X,Ys)' ทำหน้าที่ลบสมาชิกทุกตัวที่เท่ากับ X ออกจากรายการ Xs ได้เป็นรายการ Ys เช่น 'delete([1,a,2,3,a],a,[1,2,3])' เป็นต้น โปรแกรมเป็นดังตารางที่ 4-17 ต่อไปนี้

ตารางที่ 4-17 โปรแกรมลบสมาชิก

```

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

```

เราเริ่มจากการเขียนอนุประโยคฐานก่อน โดยสมมติว่าอาร์กิวเมนต์สองตัวแรกเป็นอินพุต ตัวที่สามเป็นเอาต์พุต ทำการลดลำดับของอาร์กิวเมนต์ตัวที่หนึ่ง ได้ว่า 'delete([],X,?)' และพบว่าลบ X ออกจากรายการว่างต้องได้รายการว่าง ดังนั้น ? = [] ได้ กฎข้อที่ 3 (ที่นำอนุประโยคฐานไว้เป็นกฎข้อที่ 3 เนื่องจากเหตุผลทางด้านประสิทธิภาพของโปรแกรม) จากนั้นเขียนอนุประโยคเรียกซ้ำได้ว่า 'delete([X|Ys],X,?) :- delete(Ys,X,Zs)' เป็นกรณีที่ตัวที่ต้องการลบออกเป็นตัวแรกในรายการ (X = X) เมื่อเราสมมติว่าลบ X ออกจาก Ys ได้ Zs ('delete(Ys,X,Zs)' ที่ส่วนลำตัว) แล้วหว่า ? เป็นเท่าไรของ Xs เราจะได้ว่า ? เท่ากับ Zs เพราะว่าถ้าการลบ X ออกจาก Ys ได้ Zs แล้ว การลบ X ออกจาก Ys ตัวเดิมที่มี X เหมือนกันมาปะไว้ข้างหน้า (ซึ่งหมายถึง [X|Ys]) ก็ต้องได้ผลลัพธ์เดิมคือ Zs เช่น ถ้า 'delete([2,3,a],a,[2,3])' เป็นจริงแล้ว 'delete([a,2,3,a],a,?)' ก็ต้องได้ ? เท่าเดิมคือ [2,3] เป็นต้น

กรณีของอนุประโยคฐานด้านบนเป็นกรณีที่ X เหมือนกับตัวแรกของรายการที่จะลบ ซึ่งยังไม่ครอบคลุมกรณีที่เหมือนกัน ดังนั้นเราเพิ่มกฎอีกข้อที่เป็นข้อที่ 2 ในตาราง ได้เป็น 'delete([Y|Ys],X,?) :- $X \neq Y$, delete(Ys,X,Zs)' แล้วหว่า ? 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]) :-  $X \neq Y$ , delete(Ys,X,Zs).
```

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

เราจะเห็นว่า มีการตรวจสอบเงื่อนไข $X \neq Y$ ที่กฎข้อที่ 2 ดังนั้นถ้าเราจะละเงื่อนไขนี้ เราก็ใส่ตัวตัดไว้ที่กฎข้อแรกได้เป็น

```
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 ก็เพราะว่าหากกฎข้อที่ 2 ถูกเรียกใช้งานแล้ว เราไม่จำเป็นต้องทดลองจับคู่ข้อความกับกฎข้อที่ 3 เพราะจับคู่ไปก็ทำไม่สำเร็จ (กฎข้อที่ 3 มีอาร์กิวเมนต์ตัวแรกเป็นรายการว่างต่างจากของกฎข้อที่ 1 และ 2) เราอยู่ก่อนแล้วว่าจับคู่ไปก็ทำไม่สำเร็จ เราจึงใส่ตัวตัดเพื่อให้โปรล็อกไม่ต้องทดลองทำกฎข้อที่ 3 ในกรณีที่กฎข้อที่ 2 จับคู่สำเร็จซึ่งจะช่วยให้โปรแกรมทำงานรวดเร็วขึ้นอีก

เอกสารอ่านเพิ่มเติมและแบบฝึกหัด

ตำราของ Bratko [Bratko, 1990] มีเนื้อหาของภาษาโปรล็อกไว้ค่อนข้างสมบูรณ์ โดยเฉพาะอย่างยิ่งได้เน้นเรื่องการเขียนโปรแกรมโปรล็อกสำหรับงานทางปัญญาประดิษฐ์ซึ่งมีตัวอย่างให้ดูจำนวนมาก สำหรับผู้ที่ต้องการศึกษาอย่างจริงจังเกี่ยวกับการเขียนโปรแกรมภาษานี้ ควรอ่าน [Sterling & Shapiro, 1994] ซึ่งเป็นหนังสือเล่มที่เยี่ยมที่สุดเล่มหนึ่งของโปรล็อกมีเทคนิคการเขียนโปรแกรมต่างๆ 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.

แบบฝึกหัด

1. จงเขียน `reverse(Xs, Ys)` ที่ทำหน้าที่สลับลำดับสมาชิกในรายการ `Xs` เป็นรายการ `Ys` (มีสมาชิกอยู่ในลำดับตรงกันข้ามกับของ `Xs`) โดยไม่ใช้ `append` (โปรแกรมในตารางที่ 4-9)

ตัวอย่าง : `reverse([a, b, c], Ys)` ได้ `Ys = [c, b, a]`
`reverse([1, 2, 3, 4, 5], Ys)` ได้ `Ys = [5, 4, 3, 2, 1]`
`reverse(Xs, [m, y, o, b])` ได้ `Xs = [b, o, y, m]`

2. จงเขียนโปรแกรม `last(As, L)` ที่ทำหน้าที่หา `L` ซึ่งเป็นสมาชิกตัวสุดท้ายในรายการ `As`

ตัวอย่าง : `last([1, 2, 3], L)` ได้ `L = 3`
`last([a,b,c,d], L)` ได้ `L = d`

3. จงเขียนโปรแกรม `minus(X,Y,Z)` ที่ทำหน้าที่หา `Z` ซึ่งมีค่าเท่ากับ `X` ลบ `Y` โดยที่ `X`, `Y`, `Z` เป็นจำนวนธรรมชาติ

ตัวอย่าง : `minus(s(s(s(s(0))))s(s(s(0))),Z)` ได้ `Z = s(0)`
`minus(s(s(s(s(0))))s(s(0)),Z)` ได้ `Z = s(s(0))`

4. จงเขียนโปรแกรม `power(X,Y,Z)` ที่ทำหน้าที่หา `Z` ซึ่งมีค่าเท่ากับ `X` ยกกำลัง `Y` โดยที่ `X`, `Y`, `Z` เป็นจำนวนธรรมชาติ

ตัวอย่าง : `power(s(s(0)),s(s(s(0))),Z)` ได้ `Z = s(s(s(s(s(s(s(0))))))`
`power(s(s(0)),0,Z)` ได้ `Z = s(0)`

5. พิจารณาโปรแกรมในตารางที่ 4-11 สมมติว่าโปรแกรมนี้ไม่มีตัวตัดเลย ข้อคำถาม ‘?-a(1,W).’ จะมีคำตอบกี่ตัวอะไรบ้าง

6. พิจารณาโปรแกรมด้านล่างนี้

`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,[],[]).`

หมายเหตุ: กำหนดให้อาร์กิวเมนต์ 2 ตัวแรกเป็นอินพุต (ถูกแทนค่าแล้วเวลาเรียก)

- จงอธิบายว่าโปรแกรมนี้ใช้ทำอะไร

- จงเขียนโปรแกรมนี้ใหม่โดยใช้ cut (!)

7. พิจารณาปริศนาตรรกะต่อไปนี้

กาลครั้งหนึ่งนานมาแล้ว มีชาย 5 คนอาศัยอยู่ในบ้านคนละหลังซึ่งแต่ละหลังมีสีแตกต่างจากบ้านอื่น ชายทั้ง 5 มีเชื้อชาติแตกต่างกัน เลี้ยงสัตว์คนละชนิด ชอบเครื่องดื่มและบุหรืคนละยี่ห้อ เงื่อนไขของปัญหามีดังนี้

- (1) คนอินเดียอาศัยอยู่ในบ้านสีแดง
- (2) คนไทยเลี้ยงสุนัข
- (3) กาแฟเป็นเครื่องดื่มในบ้านสีเขียว
- (4) คนลาวดื่มชา
- (5) บ้านสีเขียวอยู่ด้านขวามือของบ้านสีดำ
- (6) คนสูบกงูทองเหลืองตาก
- (7) สายฝนถูกสูบในบ้านสีเหลือง
- (8) นมเป็นเครื่องดื่มในบ้านที่อยู่ตรงกลาง
- (9) คนพม่าอยู่ที่บ้านริมสุดด้านซ้ายมือ
- (10) คนที่สูบกงูทองอาศัยอยู่ในบ้านที่ติดกับบ้านที่เลี้ยงแมว
- (11) สายฝนถูกสูบในบ้านที่ติดกับบ้านที่เลี้ยงม้า
- (12) คนสูบมาร์ลโบโรดื่มน้ำส้ม
- (13) คนญี่ปุ่นสูบเซเวนสตาร์
- (14) คนพม่าอาศัยอยู่ในบ้านที่ติดกับบ้านสีฟ้า

คำถามคือ ใครเป็นเจ้าของกบและใครดื่มกระทิงแดง ?

จงเขียนโปรแกรมโปรล็อกเพื่อแก้ปัญหานี้ พร้อมทั้งแสดงผลลัพธ์ของคำถามว่าคืออะไรและเป็นไปได้ทั้งหมดกี่ทางเลือก

การประมวลผลภาษาธรรมชาติ

5

การประมวลผลภาษาธรรมชาติ – เอ็นแอลพี (Natural Language Processing – NLP) เป็นกระบวนการที่จะทำให้คอมพิวเตอร์เข้าใจภาษามนุษย์โดยรับอินพุตเป็นข้อความในภาษาหนึ่งๆ แล้วทำความเข้าใจว่าผู้ป้อนข้อความกล่าวถึงอะไร แม้ว่าภาษามนุษย์หรือภาษาธรรมชาตินั้นมีคุณสมบัติต่างๆ ที่ทำให้เราสามารถสื่อสารและเข้าใจกันได้ แต่การที่จะทำให้ระบบเอ็นแอลพีเข้าใจภาษาธรรมชาตินั้นทำได้ยากลำบาก อันเนื่องมาจากคุณสมบัติทางภาษาที่ทำให้เกิดปัญหาและความยุ่งยากในพัฒนาระบบเอ็นแอลพี อย่างเช่นด้านล่างนี้

- ประโยคเป็นคำอธิบายสารสนเทศที่ผู้พูดตั้งใจถ่ายทอดไปยังผู้ฟัง เช่นประโยค

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: ภาษาธรรมชาติให้ผู้พูดสื่อสารด้วยข้อความที่คลุมเครือหรือชัดเจนเท่าที่ผู้พูดต้องการได้โดยผู้พูดอาจจะข้อความบางอย่างที่ผู้ฟังรู้อยู่แล้ว

- ประโยคเดียวกันอาจหมายถึงสิ่งของหลายอย่างโดยขึ้นอยู่กับสถานการณ์ที่พูด เช่น

Where's the water?

ถ้าประโยคนี้ถูกพูดในห้องทดลองเคมี “water” ก็อาจหมายถึงน้ำบริสุทธิ์ ถ้าพูดตอนหิวก็อาจหมายถึงน้ำที่เราหิบบดได้ หรือพูดในขณะที่กำลังซ่อมหลังคาบ้านที่รั่วก็อาจหมายถึงน้ำที่เปื้อนว่ารั่วตรงไหน

ข้อดี: ภาษาช่วยให้เราสามารถอธิบายถึงสิ่งที่มีมากมายไม่จำกัดโดยใช้คำพูดที่จำกัด

- ภาษามีการเปลี่ยนแปลงอยู่ตลอดเวลา มีศัพท์ใหม่ๆ เกิดขึ้นเสมอ เช่น

I'll fax it to you.

คำว่า fax ดั้งเดิมเป็นคำนามแต่มีการเปลี่ยนแปลงมาใช้เป็นกริยา

ข้อดี: ภาษาที่มีการวิวัฒนาการตามที่เราต้องการจะสื่อสาร

- มีวิธีการพูดได้หลายแบบสำหรับสิ่งเดียวกัน เช่น

Mary was born on October 11.

Mary's birthday is October 11.

ทั้งสองประโยคนี้นี้สื่อความหมายถึงเรื่องเดียวกัน

ข้อดี: เวลาที่เรารู้อะไรต่าง ๆ มากมาย ข้อเท็จจริงหนึ่งจะทำให้รู้ถึงข้อเท็จจริงอีกเรื่องหนึ่งได้ ภาษาที่มีไว้ให้ผู้รู้สิ่งต่าง ๆ มากมายใช้ได้อย่างสะดวก

5.1 ขั้นตอนในการเข้าใจภาษาธรรมชาติ

การเข้าใจภาษาธรรมชาติ (Natural Language Understanding) มีขั้นตอนดังต่อไปนี้

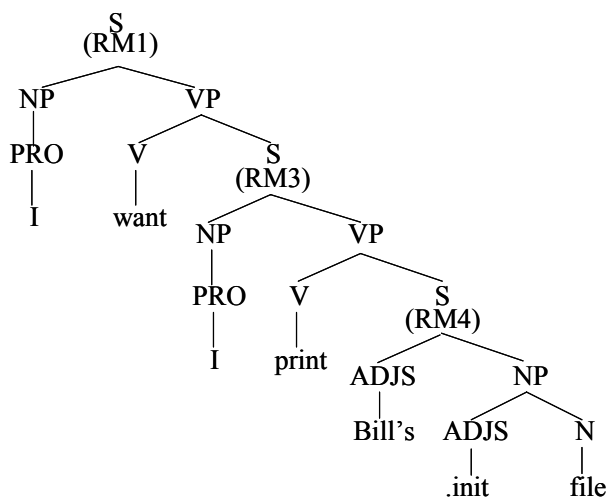
1. *การวิเคราะห์ทางองค์ประกอบ (morphological analysis)* เป็นการวิเคราะห์ในหน่วยคำ ว่าคำๆ หนึ่งสามารถแยกออกมาหน่วยย่อยได้เป็นอะไรบ้าง เช่น 'friendly' มาจาก 'friend' กับ 'ly' ก็สามารถบอกได้ว่าคำนี้มีหน้าที่อะไร หรือในภาษาไทยเช่น 'การทำงาน' ก็จะแยกเป็น 'การ' กับ 'ทำงาน' เป็นต้น
2. *การวิเคราะห์ทางวากยสัมพันธ์ (syntactic analysis)* เป็นการวิเคราะห์ทางไวยากรณ์ จุดมุ่งหมายก็เพื่อต้องการดูว่าประโยคที่รับเข้าซึ่งประกอบด้วยคำหลายๆ คำเรียงต่อกันนั้นมีโครงสร้างเชิงวากยสัมพันธ์เป็นอย่างไร คำไหนทำหน้าที่เป็นประธาน กริยา กรรม หรือส่วนใดเป็นวลี เป็นต้น โดยจะวิเคราะห์ลำดับของคำให้อยู่ในโครงสร้างบางอย่างเช่นต้นไม้เพื่อบอกความสัมพันธ์ของคำต่างๆ
3. *การวิเคราะห์ทางความหมาย (semantic analysis)* เป็นการวิเคราะห์ความหมาย เมื่อได้โครงสร้างโดยการวิเคราะห์ทางวากยสัมพันธ์มาแล้ว ก็จะกำหนดค่าของคำแต่ละคำ ว่าหมายถึงสิ่งใด
4. *บูรณาการทางวาทะ (discourse integration)* เป็นการพิจารณาความหมายของประโยคโดยดูจากประโยคข้างเคียง เนื่องจากคำบางคำในประโยคหนึ่งๆ จะเข้าใจความหมายได้ต้องดูประโยคก่อนหน้าหรือประโยคตามด้วย
5. *การวิเคราะห์ทางปฏิบัติ (pragmatic analysis)* เป็นการแปลความหมายของประโยคใหม่อีกครั้งว่าที่จริงแล้วผู้พูดตั้งใจจะหมายความว่าอย่างไรหรือต้องการสื่อความหมายอะไร

ตัวอย่างการประมวลผลภาษาธรรมชาติ

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

- เริ่มจากการวิเคราะห์ทางองค์ประกอบโดยวิเคราะห์ในระดับคำ ในกรณีนี้วิเคราะห์ได้ว่า Bill's ประกอบด้วยหน่วยย่อยสองตัวคือ Bill และ 's และผลการวิเคราะห์คือ 's ทำหน้าที่เปลี่ยนคำนาม Bill ให้อยู่ในรูปของคำคุณศัพท์ (adjective) เช่นเดียวกับ .init ในโดเมนของคำสั่งในยูนิกซ์นี้ทำหน้าที่เป็นคำคุณศัพท์เช่นกัน
- จากนั้นจะทำการวิเคราะห์ทางวากยสัมพันธ์โดยการแปลงประโยคนี้ให้อยู่ในรูปของ **ต้นไม้แจงส่วน (parse tree)** เพื่อแสดงความสัมพันธ์ของคำแต่ละคำในประโยค เช่นคำใดทำหน้าที่ขยายคำอื่นอยู่ ตัวใดเป็นประธานหรือกรรมของกริยาที่สนใจเป็นต้น ผลที่ได้ของการวิเคราะห์ทางวากยสัมพันธ์ของประโยคด้านบนแสดงดังรูปที่ 2-8

ต้นไม้แจงส่วน



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

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

ประโยคเช่นแสดงให้รู้ว่าประโยคแบ่งออกเป็นสองส่วนคือ นามวลี (noun phrase) ซึ่งแทนด้วย NP ในรูปที่ 2-8 และกริยาวลี (verb phrase) ซึ่งแทนด้วย VP ในรูปที่ 2-8 นามวลีมีตัวเดียวคือสรรพนาม (pronoun) ส่วนกริยาวลีประกอบด้วยกริยา (verb) ซึ่งแทนด้วย V และอนุประโยคย่อย S (RM4) เป็นต้น และเมื่อเราดูจนครบทั้งต้นไม้ก็จะรู้โครงสร้างทางไวยากรณ์ของประโยคนี้

เมื่อวิเคราะห์ทางวากยสัมพันธ์แล้ว ในส่วนต่อไปเราต้องวิเคราะห์ทางความหมายเพื่อทำความเข้าใจว่าคำแต่ละคำหมายถึงสิ่งใดในโดเมนที่พิจารณาอยู่ อย่างเช่น Bill หมายถึงใคร 'init file' คือวัตถุใดในฐานความรู้เกี่ยวกับโดเมนของระบบยูนิกซ์ที่กำลังพิจารณาอยู่นี้เป็นต้น ดังนั้นการวิเคราะห์ทางความหมายนี้จะทำหน้าที่ผูกโครงสร้างที่ได้จากการวิเคราะห์ทางวากยสัมพันธ์เข้ากับวัตถุในฐานความรู้ พิจารณาฐานความรู้ในโดเมนนี้แสดงในรูปที่ 5-2 ต่อไปนี้ที่ใช้การแทนความรู้แบบกรอบ

User			
isa:	Person	F1	
*login-name:	must be <string>	instance:	File-Struct
		name:	stuff
User068		extension:	.init
instance:	User	owner:	User073
login-name:	Susan-Black	in-directory:	/wsmith/
User073			
instance:	User	File-Struct	
login-name:	Bill-Smith	isa:	Information-Object
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

การวิเคราะห์ทางความหมายจะผูกโครงสร้างต้นไม้ในรูปที่ 2-8 เข้ากับวัตถุในฐานความรู้ในรูปที่ 5-2 ผลการวิเคราะห์ทางความหมายแสดงในรูปที่ 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."

หลังจากนั้นทำบูรณาการวินิจฉัยพันธ์ได้ผลดังรูปที่ 5-4

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

รูปที่ 5-4 ผลของการวิเคราะห์ทางปฏิบัติ

ผลลัพธ์สุดท้ายที่ได้จากการวิเคราะห์ทางปฏิบัติคือคำสั่งในยูนิกซ์ที่ใช้สั่งยูนิกซ์พิมพ์ไฟล์ที่ต้องการดังด้านล่างนี้

lpr /wsmith/stuff.init

ในที่นี้เราจะกล่าวถึงขั้นตอนที่สำคัญอย่างหนึ่งในการประมวลผลภาษาธรรมชาติคือการวิเคราะห์ทางวากยสัมพันธ์ดังหัวข้อต่อไปนี้

5.2 การวิเคราะห์ทางวากยสัมพันธ์

การวิเคราะห์ทางวากยสัมพันธ์ (syntactic processing) เป็นการวิเคราะห์ประโยคทางไวยากรณ์เพื่อดูโครงสร้างและความสัมพันธ์ของคำในประโยค ช่วยให้การประมวลผลภาษาธรรมชาติในขั้นตอนต่อไปทำได้ง่ายขึ้นเช่น ช่วยให้การประมวลผลทางความหมายทำได้ง่ายขึ้น ซึ่งโดยปกติการประมวลผลทางความหมายมักใช้เวลานาน การประมวลผลทางวากยสัมพันธ์จะช่วยกรองให้มีตัวเลือกเกิดขึ้นน้อยลง

การประมวลผลทางวากยสัมพันธ์มีส่วนประกอบของกระบวนการ 2 ส่วนหลักๆ ได้แก่

ไวยากรณ์
คลังศัพท์
และ
ตัวแ่งส่วน

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

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

ไวยากรณ์
ไม่พึ่งบริบท

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

รูปที่ 5-5 ไวยากรณ์ไม่พึ่งบริบท 1

ไวยากรณ์นี้แสดงด้วยกฎทั้งหมด 5 ข้อ กฎแต่ละข้อจะอธิบายลักษณะการเขียนประโยคที่ต้องสำหรับโดเมนนี้ อย่างเช่นกฎข้อแรกของ S หมายความว่าประโยค (แทนด้วยสัญลักษณ์ S) ประกอบด้วยคำนามวลี (แทนด้วย NP) และต่อด้วยกริยาวลี (แทนด้วย VP) ส่วนกฎของคำนามวลีมีอยู่ด้วยกัน 2 ข้อ กฎข้อแรกของคำนามวลีหมายถึงคำนามวลีประกอบด้วยคำนำหน้านาม (article) และต่อด้วยคำนาม เป็นต้น หมวดคำบางตัว (เช่น NP ในไวยากรณ์ด้านบน) อาจประกอบด้วยกฎมากกว่าหนึ่งข้อก็ได้

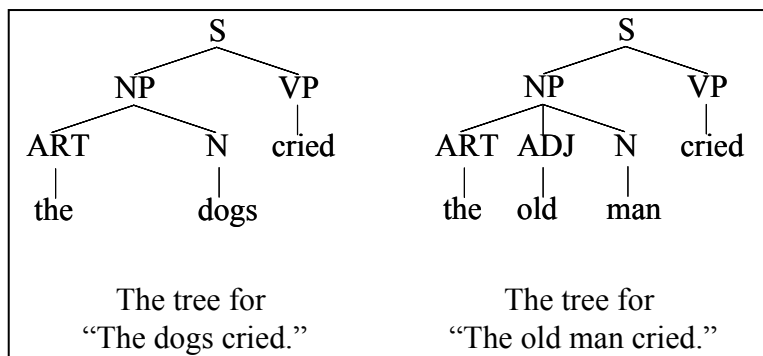
นอกจากไวยากรณ์แล้วเราจำเป็นต้องมีคลังศัพท์ (lexicon) ด้วยเพื่อบอก **หมวดคำ (category)** หรือหน้าที่ของคำที่เป็นไปได้ของคำแต่ละคำ เช่น

หมวดคำ

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

ตัวเจงส่วนจะทำหน้าที่รับประโยคอินพุต แล้วใช้ไวยากรณ์และคลังศัพท์เพื่อวิเคราะห์โครงสร้างของประโยคที่รับเข้ามา ตัวอย่างเช่นถ้าให้ประโยคอินพุตเป็น “The dogs cried.” หรือ “The old man cried.” จะได้ผลการวิเคราะห์ในรูปแบบของต้นไม้เจงส่วนดังแสดงในรูปที่ 5-6 ต่อไปนี้



รูปที่ 5-6 ตัวอย่างการวิเคราะห์ทางวากยสัมพันธ์

5.3 ตัวเจงส่วนแบบบนลงล่าง

ตัวเจงส่วนแบบบนลงล่าง (top-down parser) เป็นวิธีการหนึ่งสำหรับการวิเคราะห์ทางวากยสัมพันธ์ ซึ่งใช้กฎในไวยากรณ์และคลังศัพท์เพื่อค้นหาวิธีการทุกแบบที่สามารถจะสร้างต้นไม้เจงส่วน ต้นไม้เจงส่วนนี้มีคุณสมบัติคือสามารถอธิบายโครงสร้างของประโยคที่เข้ามา

ตัวเจงส่วนแบบบนลงล่างจะสร้างต้นไม้เจงส่วนจากบนลงล่างโดยเริ่มจาก S จากนั้นก็จะแตก S ออกเรื่อยๆ จนกระทั่งพบคำที่มาจากประโยคที่ผู้ใช้ป้อนเข้าไป หรือเขียน S เสียใหม่โดยใช้ด้านขวามือของ S เข้าไปแทนที่ S จากนั้นก็จะแทนที่ด้วยกฎเหล่านี้ไปเรื่อยๆ จนกระทั่งพบ

สัญลักษณ์ปลาย

สัญลักษณ์ปลาย (terminal symbol) ซึ่งหมายถึงสัญลักษณ์ที่เป็นตัวสุดท้ายที่อยู่ท้ายประโยคตรงกับคำในประโยค

เรามองกระบวนการแจงส่วนคือการค้นหา ในที่นี้เราแทนสถานะในการค้นหาให้อยู่ในรูปของคู่ลำดับที่ประกอบด้วยข้อมูลสองส่วนดังนี้

- รายการสัญลักษณ์: แสดงตัวกระทำที่ใช้จนถึงปัจจุบัน
- ตัวเลข: แสดงตำแหน่งปัจจุบันในประโยคที่จะทำการแจงส่วนต่อไป

เช่นกำหนดไวยากรณ์ให้ดังข้างต้น และให้ประโยค

“₁The ₂ dogs ₃ cried₄”

โดยที่ตัวเลขแสดงตำแหน่งในประโยค สมมติว่าเมื่อเราแจงส่วนไปได้ถึงตำแหน่งหนึ่งและได้สถานะดังนี้

((N VP) 2)

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

((VP) 3)

สัญลักษณ์ไม่ปลาย

ในกรณีนี้ “dogs” เป็นคำที่มีหมวดคำเป็นสัญลักษณ์ปลาย ในบางกรณีที่สัญลักษณ์ตัวแรกในรายการเป็น**สัญลักษณ์ไม่ปลาย (non-terminal symbol)** เราจะนำกฎที่มีด้านซ้ายมือตรงกับสัญลักษณ์นั้นในไวยากรณ์มาสร้างสถานะใหม่โดยแทนที่สัญลักษณ์ตัวแรกนั้นด้วยสัญลักษณ์ด้านขวามือของกฎนั้น เช่นจากสถานะ ((VP) 3) ด้านบนนี้ เราจะนำกฎข้อที่ 4 หรือข้อที่ 5 ในรูปที่ 5-5 มาสร้างสถานะใหม่ได้ดังนี้

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

อัลกอริทึมจะเก็บ**รายการสถานะที่เป็นไปได้ (possible state list)** ซึ่งประกอบด้วยสองส่วนหลักได้แก่

- สถานะปัจจุบัน: สถานะแรกในรายการสถานะที่เป็นไปได้
- สถานะสำรอง: สถานะอื่นที่เหลือในรายการสถานะที่เป็นไปได้

ตัวอย่างของรายการสถานะที่เป็นไปได้ เช่น

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

ประกอบด้วยสถานะ 3 ตัว ในกรณีนี้สถานะปัจจุบันคือ ((V) 3) ซึ่งจะถูกเลือกมาทำก่อน ถ้าทำสำเร็จก็หยุด แต่ถ้าไม่สำเร็จก็จะเลือกสถานะสำรองอีก 2 ตัวที่เหลือมาทำต่อ

อัลกอริทึมของตัวแจงส่วนแบบบนลงล่างอย่างง่าย

อัลกอริทึมของตัวแจงส่วนแบบบนลงล่างอย่างง่ายแสดงในตารางที่ 2-1

ตารางที่ 5-1 อัลกอริทึมของการแจงส่วนแบบบนลงล่างอย่างง่าย

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 จากนั้นถึงสถานะแรกออกจากรายการสถานะที่เป็นไปได้ เรียกสถานะนี้ว่า C ถ้ารายการนั้นว่างแสดงว่าการ

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

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

จากนั้นให้วนทำไปจนกระทั่งการแจงส่วนเสร็จสิ้น

ตัวอย่างที่ 1

การแจงส่วนของประโยคตัวอย่าง “₁The ₂dogs ₃cried₄” โดยวิธีการแจงส่วนแบบบนลงล่างอย่างง่ายแสดงในตารางที่ 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

ตัวอย่างนี้เป็นตัวอย่างที่มีขั้นตอนในการแจกส่วนที่ต้องใช้สถานะสำรอง ประโยคที่จะทำการแจกส่วนคือ “₁The ₂old ₃ man ₄cried₅” ขั้นตอนการแจกส่วนแสดงในตารางที่ 5-3

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

ขั้นตอน ที่	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 การแจกส่วนไม่สำเร็จ
11	((ART ADJ N) 4)	((ART ADJ N VP) 1)	การแจกส่วนไม่สำเร็จอีก
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)		การแจกส่วนสำเร็จ!

5.4 ตัวแจงส่วนตาราง

ตัวแจงส่วนตาราง (chart parser) เป็นตัวแจงส่วนที่ทำงานแบบล่างขึ้นบน โดยเริ่มจากคำก่อนแล้วนำคำหลายๆ คำมารวมกันเป็นวลีหรือหน่วยที่ใหญ่ขึ้น ถ้ามองจากต้นไม้มาก็คือเป็นการสร้างต้นไม้จากล่างขึ้นบน วิธีการแจงส่วนจากล่างขึ้นบนจะนำคำมาใส่หมวดคำของมันเข้าไปแล้วยุบรวมจากด้านขวาของกฎไปเป็นด้านซ้ายของกฎ เช่นเมื่อพบ the จะแทนด้วย ART พบ dogs แทนด้วย N จากนั้นก็เขียนแทน ART N ด้วย NP เป็นต้น

ตัวแจงส่วนตารางจะใช้ตารางมาช่วยให้การแจงส่วนทำงานได้อย่างรวดเร็ว มีประสิทธิภาพมากกว่าตัวแจงส่วนแบบบนลงล่าง ตัวแจงส่วนนี้ใช้แนวคิดเช่นเดียวกับการโปรแกรมแบบพลวัต (dynamic programming) ตารางนี้จะเก็บผลบางส่วน (partial result) ของการจับคู่ไว้ แล้วใช้กุญแจ (key) เพื่อหากฎที่ขึ้นต้นด้วยกุญแจนั้น หรือหากฎที่ได้ใช้ไปก่อนหน้านี้และต้องการกุญแจตัวนี้เพื่อขยายกฎหรือทำกฎให้สมบูรณ์

พิจารณาไวยากรณ์ดังต่อไปนี้

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

รูปที่ 5-7 ไวยากรณ์ไม่พึ่งบริบท 2

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

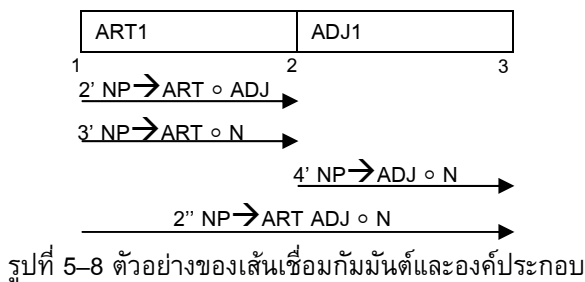
2' NP → ART ◦ ADJ N

3' NP → ART ◦ N

การจับคู่เพียงบางส่วนที่สร้างนี้แสดงให้เห็นว่า เมื่อเราพบ ART อย่างเดียวอาจจะเป็นไปได้ทั้งกฎข้อที่ 2 และ 3 เราจะใช้ \circ เป็นตัวทำเครื่องหมายว่ากฎข้อนี้ได้ถูกใช้ไปถึงจุดไหนแล้ว ในกรณีของกฎข้อ 2 (2' ด้านบน) แสดงถึงการแจງส่วนมาถึง ART แล้ว และถ้าสมมติว่าคำถัดไปมีหมวดคำแสดงด้วยสัญลักษณ์ ADJ เราจะทราบได้ทันทีว่า ADJ จะจับคู่ได้กับกฎข้อที่ 2 และจะสร้างการจับคู่บางส่วนขึ้นมาใหม่เป็น

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

เราเรียกการกระทำเช่นนี้ว่าการขยายกฎข้อที่สองออกไปและเราจะนำสัญลักษณ์ที่ได้ไปใส่ไว้ในตาราง สัญลักษณ์ที่เราเรียกอีกอย่างว่า **องค์ประกอบ (constituent)** ซึ่งเราจะเก็บองค์ประกอบเหล่านี้ไว้ในตารางทั้งหมด แล้วก็สร้าง**เส้นเชื่อมกัมมันต์ (active arc)** ซึ่งก็คือกฎที่สร้างไปแล้วบางส่วนแต่ยังไม่สมบูรณ์ (เช่น 2' 2'' 3' เหล่านี้คือเส้นเชื่อมกัมมันต์ทั้งสิ้น) ดังนั้นตารางจะประกอบด้วยสองส่วนคือเส้นเชื่อมกัมมันต์และองค์ประกอบ **รูปที่ 5-8** ด้านล่างนี้แสดงการแจງส่วนเมื่อทำไปจนถึงคำในตำแหน่งที่ 2



ตัวกระทำของการแจງส่วนตารางนี้จะทำการรวมเส้นเชื่อมกัมมันต์เข้ากับองค์ประกอบที่สมบูรณ์ ผลจากการรวมจะได้ (1) องค์ประกอบที่สมบูรณ์ตัวใหม่ หรือ (2) เส้นเชื่อมกัมมันต์ใหม่ที่เป็นการขยายจากเส้นเชื่อมเดิม และทุกครั้งที่เราได้องค์ประกอบตัวใหม่เราจะนำเส้นเชื่อมกัมมันต์มาขยายออกจนกระทั่งเป็นองค์ประกอบที่สมบูรณ์ (สามารถเขียนแทนด้านขวามือของกฎด้วยองค์ประกอบในด้านซ้ายมือของกฎได้) ถ้าได้องค์ประกอบที่สมบูรณ์ องค์ประกอบนี้จะถูกใส่ไว้ในรายการตัวหนึ่งที่เรียกว่า **อาเจนดา (agenda)** จากนั้นก็นำข้อมูลในอาเจนดาที่ได้ใส่เข้าไปในตาราง อัลกอริทึมของตัวแจງส่วนตารางแสดงใน**ตารางที่ 5-4**

ตารางที่ 5-4 อัลกอริทึมของตัวแจงส่วนตาราง

Algorithm: Chart Parsing**UNTIL** there is no input left **DO**1. **IF** the agenda is empty **THEN** look up the interpretations for the next word in the input and add them to the agenda.2. Select a constituent C from the agenda. Let C is from position p_1 to p_2 .3. **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 .4. Add C to the chart using the arc extension algorithm.

อัลกอริทึมจะวนจนกระทั่งไม่มีอินพุตเหลืออยู่ โดยอินพุตจะเป็นคำในประโยคที่ต้องการแจงส่วน อัลกอริทึมจะเริ่มจากตรวจสอบว่าอาเจเนดว่างหรือไม่ ถ้าว่างให้ไปดูหมวดคำสำหรับคำถัดไปในประโยคที่ป้อนเข้ามา นำหมวดคำที่ได้ใส่เข้าไปในอาเจเนดา (เช่น man เป็นได้ทั้ง V และ N ก็จะนำทั้ง V และ N ใส่เข้าไปในอาเจเนดา ต่อไปเราจะดึงองค์ประกอบเหล่านี้ออกมาประมวลผลทีละตัว) ให้ C เป็นองค์ประกอบที่เลือกออกมาจากอาเจเนดา ซึ่งอาจเป็น N หรือ V เป็นต้น กำหนดให้ C เริ่มจากตำแหน่ง p_1 ไป p_2 จากนั้นให้ไปดูกฎในไวยากรณ์ที่ด้านขวามือที่ตัวแรกสุดว่าขึ้นต้นด้วย C มีหรือไม่ ($X \leftarrow C X_1, \dots, X_n$) ถ้ามีให้สร้างเส้นเชื่อมกับมันที่แสดงว่าได้ประมวลผล C ไปเรียบร้อยแล้ว (จับคู่บางส่วนกับ C แล้ว) โดยใส่เครื่องหมาย \circ ไว้หลัง C เส้นเชื่อมนี้จะเริ่มจากตำแหน่ง p_1 ไป p_2 ตามคุณสมบัติของ C แล้วนำ 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

$$X \leftarrow X_1 \cdots \circ C \cdots X_n$$

from position p_0 to p_1 , add a new active arc

$$X \leftarrow X_1 \cdots C \circ \cdots X_n$$

from position p_0 to p_2 .

