ฉบับปรับปรุง

ออโตมาตาจำกัด
Finite Automata

จารุโลจน์ จงสถิตย์วัฒนา

ออโตมาตาจำกัด

Finite Automata

จารุโลจน์ จงสถิตย์วัฒนา

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

Photographs:

จารุโลจน์ จงสถิตย์วัฒนา

Cover and chapter title page design:

วิมลมาน จงสถิตย์วัฒนา

สารบัญ

คำนำ

แนวคิดทางคณิตศาสตร์เป็นพื้นฐานที่สำคัญสำหรับสาขาวิทยาการคอมพิวเตอร์เพราะใช้ในการตรวจสอบข้อจำกัดของคอมพิวเตอร์และวัดประสิทธิภาพของโปรแกรม นอกจากนั้นยังสามารถนำมาใช้ประโยชน์ในการแก้ปัญหาและเขียนโปรแกรม

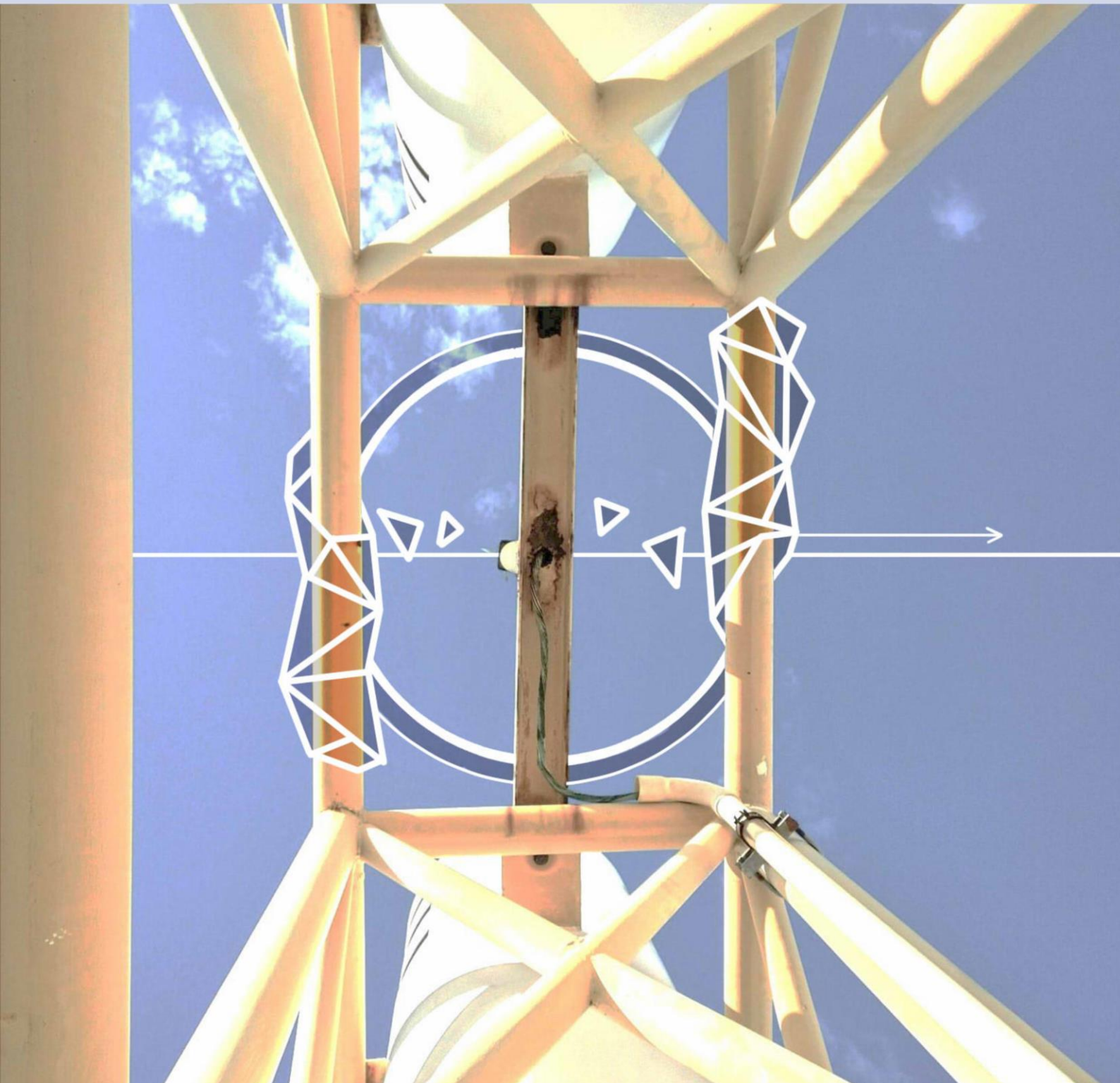
หนังสือเล่มนี้อธิบายออโตมาตาจำกัดซึ่งเป็นตัวแบบทางคณิตศาสตร์ที่เป็นพื้นฐานของสาขาวิทยาการคอมพิวเตอร์ โดยแสดงนิยามและการพิสูจน์ทางคณิตศาสตร์ พร้อมทั้งแสดงการประยุกต์ใช้ตัวแบบนี้ นอกจากนั้นยังแสดงโปรแกรมที่จำลองการทำงานของตัวแบบนี้ การทำงานเชิงไม่กำหนดแนวคิดที่เป็นนามธรรมที่อธิบายยาก แต่เนื่องจากออโตมาตาจำกัดนิยามด้วยโครงสร้างทางคณิตศาสตร์จึงทำให้สามารถใช้นิยามการทำงานเชิงไม่กำหนดได้ชัดเจน นอกจากนั้นยังสามารถนำโครงสร้างเหล่านี้มาใช้ในโปรแกรมจำลองการทำงานได้อย่างเหมาะสม

การศึกษาออโตมาตาจำกัดเป็นพื้นฐานที่จำเป็นสำหรับเข้าใจออโตมาตาชนิดอื่น ๆ นอกจากนั้นความสามารถในการสร้างตัวแบบทางคณิตศาสตร์เช่นนี้ เป็นประโยชน์ในการเขียนโปรแกรมแก้ปัญหาที่ซับซ้อนขึ้นด้วย

จารุโลจน์ จงสถิตย์วัฒนา

บทที่ 1

บทนำ



บทที่ 1 บทนำ

ออโตมาตา (automata) เป็นตัวแบบทางคณิตศาสตร์ (mathematical model) ที่ใช้อธิบายขั้นตอนการทำงานในรูปแบบของเครื่องจักร ออโตมาตาที่ใช้กันอยู่มีหลายแบบ เช่น ออโตมาตาจำกัด (finite automata) ออโตมาตาพูชดาวน์ (pushdown automata) และ เครื่องทูวริง (Turing machine) ออโตมาตาแบบแต่ละแบบสร้างมาเพื่อใช้อธิบายการทำงานของระบบต่างๆ กัน นักวิทยาศาสตร์ นักคณิตศาสตร์และวิศวกรใช้ตัวแบบทางคณิตศาสตร์เป็นเครื่องมือเพื่ออธิบาย วิเคราะห์ ตรวจสอบ และจำลองการทำงานของระบบในโลกจริง ออโตมาตา 3 ประเภทที่กล่าวมานี้เป็นตัวแบบที่เกี่ยวข้องกับการทำงานด้วยคอมพิวเตอร์ ข้อดีข้อหนึ่งของการใช้ตัวแบบทางคณิตศาสตร์ คือ สามารถอธิบายการทำงานของเครื่องได้ชัดเจน ดังนั้นเราสามารถใช้ตัวแบบเพื่อเขียนโปรแกรมจำลองการทำงานของเครื่องเหล่านี้ได้จากนิยามของตัวแบบเหล่านี้ นอกจากนั้นในสาขาวิทยาการคอมพิวเตอร์ยังใช้นำออโตมาตาเหล่านี้มาเชื่อมโยงกับปัญหาต่างๆ เพื่อพิสูจน์สมบัติของปัญหาและศึกษาข้อจำกัดของคอมพิวเตอร์

ในปี ค.ศ. 1936 ซึ่งเป็นยุคบุกเบิกคอมพิวเตอร์ อลัน ทูวริง (Alan Turing) ได้นิยามตัวแบบทางคณิตศาสตร์ของคอมพิวเตอร์ขึ้นมาเพื่อแสดงว่าคอมพิวเตอร์ไม่สามารถแก้ปัญหาบางปัญหาได้ [11] ตัวแบบนี้คือเครื่องทูวริงซึ่งยังใช้ในจำแนกความยากของปัญหามาจนถึงปัจจุบัน

ในปี ค.ศ. 1943 แมคคัลลอคช์ และ พิทท์ (McCulloch and Pitt) นิยามออโตมาตาจำกัดเพื่ออธิบายการทำงานของเครือข่ายระบบประสาทจำลอง (Artificial neural network) นอกจากนั้น วิศวกรยังใช้ออโตมาตาจำกัดในการออกแบบวงจรดิจิตัล (digital circuit)

ในปี ค.ศ. 1959 โนม ชอมสกี (Noam Chomsky) ซึ่งเป็นนักภาษาศาสตร์ ใช้ไวยากรณ์ (grammar) เพื่อนิยามภาษา [8] ต่อมาในปี ค.ศ. 1959 นีเวล ชอว์ และ ไซมอน (Newell, Shaw, and Simon) ใช้โครงสร้างพูชดาวน์ลิสต์ (push-down list) เพื่อรองรับการใช้นิยามเวียนเกิดในการสร้างโปรแกรมที่พิสูจน์ทฤษฎี (theorem prover) และในปี ค.ศ. 1961 โอททิงเกอร์ (Oettinger) นิยามพูชดาวน์ออโตมาตาในลักษณะเดียวกับที่ใช้ในโปรแกรมที่พิสูจน์ทฤษฎี ต่อมาชอมสกีได้แสดงความสัมพันธ์ระหว่างไวยากรณ์กับพูชดาวน์ออโตมาตา [3]

ในปี ค.ศ. 1967 กินส์เบิร์ก ไกรบาคช์ และ แฮร์ริสัน (Ginsburg, Greibach and Harrison) นิยามไวยากรณ์ที่ใช้อธิบายภาษาโปรแกรมและออโตมาตาพูชดาว์น แล้วแสดงว่าออโตมาตาพูชดาว์นสามารถตรวจสอบภาษาที่สร้างจากไวยากรณ์แบบนี้ได้ [6] ไวยากรณ์แบบนี้มีชื่อว่าไวยากรณ์ไม่ขึ้นกับบริบท (context-free grammar) ภาษาโปรแกรมที่ใช้กันในปัจจุบันนี้อธิบายได้ด้วยไวยากรณ์ไม่ขึ้นกับบริบท ดังนั้นออโตมาตาพูชดาว์นสามารถตรวจสอบความผิดพลาดทางไวยากรณ์ (syntax error) ของ

ภาษาเหล่านี้ นั่นคือโปรแกรมตรวจสอบความผิดพลาดทางไวยากรณ์ของที่เราใช้กันในปัจจุบันนี้ก็คือโปรแกรมที่จำลองการทำงานของออโตมาตาพูชดาว์น

หนังสือเล่มนี้จะอธิบายออโตมาตาจำกัด สมบัติของออโตมาตาจำกัด และการประยุกต์ใช้งาน ออโตมาตาจำกัดเป็นตัวแบบของเครื่องที่มีสถานะ (state) ของเครื่องที่เปลี่ยนไปตามสัญลักษณ์ที่รับเข้ามา ออโตมาตาอ่านสตริง (string) ที่เป็นข้อมูลเข้าจนหมดแล้วตอบรับ (accept) หรือไม่รับ (reject) ดังนั้น ออโตมาตาจำกัดแบ่งสตริงเป็น 2 เซต คือ เซตที่ยอมรับและเซตที่ไม่ยอมรับ เซตของสตริงนี้เรียกว่า ภาษา (language) ซึ่งใช้อธิบายว่าออโตมาตายอมรับสตริงใดบ้าง เช่น ออโตมาตาจำกัดเครื่องหนึ่งที่ยอมรับภาษาของจำนวนคู่จะยอมรับเลข 0, 2, 4, 6, … และไม่ยอมรับเลข 1, 3, 5, 7, … หรืออาจกล่าวในแง่ของปัญหาได้ว่าออโตมาตาจำกัดนี้ตอบคำถามว่า สตริงที่รับเข้าไปแทนจำนวนคู่บวกหรือไม่ และเครื่องจะตอบ “ใช่” สำหรับเลข 0, 2, 4, 6, … และ “ไม่ใช่” สำหรับเลข เลข 1, 3, 5, 7, … ดังนั้น บทที่ 2 จะเริ่มนิยามภาษาและตัวดำเนินการ (operator) สำหรับภาษา

ออโตมาตาจำกัดแบ่งได้เป็น 2 ประเภท คือ ออโตมาตาจำกัดเชิงกำหนด (deterministic finite automata) และ ออโตมาตาจำกัดเชิงไม่กำหนด (non-deterministic finite automata) ออโตมาตาสองแบบนี้ต่างกันที่การเปลี่ยนสถานะเมื่อเครื่องรับสัญลักษณ์หนึ่งเข้ามา เมื่อออโตมาตาจำกัดเชิงกำหนดรับสัญลักษณ์ตัวหนึ่งเข้ามา มันจะเปลี่ยนสถานะไปยังสถานะหนึ่งตามแต่เครื่องกำหนดไว้ แต่ออโตมาตาจำกัดเชิงไม่กำหนดอาจมีทางเลือกว่าจะเปลี่ยนไปอยู่ในสถานะใดได้มากกว่าหนึ่งสถานะและเครื่องจะรู้ว่าทางเลือกใดในทางเลือกทั้งหมดที่มีจะทำให้เครื่องทำงานจนจบได้แล้วเลือกเปลี่ยนสถานะไปทางนั้น บทที่ 3 อธิบายออโตมาตาจำกัดเชิงกำหนด สมบัติของภาษาที่ยอมรับด้วยเครื่องชนิดนี้ และการจำลองการทำงานของออโตมาตาจำกัดเชิงกำหนด จากนั้นจะแสดงการประยุกต์ใช้ออโตมาตาจำกัดเชิงกำหนดในงานประเภทต่างๆ

จากที่กล่าวมาแล้ว จะเห็นว่าออโตมาตาจำกัดเชิงไม่กำหนดสามารถตัดสินใจเลือกสถานะเพื่อให้ทำงานจนจบได้เอง ความสามารถตัดสินใจแบบนี้ เรียกว่า การเปลี่ยนสถานะเชิงไม่กำหนด (non-deterministic state transition) ซึ่งเปรียบเสมือนการหยั่งรู้อนาคตว่าจะอ่านได้สัญลักษณ์ใดในอนาคต ในบทที่ 4 หลังจากอธิบายออโตมาตาจำกัดเชิงไม่กำหนดแล้วเราจะแสดงว่าการเปลี่ยนสถานะเชิงไม่กำหนดไม่ทำให้ออโตมาตาจำกัดเชิงไม่กำหนดมีความสามารถ "มาก" กว่าออโตมาตาจำกัดเชิงกำหนด นั่นคือ*ไม่มี*ภาษาใดที่ออโตมาตาจำกัดเชิงไม่กำหนดสามารถยอมรับแต่ออโตมาตาจำกัดเชิงกำหนดไม่สามารถยอมรับได้ การเปลี่ยนสถานะเชิงไม่กำหนดเป็นสิ่งที่อำนวยความสะดวกให้ใช้ตัวแบบได้ง่ายขึ้น จากนั้น เราจะนำการเปลี่ยนสถานะเชิงไม่กำหนดมาช่วยในการพิสูจน์สมบัติของภาษาที่ยอมรับด้วยออโตมาตาจำกัด จากนั้นจะอธิบายการจำลองการทำงานของออโตมาตาเชิงไม่กำหนดด้วยอัลกอริทึมที่ทำงานแบบกำหนด สุดท้ายเราจะแสดงการประยุกต์ใช้ออโตมาตาจำกัดเชิงไม่กำหนดในงานประเภทต่างๆ

จากแง่มุมของภาษาศาสตร์ เราอธิบายภาษาได้ด้วยไวยากรณ์และใช้ออโตมาตาเพื่อตรวจสอบว่าข้อความใดเป็นข้อความในภาษานั้น โครงสร้างที่ใกล้เคียงกับไวยากรณ์ที่ใช้อธิบายภาษาที่ยอมรับด้วยออโตมาตาจำกัดคือนิพจน์ปกติ (regular expression) ซึ่งอธิบายในบทที่ 5 นอกจากนั้นเราแสดงว่าเราสามารถสร้างออโตมาตาจำกัดให้ตรวจสอบภาษาตามนิพจน์ปกติ ในทางกลับกันเราสามารถสร้างนิพจน์ปกติที่สร้างภาษาที่ออโตมาตาจำกัดยอมรับได้ด้วย นิพจน์ปกติใช้กำหนดรูปแบบของข้อมูล เช่น ชื่อตัวแปร ที่อยู่เมลอิเล็คโทรนิคส์ ที่อยู่เว็บ ดังนั้นเราสามารถสร้างออโตมาตาจำกัดเพื่อตรวจสอบรูปแบบข้อมูลตามที่กำหนดด้วยนิพจน์ปกติ บทที่ 6 อธิบายออโตมาตาพูชดาว์นและเครื่องทูวริงอย่างสั้น ออโตมาตาทั้งสองแบบนี้มีความสามารถมากกว่าออโตมาตาจำกัดคือสามารถตรวจสอบภาษาที่ซับซ้อนมากขึ้นได้

บทที่ 2

ภาษา

บทที่ 2 ภาษา

บทนี้จะเริ่มนิยามความรู้เกี่ยวกับการทำงานกับสตริง (string) และภาษา (language) ซึ่งเป็นเซตของสตริงเพื่อเป็นพื้นฐานสำหรับอธิบายการทำงานของออโตมาตาจำกัด

