บทที่ 2

การศึกษาลักษณะของตารางตัดสินใจและ เทคนิคการแปลงตารางตัดสินใจ

ตารางตัดสินใจ (Decision Table) ใช้เป็นเครื่องมือแสดงความสัมพันธ์ทางตรรกะระหว่างกันของเหตุการณ์ต่างๆ และเป็นเทคนิคหนึ่งที่รวมเอารูปแบบของรายงาน ผังงาน (Flowchart) Karnaugh map และ Boolean algebra เข้าไว้ด้วยกัน ตารางตัดสินใจจึงมีการนำไปใช้อย่างกว้างขวาง ประโยชน์บางส่วนของตารางตัดสินใจมีดังนี้

1. มีการกำหนดตรรกะอย่างชัดเจนและอยู่ในรูปแบบที่แน่นอน
2. สามารถตรวจสอบได้ว่ามีการข้ามส่วนใดและตรวจสอบความขัดแย้งทางตรรกะได้
3. สามารถแก้ไขตารางได้ง่ายเพื่อที่จะสะท้อนให้เห็นถึงความเปลี่ยนแปลงของภาวะรอบๆ
4. สามารถเขียนโปรแกรมจากตารางตัดสินใจได้โดยอัตโนมัติ
5. ตารางตัดสินใจมีประโยชน์ใช้เป็นเอกสารประกอบการวิเคราะห์ระบบได้
6. ตารางตัดสินใจมีรูปแบบที่สามารถเข้าใจได้ง่ายไม่ว่าจะมีความรู้ทางคอมพิวเตอร์หรือไม่

จากประโยชน์ที่กล่าวมาบางส่วนนี้ จึงมีการศึกษาถึงรายละเอียดของตารางตัดสินใจซึ่งจะกล่าวถึงต่อไป

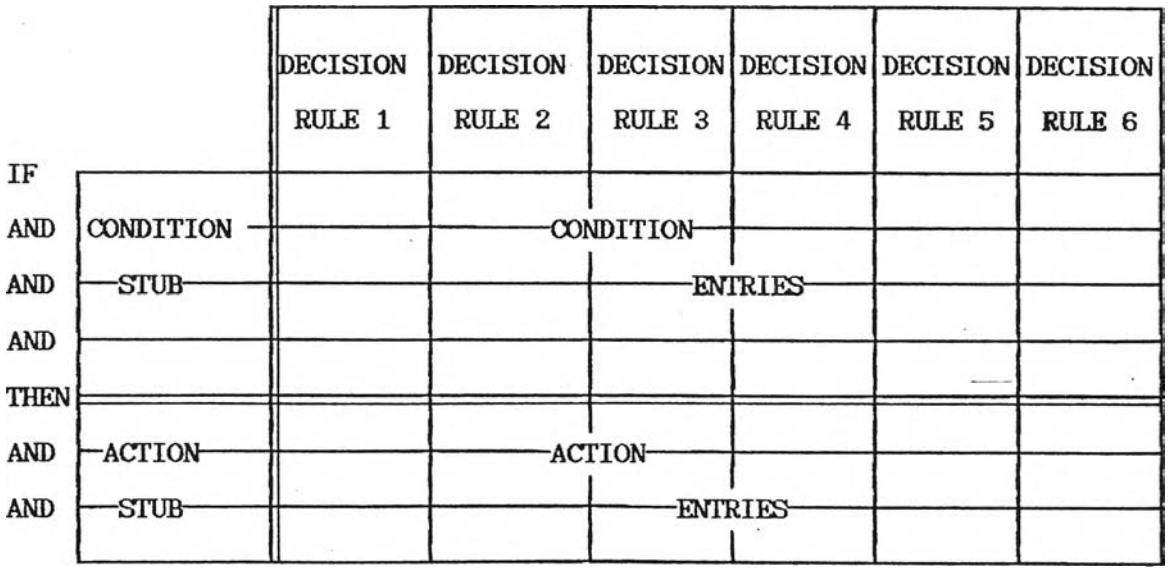
โครงสร้างของตารางตัดสินใจ (Decision Table Structure)

ตารางตัดสินใจคือแมทริกซ์ของแถวและสดมภ์ ที่แสดงถึงเงื่อนไข (Condition) การกระทำ (Action) รวมทั้งกฎการตัดสินใจ (Decision Rule) ตารางตัดสินใจเป็นโครงสร้างที่ใช้อธิบายกลุ่มของกฎการตัดสินใจที่มีความสัมพันธ์กัน กฎการตัดสินใจที่กล่าวถึงนี้คือ ประโยคที่กำหนดชุดของเงื่อนไขที่จะต้องเป็นที่พอใจ (Satisfied) แล้วจะกระทำการ (Execute) ชุดของการกระทำตามลำดับที่กำหนดไว้ในกฎนั้นๆ

รูปแบบของตารางตัดสินใจแบ่งออกเป็น 4 ส่วน คือ

1. Condition Stub
2. Condition Entries
3. Action Stub
4. Action Entries

ดังแสดงไว้ในรูปที่ 2.1



รูปที่ 2.1 แสดงโครงสร้างของตารางตัดสินใจ

จากรูปที่ 2.1 จะเห็นว่าถ้าแบ่งตารางตัดสินใจตามแนวขวางออกเป็น 2 ส่วน คือ ส่วนบนเป็นส่วนของเงื่อนไข และส่วนล่างเป็นส่วนของการกระทำที่เกิดตามมา ถ้าแบ่งตามแนวตั้งตารางจะเป็นส่วนของสทับ (Stub) และ จุดรับเข้า (Entry) ในส่วนของจุดรับเข้ายังแบ่งเป็นสดมภ์ย่อยๆ เรียกว่า กฎการตัดสินใจ แต่ละส่วนนั้นมีรายละเอียดดังนี้

Condition Stub จะอธิบายถึงเงื่อนไขที่สัมพันธ์กับเรื่องที่ศึกษาทั้งหมด

Condition Entries จะแสดงถึงค่าหรือเงื่อนไขอื่นๆ ที่เป็นประโยชน์สำหรับเงื่อนไขแต่ละอันโดยเฉพาะ

Action stub จะเป็นขั้นตอนทั้งหมดที่จะกระทำเมื่อเกิดเหตุการณ์ตามเงื่อนไขที่แน่นอน

Action Entries จะแสดงถึงชุดของการกระทำที่ถูกกำหนดว่าจะต้องกระทำเมื่อชุดของเงื่อนไขที่ถูกเลือกเป็นจริง

กฎการตัดสินใจ (Decision Rule) จะแสดงถึงชุดของการกระทำโดยเฉพาะที่จะถูกกระทำถ้าเงื่อนไขตามที่กำหนดไว้เป็นที่พอใจ (Satisfied) ซึ่งเงื่อนไขและการกระทำจะเกี่ยวข้องกันในลักษณะ "ถ้า ... แล้ว ..."

ประเภทของการตัดสินใจ (Type of Decision Table)

ตารางตัดสินใจแบ่งออกเป็น 3 ชนิดดังต่อไปนี้

1. ตารางตัดสินใจแบบการรับเข้าจำกัด (Limited-Entry Decision Table :LEDT)

ตารางตัดสินใจแบบการรับเข้าจำกัดนี้มีลักษณะที่ค่าการรับเข้าของเงื่อนไข (Condition Entries) มีข้อจำกัดที่จะมีค่าเป็น "Y"(Yes) , "N"(No) และ "-" เท่านั้น โดย "Y" จะแทนเงื่อนไขในสทับเป็นที่พอใจ "N" แทนว่าเงื่อนไขในสทับยังไม่เป็นที่พอใจ และ "-" แทนเงื่อนไขในสทับไม่มีความสำคัญไม่ว่าเงื่อนไขนี้จะเป็นที่พอใจหรือไม่

ในส่วนของการกระทำ การรับเข้าของการกระทำ (Action Entries) มีข้อจำกัดที่จะมีค่าเป็น "X" หรือช่องว่าง โดย "X" แทนการกระทำที่ระบุไว้ในสทับนี้จะถูกกระทำถ้าเงื่อนไขทั้งหมดที่มีในกฎนั้นจะเป็นที่พอใจ ถ้ามีค่าเป็นช่องว่างจะแทนว่าการกระทำในสทับนั้นจะไม่ถูกกระทำไม่ว่าเงื่อนไขทั้งหมดนั้นจะเป็นที่พอใจหรือไม่

ตัวอย่างของตารางตัดสินใจแบบการรับเข้าจำกัด แสดงในตารางที่ 2.1

2. ตารางตัดสินใจแบบการรับเข้าขยาย (Extened-Entry Decision Table :EEDT)

ตารางตัดสินใจแบบการรับเข้าขยายมีลักษณะที่รายละเอียดของเงื่อนไขและการกระทำทั้งหมดจะแสดงทั้งในสทับและจุดรับเข้า การอธิบายเงื่อนไขและการกระทำในสทับนั้นจะอธิบายไว้ไม่หมดโดยบางส่วนของเงื่อนไขและการกระทำจะเอาไปไว้ในส่วนของการรับเข้า การรับเข้าขยายสามารถแสดงความสัมพันธ์ในรูปแบบที่ง่ายและรัดกุมกว่าแบบการรับเข้าจำกัด ดังตัวอย่างในตารางที่ 2.2

3. ตารางตัดสินใจแบบการรับเข้าผสม (Mixed-Entry Decision Table : MEDT)

ตารางตัดสินใจแบบการรับเข้าผสมนี้มีรูปแบบที่รวมลักษณะของตารางตัดสินใจ

แบบการรับเข้าจำกัดและแบบการรับเข้าขยายไว้ในตารางเดียวกัน ดังตัวอย่างแสดงใน ตารางที่ 2.3

ตารางตัดสินใจแบบการรับเข้าผสม และการรับเข้าขยาย ส่วนมากจะแปลงเป็นรูปของตารางตัดสินใจแบบการรับเข้าจำกัด โดยการขยายแถวของการรับเข้าขยายทั้งหมดเป็นกลุ่มของแถวการรับเข้าจำกัดที่มีค่าเท่ากัน (ซึ่งจะกล่าวในรายละเอียดต่อไป) ตารางขยายนี้จะมีขนาดใหญ่ขึ้น การที่ตารางส่วนมากแปลงเป็นตารางแบบการรับเข้าจำกัดเสมอ เพราะการประยุกต์ใช้งานส่วนมากมักสำเร็จโดยใช้ทฤษฎีของตารางตัดสินใจเกี่ยวข้องกับตารางตัดสินใจแบบการรับเข้าจำกัด

ตารางตัดสินใจที่กล่าวมาทั้ง 3 ชนิดนั้น ยังมีอีกส่วนหนึ่งของตารางตัดสินใจคือ ELSE Rule กฎนี้จะอยู่ทางด้านขวาสุดของตารางตัดสินใจ มีวัตถุประสงค์ในการรวมเอากฎที่เกิดขึ้นซ้ำๆ กันจะรวมกันแล้วตัดออกไปรวมไว้ในส่วนของ ELSE Rule นี้ นอกจากนี้แล้วตารางตัดสินใจที่ไม่สมบูรณ์และมีการกำหนดชุดของการกระทำสืบเนื่องสำหรับรายการเปลี่ยนแปลง (Transaction) ที่ไม่เป็นไปตามกฎที่ปรากฏในตารางตัดสินใจจะรวมอยู่ที่ ELSE Rule นี้ โดยที่ ELSE Rule จะมีการกำหนดชุดของการกระทำไว้ด้วย จะเห็นว่าตารางที่มี ELSE Rule นั้นจะสมบูรณ์ เพราะทุกๆ รายการเปลี่ยนแปลงสามารถกำหนดกลุ่มของการกระทำได้ ตัวอย่างอยู่ในตารางที่ 2.4 Pollack ได้แนะนำว่า ELSE Rule ควรใช้เฉพาะกับกฎที่มีความผิดพลาด (Error) เท่านั้น ไม่ได้ใช้กับกฎที่เหลือในตาราง ซึ่ง ELSE Rule นี้มีประโยชน์มากในตารางตัดสินใจโดยเฉพาะอย่างยิ่งเมื่อมีเงื่อนไขเป็นจำนวนมาก เนื่องจากการผสมของเงื่อนไขซึ่งอาจจะอยู่ในรูปที่ไม่น่าเป็นไปได้จำนวนมาก โดยเกี่ยวข้องกับส่วนของโปรแกรมที่จัดการกับข้อผิดพลาด 1 ชุดคำสั่งกระทำความผิดพลาด ดังนั้นการใช้ ELSE Rule ควร ใช้อย่างระมัดระวัง