3. For any active arc of the form

$$X \leftarrow X_1 \cdots X_n \circ C$$

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 $X \leftarrow X_1 \cdots X_n \circ C$ เมื่อขยายเส้นเชื่อมนี้จะทำให้ได้องค์ประกอบที่สมบูรณ์ และจะเขียนเส้นเชื่อมนี้ใหม่ด้วยองค์ประกอบที่ด้านซ้ายของกฎนั้น (X) แล้วนำไปใส่ไว้ในอาเจนดาเพื่อประมวลผลต่อไป

ตัวอย่างของการแจงส่วนตาราง

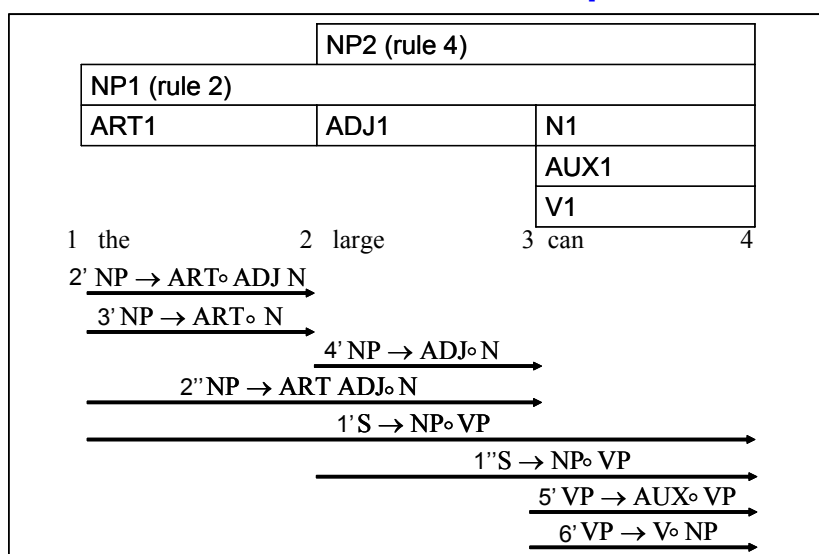
พิจารณาประโยค "The large can can hold the water" โดยใช้ไวยากรณ์ต่อไปนี้

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

และใช้คลังศัพท์ดังนี้

the: ART
 large: ADJ
 can: N, AUX, V
 hold: N, V
 water: N, V

ผลการแจกส่วนเมื่อทำการแจกส่วนจนถึงตำแหน่งที่ 3 แสดงในรูปที่ 5-9



รูปที่ 5-9 การแจกส่วนตารางจนถึง “the large can”

ขั้นตอนแรกอาเจณดา ยังคงว่างอยู่ (ไม่ได้แสดงอาเจณดาไว้ในรูป) เราดึงคำแรกจากประโยคขึ้นมาแล้วดูหมวดคำ ในที่นี้คือ the เป็น ART จึงนำ ART ใส่ไว้ในอาเจณดา ขั้นตอนที่สองดึง C ออกจากอาเจณดา (ในที่นี้ C คือ ART เพราะตอนนี้มีอยู่ตัวเดียวและอาเจณดาจะว่าง) จากนั้นไปดูกฎที่อยู่ในไวยากรณ์ว่ามีกฎข้อใดที่ด้านขวามือของกฎขึ้นต้นด้วย ART หรือไม่ พบว่ามีกฎอยู่สองข้อคือกฎที่ 2 และ 3 เราสร้างเส้นเชื่อมกัมมันต์โดยการประมวลผล ART และได้เส้นเชื่อม 2' และ 3' (ดูรูปที่ 5-9 ประกอบ) สังเกตว่าเส้นเชื่อมที่สร้างขึ้นจะมี ◦ เลื่อนไปอยู่หลัง ART

หลังจากนั้นนำ ART ใส่เข้าไปในตารางโดยใช้อัลกอริทึมขยายเส้นเชื่อม วิธีการจะเริ่มจากใส่ ART เข้าไปในตารางจากตำแหน่งที่ 1 ถึง 2 (ART1 ในรูปที่ 5-9) แล้วดูว่ามีเส้นเชื่อมกัมมันต์อื่นหรือไม่ที่กำลังรอรับ ART อยู่ ถ้ามีก็ให้ประมวลผล ART ณ ขณะนี้พบว่าจะไม่มีเส้นเชื่อมกัมมันต์ใดที่รอรับ ART อยู่ ขั้นตอนนี้จึงสิ้นสุดเท่านั้น

ณ จุดนี้อาเจณดาว่างอยู่ เราจึงดึงคำถัดไปคือ large เข้ามาและดูว่า large เป็นอะไรได้บ้างจากคลังศัพท์ พบว่าเป็น ADJ จึงนำ ADJ ใส่เข้าไปในอาเจณดา หลังจากนั้นก็ดึง ADJ ออกจากอาเจณดาและดูว่าในไวยากรณ์มี ADJ ปรากฏเป็นตัวแรกสุดทางขวามือของกฎหรือไม่ พบว่ามีปรากฏในกฎข้อที่ 4 เราจึงนำกฎข้อที่ 4 มาสร้างเป็นเส้นเชื่อมกัมมันต์โดยประมวลผล ADJ (นำ ๐ ไปไว้หลัง ADJ) ก็จะได้กฎข้อที่ 4' ดังในรูปที่ 5-9

หลังจากนั้นก็นำ ADJ ใส่เข้าไปในตารางโดยใช้อัลกอริทึมขยายเส้นเชื่อม โดยใส่เข้าไปในตำแหน่งที่ 2 และดูว่ามีเส้นเชื่อมกัมมันต์ใดที่กำลังรอรับ ADJ อยู่บ้าง พบว่ามีเส้นเชื่อม 2' เราจึงขยายเส้นเชื่อม 2' ให้ยาวขึ้นโดยเลื่อน ๐ ไปอยู่หลัง ADJ ในกฎ 2' เราจะได้กฎ 2'' เป็น NP→ART ADJ ๐ N ดังรูปที่ 5-9

ต่อมานำ can ไปหาหมวดคำในคลังศัพท์พบว่าเป็นได้ทั้งหมด 3 แบบคือ N, AUX (auxiliary verb (กริยานุเคราะห์)) และ V เรานำคำทั้งสามใส่ไว้ในอาเจณดา ณ จุดนี้เราจะเห็นประโยชน์ของอาเจณดาที่ทำหน้าที่เป็นหน่วยความจำชั่วคราวในการเก็บข้อมูลไว้ประมวลผลทีละตัว เราดึง N ออกจากอาเจณดาแล้วก็ไปดูในไวยากรณ์ว่ามีกฎข้อไหนที่ด้านขวามือขึ้นต้นด้วย N ก็จะพบว่าไม่มี จึงไม่ต้องสร้างเส้นเชื่อมกัมมันต์สำหรับ N ต่อมาเรานำ N เข้าไปใส่ไว้ในตาราง แล้วใช้อัลกอริทึมขยายเส้นเชื่อมตรวจดูว่ามีเส้นเชื่อมกัมมันต์ใดที่กำลังรอรับ N อยู่หรือไม่ พบว่ากฎข้อ 4' กำลังรอรับ N อยู่พอดี (4' NP→ADJ ๐ N) กรณีนี้เราไม่ต้องสร้างเส้นเชื่อมกัมมันต์เส้นใหม่ขึ้นอีก เพราะว่า N เป็นตัวสุดท้ายซึ่งเมื่อประมวลผลแล้วจะทำให้เส้นเชื่อมเป็นองค์ประกอบที่สมบูรณ์และมีตำแหน่งตั้งแต่ 2 จนถึง 4 ทำให้เราได้ NP และนำ NP ใส่เข้าไปในอาเจณดาเพื่อที่จะนำมันมาประมวลผลต่อไป กระบวนการนี้ยังไม่สิ้นสุดเพราะยังมีเส้นเชื่อม 2'' ที่ยังรอ N อยู่เช่นเดียวกัน ดังนั้นเมื่อเส้นเชื่อม 2'' รับ N เข้าไปก็จะเกิดองค์ประกอบสมบูรณ์ ซึ่งจะมีบริเวณตั้งแต่ 1 ถึง 4 จากนั้นก็นำ NP ใส่เข้าไปในอาเจณดาเช่นเดียวกัน (เรานำ NP ที่ได้จากเส้นเชื่อม 4' และที่ได้จากเส้นเชื่อม 2'' ไปเก็บไว้ในอาเจณดา)

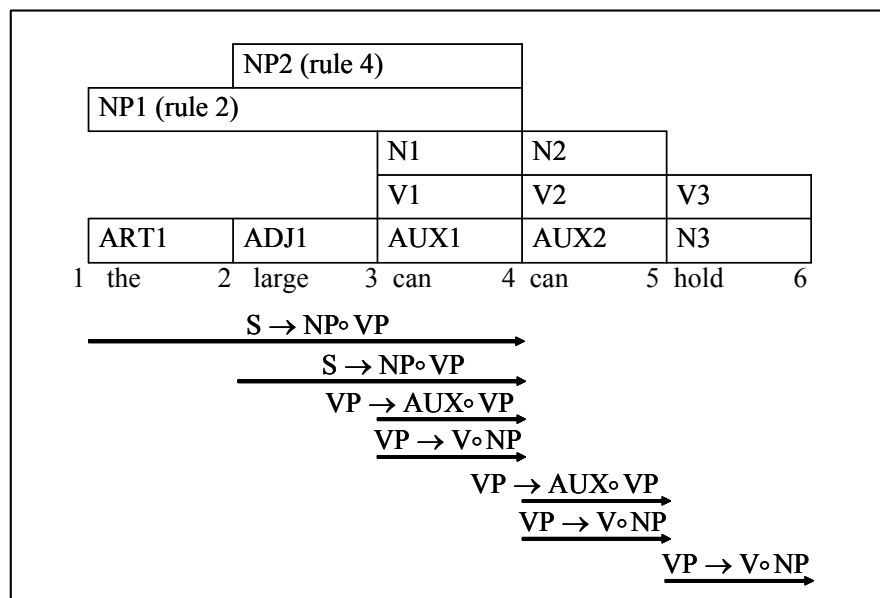
จากนั้นก็ดึง NP2 (ที่ได้จากกฎที่ 2'') มาประมวลผลและพบว่ามีความยาวตั้งแต่ตำแหน่งที่ 1 ถึง 4 จากนั้นก็ไปตรวจสอบว่ามีกฎข้อใดบ้างที่ขึ้นต้นด้วย NP ที่ด้านขวามือ พบว่ามีกฎข้อที่ 1 จึงนำมาสร้างเส้นเชื่อมกัมมันต์ของกฎข้อที่ 1 มีความยาวตั้งแต่ตำแหน่งที่ 1 ถึง 4 เพราะมาจาก NP2'' ที่ยาวตั้งแต่ตำแหน่งที่ 1-4 เราจึงได้เส้นเชื่อมกัมมันต์ 1' S→NP ๐ VP (ประมวลผล NP แล้ว กำลังรอรับ VP) ต่อมาก็นำ NP ใส่เข้าไปในตารางได้เป็น NP1 (NP ที่มาจากกฎข้อที่ 2) ส่วนอัลกอริทึมขยายเส้นเชื่อมไม่สามารถทำต่อได้อีก จึงสิ้นสุดเท่านั้น

ต่อมาถึง NP ตัวต่อมาออกจากอาเจเนตา (NP ที่มาจากกฎข้อที่ 4) และใส่เข้าไปในตารางแล้วดูว่ามีกฎข้อใดที่ด้านขวามือขึ้นต้นด้วย NP หรือไม่ ก็พบกฎข้อที่ 1 เหมือนเดิม ได้เส้นเชื่อมกับมันต์ 1" $S \rightarrow NP \circ VP$ ดูว่า ณ ตำแหน่งนี้มีเส้นเชื่อมใดที่รอรับ NP หรือไม่ พบว่าไม่มี จึงหยุดเท่านั้น

ต่อมาถึง AUX ออกจากอาเจเนตา ขั้นตอนเหมือนเดิมคือดูในไวยากรณ์ว่าตรงกับตัวแรกทางด้านขวามือของกฎข้อใดหรือไม่และพบว่าตรงกับกฎข้อที่ 5 ในไวยากรณ์ จึงได้เส้นเชื่อมกับมันต์ 5' $VP \rightarrow AUX \circ VP$ และนำ AUX ใส่เข้าไปในตาราง ณ ตำแหน่งนี้ไม่มีเส้นเชื่อมอื่นที่รอรับ AUX จึงสิ้นสุดเท่านั้น

ถึง V ออกจากอาเจเนตาไปดูในไวยากรณ์เช่นเดิม พบว่ามีกฎข้อ 6 ที่ด้านขวามือขึ้นต้นด้วย V จึงได้เส้นเชื่อมกับมันต์ 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 ข้างต้นไม่ได้เขียนเส้นเชื่อมกับมันต์ทุกตัวโดยเขียนเฉพาะเส้นเชื่อมที่เกี่ยวข้องเท่านั้น ขั้นตอนต่อไปคือการรับคำ can ตัวที่สองเพื่อมาประมวลผลต่อ ซึ่งจะเหมือนกับ can ตัวแรกคือมีหมวดคำเป็นไปได้ 3 แบบคือ N, AUX และ V นำทั้งหมดเก็บไว้

ในอาเจนดาแล้วถึง N (N2 ในรูป) ขึ้นมาทำก่อน นำ N ไปตรวจดูในกฎว่ามีกฎใดที่ทางขวามือขึ้นต้นด้วย N หรือไม่ พบว่าไม่มี นำ N ใส่เข้าไปในตารางและดูว่ามีเส้นเชื่อมใดรองรับ N หรือไม่ ก็ไม่มีอีก จบการประมวลผลสำหรับ N

นำ AUX (AUX2 ในรูป) มาทำต่อโดยดูว่าตรงกับกฎข้อใด พบว่ามีกฎข้อ 5 ดังนั้นสร้าง 5" VP → AUX ◦ VP จากนั้นนำ AUX ใส่เข้าไปในตารางและดูว่ามีเส้นเชื่อมใดรองรับ AUX หรือไม่ ปรากฏว่าไม่มี สิ้นสุด

ตัวต่อไปคือ V ถึง V ออกจากอาเจนดาใส่เข้าไปในตารางแล้วไปหากฎ พบกฎข้อ 6 จึงสร้าง 6" VP → V ◦ NP ดูว่ามีเส้นเชื่อมใดรองรับ V อยู่หรือไม่ พบว่าไม่มี สิ้นสุด

การประมวลผลตัวถัดไปคือ hold ก็ทำเช่นเดิม ดูในคลังศัพท์พบว่า hold เป็นได้ทั้ง N และ V นำทั้งคู่ไปใส่ไว้ในอาเจนดาแล้วถึง N ออกมาทำก่อน นำ N ไปใส่ไว้ในตารางแล้วเทียบกับกฎ พบว่าไม่มีกฎข้อใดที่ด้านขวาขึ้นต้นด้วย N จากนั้นทำการขยายเส้นเชื่อมและไม่พบเส้นเชื่อมกับมันต์ใดที่รองรับ N สิ้นสุด

ต่อไปถึง V ออกมา นำ V ใส่ในตารางแล้วไปหากฎในไวยากรณ์ พบว่าตรงกับกฎข้อ 6 จึงสร้าง 6" VP → V ◦ NP และดูว่าตำแหน่งนี้มีเส้นเชื่อมใดรองรับ 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)

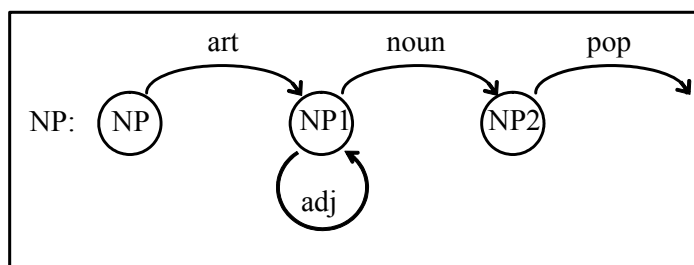
S1 (rule 1 with NP1 and VP2)						
S2 (rule 1 with NP2 and VP2)						
VP3 (rule 5 with AUX1 and VP2)						
NP2 (rule 4)			VP2 (rule 5)			
NP1 (rule 2)			VP1 (rule 6)			
		N1	N2	NP3 (rule 3)		
		V1	V2	V3	V4	
ART1	ADJ1	AUX1	AUX2	N3	ART2	N4

รูปที่ 5-11 ผลการแจงส่วนตารางจนครบประโยค "the large can can hold the water"

5.5 ไวยากรณ์ข่ายงานเปลี่ยนสถานะ

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

บัพแต่ละบัพแสดงสถานะของการแจกส่วนของประโยคและเส้นเชื่อมแต่ละเส้นแสดงองค์ประกอบหรือคำต่างๆ ข่ายงานนี้จะมีบัพอยู่หนึ่งบัพที่เป็นบัพเริ่มต้น จากนั้นจะรับอินพุตเข้ามาและท่องไปตามเส้นเชื่อม โดยเริ่มจากบัพเริ่มต้นและท่องไปที่ปลายทางของเส้นเชื่อมสำหรับคำนั้นไปยังบัพถัดไป ณ ขณะหนึ่งถ้ามาถึงสถานะสุดท้ายที่มีเส้นเชื่อม “pop” ก็แสดงว่าการแจกส่วนสมบูรณ์ **รูปที่ 5-12** แสดงตัวอย่างของไวยากรณ์ชนิดนี้และให้ชื่อว่า “ข่ายงานเปลี่ยนสถานะ 1”



รูปที่ 5-12 ข่ายงานเปลี่ยนสถานะ 1

ข่ายงานเปลี่ยนสถานะ 1 ข้างต้นนี้เป็นข่ายงานของนามวลี (NP) อธิบายว่า NP คืออะไร จากจุดเริ่มต้นที่สถานะ NP ถ้าพบ art ก็จะท่องไปที่สถานะ NP1 และถ้าพบ adj ก็ต้องมาอยู่ที่สถานะเดิม แต่ถ้าพบ noun ก็จะไปยังสถานะ NP2 และเจอเส้นเชื่อม pop ก็จบ ไวยากรณ์ข่ายงานสถานะ 1 นี้สมมูลกับไวยากรณ์ไม่พึงบริบทต่อไปนี้

NP → ART NP1

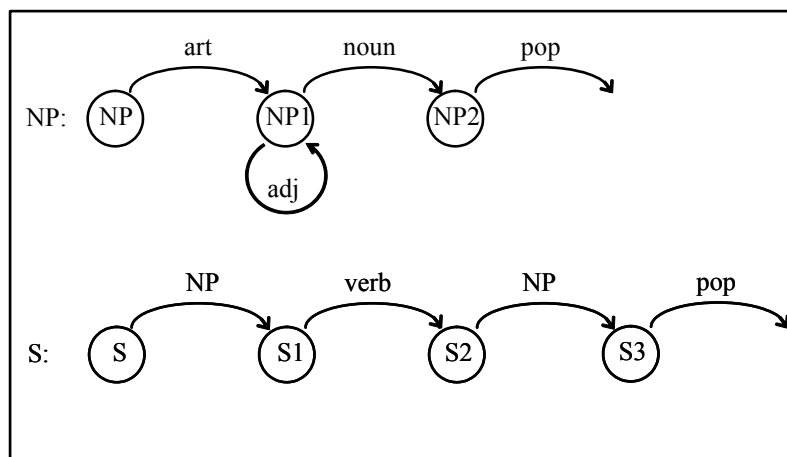
NP1 → ADJ NP1

NP1 → N

เราสามารถนำข่ายงานเปลี่ยนสถานะ 1 ข้างต้นไปแจกส่วนสำหรับนามวลีได้ เช่น “a purple cow” โดยเริ่มจากสถานะเริ่มต้น NP นำ “a” ไปตรวจดูในคลังศัพท์พบว่า “a” เป็น art ก็จะท่องไปที่สถานะ NP1 จากนั้นรับคำถัดไปคือ “purple” ซึ่งเป็น adj ก็ต้องกลับมาที่สถานะเดิม แล้วรับคำต่อไปคือ “cow” ซึ่งเป็น noun ก็เปลี่ยนไปที่สถานะ NP2 และพบเส้นเชื่อม pop ก็แสดงว่าการแจกส่วนสิ้นสุด

ข่ายงาน
เปลี่ยนสถานะ
เรียกซ้ำ
(อาร์ทีเอ็น)

อย่างไรก็ดีไวยากรณ์ข่ายงานเปลี่ยนสถานะนี้มีข้อจำกัดเมื่อนำไปแจ้งส่วนประโยคที่ซับซ้อนขึ้น จึงได้มีการขยายข่ายงานนี้ให้สามารถใช้งานได้กว้างขวางขึ้นเป็นไวยากรณ์ใหม่ ที่เรียกว่า **ข่ายงานเปลี่ยนสถานะเรียกซ้ำ – อาร์ทีเอ็น (Recursive Transition Network – RTN)** อาร์ทีเอ็นนี้สามารถอ้างถึงข่ายงานอื่นได้ เส้นเชื่อมในอาร์ทีเอ็นมีสองชนิดคือ (1) เส้นเชื่อมแบบเดิมที่เป็นหมวดคำ เช่น article, noun เป็นต้น และ (2) เส้นเชื่อมอ้างถึงข่ายงานอื่น และเพื่อแยกความแตกต่างระหว่างเส้นเชื่อมทั้งสองชนิดให้เห็นอย่างชัดเจน จึงใช้อักษรตัวเล็กแทนเส้นเชื่อมแบบเดิมและใช้อักษรใหญ่เขียนเส้นเชื่อมที่อ้างถึงข่ายงานอื่น ตัวอย่างของอาร์ทีเอ็นแสดงในรูปที่ 5-13



รูปที่ 5-13 ข่ายงานเปลี่ยนสถานะ 2 (อาร์ทีเอ็น)

ข่ายงานเปลี่ยนสถานะ 2 ในรูปด้านบนนี้ประกอบด้วยข่ายงานย่อย 2 ข่ายงานคือ ข่ายงานส่วนบนที่เป็นข่ายงานของ NP เหมือนกับตัวอย่างที่แล้ว ส่วนข่ายงานล่างเป็นของ S ซึ่งจะมีเส้นเชื่อมบางเส้นที่อ้างข่ายงาน NP (สังเกตจากตัวอักษรใหญ่) ไวยากรณ์นี้มีความหมายดังนี้คือ เริ่มต้นจากสถานะ S เราจะท่องไปยัง S1 ได้นั้น คำที่รับเข้ามาจะต้องประกอบด้วย NP ก่อน ซึ่ง NP คืออะไรนั้นก็จะต้องท่องไปที่ข่ายงาน NP ให้เรียบร้อยเสียก่อน เราสามารถใช้อาร์ทีเอ็นนี้แจ้งส่วนของประโยคที่ซับซ้อนขึ้นได้ เช่นประโยค “The purple cow ate the grass” เป็นต้น

อัลกอริทึมแจ้งส่วนบนลงล่างสำหรับอาร์ทีเอ็น

อัลกอริทึมที่จะกล่าวต่อไปนี้เป็นอัลกอริทึมสำหรับแจ้งส่วนแบบบนลงล่างโดยใช้ไวยากรณ์แบบอาร์ทีเอ็น อัลกอริทึมนี้จะเก็บสถานะของการแจ้งส่วนที่ประกอบด้วยข้อมูลต่อไปนี้

- บัพปัจจุบัน (current node): บัพที่การแจ้งส่วนอยู่ ณ ปัจจุบัน

- ตำแหน่งปัจจุบัน (current position): ตำแหน่งของคำถัดไปที่จะประมวลผล
- จุดกลับ (return points): กองซ้อน (stack) ของสถานะที่จะย้อนกลับมาเมื่อมีการอ้างอิงและต้องไปยังทำงานอื่นๆ แล้วพบเส้นเชื่อม pop

อัลกอริทึมแสดงในตารางที่ 5-6

ตารางที่ 5-6 อัลกอริทึมแจงส่วนแบบบนลงล่างสำหรับอาร์ทีเอ็น

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.

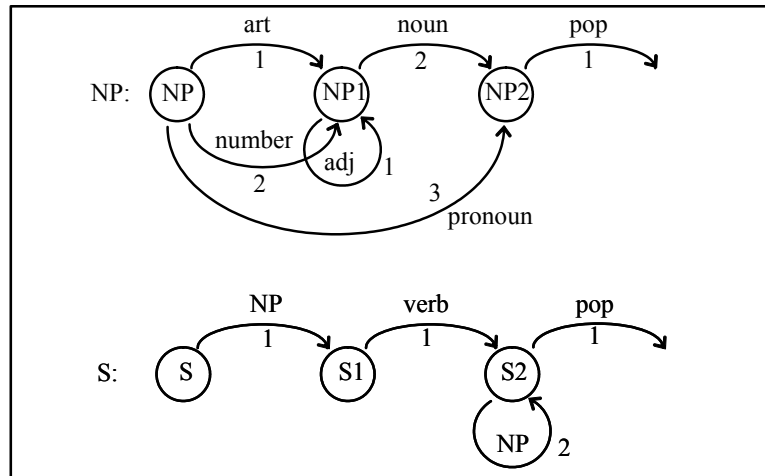
Case 4: IF arc is a pop arc, return points list is empty and there are no words left

THEN parse completes successfully.

อัลกอริทึมเริ่มจากสถานะเริ่มต้น เมื่ออยู่ที่สถานะปัจจุบันหนึ่งๆ อัลกอริทึมจะออกจากสถานะนั้นและท่องเส้นเชื่อมตามกรณีต่างๆ ดังนี้ (1) กรณีที่เส้นเชื่อมเป็นหมวดคำและคำถัดไปอยู่ในหมวดคำนั้น ก็ทำการปรับค่าของตำแหน่งปัจจุบันและบัพปัจจุบัน (2) กรณีที่เส้นเชื่อมอ้างอิงถึงทำงานอื่น ก็ให้ไปเริ่มที่สถานะเริ่มต้นในทำงานนั้น (3) กรณีที่เส้นเชื่อมเป็นเส้นเชื่อม pop และรายการจุดกลับไม่ว่าง ให้ไปยังสถานะที่กำหนดโดยจุดกลับตัวแรกในรายการ และ (4) กรณีที่เส้นเชื่อมเป็นเส้นเชื่อม pop และรายการจุดกลับว่างและไม่มีคำเหลือในประโยค ก็แสดงว่าการแจงส่วนสมบูรณ์

ตัวอย่างการแจงส่วนด้วยอาร์ทีเอ็น

พิจารณาข่ายงานเปลี่ยนสถานะ 3 ในรูปที่ 5-14 ด้านล่างนี้



รูปที่ 5-14 ข่ายงานเปลี่ยนสถานะ 3 (อาร์ทีเอ็น)

ผลการแจงส่วนของประโยค “₁The ₂old ₃man ₄cried₅” แสดงในตารางที่ 5-7ตารางที่ 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 NP/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 จะใช้ตัวเลขกำกับไว้ที่ด้านหลังของสถานะเพื่อแสดงการแจกส่วนว่าใช้เส้นเชื่อมใด ตัวอย่างเช่น NP/1 หมายถึงการใช้เส้นเชื่อมที่ 1 ของสถานะ NP

กำหนดให้คลังศัพท์เป็นดังต่อไปนี้

the: ART

old: ADJ, N

man: N

cried: V

การแจกส่วนเริ่มจากบัพแรกสุดคือ S ตำแหน่งปัจจุบันคือตำแหน่งที่ 1 และจุดกลับไม่มี เราท่องตามเส้นเชื่อมโดยเริ่มจาก NP ก่อน สังเกตว่า NP เป็น push arc หมายความว่าให้ไปยังตำแหน่งแรกในข่ายงาน NP โดยให้จำไว้ว่าจุดกลับคือปลายเส้นเชื่อมที่ 1 ของ NP ซึ่งก็คือบัพ S1 ขณะนี้เราอยู่ในข่ายงาน NP แล้วดูต่อว่า the เป็น art ได้หรือไม่ พบว่าได้ ก็ท่องไปที่ NP1 แล้วปรับค่าของตำแหน่งคำจาก 1 เป็น 2 บัพปัจจุบันอยู่ที่ NP1 และจุดกลับอยู่ที่ S1 เหมือนเดิม เตรียมรอรับคำต่อไป จากนั้นดูว่าตำแหน่งปัจจุบันคือ NP1 สามารถรับ adj ได้หรือไม่ พบว่าได้ ก็ประมวลผล old และกลับมายัง NP1 เหมือนเดิม (ตามรูป) พร้อมทั้งเลื่อนตำแหน่งปัจจุบันไปยังตำแหน่งที่ 3 จุดกลับยังเป็น S1 เช่นเดิม

คำต่อไปคือ man ซึ่งเป็น noun และที่บัพ NP1 สามารถรับ noun ได้ เราจึงท่องไปที่ NP2 เปลี่ยนตำแหน่งปัจจุบันไปยังตำแหน่งที่ 4 และจุดกลับคือ S1 เหมือนเดิม ขณะนี้เราอยู่ที่ NP2 ซึ่งมี pop arc ก็ท่องไปที่ pop arc และออกจากข่ายงาน NP กลับมายังจุดกลับ S1 ได้ตามขั้นตอนที่ 6 ในตารางและรอรับ V จากนั้นก็ดูว่าคำต่อไปคือ cried สามารถเป็น V ได้หรือไม่ พบว่า cried เป็น V ได้ก็ท่องไปที่ S2 ซึ่งเป็น pop arc และตำแหน่งปัจจุบันเป็นตำแหน่งสุดท้าย ได้ว่าแจกส่วนสำเร็จ

พิจารณาตัวอย่างอีกตัวอย่างหนึ่งของการแจกส่วนของประโยค "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		

การแจงส่วนของตัวอย่างที่แล้วไม่มีการทำย้อนรอยสามารถแจงส่วนรวดเดียวจบ ตัวอย่างนี้จะแสดงให้เห็นถึงการย้อนรอย

เริ่มต้นจาก S เหมือนเดิม และท่องโดยใช้ push arc ไปยังข่ายงาน NP เริ่มจากบัพ NP เพื่อรอรับคำต่อไป คำที่รับเข้าคือ one เป็นได้ทั้ง number และ pronoun ถ้าเลือก number ก็จะท่องไปตามเส้นเชื่อมที่ 2 ถ้าเลือก pronoun ก็จะท่องตามเส้นเชื่อมที่ 3 ครั้งแรกจะเลือก number ก่อนและท่องไปที่ NP1 และปรับค่าตำแหน่งปัจจุบันเป็นตำแหน่งที่ 2 แต่ถ้าเลือก pronoun จะท่องไปยัง NP2 (สถานะสำรองในตาราง) สถานะสำรองจะถูกเรียกมาทำก็ต่อเมื่อสถานะปัจจุบันไม่สามารถทำให้การแจงส่วนสมบูรณ์

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

ที่ NP2 ซึ่งไป pop arc กลับมาที่ S1 และตำแหน่งปัจจุบันเป็นตำแหน่งที่ 2 คำที่จะรับเข้ามาคือ saw ซึ่งเป็น verb ได้ เราจึงท่องต่อไปยัง S2 รอรับคำที่ 3 ซึ่งยังไม่จบประโยค ดังนั้นเราท่องตามเส้นเชื่อมที่ 2 เพื่อกลับไปยังขำยงาน NP คำที่ 3 คือ the ดังนั้นไปที่ NP1 รอรับคำในตำแหน่งที่ 4 ซึ่งเป็น noun ท่องไปยัง NP2 พบ pop arc กลับมาที่จุดกลับ S2 ซึ่งจบประโยคพอดี จึงท่องไปยัง pop arc สุดท้าย แสดงว่าการแจงส่วนสำเร็จ

เอกสารอ่านเพิ่มเติมและแบบฝึกหัด

หนังสือที่น่าสนใจสำหรับการประมวลภาษาธรรมชาติ คือ [Allen, 1995] ซึ่งอธิบายไวยากรณ์ชนิดต่างๆ การแจงส่วนวิธีการต่างๆ รวมถึงการวิเคราะห์ทางความหมาย ตลอดจนวิธีการอื่นๆ ในการประมวลผลภาษาธรรมชาติไว้อย่างละเอียด Manning และ Schütze เขียนหนังสือ [Manning & Schütze, 1999] ที่เกี่ยวกับการประมวลผลภาษาธรรมชาติโดยเทคนิคทางสถิติ ซึ่งอธิบายวิธีการใช้ประโยชน์จากคลังข้อความ (corpus) มาช่วยให้การประมวลผลภาษาธรรมชาติมีประสิทธิภาพ หนังสือของ Jurafsky และ Martin [Jurafsky & Martin, 2000] อธิบายถึงเทคนิคในการประมวลผลภาษาธรรมชาติและการประมวลผลทางเสียงด้วย

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

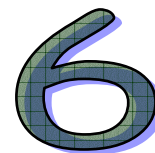
can : N, AUX, V

do : V, AUX

จงแสดงขั้นตอนการแจกแจงส่วนโดยใช้ตัวแจกแจงส่วนตารางสำหรับประโยค "The old man can do."

2. จงเขียนอาร์ทีเอ็นที่สมมูลกับไวยากรณ์ในข้อ 1.
 3. จงใช้อาร์ทีเอ็นที่ได้จากข้อ 2. แล้วทำการแจกแจงส่วนของประโยค "The old man can do." ด้วยอัลกอริทึมแจกแจงส่วนแบบบนลงล่างสำหรับอาร์ทีเอ็น
-

การเรียนรู้ของเครื่อง



บทนี้กล่าวถึงการเรียนรู้ของเครื่อง (machine learning) ซึ่งเทคนิคการเรียนรู้ส่วนมากเป็น การเรียนรู้เชิงอุปนัย (inductive learning) และมีบางเทคนิคเป็น การเรียนรู้เชิงวิเคราะห์ (analytical learning) การเรียนรู้เชิงอุปนัยคือการเรียนรู้ที่หากฎเกณฑ์หรือความรู้ที่แฝงอยู่ในชุดตัวอย่างสอน (training example set) เพื่อเรียนรู้ให้ได้ความรู้ใหม่ที่สอดคล้องกับชุดตัวอย่างสอน ส่วนการเรียนรู้เชิงวิเคราะห์เป็นการจัดรูปแบบของความรู้ใหม่เพื่อให้ใช้งานได้ อย่างมีประสิทธิภาพมากขึ้น ทำงานได้เร็วขึ้น

6.1 ขั้นตอนวิธีเชิงพันธุกรรม

ขั้นตอนวิธีเชิงพันธุกรรม – จีเอ (Genetic Algorithm – GA) [Goldberg, 1989; Mitchell, 1996] เป็นการเรียนรู้ที่จำลองการวิวัฒนาการ เราอาจมองได้ว่าจีเอเป็นกระบวนการค้นหาประเภทหนึ่งหรืออาจมองว่าจีเอเป็นการเรียนรู้ของเครื่องประเภทหนึ่งก็ได้ จีเอได้ถูกขยายขึ้นเป็นการโปรแกรมเชิงพันธุกรรม – จีพี (Genetic Programming – GP) [Koza, 1992] ข้อแตกต่างที่สำคัญอย่างหนึ่งระหว่างจีเอกับจีพีก็คือในจีเอสิ่งที่เรียนรู้ได้เป็นสายอักขระความยาวคงที่ (fixed-length string) ส่วนในจีพีจะได้สายอักขระความยาวแปรได้ (variable-length string) ซึ่งมักแสดงในรูปของโปรแกรมภาษา LISP

แนวคิดของจีเอมาจากทฤษฎีวิวัฒนาการของสิ่งมีชีวิต เช่นการไขว่เปลี่ยนของโครโมโซม (chromosome crossover) การกลายพันธุ์ของยีน (gene mutation) การวิวัฒนาการของสิ่งมีชีวิต เป็นต้น จีเอสามารถจัดการกับปัญหาค่าดีที่สุดเฉพาะที่ (local optimum) ในการค้นหาได้ การค้นหาทั่วไปจะมองว่าจุดดีที่สุดเฉพาะที่เป็นกับดักและจะหลีกเลี่ยงกับดักโดยใช้วิธีต่างๆ เช่น การย้อนรอย (backtracking) หรือการค้นหาแบบขนาน (parallel search) โดยใช้สถานะเริ่มต้นที่ต่างๆ กัน เป็นต้น แต่เทคนิคการค้นหาด้วยจีเอจะใช้วิธีการที่แตกต่างไปดังจะกล่าวต่อไป

โครโมโซมกำหนดลักษณะพิเศษที่สืบทอดได้

การไขว้เปลี่ยน

การกลายพันธุ์

เซลล์แต่ละเซลล์ในพืชชั้นสูงและสัตว์ประกอบด้วยนิวเคลียส 1 นิวเคลียส และนิวเคลียสหนึ่งๆ ประกอบด้วยโครโมโซมจำนวนมาก โครโมโซมจะอยู่กันเป็นคู่ๆ โดยได้รับมาจากพ่อและแม่อย่างละ 1 เส้น โครโมโซมแต่ละเส้นจะมียีนเป็นตัวกำหนดลักษณะพิเศษของสิ่งมีชีวิต ในขณะที่มีการจับคู่กันของโครโมโซมอาจเกิด **การไขว้เปลี่ยน (crossover)** ซึ่งเป็นการที่ยีนจากโครโมโซมพ่อแม่สลับเปลี่ยนกันทำให้เกิดโครโมโซมใหม่ขึ้น 2 คู่ และในขณะที่เซลล์แบ่งตัวจะเกิดกระบวนการ **คัดลอกโครโมโซม (chromosome copying)** ซึ่งบางครั้งจะมีการเปลี่ยนแปลงของยีนที่มาจากยีนพ่อและแม่เกิดเป็นยีนที่ไม่เคยมีมาก่อน เราเรียกการเกิดยีนลักษณะนี้ว่า **การกลายพันธุ์ (mutation)**

ชาร์ลส์ ดาร์วิน (Charles Darwin) ได้อธิบายการสืบทอดของสิ่งมีชีวิตด้วยกฎที่เรียกว่า **การวิวัฒนาการโดยผ่านการคัดเลือกตามธรรมชาติ (evolution through natural selection)** ไว้ว่าสิ่งมีชีวิตมีแนวโน้มที่จะสืบทอดลักษณะพิเศษให้ลูกหลานและธรรมชาติจะผลิตสิ่งมีชีวิตที่มีลักษณะพิเศษแตกต่างไปจากเดิม สิ่งมีชีวิตที่เหมาะสมที่สุด (fittest) ก็คือสิ่งมีชีวิตที่มีลักษณะพิเศษที่ธรรมชาติพอใจมากที่สุดจะมีแนวโน้มที่มีลูกหลานมากกว่าตัวที่ไม่เหมาะสม ดังนั้นประชากรจะโน้มเอียงไปทางตัวที่เหมาะสม เมื่อช่วงเวลาผ่านไปนานๆ การเปลี่ยนแปลงจะสะสมไปเรื่อยๆ และเกิดสปีชีส์ (species) ใหม่ที่เหมาะสมกับสภาพแวดล้อม ดังนั้นเราอาจกล่าวได้ว่าการคัดเลือกโดยธรรมชาติเกิดจากการเปลี่ยนแปลงที่เป็นผลของการไขว้เปลี่ยนและการกลายพันธุ์

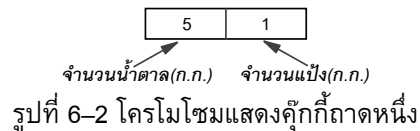
6.1.1 การออกแบบขั้นตอนวิธีเชิงพันธุกรรม

จะยกตัวอย่างปัญหาการทำคูกี้เพื่ออธิบายการออกแบบจีเอ [Winston, 1992] สมมติว่าเราต้องการหาส่วนผสมที่ดีที่สุดเพื่อทำคูกี้โดยที่คูกี้ก็มีส่วนผสมสองอย่างคือแป้งและน้ำตาล และสมมติว่าคุณภาพของคูกี้ก็เป็นฟังก์ชันแสดงใน **รูปที่ 2-8** ด้านล่างนี้

น้ำตาล	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
		1	2	3	4	5	6	7	8	9
		แป้ง								

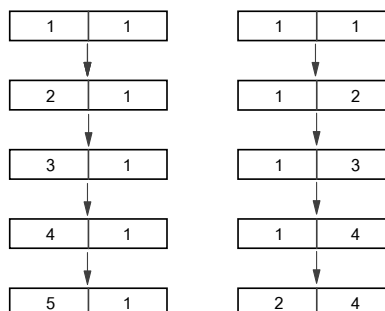
รูปที่ 6-1 ฟังก์ชันภูเขาเรียบของคุณภาพคูกี้

แนวตั้งและแนวนอนแสดงจำนวนกิโกรัมของส่วนผสมน้ำตาลและแป้งตามลำดับ เช่น น้ำตาล 1 กก. กับแป้ง 1 กก. ผลิตได้คูกี้ที่มีคุณภาพ 1 หน่วย ฟังก์ชันนี้จะมีค่าสูงสุดอยู่ที่ 5-5 (น้ำตาล 5 กก. กับแป้ง 5 กก. ผลิตได้คูกี้ที่มีคุณภาพ 9 หน่วย) เรากลับแบบให้แต่ละแถวของคูกี้ที่ถูกแทนด้วยโครโมโซมเส้นหนึ่งดังแสดงในรูปที่ 1-1



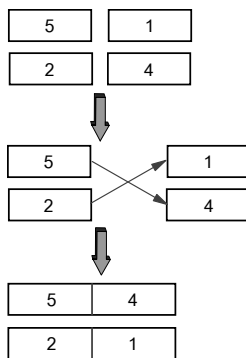
ในการออกแบบครั้งนี้กำหนดให้โครโมโซมมียีน 2 ตัว ยีนด้านซ้ายแทนจำนวนกิโกรัมของน้ำตาลและยีนด้านขวาแทนจำนวนกิโกรัมของแป้ง กำหนดให้ตัวเลขแสดงจำนวนกิโกรัมของทั้งน้ำตาลและแป้งมีค่าตั้งแต่ 1 ถึง 9 โครโมโซมแทนแถวของคูกี้ที่กำหนดความเหมาะสม (fitness) กับธรรมชาติของคูกี้ โครโมโซมสามารถสร้างขึ้นจากยีนน้ำตาลและแป้ง สร้างขึ้นจากการไขว้เปลี่ยนของโครโมโซมพ่อแม่คู่หนึ่ง หรือสร้างได้จากการกลายพันธุ์ของยีนในโครโมโซมตัวหนึ่งที่มีอยู่ และถ้าหากเรามีโครโมโซม 1 เส้น เราสามารถตัดแบ่งเอายีนของน้ำตาลหรือยีนของแป้งได้

ในการทำจีเอครั้งนี้ กำหนดให้ประชากรรุ่นหนึ่งๆ มีโครโมโซมที่เหมือนกันเพียงเส้นเดียว เราจำลองการเกิดการกลายพันธุ์ของโครโมโซมโดยการเลือกยีนตัวหนึ่งแบบสุ่มแล้วเปลี่ยนค่าของยีนโดยบวกหนึ่งหรือลบหนึ่งแบบสุ่มและยอมรับค่าที่ได้ถ้าค่านั้นอยู่ระหว่าง 1 ถึง 9 รูปที่ 6-3 แสดงวิวัฒนาการของโครโมโซมโดยการกลายพันธุ์ ในรูปแสดงการกลายพันธุ์สองรูปแบบซึ่งในแต่ละแบบแสดงการกลายพันธุ์เมื่อผ่านไป 4 ครั้ง ในแต่ละครั้งยีนที่เลือกและค่าที่เปลี่ยนไปเกิดจากการสุ่มในครั้งนั้นๆ เราเห็นได้ว่าเมื่อผ่านการกลายพันธุ์ไป 4 ครั้งโครโมโซมที่ได้มีความต่างกันค่อนข้างมาก โครโมโซมเส้นที่ดีเหมาะกับธรรมชาติก็จะถูกคัดเลือกซึ่งจะกล่าวต่อไป



รูปที่ 6-3 การจำลองการกลายพันธุ์ของโครโมโซมคูกี้

เราจำลองการไขว้เปลี่ยนของโครโมโซมโดยตัดที่กึ่งกลางของโครโมโซมพ่อแม่ 2 เส้น แล้วนำแต่ละส่วนมาต่อกัน ดังรูปที่ 4-1



รูปที่ 6-4 การจำลองการไขว้เปลี่ยนของโครโมโซมคู่ก็

จากรูปเราจะเห็นได้ว่าโครโมโซมพ่อแม่ 5-1 และ 2-4 ผลิตได้โครโมโซมลูกสองเส้นคือ 5-4 กับ 2-1 ในกรณีทั่วไปที่โครโมโซมประกอบด้วยยีนมากกว่า 2 ตัว การตัดและการต่อจะซับซ้อนยิ่งขึ้น

เมื่อพิจารณาปริภูมิก้นหาในบทที่ 2 จะพบว่าการกลายพันธุ์มีลักษณะเทียบเคียงได้กับตัวกระทำการในปริภูมิก้นหา มีหน้าที่สร้างสถานะ (โครโมโซม) ลูกของสถานะปัจจุบัน อย่างไรก็ตามการกลายพันธุ์มีความแตกต่างอยู่ที่ลักษณะสำคัญของวิธีการจีเอซึ่งใช้ความน่าจะเป็นในกระบวนการค้นหาก้าวคือการกลายพันธุ์จะสร้างโครโมโซมโดยการสุ่ม และเมื่อเราพิจารณาการไขว้เปลี่ยนจะไม่พบตัวกระทำการในปริภูมิก้นหาที่มีการทำงานในลักษณะเช่นนี้ กล่าวได้ว่าการไขว้เปลี่ยนเป็นคุณสมบัติเฉพาะของจีเอ เปรียบเสมือนการกระโดดไปยังสถานะใหม่ 2 ตัวจากสถานะพ่อแม่ 1 คู่ ซึ่งข้อดีของการไขว้เปลี่ยนจะได้กล่าวต่อไป

6.1.2 ค่าความเหมาะสมมาตรฐาน

ค่าความเหมาะสม (fitness) ของโครโมโซมคือความน่าจะเป็นที่โครโมโซมจะอยู่รอดในรุ่น (generation) ถัดไป ค่าความเหมาะสมมาตรฐานสามารถนิยามได้ดังนี้

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

โดยที่ f_i คือค่าความเหมาะสมของโครโมโซมเส้นที่ i ซึ่งมีค่าระหว่าง 0 ถึง 1 และ q_i คือคุณภาพของคู่ก็ที่ถูกกำหนดโดยโครโมโซมเส้นที่ i

ตัวอย่างเช่น สมมติว่าประชากรประกอบด้วยโครโมโซม 4 เส้นคือ 1-4, 3-1, 1-2, 1-1 ค่าความเหมาะสมของโครโมโซมแต่ละเส้นแสดงได้ในตารางที่ 6-1

ตารางที่ 6-1 ตัวอย่างค่าความเหมาะสมมาตรฐานของโครโมโซมคู่ก็

โครโมโซม	คุณภาพ	ค่าความเหมาะสมมาตรฐาน
1-4	4	0.40
3-1	3	0.30
1-2	2	0.20
1-1	1	0.10

ค่าความเหมาะสมมาตรฐานที่คำนวณได้ในตารางนี้ (เช่นค่าความเหมาะสมของโครโมโซม 1-4 จะเท่ากับ $4/(4+3+2+1)=0.40$) เป็นความน่าจะเป็นที่โครโมโซมจะอยู่รอด (ถูกเลือก) ในรุ่นถัดไป ดังนั้นโครโมโซม 1-4 จะมีโอกาสอยู่รอดมากกว่าโครโมโซมเส้นอื่นๆ และมีโอกาสอยู่รอดมากกว่าโครโมโซม 1-1 ถึง 4 เท่า แต่ก็ไม่ได้หมายความว่าถ้าให้เลือกโครโมโซมได้แค่เส้นเดียวแล้วโครโมโซม 1-4 ที่มีค่าความเหมาะสมสูงสุดจะถูกเลือกทุกครั้งไป แต่จะขึ้นอยู่กับ การสุ่มค่า แน่นอนว่า 1-4 มีโอกาสมากที่สุด และถ้าการสุ่มทำได้อย่างไม่เอนเอียงในการสุ่ม 100 ครั้ง 1-4 น่าจะมีโอกาสถูกเลือกสัก 40 ครั้ง

6.1.3 การจำลองการคัดเลือกโดยธรรมชาติ

เราได้ออกแบบโครโมโซมสำหรับปัญหาที่เราสนใจ การกลายพันธุ์ การไขว้เปลี่ยน ค่าความเหมาะสมแล้ว หัวข้อนี้จะกล่าวถึงการจำลองการคัดเลือกโดยธรรมชาติซึ่งสามารถทำได้โดยใช้ขั้นตอนทั่วไปดังต่อไปนี้

- กำหนดประชากรเริ่มต้น อาจมีโครโมโซม 1 เส้นหรือหลายเส้นก็ได้ เราอาจสุ่มโครโมโซมเหล่านี้หรือกำหนดขึ้นเองก็ได้
- ทำการกลายพันธุ์ยีนในโครโมโซมในรุ่นปัจจุบันและผลิตลูก
- ทำการไขว้เปลี่ยนโครโมโซม (พ่อแม่) ในรุ่นปัจจุบันและผลิตลูก
- เพิ่มลูกที่เกิดใหม่ในประชากร
- สร้างประชากรรุ่นใหม่โดยเลือกโครโมโซมตามค่าความเหมาะสมอย่างสุ่ม

ในการแก้ปัญหาหนึ่งๆ ที่เราสนใจด้วยวิธีนั้น เราจำเป็นต้องกำหนดพารามิเตอร์ต่างๆ ในการจำลองการคัดเลือกโดยธรรมชาติ อย่างเช่นในประชากรรุ่นหนึ่งๆ ควรมีโครโมโซมจำนวนเท่าไร? ถ้าน้อยไปก็มีแนวโน้มว่าโครโมโซมในประชากรรุ่นหนึ่งๆ จะมีลักษณะ

คล้ายกันหรือเหมือนกันเกือบทั้งหมดและการทำการไขว้เปลี่ยนก็จะมีผลมากนัก แต่ถ้ามากไปเราก็จะเสียเวลาคำนวณมาก อัตราการกลายพันธุ์เป็นเท่าไร? ถ้าต่ำไปลักษณะใหม่จะเกิดช้า ถ้าสูงไปประชากรรุ่นใหม่จะไม่เกี่ยวเนื่องกับรุ่นเดิม จะทำการไขว้เปลี่ยนด้วยหรือไม่? ถ้าทำจะเลือกคู่ผสมอย่างไร? และการไขว้เปลี่ยนกำหนดอย่างไร? ตัดครึ่งตรงกึ่งกลางหรือสุ่มจุดตัด เป็นต้น โครโมโซมเหมือนกันจะยอมให้มีหลายเส้นหรือไม่?