2.1 สตริงและการดำเนินการบนสตริง

ในหัวข้อนี้จะนิยามสตริงโดยเริ่มจากการกำหนดหน่วยเล็กที่สุดที่ใช้ในการสร้างสตริงก่อน หน่วยเล็กสุดนี้คือสัญลักษณ์ (symbol) ที่สามารถนำมาใช้ในภาษาได้ เช่น ภาษาไทยประกอบด้วยสัญลักษณ์ ก, ข, ฃ, ..., อ, ฮ, ะ, า, ..., โ ภาษาอังกฤษประกอบด้วยสัญลักษณ์ a, b, …, z, A, B, …, Z สัญลักษณ์แต่ละตัวจัดเป็นหน่วยที่ไม่สามารถแยกย่อยไปอีกได้ แล้วจะนิยามอักขระต่อจากสัญลักษณ์ได้ดังนี้

นิยาม 2.1 อักขระ (alphabet) คือเซตจำกัดของสัญลักษณ์

ตัวอย่างต่อไปนี้แสดงอักขระที่ใช้ทั่วไป

ตัวอย่าง 2.1 ตัวอย่างของอักขระ ได้แก่

- {a, b, …, z, A, B, …, Z} เป็นอักขระภาษาอังกฤษซึ่งมีสมาชิก 52 ตัว
- {0, 1, …, 9} เป็นอักขระเลขฐานสิบซึ่งมีสมาชิก 10 ตัว
- เซต {0, 1} เป็นอักขระเลขฐานสองซึ่งมีสมาชิก 2 ตัว

เรานิยมใช้สัญลักษณ์ Σ แทนอักขระ และเมื่อนำสัญลักษณ์ในอักขระมาเรียงต่อกันเรียกว่าสตริง ตามนิยามต่อไปนี้

นิยาม 2.2 สตริง (string) บนอักขระ Σ คือ สัญลักษณ์ในอักขระ Σ ที่นำมาวางเรียงต่อกัน
สตริงว่าง (empty string) ซึ่งแทนด้วยสัญลักษณ์ ε คือ สตริงที่ไม่มีสัญลักษณ์เลย

ถ้ากำหนดอักขระ $\Sigma = \{0, 1\}$ แล้วสามารถสร้างสตริง ε, 0, 1, 01110, และ 111111 บนอักขระ Σ ได้ เซตของสตริงที่เกิดจากการต่อกันของสัญลักษณ์ใดๆ ใน Σ กี่ตัวก็ได้ เรียกว่า Σ^* เช่น ถ้า $\Sigma = \{0, 1\}$ แล้ว $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, \dots\}$ สังเกตว่า ε อยู่ใน Σ^* เสมอ เนื่องจาก

ε เกิดจากการเลือกสัญลักษณ์ 0 ตัว นอกจากนั้น Σ^* เป็นเซตอนันต์เสมอเพราะสตริงใน Σ^* เกิดจากสัญลักษณ์ต่อกันกี่ตัวก็ได้ เซตของสตริงที่เกิดจากการต่อกันของสัญลักษณ์ใดๆ ใน Σ อย่างน้อยหนึ่งตัว เรียกว่า Σ^+ เช่น ถ้า $\Sigma = \{0, 1\}$ แล้ว $\Sigma^+ = \{0, 1, 00, 01, 10, 11, 000, 001, 010, \dots\}$ ความแตกต่างระหว่าง Σ^* และ Σ^+ คือ Σ^* รวมสตริงที่เกิดจากการต่อกันของสัญลักษณ์**ศูนย์**ตัวคือสตริงว่างได้ แต่สตริงใน Σ^+ ต้องเกิดจากสัญลักษณ์อย่างน้อย**หนึ่ง**ตัว ดังนั้น $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$

นิยาม 2.3 ความยาวของสตริง คือ จำนวนตำแหน่งของสัญลักษณ์ในสตริง เราใช้ $length(x)$ แทนความยาวของสตริง x และ $x(i)$ แทนสัญลักษณ์ตัวที่ i ของสตริง x เมื่อ $1 \le i \le length(x)$

จากนิยาม 2.3 จะเห็นว่า *length* เป็นฟังก์ชันจาก Σ^* ไปยังเซตของจำนวนเต็มที่ไม่เป็นลบ และ $length(\varepsilon) = 0$ ตัวอย่างต่อไปนี้แสดงความยาวของสตริงและตำแหน่งของสัญลักษณ์ในสตริง

ตัวอย่าง 2.2 กำหนดสัญลักษณ์ a, b, c,…,y, z และอักขระ Σ = {a, b, c,…,y, z}
ถ้าให้ α เป็นสตริง automata บนอักขระ Σ และ β เป็นสตริง computation บนอักขระ Σ
$length(\alpha) = 8$, $length(\beta) = 11$, $\alpha(1) = a$, และ $\beta(5) = u$

ต่อไปจะนิยามการดำเนินการ (operation) สองตัวบนสตริงคือ การต่อกัน (concatenation) และการผันกลับ (reverse)

นิยาม 2.4 กำหนด x และ y เป็นสตริงบนอักขระ Σ การต่อกัน (concatenation) ของ x และ y เขียนแทนด้วย $x \cdot y$ หรือ $x\,y$ คือ สตริง z ซึ่ง

1. $z(i) = x(i)$ เมื่อ $1 \le i \le length(x)$
2. $z(i) = y(i - length(x))$ เมื่อ $length(x) < i \le length(x) + length(y)$

จากนิยาม 2.4 จะเห็นว่าการต่อกันของสตริงมี ε เป็นเอกลักษณ์ทางซ้ายและเอกลักษณ์ทางขวา นั่นคือ สำหรับสตริง x ใดๆ $x \cdot \varepsilon = \varepsilon \cdot x = x$ ตัวอย่างต่อไปนี้แสดงการต่อกันของสตริง

ตัวอย่าง 2.3 กำหนดอักขระ Σ = {a, b, c,…,y, z} ถ้าให้ α เป็นสตริง automata บนอักขระ Σ และ β เป็นสตริง computation บนอักขระ Σ การต่อกันของสตริง α และ β ซึ่งเขียนแทนด้วย $\alpha \cdot \beta$ คือ สตริง automatacomputation

นอกจากนั้น เราเขียนแทนการต่อกัน n ครั้งของสตริง x ใดๆ ด้วย x^n เมื่อ $n \ge 0$ เช่น $x^0 = \varepsilon$, $x^1 = x$, $x^2 = x \cdot x$, $x^3 = x \cdot x \cdot x$, $x^4 = x \cdot x \cdot x \cdot x$ เป็นต้น และ เราสามารถนิยามสตริงภายในได้โดยใช้การต่อกันดังนี้

นิยาม 2.5 กำหนด x และ y เป็นสตริงบนอักขระ Σ y เป็นสตริงย่อย (substring) ของ x ถ้ามีสตริง w และ z ใน Σ^* ที่ทำให้ $x = w \cdot y \cdot z$

จากนิยาม 2.5 จะเห็นได้ว่า ε เป็นสตริงย่อยในทุกสตริงและสตริงใดๆ เป็นสตริงย่อยในตัวมันเอง ต่อไปจะนิยามส่วนผันกลับของสตริง

นิยาม 2.6 กำหนดให้ x เป็นสตริงบนอักขระ Σ ส่วนผันกลับ (reversal) ของ x เขียนแทนด้วย x^r และมีสมบัติดังนี้

1. ถ้า x เป็น ε แล้ว x^r เป็น ε
2. ถ้า a อยู่ใน Σ, x และ y อยู่ใน Σ^*, และ $x = a \cdot y$ แล้ว $x^r = y^r \cdot a$

นิยามของส่วนผันกลับของสตริงนี้เป็นนิยามแบบอุปนัยซึ่งมักจะถูกนำไปใช้ในการพิสูจน์แบบอุปนัย ตัวอย่างต่อไปนี้แสดงส่วนผันกลับของสตริง

ตัวอย่าง 2.4 กำหนดอักขระ Σ = {a, b, c,…,y, z} ส่วนผันกลับของสตริง automata (ซึ่งเขียนแทนด้วย $(\text{automata})^r$) คือ atamotua ส่วนผันกลับนี้หาตามนิยาม 2.6 ได้ดังนี้

$(\text{automata})^r$ = $(\text{utomata})^r$ a = $(\text{tomata})^r$ ua = $(\text{omata})^r$ tua = $(\text{mata})^r$ otua

= $(\text{ata})^r$ motua = $(\text{ta})^r$ amotua = $(\text{a})^r$ tamotua = $(\varepsilon)^r$ atamotua

= atamotua

นอกจากนั้น สังเกตได้ว่าสำหรับสตริง α และ β ใด ๆ $(\alpha \cdot \beta)^r = \alpha^r \cdot \beta^r$ และ $(\alpha^r)^r = \alpha$

2.2 ภาษาและการดำเนินการบนภาษา

เราจะเรียกเซตของสตริงว่า ภาษา ตามนิยามต่อไปนี้

นิยาม 2.7 กำหนด Σ เป็นอักขระ ภาษา (language) บนอักขระ Σ คือ เซตของสตริงบนอักขระ Σ

ตัวอย่างต่อไปนี้แสดงภาษาบนอักขระ {0,1}

ตัวอย่าง 2.5 กำหนดอักขระ Σ = {0, 1} และภาษา L_e บน Σ = {$\omega \in \Sigma^*$ | จำนวนของ 1 ในสตริงเป็นจำนวนคู่}

นั่นคือ L_e = {ε, 0, 00, 11, 000, 011, 101, 110, 0000, 1100, 0110, 0011, …} หรือ ε, 0, 00, 11, 000, 011, 101, 110, 0000, 1100, 0110, 0011, … เป็นสมาชิกของ L_e

เนื่องจากภาษาเป็นเซต การดำเนินการของเซตจึงใช้ได้กับภาษาด้วย เช่น ส่วนรวม (union) และ ส่วนร่วม (intersection) สำหรับส่วนเติมเต็ม (complement) จะใช้ความหมายเดียวกับส่วนเติมเต็มของเซตโดยกำหนดเอกภพ (universe) สำหรับภาษาบน Σ เป็น Σ^* ดังนี้

นิยาม 2.8 กำหนด Σ เป็นอักขระ และ L เป็นภาษาบนอักขระ Σ ส่วนเติมเต็ม (complement) ของ L (เขียนแทนด้วย $\overline{L}$) คือ $\Sigma^* - L$

ตัวอย่างต่อไปนี้แสดงส่วนเติมเต็มของ L_e ใน {0, 1}*

ตัวอย่าง 2.6 กำหนดอักขระ Σ = {0, 1} และภาษา L_e บนอักขระ Σ = {$\omega \in \Sigma^*$| จำนวนของ 1 ในสตริงเป็นจำนวนคู่} จะได้ว่า $\overline{L_e}$ = {$\omega \in \Sigma^*$ | จำนวนของ 1 ในสตริงเป็นจำนวนคี่}

ต่อไปจะนิยามการดำเนินการอื่นบนภาษา ได้แก่ การต่อกันและการผันกลับ ซึ่งใช้การดำเนินการดังกล่าวบนสตริงจากนิยาม 2.4 และนิยาม 2.6 ดังนี้

นิยาม 2.9 กำหนดให้ Σ เป็นอักขระ L_1 และ L_2 เป็นภาษาบนอักขระ Σ การต่อกันของ L_1 และ L_2 (เขียนแทนด้วย $L_1 \cdot L_2$ หรือ $L_1 L_2$) คือ {ω |$\omega = \omega_1 \cdot \omega_2$ สำหรับ ω_1 และ ω_2 ใดๆ โดยที่ $\omega_1 \in L_1$ และ $\omega_2 \in L_2$}

จากนิยามนี้จะเห็นว่า {ε} เป็นเอกลักษณ์ของการต่อกัน นั่นคือ {ε} $\cdot L = L \cdot$ {ε} = L แต่ { } $\cdot L$ = $L \cdot$ { } = { } ตัวอย่างสองตัวอย่างต่อไปนี้แสดงการต่อกันของภาษา ตัวอย่าง 2.7 แสดงการต่อกันของภาษาที่เป็นเซตจำกัดหรือเรียกว่าภาษาจำกัด

ตัวอย่าง 2.7 กำหนดอักขระ Σ = {0, 1}, L_1 = {0, 00, 000, 0000}, และ L_2 = {ε, 1, 11, 111, 1111} จะได้ $L_1 \cdot L_2$ = {0, 00, 000, 0000, 01, 001, 0001, 00001, 011, 0011, 00011, 000011, 0111, 00111, 000111, 0000111, 01111, 001111, 0001111, 00001111}

ตัวอย่าง 2.8 แสดงการต่อกันของภาษาที่เป็นเซตอนันต์

ตัวอย่าง 2.8 กำหนดอักขระ Σ = {0, 1}, L_e = {$\omega \in \Sigma^*$ | จำนวนของ 1 ในสตริงเป็นจำนวนคู่}, และ $\overline{L_e}$ = {$\omega \in \Sigma^*$ | จำนวนของ 1 ในสตริงเป็นจำนวนคี่}
จะได้ว่า $L_e \cdot \overline{L_e}$ = {$\omega \in \Sigma^*$ | จำนวนของ 1 ในสตริงเป็นจำนวนคี่} = $\overline{L_e}$

ต่อไปนี้เราจะนิยามส่วนผันกลับของภาษาซึ่งเป็นเซตของส่วนผันกลับของสตริงในภาษานั้น

นิยาม 2.10 กำหนด Σ เป็นอักขระและ L เป็นภาษาบนอักขระ Σ ส่วนผันกลับของ L (เขียนแทนด้วย L^r) คือ $\{\omega^r \mid \omega \in L\}$

อาจกล่าวอีกวิธีหนึ่งว่า $L^r = \{\omega \mid \omega^r \in L\}$ ตัวอย่างต่อไปนี้แสดงส่วนผันกลับของภาษาจำกัด

ตัวอย่าง 2.9 กำหนดอักขระ Σ = {0, 1} และ L = {0, 00, 000, 0000, 01, 001, 0001, 00001, 011, 0011, 00011, 000011, 0111, 00111, 000111, 0000111, 01111, 001111, 0001111, 00001111} จะได้ L^r = {0, 00, 000, 0000, 10, 100, 1000, 10000, 110, 1100, 11000, 110000, 1110, 11100, 111000, 1110000, 11110, 111100, 1111000, 11110000}

สำหรับ L_{01} = {$\omega \in \Sigma^*$ | 01 เป็นสตริงย่อยใน ω} $L_{01}{}^r$ = {$\omega \in \Sigma^*$ | 10 เป็นสตริงย่อยใน ω}

ต่อไปจะนิยามส่วนปิดคลุมของคลีน (Kleene's closure) สำหรับภาษา ซึ่งเป็นเซตของสตริงที่สร้างโดยนำสตริงในภาษานั้นศูนย์ตัวหรือมากกว่านั้นมาต่อกัน บางทีเราเรียกตัวดำเนินการนี้ว่า การซ้ำของคลีน (Kleene's star)

นิยาม 2.11 กำหนด Σ เป็นอักขระและ L เป็นภาษาบนอักขระ Σ ส่วนปิดคลุมของ L (เขียนแทนด้วย L^*) คือ $\{\omega \mid \omega = \omega_1 \omega_2 \ldots \omega_k$ สำหรับ $\omega_1, \omega_2, \ldots, \omega_k \in L$ เมื่อ $k \geq 0\}$

จากนิยามนี้ L^* เกิดจากการนำสตริงใน L มาต่อกัน สังเกตว่าสตริงที่นำมาต่อกันนี้ไม่จำเป็นต้องเป็นสตริงเดียวกันได้ ตัวอย่างต่อไปแสดงส่วนปิดคลุมของภาษา

ตัวอย่าง 2.10 กำหนดอักขระ Σ = {0, 1}, L_1 = {$\omega \mid \omega = 0^i$ เมื่อ i เป็นจำนวนคู่} = {ε, 00, 0000, 000000, …} และ L_2 = {$\omega \mid \omega = 0^i$ เมื่อ i เป็นจำนวนคี่} = {0, 000, 00000, 0000000 …}

$L_1{}^*$ = L_1 เนื่องจากสตริงใน L_1 มีสัญลักษณ์ 0 เป็นจำนวนคู่ ดังนั้นเมื่อนำสตริงเหล่านี้มาต่อกันแล้วจะได้สตริงที่ยังมีสัญลักษณ์ 0 เป็นจำนวนคู่

$L_2{}^*$ = {0}* = {$\omega \mid \omega = 0^i$ เมื่อ i เป็นจำนวนเต็มที่ไม่เป็นลบ} เนื่องจากสตริงใน L_2 มีสัญลักษณ์ 0 เป็นจำนวนคี่ ดังนั้นเมื่อนำสตริงเหล่านี้มาต่อกันแล้วจะอาจได้สตริงที่ยังมีสัญลักษณ์ 0 เป็นจำนวนคู่หรือคี่ก็ได้

บางครั้งเราสนใจเซตของสตริงที่สร้างโดยนำสตริงในภาษานั้นอย่างน้อยหนึ่งสตริงมาต่อกันซึ่งเรียกว่าส่วนปิดคลุม (closure) และนิยามได้ดังนี้

นิยาม 2.12 กำหนดให้ Σ เป็นอักขระและ L เป็นภาษาบนอักขระ Σ

$L^+ = L \cdot (L^*)$ = {$\omega \mid \omega = \omega_1 \omega_2 \ldots \omega_k$ สำหรับ $\omega_1, \omega_2, \ldots, \omega_k \in L$ เมื่อ $k \geq 1$}

ตัวอย่าง 2.11 แสดงตัวอย่างของ L^+ ของภาษาที่มี ε และภาษาที่ไม่มี ε สังเกตว่า ถ้า $\varepsilon \in L$ แล้ว $\varepsilon \in L^+$ แต่ ถ้า $\varepsilon \notin L$ แล้ว $\varepsilon \notin L^+$ ด้วย

ตัวอย่าง 2.11 จาก L_1 และ L_2 ในตัวอย่าง 2.10 จะได้ $L_1^+ = L_1 \cdot L_1^* = L_1$ และ $L_2^+ = L_2 \cdot L_2^* = \{0\}^+$

หัวข้อต่อไปแสดงความสัมพันธ์ระหว่างภาษาและปัญหาซึ่งเป็นพื้นฐานของวิทยาการคอมพิวเตอร์

2.3 ปัญหา

ปัญหา (problem) ที่สนใจในวิทยาการคอมพิวเตอร์หมายถึงคำถามที่มีคำตอบที่แน่นอน เช่น "จำนวนเฉพาะที่มากกว่า 10 คือจำนวนใดบ้าง ?" , "สำหรับกราฟ G ที่กำหนดให้ วิถี (path) ที่สั้นที่สุดระหว่างโหนด (node) A และ B มีความยาวเท่าใด ?" เป็นต้น แต่รูปแบบของคำตอบสำหรับคำถามในลักษณะนี้มีได้หลากหลายและทำให้การนิยามตัวแบบของการคำนวณยุ่งยาก ดังนั้นหนังสือเล่มนี้กล่าวถึงปัญหาที่มีคำตอบเป็น YES หรือ NO เท่านั้น เราเรียกปัญหาแบบนี้ว่าปัญหาเชิงตัดสินใจ (decision problem) [5] เราสามารถอธิบายปัญหาใดๆ ให้อยู่ในรูปของปัญหาเชิงตัดสินใจได้และปัญหาเชิงตัดสินใจเป็นปัญหาที่ไม่ยากกว่าปัญหาเดิม ตัวอย่างการแทนปัญหาด้วยปัญหาเชิงตัดสินใจ ได้แก่

- "จำนวนเฉพาะที่มากกว่า 10 คือจำนวนใดบ้าง ?" แทนด้วย "กำหนด n เป็นจำนวนนับ n เป็นจำนวนเฉพาะที่มากกว่า 10 หรือไม่ ?"
- "สำหรับกราฟ G ที่กำหนดให้ วิถีที่สั้นที่สุดระหว่างโหนด A และ B มีความยาวเท่าใด ?" แทนด้วย "สำหรับกราฟ G และจำนวนเต็มบวก k ที่กำหนดให้ มีวิถีระหว่างจุดต่อ A และ B ที่สั้นกว่า k หรือไม่ ?"