ตัวอย่างของตารางตัดสินใจทั้งหมดที่กล่าวมาแล้ว ยกตัวอย่างจากงานทางธุรกิจ การทำธุรกิจส่วนมากจะให้ส่วนลดในการซื้อสินค้า ซึ่งมักมีเงื่อนไขในการให้ส่วนลด ร้านค้าแห่งหนึ่งมีกฎเกณฑ์ในการให้ส่วนลดโดยดูจากใบสั่งสินค้าแบ่งได้เป็น 3 ชนิดคือ ใบสั่งที่มากกว่า 10,000 บาท ตั้งแต่ 5,000-10,000 บาท และน้อยกว่า 5,000 บาท ซึ่งให้ส่วนลดดังนี้ 3% , 2% และไม่ลดราคา ตามลำดับ ทั้งนี้การลดราคาจะลดต่อเมื่อจ่ายเงินหลังจากได้รับสินค้าภายใน 10 วัน นำมาแสดงเป็นตารางตัดสินใจได้ดังนี้

กฎการตัดสินใจ

เงื่อนไข	1	2	3	4	5	6
จ่ายเงินภายใน 10 วัน	Y	Y	Y	N	N	N
สั่งซื้อ > 10,000 บาท	Y	N	N	Y	N	N
สั่งซื้อ 5,000-10,000 บาท	N	Y	N	N	Y	N
สั่งซื้อ < 5,000 บาท	N	N	Y	Y	N	Y
ให้ส่วนลด 3%	X					
ให้ส่วนลด 2%		X				
ไม่ลดราคา			X	X	X	X

ตารางที่ 2.1 แสดงตารางตัดสินใจแบบการรับเข้าจำกัด

กฎการตัดสินใจ

เงื่อนไข	1	2	3	4	5	6
ระยะเวลา (วัน) จำนวนที่สั่ง (บาท)	< 10 >10,000	< 10 5,000– 10,000	< 10 <5,000	> 10 >10,000	>10 5,000– 10,000	>10 <5,000
ส่วนลด	3%	2%	0%	0%	0%	0%

ตารางที่ 2.2 แสดงตารางตัดสินใจแบบการรับเข้าขยาย

กฎการตัดสินใจ

เงื่อนไข	1	2	3	4	5	6
ระยะเวลา (วัน) จำนวนที่สั่ง (บาท)	< 10 >10,000	< 10 5,000– 10,000	< 10 <5,000	> 10 >10,000	>10 5,000– 10,000	>10 <5,000
ให้ส่วนลด 3% ให้ส่วนลด 2% ไม่ลดราคา	X	X	X	X	X	X

ตารางที่ 2.3 แสดงตารางตัดสินใจแบบการรับเข้าผสม

กฎการตัดสินใจ

เงื่อนไข	1	2	E L S E
ระยะเวลา (วัน) จำนวนที่สั่ง (บาท)	< 10 >10,000	< 10 5,000– 10,000	
ให้ส่วนลด 3% ให้ส่วนลด 2% ไม่ลดราคา	X	X	X

ตารางที่ 2.4 แสดงตารางตัดสินใจที่มี ELSE Rule

ทฤษฎีของตารางตัดสินใจ

1. เงื่อนไข ความสัมพันธ์ของเงื่อนไขในตารางตัดสินใจสามารถมองในรูปที่แต่ละเงื่อนไขเชื่อมกันโดยวิธีและ (AND) ได้เช่น "IF not cond-1 AND cond-2 AND cond-3 THEN execute a series of action " ซึ่งเงื่อนไขมีสมมติฐาน 2 ข้อคือ

1.1 เงื่อนไขแต่ละแถวไม่มีความสัมพันธ์กับเงื่อนไขแถวอื่นๆ นั่นคือในกฎการตัดสินใจใดๆ การตัดสินใจของเงื่อนไขหนึ่งไม่สามารถบอกถึงการตัดสินใจที่เป็นไปได้ของเงื่อนไขอื่นใดในกฎนั้นได้ สมมติฐานนี้เป็นพื้นฐานของงานที่สำคัญของการ Optimization

1.2 สำหรับการรับเข้าของการรับเข้าจำกัด แต่ละค่าในแถวของเงื่อนไขจะต้องเหมือนกันกับค่าอื่นๆ ของเงื่อนไขนั้น หรือไม่ก็ไม่เหมือนกัน หรือเป็นค่า dash (ไม่สนใจค่านั้น) คือการรับเข้าแบบจำกัดนั้น เงื่อนไขแต่ละแถวจะมีค่าเป็น "Y", "N" หรือ "-" นั่นเอง

2. การกระทำ คือขั้นตอนการดำเนินการ (Operational Step) ที่รวบรวมตามลำดับที่ระบุเป็นขั้นตอนการดำเนินการที่ถูกต้อง ขั้นตอนของการดำเนินการจะมีลักษณะเป็นข้อความคำสั่งกระบวนการ (Procedure Statement) เช่นตัวอย่างที่ใช้อธิบายในรูปภาษาโคบอล ดังนี้

MOVE A TO B. หรือ

MULTIPLY GROSS BY RATE GIVING TAX. เป็นต้น

นอกจากนี้การกระทำอาจอยู่ในรูปของกลุ่มของ Sentences หรือ Statement ที่เป็นเงื่อนไข ดังตัวอย่างต่อไปนี้

PERFORM MEAN-CALCULATION, ADD DEVIATION TO MEAN, GO TO REPORT-TABLE. หรือ GO TO PN1, PN2, PN3 DEPENDING ON VECT.

สิ่งสำคัญสำหรับข้อความของขั้นตอนการดำเนินการที่ใช้กับการกระทำนั้นจำเป็นต้องเป็นประโยคที่คอมพิวเตอร์ยอมรับตามภาษาที่กำหนดนั้นๆ

3. กฎการตัดสินใจ กฎการตัดสินใจมี 2 ชนิดคือ

3.1 กฎธรรมดา (Simple Rule) คือกฎที่มีค่าของการรับเข้าเงื่อนไขเป็น "Y" หรือ "N" เท่านั้น

3.2 กฎที่ซับซ้อน (Complex Rule) คือกฎที่มีค่าของการรับเข้าเงื่อนไขที่เป็น "-" 1 ค่าหรือหลายๆ ค่า

กฎในตารางตัดสินใจอาจมีทั้งกฎธรรมดาและกฎที่ซับซ้อน รวมกันอยู่ในตาราง ซึ่งกฎซับซ้อนสามารถขยายเป็นกฎธรรมดาได้ ตัวอย่างดังในรูปที่ 2.2 กฎธรรมดา 2 กฎ สามารถรวมกันเป็นกฎซับซ้อนได้ ซึ่งเรียกว่า Consolidation การรวมกันนั้นกฎทั้งสองจะต้องมีค่าของเงื่อนไขเพียง 1 ค่าเท่านั้นที่มีค่าที่แตกต่างกัน มีค่าเป็น "Y" และ "N" และกฎทั้งสองกฎนั้นจะต้องมีการกระทำที่เหมือนกันด้วย

Y		Y	Y		Y	N		Y	Y		N	N
-	----->	Y	N		N	Y	----->	N	N		Y	Y
N		N	N		-	Y		Y	N		Y	Y
Y		Y	Y		Y	-		Y	Y		Y	N
(ก)						(ข)						

รูปที่ 2.2 แสดงการขยายกฎซับซ้อน

กฎ 2 กฎจะถูกเรียกว่า Disjoint ถ้ากฎทั้งสองกฎไม่มีกฎธรรมดาร่วมกัน ดังรูปที่ 2.2 (ข) เมื่อขยายกฎซับซ้อน 2 กฎแล้วพบว่าได้กฎธรรมดาที่แตกต่างกัน เมื่อกฎ 2 กฎนั้นไม่ Disjoint กันจะเรียกว่าเกิดกฎซ้อนทับ (Overlapping) กฎที่เกิดการซ้อนทับกันจะทำให้เกิดปัญหาขึ้น 2 อย่างคือ เกิดความซ้ำซ้อน (Redundancy) หรือ เกิดความขัดแย้งกัน (Contradiction) บางครั้งจะรวมเรียกกฎที่ซ้อนทับกันว่า กฎกำกวม (Ambiguity) ซึ่งมีรายละเอียดดังนี้

ความซ้ำซ้อน เกิดขึ้นเมื่อกฎการตัดสินใจ 2 กฎที่มีเงื่อนไขที่เหมือนกันหมด ถ้าทั้งสองกฎถูกเลือกจะได้การกระทำสืบเนื่องที่เหมือนกัน

ความขัดแย้งกัน เกิดขึ้นเมื่อกฎ 2 กฎหรือมากกว่ามีกลุ่มของเงื่อนไขที่เหมือนกันหมด แต่การกระทำสืบเนื่องแตกต่างกัน ซึ่งแสดงให้เห็นว่าสารสนเทศของนักวิเคราะห์ระบบผิดอาจเนื่องมาจากในขั้นตอนการสัมภาษณ์ หรือได้ข้อมูลมาผิดพลาดต้องมีการกลับไปตรวจสอบ ความผิดพลาดนี้อาจเกิดในขณะสร้างตาราง เมื่อพบจะต้องแก้ไขให้ถูกต้อง

ตัวอย่างของความขัดแย้งที่แสดงในตารางที่ 2.5 จากกฎซับซ้อน 2 กฎเมื่อขยายแล้วจะเกิดกฎธรรมดา 6 กฎ ซึ่งพบว่ามีกฎ {Y,Y,Y,N} เกิดขึ้น 2 ครั้ง ซึ่งมีการกระทำที่แตกต่างกันแสดงว่าเกิดความขัดแย้ง การแก้ปัญหาโดยพิจารณาตารางใหม่อีกครั้งและกำหนดกฎของของตารางใหม่ไม่ให้เกิดความกำกวม ดังแสดงในตารางที่ 2.5 ทางด้านขวา

C1	Y	Y	Y	Y	Y	Y	Y	Y	C1	Y	Y	Y	Y
C2	-	Y	Y	Y	N	N	Y	Y	C2	Y	Y	N	Y
C3	Y	-	Y	Y	Y	Y	Y	N	C3	Y	Y	Y	N
C4	-	N	Y	N	Y	N	N	N	C4	Y	N	-	N
===	=======		===	===	======		===	===	===	==========			
A1	X		X	X	X	X			A1	X	X	X	
A2		X					X	X	A2		X		X
A3	X	X	X	X	X	X	X	X	A3	X	X	X	X