ในปัญหาการหาส่วนผสมที่ดีที่สุดของคูกี้ก็นี้ เราจะจำลองการคัดเลือกโดยธรรมชาติดังนี้

- เริ่มจากโครโมโซม 1-1 เพียงเส้นเดียว
- โครโมโซมที่เหมือนกันจะมีแค่เส้นเดียวในประชากรรุ่นหนึ่งๆ
- โครโมโซม 4 เส้นหรือน้อยกว่าจะอยู่รอดไปถึงรุ่นใหม่
- สำหรับโครโมโซมแต่ละเส้นที่อยู่รอด เลือกยีนตัวหนึ่งแบบสุ่มเพื่อทำการกลายพันธุ์ ถ้าโครโมโซมที่ได้จากการกลายพันธุ์ยังไม่เคยมีมาเลยให้เพิ่มเข้าไปในประชากร
- ไม่ทำการไขว้เปลี่ยน
- โครโมโซมที่อยู่รอดจะแข่งขันกับโครโมโซมใหม่เพื่อกำหนดโครโมโซมที่จะอยู่ในรุ่นถัดไป โครโมโซมที่มีค่าความเหมาะสมสูงสุดจะถูกเลือกเสมอให้อยู่รอดไปถึงรุ่นถัดไป ส่วนเส้นที่อยู่รอดที่เหลือจะถูกเลือกจากโครโมโซมที่เหลือแบบสุ่มตามค่าความเหมาะสม

จากการทดลอง 1,000 ครั้งโดยใช้การคัดเลือกโดยธรรมชาติด้านบน พบว่าส่วนผสมที่ทำให้คุณภาพของคูกี้ที่ดีที่สุดถูกผลิตในรุ่นที่ 16 โดยเฉลี่ย และในจำนวนนี้การทดลองที่โชคดีที่สุดผลิตโครโมโซมที่ดีที่สุดที่รุ่นที่ 8 ดังแสดงในตารางที่ 6-2 ด้านล่างนี้

ตารางที่ 6-2 ผลการทดลองดีที่สุดผลิตโครโมโซมดีที่สุดได้ในรุ่นที่ 8 โดยค่าความเหมาะสมมาตรฐาน

รุ่นที่ 0:		• 1-1 กลายพันธุ์เป็น 1-2
โครโมโซม	คุณภาพ	
1-1	1	
รุ่นที่ 1:		• 1-2 กลายพันธุ์เป็น 1-3 และ 1-1 เป็น 1-2 ซึ่งมีอยู่แล้ว
โครโมโซม	คุณภาพ	
1-2	2	
1-1	1	

รุ่นที่ 2: โครโมโซม		คุณภาพ 3 2 1	• 1-3 กลายพันธุ์เป็น 1-4, 1-2 เป็น 2-2, 1-1 เป็น 2-1 โครโมโซมทั้งหมดมี 6 เส้น และ 4 เส้นถูกเลือกตั้งแสดงในรุ่นที่ 3 (1-4 ที่มีค่าความเหมาะสมสูงสุดถูกเลือกเลย ส่วนอีกสามเส้นที่เหลือได้จากการสุ่มตามค่าความเหมาะสม สังเกตว่าแม้ว่า 2-2 จะมีค่าความเหมาะสมดีกว่า 1-2 และ 2-1 แต่ไม่ถูกเลือกในครั้งนี้)	
1-3				
1-2				
1-1				
รุ่นที่ 3: โครโมโซม		คุณภาพ 4 3 2 2	• การกลายพันธุ์ผลิตได้โครโมโซมใหม่ 3 เส้นดังต่อไปนี้	
1-4			โครโมโซม	คุณภาพ
1-3			2-4	5
1-2			2-3	4
2-1			3-1	3
รุ่นที่ 4: โครโมโซม		คุณภาพ 5 4 3 2	• โครโมโซมทุกเส้นกลายพันธุ์และผลิตลูก	
2-4				
1-4				
1-3				
2-1				
รุ่นที่ 5: โครโมโซม		คุณภาพ 6 5 4 3	• โครโมโซมทุกเส้นกลายพันธุ์และผลิตลูก	
2-5				
1-5				
2-3				
2-2				
รุ่นที่ 6: โครโมโซม		คุณภาพ 7 5 4 4	• 3-5 กลายพันธุ์เป็น 4-5, 3-2 เป็น 3-1, 1-4 เป็น 1-5, 1-5 เป็น 1-4 จะเห็นได้ว่าการผลิตลูกมักมีการซ้ำซ้อนของโครโมโซม เช่นไปซ้ำเดิมกับพ่อแม่เป็นต้น	
3-5				
1-5				
3-2				
1-4				

รุ่นที่ 7:		คุณภาพ	● ที่จุดนี้ 4-5 กลายพันธุ์เป็น 5-5 ซึ่งเป็นคำตอบในที่สุด
โครโมโซม			
4-5		8	
1-5		5	
1-4		4	
3-1		3	
รุ่นที่ 8:		คุณภาพ	
โครโมโซม			
5-5		9	
4-5		8	
2-5		6	
2-1		2	

6.1.4 การไขว้เปลี่ยนเพื่อหาจุดดีที่สุดเฉพาะที่

หัวข้อนี้เราจะดูผลของการไขว้เปลี่ยนที่มีต่อจีเอ โดยทำการทดลองเหมือนการทดลองที่แล้ว แต่เพิ่มการไขว้เปลี่ยนเพื่อสร้างโครโมโซมใหม่ด้วย การไขว้เปลี่ยนทำดังต่อไปนี้

- ทำการไขว้เปลี่ยนโดยใช้โครโมโซมที่อยู่รอดจากรุ่นที่แล้ว (อย่างมากสุด 4 เส้น)
- สำหรับโครโมโซมที่จะทำการไขว้เปลี่ยนเส้นหนึ่งๆ ให้เลือกคู่ทำการไขว้เปลี่ยนแบบสุ่ม
- สลับยีนของโครโมโซมพ่อแม่และผลิตโครโมโซมลูก 2 เส้น ถ้าโครโมโซมลูกยังไม่เคยมีมาเลยให้เพิ่มเข้าไปในประชากรเพื่อแข่งขันที่จะอยู่รอดในรุ่นถัดไป

ผลจากผลการทดลอง 1,000 ครั้ง ส่วนผสมที่ดีที่สุดถูกผลิตในรุ่นที่ 14 โดยเฉลี่ย ใช้จำนวนรุ่นน้อยกว่ากรณีไม่ใช้การไขว้เปลี่ยน 2 รุ่น แม้ว่าการไขว้เปลี่ยนจะช่วยให้เราพบส่วนผสมที่ดีที่สุดโดยใช้จำนวนรุ่นน้อยกว่าเดิม แต่เราต้องใช้เวลาคำนวณในแต่ละรุ่นมากขึ้นกว่าเดิมเนื่องจากจำนวนโครโมโซมที่มากขึ้นและการคำนวณค่าความเหมาะสมที่เพิ่มขึ้น ดังนั้นเวลาโดยรวมจะเพิ่มขึ้นกว่าเดิม สำหรับปัญหานี้เป็นปัญหาที่ไม่มีจุดดีที่สุดเฉพาะที่มีแค่จุดดีที่สุดกว้างจุดเดียว ประสิทธิภาพของการไขว้เปลี่ยนจึงไม่เห็นอย่างชัดเจน ปัญหาที่เราจะพิจารณาต่อไปเป็นปัญหาที่มีจุดดีที่สุดเฉพาะที่ซึ่งจะสร้างความยากลำบากสำหรับวิธีการค้นหาทั่วไป แต่จีเอสามารถจัดการกับปัญหาลักษณะนี้ได้ ปัญหานี้เป็นการหาส่วนผสมที่ดีที่สุดของคุกกี้เหมือนเดิมแต่ใช้ฟังก์ชันใหม่ดังรูปที่ 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

รูปที่ 6-5 ฟังก์ชันกฎเขามีกุณน้ำลอมของคุณภาพคึกก็

เริ่มต้นจากโครโมโซม 1-1 เช่นเดิม เราพบว่าในกรณีนี้การกลายพันธุ์เพียงอย่างเดียวไม่สามารถทำให้โครโมโซมในรุ่นที่อยู่ภายนอกคูกน้ำ (บริเวณที่มีค่าเป็น 0) ผลิตรโครโมโซมทะลุเข้าไปอยู่พื้นที่ภายในคูกน้ำได้ เนื่องจากโครโมโซมตรงกลางมีค่าความเหมาะสมเป็น 0 ซึ่งไม่สามารถอยู่รอดในรุ่นถัดไปได้ (ค่าความเหมาะสมของโครโมโซมเป็น 0 ทำให้ความน่าจะเป็นที่จะอยู่รอดไม่มีเลย) อย่างไรก็ตามการไขว่เปลี่ยนที่จับคู่โครโมโซมพ่อแม่ที่เหมาะสมเช่น 1-5 และ 5-1 จะสามารถผลิตลูกที่ข้ามคูกน้ำไปได้ จากการทดลอง 1,000 ครั้งพบว่าส่วนผสมที่ดีที่สุดถูกผลิตในรุ่นที่ 155 โดยเฉลี่ย!! เป็นผลที่ไม่ดี ถ้าเราคำนวณดูก็จะทราบทันทีว่าโครโมโซมที่แตกต่างกันที่เป็นไปได้ทั้งหมดมีแค่ $9 \times 9 = 81$ เส้นเท่านั้น ผลที่ได้คือรุ่นที่ 155 และแต่ละรุ่นมีโครโมโซมที่เราทดสอบมากกว่าหนึ่งเส้น (แม้ว่าจะมีโครโมโซมมากมายที่ซ้ำกันในรุ่นต่างๆ)

สาเหตุหนึ่งที่ผลไม่ดีก็เพราะว่าก่อนที่โครโมโซมจะกลายพันธุ์เป็นโครโมโซมที่อยู่บริเวณ 1-5 หรือ 5-1 นั้น โดยมากตายไปก่อนที่จะไปสู่บริเวณนั้นสำเร็จ และโอกาสที่คู่ที่เหมาะสมของโครโมโซมจะเกิดการไขว่เปลี่ยนก็มีโอกาสน้อยมาก ซึ่งที่จริงแล้วคู่ที่เหมาะสมของการไขว่เปลี่ยนมีจำนวนมากอย่างเช่น 2-6 กับ 4-2, 4-8 กับ 2-5, 6-8 กับ 2-4 เป็นต้น และโครโมโซมในคู่ทั้งหมดนี้ล้วนมีความน่าจะเป็นที่จะอยู่รอดเป็น 0 ทั้งสิ้น ที่เป็นเช่นนี้เกิดขึ้นจากฟังก์ชันความเหมาะสมมาตรฐานที่จะกำหนดให้โครโมโซมเหล่านี้มีความน่าจะเป็นที่จะอยู่รอดเป็น 0 หากเราปรับแก้ฟังก์ชันความเหมาะสมให้โครโมโซมเหล่านี้มีโอกาสอยู่รอดบ้างแม้จะน้อย ก็น่าจะช่วยให้การค้นหาส่วนผสมที่ดีที่สุดทำได้ดีขึ้น ดังจะกล่าวในหัวข้อต่อไป

6.1.5 ปรับปรุงจีเอดด้วยฟังก์ชันความเหมาะสมแบบลำดับและการใช้ความหลากหลาย

ปรับปรุงจีเอดด้วยฟังก์ชันความเหมาะสมแบบลำดับ

ฟังก์ชันความเหมาะสมใหม่ที่เราจะพิจารณากันนี้เรียกว่า **ค่าความเหมาะสมแบบลำดับ (rank fitness)** เป็นวิธีที่ใช้ควบคุมการเลือกโครโมโซมโดยไม่สนใจคุณภาพของโครโมโซมว่ามีค่าเท่าไร จะเพียงแค่จัดลำดับเรียงโครโมโซมตามคุณภาพที่มีค่าสูงสุดจนถึงต่ำสุด จากนั้นกำหนดให้ p ค่าคงที่ค่าหนึ่งเป็นความน่าจะเป็นที่โครโมโซมลำดับที่ 1 จะถูกเลือก และเป็นความน่าจะเป็นที่โครโมโซมลำดับที่ 2 จะถูกเลือกเมื่อลำดับที่ 1 ไม่ถูกเลือก และเป็นความน่าจะเป็นที่ลำดับที่ 3 จะถูกเลือกเมื่อลำดับที่ 1 และ 2 ไม่ถูกเลือก เป็นเช่นนี้ไปจนกระทั่งถึงลำดับสุดท้ายซึ่งจะถูกเลือกเมื่อลำดับก่อนหน้านั้นไม่ถูกเลือกเลย

ตัวอย่างเช่นสมมติว่า $p=2/3$ และโครโมโซมที่เราสนใจอยู่คือ 1-4, 3-1, 1-2, 1-1 และ 7-5 (ในกรณีของภูเขามีน้ล้น) จะได้ค่าความเหมาะสมของโครโมโซมดังตารางที่ 6-3 ซึ่งเปรียบเทียบค่าความเหมาะสมแบบลำดับกับค่าความเหมาะสมมาตรฐาน

ตารางที่ 6-3 เปรียบเทียบค่าความเหมาะสมแบบลำดับกับค่าความเหมาะสมมาตรฐาน

โครโมโซม	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 ค่าความเหมาะสมแบบลำดับของโครโมโซม 1-4 เท่ากับ $p = 2/3$ (ประมาณ 0.667) ส่วนโครโมโซม 1-3 มีค่าความน่าจะเป็นเท่ากับ $p(1-p)$ (ความน่าจะเป็นที่ตัวเองจะถูกเลือกเมื่อโครโมโซมลำดับที่ 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 (ส่วนผสมที่ดีที่สุดถูกผลิตในรุ่นที่ 155) ประมาณ 2 เท่า ซึ่งแสดงให้เห็นว่าค่าความเหมาะสมแบบลำดับดีกว่าค่าความเหมาะสมมาตรฐาน และจากการใช้ค่าความเหมาะสมแบบลำดับนี้ทำให้โครโมโซมที่อยู่ตรงกลางในคูน้าสามารถอยู่รอดถึงรุ่นถัดไปและวิวัฒนาการเป็น

โครโมโซมที่อยู่ภายในซึ่งมีคุณภาพสูงต่อไปได้ อย่างไรก็ตามแม้ว่าค่าความเหมาะสมแบบลำดับจะทำให้เร็วขึ้นกว่าเดิมประมาณ 2 เท่า แต่ยังคงเป็นผลที่ไม่ดีนักดังเช่นที่ได้กล่าวแล้วว่า 75 รุ่นแต่ละรุ่นเราตรวจสอบโครโมโซมมากกว่า 1 เส้น

เพิ่มประสิทธิภาพจีเอให้สูงขึ้นโดยความหลากหลาย

หัวข้อนี้แสดงการใช้ความหลากหลาย (diversity) เพื่อเพิ่มประสิทธิภาพของจีเอให้สูงขึ้นอีก ซึ่งได้แนวคิดจากการวิวัฒนาการของสิ่งมีชีวิต ที่เรามักพบว่าบ่อยครั้งในธรรมชาติที่สปีชีส์ซึ่งลักษณะแตกต่างไปจากสปีชีส์ที่เหมาะสมกับธรรมชาติสามารถอยู่รอดได้ดี ซึ่งความหลากหลายนี้จะช่วยให้โครโมโซมที่มียืนต่างจากพวกพ้องถูกคัดเลือกได้ง่ายขึ้น

การจะนำความต่างเข้าไปช่วยเลือกโครโมโซมนั้น อย่างแรกที่ต้องทำก็คือนิยามความต่างในรูปที่วัดได้ ในที่นี้เราจะวัดความต่างของโครโมโซมเส้นหนึ่งๆ โดยคำนวณค่าของ “ผลรวมของ 1/ระยะห่างกำลังสองระหว่างโครโมโซมนั้นกับโครโมโซมอื่นที่ถูกเลือกแล้วว่าให้อยู่รอดในรุ่นถัดไป” เนื่องจากเราต้องการโครโมโซมเส้นที่ต่างจากโครโมโซมที่เหมาะสมกับธรรมชาติ ดังนั้นการวัดความต่างหรือความหลากหลายจึงเทียบกับโครโมโซมเส้นที่เหมาะสมกับธรรมชาติ ส่วนระยะห่างหมายถึงระยะห่างตามระยะยูคลิด (Euclidian distance) เช่น 5-2 กับ 1-4 มีระยะห่างกำลังสองเท่ากับ $(5-1)^2 + (2-4)^2 = 20$ เป็นต้น

พิจารณาโครโมโซม 5-1, 1-4, 3-1, 1-2, 1-1 และ 7-5 โครโมโซมที่มีคุณภาพสูงสุดคือ 5-1 (ซึ่งเราจะเลือกเลยให้อยู่ในรุ่นถัดไปเป็นเส้นแรก) ตารางที่ 6-4 ด้านล่างแสดงลำดับของ 5 เส้นที่เหลือโดยเรียงตามคุณภาพและผลรวม 1/ระยะห่างกำลังสองจาก 5-1

ตารางที่ 6-4 ลำดับของโครโมโซมเรียงตามลำดับความหลากหลายและลำดับคุณภาพ

โครโมโซม	คุณภาพ	$1/d^2$	ลำดับความ หลากหลาย	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/ระยะห่างกำลังสองระหว่างโครโมโซมที่พิจารณากับ 5-1 ตัวอย่างเช่น 1-4 กับ 5-1 มีค่าเท่ากับ $1/((5-1)^2 + (1-4)^2) = 0.040$ เป็นต้น จากตารางจะพบว่าโครโมโซม 7-5 ซึ่งมีคุณภาพเป็น 0 และจะไม่เคยถูกเลือกเลยโดยค่าความเหมาะสมมาตรฐาน แต่เมื่อคำนวณค่าความเหมาะสมแบบลำดับความหลากหลายจะอยู่ในลำดับที่ 2 ซึ่งในกรณีนี้เมื่อดูจาก

รูปที่ 6-5 จะเห็นว่า 7-5 เป็นโครโมโซมที่ดีเส้นหนึ่งและมีโอกาสกลายพันธุ์เข้าสู่บริเวณด้านในของคูน้ำเพื่อเป็นคำตอบต่อไป

เมื่อเราได้ลำดับความหลากหลายแล้ว เราจำเป็นต้องนำลำดับนี้ผนวกเข้าไปใช้ร่วมกับค่าความเหมาะสมเดิม เราไม่อาจใช้ลำดับความหลากหลายอย่างเดียวได้เพราะเป็นแค่ปัจจัยหนึ่งในการเลือกโครโมโซม ลำดับคุณภาพเดิมซึ่งค่อนข้างดีอยู่แล้วก็ไม่อาจตัดทิ้งได้ ดังนั้นวิธีผนวกลำดับความหลากหลายเข้าใช้ร่วมกับลำดับคุณภาพสามารถทำได้โดยนำลำดับทั้งสองบวกกันแล้วจัดเรียงลำดับใหม่อีกครั้ง เราเรียกลำดับที่ได้ใหม่นี้ว่า **ลำดับรวม (combined rank)** เมื่อได้ลำดับรวมซึ่งคิดทั้งคุณภาพและความหลากหลายแล้ว การเลือกกระทำได้เหมือนเดิมโดยกำหนดความน่าจะเป็นของลำดับแรกเป็น $p = 2/3$ (ดูตารางที่ 6-5)

ตารางที่ 6-5 ลำดับรวมที่พิจารณาทั้งคุณภาพและความหลากหลาย

โครโมโซม	ผลรวมของลำดับ คุณภาพและลำดับ ความหลากหลาย	ลำดับผลรวม	ค่าความเหมาะสม
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

ลำดับผลรวมในตารางได้จากการเรียงลำดับผลในสดมภ์ที่สองใหม่ ในกรณีที่มีค่าเท่ากัน อย่างเช่น 3-1 กับ 7-5 มีค่าเท่ากันเท่ากับ 7 ก็ใช้การสุ่มเลือก ในที่นี้ 7-5 ถูกสุ่มให้มีลำดับผลรวมเป็นลำดับสาม จากตารางสมมติว่าเราเลือกโครโมโซมตามค่าความเหมาะสมได้เป็น 1-4 และเป็นเส้นที่สองต่อจาก 5-1 หลังจากนั้นเราจะเลือกเส้นที่ 3 ในครั้งนี้เราต้องคำนวณหา 1/ระยะห่างกำลังสอง โดยคิดทั้ง 5-1 และ 1-4 (ดูตารางถัดไป)

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

โครโมโซม	$\sum \frac{1}{d_i^2}$	ลำดับความ หลากหลาย	ลำดับ คุณภาพ	ลำดับรวม	ค่า ความเหมาะสม
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 $\sum_i \frac{1}{d_i^2}$ อย่างเช่นในกรณีของโครโมโซม 3-1 จะได้ค่าเป็น

$$\frac{1}{(5-3)^2 + (1-1)^2} + \frac{1}{(1-3)^2 + (4-1)^2} = 0.327 \text{ เป็นต้น สมมติว่าโครโมโซมที่ถูกเลือกตามค่าความ}$$

เหมาะสมต่อไปคือ 7-5 และโครโมโซมเส้นสุดท้ายเราก็สามารถทำได้ในลักษณะเดียวกัน และเลือกได้เป็น 1-1 ดังแสดงตารางที่ 6-7 ต่อไปนี้

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

โครโมโซม	$\sum_i \frac{1}{d_i^2}$	ลำดับความ หลากหลาย	ลำดับ คุณภาพ	ลำดับรวม	ค่า ความเหมาะสม
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

ค่าความเหมาะสมที่คำนวณตามลำดับรวมมีความแตกต่างจากค่าความเหมาะสมมาตรฐานที่โครโมโซม 7-5 ซึ่งเป็นโครโมโซมที่ดีเส้นหนึ่งและไม่เคยถูกเลือกเลยด้วยค่าความเหมาะสมมาตรฐาน แต่สามารถจะถูกเลือกได้ด้วยค่าความเหมาะสมตัวใหม่นี้

จากการทดลอง 1,000 ครั้งโดยใช้ลำดับรวมด้วยค่า $p = 2/3$ เริ่มจากโครโมโซม 1-1 คำตอบที่ดีที่สุดถูกผลิตได้ในรุ่นที่ 15 โดยเฉลี่ย!!! เร็วกว่าลำดับคุณภาพถึง 5 เท่า นอกจากนั้นค่าความเหมาะสมแบบลำดับรวมนี้ไม่ได้ถูกพัฒนาขึ้นโดยเฉพาะสำหรับแก้ปัญหาภูเขาหิมะน้ำล้นอย่างเดียวนั้น ยังสามารถทำงานได้ดีสำหรับปัญหาภูเขาเรียบด้วย ซึ่งดูสรุปการเปรียบเทียบค่าความเหมาะสมได้ในตารางที่ 6-8 ด้านล่างนี้ (ค่าในตารางได้จากการใช้การกลายพันธุ์และการไขว้เปลี่ยนเหมือนกันหมด)

ตารางที่ 6-8 เปรียบเทียบค่าความเหมาะสม 3 วิธี: มาตรฐาน ลำดับคุณภาพ และลำดับรวม

ฟังก์ชัน	ค่าความเหมาะสม มาตรฐาน	ค่าความเหมาะสมแบบ ลำดับคุณภาพ	ค่าความเหมาะสมแบบ ลำดับรวม
ภูเขาเรียบ	14	12	12
ภูเขาหิมะน้ำล้น	155	75	15

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

ตารางที่ 6-9 ผลการทดลองที่ดีที่สุดที่ผลิตโครโมโซมดีที่สุดได้ในรุ่นที่ 7 โดยลำดับรวม			
รุ่นที่ 0:		• 1-1 กลายพันธุ์เป็น 2-1	
โครโมโซม	คุณภาพ		
1-1	1		
รุ่นที่ 1:		• การกลายพันธุ์ผลิตได้ 3-1 ส่วนการไขว้เปลี่ยนไม่ได้ลูกตัวใหม่เพราะยีนตัวที่สองเหมือนกัน	
โครโมโซม	คุณภาพ		
2-1	2		
1-1	1		
รุ่นที่ 2:		• การกลายพันธุ์ผลิตได้ 4-1 และ 2-2 ส่วนการไขว้เปลี่ยนยังคงไม่เกิดผล	
โครโมโซม	คุณภาพ		
3-1	3		
2-1	2		
1-1	1		
รุ่นที่ 3:		• การกลายพันธุ์ผลิตได้โครโมโซมใหม่ 3 เส้นคือ 5-1, 1-2, 2-3 ส่วนการไขว้เปลี่ยนของ 2-2 กับ 4-1 ผลิตได้ 2-1 กับ 4-2 การไขว้เปลี่ยนของคู่อื่นซ้ำกับโครโมโซมที่ผลิตได้ก่อนมัน	
โครโมโซม	คุณภาพ		
4-1	4		
3-1	3		
1-1	1		
2-2	0		
		โครโมโซม	คุณภาพ
		5-1	5
		1-2	2
		2-3	0
		2-1	2
		4-2	0
รุ่นที่ 4:		• การกลายพันธุ์ผลิตได้ 6-1, 3-2, 2-2, 2-4 ส่วนการไขว้เปลี่ยนผลิต 2-1, 1-1, 5-2, 3-2, 5-3	
โครโมโซม	คุณภาพ		
5-1	5		
3-1	4		
1-2	2		
2-3	0		
		โครโมโซม	คุณภาพ
		6-1	4
		3-2	0
		2-2	0

		2-4	0
		2-1	2
		1-1	1
		5-2	0
		3-2	0
		5-3	0
<p>รุ่นที่ 5:</p> <p>โครโมโซม</p> <p>คุณภาพ</p>		<p>• ที่จุดนี้เกิดการไขว้เปลี่ยนของ 5-1 กับ 2-4 ได้ 5-4 ซึ่งเป็นโครโมโซมที่ดีในรุ่นหน้า</p>	
	5-1	5	
	3-1	3	
	1-2	2	
	2-4	0	
<p>รุ่นที่ 6:</p> <p>โครโมโซม</p> <p>คุณภาพ</p>		<p>• และในท้ายที่สุด 5-4 กลายพันธุ์เป็น 5-5</p>	
	5-4	8	
	1-4	4	
	3-1	3	
	1-2	2	
<p>รุ่นที่ 7:</p> <p>โครโมโซม</p> <p>คุณภาพ</p>			
	5-5	9	
	1-4	4	
	1-2	2	
	5-2	0	

ด้านล่างนี้แสดงการค้นหาคำตอบโดยจีเอ โดยแสดงเฉพาะโครโมโซมที่ถูกเลือกในแต่ละรุ่น

(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

(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

(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

(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

(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

(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

(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

(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 การค้นหาโดยจีเอในปัญหาภูเขามีน้ล้น

จากรูปจะเห็นว่าในรุ่นที่ 3 โครโมโซมที่คุณภาพเป็น 0 สามารถถูกเลือกได้โดยค่าความเหมาะสมแบบลำดับรวมและจะเห็นการเคลื่อนที่ของโครโมโซมจากรุ่นที่ 1 ถึง 4 ว่าโครโมโซมค่อยๆ ขยับตัวไปยังจุดสูงสุดเฉพาะที่ซึ่งมีคุณภาพเท่ากับ 5 และจะเห็นการเคลื่อนที่ของโครโมโซมที่มีคุณภาพเท่ากับ 0 ที่ค่อยๆ ขยับออกจากจุดสูงสุดเฉพาะที่ที่ละ

น้อย จนกระทั่งในรุ่นที่ 5 เมื่ออยู่ในตำแหน่งที่เหมาะสมและเกิดการไขว้เปลี่ยนกับจุดสูงสุดเฉพาะที่ แล้วสามารถทะลุผ่านคูลน้ำเข้าไปยังภายในคูลได้ แล้วเปลี่ยนเป็นจุดสูงสุดในที่สุด

จากรูปแสดงการทำงานของจีเอ เราสามารถเห็นได้ว่าการค้นหาโดยทั่วไปมักจะพยายามหลีกเลี่ยงจุดดีที่สุดเฉพาะที่ แต่การทำงานของจีเอใช้วิธีการที่ต่างไป โดยการผลิตโครโมโซมที่เป็นค่าดีที่สุดเฉพาะที่จากนั้นจึงใช้ความหลากหลายเพื่อเป็นส่วนประกอบของค่าความเหมาะสม แล้วผลิตโครโมโซมที่อยู่ห่างออกจากค่าดีที่สุดเฉพาะที่ หากมีโครโมโซมอยู่ในจุดดีที่สุดเฉพาะที่ทุกจุดแล้ว ก็มีโอกาสที่โครโมโซมเหล่านี้จะหาทางไปยังจุดที่ดีสุดวงกว้าง (global optimum) ได้ในที่สุด

6.2 การเรียนรู้โดยการจำ

การเรียนรู้โดยการจำ (rote learning) เป็นการเรียนรู้แบบที่ง่ายที่สุดของกระบวนการเรียนรู้ทั้งหลาย โดยเมื่อพบความรู้หรือข้อเท็จจริงใหม่ๆ ก็เก็บไว้ในหน่วยความจำ เวลาที่ต้องการใช้ก็เพียงแค่อ้างความรู้ขึ้นมาใช้ ถ้าเรามองว่าระบบปัญญาประดิษฐ์มีหน้าที่รับอินพุต (X_1, \dots, X_n) แล้วทำการหาเอาต์พุต $(Y_1, \dots, Y_n) = f(X_1, \dots, X_n)$ โดยที่ f เป็นฟังก์ชันใดๆ ในการคำนวณเอาต์พุตหรืออาจเป็นการอนุมานหาค่าเอาต์พุตจากอินพุตก็ได้ ดังนั้นการเรียนรู้โดยการจำก็คือการเก็บคู่ลำดับ $[(X_1, \dots, X_n), (Y_1, \dots, Y_n)]$ ไว้ในหน่วยความจำ หลังจากนั้นเมื่อเราต้องการหา $f(X_1, \dots, X_n)$ ใหม่ก็ทำโดยการดึง (Y_1, \dots, Y_n) จากคู่ลำดับนี้เท่านั้นโดยไม่ต้องคำนวณหรืออนุมานซ้ำอีกครั้งซึ่งโดยมากจะเสียต้นทุนและเวลาสูง

จะเห็นว่าแนวคิดนี้ง่ายแต่ไม่ได้หมายความว่า การเรียนรู้จะไม่มีประสิทธิภาพ มนุษย์เราก็เรียนรู้โดยการจำด้วยเช่นกันหรือซอฟต์แวร์ในปัจจุบันหลายตัวก็สามารถจำชื่อไฟล์ที่ผู้ใช้ใช้งานครั้งล่าสุดได้และช่วยให้การเปิดไฟล์ทำได้ง่ายขึ้นมีประโยชน์ในการใช้งานจริง

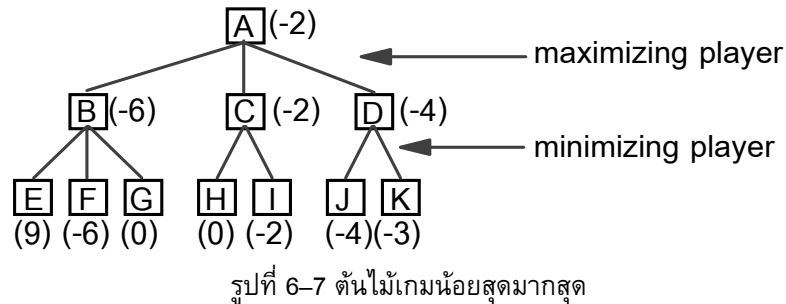
ในการเรียนรู้โดยการจำนี้ สิ่งที่เราต้องพิจารณาเพิ่มเติมได้แก่ (1) การจัดการหน่วยความจำ (memory organization) ที่ต้องมีประสิทธิภาพสามารถดึงความรู้ที่เก็บไว้ได้อย่างรวดเร็ว (2) ความเสถียรภาพของสภาพแวดล้อมต้องไม่เปลี่ยนแปลงอย่างรวดเร็วจนส่งผลให้ความรู้ที่เก็บไว้ไม่ถูกต้องเมื่อเวลาเปลี่ยนไป (3) ความสมดุลระหว่างการคำนวณใหม่กับการจัดเก็บ ต้องมีสมดุลที่ดีไม่จัดเก็บมากเกินไปจนทำให้การค้นคืนคู่ลำดับที่จัดเก็บมีประสิทธิภาพต่ำ ส่งผลให้ประสิทธิภาพโดยรวมลดลงเพราะเสียเวลามากไปเพื่อตรวจสอบว่าเป็นความรู้ที่อยู่ในหน่วยความจำหรือไม่ ดังนั้นควรเลือกจำเฉพาะความรู้ที่ใช้อยู่

แม้ว่าแนวคิดนี้จะง่ายแต่หากใช้ได้ตรงกับงานประยุกต์หนึ่งๆ ก็จะส่งผลให้ประสิทธิภาพของระบบเพิ่มขึ้นได้อย่างดี ดังเช่นที่จะแสดงในการเรียนรู้โดยการจำของโปรแกรมเชกเกอร์ (checkers) ของ Samuel เชกเกอร์² เป็นเกมที่เล่นให้เก่งยากและการพัฒนาโปรแกรมเชกเกอร์ให้เล่นแข่งชนะมนุษย์ก็ไม่ง่าย ตาเดินที่เป็นไปได้ทั้งหมดของเชกเกอร์มีประมาณ 10^{40} ตาเดิน ทำให้การสร้างตาเดินทั้งหมดโดยโปรแกรมก่อนที่จะตัดสินใจว่าจะเลือกตาเดินต่อไปอย่างไรไม่สามารถทำได้อย่างรวดเร็วโดยคอมพิวเตอร์ประสิทธิภาพไม่สูงนัก ดังนั้นโปรแกรมเล่นเกมประเภทนี้จะสร้างตาเดินได้เท่าที่เวลาอำนวย เช่นถ้าต้องเดินหมากรุกภายใน 1 นาที โปรแกรมสร้างตาเดินที่เป็นไปได้เท่าไรก็เท่านั้น แล้วเลือกจากตาเดิน

² เป็นเกมคล้ายกับหมากรุกไทย แต่มีจำนวนเบี้ยแต่ละฝ่าย 12 ตัว

การค้นหา
ต้นไม้เกมน้อย
สุดมากที่สุด

ที่สร้างได้ ลักษณะของอัลกอริทึมประเภทนี้เป็นการค้นหาแบบหนึ่งซึ่งมีผู้เล่นสองฝ่ายคือ โปรแกรมกับฝ่ายตรงข้าม อัลกอริทึมที่นิยมใช้ในเกมประเภทนี้ก็คือ **การค้นหาต้นไม้เกมน้อยสุดมากที่สุด (minimax game-tree search)** ดังแสดงในรูปที่ 6-7



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

ตัวเลขที่สถานะแต่ละตัวแสดงค่าความดีของสถานะนั้นๆ ค่าเหล่านี้เป็นค่าของ**ผู้เล่นฝ่ายทำมากที่สุด** ถ้าค่ามากแสดงว่าโอกาสชนะของผู้เล่นฝ่ายทำมากที่สุดมีมาก แต่ถ้าน้อยแสดงว่าผู้เล่นฝ่ายทำมากที่สุดมีโอกาสน้อย ดังนั้นหน้าที่ของผู้เล่นฝ่ายทำมากที่สุดคือพยายามทำให้ตัวเลขเหล่านี้มีค่ามากโดยเลือกเส้นทางที่จะทำให้ค่าสูงสุด ตัวเลขเหล่านี้แบ่งเป็น 2 จำพวกคือ (1) ตัวเลขที่สถานะปลายต้นไม้ (ใบ) (9, -6, 0,..., -3) และ (2) ตัวเลขที่สถานะเริ่มต้นและสถานะภายในต้นไม้ เราเรียกว่าตัวเลขที่ปลายต้นไม้ว่า **ค่าประเมินสถิต (static evaluation value)** ค่าเหล่านี้เป็นค่าฮิวริสติกที่วัดค่าความดีของการจัดเรียงตัวหมากบนกระดานว่าโอกาสชนะของผู้เล่นฝ่ายทำมากที่สุดมีมากแค่ไหน ค่าประเมินสถิตนี้วัดจากจำนวนเบี้ยของเราว่ามากกว่าของฝ่ายตรงข้ามมากน้อยแค่ไหน จำนวนขุน (king) ของเรามีมากกว่าฝ่ายตรงข้ามแค่ไหน ตำแหน่งของตัวหมากของเราอยู่ในตำแหน่งที่ได้เปรียบฝ่ายตรงข้ามมากน้อยแค่ไหน เป็นต้น