ปัญหาเชิงตัดสินใจสามารถแทนได้ด้วยภาษา โดยให้ภาษาเป็นเซตของสตริงที่แทนข้อมูลเข้าที่ให้คำตอบของปัญหานั้นเป็น YES เช่น ปัญหา "กำหนดจำนวนนับ n แล้ว n เป็นจำนวนเฉพาะที่มากกว่า 10 หรือไม่?" แทนได้ด้วย ภาษา $\{n \mid n$ เป็นจำนวนเฉพาะที่มากกว่า 10$\}$ = {13, 17, 19, 23,…} ดังนั้นการแก้ปัญหาหนึ่งสามารถทำได้โดยพิจารณาว่าสตริงที่แทนข้อมูลรับเข้านั้นอยู่ในภาษาที่แทนปัญหานั้นหรือไม่ ถ้าอยู่แล้วจะทำให้ทราบว่าคำตอบของปัญหานั้นเป็น YES ถ้าไม่อยู่แล้วจะทำให้ทราบว่าคำตอบของปัญหานั้นเป็น NO ดังนั้นเมื่อออโตมาตาที่ตัดสินว่าสตริงที่กำหนดให้อยู่ในภาษาที่กำหนดหรือไม่ จึงสามารถให้คำตอบสำหรับปัญหาที่แทนด้วยภาษานั้นได้ด้วย

ข้อสังเกต

- สตริงมีความยาวจำกัด แต่ภาษาอาจเป็นเซตจำกัดหรือเซตอนันต์
- สตริงว่างต่างจากเซตว่าง
- เซตของสตริงไม่เป็นเซตว่าง
- สตริงว่าง (ε) เป็นสตริงย่อยในทุกสตริง และ $length(\varepsilon) = 0$

บทที่ 3

ออโตมาตาจำกัดเชิงกำหนด

บทที่ 3 ออโตมาตาจำกัดเชิงกำหนด

ออโตมาตาจำกัดประกอบด้วยส่วนควบคุม, เทปสำหรับรับข้อมูลเข้า, และ หัวเทปดังแสดงในรูป 3.1 เทปของออโตมาตาจำกัดแบ่งเป็นช่องๆ แต่ละช่องมีสัญลักษณ์อยู่ได้หนึ่งตัว เครื่องสามารถอ่านสัญลักษณ์ในช่องหนึ่งของเทปได้เมื่อหัวเทปอยู่ที่ช่องนั้น หัวเทปเลื่อนไปทางขวาได้ทีละช่อง ส่วนควบคุมมีสถานะ (state) ที่เป็นไปได้เป็นจำนวนจำกัดและจะอยู่ในสถานะเดียว ณ เวลาหนึ่ง นั่นคือส่วนควบคุมจะต้องอยู่ในสถานะใดสถานะหนึ่ง ไม่สามารถอยู่ในสองสถานะในเวลาเดียวกัน และ ไม่สามารถไม่อยู่ในสถานะใดเลย ณ เวลาหนึ่ง

รูป 3.1 โครงสร้างของออโตมาตาจำกัด

เมื่อเริ่มต้นทำงาน เทปของออโตมาตาจำกัดเก็บข้อมูลรับเข้าซึ่งเป็นสตริงโดยเริ่มต้นที่ช่องซ้ายสุดแล้วเรียงต่อไปทางขวาเรื่อยๆ ส่วนควบคุมของออโตมาตาจำกัด (หรืออาจพูดว่าออโตมาตาจำกัด) อยู่ใน*สถานะเริ่มต้น* (start state) และหัวเทปอยู่ที่เทปช่องซ้ายสุด จากนั้นเครื่องจะอ่านสัญลักษณ์บนเทปแล้วเลื่อนหัวเทปไปทางขวาหนึ่งช่องและเปลี่ยนสถานะตามที่ถูกกำหนดใน*ฟังก์ชันเปลี่ยนสถานะ* (transition function) สถานะถัดไปจะเป็นสถานะใดขึ้นอยู่กับสถานะปัจจุบันและสัญลักษณ์ที่อ่านได้จากเทป ออโตมาตาจำกัดหยุดทำงานเมื่ออ่านสตริงบนเทปหมด ส่วนคำตอบที่จะได้จากออโตมาตาจำกัดขึ้นกับสถานะสุดท้าย ถ้าเครื่องจบการทำงานแล้วอยู่ในสถานะที่เรียกว่า*สถานะสิ้นสุด* (final state) แล้วเครื่องจะยอมรับสตริงนั้น (หรือตอบ “ใช่”) แต่ถ้าเครื่องจบการทำงานแล้วอยู่ในสถานะที่ไม่ใช่สถานะสิ้นสุด แล้วเครื่องจะไม่ยอมรับสตริงนั้น (หรือตอบ “ไม่ใช่”) นอกจากนั้น สถานะสิ้นสุดสำหรับออโตมาตาจำกัดแต่ละเครื่องอาจมีหนึ่งสถานะหรือมากกว่าหรือไม่มีเลยก็ได้

สำหรับบทนี้ หัวข้อ 3.1 อธิบายตัวแบบทางคณิตศาสตร์ของออโตมาตาจำกัดเชิงกำหนด หัวข้อ 3.2 อธิบายแนวทางที่ใช้ในการออกแบบออโตมาตาจำกัดเชิงกำหนด หัวข้อ 3.3 อธิบายการจำลองการทำงานของออโตมาตาจำกัดเชิงกำหนดด้วยโปรแกรม สุดท้าย หัวข้อ 3.4 กล่าวถึงการนำออโตมาตาจำกัดมาประยุกต์ใช้งาน

3.1 โครงสร้างของออโตมาตาจำกัดเชิงกำหนด

ออโตมาตาจำกัดเชิงกำหนดเป็นออโตมาตาที่มีการเปลี่ยนสถานะเชิงกำหนด คือ เมื่อเครื่องอยู่ในสถานะหนึ่งและอ่านได้สัญลักษณ์หนึ่ง เครื่องสามารถเปลี่ยนสถานะไปยังสถานะถัดไปได้เพียงสถานะเดียว เช่น ถ้าออโตมาตาจำกัดเชิงกำหนด M เปลี่ยนสถานะจากสถานะ p ไปยังสถานะ q เมื่ออ่านสัญลักษณ์บนเทปเป็น 1 แล้ว ทุกครั้งที่ M อยู่ในสถานะ p และอ่านได้สัญลักษณ์ 1 บนเทปแล้ว M จะเปลี่ยนไปยังสถานะ q เสมอ ดังนั้น ออโตมาตาจำกัดเชิงกำหนดมีการเปลี่ยนสถานะซึ่งอธิบายได้ด้วยฟังก์ชันของสถานะปัจจุบันและสัญลักษณ์ที่อ่านได้ เราสามารถนิยามออโตมาตาจำกัดเชิงกำหนดได้ดังนี้

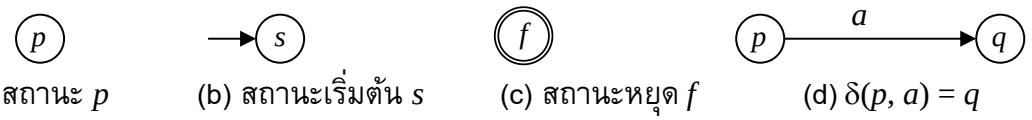
นิยาม 3.1 **ออโตมาตาจำกัดเชิงกำหนด** (deterministic finite automata) เป็นเครื่องที่อธิบายด้วย 5-สิ่งอันดับ $M = (Q, \Sigma, \delta, s, F)$ โดยที่

- เซตของสถานะ Q เป็นเซตจำกัด,
- อักขระ Σ เป็นเซตจำกัด,
- สถานะเริ่มต้น s เป็นสถานะหนึ่งใน Q,
- เซตของสถานะสิ้นสุด F ซึ่งเป็นเซตย่อยของ Q, และ
- ฟังก์ชันเปลี่ยนสถานะ δ เป็นฟังก์ชันทั้งหมด (total function) จาก $Q \times \Sigma \rightarrow Q$

จากนิยามนี้ ออโตมาตาจำกัดมีส่วนประกอบที่ใช้ในนิยาม คือ (1) เซตของสถานะ Q ซึ่งกำหนดสถานะทั้งหมดที่เป็นไปได้ของส่วนควบคุมของเครื่องนั้น (2) อักขระ Σ ซึ่งกำหนดสัญลักษณ์ที่ใช้บนเทปได้ (3) สถานะเริ่มต้น s ของออโตมาตาจำกัดคือสถานะแรกของเครื่องเมื่อเริ่มทำงาน (4) เซตของสถานะสิ้นสุด F ที่รวมอยู่ใน Q โดยที่อาจเป็นเซตว่างหรือไม่ก็ได้ และ (5) ฟังก์ชันเปลี่ยนสถานะ δ ซึ่งกำหนดการทำงานของออโตมาตาจำกัด $\delta(p, a) = q$ ระบุว่า เมื่อเครื่องอยู่ในสถานะ p และอ่านได้สัญลักษณ์ a จากเทปแล้วเครื่องจะเปลี่ยนไปอยู่ในสถานะ q และเลื่อนหัวเทปไปทางขวาหนึ่งช่อง เมื่อออโตมาตาจำกัดอ่านสตริงบนเทปหมดแล้วมันจะหยุดทำงาน ถ้าเครื่องหยุดทำงานในสถานะสิ้นสุดสถานะหนึ่งใน F แล้วเครื่องจะยอมรับสตริงบนเทป แต่ถ้าเครื่องหยุดทำงานในสถานะที่ไม่ใช่สิ้นสุดแล้วเครื่องจะไม่ยอมรับสตริงนั้น สังเกตว่าฟังก์ชันเปลี่ยนสถานะของออโตมาตาจำกัดเชิงกำหนดเป็นฟังก์ชันทั้งหมดตามนิยามทางคณิตศาสตร์ นั่นคือมันจะระบุสถานะถัดไปสำหรับสถานะปัจจุบันทุกสถานะและสัญลักษณ์ที่อ่านได้ทุกสัญลักษณ์ แต่เราอาจละการเปลี่ยนสถานะในบางกรณีซึ่งจะกล่าวถึงต่อไป

แผนภาพเปลี่ยนสถานะ (transition diagram) เป็นแผนภาพที่อธิบายการทำงานของออโตมาตาจำกัด โดยใช้วงกลมแทนสถานะดังรูป 3.2 (a) วงกลมที่ชี้ด้วยลูกศรที่หางไม่ต่อกับวงกลมใดแทนสถานะเริ่มต้นดังรูป 3.2 (b) วงกลมเส้นคู่แทนสถานะสิ้นสุดดังรูป 3.2 (c) และลูกศรจากวงกลมที่แทน

สถานะ p ไปยังวงกลมที่แทนสถานะ q แทน $\delta(p, a) = q$ หรือการเปลี่ยนสถานะจากสถานะ p ไปยังสถานะ q เมื่ออ่านได้สัญลักษณ์ a โดยสัญลักษณ์ที่กำกับบนลูกศรเป็น a ดังรูป 3.2 (d)



(a) สถานะ p (b) สถานะเริ่มต้น s (c) สถานะหยุด f (d) $\delta(p, a) = q$

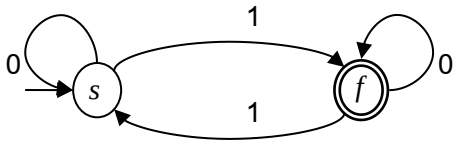
รูป 3.2 สัญลักษณ์ที่ใช้ในแผนภาพเปลี่ยนสถานะ

ตัวอย่างต่อไปนี้อธิบายการกำหนดออโตมาตาจำกัดและการทำงานของออโตมาตาจำกัดเชิงกำหนด โดยที่ออโตมาตาจำกัดนี้อ่านสตริงที่ประกอบด้วยสัญลักษณ์ 0 และ 1

ตัวอย่าง 3.1 กำหนด $M_1 = (\{s, f\}, \{0,1\}, \delta, s, \{f\})$ เป็นออโตมาตาจำกัดเชิงกำหนดที่มี δ เป็นฟังก์ชันเปลี่ยนสถานะดังกำหนดในตาราง 3.1

ตาราง 3.1 ฟังก์ชันเปลี่ยนสถานะของออโตมาตาจำกัด M_1

current state	input symbol	next state
s	0	s
s	1	f
f	0	f
f	1	s



รูป 3.3 แผนภาพเปลี่ยนสถานะของออโตมาตาจำกัดเชิงกำหนด M_1

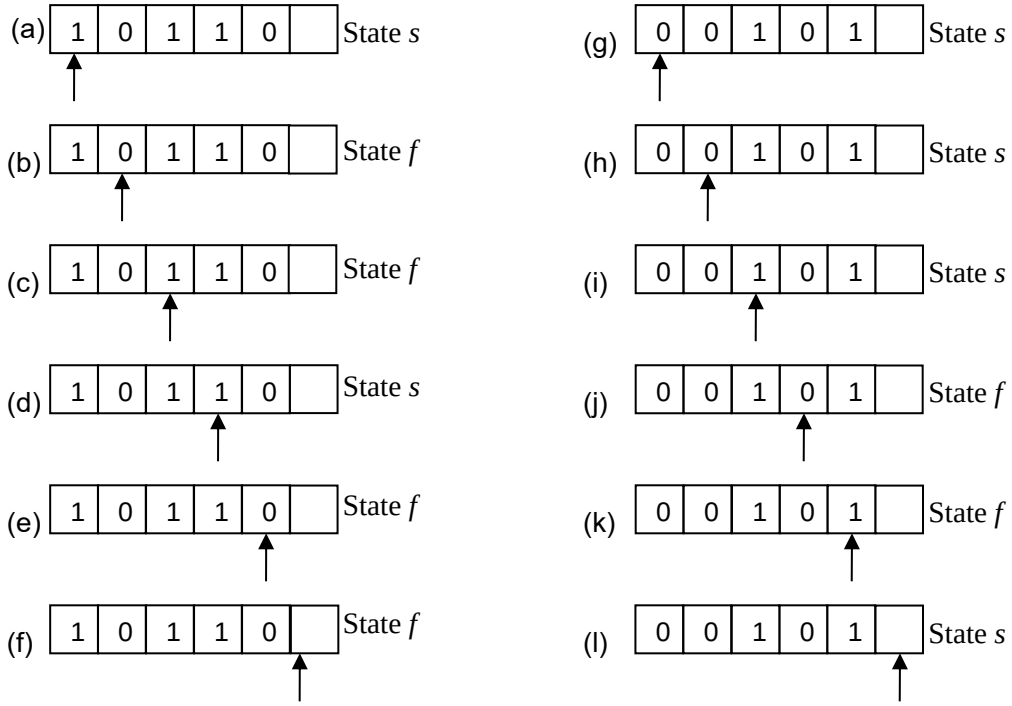
ออโตมาตาจำกัดเชิงกำหนด M_1 นี้อธิบายได้ด้วยแผนภาพเปลี่ยนสถานะดังแสดงในรูป 3.3 เมื่อกำหนดสตริงรับเข้าบนเทปต่างๆ กัน M_1 ทำงานดังที่อธิบายต่อไปนี้

สมมุติให้สตริงบนเทปเป็น 10110

- เริ่มต้น M_1 อยู่ในสถานะ s และหัวเทปอยู่ที่ช่องแรกดังแสดงในรูป 3.4 (a) ดังนั้น M_1 อยู่ในสถานะ s และอ่านได้ 1 มันจะทำงานตามฟังก์ชันเปลี่ยนสถานะ $\delta(s, 1) = f$ นั่นคือ M_1 จะเปลี่ยนสถานะไปอยู่ในสถานะ f และเลื่อนหัวอ่านไปทางขวาหนึ่งช่อง เทปและหัวเทปของ M_1 จะเป็นดังรูป 3.4 (b)
- ณ เวลานี้ M_1 อยู่ในสถานะ f และอ่านได้ 0 มันจะทำงานตามฟังก์ชันเปลี่ยนสถานะ $\delta(f, 0) = f$ นั่นคือ M_1 จะเปลี่ยนสถานะไปอยู่ในสถานะ f (ซึ่งเป็นสถานะเดิม) และเลื่อนหัวอ่านไปทางขวาหนึ่งช่อง ณ เวลานี้ เทปของ M_1 จะเป็นดังรูป 3.4 (c)
- ณ เวลานี้ M_1 อยู่ในสถานะ f และอ่านได้ 1 มันจะทำงานตามฟังก์ชันเปลี่ยนสถานะ $\delta(f, 1) = s$ นั่นคือ M_1 จะเปลี่ยนสถานะไปอยู่ในสถานะ s และเลื่อนหัวอ่านไปทางขวาหนึ่งช่อง ณ เวลานี้ เทปของ

M_1 จะเป็นดังรูป 3.4 (d)