ตารางที่ 2.5 แสดงการเกิดกฎขัดแย้งกัน

ตัวอย่างของความซ้ำซ้อน ความซ้ำซ้อนเป็นปัญหาหนึ่งที่เกิดขึ้นเมื่อกฎมีการซ้อนทับกันโดยมีการกระทำที่เหมือนกัน แต่ยังไม่เป็นปัญหาที่สำคัญเท่ากับความขัดแย้งกัน การแก้ไขก็โดยการตัดกฎที่ซ้ำซ้อนออก เพื่อให้กฎเป็น Disjoint กัน ดังแสดงในตารางที่ 2.6

C1	Y	Y	Y	Y	Y	Y	C1	Y	Y
C2	-	N	Y	N	N	N	C2	Y	N
C3	Y	Y	Y	Y	Y	Y	C3	Y	Y
C4	N	-	N	N	Y	N	C4	N	-
===	=======		===	===	===	===	===	======	
A1	X	X	X	X	X	X	A1	X	X

ตารางที่ 2.6 แสดงการเกิดกฎซ้ำซ้อน

ความสมบูรณ์ (Completeness) ของตารางตัดสินใจ การพิจารณาว่าตารางที่ได้รวบรวมเงื่อนไขที่เป็นไปได้ทั้งหมดเข้าไว้ครบหรือไม่ พิจารณาจากกฎเกณฑ์ 2 ข้อดังนี้

1. กฎชีบซ้อน มี d dashes สามารถขยายเป็นกฎธรรมดาได้เท่ากับ 2^d กฎ

2. ตารางที่มี c เงื่อนไข จะสมบูรณ์ถ้ามีกฎธรรมดาเป็นกฎแบบ Disjoint 2^c กฎ

ดังตัวอย่างในตารางที่ 2.7 มีกฎธรรมดาที่เกิดจากข้อ 1 ได้ 14 กฎ ตารางนี้ไม่สมบูรณ์เพราะตารางที่สมบูรณ์จะต้องมี 2^4 = 16 disjoint simple rules ซึ่งตารางนี้เมื่อขยายเป็น 14 กฎธรรมดาจะมีกฎที่ขาดหายไป 2 กฎคือ {Y,N,N,N} และ {N,N,Y,Y}

C1	Y	Y	Y	N	N	N	
C2	Y	N	N	Y	-	-	
C3	-	Y	N	Y	Y	N	
C4	-	-	Y	Y	N	-	
Simple Rule	4	2	1	1	2	4	====> 14

ตารางที่ 2.7 แสดงตัวอย่างของการพิจารณาความสมบูรณ์ของตารางตัดสินใจ

นอกจากที่กล่าวมาทั้งหมดนี้แล้วยังมีอีกสิ่งหนึ่งที่จะกล่าวถึงคือลักษณะของ Implied Entries เกิดเมื่อเงื่อนไขแต่ละแถวมีความสัมพันธ์กัน พิจารณาตารางที่ 2.8 (ก) ตารางนี้ดูเหมือนว่าเกิดความขัดแย้งกัน แต่ถ้าเราพิจารณาจากเงื่อนไขจะเห็นว่าเงื่อนไขที่ 1 และ เงื่อนไขที่ 2 ไม่เป็นอิสระต่อกันนั่นก็คือ ถ้าเงื่อนไขที่ 1 เป็นจริง เงื่อนไขที่ 2 จะเป็นเท็จไปโดยปริยายและในทำนองเดียวกันแต่กลับกันถ้าเงื่อนไขที่ 1 เป็นเท็จ เงื่อนไขที่ 2 จะเป็นจริงนั่นคือรายการเปลี่ยนแปลง {Y,Y} ไม่มีทางเกิดขึ้น ปัญหานี้มักเกิดเรียกว่า "Apperent Ambiguity" จะเป็นการชี้บอกว่าการรับเข้าเงื่อนไขจะถูกกำหนดค่าโดยความสัมพันธ์ระหว่างเงื่อนไข ซึ่งมีทางปรับปรุงโดยใส่วงเล็บให้กับค่านี้เพื่อที่จะแสดงว่าถูกกำหนดค่าโดยปริยายจากอีกค่าในตารางหรือเป็น Implied Entry ดังตัวอย่างในตารางที่ 2.8 (ข)

Driver	R1	R2	R3
AGE > 17	Y	-	N
AGE < 70	-	Y	N
GO TO	1	2	3

(ก)

	R1	R2	R3
	Y	(N)	N
	(N)	Y	N
	1	2	3

(ข)

ตารางที่ 2.8 แสดงตารางที่มี Implied Entry

<u>การสร้างตารางตัดสินใจ</u> (Construction of Decision Tables)

การสร้างตารางตัดสินใจมี 2 วิธีคือ

1. Classical Technique

เริ่มต้นด้วยการเขียนเงื่อนไขและการกระทำที่จำเป็นทั้งหมดออกมา ต่อมาจะทำการผสมเงื่อนไขที่เป็นไปได้แต่ละชุดแล้วระบุการกระทำที่เหมาะสมให้ เมื่อได้ตารางที่สมบูรณ์แล้วกฎจะถูกรวบรวมอยู่ในรูปตารางสุดท้ายที่รัดกุมยิ่งขึ้น จะมีการตรวจความกำกวม ซึ่งจะกล่าวถึงวิธีการนี้โดยสรุปได้ดังนี้

1.1 สร้าง Condition Stub ให้สมบูรณ์โดยการเขียนเงื่อนไขที่เป็นไปได้ทั้งหมด

1.2 สร้าง Condition Entries อย่างเป็นระเบียบ โดยการเขียนกฎธรรมดาที่เป็นไปได้ทั้งหมดตามแบบธรรมดา โดยจะใส่ค่าทั้งหมด ดังตัวอย่างในตารางที่ 2.9 ซึ่งจะแสดงโครงสร้างทั่วไปของตารางที่มีเงื่อนไข 3 เงื่อนไข

1.3 พิจารณากฎธรรมดาแต่ละกฎ และกำหนดการกระทำที่เหมาะสม โดยการสร้างสทับการกระทำและการรับเข้าของการกระทำ การเติมค่าของกากระทำควรระมัดระวังลำดับที่ของการกระทำ ดังตัวอย่างในตารางที่ 2.10 ซึ่งในตัวอย่างนี้มีค่า 2 ค่าที่มีตัวเลขกำกับแสดงถึงการใส่การกระทำที่เรียงลำดับผิด

1.4 เมื่อเขียนกฎและกลุ่มของการกระทำที่เป็นไปได้ทั้งหมดแล้วจัดการรวบรวมกฎที่มีชุดของการกระทำเดียวกันเป็นรูปกฎซับซ้อน ในเวลาเดียวกันให้จัดเรียงลำดับใหม่ตามความเหมาะสม ดังตัวอย่างในตารางที่ 2.10 กฎที่ 2 และ 4 สามารถรวมกันได้ กฎที่ 5 และ 7 สามารถรวมกันได้ กฎที่ 6 และ 8 ควรเอาออกจากตารางเพราะตรรกะที่เขียนเป็นไปไม่ได้ ตารางที่ได้รวบรวมเรียบร้อยและได้จัดลำดับการกระทำใหม่ดังแสดงในตารางที่ 2.11

C2	;;	Y	Y	Y	Y	N	N	N	N
C3	;;	Y	Y	N	N	Y	Y	N	N
C4	;;	Y	N	Y	N	Y	N	Y	N
-----+	+	----	----	----	----	----	----	----	----
	;;								

ตารางที่ 2.9 แสดงการสร้างการรับเข้าของเงื่อนไขด้วยวิธี Classical

Quadratic Roots	R1	R2	R3	R4	R5	R6	R7	R8
C1: a = 0	Y	Y	Y	Y	N	N	N	N
C2: b > 0	Y	Y	N	N	Y	Y	N	N
C3: b*b-4*a*c > 0	Y	N	Y	N	Y	N	Y	N
a1: d := b*b-4*a*c	X		X					
a2: x_1 := (-b-sqrt(d))/(2*a)	X							
a3: x_2 := c/(a*x_1)	X		X^2					
a4: write('complex roots')		X		X				
a5: x_1 := (-b+sqrt(d))/(2*a)			X^1					
a6: write('not quardratic')					X		X	
Impossible						X		X

ตารางที่ 2.10 แสดงการใส่การกระทำในตารางตัดสินใจด้วยวิธี Classical

Quadratic Roots	R1	R2	R3	R4
C1: a = 0	Y	Y	Y	N
C2: b > 0	Y	N	-	-
C3: b*b-4*a*c > 0	Y	Y	N	Y
a1: d := b*b-4*a*c	X	X		
a2: x_1 := (-b-sqrt(d))/(2*a)	X			
a3: x_1 := (-b+sqrt(d))/(2*a)		X		
a4: x_2 := c/(a*x_1)	X	X		
a5: write('complex roots')			X	
a6: write('not quardratic')				X

ตารางที่ 2.11 แสดงตารางที่สร้างสมบูรณ์ด้วยวิธี Classical

ตารางตัดสินใจดังตารางที่ 2.11 นั้นกฎที่ 4 ตามความเป็นจริงควรแทนที่ด้วย {N,-,-} เพราะเป็นไปไม่ได้ที่จะเกิด {N,-,N} ดังนั้นเงื่อนไขที่ 3 ไม่อยู่ในประเด็นที่จะต้องทดสอบ

ข้อเสียของการสร้างตารางตัดสินใจด้วยวิธีการนี้คือ เป็นไปได้ที่จะต้องมีการตรวจสอบกฎที่เป็นไปไม่ได้จำนวนมาก จากตัวอย่างตารางมี 6 เงื่อนไข ซึ่งจะขยายเป็นกฎธรรมดาได้ 2^6 = 64 กฎ และทุกๆ กฎจะถูกกำหนดการกระทำหรือไม่ก็บอกว่ากฎนั้นเป็นไปไม่ได้ และตารางสุดท้ายอาจมีกฎที่สัมพันธ์กับปัญหาเพียง 12 กฎเท่านั้น ในกรณีเช่นนี้จำเป็นที่จะ ต้องตรวจสอบทุกการผสม (combination) ซึ่งจุดประสงค์หลักคือความสมบูรณ์ของตาราง

2. วิธี Progressive Rule Development

วิธีการนี้จะสร้างตารางตัดสินใจโดยการผสมเงื่อนไขทีละเงื่อนไขในแต่ละครั้งดังในตารางที่ 2.12 แสดงถึงขั้นตอนในการสร้างตารางตัดสินใจสำหรับปัญหา "roots of the quadratic" ซึ่งเป็นตัวอย่างเดียวกันกับวิธีแรก ในตารางตัดสินใจที่ 2.12 พิจารณาเงื่อนไขพื้นฐานคือ b^2 > 4*a*c ต่อมาในตารางที่ 2.13 ได้ตัดข้อความเป็นไปได้ของการหารด้วย 0 ออก ตารางที่ 2.14 เป็นการคำนวณค่าของ root ถูกแก้ไขเพื่อหลีกเลี่ยงค่าผิดพลาดปัดเศษ (Rounding Error) จะเห็นว่าผลลัพธ์ของตารางที่ได้จากการสร้างด้วยวิธีนี้เหมือนกับ ในตารางที่ 2.10 ถึงแม้โครงสร้างจะต่างกัน เพื่อที่จะให้แน่ใจถึงความสมบูรณ์ของตาราง ตามกฎเกณฑ์การตรวจสอบทั่วไป กฎธรรมดา 2^3 = 8 กฎ จากตัวอย่างได้แสดงถึงจุดที่เกี่ยวข้องจำนวนน้อยที่สุดของการทดสอบที่ต้องการในตาราง ตารางที่ 2.15 แสดงตารางที่ใช้เงื่อนไขและการกระทำโดยใช้ตัวเลขของตารางที่ 2.14

Quadratic Roots	R1	R2
b*b-4*a*c > 0	Y	N
roots = (-b+sqrt(b*b-4*a*c))/(2*a)	X	
write('complex roots')		X