ตัวเลขที่สถานะเริ่มต้นและสถานะภายในต้นไม้เรียกว่า **ค่าแบ็คอัพ (backup value)** เป็นค่าที่ได้จากการส่งค่าประเมินสถิติจากด้านล่างย้อนกลับขึ้นไปทางด้านบนที่ระดับ ในการคำนวณค่าแบ็คอัพนั้นจะพิจารณาเป็น 2 กรณีคือ (1) กรณีที่ผู้เล่นฝ่ายทำน้อยสุดเป็นผู้สร้างสถานะลูก ค่าแบ็คอัพของสถานะพ่อแม่จะเป็นค่าต่ำสุดในจำนวนค่าทั้งหมดของสถานะลูก (2) กรณีที่ผู้เล่นฝ่ายทำมากที่สุดเป็นผู้สร้างสถานะลูก ค่าแบ็คอัพของสถานะพ่อแม่จะเป็นค่าสูงสุดในจำนวนค่าทั้งหมดของลูก เช่นกรณีการคำนวณค่าแบ็คอัพของสถานะ B ซึ่งเป็นกรณีที่ (1) นั้น ค่าของ B จะเท่ากับ $\min\{9, -6, 0\} = -6$ กรณีการคำนวณค่าแบ็คอัพของสถานะ A ซึ่งเป็นกรณีที่ (2) นั้น ค่าของ A จะเท่ากับ $\max\{-6, -2, -4\} = -2$ เนื่องจากค่าในต้นไม้เป็นค่าที่แสดงโอกาสที่ผู้เล่นฝ่ายทำมากที่สุดมีโอกาสชนะ ดังนั้นผู้เล่นฝ่ายทำมากที่สุดจึงต้องพยายามทำให้ค่าที่ได้มีค่ามากที่สุด ส่วนผู้เล่นฝ่ายทำน้อยสุดมีหน้าที่สกัดกั้นไม่ให้ผู้เล่นฝ่ายทำมากที่สุดมีโอกาสชนะ ดังนั้นจึงต้องพยายามทำให้ค่าที่ได้มีค่าน้อยสุด และเป็นที่มาของชื่ออัลกอริทึมนี้

จากตัวอย่างในรูปด้านบน เมื่อผู้เล่นฝ่ายทำมากที่สุดจะเลือกตาเดินก็ควรเลือกเส้นทางตามค่าแบ็คอัพ กล่าวคือเมื่ออยู่ที่สถานะ A ควรเลือกตาเดินไปยังสถานะ C ซึ่งคาดว่าหลังจากนั้นฝ่ายผู้เล่นทำน้อยสุดน่าจะเดินไปยัง I สังเกตว่าในจำนวนสถานะทั้งหมดค่าที่มากที่สุดคือ 9 ของสถานะ E แต่อย่างไรก็ดีผู้เล่นฝ่ายทำมากที่สุดไม่มีโอกาสที่จะได้ค่าแบ็คอัพเป็น 9 ได้ แม้ว่าตนเองจะเดินจากสถานะ A ไปยัง B เพราะว่ามีผู้เล่นฝ่ายทำน้อยสุดพยายามขัดขวางให้ค่าที่ได้มีค่าน้อยสุด ถ้าผู้เล่นฝ่ายทำมากที่สุดเดินมายังสถานะ B ผู้เล่นฝ่ายทำน้อยสุดก็จะเดินไปยัง F ทำให้โอกาสชนะของผู้เล่นฝ่ายทำมากที่สุดเหลือ -6 (อย่าลืมว่าตัวเลขในต้นไม้เป็นค่าที่แสดงโอกาสชนะของผู้เล่นฝ่ายทำมากที่สุดเท่านั้น) ในรูปที่ 6-7 นั้นแสดงการค้นหาที่มองล่วงหน้า 2 ก้าวเดิน (2 moves look-ahead)

ค่าแบ็คอัพที่คำนวณได้ของสถานะ A นี้จะไม่เท่ากับค่าประเมินสถิติของ A เนื่องจากว่าถ้าเราวัดค่าประเมินสถิติก็คือการคำนวณค่าฮิวริสติกของ A โดยตรงโดยดูที่ตัวหมาก ณ สถานะ A แต่ค่าแบ็คอัพของ A คือการมองล่วงหน้าต่อจากนี้อีก 2 ก้าวเดินในทุกเส้นทางแล้วคำนวณเส้นทางที่น่าจะเป็นที่สุด (เส้นทางที่ผู้เล่นทั้งสองเลือกตาเดินได้ดีที่สุด) แล้วส่งค่าประเมินสถิติที่ปลายต้นไม้ย้อนกลับมาที่สถานะ A ดังนั้นค่าแบ็คอัพจะมีความถูกต้องแม่นยำมากกว่าค่าประเมินสถิติโดยตรงของ A

ค่าแบ็คอัพที่ได้จากการมองล่วงหน้า 2 ก้าวเดินมีความแม่นยำมากกว่าค่าประเมินสถิติในทำนองเดียวกันค่าแบ็คอัพที่ได้จากการมองล่วงหน้า 3 ก้าวเดินก็ย่อมมีความแม่นยำมากกว่ามองล่วงหน้า 2 ก้าวเดิน ยิ่งเราเพิ่มการมองล่วงหน้าได้ลึกเท่าไร ความแม่นยำของค่าแบ็คอัพที่คำนวณได้ก็ยิ่งสูงขึ้นเท่านั้น และถ้าเราสามารถมองล่วงหน้าจนถึงสถานะที่จบ

ด้วยการใช้ค่าเบ็คคอฟของ A แทนที่จะใช้ค่าประเมินสถิติก็เหมือนกับว่าที่จุด A นี้ได้รวมการค้นหาอีก 3 ก้าวเดินล่วงหน้าเข้าไว้ด้วย ดังนั้นที่ E แม้ว่าด้วยข้อจำกัดทางเวลาทำให้เราค้นหาได้เพียง 3 ก้าวเดินล่วงหน้า แต่ก็เหมือนกับว่าในเส้นทางที่รวม A จะเป็นการค้นหาล่วงหน้าถึง 6 ก้าวเดินล่วงหน้า และด้วยการจำคู่ลำดับระหว่างสถานะกับค่าเบ็คคอฟไว้จำนวนมากก็จะทำให้การค้นหาเพิ่มจำนวนก้าวเดินล่วงหน้าเป็น 3, 6, 9, ... ตามลำดับ ซึ่งส่งผลให้ประสิทธิภาพของโปรแกรมเพิ่มขึ้น

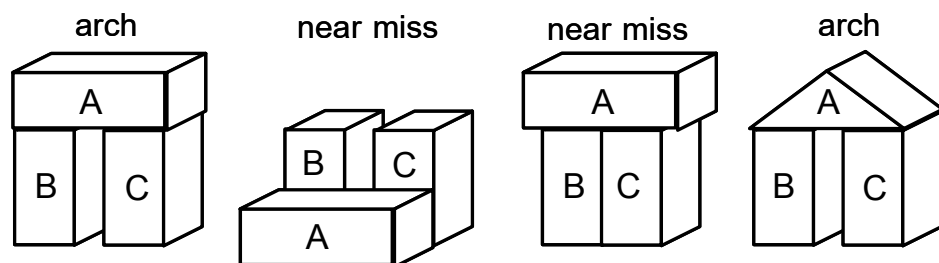
ตัวอย่างนี้แสดงให้เห็นการประยุกต์ใช้การเรียนรู้โดยการจำที่ส่งผลให้ประสิทธิภาพของงานที่กระทำดีขึ้นอย่างชัดเจน และโปรแกรมการเรียนรู้การเล่นเกมเชกเกอร์สก็ยังมีการจัดการหน่วยความจำอย่างประหยัดโดยจัดเก็บเฉพาะตำแหน่งตัวหมากบนกระดานของผู้เล่นฝ่ายทำมากที่สุดฝ่ายเดียว และเมื่อจะใช้กับผู้เล่นฝ่ายทำน้อยสุดก็สลับตำแหน่งของตัวหมากกลับด้านกันเท่านั้น นอกจากนั้นยังมีการทำดัชนีเพื่อดึงตำแหน่งตัวหมากบนกระดานให้ได้อย่างรวดเร็วโดยใช้คุณสมบัติของกระดาน เช่นจำนวนตัวหมาก การมีหรือไม่มีขุน เพื่อใช้เป็นดัชนี และยังได้จัดการปัญหาความสมดุลระหว่างการจัดเก็บกับการคำนวณใหม่โดยใช้วิธีที่เรียกว่าการแทนที่ตัวที่ถูกใช้น้อยสุด (least recently used replacement) วิธีนี้พยายามจะไม่จัดเก็บคู่ลำดับให้มากมายเกินไปเพราะจะทำให้การค้นคืนคู่ลำดับใช้เวลามาก โดยกำหนดจำนวนคู่ลำดับที่จะจำเป็นค่าคงที่ค่าหนึ่ง เช่น 100,000 คู่ลำดับ จากนั้นคู่ลำดับใดที่ถูกใช้น้อยสุด (เมื่อจำไว้แล้วถูกพบในต้นไม้เกมอื่นน้อยสุด) จะถูกลบออกจากหน่วยความจำแล้วแทนที่ด้วยคู่ลำดับใหม่ตัวอื่น วิธีนี้ทำโดยกำหนดอายุให้กับคู่ลำดับแต่ละคู่และทุกครั้งที่คู่ลำดับถูกเรียกมาใช้อายุของมันจะลดลงครึ่งหนึ่ง และทุกครั้งที่มีการจำคู่ลำดับใหม่ อายุของคู่ลำดับอื่นทุกตัวในหน่วยความจำจะถูกบวกเพิ่ม 1 หน่วย จากนั้นตัวที่มีอายุมากที่สุดจะถูกลบออกจากหน่วยความจำ

6.3 การเรียนรู้โดยการวิเคราะห์ความแตกต่าง

ตัวอย่างบวก
และ
ตัวอย่างลบ

การเรียนรู้โดยการวิเคราะห์ความแตกต่าง (learning by analyzing differences) ถูกพัฒนาโดย Winston ในปีค.ศ. 1975 [Winston, 1992] แม้ว่าจะเป็นวิธีการเรียนรู้ที่ค่อนข้างเก่ามาก แต่ก็ตาม แนวคิดต่างๆ สามารถนำไปใช้ในการเรียนรู้แบบใหม่ๆ ได้อย่างดี ในที่นี้จะยกวิธีการเรียนรู้แบบนี้มาเพื่อศึกษาแนวคิดของการเรียนรู้เชิงอุปนัย การเรียนรู้โดยการวิเคราะห์ความแตกต่างนี้ใช้เรียนรู้โมโนทัศน์ทางโครงสร้าง (structural concept) ในโดเมนปัญหาโลกของบล็อก เช่น arch, tent หรือ house เป็นต้น วิธีการเรียนรู้นี้จะวิเคราะห์ความแตกต่างที่ปรากฏในลำดับของตัวอย่างที่ผู้สอนป้อนให้ โดยตัวอย่างสอน (training example) มี 2 ประเภทคือ **ตัวอย่างบวก (positive example)** และ **ตัวอย่างลบ (negative example)** ตัวอย่างบวกคือตัวอย่างที่ถูกต้องของโมโนทัศน์ (concept) ที่สอน เช่นจะสอน house ตัวอย่างบวกก็จะเป็นบ้านหลังที่หนึ่ง บ้านหลังที่สอง เป็นต้น ตัวอย่างลบคือตัวอย่างที่ไม่ถูกต้อง เช่นจะสอน house ตัวอย่างลบก็จะเป็นเต็นท์หลังที่หนึ่ง โรงเรียนหลังที่หนึ่ง เหล่านี้เป็นต้น

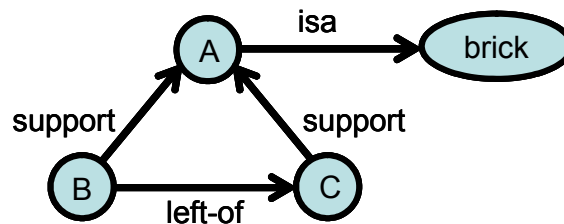
สำหรับการเรียนรู้โดยการวิเคราะห์ความแตกต่างนี้ ตัวอย่างลบที่ผู้สอนให้จะต้องเป็นตัวอย่างลบแบบที่เรียกว่า **พลาดน้อย (near miss)** กล่าวคือตัวอย่างลบแบบพลาดน้อยนี้จะต่างจากตัวอย่างบวกเพียงเล็กน้อย เช่นจะสอน house ตัวอย่างลบแบบพลาดน้อยก็จะเป็นบ้านที่ขาดประตู หรือบ้านที่ไม่มีหลังคา เป็นต้น การเรียนรู้แบบนี้ผู้สอนจะจัดเตรียมลำดับของตัวอย่างไว้ค่อนข้างดีเพื่อให้โปรแกร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



รูปที่ 6-10 ตัวอย่างบวกและตัวอย่างลบแบบพลาดน้อยของ arch

ในรูป 'arch' และ 'near miss' หมายถึงตัวอย่างบวกและตัวอย่างลบแบบพลาดน้อยตามลำดับ จากตัวอย่างที่ให้ทั้งสี่ตัวนี้ โปรแกรมจะเรียนรู้ว่าอะไรคือ arch เมื่อเราดูตัวอย่างข้างต้น เราจะพอเข้าใจได้ว่าตัวอย่างบวกตัวแรกบอกว่า arch คือสิ่งที่ประกอบด้วยอิฐ (brick) แนวตั้ง 2 ก้อนและอิฐแนวนอน 1 ก้อนที่ถูกรองรับด้วยอิฐแนวตั้ง ตัวอย่างที่สองอธิบายสิ่งที่ไม่ใช่ arch ว่าคือสิ่งที่ประกอบด้วยอิฐแนวตั้ง 2 ก้อนและอิฐแนวนอนซึ่งไม่ถูกรองรับด้วยอิฐแนวตั้ง ตัวอย่างที่ 3 และ 4 แสดงตัวอย่างของ arch และสิ่งที่ไม่ใช่ตามลำดับ โปรแกรมเรียนรู้นี้ใช้การแทนความรู้เพื่อแสดงมโนทัศน์ในรูปของ **ข่ายงานความหมาย (semantic network)** การแทนความรู้แบบนี้จะประกอบด้วยบัพ (node) และเส้นเชื่อม (link) บัพแสดงวัตถุและเส้นเชื่อมแทนความสัมพันธ์ระหว่างวัตถุในโดเมนนั้น

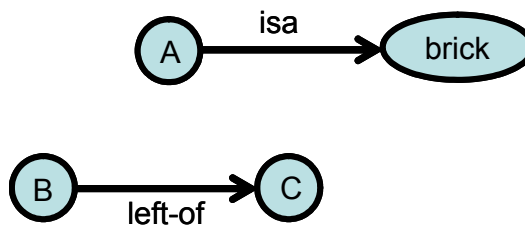
จากตัวอย่างบวกตัวที่หนึ่ง โปรแกรมจะสร้างคำอธิบายเริ่มต้น (initial description) ของมโนทัศน์ดังรูปที่ 6-11



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 ขอละไว้ในที่นี้ เช่นเส้นเชื่อม isa จาก B ไปยังบัพ brick เป็นต้น

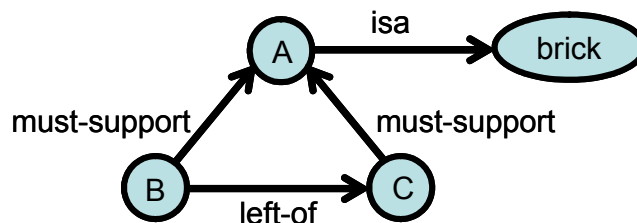
จากตัวอย่างลบแบบพลาดน้อยตัวที่สอง โปรแกรมสร้างคำอธิบายของตัวอย่างลบได้ดังรูปที่ 6-12



รูปที่ 6-12 คำอธิบายของตัวอย่างตัวที่สอง

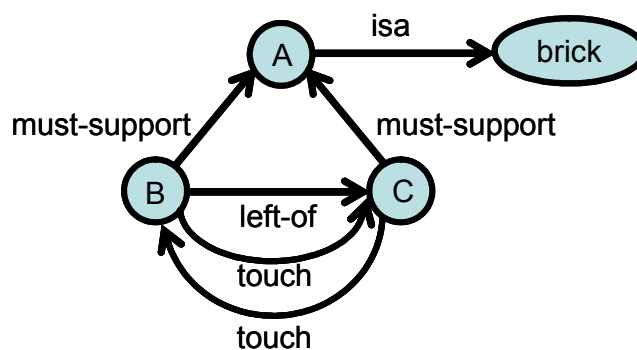
การแจง
จำเพาะของ
มโนทัศน์

ที่จุดนี้โปรแกรมจะวิเคราะห์หาความแตกต่างของตัวอย่างที่ถูกกับที่ผิดโดยการจับคู่บัปและเส้นเชื่อม และพบว่าเส้นเชื่อม support ซึ่งต่างกันในตัวอย่างทั้งสองจำเป็นสำหรับมโนทัศน์ arch โปรแกรมจึงใส่เงื่อนไขเพิ่มเข้าไปในคำอธิบายในรูปที่ 6-11 โดยใช้เส้นเชื่อมใหม่ชื่อ must-support แทนที่เส้นเชื่อมเดิมดังแสดงในรูปที่ 6-13 เราเรียกคำอธิบายใหม่ที่ได้นี้ว่า *โมเดลระหว่างวิวัฒนาการ (evolving model)* ในกรณีนี้ตัวอย่างลบให้ข้อมูลสำหรับการใช้ *ฮิวริสติกเส้นเชื่อมจำเป็น (require-link heuristic)* ที่ใส่เงื่อนไขที่มากขึ้นในเส้นเชื่อมเดิม เราเรียกการทำเช่นนี้ว่าเป็นฮิวริสติกแบบหนึ่งเนื่องจากว่าเป็นการคาดคะเนจากเหตุผลของความแตกต่างระหว่างตัวอย่างที่น่าจะเป็น แต่ก็อาจไม่ถูกต้องเสมอไปก็เป็นได้ ในกรณีนี้ตัวอย่างลบแบบพลาตน้อยเป็นตัวอย่างที่ให้ข้อมูลสำหรับการ *การแจงจำเพาะของมโนทัศน์ (specialization of concept)* ซึ่งหมายถึงว่าคำอธิบายของมโนทัศน์จะถูกทำให้แคบลง มีเงื่อนไขมากขึ้น ตรงกับตัวอย่างจำนวนน้อยลง



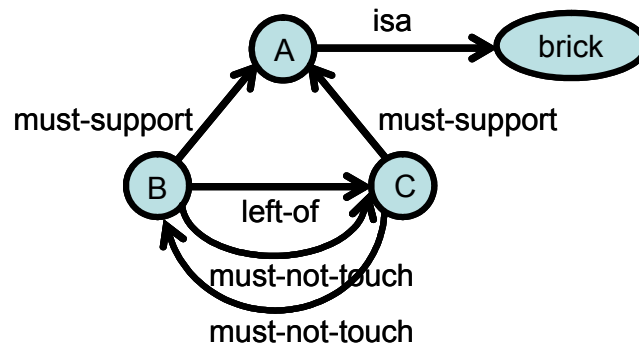
รูปที่ 6-13 โมเดลระหว่างวิวัฒนาการ

จากตัวอย่างลบตัวที่สาม โปรแกรมสร้างคำอธิบายของตัวอย่างลบได้ดังรูปที่ 6-14



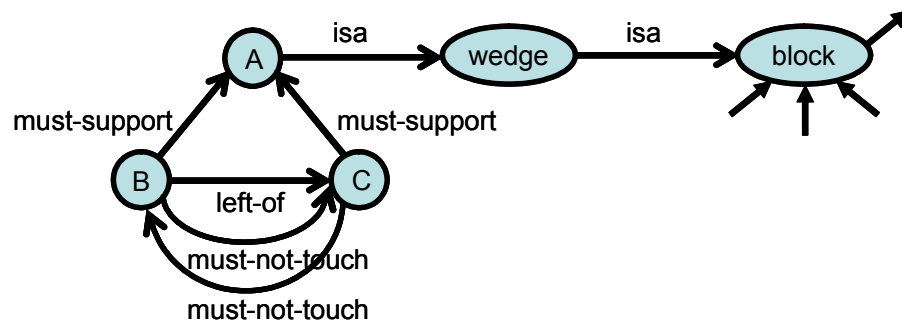
รูปที่ 6-14 คำอธิบายของตัวอย่างลบตัวที่สาม

โปรแกรมหาความแตกต่างระหว่างคำอธิบายของตัวอย่างที่สามกับโมเดล พบว่ามีเส้นเชื่อม touch อยู่ในตัวอย่างลบซึ่งไม่มีในโมเดล ดังนั้นโปรแกรมจึงเพิ่มเส้นเชื่อมเข้าไปในโมเดลและปรับโมเดลใหม่ได้ดังรูปที่ 6-15 ในกรณีนี้ตัวอย่างลบให้ข้อมูลสำหรับ*ฮิวริสติก* เส้นเชื่อมห้าม (*forbid-link heuristic*)



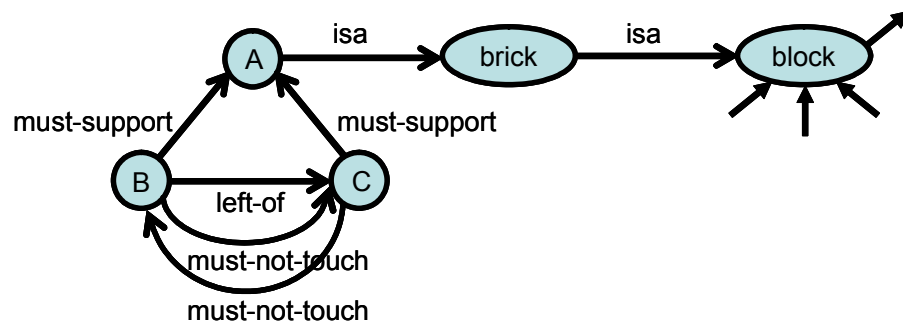
รูปที่ 6-15 โมเดลหลังรับตัวอย่างตัวที่สาม

จากตัวอย่างบวกตัวที่สี่ โปรแกรมสร้างคำอธิบายของตัวอย่างได้ดังรูปที่ 6-16



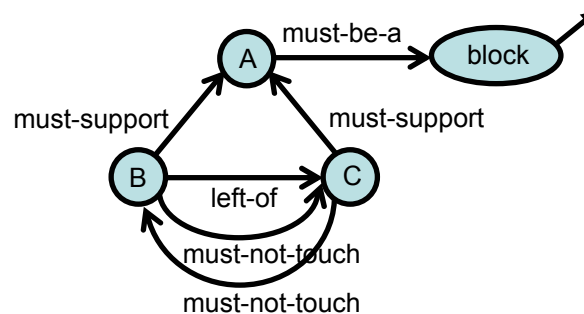
รูปที่ 6-16 คำอธิบายของตัวอย่างบวกตัวที่สี่

สมมติว่าเรามีความสัมพันธ์ของต้นไม้จำแนกประเภท (classification tree) ว่าวัตถุหนึ่งๆ จัดอยู่ในประเภทอะไรในฐานความรู้ของเราด้วย เช่นในที่นี้ wedge จัดเป็นวัตถุหนึ่งในประเภทของ block ในทำนองเดียวกัน brick ก็จัดเป็นวัตถุหนึ่งในประเภทของ block ด้วย โมเดลของเราในรูปที่ 6-15 เมื่อนำมาเขียนใหม่ให้รวมความสัมพันธ์ของต้นไม้จำแนกประเภทเข้าไปด้วยก็จะได้ดังรูปที่ 6-17



รูปที่ 6-17 โมเดลหลังรับตัวอย่างตัวที่สามที่รวมต้นไม้จำแนกประเภทด้วย

เมื่อนำโมเดลเปรียบเทียบกับคำอธิบายตัวอย่างที่สี่ข้างต้นจะพบว่ามีความแตกต่างกันที่ brick กับ wedge และทั้งคู่ต่างก็เป็นวัตถุในประเภทของ block ดังนั้นเราจึงแทนที่ brick ด้วยประเภทที่สูง (กว้าง) กว่าคือ block ได้เป็นโมเดลในรูปที่ 6-18 เราเรียกอิวิริสติกแบบนี้ว่า *อิวิริสติกปีนต้นไม้ (climb-tree heuristic)*



รูปที่ 6-18 โมเดลเมื่อรับตัวอย่างครบทุกตัว

ในกรณีที่เราไม่มีต้นไม้จำแนกประเภท โปรแกรมจะสร้างประเภทใหม่คือ “brick-or-wedge” ขึ้นมาเพื่อใช้แทนบัพ block ในรูปที่ 6-18 เราเรียกอิวิริสติกแบบนี้ว่า *อิวิริสติกขยายเซต (enlarge-set heuristic)* และถ้าหากว่าในกรณีที่เราไม่มีวัตถุอื่นอยู่ในโดเมนนี้อีกเลยที่นอกเหนือจาก brick และ wedge เราก็สามารถตัดเส้นเชื่อม isa ออกได้เลยเพื่อเป็นการลดเงื่อนไข และในกรณีนี้เราเรียกอิวิริสติกนี้ว่า *อิวิริสติกตัดเส้นเชื่อม (drop-link heuristic)* ในกรณีเหล่านี้ตัวอย่างบวกทำหน้าที่สำหรับการวางนัยทั่วไปของมโนทัศน์ (*generalization of concept*) ซึ่งหมายความว่าคำอธิบายของมโนทัศน์จะถูกทำให้กว้างขึ้น มีเงื่อนไขน้อยลง ตรงกับตัวอย่างจำนวนมากขึ้น

การวางนัย
ทั่วไปของ
มโนทัศน์

อัลกอริทึมของโปรแกรมเรียนรู้โดยวิเคราะห์ความแตกต่างแสดงในตารางที่ 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

Speicalization 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 เวอร์ชันสเปซ

เวอร์ชันสเปซ (version space) [Mitchell, 1977] เรียนรู้คำอธิบายที่อธิบายตัวอย่างบวกและไม่อธิบายตัวอย่างลบ [รูปที่ 6-19](#) ด้านล่างแสดงตัวอย่างของการเรียนมโนทัศน์ car ซึ่งใช้ *การแทนความรู้แบบกรอบ (frame)*

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

รูปที่ 6-19 ตัวอย่างบวกของมโนทัศน์ car

กรอบประกอบด้วยชื่อกรอบ ในที่นี้คือ Car023 และสล็อต (slot) ในที่นี้สล็อตมี 5 ตัวคือ origin, manufacture, color, decade และ type ซึ่งแสดงคุณสมบัติทั้งห้าอย่างของรถยนต์ สมมติว่าสล็อตแต่ละตัวมีค่าที่เป็นไปได้ตาม [ตารางที่ 6-11](#) ด้านล่างนี้

ตารางที่ 6-11 ค่าที่เป็นไปได้ของสล็อตแต่ละตัว

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

การเรียนรู้โดยเวอร์ชันสเปซจะแสดงคำอธิบายมโนทัศน์ในรูปของสล็อตและค่าของสล็อต เช่นถ้าเป็นมโนทัศน์ “Japanese economy car” จะแสดงได้ดัง [รูปที่ 6-20](#) โดยที่ x1, x2 และ x3 เป็นตัวแปรสามารถถูกแทนด้วยค่าคงที่ใดๆ

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

รูปที่ 6-20 มโนทัศน์ “Japanese economy car”

สอดคล้องกับ

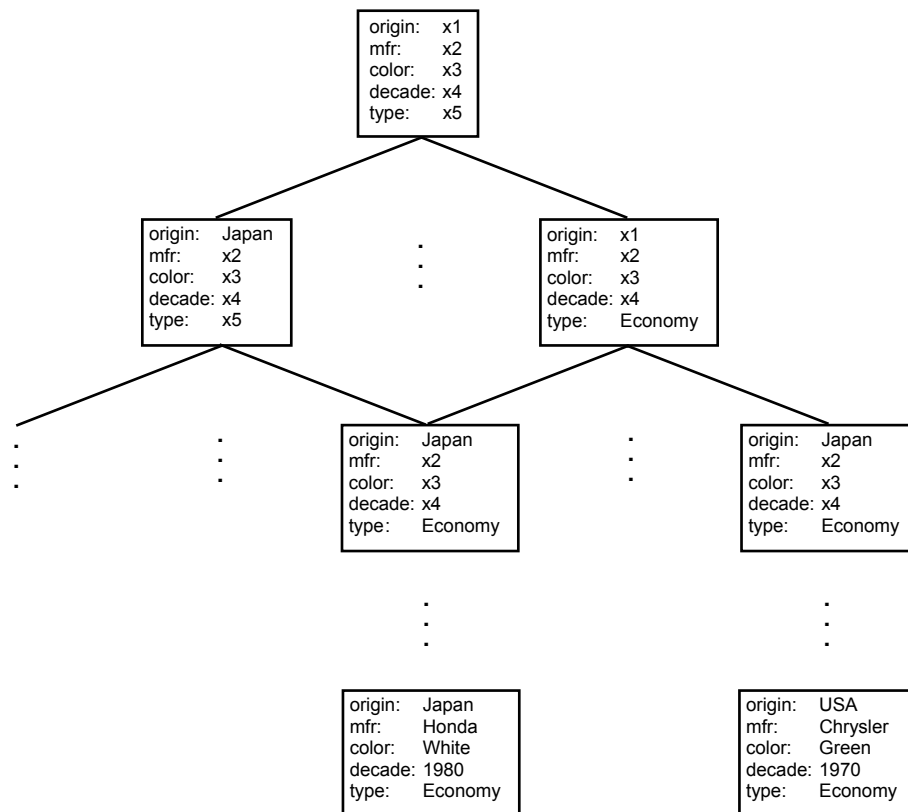
ปัญหาการเรียนรู้ที่เราสนใจคือ กำหนดค่าที่เป็นไปได้ของสล็อต ตัวอย่างบวกและตัวอย่างลบให้ จงหาค่าอธิบายโมเดลที่**สอดคล้องกับ** (*consistent with*) ตัวอย่าง (อธิบายตัวอย่างบวกและไม่อธิบายตัวอย่างลบ)

มีนัยทั่วไปกว่า

และ

จำเพาะกว่า

วิธีการเรียนรู้เวอร์ชันสเปซนี้มองว่าการเรียนรู้คือการค้นหาในปริภูมิค้นหาที่เรียกว่า **ปริภูมิโมเดล** (*concept space*) ซึ่งเป็นปริภูมิที่มีสมาชิกแต่ละตัวเป็นคำอธิบายในรูปของกรอบโดยที่สมาชิกเหล่านี้มีลำดับบางส่วน (partial ordering) ในลำดับนี้สมาชิกตัวที่**มีนัยทั่วไปกว่า** (*more general*) จะอยู่ด้านบนของสมาชิกตัวที่**จำเพาะกว่า** (*more specific*) ดังแสดงในรูปที่ 6-21 โดยที่ตัวอักษรเล็ก (x1, x2, x3, x4 และ x5) แสดงตัวแปรซึ่งสามารถแทนที่ด้วยค่าคงที่ได้ ส่วนตัวอักษรใหญ่และตัวเลข (เช่น Japan, Economy, 1980) แสดงค่าคงที่



รูปที่ 6-21 ปริภูมิโมเดล

ตัวที่อยู่บนสุดในรูปแสดง**โมเดลมีนัยทั่วไปที่สุด** (*most general concept*) ส่วนตัวที่อยู่ล่างสุดแสดง**โมเดลจำเพาะที่สุด** (*most specific concept*) ซึ่งเป็นตัวอย่างหนึ่งๆ และตัวที่

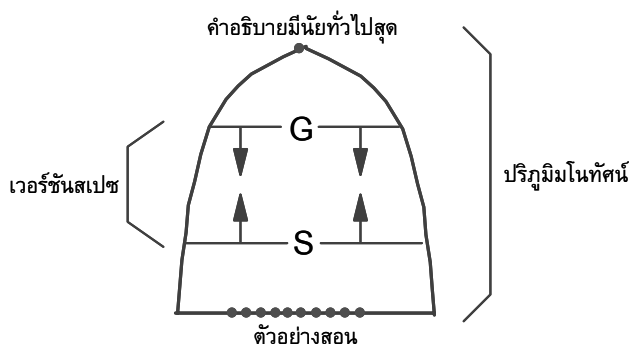
เป็นคำอธิบายโมทัศน์เป้าหมาย (*target concept description*) จะอยู่ระหว่างบนสุดกับล่างสุด วิธีการเรียนรู้เวอร์ชันสเปซคือการสร้างเซตย่อยประกอบด้วย *สมมติฐาน (hypothesis)* ที่อยู่ในปริภูมิโมทัศน์ที่สอดคล้องกับตัวอย่างสอน และเรียกเซตย่อยนี้ว่า *เวอร์ชันสเปซ (version space)*

เวอร์ชันสเปซที่สร้างขึ้นนี้จะต้องประกอบด้วยสมมติฐาน (คำอธิบาย) ที่สอดคล้องกับตัวอย่างที่เคยพบมาทั้งหมด วิธีการสร้างเวอร์ชันสเปซที่ทำได้วิธีหนึ่งคือการแจงสมาชิกทุกตัวในปริภูมิโมทัศน์ แล้วตรวจสอบกับตัวอย่างสอนทุกตัวที่รับเข้ามา หากสมาชิกตัวใดไม่สอดคล้องกับตัวอย่างก็ตัดทิ้งไป คงไว้เฉพาะตัวที่สอดคล้องเท่านั้น อย่างไรก็ตามปริภูมิโมทัศน์มีขนาดใหญ่มาก วิธีการนี้จึงไม่มีประสิทธิภาพ ตัวอย่างเช่นในกรณีของปัญหาในรูปที่ 6-21 เมื่อพิจารณาค่าที่เป็นไปได้ในสล็อตแต่ละตัวตามตารางที่ 6-11 จะเห็นว่าปริภูมิโมทัศน์มีขนาดเท่ากับ $((5+1)(7+1)(4+1)(6+1)(3+1)) = 6,720$ และในกรณีที่จำนวนสล็อตมีมากขึ้นเช่น 10 ตัว และสล็อตแต่ละตัวมีค่าที่เป็นไปได้มากขึ้นเช่น 10 ค่า จะได้ว่าปริภูมิโมทัศน์จะยังมีขนาดใหญ่ขึ้นมาก ($\approx 2.6 \times 10^{10}$)

วิธีการเรียนรู้เวอร์ชันสเปซจะใช้วิธีการแทนสเปซด้วยวิธีที่ประหยัดและมีประสิทธิภาพในการค้นหา โดยจะใช้เซตย่อย 2 เซตเรียกว่าเซต G และเซต S

- เซต G ประกอบด้วยคำอธิบายมีนัยทั่วไปที่สุดที่ยังสอดคล้องกับตัวอย่างที่เคยพบมาทั้งหมด
- เซต S ประกอบด้วยคำอธิบายจำเพาะสุดที่ยังสอดคล้องกับตัวอย่างที่เคยพบมาทั้งหมด

เวอร์ชันสเปซจะอยู่ระหว่างเซต G กับ S ดังแสดงในรูปที่ 6-22



รูปที่ 6-22 เวอร์ชันสเปซ

หลักการของเวอร์ชันสเปซคือทุกครั้งที่เราได้รับตัวอย่างบวกตัวใหม่เราจะทำให้ S มีนัยทั่วไป (general) มากขึ้นและทุกครั้งที่ได้รับตัวอย่างลบเราจะทำให้ G จำเพาะ

(specific) มากขึ้น จนในที่สุด S และ G ลู่เข้าสู่ค่าเดียวกันที่เป็นคำอธิบายโมโนทัศน์เป้าหมาย อัลกอริทึมการเรียนรู้ของเวอร์ชันสเปซเป็นดังตารางที่ 6-12 นี้

ตารางที่ 6-12 อัลกอริทึมการเรียนรู้เวอร์ชันสเปซ

Algorithm: Version-Space-Candidate-Elimination

1. $G := \{\text{most general description}\}$
2. $S := \{\text{first positive example}\}$
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 $S \neq G$ THEN
 examples were inconsistent
 ELSE goto 3.