- ณ เวลานี้ M_1 อยู่ในสถานะ s และอ่านได้ 1 มันจะทำงานตามฟังก์ชันเปลี่ยนสถานะ $\delta(s, 1) = f$ นั่นคือ M_1 จะเปลี่ยนสถานะไปอยู่ในสถานะ f และเลื่อนหัวอ่านไปทางขวาหนึ่งช่อง ณ เวลานี้ เทปของ M_1 จะเป็นดังรูป 3.4 (e)
- ณ เวลานี้ M_1 อยู่ในสถานะ f และอ่านได้ 0 มันจะทำงานตามฟังก์ชันเปลี่ยนสถานะ $\delta(f, 0) = f$ นั่นคือ M_1 จะเปลี่ยนสถานะไปอยู่ในสถานะ f (ซึ่งเป็นสถานะเดิม) และเลื่อนหัวอ่านไปทางขวาหนึ่งช่อง ณ เวลานี้ เทปของ M_1 จะเป็นดังรูป 3.4 (f) เนื่องจาก M_1 อ่านสัญลักษณ์สุดท้ายแล้ว ดังนั้น M_1 หยุดทำงานที่สถานะ f ซึ่งเป็นสถานะสิ้นสุดและยอมรับสตริง 10110



รูป 3.4 การทำงานของออโตมาตาจำกัดเชิงกำหนด M_1 เมื่อสตริงบนเทปเข้าเป็น 10110 และ 00101

ต่อมาสมมุติให้สตริงบนเทปเป็น 00101

- เริ่มต้น M_1 อยู่ในสถานะ s และหัวเทปอยู่ที่ช่องแรกดังแสดงในรูป 3.4 (g) ดังนั้น M_1 อยู่ในสถานะ s และอ่านได้ 0 มันจะทำงานตามฟังก์ชันเปลี่ยนสถานะ $\delta(s, 0) = s$ นั่นคือ M_1 จะเปลี่ยนสถานะไปอยู่ในสถานะ s (ซึ่งเป็นสถานะเดิม) และเลื่อนหัวอ่านไปทางขวาหนึ่งช่อง เทปและหัวเทปของ M_1 จะเป็นดังรูป 3.4 (h)
- ณ เวลานี้ M_1 อยู่ในสถานะ s และอ่านได้ 0 มันจะทำงานตามฟังก์ชันเปลี่ยนสถานะ $\delta(s, 0) = s$ นั่นคือ M_1 จะเปลี่ยนสถานะไปอยู่ในสถานะ s และเลื่อนหัวอ่านไปทางขวาหนึ่งช่อง ณ เวลานี้ เทปของ M_1 จะเป็นดังรูป 3.4 (i)

- ณ เวลานี้ M_1 อยู่ในสถานะ s และอ่านได้ 1 มันจะทำงานตามฟังก์ชันเปลี่ยนสถานะ $\delta(s, 1) = f$ นั่นคือ M_1 จะเปลี่ยนสถานะไปอยู่ในสถานะ f และเลื่อนหัวอ่านไปทางขวาหนึ่งช่อง ณ เวลานี้ เทปของ M_1 จะเป็นดังรูป 3.4 (j)
- ณ เวลานี้ M_1 อยู่ในสถานะ f และอ่านได้ 0 มันจะทำงานตามฟังก์ชันเปลี่ยนสถานะ $\delta(f, 0) = f$ นั่นคือ M_1 จะเปลี่ยนสถานะไปอยู่ในสถานะ f และเลื่อนหัวอ่านไปทางขวาหนึ่งช่อง ณ เวลานี้ เทปของ M_1 จะเป็นดังรูป 3.4 (k)
- ณ เวลานี้ M_1 อยู่ในสถานะ f และอ่านได้ 1 มันจะทำงานตามฟังก์ชันเปลี่ยนสถานะ $\delta(f, 1) = s$ นั่นคือ M_1 จะเป็นดังรูป 3.4 (l) เนื่องจาก M_1 อ่านสัญลักษณ์สุดท้ายแล้ว ดังนั้น M_1 หยุดทำงานที่สถานะ s ที่ไม่ใช่สถานะสิ้นสุดและไม่ยอมรับสตริง 00101

ในตัวอย่าง 3.1 จะเห็นว่าสิ่งที่ต้องระบุเพื่ออธิบายสถานภาพในขณะใดขณะหนึ่งของออโตมาตาจำกัดเชิงกำหนดประกอบด้วยสถานะของส่วนควบคุมและสตริงที่อยู่บนเทปที่ยังไม่ได้อ่านในขณะนั้น ดังนั้น สถานภาพในขณะหนึ่งของออโตมาตาจำกัดอธิบายได้ด้วย *โครงแบบ* (configuration) ซึ่งเป็นคู่อันดับ (q, α) โดยที่ q เป็นสถานะของส่วนควบคุมและ α เป็นสตริงที่ยังไม่ได้อ่าน ตัวอย่าง 3.2 แสดงโครงแบบของออโตมาตาจำกัด M_1 ซึ่งกำหนดไว้ในตัวอย่าง 3.1 ตั้งแต่เริ่มต้นจนจบการทำงาน เมื่อสตริงบนเทปเป็น 10110 และ 00101

ตัวอย่าง 3.2 เมื่อให้สตริงบนเทปของออโตมาตาจำกัด M_1 ในรูป 3.4 เป็น 10110 แล้ว M_1 มีโครงแบบเริ่มต้นเป็น $(s, 10110)$ และมีการเปลี่ยนโครงแบบในแต่ละขั้นการทำงานดังต่อไปนี้คือ $(f, 0110)$, $(f, 110)$, $(s, 10)$, $(f, 0)$, และ (f, ε) ตามลำดับ เมื่อให้สตริงบนเทปของ M_1 เป็น 00101 แล้ว M_1 มีโครงแบบเริ่มต้นเป็น $(s, 00101)$ และมีการเปลี่ยนโครงแบบในแต่ละขั้นการทำงานดังต่อไปนี้คือ $(s, 0101)$, $(s, 101)$, $(f, 01)$, $(f, 1)$, และ (s, ε) ตามลำดับ

การทำงานของออโตมาตาจำกัดอธิบายได้โดยแสดงลำดับการเปลี่ยนโครงแบบตั้งแต่เครื่องเริ่มต้นทำงานจนหยุดทำงาน การเปลี่ยนโครงแบบของออโตมาตาจำกัดเชิงกำหนดนิยามได้ดังต่อไปนี้

นิยาม 3.2 กำหนด $M = (Q, \Sigma, \delta, s, F)$ เป็นออโตมาตาจำกัดเชิงกำหนด, q_1, q_2 เป็นสถานะใน Q และ α_1, α_2 เป็นสตริงใน Σ^* โครงแบบ (q_1, α_1) **เปลี่ยนเป็นโครงแบบ** (q_2, α_2) **ในหนึ่งขั้น** (yield in one step) (ซึ่งสามารถเขียนแทนว่า $(q_1, \alpha_1) \vdash_M (q_2, \alpha_2)$) ถ้า $\delta(q_1, a) = q_2$ และ $\alpha_1 = a \cdot \alpha_2$ สำหรับ a บางตัวใน Σ

ตัวอย่าง 3.3 แสดงการเปลี่ยนโครงแบบของออโตมาตาจำกัดเชิงกำหนด M_1 ที่อธิบายในตัวอย่าง 3.2 ตามนิยาม 3.2

ตัวอย่าง 3.3 จากตัวอย่าง 3.2 เมื่อให้สตริงบนเทปของ M_1 เป็น 10110 การเปลี่ยนโครงแบบของ M_1 จะเป็นดังนี้

$(s, 10110) \vdash_{M_1} (f, 0110) \vdash_{M_1} (f, 110) \vdash_{M_1} (s, 10) \vdash_{M_1} (f, 0) \vdash_{M_1} (f, \varepsilon)$

เมื่อให้สตริงบนเทปของ M_1 เป็น 00101 การเปลี่ยนโครงแบบของ M_1 เป็นดังนี้

$(s, 00101) \vdash_{M_1} (s, 0101) \vdash_{M_1} (s, 101) \vdash_{M_1} (f, 01) \vdash_{M_1} (f, 1) \vdash_{M_1} (s, \varepsilon)$

บางครั้งเมื่อต้องการกล่าวถึงการเปลี่ยนโครงแบบโดยไม่ระบุจำนวนขั้นที่เกิดขึ้นเช่น ในนิยามหรือการพิสูจน์ที่เกี่ยวกับการทำงานของออโตมาตาจำกัดเมื่อสตริงบนเทปเป็นสตริงใดๆ เราไม่สามารถระบุจำนวนขั้นของการเปลี่ยนโครงแบบได้ นิยาม 3.3 กำหนดการเปลี่ยนโครงแบบที่ใช้กี่ขั้นก็ได้

นิยาม 3.3 กำหนด $M = (Q, \Sigma, \delta, s, F)$ เป็นออโตมาตาจำกัดเชิงกำหนด, q_1, q_2 เป็นสถานะใน Q และ α_1, α_2 เป็นสตริงใน Σ^* โครงแบบ (q_1, α_1) **เปลี่ยนเป็นโครงแบบ** (q_2, α_2) **ในศูนย์ขั้นหรือมากกว่า** (yield in zero step or more) ซึ่งสามารถเขียนแทนว่า $(q_1, \alpha_1) \vdash^*_M (q_2, \alpha_2)$ ถ้า $(q_1, \alpha_1) = (q_2, \alpha_2)$ หรือมีโครงแบบ (q, α) ของ M อย่างน้อยหนึ่งโครงแบบที่ $(q_1, \alpha_1) \vdash^*_M (q, \alpha)$ และ $(q, \alpha) \vdash_M (q_2, \alpha_2)$

จากนิยาม 3.3 การเปลี่ยนโครงแบบจากโครงแบบเริ่มต้นไปเป็นโครงแบบสิ้นสุดในตัวอย่าง 3.3 อธิบายได้ดังนี้

$(s, 10110) \vdash^*_{M_1} (f, \varepsilon)$

หรืออาจระบุการเปลี่ยนโครงแบบโดยแสดงการเปลี่ยนแปลงบางขั้น เช่น

$(s, 10110) \vdash^*_{M_1} (s, 10) \vdash^*_{M_1} (f, \varepsilon)$

นอกจากนั้นยังสามารถพิสูจน์ได้ด้วยว่า สำหรับออโตมาตาจำกัดเชิงกำหนด $M = (Q, \Sigma, \delta, s, F)$ ใดๆ ถ้า $(q, \alpha) \vdash^*_M (r, \beta)$ สำหรับสถานะ q และ r ใดๆ ใน Q, สตริง α และ β ใดๆ ใน Σ^* แล้ว $(q, \alpha\cdot\gamma) \vdash^*_M (r, \beta\cdot\gamma)$ สำหรับสตริง γ ใดๆ ใน Σ^* สมบัตินี้ใช้การพิสูจน์บ่อยๆ และผู้อ่านควรทดลองพิสูจน์สมบัตินี้เองโดยใช้การพิสูจน์แบบอุปนัยบนจำนวนขั้นของการเปลี่ยนโครงแบบ จากนั้นเราสามารถนิยามการยอมรับสตริงโดยออโตมาตาจำกัดเชิงกำหนดได้โดยใช้การเปลี่ยนโครงแบบในศูนย์ขั้นหรือมากกว่าได้ดังนี้

นิยาม 3.4 กำหนด $M = (Q, \Sigma, \delta, s, F)$ เป็นออโตมาตาจำกัดเชิงกำหนด, ω เป็นสตริงใน Σ^* M **ยอมรับ** (accept) สตริง ω ถ้า $(s, \omega) \vdash^*_M (f, \varepsilon)$ สำหรับบางสถานะ f ใน F และ M **ไม่ยอมรับ** (reject) สตริง ω ถ้า $(s, \omega) \vdash^*_M (q, \varepsilon)$ สำหรับบางสถานะ q ที่ไม่อยู่ใน F

ภาษาที่ออโตมาตาจำกัดยอมรับคือเซตของสตริงที่ออโตมาตาจำกัดยอมรับและสามารถนิยามได้ดังนี้

นิยาม 3.5 กำหนด $M = (Q, \Sigma, \delta, s, F)$ เป็นออโตมาตาจำกัดเชิงกำหนด และ L เป็นภาษาบน Σ M **ยอมรับ** (accept) ภาษา L ถ้าสำหรับทุกสตริง ω ใน L M ยอมรับ ω เราใช้ $L(M)$ แทนภาษาที่ M ยอมรับ

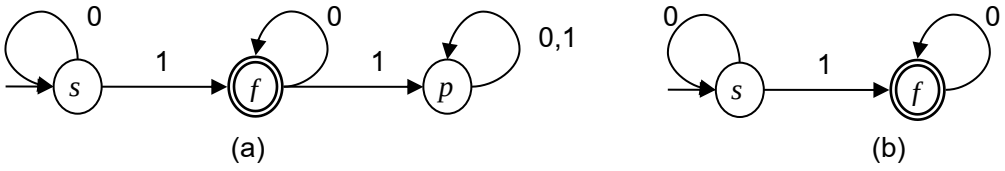
นั่นคือ $L(M) = \{\omega \in \Sigma^* \mid (s, \omega) \vdash_M^* (f, \varepsilon)$ สำหรับบางสถานะ f ใน $F\}$

ตัวอย่าง 3.4 แสดงภาษาที่ออโตมาตาจำกัดเชิงกำหนดยอมรับ

ตัวอย่าง 3.4 กำหนดออโตมาตาจำกัดเชิงกำหนด M_1 ในตัวอย่าง 3.1 ภาษาที่เครื่อง M_1 ยอมรับคือ $L(M_1) = \{\omega \in \Sigma^* \mid$ จำนวน 1 ใน ω เป็นจำนวนคี่$\}$

จากนิยาม 3.1 ฟังก์ชันเปลี่ยนสถานะ δ ของออโตมาตาจำกัดเชิงกำหนดต้องเป็นฟังก์ชันทั้งหมด (total function) จาก $Q \times \Sigma \to Q$ แต่บางครั้งออโตมาตาจำกัดมีสถานะที่ไม่นำไปสู่สถานะอื่น สถานะเหล่านี้เรียกว่า*สถานะตาย* (dead state) ดังนั้นอาจละเว้นไม่แสดงสถานะตายและการเปลี่ยนสถานะไปยังสถานะตายในการกำหนดออโตมาตาจำกัดเชิงกำหนด ทั้งนี้จะทำให้ฟังก์ชันเปลี่ยนสถานะเป็นฟังก์ชันบางส่วน (partial function) ตัวอย่าง 3.5 แสดงของออโตมาตาจำกัดเชิงกำหนดที่มีสถานะตาย

ตัวอย่าง 3.5 กำหนด $M_2 = (\{s, p, f\}, \{0,1\}, \delta, s, \{f\})$ เป็นออโตมาตาจำกัดเชิงกำหนดโดยที่ δ เป็นฟังก์ชันเปลี่ยนสถานะที่กำหนดในแผนภาพเปลี่ยนสถานะในรูป 3.5 (a) สถานะ p เป็นสถานะตาย ดังนั้น M_2 นิยาม ใหม่ได้เป็น $M_3 = (\{s, f\}, \{0,1\}, \delta', s, \{f\})$ โดยที่ δ' เป็นฟังก์ชันเปลี่ยนสถานะที่กำหนดในแผนภาพเปลี่ยนสถานะในรูป 3.5 (b)



รูป 3.5 แผนภาพเปลี่ยนสถานะของออโตมาตาจำกัด M_2 และ M_3

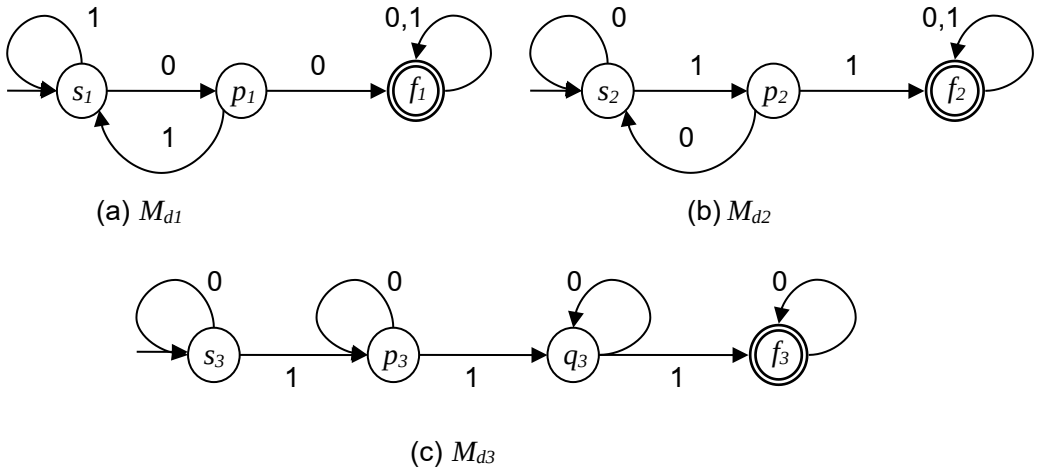
เมื่อเข้าใจการทำงานของออโตมาตาจำกัดเชิงกำหนดและสามารถระบุภาษาที่ออโตมาตาจำกัดเชิงกำหนดเครื่องหนึ่งยอมรับได้แล้วเรายังต้องสามารถสร้างออโตมาตาจำกัดเชิงกำหนดได้ ตัวอย่างต่อไปนี้แสดงออโตมาตาจำกัดเชิงกำหนดที่ยอมรับภาษาที่กำหนด

ตัวอย่าง 3.6 กำหนดอักขระ $\Sigma = \{0, 1\}$ จงสร้างออโตมาตาจำกัดเชิงกำหนดที่ยอมรับภาษาต่อไปนี้