ตารางที่ 2.12 แสดงการสร้างตารางตัดสินใจแบบ Progressive ขั้นที่ 1

Quadratic Roots	R1	R2	R3
b*b-4*a*c > 0	Y	Y	N
a = 0	Y	N	-
roots = (-b+sqrt(b*b-4*a*c))/(2*a)	X		
write('complex roots')			X
write('not quadratic')		X	

ตารางที่ 2.13 แสดงการสร้างตารางตัดสินใจแบบ Progressive ขั้นที่ 2

Quadratic Roots	R1	R2	R3	R4
C1: b*b-4*a*c > 0	Y	Y	Y	N
C2: a = 0	Y	Y	N	-
C3: b > 0	Y	N	-	-
a1: x_1 := (-b-sqrt(d))/(2*a)	X			
a2: x_1 := (-b+sqrt(d))/(2*a)		X		
a3: x_2 := c/(a*x_1)	X	X		
a4: write('complex roots')				X
a5: write('not quardratic')			X	

ตารางที่ 2.14 แสดงการสร้างตารางตัดสินใจแบบ Progressive ขั้นที่ 3

	R1	R2	R3	R4
C1	Y	Y	Y	N
C2	Y	Y	N	-
C3	Y	N	-	-
A1	X			
A2		X		
A3	X	X		
A4				X
A5			X	

ตารางที่ 2.15 แสดงตารางตัดสินใจที่สร้างสมบูรณ์ด้วยวิธี Progressive

<u>เทคนิคการแปลงตารางตัดสินใจ</u>

การแปลงตารางตัดสินใจเป็นโปรแกรมคอมพิวเตอร์มีอยู่หลายวิธีซึ่งในที่นี้ถ้าจะแบ่งเป็นวิธีหลักๆ ได้ 2 วิธีการคือ

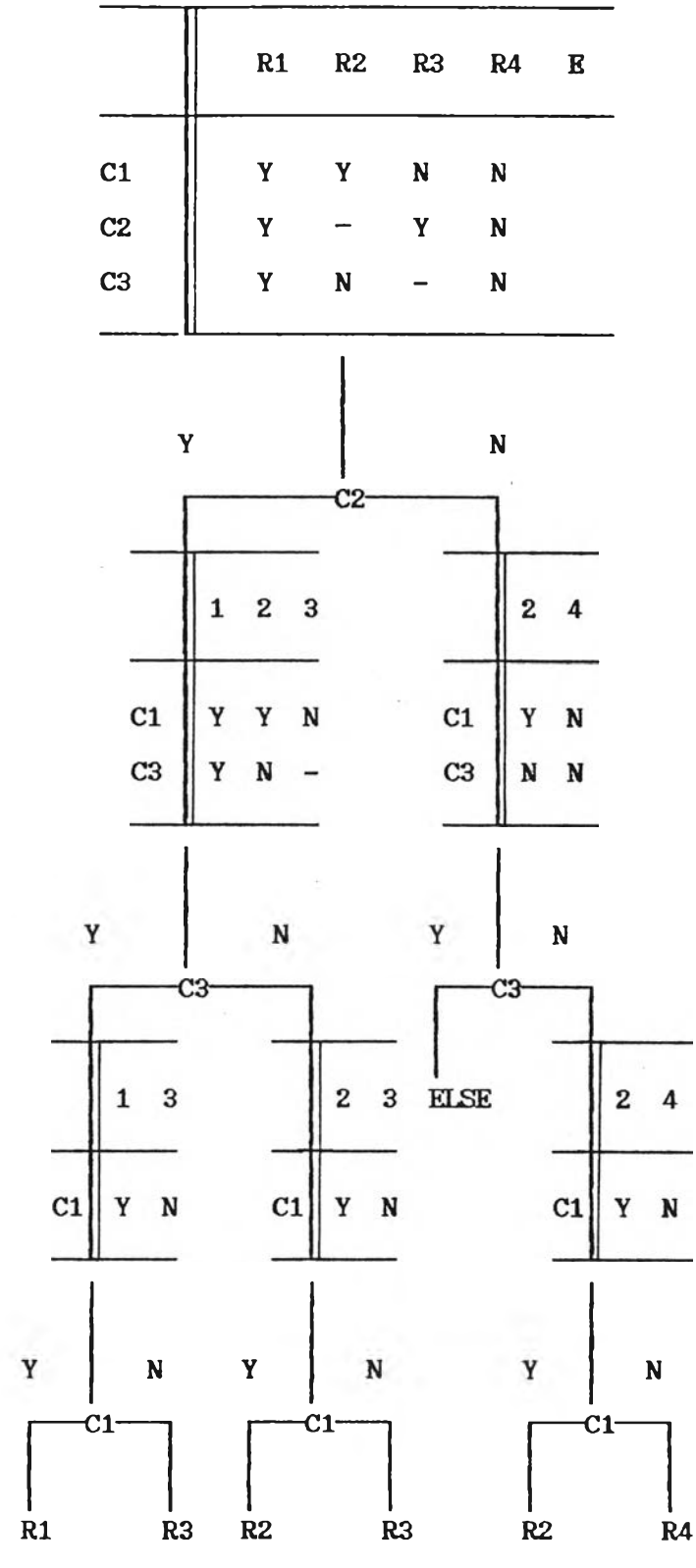
1. วิธีการแปลงตารางตัดสินใจโดยการสร้างเป็นต้นไม้ตัดสินใจ (Decision Tree Technique)

2. วิธีรูมาสค์ (Rule Mask technique)

ซึ่งมีรายละเอียดของแต่ละวิธีดังต่อไปนี้

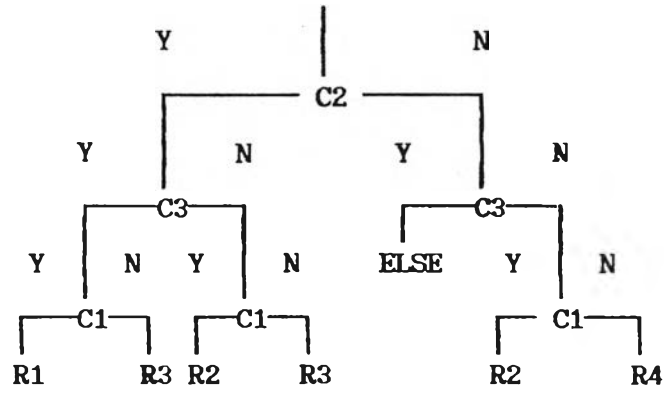
1. การแปลงเป็นต้นไม้ตัดสินใจ (Decision Tree Technique)

วิธีการนี้คือการแปลงจากตารางตัดสินใจให้อยู่ในรูปต้นไม้ตัดสินใจ (Decision Tree Structure) และจะถูกแปลงเข้ารหัส (Encode) อย่างเป็นระบบ จะสร้างจากตารางตัดสินใจแบบการรับเข้าจำกัด ไปอยู่ในรูปของต้นไม้ตัดสินใจแบบทวิภาค (binary decision trees) โดยเงื่อนไขที่ทดสอบจะถูกแทนที่ที่แต่ละจุดแตกกิ่ง (node) ของต้นไม้ลักษณะของการสร้าง เริ่มต้นด้วยการเลือกเงื่อนไขหนึ่งคือ C_1 และตัดเงื่อนไขนั้นออกจากตารางตัดสินใจที่กำหนดให้ จากนั้นก็สร้างตารางย่อย (Subtables) 2 ตาราง ดังนี้ ตารางแรกจะเป็นตารางที่มีค่า C_1 เป็นจริง ("Y" หรือ "-") และ อีกตารางจะมีค่า C_1 เป็นเท็จ ("N" หรือ "-") วิธีการนี้จะใช้กับตารางย่อยแต่ละตารางเพื่อที่จะสร้างตารางย่อยต่อๆ ไปอีก ซึ่งจะสร้างเป็นรูปต้นไม้แบบทวิภาค (binary tree) โดยแต่ละส่วนของจุดแตกกิ่งของต้นไม้จะแสดงการทดสอบเงื่อนไข ส่วนของจุดแตกกิ่งสุดท้าย (terminal node) จะบอกถึงกฎที่ตรงกับในตารางตัดสินใจ ขั้นตอนวิธี (Algorithm) นี้จะถูกทำงานหลายครั้งเพื่อให้ได้ต้นไม้ตัดสินใจที่เหมาะสมจากตารางที่กำหนดขบวนการนี้เรียกอีกอย่างว่าวิธีไบเฟอร์เคชั่น (Bifurcation) ตัวอย่างของการแปลงตารางตัดสินใจเป็นต้นไม้ตัดสินใจแสดงในรูปที่ 2.3



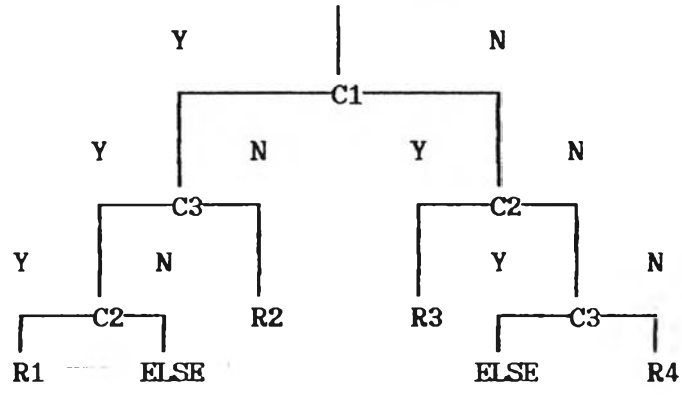
รูปที่ 2.3 แสดงการแปลงตารางตัดสินใจเป็นต้นไม้ตัดสินใจ

จากรูปที่ 2.3 ถ้าตัดตารางย่อยออกจะได้ต้นไม้ตัดสินใจดังรูปที่ 2.4



รูปที่ 2.4 แสดงต้นไม้ตัดสินใจที่ได้จากรูปที่ 2.3 แล้วตัดตารางย่อยออก

การสร้างต้นไม้ตัดสินใจนั้นจากรูปที่ 2.4 เป็นเพียงหนึ่งในต้นไม้ตัดสินใจที่เป็นไปได้ทั้งหมดจากตารางตัดสินใจที่กำหนด จากตารางตัดสินใจเดียวกันนี้ถ้าเริ่มต้นสร้างต้นไม้ตัดสินใจโดยการทดสอบเงื่อนไข C1 เป็นเงื่อนไขแรกจะได้ต้นไม้ตัดสินใจดังรูปที่ 2.5



รูปที่ 2.5 แสดงต้นไม้ตัดสินใจที่เลือกทดสอบเงื่อนไขที่ 1 เป็นเงื่อนไขแรก

จากตัวอย่างทั้งสองตัวอย่างนำมาสรุปหลักในการแปลงจากตารางตัดสินใจเป็นต้นไม้ตัดสินใจได้ดังนี้

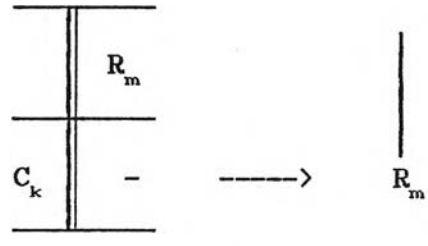
1.1 เลือกเงื่อนไข C_i สำหรับแต่ละกฎที่มีค่าของเงื่อนไข C_i เป็น "Y" (Yes) หรือ "-" ให้ลอกกฎนั้นลงในตารางย่อย Y และตัดแถวเงื่อนไข C_i ออกจากตารางย่อย