6.4.1 ตัวอย่างการเรียนรู้โมโนทัศน์ car

กำหนดเซตตัวอย่างสอนที่ประกอบด้วยตัวอย่างบวกและตัวอย่างลบดังรูปที่ 6-23 อัลกอริทึมในตารางที่ 6-12 จะเรียนรู้ดังต่อไปนี้

<div>origin: Japan mfr: Honda color: Blue decade: 1980 type: Economy</div> <div>(+)</div>	<div>origin: Japan mfr: Toyota color: Green decade: 1970 type: Sports</div> <div>(-)</div>	<div>origin: Japan mfr: Toyota color: Blue decade: 1990 type: Economy</div> <div>(+)</div>
<div>origin: USA mfr: Chrysler color: Red decade: 1980 type: Economy</div> <div>(-)</div>	<div>origin: Japan mfr: Honda color: White decade: 1980 type: Economy</div> <div>(+)</div>	

รูปที่ 6-23 ตัวอย่างสอนของโมโนทัศน์ car

- จากตัวอย่างบวก 3 ตัวและตัวอย่างลบ 2 ตัวตามรูปด้านบน เราเริ่มด้วยการสร้าง G และ S ตามตัวอย่างแรกได้

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

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

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

คลุม

- ตัวอย่างที่ 2 เป็นตัวอย่างลบ ดังนั้นเราทำการแจกจำเพาะของ G เพื่อให้เวอร์ชันสเปซอธิบายหรือคลุม (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) เรากำจัดคำอธิบายใน G ที่ไม่สอดคล้องกับตัวอย่างนี้

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

และทำการวางนัยทั่วไปของ S ให้รวมตัวอย่างนี้

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

ที่จุดนี้เราได้เวอร์ชันสเปซที่แสดง "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 $S=G$ แสดง "Japanese economy car"

6.4.2 ข้อจำกัดของเวอร์ชันสเปซ

ดังที่แสดงในตัวอย่างด้านบนนี้ เวอร์ชันสเปซสามารถเรียนรู้ได้จากตัวอย่างที่สอน อย่างไรก็ตาม ดีเวอร์ชันสเปซก็ยังมีข้อจำกัดดังต่อไปนี้

- อัลกอริทึมเรียนรู้เป็นแบบทำน้อยสุด (least-commitment algorithm) กล่าวคือในแต่ละขั้นตอนเวอร์ชันสเปซจะถูกตัดเล็มให้เล็กลงน้อยที่สุดเท่าที่เป็นไปได้ ดังนั้นถึงแม้ว่าตัวอย่างบวกทุกตัวเป็น Japanese cars ก็ตาม อัลกอริทึมก็จะไม่ตัดความน่าจะเป็นที่มันทัศน์อาจจะรวม car อื่นๆ ทั้งจนกระทั่งพบตัวอย่างลบ ซึ่งหมายถึงเวอร์ชันสเปซจะเรียนรู้ไม่สำเร็จถ้าไม่มีตัวอย่างลบเลย
- กระบวนการค้นหาเป็นการค้นหาแนวกว้างแบบทั้งหมด (exhaustive breadth-first search) ซึ่งเห็นได้จากการปรับค่าของเซต G ที่จะทดลองทำที่สล็อตทุกตัวให้ได้ทุกแบบที่เป็นไปได้ ดังนั้นทำให้อัลกอริทึมมีประสิทธิภาพต่ำในกรณีที่มีปริภูมิใหญ่มากๆ ซึ่งอาจทำให้ดีขึ้นโดยใช้วิธีสุ่มเข้ามาช่วยในการค้นหาโดยลองเปลี่ยนตัวแปรเป็นค่าคงที่ในบางสล็อตที่น่าจะนำไปสู่คำตอบก่อน เป็นต้น
- เซต S ประกอบด้วยสมาชิกเพียงตัวเดียว เพราะว่าตัวอย่างบวก 2 ตัวใดๆ มีการวางนัยทั่วไปเพียงหนึ่งเดียว ดังนั้นเวอร์ชันสเปซจึงไม่สามารถเรียนมนโนทัศน์แบบ 'หรือ' (disjunctive concept) ซึ่งเป็นมนโนทัศน์ที่อยู่ในรูปของ or เช่น "Japanese economy car or Japanese sport car"
- ข้อจำกัดอีกอย่างของเวอร์ชันสเปซคือไม่สามารถจัดการกับตัวอย่างที่มีสัญญาณรบกวน (noisy example) ซึ่งเป็นตัวอย่างที่มีข้อมูลบางส่วนผิดพลาด เช่นถ้าตัวอย่างตัวที่ 3 ในรูปที่ 6-23 (Japan Toyota Blue 1990 Economy) เราให้ประเภทผิดเป็นตัวอย่างลบ (-) อัลกอริทึมจะไม่สามารถเรียนมนโนทัศน์ "Japanese economy car" ได้ถูกต้อง

6.5 การเรียนรู้ต้นไม้ตัดสินใจ

ประเภท

การเรียนรู้ต้นไม้ตัดสินใจ (decision tree learning) [Quinlan, 1986; Quinlan, 1993] เป็นการเรียนรู้ที่ใช้การแทนความรู้ในรูปของต้นไม้ตัดสินใจ ใช้สำหรับจำแนกประเภทของตัวอย่าง วิธีการเรียนรู้คล้ายกับการเรียนรู้เวอร์ชันสเปซโดยเริ่มจากการป้อนตัวอย่างเข้าไปในระบบ ซึ่งตัวอย่างที่ป้อนให้เป็นตัวอย่างบวกกับตัวอย่างลบก็ได้และนอกจากนั้นเรายังสามารถป้อนตัวอย่างที่มากกว่า 2 **ประเภท (class)** ได้ กล่าวคือแทนที่จะมีแค่บวกกับลบ ก็สามารถมีได้หลายประเภท เช่นในการรู้จำตัวอักษร จะมีตัวอย่างมาจากหลายประเภทที่แตกต่างกันคือประเภท 'ก', ประเภท 'ข', ประเภท 'ค', ประเภท 'ง' ฯลฯ แต่เพื่อให้ง่ายต่อการอธิบาย ตัวอย่างที่จะยกให้ดูต่อไปนี้มีเพียง 2 ประเภทเท่านั้น โดยเราจะใช้ปัญหาการผิ้งแดดเป็นตัวอย่างอธิบาย

ปัญหาการผิ้งแดด: เราไปเที่ยวที่ชายหาดและพบว่าคนที่ไปผิ้งแดดตามชายหาด บางคนก็จะมีผิวเปลี่ยนเป็นสีแทน แต่บางคนต้องได้รับความทรมานจากผิวไหม้ เราต้องการหาว่าอะไรคือปัจจัยที่ทำให้คนที่ไปผิ้งแดดตามชายหาดแล้วผิวไหม้หรือไม่ไหม้ โดยข้อมูลที่สังเกตได้ประกอบด้วยความแตกต่างของสีผิว น้ำหนัก ส่วนสูงของผู้ที่ไปผิ้งแดด และการใช้โลชั่น ซึ่งบางคนก็ใช้โลชั่น บางคนก็ไม่ใช้

สมมติว่าเราบันทึกข้อมูลของตัวอย่างสอนได้ตามตารางที่ 6-13 เพื่อใช้สร้างต้นไม้ตัดสินใจ

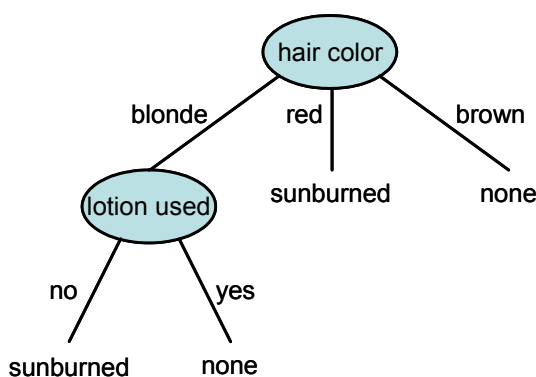
ตารางที่ 6-13 ตัวอย่างสอนที่สังเกตได้

						ประเภท
						↓
คุณสมบัติ → คำ {	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

แถวแรกสุดในตารางแสดง **คุณสมบัติ (attribute)** ของข้อมูลซึ่งประกอบด้วยชื่อ (Name) สีผม (Hair) ส่วนสูง (Height) น้ำหนัก (Weight) และการใช้โลชั่น (Lotion) ส่วนสุดท้ายแสดงแทนประเภทของตัวอย่าง คุณสมบัติ Name ไว้สำหรับอ้างอิงตัวอย่างและไม่มีผลต่อการจำแนกข้อมูล เราจึงจะไม่ใช้ Name ในการเรียนรู้ด้านล่างนี้ แต่ละแถวในตารางนอกเหนือจากแถวแรกแทนตัวอย่างหนึ่งตัว เช่นแถวที่สองแสดงตัวอย่างของคนที่ชื่อ Sarah ซึ่งมีสีผม ส่วนสูง น้ำหนัก และการใช้โลชั่น เป็น blond, average, light และ no ตามลำดับ ตัวอย่างนี้อยู่ในประเภท sunburned เป็นต้น

เมื่อเราได้ข้อมูลตัวอย่างทั้ง 8 ตัวแล้ว สิ่งที่เราต้องการทำก็คือทำการวางนัยทั่วไปของตัวอย่างเพื่อสร้างเป็นโมเดลสำหรับทำนายประเภทของข้อมูลของคนอื่นที่ไม่ได้บันทึกไว้ วิธีที่ง่ายที่สุดก็คือการเรียนรู้โดยการจำ และเมื่อมีตัวอย่างในอนาคตที่เรายังไม่ทราบประเภทและถ้าต้องการทำนาย เราก็นำตัวอย่างนั้นมาเปรียบเทียบกับตัวอย่างสอนในตาราง ถ้าตัวอย่างที่นำมาเปรียบเทียบมีคุณสมบัติตรงกับข้อมูลในตาราง เราก็นำประเภทของตัวอย่างสอนที่ตรงกันทำนายให้กับตัวอย่างนั้น อย่างไรก็ตามวิธีการนี้ทำงานได้ไม่ดีนักเนื่องจากว่าโอกาสที่เราจะพบตัวอย่างทดสอบที่ตรงกับตัวอย่างสอนมีน้อย สมมติว่าสีผมมีค่าที่เป็นไปได้ทั้งหมด 3 ค่าคือ blonde, brown, red ส่วนสูงมีได้ 3 ค่าคือ tall, average, short น้ำหนักมีได้ 3 ค่าคือ heavy, average, light และการใช้โลชั่นมีได้ 2 ค่าคือ yes, no เราจะพบว่าความน่าจะเป็นที่ตัวอย่างทดสอบจะตรงกับตัวอย่างสอนมีค่าเท่ากับ $8/(3 \times 3 \times 3 \times 2) = 15\%$ (สมมติว่าความน่าจะเป็นที่ค่าแต่ละค่าสำหรับคุณสมบัติหนึ่งๆ มีความน่าจะเป็นที่จะเกิดขึ้นเท่ากัน)

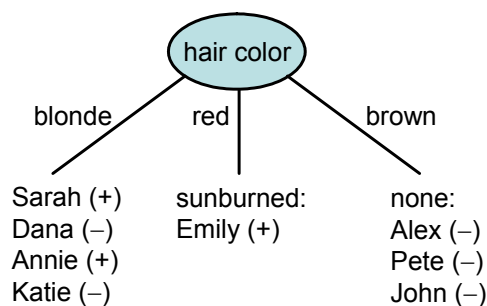
การเรียนรู้ต้นไม้ตัดสินใจจะทำการวางนัยทั่วไปของข้อมูลโดยสร้างเป็นโมเดลอยู่ในรูปต้นไม้ตัดสินใจ ตัวอย่างของต้นไม้ตัดสินใจแสดงในรูปที่ 6-24



รูปที่ 6-24 ตัวอย่างของต้นไม้ตัดสินใจ

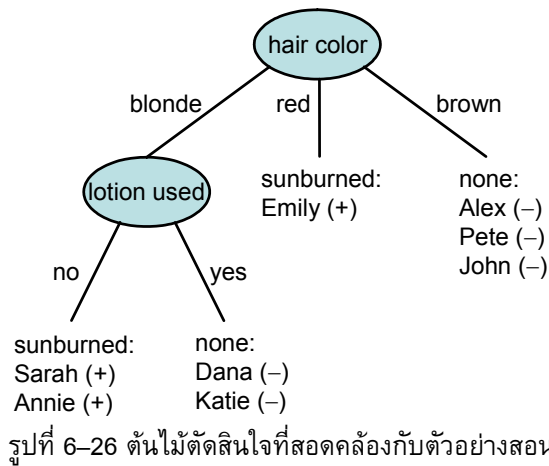
ต้นไม้ตัดสินใจประกอบด้วย**บัพ (node)** และ**กิ่ง (link)** ที่ต่อกับบัพ บัพที่ปลายสุดเรียกว่า**บัพใบ (leaf node)** หรือเรียกย่อๆ ว่าใบ บัพแสดงคุณสมบัติและกิ่งแสดงค่าของคุณสมบัตินั้น ใบ (leaf) แสดงประเภท การสร้างต้นไม้ตัดสินใจทำโดยสร้างบัพที่ละบัพเพื่อตรวจสอบคุณสมบัติของตัวอย่าง แล้วแยกตัวอย่างลงตามค่าของกิ่ง ทำจนกระทั่งตัวอย่างในใบแต่ละใบอยู่ในประเภทเดียวกันทั้งหมด

สมมติว่าเราเลือกคุณสมบัติ hair color เป็นบัพแรกหรือบัพรากของต้นไม้ เราจะแยกตัวอย่างลงตามกิ่งของบัพ hair color ตัวอย่างใดที่มีค่าของ hair color เป็น blonde ก็แยกลงตามกิ่งซ้าย ถ้าเป็น red ก็แยกลงตามกิ่งกลาง และถ้าเป็น brown ก็แยกลงตามกิ่งขวา ผลที่ได้แสดงในรูปที่ 6-25 เครื่องหมาย + และ - แสดงประเภท sunburned และ none ตามลำดับ



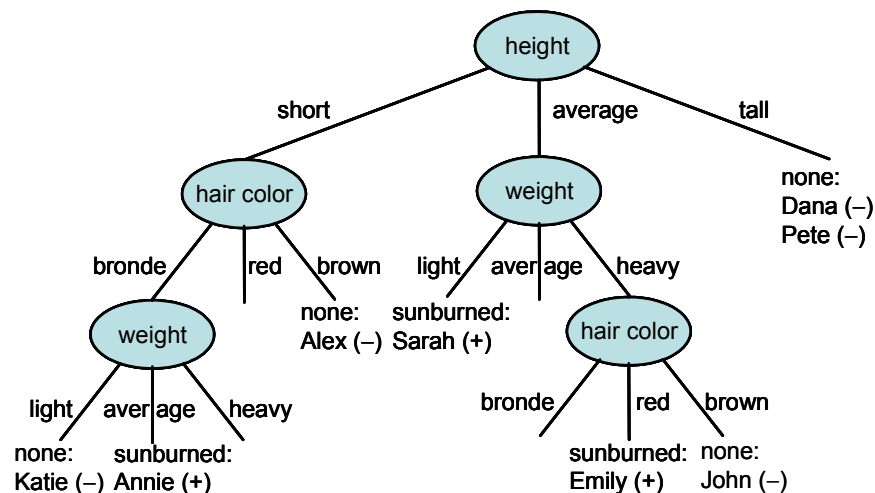
รูปที่ 6-25 ผลการแยกตัวอย่างลงตามกิ่งของบัพ 'hair color'

ต้นไม้ที่สร้างขึ้นนี้แยกตัวอย่างได้ในกรณีที่ hair color เป็น red (ตัวอย่างทุกตัวมีประเภทเป็น sunburned) และ brown (ตัวอย่างทุกตัวมีประเภทเป็น none) แต่ในกรณีที่ hair color เป็น blonde ยังแยกตัวอย่างไม่ได้ กล่าวคือมีตัวอย่างที่เป็นทั้ง sunburned และ none ปะปนกันอยู่ ในกรณีที่ hair color มีค่าเป็น brown เราสรุปได้ว่าตัวอย่างทุกตัวจะมีประเภทเป็น none หมดเนื่องจากตัวอย่างสอนทุกตัวที่ลักษณะเช่นนั้น หรือในกรณีที่ hair color มีค่าเป็น red เราก็สรุปได้ในทำนองเดียวกันว่าตัวอย่างจะมีประเภทเป็น sunburned แต่ในกรณีของ hair color เป็น blonde เราต้องการคุณสมบัติอื่นเข้าช่วยจำแนกประเภทตัวอย่างต่อไป ที่จุดนี้สมมติว่าเราใช้คุณสมบัติ lotion เพื่อแยกข้อมูลในกิ่งของ blonde ต่อไป ผลที่ได้แสดงในรูปที่ 6-26



ต้นไม้ตัดสินใจในรูปที่ 6-26 ด้านบนนี้สอดคล้องกับตัวอย่างสอนทุกตัว หมายความว่า ถ้านำตัวอย่างสอนมาตรวจสอบด้วยต้นไม้ตัดสินใจ ต้นไม้จะทำนายประเภทได้ถูกต้องทุกตัว การตรวจสอบทำได้โดยดูว่าตัวอย่างมี 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 ต้นไม้ในรูปที่ 6-27 ก็เป็นต้นไม้อีกต้นหนึ่งที่สอดคล้องกับตัวอย่าง



รูปที่ 6-27 ต้นไม้ตัดสินใจอีกต้นหนึ่งที่มีความซับซ้อนมากกว่าต้นแรก

เมื่อเราพิจารณาด้านไม้ต้นแรกในรูปที่ 6-26 และด้านที่สองในรูปที่ 6-27 เราพบว่าด้านไม้ต้นแรกน่าจะถูกต้องมากกว่าด้านที่สอง เนื่องจากว่าในต้นแรกนั้นใช้คุณสมบัติสีผืนและการใช้โลชันในการจำแนกข้อมูล ซึ่งน่าจะเป็นไปได้เพราะสีผืนมีความสัมพันธ์อย่างมากกับความแข็งแรงของผิวเรา คนที่มีผผสีน้ำตาลน่าจะมีผิวที่แข็งแรงไปฝั่งแดดแล้วมักจะไม่เป็นอะไร ส่วนผผสีแดงมีผิวบอบบาง และผผสีบรอนซ์มีผิวปานกลางซึ่งจะขึ้นกับการใช้โลชันหรือไม่ใช้ ถ้าใช้ไปฝั่งแดดก็จะเป็นอะไร ถ้าไม่ใช้ไปฝั่งแดดแล้วผิวจะไหม้ ส่วนด้านไม้ต้นที่สองเราไม่สามารถอธิบายได้ว่าทำไมส่วนสูงที่ใช้เป็นบัพราคหรือน้ำหนักที่บัพในระดับถัดมาจึงมีความสำคัญต่อการที่ผิวจะไหม้หรือไม่ไหม้

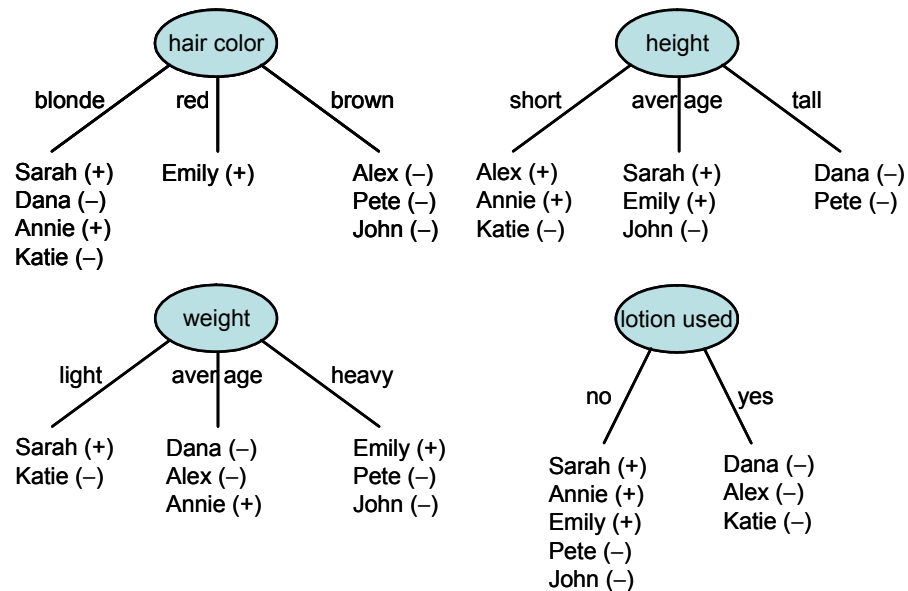
ความแตกต่างที่เห็นได้เด่นชัดอีกประการของต้นไม้ทั้งสองคือจำนวนบัพภายในต้นไม้ จะเห็นได้ว่าจำนวนบัพของต้นไม้ต้นที่สองมีจำนวนมากกว่า หรือกล่าวอีกนัยหนึ่งคือต้นไม้ต้นที่สองมีความซับซ้อนมากกว่า หรือกล่าวได้ว่าต้นไม้ต้นแรกมีขนาดเล็กกว่าต้นไม้ต้นที่สองโดยที่ขนาดวัดจากจำนวนบัพภายในต้นไม้

มีดโกนของ
อ็อกแคม

ในการเรียนรู้ของเครื่องนั้น เรามีอิทธิพลจากตัวหนึ่งที่นิยมใช้กันและพบว่าทำงานได้อย่างดีในหลายกรณีเรียกว่า *มีดโกนของอ็อกแคม (occam's razor)* เมื่อเรานำมีดโกนของอ็อกแคมมาใช้ในการเลือกต้นไม้ตัดสินใจ เราก็จะได้ว่า “ต้นไม้ตัดสินใจขนาดเล็กที่สุดที่สอดคล้องกับตัวอย่างสอนคือต้นไม้ตัดสินใจที่ดีที่สุด” อย่างไรก็ตามถ้าเราจะหาต้นไม้ตัดสินใจที่มีขนาดเล็กที่สุดที่สอดคล้องกับตัวอย่างสอนก็ไม่สามารถทำได้โดยง่าย เราต้องสร้างต้นไม้ตัดสินใจจำนวนมาก โดยเริ่มจากต้นไม้ที่มีจำนวนบัพ 1 บัพทุกต้นที่เป็นไปได้แล้วดูว่ามีต้นไหนหรือไม่ที่สอดคล้องกับตัวอย่างสอน ถ้าไม่มีก็เพิ่มจำนวนบัพเป็น 2 บัพ ทำอย่างนี้ไปจนกระทั่งพบต้นไม้ตัดสินใจที่สอดคล้องกับตัวอย่าง เราพบว่าวิธีการนี้จะมีจำนวนต้นไม้ที่ต้องสร้างเป็นฟังก์ชันเลขยกกำลังของจำนวนคุณสมบัติซึ่งไม่เหมาะกับการใช้งานจริง

6.5.1 ฟังก์ชันเกณฑ์สำหรับการเลือกบัพทดสอบ

ส่วนนี้จะกล่าวถึงวิธีการเลือกบัพเพื่อสร้างต้นไม้ตัดสินใจโดยใช้หลักการว่า เนื่องจากจุดมุ่งหมายของการสร้างต้นไม้คือเพื่อจำแนกประเภทของข้อมูลเพื่อให้ตัวอย่างในแต่ละบัพไปอยู่ในประเภทเดียวกันทั้งหมด ดังนั้นบัพที่ดีควรเป็นบัพที่แยกตัวอย่างออกเป็นเซตย่อยตามกิ่งของบัพนั้นและเซตย่อยในแต่ละกิ่งประกอบด้วยสมาชิกที่ส่วนใหญ่เป็นประเภทเดียวกันมากที่สุด ตัวอย่างในรูปที่ 6-28 แสดงผลของบัพทดสอบแต่ละบัพ



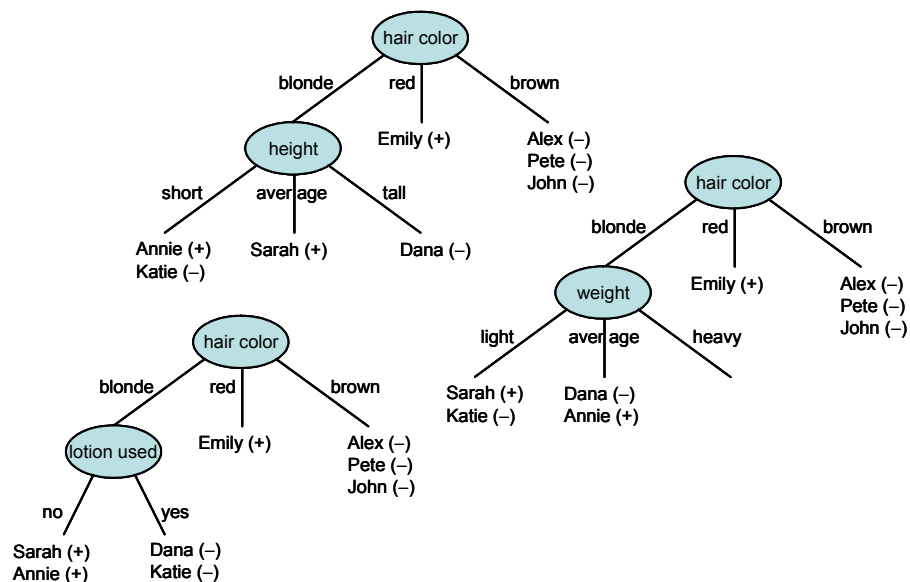
รูปที่ 6-28 ผลของบัพทดสอบแต่ละบัพในการแยกตัวอย่างเพื่อเลือกบัพราก

ดังแสดงในรูปด้านบน บัพแต่ละบัพแยกตัวอย่างได้ดีต่างกันดังนี้

- ในกรณีของบัพทดสอบเป็น hair color สามารถแยกตัวอย่างเป็น 3 เซตย่อย เซตย่อยแรก (blonde) มีตัวอย่างของ 2 ประเภทปนกันอยู่ ส่วนเซตย่อยที่ 2 (red) และ 3 (brown) มีตัวอย่างของประเภท sunburned และ none อยู่อย่างเดียวตามลำดับ ซึ่งกรณีนี้ hair color แยกตัวอย่างได้ดีเมื่อเทียบกับบัพอื่นด้านล่างนี้
- ในกรณีของบัพทดสอบเป็น height สามารถแยกตัวอย่างเป็น 3 เซตย่อย เซตย่อยแรก (short) และเซตย่อยที่ 2 (average) มีตัวอย่าง 2 ประเภทปนกันอยู่ในแต่ละเซต ส่วนเซตย่อยที่ 3 (tall) มีตัวอย่างของ none อยู่อย่างเดียว จะเห็นว่ากรณีนี้แยกตัวอย่างไม่ดีเท่ากรณีของ hair color
- ในกรณีของบัพทดสอบเป็น weight สามารถแยกตัวอย่างเป็น 3 เซตย่อย เซตย่อยทั้งสามเซต (light, average, heavy) ต่างก็มีตัวอย่าง 2 ประเภทปนกันอยู่ ซึ่งกรณีนี้เป็นกรณีที่แย่ที่สุด
- ในกรณีของบัพทดสอบเป็น lotion used สามารถแยกตัวอย่างเป็น 2 เซตย่อย เซตย่อยแรก (no) มีตัวอย่างของ 2 ประเภทปนกัน ส่วนเซตย่อยที่ 2 (yes) มีตัวอย่างของ none อยู่อย่างเดียว และในเซตย่อยแรกสมาชิกส่วนใหญ่ของเซตนี้เป็น sunburned (+) เกือบทั้งหมด ซึ่งเมื่อเทียบกับกรณีแรกของบัพ hair color ถือได้ว่ามีความสามารถในการแยกตัวอย่างได้ใกล้เคียงกัน

ดังจะเห็นได้ในตัวอย่างด้านบนนี้ เราพอจะเปรียบเทียบได้ว่าบัพหนึ่งๆ มีความสามารถในการแยกตัวอย่างดีกว่าบัพอีกบัพหนึ่งหรือไม่ แต่ในบางกรณีเช่น hair color กับ lotion used เราอาจบอกความแตกต่างไม่ได้ ดังนั้นเราจำเป็นต้องหาการวัดที่สามารถบอกความต่างได้อย่างชัดเจนโดยการนิยามฟังก์ชันเพื่อวัดประสิทธิภาพของบัพออกเป็นค่าที่วัดได้อย่างละเอียด ซึ่งจะกล่าวต่อไป

ณ จุดนี้ เพื่อให้เข้าใจถึงการสร้างต้นไม้ตัดสินใจจะขอสมมติว่าเรามีฟังก์ชันนั้นอยู่ และสมมติว่าระหว่างบัพ hair color กับ lotion used ค่าฟังก์ชันของ hair color ดีกว่า และได้รับเลือกเป็นบัพราก ในขั้นตอนต่อไปก็คือเราต้องพิจารณาต่อว่าในแต่ละกิ่งของบัพราก มีกิ่งใดหรือไม่ที่ยังมีตัวอย่างจากหลายประเภทปะปนกันอยู่ ถ้ามีเราต้องเพิ่มบัพของคุณสมบัติอื่นเพื่อช่วยแยกตัวอย่างที่ยังปะปนกันอยู่ต่อไป ในกรณีของบัพ hair color ในรูปที่ 6-28 กิ่ง blonde เท่านั้นที่ยังมีตัวอย่างจากหลายประเภทปะปนกัน เราจึงจำเป็นต้องเพิ่มบัพต่อไปโดยทดลองเพิ่มคุณสมบัติที่เหลือทั้งสาม (height, weight และ lotion used) ผลที่ได้แสดงในรูปที่ 6-29



รูปที่ 6-29 ผลของบัพทดสอบแต่ละบัพในการแยกตัวอย่างเพื่อเลือกบัพต่จากกิ่ง blonde

จากรูปด้านบนจะเห็นได้ว่าบัพ lotion used เป็นบัพที่แยกตัวอย่างออกเป็นเซตย่อยโดยที่แต่ละเซตย่อยมีสมาชิกอยู่ในประเภทเดียวกัน ดังนั้นบัพ lotion used ถูกเลือกในขั้นตอนนี้

6.5.2 ฟังก์ชันเกน

ฟังก์ชันเกน

ฟังก์ชันที่ใช้วัดความสามารถในการแยกตัวอย่างของบัพทดสอบที่มีประสิทธิภาพมาก ฟังก์ชันหนึ่งคือ **ฟังก์ชันเกน (Gain function)** ฟังก์ชันเกนนี้ใช้ในการตัดสินใจเลือกคุณสมบัติที่จะใช้เป็นรากหรือบัพในต้นไม้โดยการคำนวณค่าเกนของคุณสมบัติแต่ละตัวเมื่อทดลองใช้คุณสมบัตินั้นแบ่งตัวอย่าง แล้วเลือกคุณสมบัติที่มีค่าเกนสูงที่สุดมาเป็นรากหรือบัพ ค่าเกนนี้คำนวณได้โดยใช้ความรู้จาก**ทฤษฎีสารสนเทศ (information theory)** ซึ่งมีสาระสำคัญคือค่าสารสนเทศหรือของข้อมูลขึ้นอยู่กับค่าความน่าจะเป็นของข้อมูลซึ่งสามารถวัดอยู่ในรูปของบิต (bits) จากสูตร

ทฤษฎีสารสนเทศ
และ
เอนโทรปี

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

เอนโทรปี

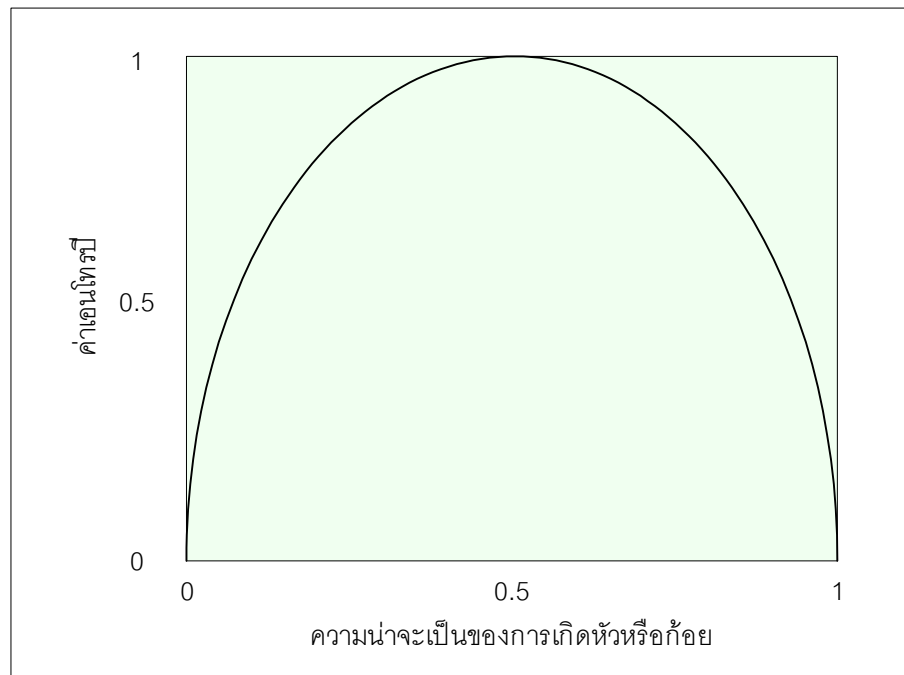
ถ้าให้ชุดของข้อมูล M ประกอบด้วยค่าที่เป็นไปได้ คือ $\{m_1, m_2, \dots, m_n\}$ และให้ความน่าจะเป็นที่จะเกิดค่า m_i มีค่าเท่ากับ $P(m_i)$ จะได้ว่าค่า**เอนโทรปี (entropy)** ของ M ซึ่งใช้วัดค่าสารสนเทศโดยเฉลี่ยเพื่อระบุประเภทของข้อมูลสามารถเขียนแทนด้วย $I(M)$ คำนวณได้จากสูตร

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

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 แสดงให้เห็นว่าค่าเอนโทรปีที่น้อยจะบ่งบอกว่าข้อมูลชุดนั้นมีความแตกต่างกันน้อยหรือเกือบจะเป็นพวกเดียวกัน แต่ถ้าค่าเอนโทรปีสูงจะบ่งบอกว่าข้อมูลชุดนั้นมีความแตกต่างกันมากหรือประกอบด้วยตัวอย่างหลายพวกที่มีจำนวนใกล้เคียงกัน



รูปที่ 6-30 ค่าเอนโทรปีของการโยนหัวโยนก้อย

ในการเลือกคุณสมบัติที่จะมาเป็นบัพราจะอาศัยค่าเอนโทรปี ซึ่งคำนวณจากค่าเอนโทรปีทั้งหมดของชุดข้อมูลนั้นลบด้วยค่าเอนโทรปีหลังจากเลือกคุณสมบัติใดคุณสมบัติหนึ่งเป็นราก ค่าเอนโทรปีหลังจากแบ่งตามคุณสมบัติที่เลือกแล้วคำนวณได้จาก ค่าผลรวมของผลคูณระหว่างค่าเอนโทรปีของแต่ละบัพกับอัตราส่วนของตัวอย่างในแต่ละกิ่งต่อตัวอย่างทั้งหมดที่บัพนั้นๆ

ถ้าให้ข้อมูลสอนคือ T และคุณสมบัติที่เป็นบัพคือ X และมีค่าทั้งหมดที่เป็นไปได้ n ค่า บัพปัจจุบันจะแบ่งตัวอย่าง T ออกตามกิ่งเป็น $\{t_1, t_2, \dots, t_n\}$ ตามค่าที่เป็นไปได้ของ X ดังนั้นจึงสามารถคำนวณค่าเอนโทรปีหลังจากแบ่งตามคุณสมบัติ X ดังนี้

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

ค่าเอนโทรปีของคุณสมบัติ X ที่ใช้แบ่งข้อมูลที่บัพหนึ่งๆ สามารถคำนวณได้จากการลบค่าเอนโทรปีทั้งหมดที่บัพนี้กับค่าเอนโทรปีที่ได้หลังจากแบ่งด้วยคุณสมบัติ X ดังนี้

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

พิจารณาคูณสมบัติ hair color ซึ่งแบ่งแยกข้อมูลได้ดังรูปที่ 6-25 ในกรณีที่ใช้ hair color เป็นบัพราค เราคำนวณหาค่าเกณฑ์ดังนี้

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

ในทำนองเดียวกัน คูณสมบัติอื่นจะมีค่ามาตรฐานเกณฑ์เป็นดังต่อไปนี้

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

จึงเลือกคูณสมบัติ hair color มาเป็นบัพแรกของต้นไม้ตัดสินใจ แต่คูณสมบัตินี้เพียงอย่างเดียวไม่สามารถแยกตัวอย่างบวกและลบออกจากกันได้ในกิ่งของค่าคูณสมบัติ blonde จึงต้องพิจารณาคูณสมบัติอื่นเพื่อแบ่งแยกข้อมูลที่ตกลงมายังกิ่งนี้ (ดูรูปที่ 6-29 ประกอบ) โดยค่าฟังก์ชันเกณฑ์ของคูณสมบัติแต่ละตัวมีค่าดังนี้

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

เราใช้คูณสมบัติ lotion ซึ่งมีค่าเกณฑ์มากที่สุดมาแบ่งแยกข้อมูลต่อไป ซึ่งพบว่าเมื่อแบ่งแยกแล้วข้อมูลที่ผ่านการแบ่งแยกมีกลุ่มเดียวกัน จึงได้ต้นไม้ตัดสินใจดังรูปที่ 6-24

6.5.3 การเปลี่ยนต้นไม้เป็นกฎ

ระบบปัญญาประดิษฐ์ส่วนใหญ่ใช้การแทนความรู้ในรูปของกฎ ดังนั้นเมื่อเราสร้างต้นไม้ตัดสินใจแล้วเราสามารถเปลี่ยนต้นไม้ให้อยู่ในรูปของกฎเพื่อใช้กับในกรณีที่ระบบของเราใช้การแทนความรู้ของกฎเป็นหลัก วิธีการแปลงต้นไม้เป็นกฎ "IF THEN" ทำได้โดยแสดงทุกเส้นทางเริ่มต้นจากบัพรากไปยังบัพใบและทุกครั้งที่พบบัพทดสอบก็ให้เพิ่มบัพทดสอบกับค่าของการทดสอบไว้ในส่วนของ IF และเมื่อพบบัพใบก็ให้ใส่ประเภทไว้ในส่วนของ THEN จากต้นไม้รูปที่ 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

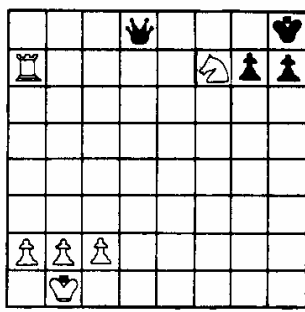
	Application	No. of Rules	Develop (Man Ys)	Maintain (Man Ys)	Learning Tools
MYCIN	Medical Diagnosis	400	100	N/A	N/A
XCON	VAX computer configuration	8,000	180	30	N/A
GASOIL	Hydrocarbon separation system configuration	2,800	1	0.1	ExpertEase and Extran7
BMT	Configuration of fire-protection equipment in buildings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

ยกตัวอย่างการเรียนรู้มโนทัศน์ fork ในการเล่นเกมหมากรุกสากล (chess) โดยให้ตัวอย่างบวกของ fork ดังด้านล่างนี้



รูปที่ 6-31 ตัวอย่างบวกของ fork

ตัวอย่างด้านบนนี้แสดงสถานการณ์ที่ “ม้าขาวโจมตีคิงดำและควีนดำพร้อมกัน” ในกรณีนี้ฝ่ายดำต้องยอมเสียควีน ไม่เช่นนั้นจะแพ้ จากตัวอย่างบวกตัวเดียวด้านบน อีบีแอลจะเรียนได้กฎดังนี้ “ถ้าตัวหมาก x โจมตีคิงกับตัวหมาก y ของฝ่ายตรงข้ามพร้อมกันแล้ว ฝ่ายตรงข้ามจะเสีย y ” ซึ่งวิธีการเรียนรู้กฎจะกล่าวต่อไป กฎที่เรียนรู้ได้นี้สามารถใช้กับสถานการณ์อื่นๆ นอกเหนือจากตัวอย่างสอนอีกด้วย กล่าวคือ x ไม่จำเป็นต้องเป็นม้าหรือ y ไม่จำเป็นต้องเป็นควีน นอกจากนั้นตำแหน่งของตัวหมากอื่นๆ ที่ไม่เกี่ยวข้องกับมโนทัศน์นี้ก็จะไม่ปรากฏในกฎ หมายความว่าตำแหน่งของตัวหมากอื่นๆ จะอยู่ที่ใดก็ได้ ตราบเท่าที่ตัวหมากเรากำลังโจมตีคิงและตัวหมากอีกตัวของฝ่ายตรงข้ามพร้อมกัน

จะเห็นได้ว่าประสิทธิภาพของอีบีแอลสูงมากเพราะใช้ตัวอย่างแค่ตัวเดียวก็สามารถทำการวางนัยทั่วไปได้ สาเหตุที่สามารถทำได้เช่นนี้เนื่องจากว่าในอีบีแอลนี้เราต้องให้ความรู้ใน

โดเมนกับระบบเรียนรู้แบบนี้ด้วย ความรู้ในโดเมนของหมากรุกสากลก็อย่างเช่นกฎการเล่นหมากรุก ตัวหมากแต่ละตัวเดินอย่างไร ม้า ถึง ควีน เดินอย่างไร การกินกันเกิดขึ้นได้เมื่อไร เกมจบเมื่อไร เป็นต้น ซึ่งกฎเหล่านี้เราสามารถให้ได้ไม่ยากนักเพราะมีเขียนไว้ในหนังสืออธิบายวิธีเล่นหมากรุกอยู่แล้ว อย่างไรก็ตามถ้าเราจะให้ความรู้ในโดเมนแล้วก็ได้ไม่ได้หมายความว่าเราไม่ต้องสอนอีบีแอล เปรียบเสมือนการเรียนรู้ของนักเรียนมัธยม แม้ว่าเราจะยกทฤษฎีเกี่ยวกับการเท่ากันของสามเหลี่ยมไปครบทุกทฤษฎีบท ก็ไม่ได้หมายความว่านักเรียนจะพิสูจน์การเท่ากันของสามเหลี่ยมสองรูปใดๆ ได้ทันที ครูก็ยังคงต้องยกตัวอย่างการพิสูจน์แสดงสามเหลี่ยม 2 รูปคู่หนึ่งๆ แล้วอธิบายว่าต้องใช้ทฤษฎีบทใดบ้างเพื่อการพิสูจน์และทำไมทฤษฎีบทเหล่านี้จึงพิสูจน์การเท่ากันของสามเหลี่ยมที่ยกตัวอย่างให้ดูได้ ซึ่งจะช่วยให้นักเรียนเข้าใจได้ดีขึ้น และเมื่อทำโจทย์การพิสูจน์สามเหลี่ยม 2 รูปที่ใช้ทฤษฎีบทซึ่งเหมือนกับครูยกตัวอย่างก็จะทำโจทย์ได้

กระบวนการเรียนรู้ของอีบีแอลประกอบด้วย 2 ขั้นตอนหลักคือ

- ใช้ความรู้ในโดเมนอธิบายให้ได้ว่าทำไมตัวอย่างจึงเป็นตัวอย่างของมโนทัศน์ในรูปของกฎ
 - ทำการวางนัยทั่วไปของกฎที่ได้เพื่อให้ใช้กับกรณีอื่นได้
- อินพุตและเอาต์พุตของอีบีแอลเป็นดังตารางที่ 6-15 ต่อไปนี้

ตารางที่ 6-15 อินพุตและเอาต์พุตของอีบีแอล

อินพุต:

- ตัวอย่างสอน (training example) – ตัวอย่างบวกของมโนทัศน์ที่จะสอน เช่นในกรณีของ fork ตัวอย่างสอนคือตำแหน่งตัวหมากบนกระดานที่เกิด fork
- มโนทัศน์เป้าหมาย (goal concept) – มโนทัศน์ที่จะสอนเช่นมโนทัศน์ fork
- เกณฑ์ดำเนินการ (operational criterion) – คำอธิบายที่สามารถนำไปใช้ได้ทันที เช่นในกรณีของ fork นั้น เปรดิเคต `attack-both(WKn,BK,BQ)` ไม่สามารถนำไปใช้ได้ทันทีที่ต้องแสดงในรูปของตำแหน่งตัวหมากบนกระดาน เช่น `position(WKn,f7)`, `position(BK,h8)`, `position(BQ,d8)` เป็นต้น
- ความรู้ในโดเมน (domain knowledge) – กฎต่างๆ ที่ใช้แสดงความสัมพันธ์ของวัตถุและการกระทำต่างๆ ในโดเมนนั้น เช่น กฎการเล่นหมากรุกสากล เป็นต้น

เอาต์พุต:

- การวางนัยทั่วไปของตัวอย่างสอนซึ่งเพียงพอสำหรับอธิบายมโนทัศน์เป้าหมายและสอดคล้องกับเกณฑ์ดำเนินการ

ตัวอย่างการเรียนรู้มโนทัศน์ cup

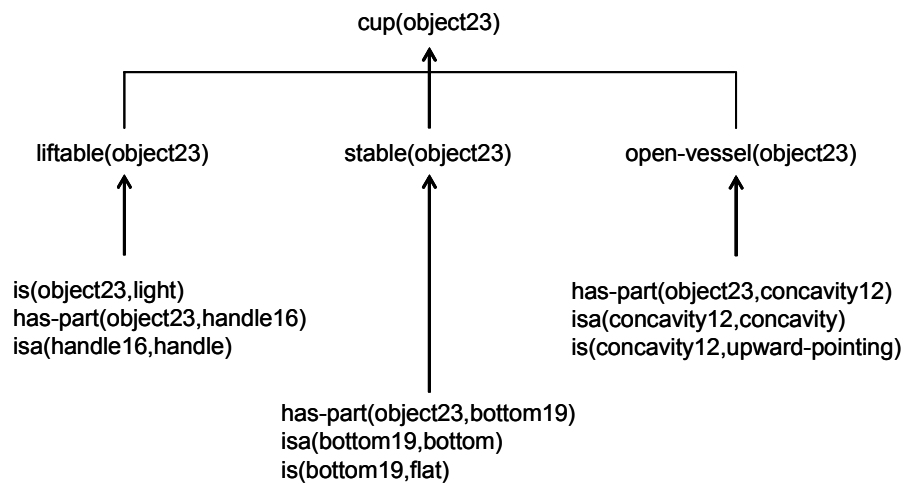
ตัวอย่างการเรียนรู้ที่จะยกมานี้เป็นตัวอย่างการเรียนรู้มโนทัศน์ cup (อะไรคือถ้วย) โดยมีอินพุตที่ให้ดังนี้