$L_1 = \{\omega \in \Sigma^* \mid 00$ เป็นสตริงย่อยใน $\omega\}$

$L_2 = \{\omega \in \Sigma^* \mid 11$ เป็นสตริงย่อยใน $\omega\}$

L_3 = {ω∈Σ* | จำนวนของ 1 ใน ω เป็น 3}



(a) M_{d1} (b) M_{d2}

(c) M_{d3}

รูป 3.6 ออโตมาตาจำกัดที่ยอมรับ L_1, L_2 และ L_3

M_{d1}, M_{d2} และ M_{d3} ที่แสดงในรูป 3.6 (a), (b) และ (c) เป็นออโตมาตาจำกัดเชิงกำหนดที่ยอมรับ L_1, L_2 และ L_3 ตามลำดับ ออโตมาตา M_{d3} ไม่ได้แสดงสถานะตายไว้ สังเกตได้จากสถานะ f_3 ที่ไม่ระบุการเปลี่ยนสถานะเมื่ออ่านได้สัญลักษณ์ 1 ดังนั้น ถ้า M_{d3} อ่านได้สัญลักษณ์ 0 เป็นตัวที่ 4 มันจะไม่สามารถเปลี่ยนสถานะออกจากสถานะ f_3 ได้และจะหยุดทำงานโดยไม่ยอมรับสตริงบนเทป

บางครั้งการสร้างออโตมาตาจำกัดเชิงกำหนดเป็นงานที่ไม่ง่ายนักสำหรับผู้ที่ไม่มีประสบการณ์ ในหัวข้อต่อไปนี้อธิบายแนวทางในการสร้างออโตมาตาจำกัดเชิงกำหนดซึ่งจะช่วยให้ทำงานได้ง่ายขึ้น

3.2 การสร้างออโตมาตาจำกัดเชิงกำหนด

การสร้างออโตมาตาจำกัดเชิงกำหนดใช้แนวคิดว่า สถานะของออโตมาตาจำกัดทำหน้าที่จำสมบัติของสตริงที่อ่านเข้ามา ดังนั้นสิ่งที่สำคัญที่ต้องทราบในการสร้างออโตมาตาจำกัดคือแต่ละสถานะของออโตมาตาจำกัดจำอะไร เช่น สถานะของออโตมาตาจำกัด M_1 ในตัวอย่าง 3.1 จำจำนวนสัญลักษณ์ “1” ที่อยู่ในสตริงที่อ่านมาว่าเป็นจำนวนคู่หรือจำนวนคี่ บางครั้งอาจตัดสินใจได้ไม่ง่ายนักว่าสิ่งที่ออโตมาตาจำกัดต้องจำคืออะไร อย่างไรก็ตามสมบัติของสตริงในภาษานั้นอาจเป็นตัวชี้แนะที่ดีได้

นอกจากนั้นเราต้องพิจารณาว่าออโตมาตาจำกัดต้องการสถานะกี่สถานะ หากใช้สถานะน้อยกว่าที่จำเป็นออโตมาตาจำกัดจะทำงานไม่ได้ หากใช้สถานะมากกว่าที่จำเป็นออโตมาตาจำกัดอาจทำงานได้แต่ซับซ้อนมากขึ้น ตัวอย่าง 3.1 ใช้สองสถานะจำว่าจำนวนสัญลักษณ์ “1” ที่อยู่ในสตริงที่อ่านมาเป็นจำนวนคู่หรือจำนวนคี่ จากนั้นต้องทดลองสร้างฟังก์ชันเปลี่ยนสถานะ ถ้าหากสร้างไม่ได้ต้องเพิ่มจำนวนสถานะหรือเปลี่ยนสิ่งที่ให้ออโตมาตาจำกัดจำแล้วสร้างฟังก์ชันเปลี่ยนสถานะใหม่

ตัวอย่างต่อไปนี้แสดงการสร้างออโตมาตาจำกัดที่ซับซ้อนขึ้น สำหรับผู้อ่านที่ต้องการพัฒนาทักษะการสร้างออโตมาตาจำกัด ขอให้ทดลองใช้เวลาคิดเองก่อนที่จะอ่านคำอธิบายที่ให้ไว้ในตัวอย่างนี้

ตัวอย่าง 3.7 จงสร้างออโตมาตาจำกัดเชิงกำหนดที่ยอมรับ $L_4 = \{\omega \in \{0,1\}^* | \ \omega$ เป็นตัวแทนเลขฐาน 2 ที่หารด้วย 3 ลงตัว$\}$

ขั้นแรก กำหนดว่าจะให้ออโตมาตาจำกัดจำอะไร ในกรณีนี้ตัวชี้แนะคือสมบัติของสตริงในภาษา L_4 ซึ่งเป็นสตริงที่แทนเลขฐาน 2 ที่หารด้วย 3 ลงตัว ดังนั้นอาจเดาว่าออโตมาตาจำกัดต้องมีสถานะที่จำว่าสตริงที่อ่านเข้ามาแทนเลขฐาน 2 ที่หารด้วย 3 ลงตัว

จากนั้นจึงตัดสินใจว่าออโตมาตาจำกัดต้องใช้สถานะกี่สถานะ เราจะลองพิจารณาสองทางเลือก ทางที่หนึ่งใช้สองสถานะและทางที่สองใช้สามสถานะ สำหรับทางแรก ให้ออโตมาตาจำกัดมีสองสถานะจำว่าสตริงที่อ่านเข้ามาแทนเลขฐาน 2 ที่หารด้วย 3 ลงตัวและไม่ลงตัว แต่เราไม่สามารถสร้างฟังก์ชันเปลี่ยนสถานะได้สำหรับทางเลือกนี้เพราะถ้าสตริง ω ที่อ่านเข้ามาแทนเลขที่หารด้วย 3 ไม่ลงตัวและสัญลักษณ์ต่อไปที่อ่านเข้ามาได้คือ 1 แล้ว สตริงใหม่คือ ω1 อาจแทนเลขที่หารด้วย 3 ลงตัวหรือไม่ก็ได้ เช่น ถ้า $\omega=(100)_2=(4)_{10}$ และสัญลักษณ์ต่อไปที่อ่านเข้ามาได้คือ 1 แล้ว $\omega1=(1001)_2=(9)_{10}$ ซึ่งหารด้วย 3 ลงตัว แต่ถ้า $\omega=(101)_2=(5)_{10}$ และสัญลักษณ์ต่อไปที่อ่านเข้ามาได้คือ 1 แล้ว $\omega1=(1011)_2=(11)_{19}$ ซึ่งหารด้วย 3 ไม่ลงตัว

สำหรับทางที่สอง ให้ออโตมาตาจำกัดมี 3 สถานะจำเศษของการหารจำนวนที่แทนด้วยสตริงที่อ่านเข้ามาด้วย 3 ซึ่งอาจเป็น 0 หรือ 1 หรือ 2 (ถ้าเศษของการหารด้วย 3 เป็น 0 คือจำนวนหารด้วย 3 ลงตัว) เราสามารถแทนจำนวนที่หารด้วย 3 แล้วเหลือเศษเป็น i ด้วย 3n+i เมื่อ n เป็นจำนวนเต็มที่ไม่เป็นลบ จากนั้นเราพิจารณาการเปลี่ยนจำนวนที่แทนด้วยสตริงเมื่ออ่านหลักถัดไปเข้ามา สมมติว่าเครื่องอ่านสตริง 3 สัญลักษณ์แรกมาได้ 101 ซึ่งแทนจำนวน 5 ถ้าอ่านได้สัญลักษณ์ต่อมาเป็น 0 จะได้สตริง 1010 ซึ่งแทนจำนวน 10 นั่นคือจำนวนใหม่เป็น 2 เท่าของจำนวนเดิม และ ถ้าจำนวนเดิมเป็น 3n+i แล้วจำนวนใหม่ที่ได้เป็น 2(3n+i)= 6n+2i ถ้าอ่านได้สัญลักษณ์ต่อมาเป็น 1 จะได้สตริง 1011 ซึ่งแทนจำนวน 11 นั่นคือจำนวนใหม่มากกว่า 2 เท่าของจำนวนเดิมอยู่ 1 และ ถ้าจำนวนเดิมเป็น 3n+i แล้วจำนวนใหม่ที่ได้เป็น 2(3n+i)+1= 6n+2i+1 ดังนั้นเราสามารถสร้างฟังก์ชันเปลี่ยนสถานะได้โดยพิจารณาตาราง 3.2

ตาราง 3.2 การเปลี่ยนเศษของสตริงที่อ่านเข้ามาได้เมื่อ *n* เป็นจำนวนเต็มที่ไม่เป็นลบ

เลข x ที่แทนด้วยสตริงที่อ่านเข้ามา	เศษของการหาร x ด้วย 3	สัญลักษณ์ถัดมาบนเทป	เลข y ที่แทนด้วยสตริงที่อ่านเข้ามาตามด้วยสัญลักษณ์ถัดมา	เศษของการหาร y ด้วย 3
3*n*	0	0	6*n*=3(2n)	0
3*n*	0	1	6*n*+1=3(2n)+1	1
3*n*+1	1	0	6*n*+2=3(2n)+2	2
3*n*+1	1	1	6*n*+3=3(2n+1)	0
3*n*+2	2	0	6*n*+4=3(2n+1)+1	1
3*n*+2	2	1	6*n*+5=3(2n+1)+2	2

จากตาราง 3.2 ออโตมาตาจำกัดสามารถใช้สามสถานะโดยให้สถานะ q_0, q_1 และ q_2 จำว่าเศษของการหารเลขที่แทนด้วยสตริงที่อ่านเข้ามาด้วย 3 เป็น 0, 1 และ 2 ตามลำดับ ฟังก์ชันเปลี่ยนสถานะเป็นดังตาราง 3.3

ตาราง 3.3 ฟังก์ชันเปลี่ยนสถานะของ M_4

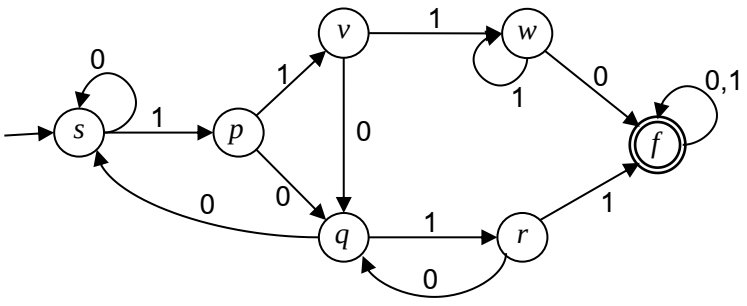
สถานะปัจจุบัน	สัญลักษณ์ที่อ่านเข้ามา	สถานะถัดไป
q_0	0	q_0
q_0	1	q_1
q_1	0	q_2
q_1	1	q_0
q_2	0	q_1
q_2	1	q_2

ดังนั้น ออโตมาตาจำกัดเชิงกำหนด $M_4 = (\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_0\})$ ซึ่ง δ กำหนดในตาราง 3.3 เป็นออโตมาตาจำกัดที่ยอมรับ L_4

เนื่องจากสถานะ q_1 และ q_2 จำว่าเศษของการหารเลขที่แทนด้วยสตริงที่อ่านเข้ามาด้วย 3 เป็น 1 และ 2 ตามลำดับ ถ้าเปลี่ยนสถานะสิ้นสุดของ M_4 เป็น q_1 และได้ออโตมาตาจำกัด $(\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_1\})$ เครื่องนี้จะยอมรับ {ω∈{0,1}*| ω เป็นตัวแทนเลขฐาน 2 ที่หารด้วย 3 แล้วเหลือเศษ 1} แต่ถ้าเปลี่ยนสถานะสิ้นสุดของ M_4 เป็น q_2 และได้ออโตมาตาจำกัด $(\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_2\})$ เครื่องนี้จะยอมรับ {ω∈{0,1}*| ω เป็นตัวแทนเลขฐาน 2 ที่หารด้วย 3 แล้วเหลือเศษ 2} ซึ่ง δ กำหนดไว้ในตาราง 3.3

ตัวอย่างต่อไปนี้แสดงการสร้างออโตมาตาจำกัดที่ตรวจสอบว่าสตริงย่อยหนึ่งในสองสตริงที่กำหนดอยู่ในสตริงที่รับเข้ามาหรือไม่

<u>ตัวอย่าง 3.8</u> ออโตมาตาจำกัดเชิงกำหนดในรูป 3.7 ยอมรับ L_5 = {ω∈{0,1}*| 1011 หรือ 1110 เป็นสตริงย่อยใน ω}



รูป 3.7 ออโตมาตาจำกัดเชิงกำหนดที่ยอมรับ L_5

สถานะในออโตมาตาจำกัดนี้จำว่าส่วนท้าย (suffix) ของสตริงที่อ่านมาเป็นส่วนต้น (prefix) ใดของสตริง 1011 และ 1110 คือ ε, 1, 10, 11, 101 และ 111 รวมทั้งสถานะที่จำว่าอ่านได้สตริง 1011

หรือ 1110 แล้ว ดังนั้น ออโตมาตานี้มีสถานะดังนี้

- สถานะ s ที่จำว่าส่วนท้ายของสตริงที่อ่านมาถึงขณะนั้นยังไม่มีส่วนต้นใดของทั้งสองสตริงที่ต้องการ
- สถานะ p ที่จำว่าส่วนท้ายของสตริงที่อ่านมาถึงขณะนั้นมี 1 ที่เป็นส่วนต้นของทั้งสองสตริง
- สถานะ q และ v ที่จำว่าส่วนท้ายของสตริงที่อ่านมาถึงขณะนั้นมี 10 และ 11 ที่เป็นส่วนต้นของสตริง 1011 และ 1110 ตามลำดับ
- สถานะ r และ w ที่จำว่าส่วนท้ายของสตริงที่อ่านมาถึงขณะนั้นมี 101 และ 111 ที่เป็นส่วนต้นของสตริง 1011 และ 1110 ตามลำดับ
- สถานะ f ที่จำว่าอ่านได้สตริง 1011 หรือ 1110 แล้ว

สำหรับการเปลี่ยนสถานะที่แสดงในรูป 3.8 เมื่อออโตมาตานี้อยู่ที่สถานะ s ถ้าอ่านได้สัญลักษณ์ 0 จะยังอยู่ที่สถานะ s เพื่อจำว่าขณะนั้นยังไม่ได้ส่วนต้นของสตริงที่ต้องการ ถ้าอ่านได้สัญลักษณ์ 1 จะเปลี่ยนไปอยู่ที่สถานะ p เพื่อจำว่าขณะนั้นอ่านได้ส่วนต้น 1 แล้ว

เมื่อออโตมาตานี้อยู่ที่สถานะ p ซึ่งหมายถึงว่าอ่านได้ 1 มาแล้ว ดังนั้นถ้าอ่านได้สัญลักษณ์ 0 จะเปลี่ยนไปอยู่ที่สถานะ q เพื่อจำว่าขณะนั้นอ่านได้ส่วนต้น 10 แล้ว ถ้าอ่านได้สัญลักษณ์ 1 จะเปลี่ยนไปอยู่ที่สถานะ v เพื่อจำว่าขณะนั้นอ่านได้ส่วนต้น 11 แล้ว

เมื่อออโตมาตานี้อยู่ที่สถานะ q ซึ่งหมายถึงว่าอ่านได้ 10 มาแล้ว ดังนั้นถ้าอ่านได้สัญลักษณ์ 0 สตริงที่อ่านมาลงท้ายด้วย 100 และจะเปลี่ยนไปยังสถานะ s เพราะ 100 ไม่มีส่วนต้นใดของสตริงที่ต้องการ ถ้าอ่านได้สัญลักษณ์ 1 จะเปลี่ยนไปอยู่ที่สถานะ r เพื่อจำว่าขณะนั้นอ่านได้ส่วนต้น 101 ของสตริง 1011 แล้ว

เมื่อออโตมาตานี้อยู่ที่สถานะ v ซึ่งหมายถึงว่าอ่านได้ 11 มาแล้ว ดังนั้นถ้าอ่านได้สัญลักษณ์ 0 สตริงที่อ่านมาลงท้ายด้วย 110 และจะเปลี่ยนไปยังสถานะ q เพราะ 110 ลงท้ายด้วย 10 ที่เป็นส่วนต้นของสตริง 1011 ถ้าอ่านได้สัญลักษณ์ 1 จะเปลี่ยนไปอยู่ที่สถานะ w เพื่อจำว่าขณะนั้นอ่านได้ส่วนต้น 111 ของสตริง 1110 แล้ว

เมื่อออโตมาตานี้อยู่ที่สถานะ r ซึ่งหมายถึงว่าอ่านได้ 101 มาแล้ว ดังนั้นถ้าอ่านได้สัญลักษณ์ 0 สตริงที่อ่านมาลงท้ายด้วย 1010 และจะเปลี่ยนไปยังสถานะ q เพราะ 1010 ลงท้ายด้วย 10 ที่เป็นส่วนต้นของสตริง 1011 ถ้าอ่านได้สัญลักษณ์ 1 จะเปลี่ยนไปอยู่ที่สถานะ f เนื่องจากได้สตริง 1011 แล้ว

เมื่อออโตมาตานี้อยู่ที่สถานะ w ซึ่งหมายถึงว่าอ่านได้ 111 มาแล้ว ดังนั้นถ้าอ่านได้สัญลักษณ์ 0 สตริงที่อ่านมาลงท้ายด้วย 1110 และจะเปลี่ยนไปยังสถานะ f เนื่องจากได้สตริง 1110 แล้ว ถ้าอ่านได้สัญลักษณ์ 1 จะยังอยู่ที่สถานะ w เพราะ 1111 ลงท้ายด้วย 111 ที่เป็นส่วนต้นของสตริง 1110

เมื่อออโตมาตานี้อยู่ที่สถานะ f จะไม่ต้องตรวจสอบสัญลักษณ์หลังจากนั้น ดังนั้นไม่ว่าจะอ่านได้ 0 หรือ 1 ออโตมาตาจะไม่เปลี่ยนสถานะอีก

ออโตมาตาจำกัดที่กล่าวถึงในหัวข้อนี้เป็นเครื่องเชิงกำหนด นั่นคือเมื่อออโตมาตาจำกัดอยู่ที่

สถานะหนึ่งและอ่านได้สัญลักษณ์หนึ่งแล้วมันจะเปลี่ยนสถานะแบบเดียวกันทุกครั้ง หัวข้อต่อไปนี้กล่าวถึงออโตมาตาจำกัดเชิงไม่กำหนดที่มีการทำงานต่างออกไป แต่จะช่วยให้สร้างออโตมาตาจำกัดที่ทำงานที่ซับซ้อนได้ง่ายขึ้น