สำหรับแต่ละกฎที่มีค่าของเงื่อนไข C_i เป็น "N" (No) หรือ "-" ให้ลอกกฎนั้นลงตารางย่อย N โดยตัดแถวของเงื่อนไข C_i ออกจากตารางย่อย

1.2 สำหรับตารางย่อยที่มีแถวมากกว่า 1 แถวให้ทำขั้นตอนที่ 1.1 ซ้ำ

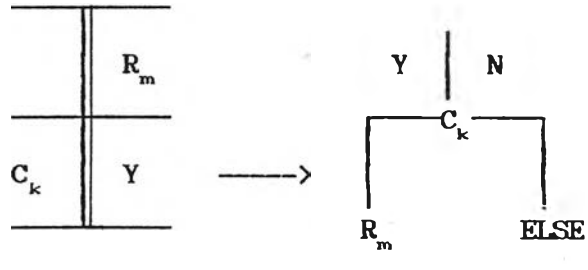
1.3 รายละเอียดปลีกย่อยของตารางย่อยที่สร้างขึ้น

1.3.1 ตารางย่อยที่มีขีด 1 ขีด ดังแสดงในรูปที่ 2.6 แสดงว่าเงื่อนไข C_k ไม่มีความสัมพันธ์กับกฎนี้ ดังนั้นสาขานี้สามารถแทนที่ได้ด้วย R_m ดังรูปที่สาขาจะชี้ไปยังกฎที่ M



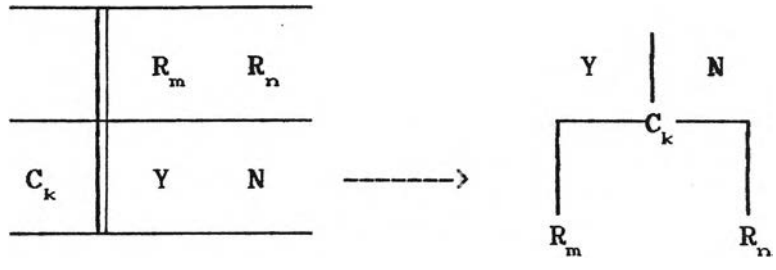
รูปที่ 2.6 แสดงตารางย่อยที่มี dash 1 รายการ

1.3.2 ตารางที่บรรจุการรับเข้าที่มีค่าแน่นอน ดังตัวอย่างในรูปที่ 2.7 ในกรณีนี้สาขาของ Y มาจากการทดสอบเงื่อนไข C_k จะชี้ไปยังกฎ R_m ในขณะที่สาขา N จะเป็น "ELSE" ในทำนองเดียวกัน ถ้าการรับเข้าเดียวกันนี้มีค่าเป็น "N" ดังนั้นสาขาของ Y จะเป็น "ELSE" ในขณะที่สาขา N จะเป็น R_m



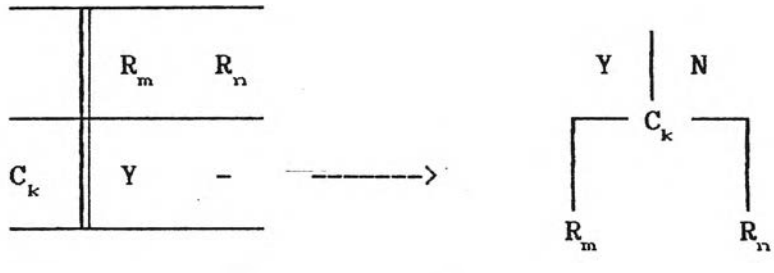
รูปที่ 2.7 แสดงตารางย่อยที่มีค่าที่แน่นอน

1.3.3 ตารางย่อยที่บรรจุค่า "Y" และ "N" ดังตัวอย่างในรูปที่ 2.8 จะเหลือเงื่อนไขที่ใช้ทดสอบคือ C_k ถ้าเป็น "Y" สาขา Y จะชี้ไปที่กฎ R_m และสาขา N ชี้ไปที่กฎ R_n



รูปที่ 2.8 แสดงตารางย่อยที่มีค่า "Y" และ "N"

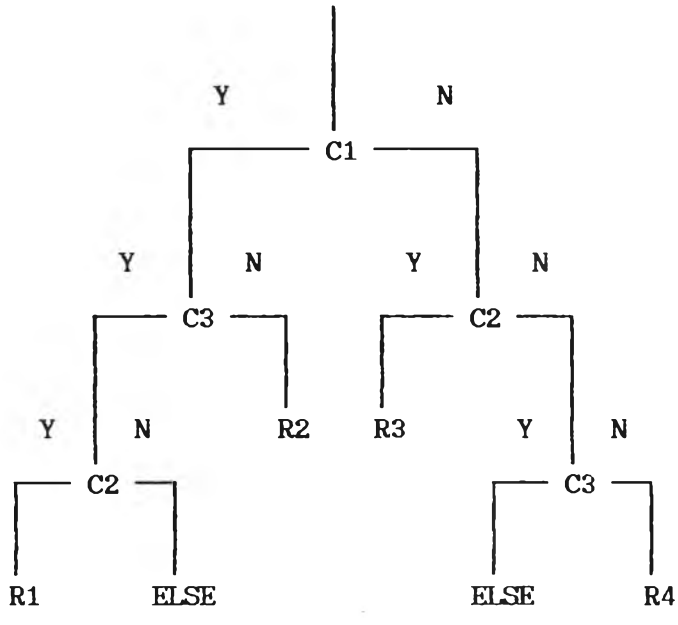
1.3.4 ตารางย่อยที่มีค่ามากกว่า 1 ค่า และรายการนั้นๆมีค่าเป็น "–" ดังรูปที่ 2.9 ในกรณีนี้แสดงว่าเกิดความกำกวมในตารางตัดสินใจที่กำหนด เพราะ C_k ="Y" จะทำให้ชี้ไปยังกฎ R_m และ R_n การกระทำที่จะถูกกระทำขึ้นอยู่กับการแปลความหมายของตารางตัดสินใจที่ใช้ ดังนั้นควรจะทำการวิเคราะห์ปัญหาใหม่และสร้างตารางตัดสินใจใหม่



รูปที่ 2.9 แสดงตารางย่อยที่มีรายการเป็น "–"

เมื่อสร้างต้นไม้ตัดสินใจจากตารางตัดสินใจเสร็จเรียบร้อยจะนำไปสร้างเป็นโปรแกรมต่อไป โดยเลือกใช้ภาษาคอมพิวเตอร์ตามจะมุ่งหมายที่จะใช้งานดังตัวอย่างต่อไปนี้เป็นการแปลงตารางตัดสินใจเป็นต้นไม้ตัดสินใจและเปลี่ยนเป็นโปรแกรมภาษาปาสคาล (Pascal) ดังรูปที่ 2.10

	R1	R2	R3	R4	E
C1	Y	Y	N	N	
C2	Y	-	Y	N	
C3	Y	N	-	N	




```
var
     c1, c2, c3  : boolean;
....
....
if C1 then
      if C2 then
            if C3 then
                  { action set R1 }
```

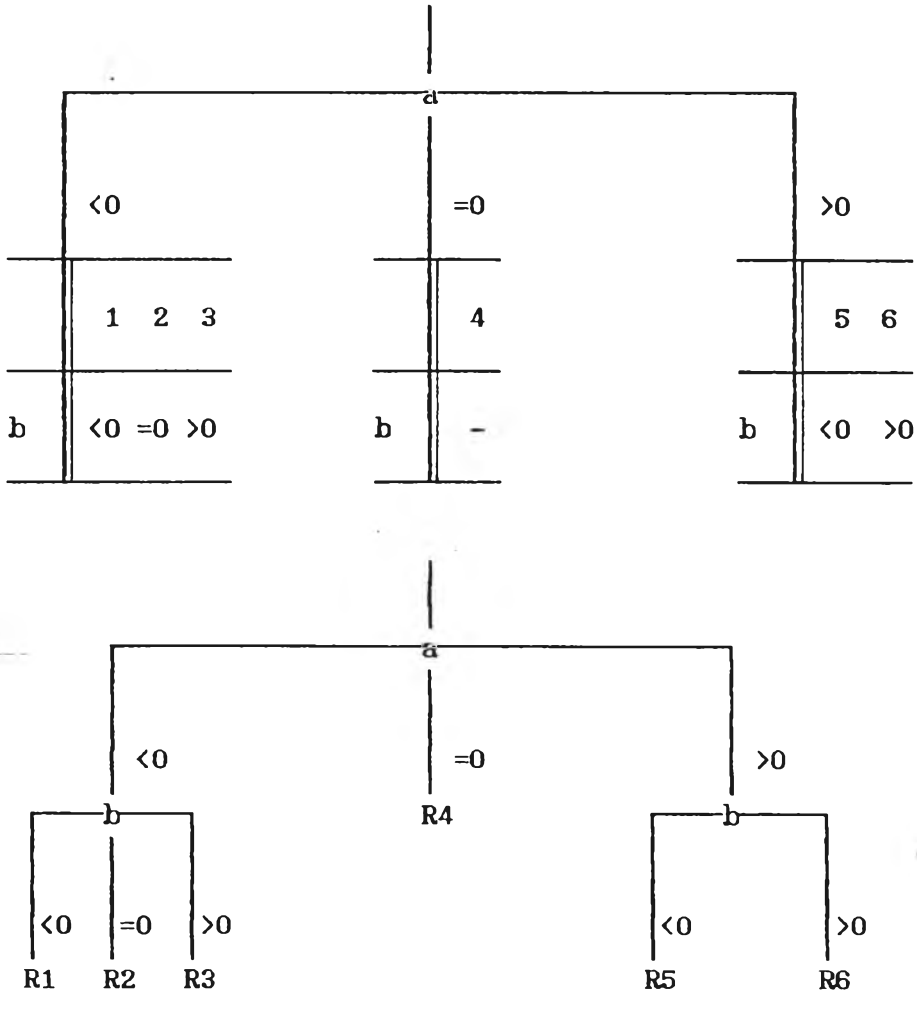
```
                else
                        { action set ELSE }
        else
             { action set R2}
else
        if C2 then
                 { action set R3 }
        else
                if C3 then
                     { action set ELSE }
                else
                      { action set R4}
```

รูปที่ 2.10 แสดงการแปลงตารางตัดสินใจเป็นต้นไม้ตัดสินใจและเปลี่ยนเป็นส่วนของโปรแกรมภาษาปาสคาล

สำหรับวิธีการไบเฟอร์เคชั่นนี้สามารถปรับปรุงเพื่อใช้กับตารางแบบการรับเข้าขยาย โดยที่ไม่ได้แบ่งตารางเป็น 2 ส่วนแต่แบ่งออกเป็นหลายๆ ส่วนขึ้นอยู่กับจำนวนของจุดรับเข้าแบบขยายที่แตกต่างกันที่ถูกกำหนดให้กับเงื่อนไขตัวอย่างของการแปลงดังตารางง่ายๆ ในตารางที่ 2.16 ตารางมีเงื่อนไข 2 แถว ถ้าเงื่อนไขแรกถูกทดสอบจะทำให้เกิดทาง 3 สาขา ในต้นไม้ตัดสินใจ ตัวอย่างของการสร้างต้นไม้ตัดสินใจดังในรูปที่ 2.11

	R1	R2	R3	R4	R5	R6
a	<0	<0	<0	=0	>0	>0
b	<0	=0	>0	-	<0	>0

ตารางที่ 2.16 แสดงตารางตัดสินใจแบบการรับเข้าขยายใช้สร้างต้นไม้ตัดสินใจ



รูปที่ 2.11 แสดงการสร้างต้นไม้ตัดสินใจจากตารางแบบการรับเข้าขยาย

จากวิธีการที่กล่าวมาข้างต้นเป็นวิธีการพื้นฐานที่ใช้สร้างต้นไม้ตัดสินใจจากตารางตัดสินใจ เราจะมาพิจารณาถึงการหาต้นไม้ตัดสินใจที่เหมาะสมจากตารางตัดสินใจ ซึ่งจะพิจารณาว่าจะให้อะไรที่เหมาะสม (Optimized) ซึ่งโดยทั่วไปแล้วจะแบ่งเป็น 2 ขั้นตอนวิธี คือ