- ตัวอย่างบวก:
 $\text{owner}(\text{object23}, \text{ralph}), \text{has-part}(\text{object23}, \text{concavity12}),$
 $\text{isa}(\text{concavity12}, \text{concavity}), \text{is}(\text{concavity12}, \text{upward-pointing}),$
 $\text{has-part}(\text{object23}, \text{handle16}), \text{isa}(\text{handle16}, \text{handle}), \text{is}(\text{object23}, \text{light}),$
 $\text{color}(\text{object23}, \text{brown}), \text{has-part}(\text{object23}, \text{bottom19}), \text{is}(\text{bottom19}, \text{bottom}),$
 $\text{is}(\text{bottom19}, \text{flat}), \dots$
- ความรู้ในโดเมน:
 $\text{liftable}(X), \text{stable}(X), \text{open-vessel}(X) \rightarrow \text{cup}(X)$
 $\text{is}(X, \text{light}), \text{has-part}(X, Y), \text{isa}(Y, \text{handle}) \rightarrow \text{liftable}(X)$
 $\text{small}(X), \text{made-from}(X, Y), \text{low-density}(Y) \rightarrow \text{liftable}(X)$
 $\text{has-part}(X, Y), \text{isa}(Y, \text{bottom}), \text{is}(Y, \text{flat}) \rightarrow \text{stable}(X)$
 $\text{has-part}(X, Y), \text{isa}(Y, \text{concavity}), \text{is}(Y, \text{upward-pointing}) \rightarrow \text{open-vessel}(X)$
- มโนทัศน์เป้าหมาย: $\text{cup}(X)$
 X เป็นถ้วยก็ต่อเมื่อ X ยกได้ (liftable) เสถียร (stable) และเป็นภาชนะเปิด (open-vessel)
- เกณฑ์ดำเนินการ: สิ่ง que แสดงลักษณะต่างๆ ของถ้วย เช่น $\text{isa}, \text{has-part}, \text{color}, \text{owner}$ เป็นต้น

ขั้นตอนการเรียนรู้ประกอบด้วย 2 ขั้นตอนดังนี้

ต้นไม้พิสูจน์

- (1) ใช้ความรู้ในโดเมนอธิบายว่าทำไม object23 จึงเป็น cup โดยการสร้าง **ต้นไม้พิสูจน์ (proof tree)** ของ object23 ดังแสดงใน **รูปที่ 6-32**



รูปที่ 6-32 ต้นไม้พิสูจน์ของตัวอย่างบวก cup

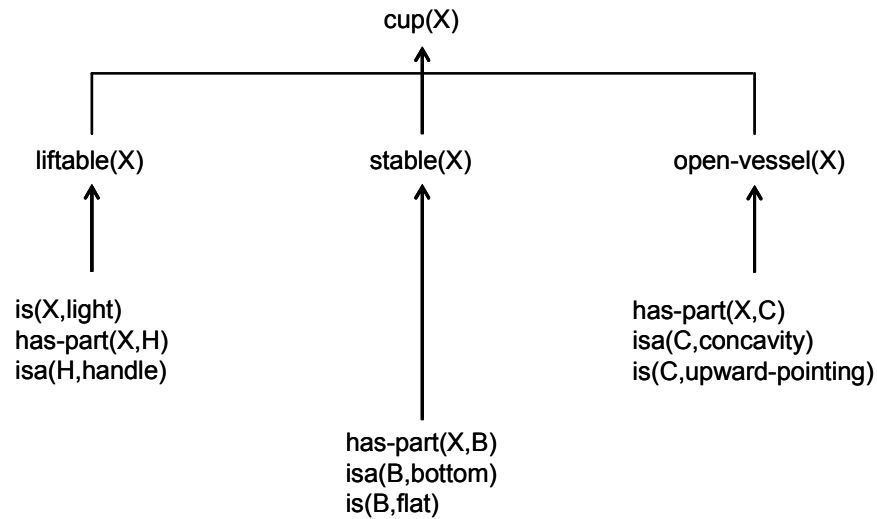
ต้นไม้พิสูจน์แสดงว่า object23 เป็น cup โดยมีคุณสมบัติ 3 อย่างคือยกได้ เสถียร และเป็นภาชนะเปิด เมื่อเราสังเกตความรู้ในโดเมนเรื่องถ้วยจะพบว่ากรวยได้ของถ้วยมีกฎ 2 ข้อที่ใช้อธิบายได้และ object23 ตรงกับกฎข้อแรกของการยกได้ กล่าวคือเป็นถ้วยที่มีหูหิ้วและเบา นอกจากนั้นในต้นไม้พิสูจน์นี้จะไม่พบเพรดิเคตที่ไม่เกี่ยวข้องกับการเป็นถ้วย อย่างเช่น owner, color เป็นต้น ซึ่งสิ่งนี้เป็นการทวงนัยทั่วไปแบบหนึ่งที่ตัดเงื่อนไขไม่จำเป็นทิ้งไป ดังนั้น ณ จุดนี้ถ้าเราสร้างกฎขึ้นเพื่ออธิบายการเป็นถ้วยของ object23 ก็จะได้กฎดังนี้

is(object23,light), has-part(object23,handle16), isa(handle16,handle),
 has-part(object23,bottom19), isa(bottom19,bottom), is(bottom19,flat),
 has-part(object23,concavity12), isa(concavity12,concavity),
 is(concavity12,upward-pointing) → cup(object23)

อย่างไรก็ดีแม้ว่ากฎนี้จะไม่พบเพรดิเคตที่ไม่เกี่ยวข้อง แต่กฎนี้ยังคงอธิบายได้เฉพาะ object23 เท่านั้น เราจำเป็นต้องทำการทวงนัยทั่วไปเพิ่มเติมขึ้นเพื่อให้ใช้กับถ้วยที่มีคุณสมบัติเหมือนกับ object23 ได้

- (2) การทวงนัยทั่วไปและดึงเพรดิเคตที่อยู่ในเกณฑ์ดำเนินการมาสร้างกฎ ขั้นตอนนี้ทำการทวงนัยทั่วไปโดยทำตามความรู้ในโดเมน กล่าวคือถ้าอาร์กิวเมนต์ของเพรดิเคตในต้นไม้พิสูจน์ที่ตรงกันกับอาร์กิวเมนต์ของเพรดิเคตของความรู้ในโดเมนเป็นตัวแปรก็เปลี่ยนอาร์กิวเมนต์ที่เป็นค่าคงที่ให้เป็นตัวแปร แต่ถ้าอาร์กิวเมนต์ของเพรดิเคตที่

ตรงกันกับความรู้ในโดเมนเป็นค่าคงที่ที่ไม่ต้องเปลี่ยน เช่น $is(object23, light)$ เปลี่ยนเป็น $is(X, light)$ เป็นต้น ผลที่ได้แสดงในรูปที่ 6-33



รูปที่ 6-33 การวางนัยทั่วไปของตัวอย่างบวก cup

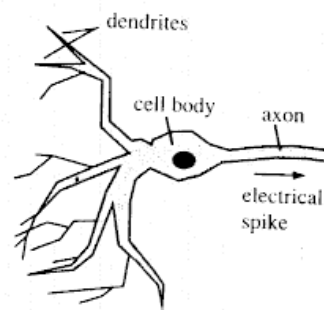
ดังนั้นเราจะได้กฎดังนี้

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

การเรียนรู้ฮิปโปไลต์เป็นการเรียนรู้ประเภทที่เรียกว่า **การเรียนรู้เชิงวิเคราะห์ (analytical learning)** กล่าวคือการเรียนรู้ประเภทนี้จะเป็นการจัดความรู้ (ความรู้ในโดเมน) ในรูปแบบใหม่ให้ใช้งานได้มีประสิทธิภาพ ดังจะเห็นได้ว่ากฎที่ได้โดยฮิปโปไลต์ประกอบด้วยเพรดิเคตที่อยู่ในเกณฑ์ดำเนินการเท่านั้น ซึ่งเพรดิเคตเหล่านี้จะใช้งานได้มีประสิทธิภาพสามารถจับคู่ (match) กับข้อมูลในตัวอย่างที่สอนแล้วทราบทันทีว่าตรงกันหรือไม่ ต่างกับความรู้ในโดเมนเดิมที่ประกอบด้วยเพรดิเคตบางตัว เช่น $liftable$ ที่ต้องการการอธิบายโดยพิสูจน์ต่อว่าเพรดิเคตนี้ตรงกับตัวอย่างหรือไม่

6.7 ข่ายงานประสาทเทียม

ข่ายงานประสาทเทียม (Artificial Neural Network) เป็นการจำลองการทำงานบางส่วนของสมองมนุษย์ เซลล์ประสาท (neuron) ในสมองของคนเราประกอบด้วยนิวเคลียส (nucleus) ตัวเซลล์ (cell body) โยประสาทนำเข้า (dendrite) แกนประสาทนำออก (axon) แสดงในรูปที่ 6-34

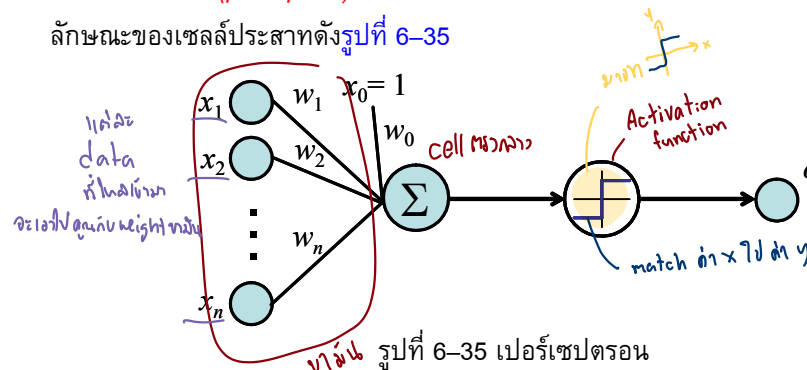


รูปที่ 6-34 เซลล์ประสาท

เดนไดรต์ทำหน้าที่รับสัญญาณไฟฟ้าเคมีซึ่งส่งมาจากเซลล์ประสาทใกล้เคียง เซลล์ประสาทตัวหนึ่งๆ จะเชื่อมต่อกับเซลล์ตัวอื่นๆ ประมาณ 10,000 ตัว เมื่อสัญญาณไฟฟ้าเคมีที่รับเข้ามาเกินค่าค่าหนึ่ง เซลล์จะถูกกระตุ้นและส่งสัญญาณไปทางแกนประสาทนำออกไปยังเซลล์อื่นๆ ต่อไป ประมาณกันว่าสมองของคนเรามีเซลล์ประสาทอยู่ทั้งสิ้นประมาณ 10^{11} ตัว

6.7.1 เพอร์เซปตรอน

เพอร์เซปตรอน (perceptron) เป็นข่ายงานประสาทเทียมแบบง่ายมีหน่วยเดียวที่จำลองลักษณะของเซลล์ประสาทดังรูปที่ 6-35



$$x_0 w_0 + x_1 w_1 + x_2 w_2 + \dots + x_n w_n =$$

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

ฟังก์ชันกระตุ้น



ในรูปแสดงฟังก์ชันกระตุ้น (activation function) ชนิดที่เรียกว่าฟังก์ชันสองขั้ว (bipolar function) ซึ่งแสดงผลของเอาต์พุตเป็น 1 กับ -1 ฟังก์ชันกระตุ้นอื่นๆ ที่นิยมใช้ก็อย่างเช่น ฟังก์ชันไบนารี (binary function) ซึ่งแสดงผลของเอาต์พุตเป็น 1 กับ 0 และเขียน



แทนด้วยรูป

เราสามารถแสดงเอาต์พุต (o) ในรูปของฟังก์ชันของอินพุต (x_1, x_2, \dots, x_n) ได้ดังนี้

$$o(x_1, x_2, \dots, x_n) = \begin{cases} 1 & \text{if } w_1x_1 + w_2x_2 + \dots + w_nx_n > \theta \\ -1 & \text{if } w_1x_1 + w_2x_2 + \dots + w_nx_n < \theta \end{cases} \quad (6.7)$$

เอาต์พุตเป็นฟังก์ชันของอินพุตในรูปของผลรวมเชิงเส้นแบบถ่วงน้ำหนัก น้ำหนักจะเป็นตัวกำหนดว่าในจำนวนอินพุตนั้น อินพุต (x_i) ตัวใดมีความสำคัญต่อการกำหนดค่าเอาต์พุต ตัวที่มีความสำคัญมากจะมีค่าสัมบูรณ์ของน้ำหนักมาก ส่วนตัวที่มีความสำคัญน้อยจะมีค่าใกล้เคียงศูนย์ ในกรณีที่ผลรวมเท่ากับค่าขีดแบ่งค่าเอาต์พุตไม่นิยาม (จะเป็น 1 หรือ -1 ก็ได้)

จากฟังก์ชันในสูตรที่ (6.7) เราจัดรูปใหม่โดยย้าย θ ไปรวมกับผลรวมเชิงเส้นแล้วแทน $-\theta$ ด้วย w_0 เราจะได้ฟังก์ชันของเอาต์พุตดังด้านล่างนี้

$$o(x_1, x_2, \dots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n > 0 \\ -1 & \text{if } w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n < 0 \end{cases} \quad (6.8)$$

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

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

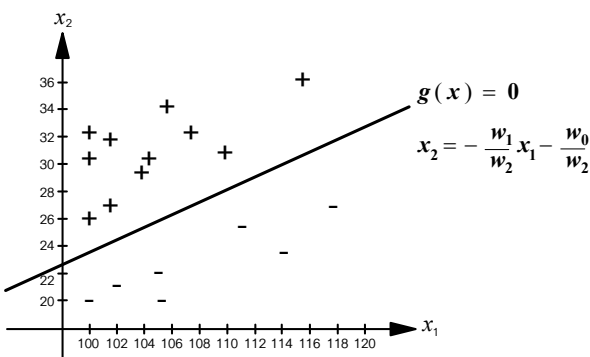
สมมติว่าเรามีอินพุตสองตัวคือ x_1 และ x_2 ซึ่งแสดงค่าส่วนสูงและน้ำหนักของเด็กนักเรียนประถมและหลังจากที่แพทย์ตรวจร่างกายของเด็กโดยละเอียดแล้วได้จำแนกนักเรียน

ออกเป็นสองกลุ่มคือเด็กอ้วนและเด็กไม่อ้วน เราให้เอาต์พุตเป็นค่าที่แสดงเด็กอ้วนแทนด้วย +1 กับไม่อ้วนแทนด้วย -1 ดังตารางที่ 6-16

ตารางที่ 6-16 ข้อมูลเด็กอ้วนและเด็กไม่อ้วน

เด็กคนที่	ส่วนสูง (ซม.)	น้ำหนัก (กก.)	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

ในกรณีที่อินพุต 2 ตัว (ไม่รวม x_0) เราจะได้ $g(\vec{x}) = w_0 + w_1x_1 + w_2x_2$ ซึ่งถ้าเราให้ $g(\vec{x}) = 0$ จะได้ว่า $w_0 + w_1x_1 + w_2x_2 = 0$ ซึ่งแทนสมการเส้นตรงในระนาบสองมิติ x_1 , x_2 สมการนี้มีจุดตัดแกนอยู่ที่ $-\frac{w_0}{w_2}$ และมีความชันเท่ากับ $-\frac{w_1}{w_2}$ เมื่อนำสมการนี้ไปวาดในระนาบสองมิติร่วมกับตัวอย่างสอนในตารางที่ 6-16 โดยกำหนดค่า w_0 , w_1 , w_2 ที่เหมาะสมจะได้ดังรูปที่ 6-36



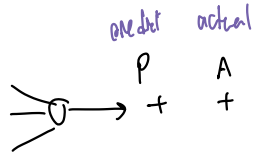
รูปที่ 6-36 สมการเส้นตรงสร้างโดยเพอร์เซปตรอน

เครื่องหมาย + และ - ในรูปแบบตัวอย่างบวก (เด็กอ้วน) และตัวอย่างลบ (เด็กไม่อ้วน) ตามลำดับ ดังจะเห็นได้ในรูปว่าเส้นตรงนี้เมื่อกำหนดจุดตัดแกนและความชันที่เหมาะสมซึ่งกำหนดโดย w_0 , w_1 , w_2 เส้นตรงนี้จะแบ่งตัวอย่างออกเป็นสองกลุ่มซึ่งอยู่คนละด้านของเส้นตรง และเมื่อมีข้อมูลส่วนสูงและน้ำหนักของเด็กคนอื่นที่เราต้องการทำนายว่าจะเป็นเด็กอ้วนหรือไม่ ก็ใช้เส้นตรงนี้โดยดูว่าข้อมูลใหม่นี้อยู่ด้านใดของเส้นตรง ถ้าด้านบนก็ทำนายว่าเป็นเด็กอ้วน (+) ถ้าด้านล่างก็ทำนายว่าเด็กไม่อ้วน (-)

ระนาบตัดสินใจ
หลายมิติ

ตัวอย่างด้านบนแสดงกรณีของอินพุตในสองมิติ จะเห็นได้ว่าเพอร์เซปตรอนจะเป็นเส้นตรง ในกรณีที่อินพุตมากกว่าสองมิติเพอร์เซปตรอนจะเป็น **ระนาบตัดสินใจหลายมิติ (hyperplane decision surface)** ปัญหาการเรียนรู้เพอร์เซปตรอนก็คือการหาค่าเวกเตอร์น้ำหนัก (\vec{w}) ที่เหมาะสมในการจำแนกประเภทของข้อมูลสอนเพื่อให้เพอร์เซปตรอนแสดงเอาต์พุตได้ตรงกับค่าที่สอน **กฎการเรียนรู้เพอร์เซปตรอน (perceptron learning rule)** ใช้สำหรับสอนเพอร์เซปตรอนโดยจะหาค่าเวกเตอร์น้ำหนักดังแสดงในตารางที่ 6-17

อัลกอริทึมเริ่มต้นจากสุ่มค่าเวกเตอร์น้ำหนัก ซึ่งโดยมากค่าที่สุ่มมานี้จะไม่ได้ระนาบหลายมิติที่แบ่งตัวอย่างได้ถูกต้องทุกตัวดังนั้นจึงต้องมีการแก้ไขน้ำหนักโดยเทียบเพอร์เซปตรอนกับตัวอย่างที่สอน หมายถึงว่าเมื่อเราป้อนตัวอย่างสอนเข้าไปในเพอร์เซปตรอนเราจะคำนวณค่าเอาต์พุตได้ นำค่าเอาต์พุตที่คำนวณได้โดยเพอร์เซปตรอนเทียบกับเอาต์พุตเป้าหมาย ถ้าตรงกันแสดงว่าจำแนกตัวอย่างได้ถูกต้อง ไม่ต้องปรับน้ำหนักสำหรับตัวอย่งนั้น แต่ถ้าไม่ตรงกันก็จะทำการปรับน้ำหนักตามสมการในอัลกอริทึม ส่วนอัตราการเรียนรู้เป็นตัวเลขบวกจำนวนน้อยๆ เช่น 0.01, 0.005 เป็นต้น อัตราการเรียนรู้นี้จะส่งผลต่อการลู่เข้าของเพอร์เซปตรอน ถ้าอัตราการเรียนรู้มีค่ามากเพอร์เซปตรอนก็จะเรียนรู้ได้เร็ว แต่ก็อาจเรียนรู้ไม่สำเร็จเนื่องจากการปรับค่ามีความหยวบเกินไป อัตราการเรียนรู้ที่มีค่าน้อยก็จะทำให้การปรับน้ำหนักทำได้อย่างละเอียดแต่ก็อาจเสียเวลาในการเรียนรู้นาน



ตารางที่ 6-17 อัลกอริทึมการเรียนรู้เพอร์เซปตรอน

stopping condition

Algorithm: Perceptron-Learning-Rule

1. Initialize weights w_i of the perceptron.
2. **UNTIL** the termination condition is met **DO**
 - 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.

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

$\Delta w_i \leftarrow \alpha(t-o)x_i$

learning rate α target t output o input feature x_i

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

การปรับน้ำหนักตามกฎการเรียนรู้เพอร์เซปตรอนโดยใช้อัตราการเรียนรู้ที่มีค่าน้อยเพียงพอ จะได้ระนาบหลายมิติที่จะเข้าสู่ระนาบหนึ่งที่สามารถแบ่งข้อมูลออกเป็นสองส่วน (ในกรณีที่ข้อมูลสามารถแบ่งได้) เพื่ออธิบายผลที่เกิดจากการปรับค่าน้ำหนัก เราจะลองพิจารณาพฤติกรรมของกฎการเรียนรู้นี้ดูว่าทำไมการปรับน้ำหนักเช่นนี้จึงเข้าสู่ระนาบที่แบ่งข้อมูลได้อย่างถูกต้อง

- พิจารณากรณีที่เพอร์เซปตรอนแยกตัวอย่างสอนตัวหนึ่งที่ได้รับเข้ามาได้ถูกต้อง กรณีนี้จะพบว่า $(t-o)$ จะมีค่าเป็น 0 ดังนั้น Δw_i ไม่เปลี่ยนแปลงเพราะ $\Delta w_i = \alpha(t-o)x_i$
- พิจารณาในกรณีที่เพอร์เซปตรอนให้เอาต์พุตเป็น -1 แต่เอาต์พุตเป้าหมายหรือค่าที่แท้จริงเท่ากับ 1 ในกรณีนี้หมายความว่าค่าที่เราต้องการคือ 1 แต่ค่าน้ำหนักไม่เหมาะสม ดังนั้นเพื่อที่จะทำให้เพอร์เซปตรอนให้เอาต์พุตเป็น 1 น้ำหนักต้องถูกปรับให้สามารถเพิ่มค่าของ $\vec{w} \cdot \vec{x}$ ในกรณีนี้หมายความว่าผลรวมเชิงเส้นน้อยเกินไปและน้อยกว่า 0 จึงได้เอาต์พุตเป็น -1 ดังนั้นสิ่งที่เราต้องการคือการเพิ่มค่าผลรวมเชิงเส้นเพราะถ้าเราเพิ่มค่าได้เรื่อยๆ จนมากกว่า 0 เพอร์เซปตรอนจะให้เอาต์พุตเป็น 1 ซึ่งตรงกับที่เราต้องการ พิจารณาดูดังต่อไปนี้ว่าการปรับค่าโดยกฎการเรียนรู้ทำให้ผลรวมเชิงเส้นเพิ่มขึ้นได้อย่างไร กรณีนี้เราจะได้ว่า $(t-o)$ 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 จะทำให้เพอร์เซปตรอนเอาต์พุตได้ถูกต้องยิ่งขึ้น
- ถ้า $x_i < 0$ เราจะได้ว่า $\alpha(t-o)x_i$ จะมีค่าน้อยกว่า 0 แสดงว่า w_i ตัวที่คูณกับ x_i ที่น้อยกว่า 0 จะลดลงทำให้ $\sum w_i x_i$ เพิ่มขึ้นเหมือนเดิม เพราะ x_i เป็นค่าลบและ w_i มีค่าลดลง ในที่สุดก็จะทำให้เพอร์เซปตรอนให้เอาต์พุตได้ถูกต้องยิ่งขึ้น
- ในกรณีที่เพอร์เซปตรอนให้เอาต์พุตเป็น 1 แต่เอาต์พุตเป้าหมายหรือค่าที่แท้จริงเท่ากับ -1 จะได้ว่า w_i ของ x_i ที่เป็นค่าบวกจะลดลง ส่วน w_i ของ x_i ที่เป็นค่าลบจะเพิ่มขึ้นและทำให้การปรับเป็นไปในทิศทางที่ถูกต้องเช่นเดียวกับในกรณีแรก

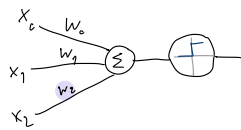
6.7.2 ตัวอย่างการเรียนรู้ฟังก์ชัน AND และ XOR ด้วยกฎการเรียนรู้เพอร์เซปตรอน

พิจารณาตัวอย่างการเรียนรู้ของเพอร์เซปตรอนโดยจะให้เรียนรู้ฟังก์ชัน 2 ฟังก์ชัน ฟังก์ชันแรกคือฟังก์ชัน AND แสดงในตารางที่ 6-18 ในกรณีนี้เราใช้ฟังก์ชันไบนารีเป็นฟังก์ชันกระตุ้น

ตารางที่ 6-18 ฟังก์ชัน AND(x_1, x_2)

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

ฟังก์ชัน AND ตามตารางด้านบนนี้จะให้ค่าที่เป็นจริงก็ต่อเมื่อ x_1 และ x_2 เป็นจริงทั้งคู่ (ดูที่สดมภ์เอาต์พุตเป้าหมาย) ผลการใช้กฎการเรียนรู้เพอร์เซปตรอนกับฟังก์ชัน AND แสดงในตารางที่ 6-19



ตารางที่ 6-19 ผลการเรียนรู้ฟังก์ชัน AND โดยกฎการเรียนรู้เพอร์เซปตรอน

Perceptron Learning Example - Function AND											
		Bias Input $x_0 = 1$						Alpha = 0.5			
Input	Input				Net Sum	Target	Actual	Alpha*	Weight Values		
x_1	x_2	$1.0 * w_0$	$x_1 * w_1$	$x_2 * w_2$	Input	Output	Output	Error	w_0	w_1	w_2
								$0.5(0-1)$	0.1	0.1	0.1
0	0	0.10	0.00	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	0.00	-0.40	0.10	0.10
1	0	-0.40	0.10	0.00	-0.30	0	0	0.00	-0.40	0.10	0.10
1	1	-0.40	0.10	0.10	-0.20	1	0	0.50	-0.40	0.60	0.60
0	0	0.10	0.00	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	0.10
1	0	-0.90	0.60	0.00	-0.30	0	0	0.00	-0.90	0.60	0.10
1	1	-0.90	0.60	0.10	-0.20	1	0	0.50	-0.40	1.10	0.60
0	0	-0.40	0.00	0.00	-0.40	0	0	0.00	-0.40	1.10	0.60
0	1	-0.40	0.00	0.60	0.20	0	1	-0.50	-0.90	1.10	0.10
1	0	-0.90	1.10	0.00	0.20	0	1	-0.50	-1.40	0.60	0.10
1	1	-1.40	0.60	0.10	-0.70	1	0	0.50	-0.90	1.10	0.60
0	0	-0.90	0.00	0.00	-0.90	0	0	0.00	-0.90	1.10	0.60
0	1	-0.90	0.00	0.60	-0.30	0	0	0.00	-0.90	1.10	0.60
1	0	-0.90	1.10	0.00	0.20	0	1	-0.50	-1.40	0.60	0.60
1	1	-1.40	0.60	0.60	-0.20	1	0	0.50	-0.90	1.10	1.10
0	0	-0.90	0.00	0.00	-0.90	0	0	0.00	-0.90	1.10	1.10
0	1	-0.90	0.00	1.10	0.20	0	1	-0.50	-1.40	1.10	0.60
1	0	-1.40	1.10	0.00	-0.30	0	0	0.00	-1.40	1.10	0.60
1	1	-1.40	1.10	0.60	0.30	1	1	0.00	-1.40	1.10	0.60
0	0	-1.40	0.00	0.00	-1.40	0	0	0.00	-1.40	1.10	0.60
0	1	-1.40	0.00	0.60	-0.80	0	0	0.00	-1.40	1.10	0.60
1	0	-1.40	1.10	0.00	-0.30	0	0	0.00	-1.40	1.10	0.60
1	1	-1.40	1.10	0.60	0.30	1	1	0.00	-1.40	1.10	0.60

$$-0.9 + 1.1 + 0 = 0.2$$

$$x_1 = 1$$

$$x_2 = 0$$

ขั้นตอนแรกเริ่มจากการสุ่มค่า w_0 จนถึง w_2 ในที่นี้กำหนดให้เป็น 0.1 ทั้งสามตัว จากนั้นก็เริ่มป้อนตัวอย่างเข้าไป (ทีละแถว) ตัวอย่างแรกได้ผลรวมเชิงเส้น (Net Sum) เป็น 0.10 ซึ่งมากกว่า 0 ดังนั้นเพอร์เซปตรอนจะให้เอาต์พุตจริง (Actual Output) ออกมาเป็น 1 ซึ่งผิดเพราะเอาต์พุตเป้าหมาย (Target Output) จะต้องได้เป็น 0 ทำให้อัตราการเรียนรู้คูณค่าผิดพลาด (Alpha x Error) ได้ -0.50 หลังจากนั้นก็นำไปปรับน้ำหนักตาม $w_i \leftarrow w_i + \Delta w_i$ และ $\Delta w_i \leftarrow \alpha(t-o)x_i$ ดังนั้นจะได้เป็น $w_0 \leftarrow w_0 + \alpha(t-o)x_0 = w_0 + 0.50(-1) \times 1 = 0.10 + (-0.5) = -0.4$ ต่อก็ปรับค่า 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

ให้ผลคูณเป็น 0 จึงไม่ได้ปรับ และเป็นข้อเสียของฟังก์ชันกระตุ้นแบบไบนารีซึ่งถ้าผลออกมาเป็น 0 จะไม่มีการปรับค่าให้ (ถ้าเราเปลี่ยน 0 เป็น -1 การปรับค่าจะดีขึ้น w_i จะถูกปรับทันทีตั้งแต่รอบแรก)

ตัวอย่างที่สองจนถึงตัวอย่างที่สี่ก็ทำเช่นเดียวกัน และเมื่อทำครบ 1 รอบการสอน (epoch) แล้วจะต้องทำการสอนซ้ำด้วยข้อมูลชุดเดิม นี่คือการสอนของข่ายงานประสาทเทียมซึ่งต่างจากวิธีอื่นๆ ที่ต้องใช้ข้อมูลชุดเดิมสอนซ้ำไปจนกระทั่งค่าผิดพลาดลดลงจนถึงจุดที่เราต้องการ ในที่นี้คือ 0 เนื่องจากเราต้องการให้มีการแบ่งข้อมูลอย่างเด็ดขาด และสมการเส้นตรงที่ได้จะมีค่า $w_0 = -1.40$, $w_1 = 1.10$ และ $w_2 = 0.60$

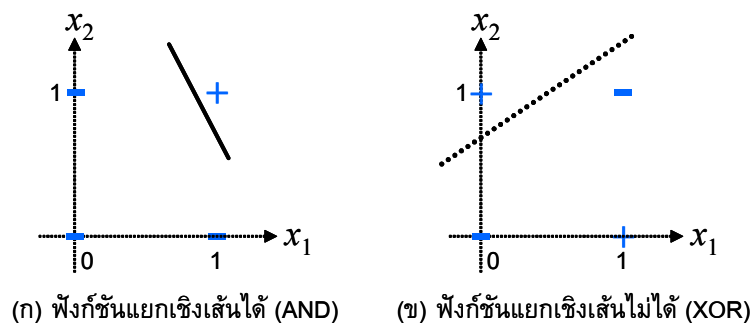
ฟังก์ชันที่สองที่จะทดลองเรียนรู้ด้วยกฎการเรียนรู้เพอร์เซปตรอนคือฟังก์ชัน XOR แสดงในตารางที่ 6-20

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 (ดูที่สัณฐานเอาต์พุตเป้าหมาย) ผลการใช้กฎการเรียนรู้เพอร์เซปตรอนกับฟังก์ชัน XOR แสดงในตารางที่ 6-21

ในกรณีของฟังก์ชัน XOR นี้พบว่าค่าผิดพลาดไม่ลดลง และค่าน้ำหนักจะแกว่งไปมาโดยไม่ลู่เข้าแม้ว่าจะสอนต่อจากนี้ไปอีกกี่รอบการสอนก็ตาม จึงสรุปว่าฟังก์ชัน XOR เรียนไม่สำเร็จด้วยเพอร์เซปตรอน เมื่อเรานำฟังก์ชัน AND และ XOR ไปวาดกราฟในสองมิติจะได้กราฟดังรูปที่ 6-37



รูปที่ 6-37 ฟังก์ชันแยกเชิงเส้นได้และไม่ได้

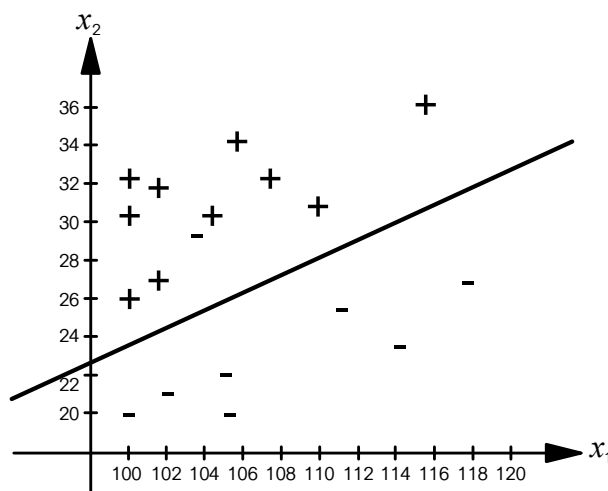
ตารางที่ 6-21 ผลการเรียนรู้ฟังก์ชัน XOR โดยกฎการเรียนรู้เพอร์เซปตรอน

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	Error	w0	w1	w2	
									0.1	0.1	0.1	
0	0	0.10	0.00	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	0.60	
1	0	0.10	0.10	0.00	0.20	1	1	0.00	0.10	0.10	0.60	
1	1	0.10	0.10	0.60	0.80	0	1	-0.50	-0.40	-0.40	0.10	
0	0	-0.40	0.00	0.00	-0.40	0	0	0.00	-0.40	-0.40	0.10	
0	1	-0.40	0.00	0.10	-0.30	1	0	0.50	0.10	-0.40	0.60	
1	0	0.10	-0.40	0.00	-0.30	1	0	0.50	0.60	0.10	0.60	
1	1	0.60	0.10	0.60	1.30	0	1	-0.50	0.10	-0.40	0.10	
0	0	0.10	0.00	0.00	0.10	0	1	-0.50	-0.40	-0.40	0.10	
0	1	-0.40	0.00	0.10	-0.30	1	0	0.50	0.10	-0.40	0.60	
1	0	0.10	-0.40	0.00	-0.30	1	0	0.50	0.60	0.10	0.60	
1	1	0.60	0.10	0.60	1.30	0	1	-0.50	0.10	-0.40	0.10	
0	0	0.10	0.00	0.00	0.10	0	1	-0.50	-0.40	-0.40	0.10	
0	1	-0.40	0.00	0.10	-0.30	1	0	0.50	0.10	-0.40	0.60	
1	0	0.10	-0.40	0.00	-0.30	1	0	0.50	0.60	0.10	0.60	
1	1	0.60	0.10	0.60	1.30	0	1	-0.50	0.10	-0.40	0.10	
0	0	0.10	0.00	0.00	0.10	0	1	-0.50	-0.40	-0.40	0.10	
0	1	-0.40	0.00	0.10	-0.30	1	0	0.50	0.10	-0.40	0.60	
1	0	0.10	-0.40	0.00	-0.30	1	0	0.50	0.60	0.10	0.60	
1	1	0.60	0.10	0.60	1.30	0	1	-0.50	0.10	-0.40	0.10	
0	0	0.10	0.00	0.00	0.10	0	1	-0.50	-0.40	-0.40	0.10	
0	1	-0.40	0.00	0.10	-0.30	1	0	0.50	0.10	-0.40	0.60	
1	0	0.10	-0.40	0.00	-0.30	1	0	0.50	0.60	0.10	0.60	
1	1	0.60	0.10	0.60	1.30	0	1	-0.50	0.10	-0.40	0.10	
0	0	0.10	0.00	0.00	0.10	0	1	-0.50	-0.40	-0.40	0.10	
0	1	-0.40	0.00	0.10	-0.30	1	0	0.50	0.10	-0.40	0.60	
1	0	0.10	-0.40	0.00	-0.30	1	0	0.50	0.60	0.10	0.60	
1	1	0.60	0.10	0.60	1.30	0	1	-0.50	0.10	-0.40	0.10	

ฟังก์ชันแยก
เชิงเส้นไม่ได้

จากรูปจะเห็นได้ว่าฟังก์ชัน AND เป็นฟังก์ชันที่แยก (ระหว่างตัวอย่างบวกกับตัวอย่างลบ) ได้ด้วยเส้นตรง ส่วนฟังก์ชัน XOR เราไม่สามารถหาเส้นตรงที่มาแบ่งตัวอย่างบวกและลบออกจากกัน (ไม่สามารถลากเส้นตรงให้ตัวอย่างบวกและลบให้อยู่คนละด้านของเส้น) ตัวอย่างการเรียนรู้ฟังก์ชัน XOR ข้างต้นได้แสดงให้เห็นว่า เพอร์เซปตรอนเรียนรู้บางฟังก์ชันไม่ได้ ฟังก์ชันเหล่านี้เรียกว่า **ฟังก์ชันแยกเชิงเส้นไม่ได้ (linearly non-separable function)** ส่วนฟังก์ชันที่แยกได้เรียกว่า **ฟังก์ชันแยกเชิงเส้นได้ (linearly separable function)** ซึ่งเป็นข้อจำกัดของเพอร์เซปตรอน เมื่อเราย้อนกลับไปดูตารางที่ 6-21 จะพบว่า นอกจากการเรียนรู้ฟังก์ชัน XOR ไม่สำเร็จแล้ว การเรียนรู้ก็จะไม่ลู่เข้าสู่เส้นตรงใดเส้นตรงหนึ่งอีกด้วย ดังจะเห็นได้จากการที่เวกเตอร์น้ำหนักจะแกว่งไปมา การไม่ลู่เข้าก่อให้เกิดปัญหาในการเรียนรู้เพราะเราจะไม่รู้ว่าจะหยุดอัลกอริทึม พิจารณาตัวอย่างใน

รูปที่ 6-38 ซึ่งมีตัวอย่างสอนเหมือนกับในรูปที่ 6-36 ยกเว้นว่าตัวอย่างตัวที่แปดในตารางที่ 6-16 มีการบันทึกค่าของประเภทผิดจาก 1 เป็น -1 ทำให้เกิดตัวอย่างลบบปะปนไปในกลุ่มของตัวอย่างบวกดังแสดงในรูป ในกรณีเช่นนี้เส้นตรงที่ดีที่สุดก็ยังคงเป็นเส้นตรงเดิมเหมือนกับในรูปที่ 6-36 แต่ว่ากฎการเรียนรู้เพอร์เซปตรอนจะไม่ให้คำตอบเป็นเส้นตรงนี้เนื่องจากอัลกอริทึมไม่ลู่เข้าสู่เส้นตรงเดียว แต่จะแกว่งไปมา

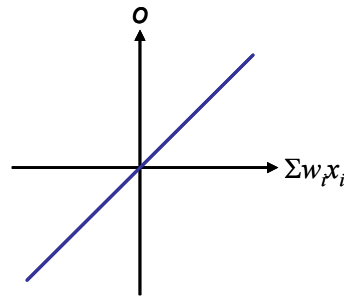


รูปที่ 6-38 เส้นตรงที่ให้ค่าผิดพลาดน้อยสุด

กฎเดลต้า (delta rule) เป็นกฎการเรียนรู้สำหรับหาค่าเวกเตอร์น้ำหนักของเพอร์เซปตรอนอีกกฎหนึ่งและมีข้อดีที่การเรียนรู้จะลู่เข้าสู่ระนาบหลายมิติที่ให้ค่าผิดพลาดน้อยสุด แม้ว่าตัวอย่างจะเป็นฟังก์ชันแบบแยกเชิงเส้นไม่ได้ กฎนี้ใช้หลักการของ**การเคลื่อนลงตามความชัน (gradient descent)** เพื่อหาคำตอบจากปริภูมิของเวกเตอร์น้ำหนักที่เป็นไปได้ ซึ่งกฎนี้เป็นพื้นฐานของอัลกอริทึมการแพร่กระจายย้อนกลับ (back-propagation) ดังจะกล่าวต่อไป

กฎเดลต้านี้จะหาเวกเตอร์น้ำหนักที่ให้ค่าผิดพลาดของตัวอย่างสอนน้อยสุดโดยใช้การหาอนุพันธ์ทางคณิตศาสตร์ ซึ่งในการหาค่าน้อยสุดด้วยอนุพันธ์นั้นจำเป็นต้องใช้ฟังก์ชันกระตุ้นที่หาอนุพันธ์ได้ ฟังก์ชันที่เราเคยใช้ก่อนหน้านี้เช่นฟังก์ชันสองขั้วและฟังก์ชันไปนารีเป็นฟังก์ชันที่หาอนุพันธ์ไม่ได้ในบางจุด ดังนั้นในกฎเดลต้านี้เราจะใช้ฟังก์ชันกระตุ้นแบบ**ฟังก์ชันเชิงเส้น (linear function)** ดังแสดงในรูปที่ 6-39 ซึ่งค่าเอาต์พุต (o) แสดงโดย
$$o(\vec{x}) = \vec{w} \cdot \vec{x} = \sum w_i x_i$$
 กล่าวคือเอาต์พุตจะเท่ากับผลรวมเชิงเส้น แม้ว่าตัวอย่างสอนจะแบ่งเป็นสองกลุ่ม (เช่น 1 กับ -1) ก็ไม่เกิดปัญหาเมื่อเราใช้ฟังก์ชันเชิงเส้นโดยจะทำนาย