3.3 การจำลองการทำงานของออโตมาตาจำกัดเชิงกำหนด

เนื่องจากออโตมาตาจำกัดถูกนำไปใช้ในการสร้างวงจร ตัวแปลภาษา (compiler) วิศวกรรมซอฟต์แวร์ การจำลองการทำงานของออโตมาตาจำกัดจึงจำเป็นสำหรับการประยุกต์เหล่านี้ หัวข้อนี้จะแสดงโปรแกรมจำลองการทำงานของออโตมาตาจำกัดเชิงกำหนด รูป 3.8 แสดงตัวอย่างโปรแกรมภาษาไพธอนที่จำลองออโตมาตา M_1 ในตัวอย่าง 3.1 โดยอักขระ Σ = {0, 1}

```
01: startState='s'            # start state
02: finalStates={'f'}         # SET of final states
03: tFunc={                   # transition function
04:        ('s','0'):'s',
05:        ('s','1'):'f',
06:        ('f','0'):'f',
07:        ('f','1'):'s'}
08: tape='10110'              # input tape
09: state=startState         # At the beginning, the automata is in the start state
10: for sym in tape:                  # Loop until the whole input tape is read.
11:     nextState=tFunc[(state,sym)] # Get next state from the transition function
12:     state=nextState               # Change state
13: if (state in finalStates):        # If the automata stops in a final state.
14:    print('ACCEPT')                # accept the input on the tape.
15: else:                             # Otherwise,
16:    print('REJECT')                # reject the input string.
```

รูป 3.8 โปรแกรมจำลองการทำงานของออโตมาตา M_1 ในตัวอย่าง 3.1

ตัวแปร `startState` เก็บสถานะเริ่มต้นของออโตมาตาจำกัด ตัวแปร `finalStates` เก็บเซตของสถานะสิ้นสุด ตัวแปร `tFunc` เก็บฟังก์ชันเปลี่ยนสถานะของออโตมาตาจำกัดโดยใช้โครงสร้าง dictionary ของภาษาไพธอนและมี key เป็น tuple ของสถานะและสัญลักษณ์ที่อ่านจากเทป เช่น ค่าแรกใน `tFunc` คือ `('s', '0'): 's'` ระบุว่า ถ้าออโตมาตานี้อยู่ที่สถานะ `s` และอ่านได้สัญลักษณ์ `0` แล้วมันจะไปอยู่ในสถานะ `s` อีก ตัวแปร `tape` เก็บสตริงที่อยู่บนเทปของ M_1 ตัวแปร `sym` เก็บสัญลักษณ์ที่อ่านจากเทป ตัวแปร `state` เก็บสถานะปัจจุบันของออโตมาตาจำกัด

เมื่อเริ่มทำงาน ออโตมาตาอยู่ในสถานะเริ่มต้น ดังนั้นจึงให้ตัวแปร `state` เก็บค่าของตัวแปร `startState` (บรรทัด 9) จากนั้นจะวนอ่านสัญลักษณ์จากเทปจนหมด (บรรทัด 10) ในการเปลี่ยนสถานะแต่ละครั้ง เราจะอ่านสถานะถัดไปจากตัวแปร `tFunc` โดยระบุ key เป็นสถานะปัจจุบันและสัญลักษณ์ที่อ่านได้ในตัวแปร `state` และ `sym` ตามลำดับ (บรรทัด 11-12)

เมื่ออ่านเทปจนหมดแล้วเครื่องอยู่ในสถานะสิ้นสุด ก็จะยอมรับสตริงบนเทป (บรรทัด 13-14) ถ้าเครื่องไม่อยู่ในสถานะสิ้นสุด ก็จะไม่ยอมรับสตริงบนเทป (บรรทัด 15-16)

หัวข้อถัดไปแสดงตัวอย่างการนำออโตมาตาจำกัดเชิงกำหนดไปใช้ในงานต่าง ๆ

3.4 การประยุกต์ใช้ออโตมาตาจำกัดเชิงกำหนด

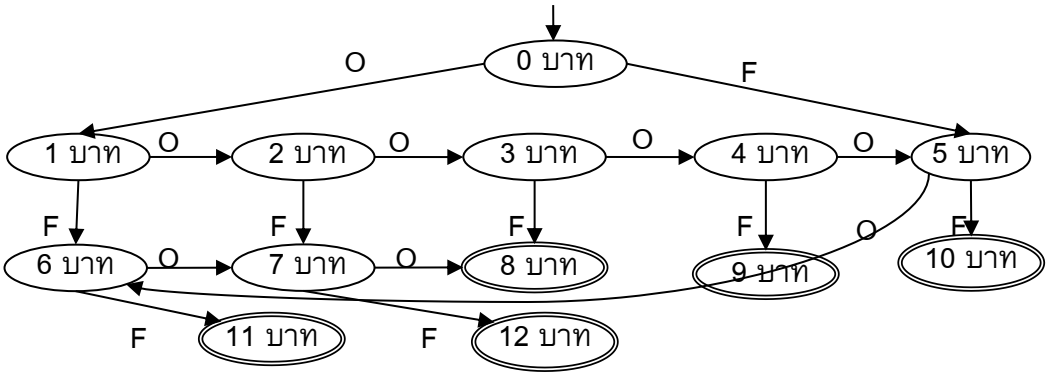
ออโตมาตาจำกัดเป็นตัวแบบที่อธิบายการทำงานของระบบตามข้อมูลรับเข้า ดังนั้นจึงนำไปประยุกต์ใช้ได้งานหลายประเภท ในหัวข้อนี้จะแสดงการนำไปใช้ออกแบบวงจร การตรวจหาสตริงย่อย และ การอธิบายการทำงานของโปรแกรมด้วย UML (Unified Modeling Language)

3.4.1 การออกแบบวงจรเชิงลำดับ

วิศวกรใช้ออโตมาตาจำกัดอธิบายพฤติกรรมของระบบเพื่อสร้างวงจรเชิงลำดับ (Sequential circuit) ให้ทำงานตามพฤติกรรมที่ต้องการ [7], [10] ระบบเหล่านี้มีสถานะที่เปลี่ยนไปตามสิ่งที่รับเข้ามาในระบบโดยสถานะของออโตมาตาจำกัดจำสิ่งที่จำเป็นสำหรับการทำงานของระบบ ออโตมาตาจำกัดที่แสดงในรูป 3.9 (a) อธิบายพฤติกรรมของวงจรควบคุมการทอนเงินในเครื่องขายสินค้าอัตโนมัติที่ขายสินค้าราคา 8 บาท ออโตมาตานี้มีสถานะที่จำว่าเครื่องขายสินค้ารับเงินมาเท่าไรแล้ว แผนภาพเปลี่ยนสถานะของออโตมาตานี้ระบุว่าแต่ละสถานะจำว่ารับเงินมาเท่าไรตามข้อความที่แสดงในสถานะ

เครื่องเปลี่ยนสถานะตามเงินที่รับเข้ามาว่าเป็นเหรียญ 1 บาท หรือ 5 บาท ซึ่งแสดงด้วยสัญลักษณ์ O และ F ตามลำดับ การเปลี่ยนสถานะของออโตมาตานี้ขึ้นอยู่กับเงินที่มีอยู่แล้วและเงินที่รับเข้ามา เช่น เมื่อออโตมาตาอยู่ในสถานะที่ระบุว่าได้เงินมา 3 บาทแล้วและได้รับเหรียญ 1 บาท ออโตมาตาจะเปลี่ยนไปอยู่ในสถานะที่ระบุว่าได้เงิน 4 บาทแล้ว แต่ถ้าได้รับเหรียญ 5 บาท ออโตมาตาจะเปลี่ยนไปอยู่ในสถานะที่ระบุว่าได้เงิน 8 บาทแล้ว

จากออโตมาตาจำกัดนี้เราสามารถสร้างวงจรได้โดยให้แต่ละสถานะมีเลขกำกับเป็นเลขฐานสองของจำนวนเงินที่รับเข้ามา เช่น เลขกำกับ 0011 สำหรับสถานะ 3 บาท ข้อมูลรับเข้าเป็นการรับเหรียญ 1 บาทหรือ 5 บาท (ในที่นี้ ไม่อนุญาตให้หยอดเหรียญ 10 บาทเพื่อลดขนาดของวงจร) นั่นคือการรับเหรียญ 1 บาทแทนด้วย 0 และการรับเหรียญ 5 บาทแทนด้วย 1 จากนั้นเราสร้างตารางค่าความจริงดังแสดงในรูป 3.9 (b) โดยให้ข้อมูลเข้าในตารางนี้เป็นสถานะปัจจุบัน ($s_3\, s_2\, s_1\, s_0$) และข้อมูลเหรียญที่ได้ (c) ข้อมูลออกในตารางนี้เป็นสถานะถัดไป ($s'_3\, s'_2\, s'_1\, s'_0$) ดังนั้นเราสามารถเขียนฟังก์ชันของสถานะถัดไปได้ดังรูป 3.9 (c) และฟังก์ชันนี้สามารถนำไปสร้างเป็นวงจรเชิงลำดับได้ดังแสดงในรูป 3.9 (d)



(a) ออโตมาตาจำกัดที่อธิบายพฤติกรรมของวงจรควบคุมการทอนเงินในเครื่องขายสินค้าอัตโนมัติ

state $s_3s_2s_1s_0$	input c	next state $s'_3s'_2s'_1s'_0$	state $s_3s_2s_1s_0$	input c	next state $s'_3s'_2s'_1s'_0$
0000	0	0001	0100	0	0101
0000	1	0101	0100	1	1001
0001	0	0010	0101	0	0110
0001	1	0110	0101	1	1010
0010	0	0011	0110	0	0111
0010	1	0111	0110	1	1011
0011	0	0100	0111	0	1000
0011	1	1000	0111	1	1100

(b) ตารางค่าความจริง

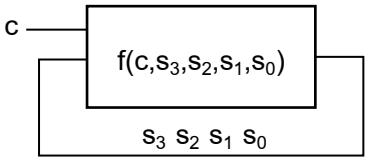
$s'_0 = \bar{s}_0$

$s'_1 = \bar{s}_0 s_1 + \bar{s}_1 s_0$

$s'_2 = \bar{c}(\bar{s}_1 s_2 + \bar{s}_0 s_2 + \bar{s}_3 \bar{s}_2 s_1 s_0) + \bar{s}_3 \bar{s}_2 c(\bar{s}_1 + \bar{s}_0) + c s_2 s_1 s_0$

$s'_3 = c(s_2 + s_1 s_0) + s_2 s_1 s_0$

(c) ฟังก์ชัน $f(c,s_3,s_2,s_1,s_0)$ ของสถานะถัดไป



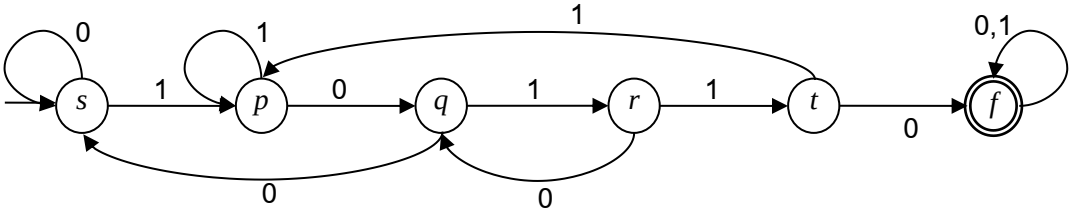
(d) วงจร

รูป 3.9 การใช้ออโตมาตาจำกัดอธิบายพฤติกรรมของเครื่องขายสินค้าอัตโนมัติเพื่อนำไปออกแบบวงจรเชิงลำดับ

3.4.2 ขั้นตอนวิธีตรวจหาสตริงย่อย

การตรวจหาสตริงย่อยในข้อความเป็นงานพื้นฐานที่จำเป็นสำหรับการประมวลผลข้อความ (text processing) [2] เมื่อกำหนดสตริงย่อยที่ต้องการหา เราสามารถสร้างออโตมาตาจำกัดเชิงกำหนดที่ตรวจหาสตริงนั้นได้ดังแสดงในตัวอย่าง 3.9

ตัวอย่าง 3.9 กำหนด $\Sigma = \{0, 1\}$ และ $\alpha = 10110$ เราสามารถสร้างออโตมาตาจำกัดเชิงกำหนดในรูป 3.10 ที่ตรวจสอบว่า α เป็นสตริงย่อยในสตริงที่รับเข้ามาหรือไม่



รูป 3.10 ออโตมาตาจำกัดที่ตรวจสอบว่า α = 10110 อยู่ในสตริงที่รับเข้ามาหรือไม่

สถานะใน M จำว่าสตริงที่อ่านเข้ามาลงท้ายด้วยส่วนหน้า (prefix) ใดของสตริง α ดังนี้

- สถานะ s จำว่าเรายังไม่อ่านได้ส่วนใดของสตริง α เลย
- สถานะ p จำว่าเราอ่านได้ส่วนหน้าของสตริง α มา 1 ตัว คือ 1
- สถานะ q จำว่าเราอ่านได้ส่วนหน้าของสตริง α มา 2 ตัว คือ 10
- สถานะ r จำว่าเราอ่านได้ส่วนหน้าของสตริง α มา 3 ตัว คือ 101
- สถานะ t จำว่าเราอ่านได้ส่วนหน้าของสตริง α มา 4 ตัว คือ 1011
- สถานะ f จำว่าเราอ่านได้สตริง α มาแล้ว

เมื่อออโตมาตาอยู่ในสถานะ s แล้วอ่านได้สัญลักษณ์ 0 หมายความว่า สตริงที่ออโตมาตาอ่านได้ในขณะนั้นลงท้ายด้วย 0 ดังนั้น สตริงที่อ่านมานี้ยังไม่มีส่วนหน้าของ α เลย ดังนั้นจึงอยู่ที่สถานะ s เช่นเดิม แต่ถ้าอ่านได้สัญลักษณ์ 1 หมายความว่า สตริงที่ออโตมาตาอ่านมาลงท้ายด้วยส่วนหน้าตัวแรกของ α ดังนั้นจึงเปลี่ยนไปอยู่ที่สถานะ p

เมื่อออโตมาตาอยู่ในสถานะ p แล้วอ่านได้สัญลักษณ์ 0 หมายความว่า สตริงที่ออโตมาตาอ่านได้ขณะนั้นลงท้ายด้วย 10 ที่เป็นส่วนหน้า 2 ตัวของ α ดังนั้นจึงเปลี่ยนไปอยู่ที่สถานะ q แต่ถ้าอ่านได้สัญลักษณ์ 1 หมายความว่า สตริงที่ออโตมาตาอ่านมาลงท้ายด้วย 11 ที่มีเพียงส่วนหน้าตัวแรกของ α ดังนั้นจึงยังอยู่ที่สถานะ p

เมื่อออโตมาตาอยู่ในสถานะ q แล้วอ่านได้สัญลักษณ์ 0 หมายความว่า สตริงที่ออโตมาตาอ่านได้ขณะนั้นลงท้ายด้วย 100 ซึ่งไม่ลงท้ายด้วยส่วนหน้าใดของ α ดังนั้นจึงเปลี่ยนไปอยู่ที่สถานะ s แต่ถ้าอ่านได้สัญลักษณ์ 1 หมายความว่า สตริงที่ออโตมาตาอ่านมาลงท้ายด้วย 101 มีส่วนหน้า 3 ตัวแรกของ α ดังนั้นจึงเปลี่ยนไปอยู่ที่สถานะ r

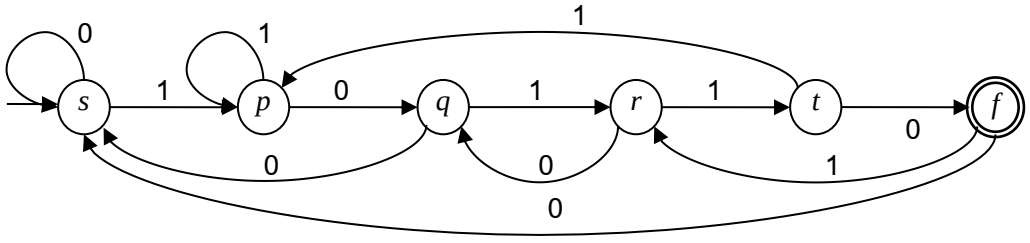
เมื่อออโตมาตาอยู่ในสถานะ r แล้วอ่านได้สัญลักษณ์ 0 หมายความว่า สตริงที่ออโตมาตาอ่านได้ในขณะนั้นลงท้ายด้วย 1010 ที่ลงท้ายด้วยส่วนหน้า 2 ตัวของ α ดังนั้นจึงเปลี่ยนไปอยู่ที่สถานะ q แต่ถ้าอ่านได้สัญลักษณ์ 1 หมายความว่า สตริงที่ออโตมาตาอ่านมาลงท้ายด้วย 1011 ที่ส่วนหน้า 4 ตัวแรกของ α ดังนั้นจึงยังอยู่ที่สถานะ p

เมื่อออโตมาตาอยู่ในสถานะ t แล้วอ่านได้สัญลักษณ์ 0 หมายความว่า สตริงที่ออโตมาตาอ่านมาลงท้ายด้วยสตริง α ที่ต้องการ ดังนั้นจึงเปลี่ยนไปอยู่ที่สถานะสิ้นสุด f แต่ถ้าอ่านได้สัญลักษณ์ 1 หมายความว่า สตริงที่ออโตมาตาอ่านเข้ามาลงท้ายด้วย 10110 ซึ่งมีเพียงส่วนหน้าตัวแรกของ α ดังนั้นจึงเปลี่ยนไปอยู่ที่สถานะ p

เมื่ออยู่ที่สถานะ f แล้วแสดงว่าอ่านได้สตริงที่ต้องการแล้วอย่างน้อยหนึ่งครั้ง สัญลักษณ์ที่อ่านเข้ามาต่อจากนั้นจึงไม่ทำให้เปลี่ยนสถานะ นั่นคือที่สถานะ f ไม่ว่าอ่านได้สัญลักษณ์ใดเข้ามาก็ไม่เปลี่ยนสถานะอีกต่อไป