1. ความพยายามที่จะลดเวลากระทำการให้น้อยที่สุด (Minimize the expected excution time) ของโปรแกรมที่สร้างจากตารางตัดสินใจ

2. การลดขนาดของหน่วยเก็บที่ต้องใช้สำหรับโปรแกรมที่ถูกสร้างใช้ (Minimize the storage space)

จากขั้นตอนวิธีพื้นฐานที่กล่าวมานี้มีผู้คิดค้นวิธีการแปลงเป็นต้นไม้ตัดสินใจที่เหมาะสมหลายคน ซึ่งจะกล่าวถึงในวิทยานิพนธ์ดังต่อไปนี้

1.1 ขั้นตอนวิธีของ Pollack's

Pollack's ได้เสนอขั้นตอนวิธีไว้ 2 ขั้นตอนวิธี ซึ่งเป็นการแปลงตารางตัดสินใจแบบการรับเข้าจำกัดเป็นต้นไม้ตัดสินใจ ขั้นตอนวิธีมีดังนี้

1.1.1 การลดขนาดของหน่วยเก็บ (Storage Space Minimization) เป็นการลดจำนวนของการทดสอบในตารางตัดสินใจซึ่งจะทำให้ลดเนื้อที่ที่ต้องการใช้ในการสร้างโปรแกรม วิธีการในการสร้างต้นไม้ตัดสินใจมีวิธีเดียวกันกับการสร้างด้วยวิธีไบเฟอร์เคชั่น คือการสร้างตารางย่อยโดยแยกออกเป็น 2 ทางซึ่งมีข้อเพิ่มเติม คือการเพิ่มข้อกำหนดในการเลือกเงื่อนไขที่ใช้ทดสอบในการสร้างตารางย่อยแต่ละครั้ง วิธีการมีดังนี้

ก) จากตารางคำนวณค่า Column Count(CC) ของแต่ละกฎในตารางตัดสินใจ ถ้ากำหนดให้ r เป็นจำนวนของขีด(-) ในกฎค่าของ Column Count คือ 2^r

ข) คำนวณค่า Dash Count(DC) ให้กับเงื่อนไขทุกแถวในตารางตัดสินใจ ค่า DC นี้ได้จากการรวมค่าของ column count ของกฎทั้งหมดที่มีค่าของเงื่อนไขเป็นขีด(-) ในแถวนั้นๆ

รายละเอียดของขั้นตอนต่างๆ รวมทั้งตัวอย่างประกอบของขั้นตอนวิธีนี้จะกล่าวอย่างละเอียดในบทถัดไป เนื่องจากใช้วิธีการนี้ในการพัฒนาโปรแกรม

1.1.2 การลดเวลากระทำการ (Minimize the expected execution time) ของโปรแกรมที่ถูกสร้างขึ้น เป็นการลดจำนวนของเงื่อนไขที่ทดสอบ ซึ่งขึ้นอยู่กับความถี่ของกฎ (Rule Frequencies) และเป็นการลดเวลาที่คาดว่าจะใช้ทั้งหมดในการกระทำการสำหรับโปรแกรมที่ถูกสร้างขึ้นขั้นตอนวิธีนี้จะใช้ความน่าจะเป็นของกฎ (Rule probabilities) หมายความว่าจะต้องหาค่าความน่าจะเป็นของการปรากฏของกฎแต่ละกฎในรายการเปลี่ยนแปลงที่รับเข้ามา ซึ่งมีการตั้งสมมติฐานว่า ELSE rule จะมีค่าความน่าจะเป็นต่ำสุดซึ่งครอบคลุมเฉพาะสถานการณ์ที่ไม่ค่อยเกิดขึ้น

นอกจากที่ค่าที่กำหนดในขั้นตอนวิธีที่ 1 คือค่า Column Count และ dash count แล้ว ยังมีค่าที่จะต้องคำนวณเพิ่มคือ Weighted dash count (WDC) ขึ้นอยู่กับความน่าจะเป็น P_j ของกฎ R_j ซึ่งค่ารวมของความน่าจะเป็นทั้งหมดเป็น 1 ค่าของ Weighted dash count สำหรับเงื่อนไขที่กำหนดคือ ค่ารวมของผลคูณของความน่าจะเป็นของกฎกับ Column Count สำหรับกฎที่มี Dash ในแถวเงื่อนไข

สำหรับขั้นตอนวิธีนี้วิธีการสร้างตารางย่อยและต้นไม้ตัดสินใจมีวิธีการเหมือนกับวิธีไบเฟอร์เคชั่น แต่มีการกำหนดการเลือกเงื่อนไขที่ใช้ทดสอบโดยมีกฎเกณฑ์ดังนี้

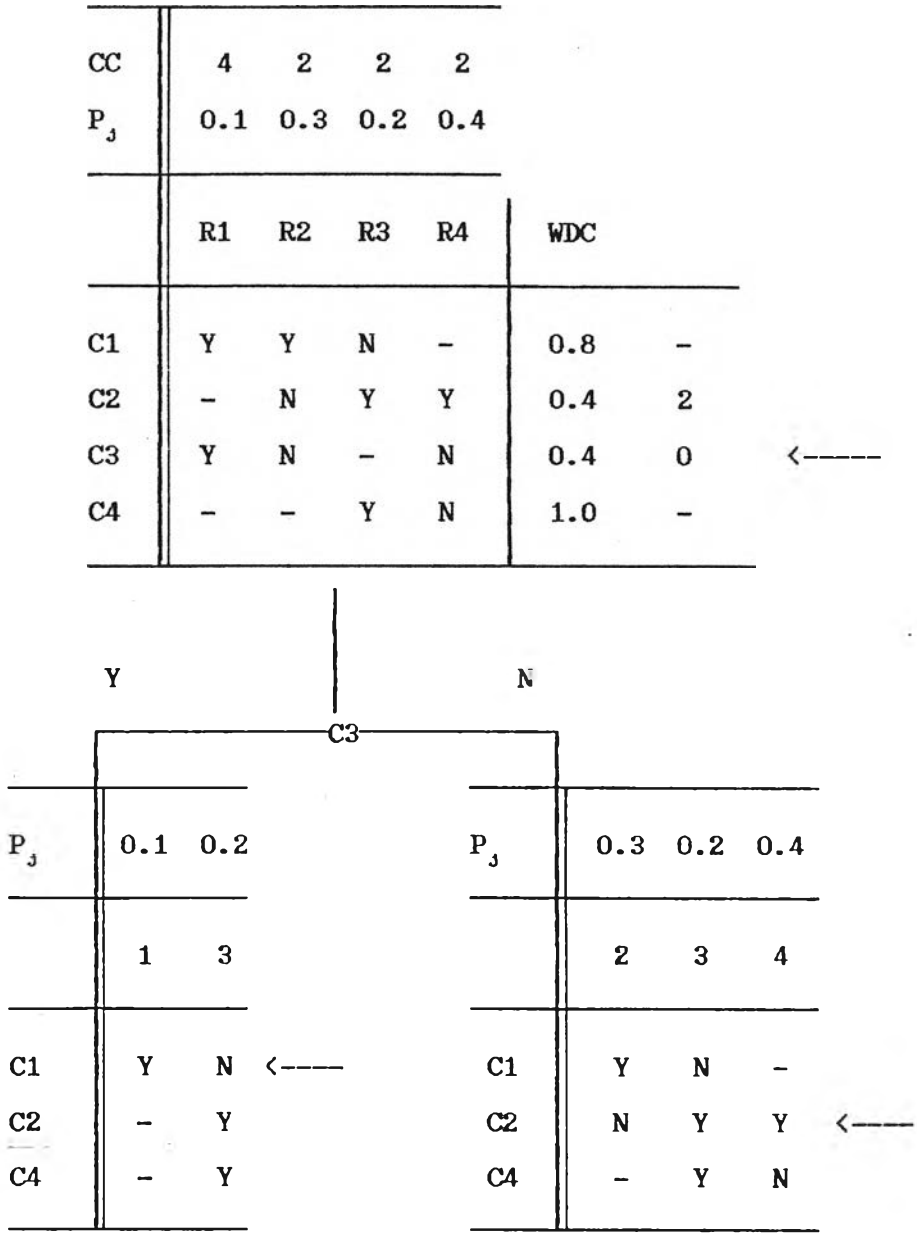
ก) เลือกแถวที่มีค่า Weighted dash count ต่ำสุด

ข) ถ้ามีแถวที่มีค่า WDC ต่ำสุดหลายแถวจะเลือกแถวที่มีค่า ^ ต่ำสุดซึ่งคำนวณโดยเป็นค่าแตกต่างระหว่าง Y count และ N count Y count คำนวณโดยเป็นจำนวนรวมของ Column Count ของกฎที่มีค่า "Y" ในแถวนั้นๆ N count ก็คำนวณเช่นเดียวกันกับ Y count

ค) ถ้ายังคงมีแถวที่ตรงตามกฎเกณฑ์ทั้ง 2 ข้อมากกว่า 1 แถว จะเลือกแถวที่มีค่า dash count ต่ำสุด

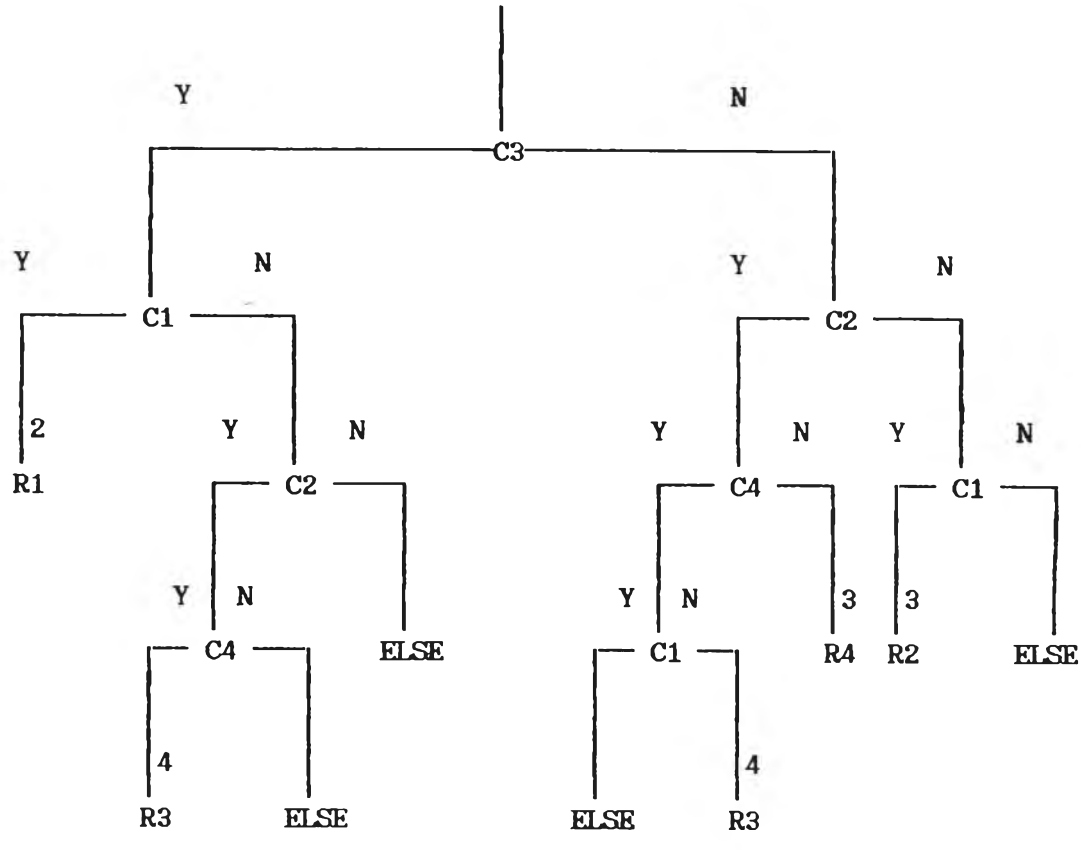
ง) ถ้ายังเลือกไม่ได้ จะเลือกแถวไหนก็ได้โดยไม่มีกฎเกณฑ์ตามแต่จะตัดสินใจ