ประเภทของตัวอย่างได้โดยดูที่เครื่องหมาย เช่นถ้าฟังก์ชันเชิงเส้นให้เอาต์พุตเป็น -0.21 ก็ให้ทำนายประเภทเป็น -1 เป็นต้น



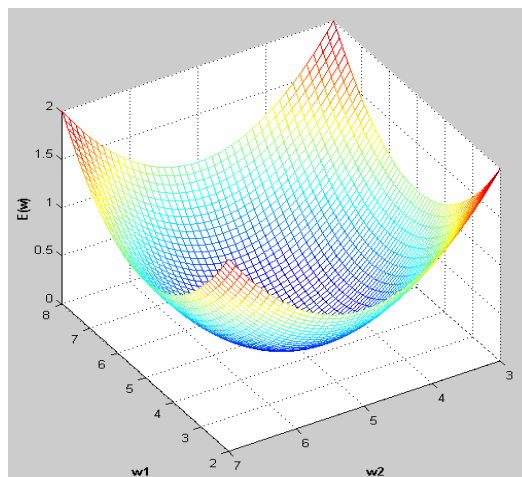
รูปที่ 6-39 ฟังก์ชันเชิงเส้น

ดังที่กล่าวข้างต้น กฎเดลต้าจะหาค่าเวกเตอร์น้ำหนักที่ให้ค่าผิดพลาดต่ำสุด ดังนั้นเรานิยามฟังก์ชันค่าผิดพลาดการสอน (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 เป็นเอาต์พุตเป้าหมายของตัวอย่าง d และ o_d เป็นเอาต์พุตของเพอร์เซปตรอนสำหรับตัวอย่าง d

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



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

คุณสมบัติของฟังก์ชันพาราโบลา คือจะมีค่าต่ำสุดเพียงค่าเดียว ในการหาค่าต่ำสุดเราสามารถทำได้โดยกำหนดจุด (w_1, w_2) เริ่มต้น สมมติว่าเป็น (w_{10}, w_{20}) จากนั้นหาเวกเตอร์สัมผัสพาราโบลา ณ ตำแหน่ง $E(w_{10}, w_{20})$ แล้วเราจะวิ่งลงตามความชันของเวกเตอร์ที่สัมผัสกับผิวค่าผิดพลาด (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 แต่เราต้องการวิ่งลง ดังนั้นเวกเตอร์ในแนวลงจึงเป็น $-\nabla E(\vec{w})$ เราจะได้ว่ากฎการปรับค่าเวกเตอร์น้ำหนักเป็น $\vec{w} \leftarrow \vec{w} + \Delta \vec{w}$ โดยที่ $\Delta \vec{w} = -\eta \nabla E(\vec{w})$ และ η คืออัตราการเรียนรู้เป็นค่าคงที่ตัวเลขบวก กฎเดลด้านนี้สามารถเขียนให้อยู่ในรูปของสมาชิกแต่ละตัวของเวกเตอร์น้ำหนักได้เป็น $w_i \leftarrow w_i + \Delta w_i$ โดยที่ $\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$

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

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

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

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

อัลกอริทึมของกฎเดลต้าเป็นดังตารางที่ 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**

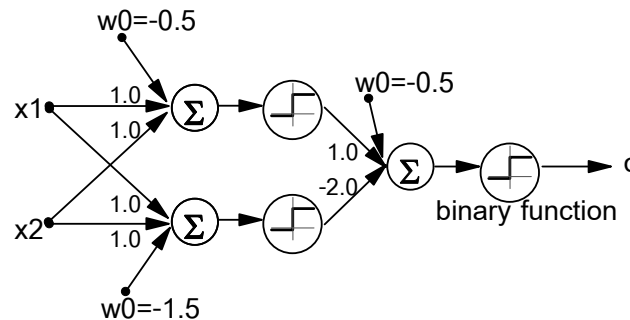
$$\Delta w_i \leftarrow \Delta w_i + \eta(t - o)x_i$$
 - 2.3 **FOR EACH** linear weight w_i **DO**

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

อัลกอริทึมกฎเดลต้าข้างต้นนี้จะหาค่าเวกเตอร์น้ำหนักที่ให้ค่าความผิดพลาดน้อยสุด ซึ่งมีข้อดีที่อัลกอริทึมจะลู่เข้า อย่างไรก็ตามก็ตีฟังก์ชันแยกเชิงเส้นไม่ได้ที่เรียนรู้ไม่ได้ด้วยกฎการเรียนรู้เพอร์เซปตรอนก็ไม่สามารถแยกได้อย่างถูกต้องสมบูรณ์ด้วยกฎเดลต้าเช่นกัน ในหัวข้อต่อไปจะกล่าวถึงข่ายงานหลายชั้นที่สามารถแยกฟังก์ชันประเภทนี้ได้

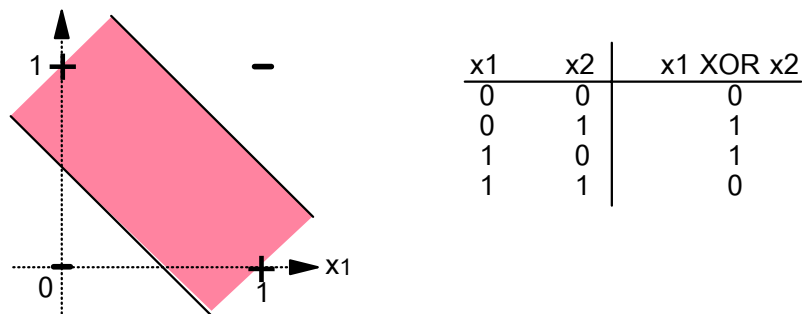
6.7.3 ข่ายงานหลายชั้นและการแพร่กระจายย้อนกลับ

จากข้างต้นจะเห็นว่าเพอร์เซปตรอนสามารถเรียนรู้ฟังก์ชันแยกได้เชิงเส้นเท่านั้น ในส่วนนี้จะอธิบายการนำเพอร์เซปตรอนหลายๆ ตัวมาเชื่อมต่อกัน เพื่อสร้างเป็นข่ายงานประสาทหลายชั้น (multilayer neural network) ที่สามารถแสดงผิวตัดสินใจไม่เชิงเส้น (non-linear decision surface) เพื่อให้เห็นถึงประสิทธิภาพของข่ายงานหลายชั้น จะยกตัวอย่างการต่อเพอร์เซปตรอน 3 ตัวเข้าด้วยกันเพื่อเรียนรู้ฟังก์ชัน XOR ดังแสดงในรูปที่ 6-41



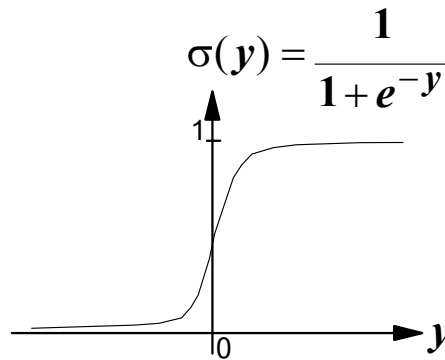
รูปที่ 6-41 ข่ายงานหลายชั้นสามารถเรียนรู้ฟังก์ชัน XOR

รูปที่ 6-41 แสดงการเชื่อมต่อเพอร์เซปตรอน 3 ตัวเข้าด้วยกัน เพอร์เซปตรอนสองตัวแรกได้รับอินพุตโดยตรงส่วนเพอร์เซปตรอนตัวที่สามรับอินพุตจากเอาต์พุตของเพอร์เซปตรอนสองตัวแรก จะเห็นได้ว่าเพอร์เซปตรอนตัวแรกที่อยู่ด้านซ้ายบนของรูปนั้นแทนฟังก์ชันเชิงเส้น $x_1 + x_2 = 0.5$ ส่วนเพอร์เซปตรอนตัวที่สองที่อยู่ด้านซ้ายล่างของรูปนั้นแทนฟังก์ชันเชิงเส้น $x_1 + x_2 = 1.5$ ฟังก์ชันเชิงเส้นทั้งสองมีความชันเท่ากันเท่ากับ -1 แต่มีจุดตัดแกนต่างกันดังแสดงในรูปที่ 6-42 ส่วนเพอร์เซปตรอนตัวที่สามทำหน้าที่รวมผลลัพธ์จากเพอร์เซปตรอนสองตัวแรก และโดยการกำหนดเวกเตอร์น้ำหนักที่เหมาะสมของเพอร์เซปตรอนตัวที่สามทำให้ได้ผิวตัดสินใจที่อยู่ระหว่างเส้นตรงทั้งสองเป็นตัวอย่างบวก และที่อยู่ด้านนอกเป็นตัวอย่างลบ



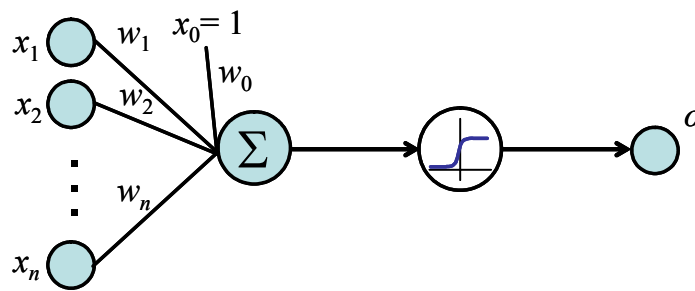
รูปที่ 6-42 ผิวตัดสินใจของข่ายงานในรูปที่ 6-41

ในการเชื่อมต่อครั้งนี้ใช้ฟังก์ชันกระตุ้นแบบไบนารีเพื่อให้ง่ายต่อการทำความเข้าใจ แต่การคำนวณหากฎเรียนรู้สำหรับข่ายงานหลายชั้นต้องใช้ฟังก์ชันกระตุ้นที่หาอนุพันธ์ได้ ดังนั้นเราจะไม่ใช้ฟังก์ชันไบนารีกับข่ายงานหลายชั้น แต่จะใช้ฟังก์ชันเชิงเส้นหรืออาจใช้ฟังก์ชันซิกมอยด์ (sigmoid function) ดังแสดงในรูปที่ 6-43



รูปที่ 6-43 ฟังก์ชันซิกมอยด์

เพอร์เซปตรอนที่ใช้ฟังก์ชันกระตุ้นเป็นฟังก์ชันซิกมอยด์แสดงในรูปที่ 6-44

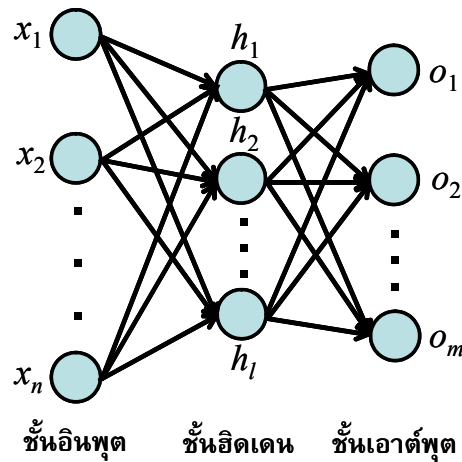


รูปที่ 6-44 เพอร์เซปตรอนที่ใช้ฟังก์ชันซิกมอยด์

คุณสมบัติหนึ่งของฟังก์ชันซิกมอยด์ก็คือสามารถแสดงอนุพันธ์ของฟังก์ชันในรูปของเอาต์พุตได้อย่างง่าย กล่าวคือ

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

อัลกอริทึมการแพร่กระจายย้อนกลับ (backpropagation algorithm) [Rumelhart & McClelland, 1986] เรียนรู้ค่าเวกเตอร์น้ำหนักสำหรับข่ายงานป้อนไปหน้าแบบหลายชั้น (multilayer feedforward network) โดยการใช้การเคลื่อนลงตามความชันเพื่อหาค่าต่ำสุดของค่าผิดพลาดระหว่างเอาต์พุตของข่ายงานกับเอาต์พุตเป้าหมาย ตัวอย่างของข่ายงานป้อนไปหน้าแบบหลายชั้นแสดงในรูปที่ 6-45



รูปที่ 6-45 ตัวอย่างข่ายงานป้อนไปหน้าแบบหลายชั้น

ตัวอย่างในรูปด้านบนแสดงข่ายงานป้อนไปหน้าแบบหลายชั้นซึ่งประกอบด้วยชั้นอินพุต ชั้นฮิดเดนหรือชั้นซ่อน และชั้นเอาต์พุต ในรูปแสดงชั้นฮิดเดนเพียงชั้นเดียวแต่อาจมีมากกว่าหนึ่งชั้นก็ได้ เส้นเชื่อมจะเชื่อมต่อเป็นชั้นๆ ไม่ข้ามชั้นจากชั้นอินพุตไปชั้นฮิดเดน ถ้ามีชั้นฮิดเดนมากกว่าหนึ่งชั้นก็เชื่อมต่อกันไป และสุดท้ายจากชั้นฮิดเดนไปชั้นเอาต์พุต ข่ายงานป้อนไปหน้าแบบหลายชั้นนี้จะไม่มีการเชื่อมต่อย้อนกลับจะมีแต่เส้นเชื่อมไปข้างหน้าอย่างเดียวเช่นไม่มีเส้นเชื่อมจากบัพในชั้นเอาต์พุตส่งกลับมายังบัพในชั้นฮิดเดนหรือชั้นอินพุต 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 ดังนั้นคำตอบของการแพร่กระจายย้อนกลับจึงเป็นค่าต่ำสุดเฉพาะที่ อัลกอริทึมแสดงในตารางที่ 6-23

ตารางที่ 6-23 อัลกอริทึมการแพร่กระจายย้อนกลับ

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

$$\delta_h = o_h(1 - o_h) \sum_{k \in \text{outputs}} w_{kh} \delta_k$$

- 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 ซึ่งเป็นค่าที่บันทึกได้จากการสังเกต จากนั้นนำโมเดลมาหาว่าสมมติฐานใดถูกต้องที่สุดโดยใช้ความน่าจะเป็นเข้ามาช่วย ข้อดีก็คือเราสามารถใช้อ้างอิงและ**ความรู้ก่อนหน้า (prior knowledge)** เข้ามาช่วยในการเรียนรู้ได้ด้วย ความรู้ก่อนหน้าหมายถึงความรู้ที่เราได้เกี่ยวกับสมมติฐานแต่ละตัวก่อนที่จะเก็บข้อมูล เมื่อใช้งานเราจะนำความน่าจะเป็นของข้อมูลที่เก็บได้มาปรับสมมติฐานซ้ำอีกครั้ง ซึ่งพบว่าวิธีนี้ให้ประสิทธิภาพในการเรียนรู้ได้ดีไม่ด้อยกว่าวิธีการเรียนรู้ประเภทอื่น

6.8.1 ทฤษฎีของเบส์

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

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

ความน่าจะเป็นก่อน
และ
ความน่าจะเป็นภายหลัง

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

ในกรณีของการเรียนรู้ของเครื่องนั้น สิ่งที่เราสนใจก็คือเมื่อเรามีชุดข้อมูลหรือเซตของตัวอย่างสอน D เราต้องการหาค่าความน่าจะเป็นที่สมมติฐาน (h) ที่เราสนใจว่ามีโอกาสจะเกิดขึ้นเท่าไร เราก็สามารถใช้ทฤษฎีของเบส์ในการคำนวณได้ดังนี้

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

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

ยกตัวอย่างเช่นถ้าเรามีต้นไม้ตัดสินใจหลายๆ ต้นและอยากทราบว่าแต่ละต้นมีโอกาสเกิดขึ้นหรือมีความถูกต้องเท่าไร ก็คือเราต้องการหา $P(h|D)$ นั่นเอง โดยที่ h แทนต้นไม้ตัดสินใจต้นหนึ่งที่เรากำลังพิจารณา เราอาจจะมีความเชื่อว่าต้นไม้ต้นเล็กมีโอกาที่จะเป็นจริงมากกว่าต้นไม้ใหญ่ (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 ที่ดีที่สุดนั้นโดยมากเรามักจะค่านี้ได้โดยไม่ต้องนำมาคำนวณดังจะกล่าวต่อไป ดังนั้นจะเห็นได้ว่าการใช้ทฤษฎีของเบย์สามารถใช้คำนวณความน่าจะเป็นของสมมติฐานแต่ละตัว เมื่อรู้ว่าเซตตัวอย่างสอนเป็นจริงซึ่งจะช่วยให้เราเลือกสมมติฐานที่ดีที่สุดได้

สมมติฐานภายหลังมากที่สุด

เราเรียกสมมติฐานที่ดีที่สุดว่า **สมมติฐานภายหลังมากที่สุด – เอ็มเอพี (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) ได้จากการใช้ทฤษฎีของเบย์และเนื่องจากว่าสำหรับ $h \in H$ ทุกตัวมี

ค่า $P(D)$ เท่ากันหมด ดังนั้นเราจึงสามารถละ $P(D)$ ได้และได้สมการที่ (6.18) กล่าวคือ h ที่ดีที่สุดตามเอ็มเอฟคือ h ที่ทำให้ค่า $P(D|h)P(h)$ มีค่าสูงสุด

เทคนิคการเรียนรู้ของเครื่องหลายวิธีไม่ได้หาค่า h_{MAP} แต่มักหา h_{ML} (Maximum Likelihood hypothesis) ดังในสมการที่ (6.19) ด้านล่างนี้ ซึ่งหมายถึงสมมติฐานที่ตรงหรือสอดคล้องกับข้อมูลสอนมากที่สุดจะเป็นสมมติฐานที่ดีที่สุดโดยไม่ได้พิจารณาความน่าจะเป็นก่อน

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

ยกตัวอย่างการใช้ทฤษฎีของเบส์เพื่อเลือกสมมติฐานที่น่าจะเป็นที่สุด สมมติว่าคนไข้คนหนึ่งไปตรวจหามะเร็งและผลการตรวจเป็นบวก อย่างไรก็ตามเราไม่แน่ใจว่าผลการตรวจเมื่อเป็นบวกจะให้ความถูกต้อง 98% ของกรณีที่มีโรคนั้นอยู่จริง และผลการตรวจเมื่อเป็นลบจะให้ความถูกต้อง 97% ของกรณีที่ไม่มีโรคนั้น นอกจากนั้นเรายังมีสถิติของการเป็นโรคมะเร็งว่า 0.008 ของประชากรทั้งหมดเป็นโรคมะเร็ง คำถามคือว่าคนไข้คนนี้มีโอกาสเป็นมะเร็งหรือไม่เป็นมะเร็งมากกว่ากัน?

เราใช้ทฤษฎีของเบส์สำหรับปัญหานี้ โดยกำหนดให้ $H = \{\text{cancer}, \sim\text{cancer}\}$ กล่าวคือมีสมมติฐานที่เป็นไปได้สองข้อคือคนไข้คนนี้เป็นมะเร็งกับไม่เป็นมะเร็ง เซตตัวอย่างสอนหรือข้อมูลของเราคือผลการตรวจเป็นบวก 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 $h_{MAP} = \sim\text{cancer}$ กล่าวคือมีโอกาสน้อยกว่าที่จะเป็นมะเร็งมากกว่า

6.8.2 สูตรพื้นฐานของความน่าจะเป็น

สูตรพื้นฐานเกี่ยวกับความน่าจะเป็น ที่จะใช้บ่อยครั้งในการเรียนรู้แบบเบย์มีดังต่อไปนี้

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 $\sum_{i=1}^n P(A_i) = 1$ แล้ว 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 เหตุการณ์จะได้ว่าความน่าจะเป็นรวม $P(A_1, \dots, A_n)$ มีค่าเท่ากับ

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

6.8.3 การจำแนกประเภทที่น่าจะเป็นที่สุดสำหรับตัวอย่าง

ดังที่กล่าวข้างต้น ในกรณีที่กำหนดให้เราใช้สมมติฐานได้เพียงข้อเดียวในการจำแนกประเภทของตัวอย่าง จะได้ว่า h_{MAP} เป็นสมมติฐานที่ดีที่สุด แต่การจำแนกประเภทของตัวอย่างด้วย h_{MAP} ไม่ใช่การจำแนกประเภทที่น่าจะเป็นที่สุด (most probable classification) สำหรับตัวอย่างนั้น ในบางกรณีที่เราสามารถใช้สมมติฐานหลายข้อเราสามารถจำแนกประเภทของตัวอย่างได้ดีกว่าการใช้ 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

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

ในกรณีนี้เราควรจะทำนายประเภทของ x เป็นบวกหรือลบ? ซึ่งถ้าใช้ h_{MAP} ก็จะได้ว่า h_1 เป็นสมมติฐานที่ดีที่สุดเนื่องจาก h_1 มีค่าความน่าจะเป็นภายหลังมากที่สุด แต่เมื่อพิจารณาสมมติฐานอื่นในปริภูมิของสมมติฐาน เราพบว่า h_{MAP} ให้คำตอบเป็น + เพียงตัวเดียว แต่สมมติฐานอีกสองตัวให้คำตอบเป็น - เราจะได้ว่าการทำนายประเภทที่น่าจะเป็นที่สุดในแบบของเบย์มีสูตรการคำนวณดังนี้

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

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 ตัวทำนายประเภทเบส์อย่างง่าย

ตัวทำนายประเภทเบส์อย่างง่าย (naive Bayes classifier) เป็นตัวทำนายประเภทแบบหนึ่งที่ใช้กันได้ดี เหมาะกับกรณีของเซตตัวอย่างมีจำนวนมากและคุณสมบัติ (attribute) ของตัวอย่างไม่ขึ้นต่อกัน มีการนำตัวทำนายประเภทเบส์อย่างง่ายไปประยุกต์ใช้งานในด้านการจำแนกประเภทข้อความ (text classification) การวินิจฉัย (diagnosis) และพบว่าใช้งานได้ดีไม่ต่างจากการทำนายประเภทวิธีการอื่น เช่นการเรียนรู้ต้นไม้ตัดสินใจ ข่ายงานประสาท เป็นต้น

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

$$\begin{aligned}
 v_{MAP} &= \arg \max_{v_j \in V} P(v_j | a_1, a_2, \dots, a_n) \\
 &= \arg \max_{v_j \in V} \frac{P(a_1, a_2, \dots, a_n | v_j) P(v_j)}{P(a_1, a_2, \dots, a_n)}
 \end{aligned} \tag{6.22}$$

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

โดยที่ a_i ในสมการเป็นค่าของคุณสมบัติ A_i V เป็นเซตของประเภทหรือค่าที่เป็นไปได้ของ x สมการที่ (6.23) แสดงการหาประเภทที่ดีที่สุดของตัวอย่าง x แต่เราจะพบว่าสมการนี้ใช้งานไม่ได้อย่างมีประสิทธิภาพ เนื่องจากการคำนวณค่าของ $P(a_1, a_2, \dots, a_n | v_j)$ ทำได้ยากลำบากมากเพื่อให้ได้ค่าที่น่าเชื่อถือในเชิงสถิติ ที่เป็นเช่นนี้เพราะว่าถ้าให้คุณสมบัติ A_i แต่ละตัวของตัวอย่างมีค่าที่เป็นไปได้ 10 ค่า และคุณสมบัติทั้งหมดมี 10 ตัว เราจะได้ว่ามีลำดับ a_1, a_2, \dots, a_n ที่เป็นไปได้ทั้งสิ้นเท่ากับ 10^{10} รูปแบบ ซึ่งหมายความว่าเราต้องหาดำเนินการทั้งหมด 10^{10} ตัว จึงจะมีโอกาสพบรูปแบบหนึ่งของ a_1, a_2, \dots, a_n สักหนึ่งครั้งโดยประมาณ ดังนั้นถ้าต้องการให้ค่า $P(a_1, a_2, \dots, a_n | v_j)$ มีความน่าเชื่อถือเชิงสถิติ เราต้องการตัวอย่างมากกว่า 10^{10} ตัวอย่างเท่า ซึ่งการที่จะหาดำเนินการจำนวนมากขนาดนั้นแทบจะทำได้จริงในทางปฏิบัติ 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

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

โดยที่ \prod หมายถึงการนำค่า $P(a_i | v_j)$ ทั้งหมดมาคูณกัน สูตรนี้ถ้าใช้กฎลูกโซ่มาคำนวณค่าความน่าจะเป็นที่ด้านซ้ายของสูตรจะได้เท่ากับ $P(a_1 | v_j) \times P(a_2 | a_1, v_j) \times P(a_3 | a_1, a_2, v_j) \times \dots \times P(a_n | a_{n-1}, a_{n-2}, \dots, a_1, v_j)$ ดังนั้นค่าความน่าจะเป็นทางด้านซ้ายของสมการจะเท่ากับผลคูณค่าความน่าจะเป็นทางด้านขวาก็ต่อเมื่อคุณสมบัติ a_1, a_2, \dots, a_n ไม่ขึ้นต่อกัน เช่นสีผมไม่ขึ้นกับส่วนสูง ฯลฯ แต่ในความเป็นจริงแล้วคุณสมบัติส่วนใหญ่ มักจะมีความสัมพันธ์กัน เช่นส่วนสูงกับน้ำหนัก เพราะถ้าตัวสูงน้ำหนักก็จะมากตามไปด้วย แต่อย่างไรก็ตามการใช้สมมติฐานความไม่ขึ้นต่อกัน (conditional independence assumption) นี้จะช่วยให้เราคำนวณค่าความน่าจะเป็นในสูตรที่ (6.24) ได้ง่ายขึ้น เพราะค่าความน่าจะเป็นของ a_i เมื่อรู้ v_j หาได้ง่ายกว่า เช่นถ้าจะหาคนผมสีน้ำตาล ส่วนสูงมาก น้ำหนักมาก และไม่ใช้โลชันไปผึ่งแดดแล้วผิวจะไหม้หรือไม่ เมื่อเอาไปหาดูในฐานข้อมูลอาจจะมีโอกาส

สมมติฐาน
ความไม่ขึ้นต่อกัน

พบข้อมูลที่มีค่าครบทั้ง 4 คำนี้น้อยมาก ๆ หรือต้องใช้จำนวนตัวอย่างมากมายมหาศาลถึงจะพบข้อมูลที่มีค่าครบตรงที่ต้องการ แต่ถ้าเราแยกคุณสมบัติออกจากกันเช่นหาคนผมสีน้ำตาลที่เป็นตัวอย่างบวก หรือหาคนไม่ใช่โลชั่นที่เป็นตัวอย่างบวก ทำให้ใช้ตัวอย่างไม่มาก และได้คำตอบ ถึงแม้ว่าคำตอบที่ได้อาจจะไม่ถูกต้องสมบูรณ์แต่ก็พบว่าทำงานได้ดีในทางปฏิบัติ

ดังนั้นเราจะได้ว่าตัวจำแนกประเภทแบบเบสอย่างง่ายคือ

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

จากสมการด้านบนนี้เราจะได้อัลกอริทึมการเรียนรู้เบสอย่างง่ายดังตารางที่ 6-24

ตารางที่ 6-24 อัลกอริทึมการเรียนรู้เบสอย่างง่าย

Algorithm: Naïve-Bayes

- **Naive_Bayes_Learn(examples)**
 FOR EACH target value v DO
 $\bar{P}(v_j) \leftarrow \text{estimate } P(v_j)$
 FOR EACH attribute value a of each attribute DO
 $\bar{P}(a_i | v_j) \leftarrow \text{estimate } P(a_i | v_j)$
- **Classify_New_Example(x)**

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

ยกตัวอย่างการใช้อัลกอริทึมการเรียนรู้เบสอย่างง่าย โดยใช้ชุดตัวอย่างสอนในตารางที่ 6-25 ต่อไปนี้

ตารางที่ 6-25 ตัวอย่างสอนสำหรับการเรียนรู้เบสอย่างง่าย (เหมือนกับตารางที่ 6-13)

						class
						↓
attribute →	Name	Hair	Height	Weight	Lotion	Result
	Sarah	blonde	average	light	no	Sunburned
value {	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 $v_{NB} = +$

การเรียนรู้เพื่อจำแนกประเภทข้อความโดยเบสอย่างง่าย

ในการเรียนรู้เพื่อจำแนกประเภทข้อความโดยใช้เบสอย่างง่ายนี้ สมมติว่าเรามีข้อความที่เราสนใจกับไม่สนใจ เมื่อทำการเรียนรู้แล้วเราต้องการทำนายว่าเอกสารหนึ่งๆ จะเป็นเอกสารที่เราสนใจหรือไม่ สามารถนำประยุกต์ใช้งานเช่นการกรองข่าวสารเลือกเฉพาะข่าวที่สนใจ เป็นต้น

ก่อนอื่นเราให้เอกสารหนึ่งๆ คือตัวอย่างหนึ่งตัว และเราแทนเอกสารแต่ละฉบับด้วยเวกเตอร์ของคำโดยใช้คำที่ปรากฏในเอกสารเป็นคุณสมบัติของเอกสาร กล่าวคือคำที่หนึ่งในเอกสารเป็นคุณสมบัติตัวที่หนึ่ง คำที่สองในเอกสารเป็นคุณสมบัติตัวที่สอง ตามลำดับ ดังนั้นจะได้ว่า a_1 คือคำที่หนึ่ง a_2 คือคำที่สองตามลำดับ จากนั้นก็ทำการเรียนรู้โดยใช้ตัวอย่างสอนเพื่อประมาณค่าความน่าจะเป็นต่อไปนี้เป็นคือ

1. $P(+)$ 2. $P(-)$ 3. $P(\text{doc}|+)$ 4. $P(\text{doc}|-)$

จากสมมติฐานเรื่องความไม่ขึ้นต่อกันของคุณสมบัติของเบสอย่างง่ายทำให้เราได้ว่า

$$P(\text{doc} | v_j) = \prod_{i=1}^{\text{length}(\text{doc})} P(a_i = w_k | v_j) \quad (6.26)$$

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 แต่พบว่าสูตรนี้ก็ยังไม่สามารถคำนวณได้จากเหตุผลในทำนองเดียวกันกับสมมติฐานความไม่ขึ้นต่อกันข้างต้น

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 ทำให้การคำนวณง่ายขึ้นเพราะไม่ต้องสนใจว่าคำหนึ่งๆ จะไปปรากฏในตำแหน่งใด หรือคำแต่ละคำจะไม่ขึ้นกับตำแหน่ง อัลกอริทึมสำหรับการจำแนกประเภทข้อความโดยใช้การเรียนรู้แบบอย่างง่ายเป็นดังตารางที่ 6-26 ต่อไปนี้

ตารางที่ 6-26 อัลกอริทึมการเรียนรู้แบบอย่างง่ายสำหรับจำแนกประเภทข้อความ

Algorithm: Learn_naive_Bayes_text(*Examples*, *V*)

1. Collect all words and other tokens that occur in *Examples*.
 - *Vocabulary* ← 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**
 - $docs_j$ ← subset of *Examples* for which the target value is v_j
 - $P(v_j) = \frac{|docs_j|}{Examples}$
 - $Text_j$ ← a single document created by concatenating all members of $docs_j$
 - n ← total number of words in $Text_j$ (counting duplicate words multiple times)
 - **FOR EACH** word w_k in *Vocabulary* **DO**
 - n ← 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* ← 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 ข่ายงานความเชื่อเบส

ข่ายงานความเชื่อเบส (Bayesian belief network) หรือเรียกโดยย่อว่า **ข่ายงานเบส (Bayes net)** เป็นวิธีการเรียนรู้ที่ลดข้อจำกัดของการเรียนรู้แบบสัจอย่างง่ายในสมมติฐานของความไม่ขึ้นต่อกันระหว่างคุณสมบัติ ในวิธีการเรียนรู้แบบสัจอย่างง่ายในหัวข้อที่แล้วจะตั้งสมมติฐานว่าคุณสมบัติใดๆ ไม่ขึ้นต่อกัน แต่ในความเป็นจริงเราพบว่าคุณสมบัติบางตัวจะขึ้นต่อกันบ้าง และควรที่จะนำความขึ้นต่อกันนี้เข้ามาใส่ไว้ในโมเดลด้วย เราจึงใช้ข่ายงานความเชื่อเบสในการอธิบายความไม่ขึ้นต่อกันอย่างมีเงื่อนไข (condition independent) ระหว่างตัวแปร (ในบริบทของข่ายงานความเชื่อเบสนิยมใช้คำว่า ‘ตัวแปร’ (variable) แทนคำว่า ‘คุณสมบัติ’) และในโมเดลนี้เราสามารถใส่ (1) ความรู้ก่อน (prior knowledge) เกี่ยวกับความ (ไม่) ขึ้นต่อกันระหว่างตัวแปร ร่วมกับ (2) ตัวอย่างสอน เพื่อให้กระบวนการเรียนรู้มีประสิทธิภาพ โดยเราสามารถใส่ความรู้ก่อนในข่ายงานความเชื่อเบสให้อยู่ในรูปของโครงสร้างข่ายงาน และตารางความน่าจะเป็นมีเงื่อนไข ดังจะกล่าวต่อไป

ก่อนอื่นเรานิยามความไม่ขึ้นต่อกันอย่างมีเงื่อนไขดังนี้

ความไม่ขึ้นต่อกัน
อย่างมีเงื่อนไข

นิยามที่ 5.1 ความไม่ขึ้นต่อกันอย่างมีเงื่อนไข

X ไม่ขึ้นกับ Y อย่างมีเงื่อนไขเมื่อรู้ Z ถ้าความน่าจะเป็นของ X ไม่ขึ้นกับค่าของ Y เมื่อรู้ค่าของ 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

$$P(X | Y, Z) = P(X | Z) \quad \square$$

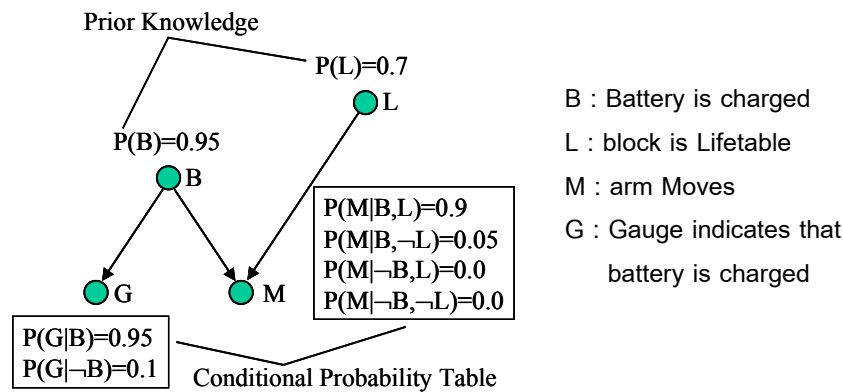
นิยามด้านบนนี้หมายความว่าสำหรับ x_i, y_j, z_k ใดๆ ความน่าจะเป็นที่ X จะมีค่าเป็น x_i (X เป็นตัวแปรส่วน x_i คือค่าของมัน) เมื่อรู้ว่า Y มีค่าเป็น y_j และ Z มีค่าเป็น z_k จะมีค่าเท่ากับ ความน่าจะเป็นของ X จะมีค่าเป็น x_i เมื่อรู้ว่า Z มีค่าเป็น z_k ในกรณีที่ความน่าจะเป็นทั้งสองเท่ากันเช่นนี้ เราเรียกว่าค่าของ X ไม่ขึ้นกับค่าของ Y อย่างมีเงื่อนไขเมื่อรู้ค่าของ Z เราจึงสามารถตัด Y ทิ้งไปได้

ตัวอย่างเช่นฟ้าร้องไม่ขึ้นกับฝนตกถ้ารู้ว่าฟ้าแลบ หรือเขียนได้เป็น

$$P(\text{Thunder} | \text{Rain}, \text{Lightning}) = P(\text{Thunder} | \text{Lightning})$$

ดังนั้นถ้ามีฟ้าแลบสามารถบอกได้เลยว่าจะต้องได้ยินเสียงฟ้าร้องด้วยความน่าจะเป็นเท่าไร โดยไม่ต้องสนใจว่าเกิดฝนตกหรือไม่

จากความสัมพันธ์กันอย่างมีเงื่อนไขข้างต้น เราสร้างข่ายงานของเบย์ได้ดังตัวอย่างในรูปที่ 6-46 ต่อไปนี้



รูปที่ 6-46 ตัวอย่างของข่ายงานเบย์

ตารางความน่าจะเป็น
มีเงื่อนไข - ซีพีที

จากรูปจะเห็นได้ว่าข่ายงานประกอบด้วยบัพหลายบัพ บัพแต่ละบัพหมายถึงคุณสมบัติของข้อมูลหรือตัวแปร และบัพแต่ละบัพจะมีตารางความน่าจะเป็นมีเงื่อนไข - ซีพีที (conditional probability table – CPT) ติดอยู่ด้วย ข่ายงานเบย์นี้แสดงในรูปของกราฟมีทิศทางซึ่งสามารถบอกได้ว่ามีตัวแปรใดบ้างที่ขึ้นกับตัวแปรอื่น และตัวแปรตัวใดบ้างที่ไม่ขึ้นกับตัวอื่น ตัวอย่างเช่นบัพ M ขึ้นกับบัพ B และบัพ L หรือถ้ามองเป็นลักษณะความสัมพันธ์ของบัพพ่อแม่กับบัพลูกจะเห็นว่าบัพพ่อแม่ของ M คือ B และ L ส่วนบัพพ่อแม่ของ G คือ B และสามารถบอกต่อไปได้ว่าบัพ G จะไม่ขึ้นกับบัพ L ถ้ารู้ B และได้ว่า G ไม่ขึ้นกับ M เมื่อรู้ B (สมมติว่าตัวแปรทั้งสี่คือ G, M, B และ L เป็นตัวแปรแบบบูล และเขียนแทนค่าของตัวแปรอย่างง่ายโดยใช้ตัวแปรนั้นแทนค่าจริงและใส่เครื่องหมาย \neg แทนค่าเท็จ เช่น G แทนค่าตัวแปร G เป็นจริง ส่วน $\neg G$ แทนค่าตัวแปร G เป็นเท็จ)

บัพใดๆ จะไม่ขึ้นกับบัพอื่นถ้ารู้บัพพ่อแม่โดยตรงของมัน จึงได้ว่า G จะไม่ขึ้นกับบัพอื่นถ้ารู้บัพ B ส่วน L ไม่มีบัพพ่อแม่ แสดงว่า L ไม่ขึ้นกับบัพอื่นๆ เช่นเดียวกับบัพ B ก็ไม่ขึ้นกับบัพอื่น ส่วน M ขึ้นกับ B และ L

จากข่ายงานเบย์ข้างต้น สมมติว่าเรากำลังจะเขียนข่ายงานที่อธิบายหุ่นยนต์ตัวหนึ่งที่ กำลังจะย้ายของในโดเมนโลกของบล็อก หุ่นยนต์ตัวนี้จะชาร์จแบตเตอรี่และมีเกจ (G) คอยวัดว่าขณะนี้แบตเตอรี่เหลืออยู่หรือไม่ หุ่นยนต์ทำงานด้วยการเคลื่อนแขนไปยกบล็อก เมื่อเราจำลองเหตุการณ์นี้ในข่ายงานเบย์จะได้ว่าแบตเตอรี่ (B) จะส่งผลต่อเกจ G นอกจากนั้นยังส่งผลต่อ M (การเคลื่อนแขนของหุ่นยนต์) และเราได้ใส่ความรู้ก่อนหน้าเข้าไปในรูปของ

ตารางความน่าจะเป็นมีเงื่อนไขว่า 70% ของบล็อกทั้งหมดสามารถยกได้ ($P(L)=0.7$) และในเวลา 100 ชั่วโมงมี 95 ชั่วโมงที่แบตเตอรี่มีไฟ ($P(B)=0.95$)

เมื่อดูที่ซีพีทีของ G พบว่า ถ้าแบตเตอรี่มีไฟ เกจซึ่งมีความบกพร่องอยู่บ้างนี้จะแสดงผลว่ามีไฟด้วยความน่าจะเป็นเท่ากับ 0.95 ($P(G|B)=0.95$) และถ้าไฟหมดแต่เกจยังแสดงว่ามีไฟด้วยความน่าจะเป็นเท่ากับ 0.1 ($P(G|\neg B)=0.1$)

ในตารางซีพีทีของบัพ M นั้น ตัวแรก $P(M|B,L)=0.9$ หมายความว่าหุ่นยนต์จะเคลื่อนแขนถ้าแบตเตอรี่มีไฟและบล็อกสามารถยกได้ และถ้ามีไฟแต่บล็อกไม่สามารถยกได้แขนจะเคลื่อนด้วยความน่าจะเป็น 0.05 ($P(M|B,\neg L)=0.05$) ถ้าไม่มีไฟและบล็อกสามารถยกได้หุ่นยนต์ก็จะไม่เคลื่อนแขน ($P(M|\neg B, L)=0.0$) และสุดท้ายถ้าบล็อกยกไม่ได้และไฟไม่มีแขนก็จะไม่เคลื่อนเช่นกัน ($P(M|\neg B, \neg L)=0.0$)

ทั้งหมดนี้คือความน่าจะเป็นทั้งหมดที่เราป้อนให้ระบบในรูปของซีพีที ผู้ที่ป้อนข้อมูลคือผู้เชี่ยวชาญที่ทำงานเกี่ยวกับหุ่นยนต์ เมื่อเราทราบค่าต่างๆ ทั้งหมดเราก็สามารถที่จะคำนวณความน่าจะเป็นต่างๆ ที่เกิดขึ้นภายในระบบนี้ได้เช่น ถ้าต้องการคำนวณหาว่าความน่าจะเป็นที่ แบตเตอรี่มีไฟ บล็อกสามารถยกได้ เกจขึ้นและหุ่นยนต์เคลื่อนแขน ทั้งสี่เหตุการณ์เกิดขึ้นพร้อมกันว่ามีค่าเท่าไรก็สามารถคำนวณได้จากข้างงานเบสนี้