ขั้นตอนวิธีสำหรับตรวจสอบหาสตริงย่อยในข้อความสามารถสร้างจากออโตมาตาจำกัด M ในตัวอย่าง 3.9 แต่หากต้องการให้ตรวจสอบหาสตริงย่อยทั้งหมดในข้อความ จะต้องปรับออโตมาตา M นี้ให้รับข้อความที่ลงท้ายด้วยสตริง α ดังแสดงในตัวอย่าง 3.10 ทุกครั้งที่ออโตมาตา M อยู่ในสถานะ f หมายถึงตำแหน่งของสตริงที่อ่านเข้ามาถึงขณะนั้นมีสตริง α แล้วยังต้องตรวจสอบว่าสตริงที่อ่านต่อไปจะมี α ที่ต้องการหาอีกหรือไม่และหยุดทำงานเมื่ออ่านสตริงที่รับเข้าจนหมด

<u>ตัวอย่าง 3.10</u> กำหนด $\Sigma = \{0, 1\}$ และ $\alpha = 10110$ เราปรับออโตมาตา M ในรูป 3.10 ให้ตรวจสอบว่าสตริงที่รับเข้ามาลงท้ายด้วย α หรือไม่ โดยให้มีการเปลี่ยนสถานะจากสถานะ f เพื่อให้จำว่าส่วนท้ายสุดของสตริงที่อ่านเข้ามาเป็นส่วนหน้าใดของ α นั่นคือ เมื่ออยู่ที่สถานะ f สตริงที่อ่านเข้ามาถึงขณะนั้นลงท้ายด้วย 10110 ดังนั้นถ้าอ่านสัญลักษณ์ถัดมาเป็น 0 จะได้สตริงที่ลงท้ายด้วย 101100 ที่ไม่ลงท้ายด้วยส่วนต้นใดๆ ของ α จึงเปลี่ยนสถานะไปยังสถานะ s และ ถ้าอ่านสัญลักษณ์ถัดมาเป็น 1 จะได้สตริงที่ลงท้ายด้วย 101101 ที่ลงท้ายด้วยส่วนต้นของ α คือ 101 จึงเปลี่ยนสถานะไปยังสถานะ r ดังแสดงในรูป 3.11



รูป 3.10 ออโตมาตาจำกัดที่ตรวจสอบว่าสตริงที่รับเข้ามาลงท้ายด้วย α = 10110 หรือไม่

จากออโตมาตาในรูป 3.11 เราสามารถเขียนโปรแกรมจำลองการทำงานของออโตมาตานี้แล้วให้แสดงตำแหน่งของสตริงย่อยเมื่อออโตมาตาอยู่ในสถานะ f ดังแสดงในรูป 3.12

โปรแกรมนี้ทำงานเหมือนโปรแกรมจำลองการทำงานของออโตมาตาจำกัดในรูป 3.8 เว้นแต่โปรแกรมนี้ตรวจสอบในบรรทัด 16 ว่าออโตมาตาอยู่ในสถานะ `f` หรือไม่ เมื่ออยู่ในสถานะ `f` จะพิมพ์ตำแหน่งเริ่มต้นของ α ดังกำหนดในบรรทัด 17 ของโปรแกรม ดังนั้นจะแสดงตำแหน่งทุกตำแหน่งที่ α ปรากฏในสตริงรับเข้า

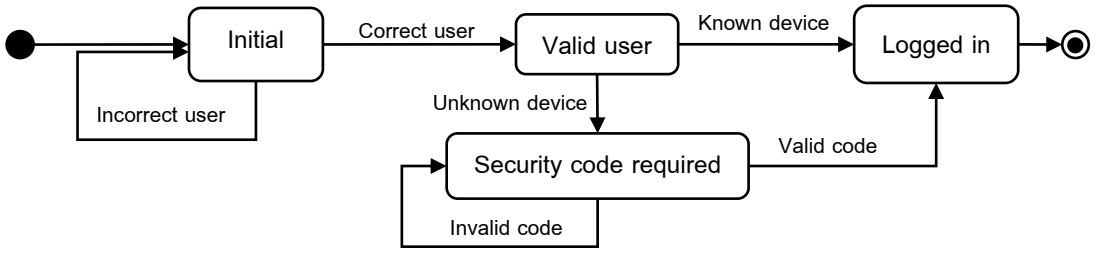
```
01: startState='s'
02: finalStates={'f'}
03: tFunc={                        // transition function
04:        ('s','0'):'s', ('s','1'):'p',
05:        ('p','0'):'q', ('p','1'):'p',
06:        ('q','0'):'s', ('q','1'):'r',
07:        ('r','0'):'q', ('r','1'):'t',
08:        ('t','0'):'f', ('t','1'):'p',
09:        ('f','0'):'s', ('f','1'):'r'}
10: state=startState
11: tape=input()
12: pos=0
13: for sym in tape:
14:     nextState = tFunc[(state, sym)]
15:     state=nextState
16:     if (state in finalStates):
17:        print('Found at', pos-4, '-', pos)
18:     pos=pos+1
```

รูป 3.11 โปรแกรมที่หา α = 10110 ในสตริงที่รับเข้ามา

3.4.3 การอธิบายการทำงานของโปรแกรมในภาษา UML

ภาษา UML ใช้แผนภาพเครื่องจักรสถานะ (state machine diagram) อธิบายการทำงานของโปรแกรม [15] แผนภาพเครื่องจักรสถานะประยุกต์มาจากออโตมาตาจำกัด แผนภาพเครื่องจักรสถานะที่แสดงในรูป 3.13 อธิบายการทำงานของโปรแกรมสำหรับเข้าใช้ระบบด้วยการทวนสอบ 2 ขั้น (2-step verification) การทวนสอบ 2 ขั้นทำงานโดยตรวจสอบว่าชื่อผู้ใช้ตรงกับรหัสผ่านหรือไม่ จากนั้นจึงตรวจสอบว่าเข้าใช้จากอุปกรณ์ที่ได้อนุญาตไว้ก่อนหรือไม่ หากไม่ใช่ ต้องตรวจสอบรหัสความปลอดภัย (security code) ด้วย

เมื่อเราได้ออโตมาตาจำกัดแล้วอาจแปลงออโตมาตาจำกัดนั้นๆ เป็นโปรแกรมโดยตรงหรือเขียนโปรแกรมที่สามารถจำลองการทำงานของออโตมาตาจำกัดเชิงกำหนดใดๆ นอกจากนั้นยังหาได้ว่าลำดับการทำงานใดบ้างที่ทำให้โปรแกรมทำงานจนจบได้ ซึ่งก็คือสตริงที่อยู่ในภาษาที่ออโตมาตายอมรับ



รูป 3.13 แผนภาพเครื่องจักรสถานะสำหรับการเข้าใช้ระบบด้วยการทวนสอบ 2 ขั้น

ข้อสังเกตสำหรับออโตมาตาจำกัดเชิงกำหนด

- ออโตมาตาจำกัดมีจำนวนสถานะเป็นจำนวนจำกัด ข้อมูลบนเทปเป็นสตริงจึงมีความยาวจำกัดด้วย
- หัวเทปของออโตมาตาจำกัดเลื่อนไปทางขวาทีละช่องและไม่ย้อนไปทางซ้าย ดังนั้น สัญลักษณ์ที่อ่านไปแล้วจะไม่มีผลต่อการทำงานของเครื่องในอนาคต ดังนั้นสตริงที่เก็บในโครงแบบจึงไม่รวมสัญลักษณ์ที่อ่านไปแล้ว
- สำหรับออโตมาตาจำกัดที่รวมสถานะตาย จำนวนครั้งของการเปลี่ยนสถานะเท่ากับจำนวนสัญลักษณ์บนเทป เนื่องจากการอ่านสัญลักษณ์หนึ่งตัวทำให้เปลี่ยนสถานะหนึ่งครั้งและออโตมาตาหยุดทำงานเมื่ออ่านสัญลักษณ์สุดท้ายบนเทป
- เซตของสตริงที่ยอมรับด้วยออโตมาตาจำกัดอาจเป็นเซตจำกัดหรือเซตอนันต์
- ถ้าภาษาเป็นเซตจำกัด มีออโตมาตาจำกัดที่ยอมรับภาษานั้น

บทที่ 4

ออโตมาตาจำกัดเชิงไม่กำหนด

บทที่ 4 ออโตมาตาจำกัดเชิงไม่กำหนด

ออโตมาตาจำกัดเชิงไม่กำหนดเป็นออโตมาตาจำกัดซึ่งมีการเปลี่ยนสถานะเชิงไม่กำหนด นั่นคือ เมื่อออโตมาตาจำกัดอยู่ในสถานะหนึ่งและอ่านได้สัญลักษณ์หนึ่งแล้วสถานะต่อไปอาจเป็นได้มากกว่าหนึ่งสถานะ ตัวอย่างเช่น เมื่อเครื่องอยู่ในสถานะ p และอ่านได้สัญลักษณ์ a แล้วสถานะต่อไปอาจเป็นสถานะ q_1 หรือ q_2 หรือ ... หรือ q_n นอกจากนั้นเครื่องเชิงไม่กำหนดอาจเลือกเปลี่ยนสถานะโดยไม่อ่านสัญลักษณ์บนเทป ตัวอย่างเช่น เมื่อเครื่องอยู่ในสถานะ p แล้วมันอาจไม่อ่านสัญลักษณ์ใดเลยและเปลี่ยนไปอยู่ในสถานะ q หรืออยู่ที่สถานะเดิมแล้วอ่านสัญลักษณ์หนึ่งก่อนที่จะเปลี่ยนสถานะต่อไปก็ได้

เมื่อออโตมาตาจำกัดเชิงไม่กำหนดมีทางเลือกเปลี่ยนสถานะได้มากกว่าหนึ่งทาง ออโตมาตาจะเลือกเปลี่ยนสถานะไปทางที่ทำให้ยอมรับสตริงที่อยู่บนเทปหากเป็นไปได้ แต่สำหรับสตริงที่ออโตมาตาไม่ยอมรับ ออโตมาตาต้อง*ไม่มีทางใดเลย*ที่เปลี่ยนสถานะไปจนทำงานจบที่สถานะสิ้นสุดได้

4.1 โครงสร้างของออโตมาตาจำกัดเชิงไม่กำหนด

ออโตมาตาจำกัดเชิงไม่กำหนดมีโครงสร้างคล้ายกับออโตมาตาจำกัดเชิงกำหนด กล่าวคือ ประกอบด้วยเซตของสถานะ อักขระ สถานะเริ่มต้น และ เซตของสถานะสิ้นสุด แต่ฟังก์ชันเปลี่ยนสถานะของออโตมาตาจำกัดเชิงไม่กำหนดเป็นฟังก์ชันที่ให้ค่าเป็นเซตของสถานะ และ ให้สตริงว่าง (ε) เป็นสัญลักษณ์ที่อ่านได้ดังแสดงในนิยามต่อไปนี้

นิยาม 4.1 **ออโตมาตาจำกัดเชิงไม่กำหนด** (nondeterministic finite automata) เป็นเครื่องที่อธิบายด้วย 5-สิ่งอันดับ $M = (Q, \Sigma, \delta, s, F)$ โดยที่

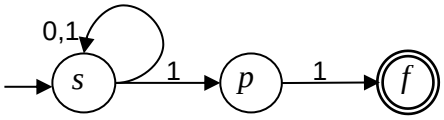
- เซตของสถานะ Q เป็นเซตจำกัด,
- อักขระ Σ เป็นเซตจำกัด,
- สถานะเริ่มต้น s เป็นสถานะหนึ่งใน Q,
- เซตของสถานะสิ้นสุด F ซึ่งเป็นเซตย่อยของ Q, และ
- ฟังก์ชันเปลี่ยนสถานะ δ เป็นฟังก์ชันจาก $Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$

จากนิยามนี้ สังเกตว่าฟังก์ชันเปลี่ยนสถานะ $\delta(q, a) = P$ โดยที่ P เป็นเซตของสถานะ นั่นคือในการเปลี่ยนสถานะแต่ละครั้ง สถานะถัดไปที่เป็นไปได้อาจมีมากกว่าหนึ่งสถานะหรือไม่มีเลย นอกจากนั้น $\delta(q, \varepsilon) = P$ ได้ นั่นคือเครื่องสามารถเปลี่ยนสถานะโดยไม่อ่านสัญลักษณ์ใดๆ บนเทปซึ่ง

เรียกว่า*การเปลี่ยนสถานะด้วยสตริงว่าง* (empty-string transition)

เราจะพิจารณาการเปลี่ยนสถานะที่อ่านสัญลักษณ์ตัวหนึ่งจากเทปและ $\delta(q, a) = P$ โดยที่ a ไม่เป็นสตริงว่างและ P เป็นเซตก่อน นั่นคือ ถ้าให้ $\delta(q, a) = P$ แล้ว P อาจ (1) มีสมาชิกมากกว่าหนึ่งตัว (2) มีสมาชิกหนึ่งตัว หรือ (3) เป็นเซตว่าง ในกรณี (1) ซึ่ง P มีสมาชิกมากกว่าหนึ่งตัว หมายความว่า เมื่อเครื่องอยู่ในสถานะ q และอ่านเทปได้สัญลักษณ์ a แล้วเครื่องอาจจะไปอยู่ในสถานะใดสถานะหนึ่งใน P ก็ได้และเลื่อนหัวเทปไปทางขวาหนึ่งช่อง การตัดสินว่าสถานะถัดไปเป็นสถานะใดเป็นการตัดสินใจเชิงไม่กำหนด นั่นคือเครื่องจะบอกได้ว่ามันต้องเลือกสถานะใดเพื่อให้ไปถึงสถานะสิ้นสุดหากมีทางให้ไปถึงได้และออโตมาตาจำกัดเชิงไม่กำหนดจะบอกได้เองอย่างไม่ผิดพลาด ในกรณี (2) ซึ่ง P มีสมาชิกหนึ่งตัว หมายความว่า เมื่อเครื่องอยู่ในสถานะ q และอ่านได้สัญลักษณ์ a จากเทปแล้วเครื่องจะต้องเปลี่ยนไปอยู่ในสถานะใน P ซึ่งมีเพียงสถานะเดียว ดังนั้นการเปลี่ยนสถานะแบบนี้เหมือนการเปลี่ยนสถานะในออโตมาตาจำกัดเชิงกำหนด ในกรณี (3) ซึ่ง P เป็นเซตว่าง หมายความว่า เมื่อเครื่องอยู่ในสถานะ q และอ่านได้สัญลักษณ์ a จากเทปแล้วเครื่องไม่สามารถเปลี่ยนสถานะได้แต่เครื่องจะหยุดทำงานที่จุดนั้นและไม่ยอมรับสตริงที่ป้อนเข้ามาซึ่งเปรียบได้กับการเปลี่ยนสถานะไปยังสถานะตายของออโตมาตาจำกัดเชิงกำหนด ตัวอย่าง 4.1 แสดงออโตมาตาจำกัดเชิงไม่กำหนดที่มี $\delta(q, a) = P$ โดยที่ a ไม่เป็นสตริงว่างและ P เป็นเซตที่มีสมาชิกมากกว่าหนึ่งตัว

ตัวอย่าง 4.1 กำหนด $M_5 = (\{s, p, f\}, \{0, 1\}, \delta, s, \{f\})$ เป็นออโตมาตาจำกัดเชิงไม่กำหนด โดยที่ฟังก์ชันเปลี่ยนสถานะ δ แสดงในแผนภาพเปลี่ยนสถานะในรูป 4.1 และในตาราง 4.1 เมื่อ M_5 อยู่ในสถานะ s และอ่านได้ 1 จากเทป M_5 อาจเปลี่ยนไปอยู่ในสถานะ p หรืออยู่ในสถานะ s ตามเดิมก็ได้ ซึ่งเป็นการเปลี่ยนสถานะเชิงไม่กำหนด



รูป 4.1 แผนภาพเปลี่ยนสถานะของออโตมาตาจำกัดเชิงไม่กำหนดแผนภาพเปลี่ยนสถานะ (transition diagram) M_5 ในตัวอย่าง 4.1

ตาราง 4.1 ตารางแสดงฟังก์ชันเปลี่ยนสถานะของออโตมาตา M_5 ในตัวอย่าง 4.1

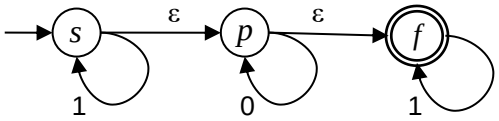
q	a	$\delta(q, a)$
s	0	$\{s\}$
s	1	$\{s, p\}$
p	0	$\{\}$
p	1	$\{f\}$
f	0	$\{\}$
f	1	$\{\}$

จากตาราง 4.1 $\delta(p, 0) = \{\}$, $\delta(f, 0) = \{\}$ และ $\delta(f, 1) = \{\}$ ดังแสดงในแถวที่แรเงา เราอาจจะไม่แสดงการเปลี่ยนสถานะเหล่านี้ในตารางแสดงฟังก์ชันเปลี่ยนสถานะก็ได้ ในกรณีเหล่านี้เมื่อเครื่องไม่สามารถเปลี่ยนสถานะได้ (เช่น เมื่อ M_5 อยู่ในสถานะ p และอ่านได้สัญลักษณ์ 0) เครื่องจะหยุดทำงานและไม่ยอมรับสตริงบนเทป

ในกรณีที่ออโตมาตาจำกัดอ่านสตริงบนเทปได้จนหมดแล้วมันจะหยุดทำงาน (ถ้าเครื่องหยุดทำงานในสถานะสิ้นสุดสถานะหนึ่งใน F แล้วเครื่องจะยอมรับสตริงบนเทป แต่ถ้าเครื่องหยุดทำงานในสถานะที่ไม่ใช่สิ้นสุด แล้วเครื่องจะไม่ยอมรับสตริงนั้นเช่นเดียวกับออโตมาตาจำกัดเชิงกำหนด)

ต่อไปเราจะพิจารณาการเปลี่ยนสถานะโดยใช้สตริงว่าง นั่นคือถ้าให้ $\delta(q, \varepsilon) = P$ โดยที่ P ไม่เป็นเซตว่างแล้วออโตมาตาจำกัดเชิงไม่กำหนดสามารถเปลี่ยนสถานะจากสถานะ q ไปอยู่ในสถานะใดสถานะหนึ่งใน P ได้โดยไม่ต่องอ่านสัญลักษณ์บนเทปและไม่เลื่อนหัวเทป ตัวอย่าง 4.2 แสดงตัวอย่างของออโตมาตาจำกัดเชิงไม่กำหนดที่มีการเปลี่ยนสถานะโดยใช้สตริงว่าง