เมื่อเลือกเงื่อนไขที่ใช้ทดสอบได้แล้วจะแยกออกเป็นตารางย่อยซึ่งตารางย่อยนั้นๆ จะมีค่าความน่าจะเป็นเท่ากับที่กำหนดไว้ในตารางตัดสินใจเริ่มต้นจะทำเช่นนี้เรื่อยไป ตัวอย่างของขั้นตอนวิธีนี้แสดงในรูปที่ 2.12



รูปที่ 2.12 แสดงการสร้างต้นไม้ตัดสินใจโดยใช้วิธีการตามขั้นตอนวิธีลดเวลากระทำการ

ซึ่งเมื่อสร้างตามวิธีการดังกล่าวจะได้ต้นไม้ตัดสินใจดังรูปที่ 2.13



รูปที่ 2.13 แสดงต้นไม้ตัดสินใจสมบูรณ์ที่ได้จากขั้นตอนวิธีของ Pollack's ลดเวลากระทำการ

1.2 ขั้นตอนวิธีของ Verhelst's

ขั้นตอนวิธีของ Verhelst's นี้พิจารณาถึงลำดับขั้นของวิธีการไบเฟอร์เคชั่นสำหรับสร้างต้นไม้ตัดสินใจ โดยตั้งสมมติฐานว่าตารางที่พิจารณาไม่มี ELSE Rule โดยพิจารณาถึงปัญหาของเงื่อนไขที่เป็นแบบ Implied ซึ่งได้พัฒนาการเลือกกฎสำหรับตาราง ถ้าค่าประมาณความน่าจะเป็นของการปรากฏของกฎที่ j คือ p_j โดย j=1..r และเวลาในการทดสอบเงื่อนไขที่ i คือ t_i โดย i=1..c ดังนั้นค่า Lower bound บนจำนวนรวมของการทดสอบของตารางตัดสินใจคือ

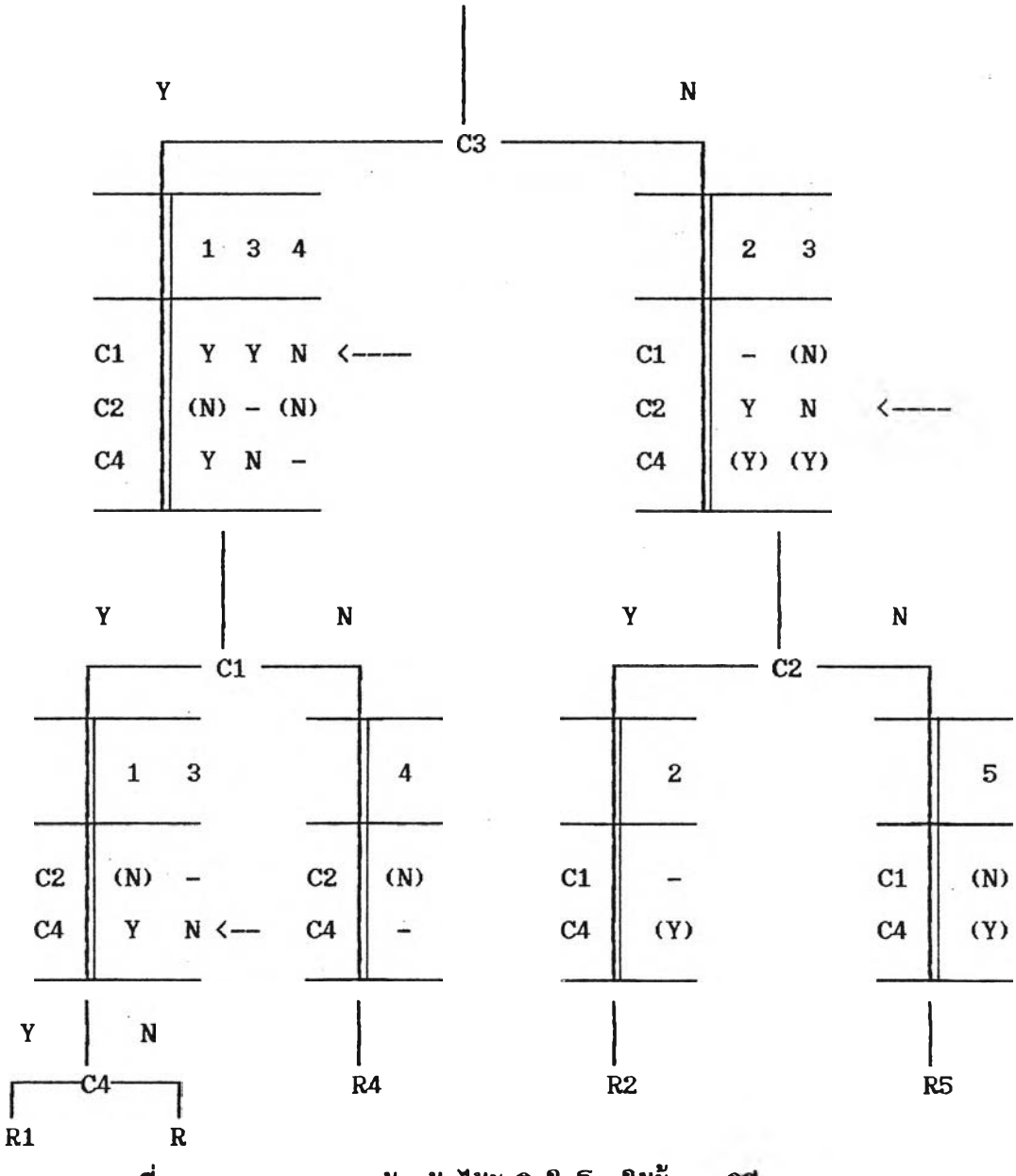
$$S = \sum_{i=1}^{c} t_i \sum_j p_j$$

สำหรับทุกค่าของ j ที่รายการ (i,j) ไม่เป็น dash('-') Verhelst's Optimum Approach Algorithm มีพื้นฐานมาจากขั้นตอนวิธีของ Pollack's โดย Verhelst ได้แนะนำว่าสำหรับแต่ละขั้นของการประมวลผลจะต้องมีการคำนวณค่าต่อไปนี้ทุกๆ แถวของเงื่อนไข คือ

$$T_i = t_i \sum_j p_j$$

สำหรับทุกค่าของ j ที่ รายการ (i,j) เป็น dash จะเลือกเงื่อนไขที่ใช้ทดสอบโดยเลือกจากเงื่อนไขที่ให้ค่า T_i ที่น้อยที่สุด เมื่อเงื่อนไขที่ถูกเลือกมีกฎที่มีค่าเป็น dash 1 หรือมากกว่า 1 กฎ ค่าความน่าจะเป็นของสดมภ์ที่เกี่ยวข้องจะถูกแยกออกเท่าๆ กันระหว่างตารางย่อย Y และ N ตัวอย่างการใช้ขั้นตอนวิธีนี้อยู่ในรูปที่ 2.14

P_j	0.15	0.2	0.1	0.25	0.3		
	R1	R2	R3	R4	R5	t_i	T_i
C1	Y	-	Y	N	(N)	10	5.0
C2	(N)	Y	-	(N)	N	5	2.5
C3	Y	N	(Y)	Y	N	15	1.5
C4	Y	(Y)	N	-	(Y)	10	7.5



รูปที่ 2.14 แสดงการสร้างต้นไม้ตัดสินใจโดยใช้ขั้นตอนวิธีของ Verhelst's

1.3 ขั้นตอนวิธีของ Shwayder's

เป็นการแปลงตารางตัดสินใจเป็นต้นไม้ตัดสินใจโดยใช้แนวความคิดจากทฤษฎีสารสนเทศ (Information theory) Shwayder ได้ดัดแปลงจากขั้นตอนวิธีของ Pollack's โดยตั้งสมมติฐานว่าการเลือกเงื่อนไขที่ใช้ทดสอบควรเลือกจากเงื่อนไขที่ให้ค่า Entropy สูงสุด ค่า Entropy คือการวัดสารสนเทศ (Information) ที่ได้จากการทดสอบเงื่อนไขที่ทุกๆ ขั้นของการแยกตารางตัดสินใจ การคำนวณค่า Entropy ดังนี้

$$\text{Entropy} = -P(p_Y \text{ Log}_2 p_Y + p_N \text{ Log}_2 p_N)$$

P คือค่าความน่าจะเป็นของกฎที่จะไม่พบ dash("-")
= 1 - จำนวนรวมของความเป็นไปได้ของกฎที่มี dash ในเงื่อนไข

p_Y คือค่าความน่าจะเป็นของเงื่อนไขที่พบ "Y" ถ้าไม่เป็น dash
= (จำนวนรวมของความน่าจะเป็นของกฎที่มี "Y" ปรากฏในกฎนั้นๆ)/P

p_N คือค่าความน่าจะเป็นของเงื่นไขที่พบ "N" ถ้าไม่เป็น dash
= $1 - p_Y$

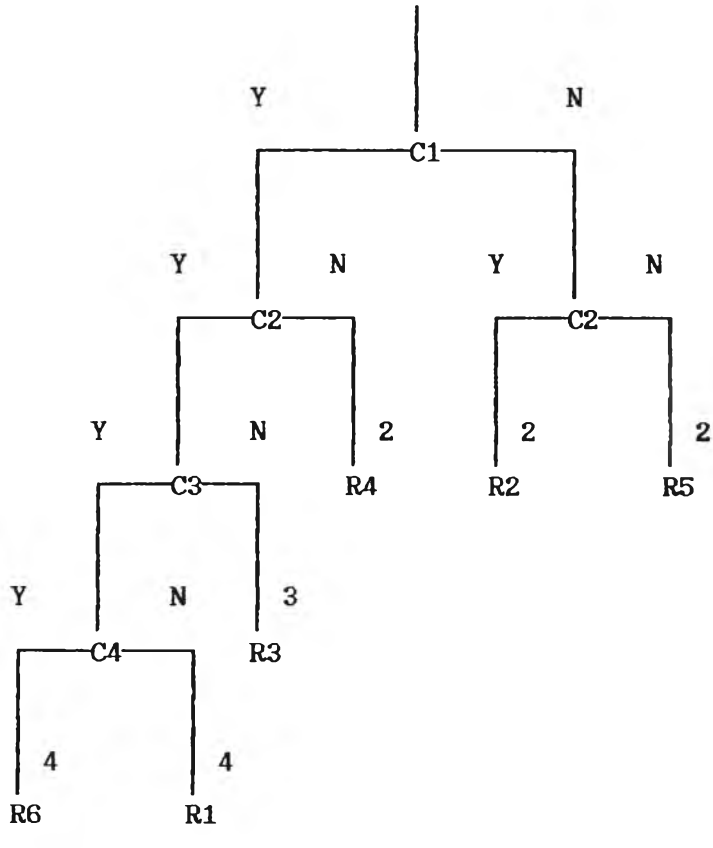
ตัวอย่างของการใช้ขั้นตอนวิธีของ Shwayder's กับตารางตัดสินใจดังในตารางที่ 2.17 รูปที่ 2.15 เป็นการกำหนดค่าความน่าจะเป็นและการคำนวณค่า Entropy ซึ่งนำมาสร้างเป็นต้นไม้ตัดสินใจได้สมบูรณ์