ความน่าจะเป็นร่วม (Joint probability) ระหว่างตัวแปรคือความน่าจะเป็นที่ตัวแปรหลายตัวจะมีค่าตามที่กำหนด เช่น $P(\text{Battery, Lifiable, Gauge, Move})$ 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

$$\begin{aligned} P(G,M,B,L) &= P(G|B,M,L)P(M|B,L)P(B|L)P(L) \\ &= P(G|B)P(M|B,L)P(B|L)P(L) \\ &= (0.95)(0.9)(0.95)(0.7) \\ &= 0.57 \end{aligned}$$

สังเกตได้ว่าบรรทัดแรกใช้กฎลูกโซ่กระจาย $P(G,M,B,L)$ ออกมาเป็นด้านขวามือ และเมื่อกระจายแล้วจะเห็นว่าตัวแปรบางตัวไม่ขึ้นกับตัวอื่น เช่นเราสังเกตได้ว่า G จะขึ้นกับ B ตัวเดียวไม่ขึ้นกับ M หรือ L ดังนั้น $P(G|M,L)$ จึงลดรูปลงมาเหลือเป็น $P(G|B)$ เท่านั้น และ B ไม่ขึ้นกับ L ดังนั้น $P(B|L)$ จึงเหลือแค่ $P(B)$ พอลดรูปครบทุกตัวก็นำค่ามาใส่ไว้ในสมการแล้วหาผลลัพธ์ออกมา จะสังเกตได้ว่าเมื่อลดรูปลงมาแล้วบัพที่เราสงสัยจะขึ้นกับพ่อแม่ของมันเท่านั้นเช่น $P(G|M,L)$ ก็จะเหลือ $P(G|B)$ หรือหาความน่าจะเป็นของ G เมื่อรู้ B กรณีตัวอย่างที่ยกมาเป็นกรณีง่ายๆ เพราะเรารู้ค่าความน่าจะเป็นครบทั้งสี่ตัวแล้ว แต่ในบางกรณีเช่นเราทราบค่าของตัวแปรเพียงแค่ 2 ตัว หรือ 3 ตัว ไม่ใช่ทั้งหมดก็สามารถใช้เทคนิคในการอนุมานของข่ายงานเบย์เพื่อหาความน่าจะเป็นร่วมได้เช่นกัน ดังจะได้อธิบายด้านล่างนี้ ซึ่งเป็นเทคนิคการอนุมานที่ใช้ทั่วไปสำหรับข่ายงานเบย์ 3 เทคนิคเพื่อหาความน่าจะเป็นของตัวแปรที่เราสนใจ

1. **การอนุมานจากเหตุ (causal reasoning):** เมื่อเราทราบเหตุ เราสามารถหาได้ว่าผลจะเกิดขึ้นด้วยความน่าจะเป็นเท่าไร เช่น $P(M|L)$ หรือความน่าจะเป็นที่แขนจะเคลื่อนเมื่อรู้ว่าปลอกสามารถยกได้ (ปลอกยกได้เป็นสาเหตุหนึ่งของการที่หุ่นยนต์จะเคลื่อนแขน) แต่เราไม่ทราบค่า B (ไม่ทราบว่าขณะนี้แบตเตอรี่มีไฟหรือไม่) ถ้าเราย้อนกลับไปดูในข่ายงานเบย์ในรูปที่ 6-46 จะเห็นว่าเราไม่สามารถคำนวณ $P(M|L)$ ได้โดยตรง เพราะไม่มีค่าบอกไว้ในตาราง ในตารางมีค่าที่ใกล้เคียงที่สุดคือ $P(M|B,L)$ ดังนั้นเราต้องพยายามกระจาย $P(M|L)$ ให้อยู่ในรูปที่เกี่ยวข้อง ในที่นี้จะใช้ทฤษฎีความน่าจะเป็นทั้งหมดที่กล่าวว่า ถ้าเหตุการณ์ A_1, \dots, A_n ไม่เกิดร่วมกันและ $\sum P(A_i)=1$ แล้ว $P(B) = \sum P(B|A_i)P(A_i) = \sum P(B,A_i)$ เราสามารถนำ A_1, \dots, A_n กระจายเข้ามาได้ ดังนั้นเราสามารถกระจาย $P(M|L)$ ให้อยู่ในรูปของผลรวมของความน่าจะเป็นร่วมระหว่าง M กับ บัพพ่อแม่อื่นนอกจาก L (ซึ่งก็คือ B ที่เป็นบัพพ่อแม่ของ M ด้วย) ได้เป็น

$$P(M|L) = P(M, B|L) + P(M, \neg B|L)$$

เมื่อกระจายแล้วก็ยังพบว่าเรายังไม่ทราบค่าของ $P(M,B|L)$ อยู่ สิ่งที่เราารู้คือ $P(M|B,L)$ เราจึงต้องใช้กฎลูกโซ่กระจายแต่ละตัวได้เป็น

$$\begin{aligned} P(M|L) &= P(M|B, L)P(B|L) + P(M|\neg B, L)P(\neg B|L) \\ &= P(M|B, L)P(B) + P(M|\neg B, L)P(\neg B) \\ &= (0.9)(0.95) + (0.0)(0.05) \\ &= 0.855 \end{aligned}$$

2. **การอนุมานจากผล (diagnosis reasoning):** ข้อนี้จะตรงข้ามกับข้อแรก กล่าวคือเราทราบผลแล้วแต่อยากทราบว่าสาเหตุจะเกิดขึ้นด้วยความน่าจะเป็นเท่าไร เช่นต้องการคำนวณ $P(\neg L | \neg M)$ หรือความน่าจะเป็นที่บล็อกยกไม่ได้เมื่อรู้ว่าแขนไม่ได้เคลื่อนและหาไม่ได้โดยตรง ในกรณีนี้เราใช้ทฤษฎีของเบย์ดังที่กล่าวในตอนแรกว่า

$$P(A | B) = \frac{P(B | A)P(A)}{P(B)} \quad \text{ดังนั้น} \quad P(\neg L | \neg M) = \frac{P(\neg M | \neg L)P(\neg L)}{P(\neg M)}$$

ในส่วนของ $P(\neg M | \neg L)$ สามารถคำนวณได้โดยใช้การอนุมานจากเหตุในข้อที่แล้ว (ลองคำนวณดู) จะได้ $P(\neg M | \neg L) = 0.9525$ ส่วน $P(\neg L) = 0.3$ ดังนั้นจะได้ว่า

$$P(\neg L | \neg M) = \frac{0.9525 \times 0.3}{P(\neg M)} \quad \text{และพบยังมี } P(\neg M) \text{ ที่ยังไม่ทราบค่าอีก ซึ่งการหาค่า}$$

โดยตรงค่อนข้างยุ่ง เราจึงไปหาค่า $P(L | \neg M)$ เนื่องจาก $P(\neg L | \neg M) + P(L | \neg M) = 1$

$$\text{ดังนั้นเราจะได้ว่า } P(L | \neg M) = \frac{P(\neg M | L)P(L)}{P(\neg M)} = \frac{0.145 \times 0.7}{P(\neg M)} = \frac{0.1015}{P(\neg M)}$$

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):** เป็นการทำการอนุมานจากเหตุภายในการอนุมานจากผล เป็นการผสมระหว่างวิธีการทั้งสองแบบข้างต้น เช่นถ้าเราทราบ $\neg M$ (แขนไม่เคลื่อน) เราสามารถคำนวณ $\neg L$ หรือความน่าจะเป็นที่บล็อกไม่สามารถยกได้ แต่ถ้าเรารู้ $\neg B$ แล้ว $\neg L$ ควรจะมีค่าความน่าจะเป็นน้อยลง ในกรณีนี้เรียกว่า $\neg B$ 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

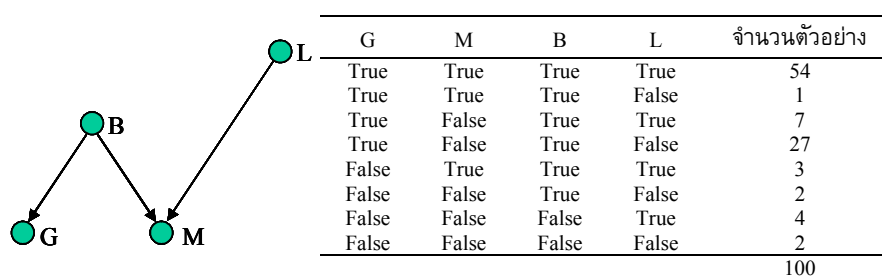
การเรียนรู้ข่ายงานเบสคือการหาโครงสร้างข่ายงานและ/หรือซีฟี่ที่สอดคล้องกับตัวอย่างสอนมากที่สุด ปัญหาการเรียนรู้ข่ายงานเบสแบ่งออกเป็นกรณีดังต่อไปนี้

1. โครงสร้างไม่รู้ (structure unknown)
2. โครงสร้างรู้ (structure known)
 - 2.1 ข้อมูลมีค่าครบ (no missing value data)
 - 2.2 ข้อมูลมีค่าหาย (missing value data)

กรณีที่ 1 เป็นกรณียากที่สุด เพราะเราไม่รู้โครงสร้างของข่ายงานเบสที่มีรูปร่างเป็นอย่างไร มีการเชื่อมต่อระหว่างบัพอย่างไร และแน่นอนว่าเราไม่รู้ค่าในซีฟี่ที่อีกด้วย ดังนั้นการเรียนรู้ต้องคำนวณหาทั้งโครงสร้างข่ายงานและซีฟี่ที่ ส่วนกรณีที่สองเป็นกรณีที่รู้โครงสร้างแล้ว ซึ่งบ่อยครั้งผู้เขียนข่ายงานเบสเป็นผู้เชี่ยวชาญในปัญหานั้นสามารถบอกโครงสร้างได้อย่างชัดเจน รู้ความสัมพันธ์ระหว่างตัวแปรในปัญหานั้นแต่อาจไม่รู้ค่าที่ถูกต้องและแม่นยำในตารางซีฟี่ที่ ดังนั้นกรณีนี้การเรียนรู้เป็นการหาค่าในซีฟี่ที่โดยอาศัยตัวอย่างสอน กรณีที่สองนี้ยังแบ่งเป็นกรณีย่อยอีกสองกรณีคือ กรณีที่ข้อมูลหรือตัวอย่างสอนทุกตัวมีค่าครบถ้วน กับอีกกรณีที่ตัวอย่างสอนบางตัวหรือทุกตัวมีค่าบางส่วนหายไป เช่น ไม่มีค่าของคุณสมบัติบางตัว เป็นต้น กรณีที่ 2.1 เป็นกรณีที่ง่ายที่สุดสามารถทำการเรียนรู้ได้ในลักษณะเดียวกับการเรียนรู้ของตัวจำแนกประเภทเบสอย่างง่าย โดยนับจำนวนครั้งที่เกิดขึ้นของข้อมูลเพื่อไปคำนวณซีฟี่ที่ของแต่ละบัพว่ามีค่าเท่าไรจึงจะแสดงต่อไปนี้ ส่วนกรณีที่ 1 ไม่ขออธิบายในที่นี้

การเรียนรู้ข่ายงานเบสในกรณีที่รู้โครงสร้างและข้อมูลครบ

ดูตัวอย่างต่อไปนี้ เรานับความถี่ของการเกิดค่าต่างๆ ของ G, M, B, L ว่าเกิดขึ้นกี่ครั้งได้ดังรูปที่ 6-47 โดยที่สมมติว่าโครงสร้างถูกกำหนดแล้วดังรูปที่ 6-47



รูปที่ 6-47 ตัวอย่างสอนสำหรับเรียนรู้ซีฟี่ที่ในกรณีข้อมูลครบ

$$\text{จาก } P(V_i=v_i|\text{Parents}(V_i)=\mathbf{P}_i) = \frac{\text{จำนวนตัวอย่างที่มี } V_i = v_i}{\text{จำนวนตัวอย่างที่มี Parents}(V_i)=\mathbf{P}_i}$$

ดังนั้นจะได้ค่าความน่าจะเป็นต่างๆ ดังนี้

$$P(B=\text{true}) = (54+1+7+27+3+2)/100 = 0.94$$

คือนับจำนวนตัวอย่างที่ B เป็นจริงในตารางหารด้วยจำนวนตัวอย่างทั้งหมด ค่าความน่าจะเป็นอื่นๆ ก็คำนวณในทำนองเดียวกัน

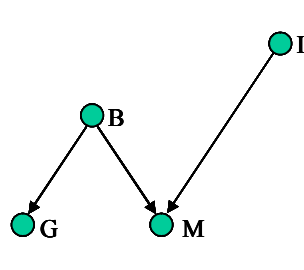
$$P(L=\text{true}) = (54+7+3+4)/100 = 0.68$$

$$P(M|B,L) \text{ เท่ากับอัตราส่วนที่ } M=\text{true} \text{ เมื่อ } B=\text{true}, L=\text{false} \text{ เท่ากับ } 1/(1+27+2) = 0.03$$

ด้วยวิธีนี้เราสามารถนำไปคำนวณหาความน่าจะเป็นของบัพ G ได้เช่นเดียวกัน

การเรียนรู้ข่ายงานเบสในกรณีที่โครงสร้างรู้และข้อมูลมีค่าหาย

กรณีต่อไปที่จะพิจารณาก็คือกรณีที่ข้อมูลบางตัวมีค่าบางค่าหายไปดังแสดงในรูปที่ 6-48 โดยที่สมมติว่ารู้โครงสร้างของข่ายงานเบสแล้ว



	G	M	B	L	จำนวนตัวอย่าง
True	True	True	True	True	54
True	True	True	True	False	1
*	*	True	True	True	7
True	False	True	True	False	27
False	True	*	True	True	3
False	False	True	False	False	2
False	False	False	True	True	4
False	False	False	False	False	2
					100

รูปที่ 6-48 ตัวอย่างสอนสำหรับเรียนรู้ซึฟฟี่ในกรณีข้อมูลมีค่าหาย

‘*’ ในตารางหมายถึงค่าหายไป พิจารณาแถวที่ห้าของข้อมูลในรูปซึ่งเป็นกรณีของตัวอย่าง 3 ตัวที่มีค่า $G=\text{false}$, $M=\text{true}$, $L=\text{true}$ ในกรณีนี้เราไม่รู้ค่าของ B แต่อาจคำนวณ $P(B|\neg G, M, L)$ หรือ $P(\neg B|\neg G, M, L)$ ได้ถ้าหากเรารู้ซึฟฟี่ (แต่เรายังไม่รู้) สมมติว่าเรารู้ซึฟฟี่ซึ่งจะทำให้เราหาความน่าจะเป็นที่ B จะเป็นจริง (หรือเท็จ) ของตัวอย่างทั้ง 3 ตัวได้ จากนั้นเราจะแทนที่ตัวอย่างทั้งสามนี้ด้วยตัวอย่างมีน้ำหนัก (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 ตัวในแถวที่สองที่มีค่า $B=\text{true}$, $L=\text{true}$ ส่วน G และ M ไม่รู้ค่านั้น 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 ตัวนี้ได้ถ้าเรารู้ค่าความน่าจะเป็นในซีฟิตี จากนั้นเราจะใช้ตัวอย่างมีน้ำหนักเหล่านี้ร่วมกับตัวอย่างที่เหลือในรูปที่ 6-48 เพื่อคำนวณซีฟิตีซึ่งเป็นสิ่งที่เราต้องการเรียน (ตัวอย่างที่ไม่รู้ค่าถูกแทนที่ด้วยตัวอย่างมีน้ำหนัก) แต่อย่างไรก็ดีเราจะทำเช่นนี้ได้โดยมีเงื่อนไขว่าเราต้องรู้ค่าในซีฟิตีที่ก่อน ซึ่งเรายังไม่รู้

วิธีการทำก็คือเราจะสมมติค่าความน่าจะเป็นในซีฟิตีโดยสุ่มค่าเริ่มต้นเข้าไปในซีฟิตี ซึ่งก็จะเสมือนว่าเรามีค่าในซีฟิตีแล้ว และเราจะสามารถหาค่าน้ำหนักของตัวอย่างไม่ทราบค่าได้ทุกตัว ก็จะทำให้เซตตัวอย่างเดิมที่มีตัวอย่างไม่รู้ค่าเป็นเซตตัวอย่างที่เรารู้ค่าทุกตัว การเรียนรู้ก็จะเหมือนกับกรณีที่ตัวอย่างมีข้อมูลครบแน่นอนว่าการคำนวณค่าน้ำหนักจะไม่ได้ค่าน้ำหนักที่ถูกต้องเพราะว่าเราสุ่มซีฟิตีที่เริ่มต้นที่ไม่ใช่ซีฟิตีที่ถูกต้อง แต่เนื่องจากว่าเมื่อเราได้ค่าน้ำหนักแล้วนำตัวอย่างไปรวมกับตัวอย่างที่เหลือที่เป็นตัวอย่างมีข้อมูลครบ ก็จะทำให้การประมาณค่าซีฟิตีที่ครั้งใหม่มีความถูกต้องเพิ่มขึ้นกว่าซีฟิตีที่เริ่มต้น เพราะว่าตัวอย่างส่วนใหญ่ของเราเป็นตัวอย่างที่ถูกต้อง จะมีตัวอย่างมีน้ำหนักเท่านั้นที่ไม่ถูกต้องสมบูรณ์ แสดงว่าการปรับค่าซีฟิตีทำให้ได้ซีฟิตีใหม่ที่ดีขึ้น และถ้าเราทำซ้ำกระบวนการเดิมด้วยซีฟิตีที่ดีขึ้นก็จะทำให้การหาค่าน้ำหนักมีความแม่นยำยิ่งขึ้น และส่งผลให้การปรับซีฟิตีในรอบต่อไปดีขึ้นอีกเมื่อวนซ้ำไปเรื่อยๆ ก็จะได้ซีฟิตีที่ดีขึ้นเรื่อยๆ จนกระทั่งซีฟิตีไม่เปลี่ยนแปลง เราก็หยุดกระบวนการเรียนรู้ได้ อัลกอริทึมการเรียนรู้แบบนี้เรียกว่า **อัลกอริทึมอีเอ็ม (EM – expectation maximization algorithm)** [Dempster, et al., 1977; McLachlan & Krishnan, 1996] ซึ่งแสดงในตารางที่ 6-27 ต่อไปนี้

ตารางที่ 6-27 อัลกอริทึมอีเอ็มสำหรับคำนวณค่าน้ำหนักของตัวอย่างไม่รู้ค่า

Algorithm: EM
1. Initialize all entries in all CPTs to some random values.
2. UNTIL the termination condition is met DO
2.1 Use the CPTs to calculate weights of the weighted examples.
2.2 Use the calculated weighted to estimate new CPTs.

อัลกอริทึมอีเอ็มนี้โดยทั่วไปจะใช้เวลาในการลู่เข้าไม่มาก ดังจะได้แสดงในตัวอย่างการเรียนรู้ซีฟี่ที่ของตัวอย่างสอนในรูปที่ 6-48 ดังนี้

(1) สุ่มค่าสำหรับตารางซีฟี่ที่

- $P(L) = 0.5$ $(P(\neg L) = 1 - P(L))$
- $P(B) = 0.5$ $(P(\neg B) = 1 - P(B))$
- $P(M|B, L) = 0.5$ $(P(\neg M|B, L) = 1 - P(M|B, L))$
 $P(M|B, \neg L) = 0.5$ $(P(\neg M|B, \neg L) = 1 - P(M|B, \neg L))$
- $P(M|\neg B, L) = 0.5$ $(P(\neg M|\neg B, L) = 1 - P(M|\neg B, L))$
 $P(M|\neg B, \neg L) = 0.5$ $(P(\neg M|\neg B, \neg L) = 1 - P(M|\neg B, \neg L))$
- $P(G|B) = 0.5$ $(P(\neg G|B) = 1 - P(G|B))$
 $P(G|\neg B) = 0.5$ $(P(\neg G|\neg B) = 1 - P(G|\neg B))$

(2) ใช้ซีฟี่ที่ที่สุ่มมาได้ใน การหาน้ำหนักของตัวอย่างไม่รู้ค่า
ตัวอย่างไม่รู้ค่าคือ

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 เราสามารถใส่น้ำหนักให้เป็นตัวอย่างมีน้ำหนักดังต่อไปนี้

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 เพื่อประมาณค่าพิกซ์ใหม่

- $P(L) = 68/100 = 0.680$ $(P(\neg L) = 1 - P(L))$
- $P(B) = 92.5/100 = 0.925$ $(P(\neg B) = 1 - P(B))$
- $P(M|B, L) = 59/62.5 = 0.944$ $(P(\neg M|B, L) = 1 - P(M|B, L))$
 $P(M|B, \neg L) = 1/30 = 0.033$ $(P(\neg M|B, \neg L) = 1 - P(M|B, \neg L))$
 $P(M|\neg B, L) = 1.5/5.5 = 0.273$ $(P(\neg M|\neg B, L) = 1 - P(M|\neg B, L))$
 $P(M|\neg B, \neg L) = 0/2 = 0.000$ $(P(\neg M|\neg B, \neg L) = 1 - P(M|\neg B, \neg L))$
- $P(G|B) = 85.5/92.5 = 0.924$ $(P(\neg G|B) = 1 - P(G|B))$
 $P(G|\neg B) = 0/7.5 = 0.000$ $(P(\neg G|\neg B) = 1 - P(G|\neg B))$

(2) ใช้ซีพีทีเพื่อคำนวณน้ำหนักของตัวอย่างไม่รู้ค่าใหม่

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	$7 \times 0.872 = 6.11$
True	False	True	True	$7 \times 0.052 = 0.36$
False	True	True	True	$7 \times 0.072 = 0.50$
False	False	True	True	$7 \times 0.004 = 0.03$

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	$3 \times 0.764 = 2.29$
False	True	False	True	$3 \times 0.236 = 0.71$

เมื่อทำซ้ำขั้นตอน (2), (3) จนครบ 20 รอบซีพีทีที่ใส่เข้าดังนี้ (ค่าความน่าจะเป็นทุกตัวใน
ทุกตารางซีพีทีที่มีค่าเปลี่ยนแปลงน้อยกว่า 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 มีคำอธิบายครอบคลุมเทคนิคของการเรียนรู้ของเครื่องไว้ค่อนข้างครบถ้วน แต่ถ้าต้องการศึกษาอย่างละเอียดเฉพาะเทคนิคหนึ่งๆ เช่นถ้าเกี่ยวกับอัลกอริทึมเชิงพันธุกรรมแนะนำให้ดูหนังสือของ Mitchell [Mitchell, 1996] และ Goldberg [Goldberg, 1989] ถ้าเป็นการเรียนรู้ต้นไม้ตัดสินใจให้ดูหนังสือของ Quinlan [Quinlan, 1993] ถ้าเกี่ยวกับข่ายงานประสาทเทียมก็แนะนำให้อ่านหนังสือ [Hassoun, 1995] ส่วนหนังสือของ Pearl [Pearl, 1988] เป็นหนังสือเกี่ยวกับข่ายงานเบย์ที่น่าศึกษาอย่างยิ่ง

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 Krishnan, 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*. Third Edition. Addison Wesley.

แบบฝึกหัด

1. กำหนดให้ความรู้ในโดเมนและตัวอย่างสอนเป็นดังต่อไปนี้

ความรู้ในโดเมน:

$a(X,X,Y), b(\text{red},Z) \rightarrow c(W,X,Y,Z)$

$d(Z,Z), d(Y,X), e(X,Y) \rightarrow a(X,Y,Z)$

$f(Y,X) \rightarrow b(X,Y)$

$g(X,X) \rightarrow d(X,Y)$

ตัวอย่างสอน:

$e(\text{eyes}, \text{eyes})$

$e(\text{eyes}, \text{ears})$

$e(\text{eyes}, \text{nose})$

$f(\text{fire}, \text{red})$

$f(\text{tree}, \text{green})$

$f(\text{snow}, \text{white})$

$g(2,2)$

$g(2,1)$

$g(3,2)$

$g(\text{eyes}, \text{eyes})$

$g(\text{eyes}, \text{ears})$

$g(\text{eyes}, \text{nose})$

- จงแสดงให้เห็นว่า $c(\text{white}, \text{eyes}, 2, \text{fire})$ เป็นตัวอย่างที่ถูกโดยใช้ต้นไม้พิสูจน์
- กำหนดให้เกณฑ์ดำเนินการประกอบด้วยเพรดิเคต 3 ตัวคือ e, f และ g จงเขียนกฎที่เรียนได้จากตัวอย่างด้านบน

2. ในการเรียนโมทัศน์ของ “EnjoySport” เราสังเกตว่าเพื่อนของเราคนหนึ่งจะสนุกกับการเล่นกีฬาทางน้ำหรือไม่ โดยได้พิจารณาถึงปัจจัย 6 อย่างคือ Sky (ท้องฟ้า), AirTemp (อุณหภูมิอากาศ), Humidity (ความชื้น), Wind (ลม), Water (น้ำ), Forecast (คำพยากรณ์) และได้บันทึกตัวอย่างบวก (3 ตัว) และตัวอย่างลบ (1 ตัว) ดังแสดงด้านล่าง

(Sunny, Warm, Normal, Strong, Warm, Same) +

(Sunny, Warm, High, Strong, Warm, Same) +

(Rainy, Cold, High, Strong, Warm, Change) -

(Sunny, Warm, High, Strong, Cool, Change) +

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

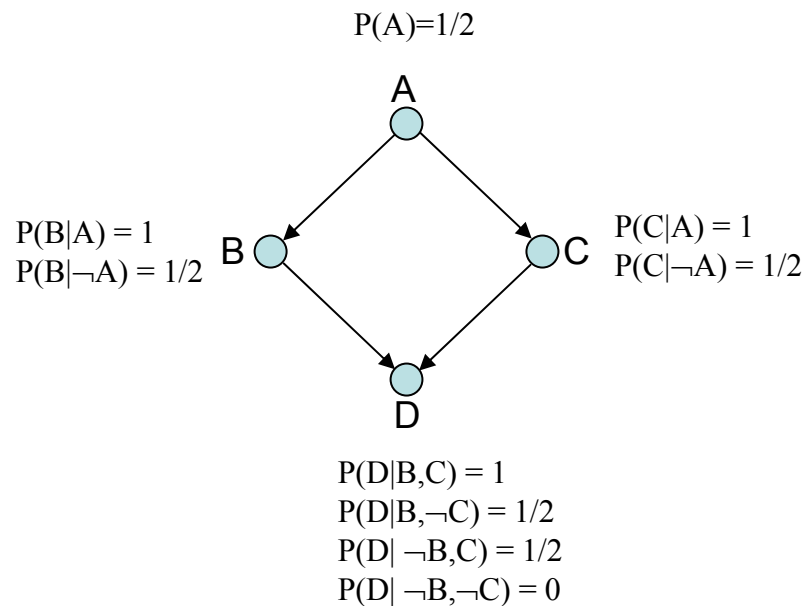
จึงตอบคำถามต่อไปนี้

- ปริภูมิโมนัทสน์ในกรณีนี้มีขนาดเท่าไร
- จงแสดงเซต S และ G เมื่อรับตัวอย่างเข้าไปที่ละตัวตามลำดับ
- เมื่อรับตัวอย่างทั้ง 4 ตัวเข้าไปหมดแล้ว เวอร์ชันสเปซจะประกอบด้วย สมมติฐานที่เป็นไปได้ทั้งหมดกี่ตัว อะไรบ้าง (สมมติฐานทั้งหมดที่อยู่ระหว่าง S และ G (รวม S และ G ด้วย))

3. ตารางด้านล่างนี้เป็นข้อมูลของผู้ที่มาขอทำบัตรเครดิตจากธนาคารแห่งหนึ่ง ข้อมูลของคนหนึ่งๆ ประกอบด้วย account, employed, cash ธนาคารใช้ข้อมูลเหล่านี้สำหรับกำหนดว่าจะทำบัตรให้หรือไม่ 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 แล้วมีคุณสมบัติดีจริงหรือไม่ จึงได้สร้างข่ายงานเบสดังรูปต่อไปนี้



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 เป็นตัวแปรแบบบูลคือมีค่าได้ 2 ค่าคือจริงกับเท็จ และการเขียนค่าตัวแปรในข่ายงานเป็นการเขียนแบบย่อ กล่าวคือ

X แทนตัวแปร X มีค่าความจริงเป็นจริง

$\neg X$ แทนตัวแปร X มีค่าความจริงเป็นเท็จ

เช่น $P(D|\neg B,C)$ คือความน่าจะเป็นที่ D มีค่าเป็นจริงเมื่อรู้ว่า B มีค่าเป็นเท็จและ C มีค่าเป็นจริง เป็นต้น

จงแสดงวิธีการคำนวณหาค่า $P(A|D)$ ว่ามีค่าเท่ากับเท่าไร

5. พิจารณาเพอร์เซปตรอนยูนิตเดียวที่มีอินพุต 3 บิตคือ x, y, z และมีเอาต์พุต 1 บิตคือ f ถ้าให้ตัวอย่าง 4 ตัวต่อไปนี้ จงแสดงให้เห็นว่าเพอร์เซปตรอนนี้จะเรียนรู้ได้สำเร็จหรือไม่

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
- กฎ, 63, 67
- กฎเจาะจงตัวแปรเอกภาพ, 49
- กฎการอนุมาน, 49
- กฎผลคูณ, 189
- กฎผลรวม, 189
- กฎลูกโซ่, 189
- กรอบ, 147
- การเข้าใจภาษาธรรมชาติ, 92
- การเคลื่อนลงตามความชัน, 178
- การเรียนรู้ของเครื่อง, 119
- การเรียนรู้เชิงวิเคราะห์, 173
- การเรียนรู้โดยการจำ, 136
- การเรียนรู้โดยการวิเคราะห์ความแตกต่าง, 141
- การเรียนรู้โดยการอธิบาย, 164
- การเรียนรู้ต้นไม้ตัดสินใจ, 153
- การเรียนรู้แบบเบย์, 186
- การแจกจ่ายเฉพาะ, 143
- การแทนความรู้, 45
- การแทนค่า, 50, 67
- การแปลความหมาย, 46
- การแพร่กระจายย้อนกลับ, 183
- การโปรแกรมแบบเรียกซ้ำ, 73
- การโปรแกรมอัตโนมัติ, 4
- การไขว้เปลี่ยน, 120
- การกลายพันธุ์, 120
- การค้นหา
- การค้นหาดาว, 31
- การค้นหาต้นไม้เกมน้อยสุดมากที่สุด, 137
- การค้นหาทั้งหมด, 10
- การค้นหาแนวกว้างก่อน, 10, 12
- การค้นหาแนวลึกก่อน, 13
- การค้นหาในปริภูมิสถานะ, 7
- การค้นหาลำดับ, 10
- การค้นหาแบบบอด, 10, 11
- การค้นหาแบบฮิวริสติก, 10, 16
- การจับคู่, 69
- การจำแนกประเภทที่น่าจะเป็นที่สุด, 189
- การทดสอบทัวริง, 2
- การทำเหมืองข้อมูล, 3
- การทำให้เท่ากัน, 51, 70
- การปฏิเสธแบบรีโซลูชัน, 58
- การประมวลผลภาษาธรรมชาติ, 3, 91
- การพิสูจน์, 49
- การพิสูจน์ทฤษฎี, 4
- การย้อนรอย, 69
- การวางนัยทั่วไปของมโนทัศน์, 145
- การวิเคราะห์ทางความหมาย, 92
- การวิเคราะห์ทางปฏิบัติ, 92
- การวิเคราะห์ทางวากยสัมพันธ์, 92, 96
- การวิเคราะห์ทางองค์ประกอบ, 92
- การวิวัฒนาการโดยผ่านการคัดเลือกตามธรรมชาติ, 120
- การหือของสัญญาณ, 54
- การอธิบายลดความเป็นไปได้, 199
- การอนุมานจากเหตุ, 198
- การอนุมานจากผล, 199
- ข้อเท็จจริง, 63, 65
- ข้อคำถาม, 63, 66
- ขั้นตอนวิธีเชิงพันธุกรรม, 119
- ข่ายงานเบย์, 195
- ข่ายงานเปลี่ยนสถานะเรียกซ้ำ, 111

- ข่ายงานความเชื่อเบสส์, 195
 ข่ายงานความหมาย, 142
 ข่ายงานประสาทเทียม, 169
 ข่ายงานหลายชั้น, 181
 คลังศัพท์, 96
 ความไม่ขึ้นต่อกันอย่างมีเงื่อนไข, 195
 ความขัดแย้ง, 58
 ความน่าจะเป็นก่อน, 186
 ความน่าจะเป็นภายหลัง, 186
 ความหลากหลาย, 129
 ค่าคงที่, 45, 67
 ค่าความเหมาะสม, 122
 ค่าดีสุดเฉพาะที่, 25
 ค่าดีสุดวงกว้าง, 25
 ค่าฮิวริสติก, 17
 คุณสมบัติ, 154
 งานประยุกต์ทางปัญญาประดิษฐ์, 3
 จำเพาะกว่า, 148
 ต้นไม้แฉ่งส่วน, 93
 ต้นไม้ตัดสินใจ, 153
 ต้นไม้พิสูจน์, 166
 ตรรกะเพรดิเคต, 45
 ตรรกะเพรดิเคตอันดับที่หนึ่ง, 48
 ตัวเชื่อม, 47
 ตัวแฉ่งส่วน, 96
 ตัวแฉ่งส่วนแบบบนลงล่าง, 97
 ตัวแฉ่งส่วนตาราง, 102
 ตัวแปร, 45, 63, 66
 ตัวแปรไม่สนใจ, 71
 ตัวแปลภาษาโปรแกรม, 66
 ตัวกระทำ, 8
 ตัวจำแนกประเภทเบสส์อย่างง่าย, 190
 ตัวตัด, 81
 ตัวตัดเขี้ยว, 83
 ตัวตัดแดง, 86
 ตัวทำให้เท่ากัน, 51
 ตัวทำให้เท่ากันกว้างสุด, 51
 ตัวบ่งปริมาณ, 47
 ตัวบ่งปริมาณเอกภาพ, 48
 ตัวบ่งปริมาณมีอยู่, 48
 ตัวอย่างการแทนค่า, 50
 ตัวอย่างบวก, 141
 ตัวอย่างลบ, 141
 ตามรอย, 72
 ตารางความน่าจะเป็นมีเงื่อนไข, 196
 ทฤษฎี, 49
 ทฤษฎีของเบสส์, 186
 ทฤษฎีความน่าจะเป็นทั้งหมด, 189
 ทฤษฎีสารสนเทศ, 160
 ทำให้เท่ากัน, 51
 นิเสธ, 80
 นิยามของปัญญาประดิษฐ์, 1
 บุคลากรทางวางนิพนธ์, 92
 ประเภท, 153
 ปริภูมิโนทัศน์, 148
 ปัญหาโลกของบล็อก, 11
 ปัญหาการเดินทางของพนักงานขาย, 16
 ปัญหาการจัดตาราง, 4
 ปัญหาต้นไม้ k กิ่งน้อยสุด, 34
 ปัญหาทางมโนทรรศน์, 4
 ปัญหาลิงกินกล้วย, 71
 โปรแกรมแทรกตัวเลข, 85
 โปรแกรมคูณจำนวนธรรมชาติ, 76
 โปรแกรมจำนวนธรรมชาติ, 74
 โปรแกรมต่อรายการ, 79
 โปรแกรมบวกเลข, 75
 โปรแกรมภาวะสมาชิก, 78
 โปรแกรมลบสมาชิก, 87
 เป้าหมาย, 67
 พจน์, 50, 67
 พลาตน้อย, 141
 เพรดิเคต, 45, 65

- เพอร์เซปตรอน, 169
 ฟังก์ชัน, 45
 ฟังก์ชันเกน, 160
 ฟังก์ชันแมนฮัตตัน, 20
 ฟังก์ชันแยกเชิงเส้นได้, 177
 ฟังก์ชันแยกเชิงเส้นไม่ได้, 177
 ฟังก์ชันกระตุ้น, 170
 ฟังก์ชันสคอเล็ม, 55
 ฟังก์ชันฮิวริสติก, 17
 ภาวะต้องห้าม, 32
 ภาษาโปรแกรม, 63
 มโนทัศน์เป้าหมาย, 149
 มิติโกนของฮิลบर्ट, 157
 มีนัยทั่วไปกว่า, 148
 โมดัลฟิสิกส์, 49
 ระบายตัดสินใจหลายมิติ, 172
 ระบบผู้เชี่ยวชาญ, 4, 163
 ระยะเวลาต้องห้าม, 35
 รายการ, 77
 รีโซลูชัน, 54
 ลำตัว, 68
 วิทยาการหุ่นยนต์, 4
 เวอร์ชันสเปซ, 147
 ไวยากรณ์, 96
 ไวยากรณ์ไม่พึงปรารถนา, 96
 ไวยากรณ์ย้ายงานเปลี่ยนสถานะ, 110
 สถานภาพต้องห้าม, 32
 สถานะเป้าหมาย, 8
 สถานะเริ่มต้น, 8
 สถานะสุดท้าย, 8
 สมมติฐาน, 149
 สมมติฐานความไม่ขึ้นต่อกัน, 191
 สมมติฐานภายหลังมากที่สุด, 187
 ส่วนหัว, 68, 77
 ส่วนหาง, 77
 สอดคล้องกับ, 148
 สัญพจน์, 68
 สัญพจน์เต็มเต็ม, 57
 สัญลักษณะไม่ปลาย, 98
 สัญลักษณะปลาย, 98
 สูตรรูปดี, 45
 สูตรระดม, 46
 เส้นเชื่อมกับมันต์, 103
 หน่วยความจำระยะยาว, 38
 หน่วยความจำระยะสั้น, 33
 หมวดคำ, 97
 หมายกำหนดการรอบเหิน, 26
 ห้องจีน, 3
 องค์ประกอบ, 103
 อนุประโยค, 54, 67
 อนุประโยคของฮอร์น, 68
 อนุประโยคฐาน, 74
 อนุประโยคพื้นฐาน, 56
 อนุประโยคเรียกซ้ำ, 74
 อนุमान, 49
 อะตอม, 66
 อัลกอริทึม A*, 30
 อัลกอริทึมกฎเดลาต้า, 181
 อัลกอริทึมการเรียนรู้เพอร์เซปตรอน, 173
 อัลกอริทึมการค้นหาแนวกว้างก่อน, 13
 อัลกอริทึมการค้นหาแนวลึกก่อน, 14
 อัลกอริทึมการค้นหาตาม, 41
 อัลกอริทึมการทำให้เท่ากัน, 52
 อัลกอริทึมการปฏิเสธแบบรีโซลูชัน, 59
 อัลกอริทึมการแพร่กระจายย้อนกลับ, 183, 185
 อัลกอริทึมการเรียนรู้แบบอย่างง่าย, 192
 อัลกอริทึมการเรียนรู้เวอร์ชันสเปซ, 150
 อัลกอริทึมการเรียนรู้โดยวิเคราะห์ความแตกต่าง,
 146
 อัลกอริทึมการรอบเหินจำลอง, 25, 28
 อัลกอริทึมของการแจกแจงแบบบนลงล่างอย่างง่าย,
 99

อัลกอริทึมของตัวแรงแสวนตาราง, 104	อัลกอริทึมอีเอ็ม, 202
อัลกอริทึมแรงแสวนแบบบนลงล่างสำหรับอาร์ทีเอ็น, 112	อาเจนดา, 103
อัลกอริทึมดีสุดก่อน, 28, 29	อาร์กิวเมนต์, 46, 65
อัลกอริทึมตะกราม, 34	เอนโทรปี, 160
อัลกอริทึมปีนเขา, 22	เอ็มจียู, 51
อัลกอริทึมปีนเขาขั้นสุด, 24	อีวริสติก, 16
อัลกอริทึมปีนเขาอย่างง่าย, 23	
อัลกอริทึมอบเหนียวจำลอง, 25	

Index

8-Puzzle, 7	algorithm: Version-Space-Candidate-Elimination, 150
A* Search, 30	analytical learning, 168
activation function, 170	annealing schedule, 26
active arc, 103	argument, 46, 65
agenda, 103	artificial neural network, 169
algorithm: Backpropagation, 185	atom, 66
algorithm: Best-First Search, 29	atomic formula, 46
algorithm: Breadth-First Search, 13	attribute, 154
algorithm: Chart Parsing, 104	automatic programming, 4
algorithm: Delta-Rule, 181	backpropagation algorithm, 183
algorithm: Depth-First Search, 14	backtracking, 69
algorithm: EM, 202	base clause, 74
algorithm: Learning by Analyzing Differences, 146	Bayes net, 195
algorithm: Naive-Bayes, 192	Bayesian belief network, 195
algorithm: Perceptron-Learning-Rule, 173	Bayesian learning, 186
algorithm: Resolution Refutation, 59	best-first search, 28
algorithm: RTN Parsing, 112	blind search, 10
algorithm: Simple Hill-Climbing Search, 23	block world problem, 11
algorithm: Simple Top-Down Parsing, 99	body, 68
algorithm: Simulated Annealing Search, 28	breadth-first search, 12
algorithm: Tabu Search, 41	category, 96
algorithm: Unify, 52	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 formular, 45
traveling salesman problem, 16	