	R1	R2	R3	R4	R5	R6
C1	Y	N	Y	Y	N	Y
C2	Y	Y	Y	N	N	Y
C3	Y	-	N	-	-	Y
C4	N	-	-	-	-	Y

ตารางที่ 2.17 แสดงตารางที่ใช้เป็นตัวอย่างในการสร้างต้นไม้ตัดสินใจ

	P	P_Y	P_N	Entropy
C1	1	0.5	0.5	1
C2	1	0.4375	0.5625	0.99
C3	0.25	0.5	0.5	0.25
C4	0.125	0.5	0.5	0.125

รูปที่ 2.15 แสดงค่าความน่าจะเป็นของเงื่อนไขแต่ละเงื่อนไข และการคำนวณค่า Entropy



รูปที่ 2.16 แสดงต้นไม้ตัดสินใจที่สร้างโดยใช้ขั้นตอนวิธีของ Shwayder

2. วิธีรูมาสค์

เป็นวิธีการแบบ Interpreting decision tables เป็นการเปลี่ยนรูปของ Condition entries อยู่ในรูปเลขฐานสอง (binary) ตามทฤษฎีของ Kirk ซึ่งจะทำให้เข้าใจได้ง่ายขึ้นจึงอธิบายประกอบตัวอย่าง ดังตัวอย่างในตารางที่ 2.18 ซึ่งเป็นตารางที่สมบูรณ์ และไม่กำกวม จากตารางเริ่มต้นจะทำการสร้างแมทริกซ์ (Matrix) ของเลขฐานสอง 2 แมทริกซ์ คือ mask matrix (M) และ decision matrix (D)

Mask Matrix สร้างจากการแปลงค่ารายการเงื่อนไขในตารางตัดสินใจโดยการแทนที่ค่า "Y" และ "N" ในตารางด้วยบิต "1" และค่าที่เหลือจะถูกแทนที่ด้วยบิต "1" และค่าที่เหลือจะถูกแทนที่ด้วย "0" ทุกๆ ตำแหน่งในตารางตัดสินใจเริ่มต้น ดังตัวอย่างในรูปที่ 2.17(ก)

Decision Matrix สร้างโดยการใส่ "1" ในตารางที่ค่าของเงื่อนไขมีเป็น "Y" และค่าที่เหลือจะถูกแทนที่ด้วย "0" ทุกตำแหน่งในตารางตัดสินใจเริ่มต้น ดังตัวอย่างในรูปที่ 2.17(ข)

แมทริกซ์ที่สร้างขึ้นทั้ง 2 แมทริกซ์ เป็นแมทริกซ์ที่มีลักษณะคงที่ (Static) เพราะจะไม่มีการเปลี่ยนแปลงในระหว่างโปรแกรมกระทำการ (Program Execution) และจะถูกสร้างขึ้นในขณะแปลชุดคำสั่ง (Compile) ในขณะที่มีการปฏิบัติการ (Execution Time) เมื่อนำตารางตัดสินใจเข้ามา จะมีการสร้าง Data vector (d) เพื่อแสดงถึงสถานะของเงื่อนไขในขณะนั้นที่กำลังถูกทดสอบ นั่นคือจะให้ 1 แทนเงื่อนไขที่เป็นจริง (true) และ 0 แทนเงื่อนไขที่เป็นเท็จ (false) สมมติว่าในขณะที่ตารางตัดสินใจกำลังถูกกระทำการเงื่อนไขที่ 1 (C1) เป็นจริง เงื่อนไขที่ 2 (C2) เป็นเท็จ และเงื่อนไขที่ 3 (C3) เป็นจริง ตัวอย่างในรูปที่ 2.17(ค)

เวคเตอร์ข้อมูล (d) จะทำการคูณทางตรรกะ (Logically Multiplied หรือ Anded) กับทุกๆ สดมภ์ของแมทริกซ์ M เป็นการตัดสินใจที่ไม่สัมพันธ์กับรหัสในเวกเตอร์ ผลลัพธ์ของแต่ละการกระทำการจะนำมาเปรียบเทียบกับสดมภ์ของ D ที่ตรงกับสดมภ์ของ M ที่ถูกทดสอบ ถ้าผลลัพธ์ที่ได้จากการ AND นั้นเท่ากับค่าในสดมภ์ของ D กฎนั้นจะเป็นที่พอใจ (Satisfied) ตัวอย่างของกระบวนการนี้แสดงในรูปที่ 2.19

	R1	R2	R3	R4
C1	Y	Y	Y	N
C2	Y	N	N	-
C3	-	Y	N	-
A1	X	X	X	
A2			X	
A3				X

ตารางที่ 2.18 แสดงตารางตัดสินใจตัวอย่างประกอบเทคนิครูมาสค์

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad (ก) \qquad D = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (ข) \qquad d = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \quad (ค)$$

รูปที่ 2.17 แสดงเมตริกซ์ที่ได้สร้างตารางที่ 2.18

$$\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \wedge \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \neq \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

d $\quad M_{*1} \quad D_{*1}$ — Rule 1 does not hold - continue

$$\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \wedge \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

d $\quad M_{*2} \quad D_{*2}$ — Rule 2 holds - branch to action routine

รูปที่ 2.19 แสดงการกระทำการทางตรรกะ

โดยทั่วไปกระบวนการของการคูณทางตรรกะและการเปรียบเทียบจะกระทำต่อเนื่องไปจนกว่าทุกสดมภ์ในแมทริกซ์ D จะถูกทดสอบทั้งหมด หมายความว่าสภาวะของกฎที่ปรากฏในตารางตัดสินใจเริ่มต้นได้ถูกทดสอบทั้งหมด จากตารางตัวอย่างที่ 2.19 จะพบการเท่ากันบ่อย และมีการพอใจตลอดเพราะตารางสมบูรณ์และไม่กำกวม อย่างไรก็ดี ถ้าตารางไม่สมบูรณ์จะเกิดการล้มเหลวที่จะพบการตรงกัน (Match) ระหว่างสดมภ์ที่เกิดจากการคูณกันของ d ^ M และ D ซึ่งจะต้องนำกฎ ELSE Rule เข้ามาใช้

ตัวอย่างตารางตัดสินใจที่ใช้เทคนิคของ Kirk's ที่มีกฎ ELSE ดังแสดงในตารางที่ 2.19 และแมทริกซ์ที่เกี่ยวข้องคือ M และ D แสดงในรูปที่ 2.19 สมมติว่ารายการเปลี่ยนแปลงขณะนั้น เงื่อนไขที่ 1 (C1) เป็นจริง เงื่อนไขที่ 2 (C2) เป็นจริง เงื่อนไขที่ 3 (C3) เป็นเท็จ ดังนั้นจะได้เวคเตอร์ข้อมูล d ดังในรูปที่ 2.19 สำหรับรายการเปลี่ยนแปลงนี้ เมื่อกระทำการทางตรรกะกับทุกๆ สดมภ์ของ M ผลลัพธ์ที่ได้ไม่ตรงกับค่าในแมทริกซ์ D ดังในรูปที่ 2.21 ดังนั้นจึงนำ ELSE Rule มาใช้ กลับไปตรวจสอบตารางตัดสินใจเริ่มต้นของตารางที่ 2.19 พบว่ารายการเปลี่ยนแปลง {Y,Y,N} อยู่ในกฎ ELSE Rule

	R1	R2	R3	ELSE
C1	Y	N	N	
C2	-	Y	N	
C3	Y	-	N	
A1	X			
A2		X		
A3			X	
A4				X

ตารางที่ 2.19 แสดงตารางตัดสินใจตัวอย่างประกอบเทคนิครูมาสค์ที่มีกฎ ESLE

$$M = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \quad (ก) \qquad D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad (ข) \qquad d = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad (ค)$$

รูปที่ 2.19 แสดงแมทริกซ์ที่ได้สร้างตารางที่ 2.19

$$\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad d \qquad \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \quad M_{*1} \qquad = \qquad \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \neq \quad \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \quad D_{*1}$$

Rule 1 does not hold - continue

$$d \wedge M_{*1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \neq D_{*1}$$

$$d \wedge M_{*2} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \neq D_{*2}$$

$$d \wedge M_{*3} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \neq D_{*3}$$

รูปที่ 2.21 แสดงการกระทำการทางตรรกะ

สำหรับขั้นตอนวิธีของ Kirk's นี้สามารถสรุปได้ดังนี้

1.1 การสร้างแมทริกซ์ Mask และ Decision จากตารางตัดสินใจที่กำหนดโดย

1.1.1 สร้างแมทริกซ์ M[1..c,1..r] โดย c คือจำนวนของแถวเงื่อนไขในตารางตัดสินใจ และ r คือจำนวนของกฎในตารางตัดสินใจ ซึ่งสามารถแสดงสมาชิกของ M ได้ดังนี้

$$M_{ij} = \begin{cases} 1 & \text{ถ้าค่าของเงื่อนไขที่ (i,j) เป็น 'Y'หรือ 'N'} \\ 0 & \text{ในกรณีอื่นๆ} \end{cases}$$

1.1.2 สร้างแมทริกซ์ D[1..c,1..r] โดยสมาชิกของ D มีดังนี้

$$D_{ij} = \begin{cases} 1 & \text{ถ้าค่าเงื่อนไขที่ (i,j) เป็น 'Y'} \\ 0 & \text{ในกรณีอื่นๆ} \end{cases}$$

1.2 แต่ละรายการเปลี่ยนแปลงจะกำหนดกฎไหนเป็นที่พอใจโดย

1.2.1 หาค่าของเงื่อนไขและสร้างเวกเตอร์ข้อมูล d[1..c] ดังนี้

$$d_i = \begin{cases} 1 & \text{ถ้าเงื่อนไขที่ i เป็นจริง} \\ 0 & \text{ในกรณีอื่นๆ} \end{cases}$$

1.2.2 ทำการคูณทางตรรกะระหว่าง d กับ M แล้วเปรียบเทียบผลลัพธ์กับ D ดังต่อไปนี้

```
for j:=1 to r do
    if (d ∧ M[*,j]) = (D[*,j] then
       branch to rule_j;
branch to else_rule;
```

สังเกตเห็นว่าวิธีการนี้จะต่างจากวิธีการใช้ต้นไม้ตัดสินใจที่ขั้นตอนวิธีของ Kirk's จะไม่ล้มเหลว เมื่อตารางมีความกำกวม ถ้ากฎ 2 กฎหรือมากกว่านั้นเกิดการซ้อนทับกัน ขั้นตอนของ Kirk's นั้นจะสมมติให้กฎที่พบกฎแรกเป็นกฎที่เป็นที่พอใจแต่การหลีกเลี่ยงนี้จะทำให้เกิดข้อเสีย 2 อย่างคือ ประการแรกการที่ไม่พยายามหาขบวนการทดสอบที่ดีที่สุดหรือเหมาะสมที่สุด เงื่อนไขทั้งหมดจะถูกกำหนดค่าก่อนการใช้ขั้นตอนวิธี ประการที่ 2 คือ ขั้นตอนวิธีนี้ใช้ได้เฉพาะกับตารางตัดสินใจแบบการรับเข้าจำกัด