ตัวอย่าง 4.2 กำหนด $M_6 = (\{s, p, f\}, \{0, 1\}, \delta, s, \{f\})$ เป็นออโตมาตาจำกัดเชิงไม่กำหนด โดยที่ฟังก์ชันเปลี่ยนสถานะ δ แสดงในแผนภาพเปลี่ยนสถานะในรูป 4.2 จาก $\delta(s, \varepsilon) = \{p\}$ เมื่อ M_6 อยู่ในสถานะ s มันสามารถเลือกเปลี่ยนสถานะไปยังสถานะ p หรืออยู่ในสถานะ s เช่นเดิมก็ได้ ทั้งนี้เมื่อ M_6 จะเปลี่ยนสถานะที่จุดนี้มันไม่ต้องอ่านสัญลักษณ์ใดๆ จากเทปเลย ทำนองเดียวกันจาก $\delta(p, \varepsilon) = \{f\}$ เมื่อ M_6 อยู่ในสถานะ p มันอาจเลือกเปลี่ยนสถานะไปยังสถานะ f โดยไม่ต้องอ่านสัญลักษณ์ใดๆ จากเทปเลยหรืออยู่ในสถานะ p เช่นเดิมก็ได้



รูป 4.2 แผนภาพเปลี่ยนสถานะของ M_6 ในตัวอย่าง 4.2

ในทำนองเดียวกับออโตมาตาจำกัดเชิงกำหนด สิ่งที่ต้องระบุเพื่ออธิบายสถานภาพในขณะใดขณะหนึ่งของออโตมาตาจำกัดไม่เชิงกำหนดประกอบด้วยสถานะของส่วนควบคุมและสตริงบนเทปที่ยังไม่ได้อ่านในขณะนั้น ดังนั้นสถานภาพในขณะหนึ่งของออโตมาตาจำกัดเชิงไม่กำหนดอธิบายได้ด้วยโครงแบบ (configuration) เช่นเดียวกับโครงแบบของออโตมาตาจำกัดเชิงกำหนด นั่นคือโครงแบบเป็นคู่อันดับ (q, α) โดยที่ q เป็นสถานะของส่วนควบคุมและ α เป็นสตริงบนเทปที่ยังไม่ได้อ่าน

นอกจากนั้นการทำงานของออโตมาตาจำกัดเชิงไม่กำหนดอธิบายได้โดยแสดงลำดับการเปลี่ยนโครงแบบตั้งแต่เครื่องเริ่มต้นทำงานจนหยุดทำงาน การเปลี่ยนโครงแบบของออโตมาตาจำกัดเชิงไม่กำหนดนิยามได้ดังต่อไปนี้

นิยาม 4.2 กำหนด $M = (Q, \Sigma, \delta, s, F)$ เป็นออโตมาตาจำกัดเชิงไม่กำหนด, q_1, q_2 เป็นสถานะใน Q และ α_1, α_2 เป็นสตริงใน Σ^* โครงแบบ (q_1, α_1) **เปลี่ยนเป็นโครงแบบ** (q_2, α_2) **ในหนึ่งขั้น** (yield in one step) (ซึ่งสามารถเขียนแทนว่า $(q_1, \alpha_1) \vdash_M (q_2, \alpha_2)$) ถ้า

- $\delta(q_1, a) = P$ โดยที่ $q_2 \in P$ และ $\alpha_1 = a \cdot \alpha_2$ สำหรับ a บางตัวใน Σ หรือ
- $\delta(q_1, \varepsilon) = P$ โดยที่ $q_2 \in P$ และ $\alpha_1 = \alpha_2$

จากนิยาม 4.2 จะเห็นว่าถ้า $\delta(q_1, a) = \{p_1, p_2\}$ แล้ว $(q_1, a \cdot \alpha) \vdash_M (p_1, \alpha)$ หรือ $(q_1, a \cdot \alpha) \vdash_M (p_2, \alpha)$ ก็ได้ เราสามารถอธิบายการทำงานของออโตมาตาจำกัดเชิงไม่กำหนดได้สองวิธี คือ แบบที่หนึ่งเป็นการลองทุกทางเลือกแบบขนานกัน และ แบบที่สองเป็นการเดาทางเลือกที่ถูกต้องเสมอ

สำหรับแบบที่หนึ่ง เรามองการทำงานของออโตมาตาจำกัดเชิงไม่กำหนดเป็นเหมือนการทำงานของออโตมาตาจำกัดเชิงกำหนดที่มีการสร้างออโตมาตาจำกัดเลียนแบบตัวเองขึ้นมาทุกครั้งที่มีทางเลือกในการทำงานมากกว่าหนึ่งทาง แล้วให้เครื่องเหล่านี้ทำงานไปพร้อมๆ กัน หากมีเครื่อง*แม้เพียงหนึ่งเครื่อง*ในชุดนี้ยอมรับสตริงบนเทปแล้วหมายความว่าออโตมาตาจำกัดเชิงไม่กำหนดยอมรับสตริงนั้น หากว่าเครื่องเลียนแบบที่สร้างขึ้นมา*ทุกเครื่อง*ไม่ยอมรับสตริงบนเทปแล้วหมายความว่าออโตมาตาจำกัดเชิงไม่กำหนดไม่ยอมรับสตริงนั้นเช่นกัน

สำหรับแบบที่สอง เรามองการทำงานของออโตมาตาจำกัดเชิงไม่กำหนดว่าใช้การตัดสินใจเชิงไม่กำหนด คือ เมื่อต้องตัดสินใจว่าจะต้องการเปลี่ยนสถานะแบบใด (กรณีที่มีทางเลือกมากกว่าหนึ่งแบบ) ออโตมาตาจำกัดจะสามารถ*เดา*ได้ว่าทางเลือกใดจะทำให้เครื่องยอมรับสตริงบนเทปได้อย่างถูกต้อง (ถ้ามี) แล้วออโตมาตาจำกัดจะเลือกทางนั้น เมื่อมองการทำงานของออโตมาตาจำกัดเชิงไม่กำหนดว่าใช้การเดาแล้วอาจทำให้การสร้างออโตมาตาจำกัดเชิงไม่กำหนดง่ายขึ้นได้

ตัวอย่าง 4.3 แสดงการเปลี่ยนโครงแบบของเครื่อง M_5 และ M_6 ที่อธิบายในตัวอย่าง 4.2

ตัวอย่าง 4.3 จากตัวอย่าง 4.1 เมื่อให้สตริงบนเทปของ M_5 เป็น 10011 การทำงานของ M_5 อธิบายตามแบบที่หนึ่งได้ด้วยต้นไม้ที่แสดงในรูป 4.3 (a) ซึ่งการแตกกิ่งแต่ละครั้งเป็นการสร้างเครื่องเลียนแบบเครื่องเดิมขึ้นมา นอกจากนั้นเมื่อให้สตริงบนเทปของเครื่อง M_5 เป็น 0010 การทำงานของ M_5 อธิบายตามแบบที่หนึ่งได้ด้วยต้นไม้ที่แสดงในรูป 4.3 (b)

การทำงานของ M_5 ในรูป 4.3 (a) อธิบายตามแบบที่สองได้โดยเลือกเส้นทางในต้นไม้ที่ทำให้มาถึงโครงแบบที่เป็นการยอมรับสตริงบนเทปดังนี้

$$(s, 10011) \vdash_{M_5} (s, 0011) \vdash_{M_5} (s, 011) \vdash_{M_5} (s, 11) \vdash_{M_5} (p, 1) \vdash_{M_5} (f, \varepsilon)$$

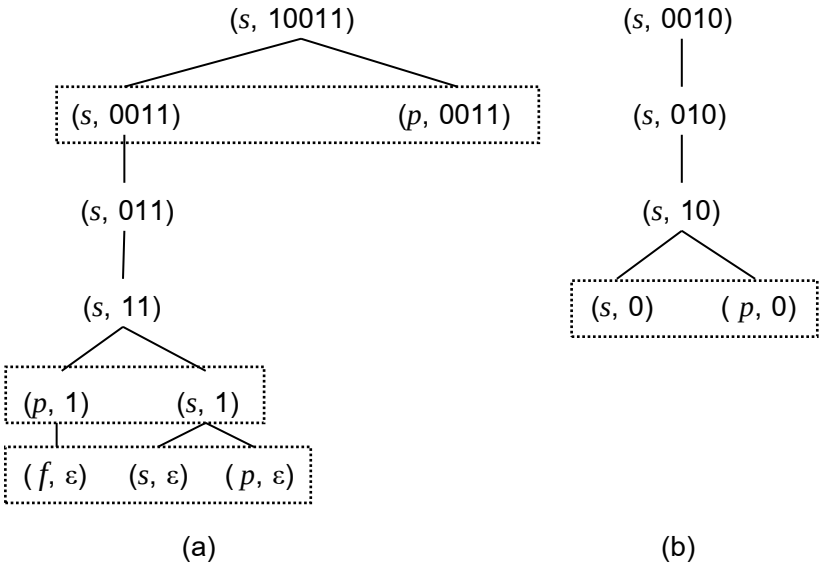
หากเปรียบเทียบการเปลี่ยนโครงแบบ $(s, 10011) \vdash_{M_5} (s, 0011)$ และ $(s, 11) \vdash_{M_5} (p, 1)$ จะเห็นว่าเมื่อ M_5 อ่านได้ 1 แล้วอาจเปลี่ยนจากสถานะ s ไปอยู่ในสถานะ s หรือสถานะ p ซึ่งเปรียบเสมือน

ว่า M_5 เลือกตัดสินใจได้ถูกว่าต้องเลือกสถานะใดเมื่อไร

จากรูป 4.3 (b) ทางเลือกในต้นไม้ทั้งสองทางไม่ทำให้ M_5 ไปถึงสถานะสิ้นสุดได้ ในกรณีนี้เราอธิบายการทำงานของ M_5 ตามแบบที่สองได้สองทางดังนี้

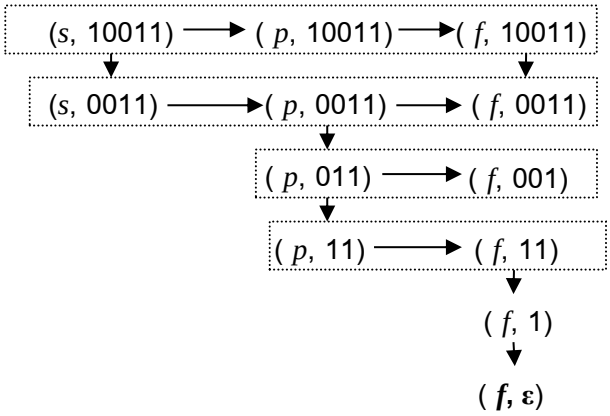
$(s, 0010) \vdash_{M_5} (s, 010) \vdash_{M_5} (s, 10) \vdash_{M_5} (s, 0)$

$(s, 0010) \vdash_{M_5} (s, 010) \vdash_{M_5} (s, 10) \vdash_{M_5} (p, 0)$



รูป 4.3 การอธิบายการเปลี่ยนโครงแบบของออโตมาตาจำกัดเชิงไม่กำหนด M_5 โดยใช้ต้นไม้

จากตัวอย่าง 4.2 ถ้าให้สตริงบนเทปของ M_6 เป็น 10011 การทำงานของ M_6 อธิบายตามแบบที่หนึ่งได้ด้วยต้นไม้ที่แสดงในรูป 4.4 ซึ่งการแตกกิ่งแต่ละครั้งเป็นการสร้างเครื่องเลียนแบบเครื่องเดิมขึ้นมาและการแตกกิ่งที่ให้โหนดที่อยู่ในบรรทัดเดียวกันเป็นการเปลี่ยนโครงแบบโดยใช้สตริงว่าง การเปลี่ยนโครงแบบของ M_6 แสดงได้ดังนี้



รูป 4.4 ต้นไม้แสดงการเปลี่ยนโครงแบบของออโตมาตาจำกัดเชิงไม่กำหนด M_6

การทำงานของ M_6 อธิบายตามแบบที่สองโดยเลือกเส้นทางที่อ่านเทปหมดได้ดังนี้

$(s, 10011) \vdash_{M_6} (s, 0011) \vdash_{M_6} (p, 0011) \vdash_{M_6} (p, 011) \vdash_{M_6} (p, 11) \vdash_{M_6} (f, 11) \vdash_{M_6} (f, 1) \vdash_{M_6} (f, \varepsilon)$

หากพิจารณาการเปลี่ยนโครงแบบ $(s, 10011) \vdash_{M_6} (s, 0011)$ และ $(s, 0011) \vdash_{M_6} (p, 0011)$ จะเห็นว่า M_6 เลือกตัดสินใจได้ถูกว่าในกรณีแรกต้องอ่าน 1 เพื่อเปลี่ยนสถานะ แต่ในกรณีหลังต้องเลือกเปลี่ยนสถานะด้วยสตริงว่าง

สำหรับออโตมาตาจำกัดเชิงไม่กำหนด การเปลี่ยนโครงแบบในศูนย์ขั้นหรือมากกว่านิยามได้เหมือนกับออโตมาตาจำกัดเชิงกำหนดดังนี้

นิยาม 4.3 กำหนด $M = (Q, \Sigma, \delta, s, F)$ เป็นออโตมาตาจำกัดเชิงไม่กำหนด, q_1, q_2 เป็นสถานะใน Q และ α_1, α_2 เป็นสตริงใน Σ^* โครงแบบ (q_1, α_1) **เปลี่ยนเป็นโครงแบบ** (q_2, α_2) **ในศูนย์ขั้นหรือมากกว่า** (yield in zero step or more) ซึ่งสามารถเขียนแทนว่า $(q_1, \alpha_1) \vdash^*_M (q_2, \alpha_2)$ ถ้า $(q_1, \alpha_1) = (q_2, \alpha_2)$ หรือมีโครงแบบ (q, α) ของ M อย่างน้อยหนึ่งโครงแบบที่ $(q_1, \alpha_1) \vdash^*_M (q, \alpha)$ และ $(q, \alpha) \vdash_M (q_2, \alpha_2)$

จากนิยาม 4.3 นี้ เราสามารถระบุการเปลี่ยนโครงแบบของ M_5 และ M_6 จากโครงแบบเริ่มต้นไปเป็นโครงแบบสิ้นสุดในตัวอย่าง 4.3 ได้ดังนี้

$(s, 10011) \vdash^*_{M_5} (f, \varepsilon)$

$(s, 10011) \vdash^*_{M_6} (f, \varepsilon)$

เช่นเดียวกับออโตมาตาจำกัดเชิงกำหนด เราสามารถพิสูจน์ได้ด้วยว่า สำหรับออโตมาตาจำกัดเชิงไม่กำหนด $M = (Q, \Sigma, \delta, s, F)$ ใดๆ ถ้า $(q, \alpha) \vdash^*_M (r, \beta)$ สำหรับสถานะ q และ r ใดๆ ใน Q และสตริง α และ β ใดๆ ใน Σ^* แล้ว $(q, \alpha\cdot\gamma) \vdash^*_M (r, \beta\cdot\gamma)$ สำหรับสตริง γ ใดๆ ใน Σ^* นอกจากนั้นการยอมรับสตริงโดยออโตมาตาจำกัดเชิงไม่กำหนดนิยามได้เช่นเดียวกับการยอมรับสตริงโดยเครื่องเชิงกำหนดดังนี้

นิยาม 4.4 กำหนด $M = (Q, \Sigma, \delta, s, F)$ เป็นออโตมาตาจำกัดเชิงไม่กำหนด, ω เป็นสตริงใน Σ^* M **ยอมรับ** (accept) สตริง ω ถ้า $(s, \omega) \vdash^*_M (f, \varepsilon)$ สำหรับบางสถานะ f ใน F และ M **ไม่ยอมรับ** (reject) สตริง ω ถ้าไม่มีสถานะ q ใดใน F ที่ $(s, \omega) \vdash^*_M (q, \varepsilon)$

สังเกตว่าการนิยามสตริงที่ไม่ยอมรับด้วยเครื่องเชิงไม่กำหนดจะต่างจากสตริงที่ไม่ยอมรับด้วยเครื่องเชิงกำหนด ภาษาที่ออโตมาตาจำกัดเชิงไม่กำหนดยอมรับคือเซตของสตริงที่ออโตมาตานั้นยอมรับซึ่งสามารถนิยามได้เช่นเดียวกับภาษาที่ออโตมาตาจำกัดเชิงกำหนดยอมรับดังนี้

นิยาม 4.5 กำหนด $M = (Q, \Sigma, \delta, s, F)$ เป็นออโตมาตาจำกัดเชิงไม่กำหนด, L เป็นภาษาบน Σ M **ยอมรับ** (accept) ภาษา L ถ้า M ยอมรับ (accept) สตริง ω สำหรับทุกสตริง ω ใน L นั่นคือ $L(M) = \{\omega \in \Sigma^* \mid (s, \omega) \mid\!-^*_M (f, \varepsilon)$ สำหรับบางสถานะ f ใน $F\}$

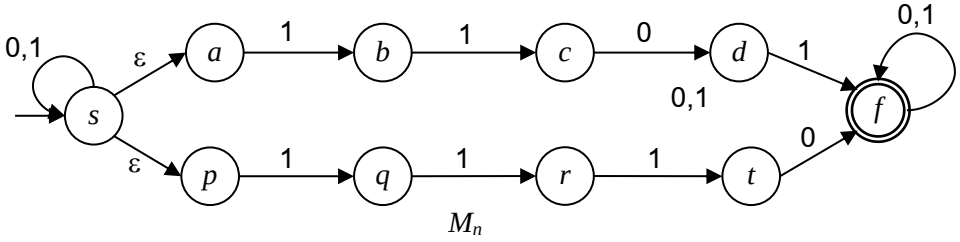
เราสามารถระบุได้ว่าออโตมาตาจำกัดเชิงไม่กำหนดยอมรับภาษาใดตามนิยาม 4.5 ดังแสดงในตัวอย่าง 4.4

ตัวอย่าง 4.4 จาก M_5 ในตัวอย่าง 4.1 และ ตัวอย่าง 4.3 จะเห็นว่า 10011 $\in L(M_5)$ เพราะ $(s, 10011) \mid\!-^*_{M_5} (f, \varepsilon)$ แต่ 0010 $\notin L(M_5)$ เพราะ $(s, 0010) \mid\!-^*_{M_5} (s, 0)$ ภาษาที่ M_5 ยอมรับประกอบด้วยสตริงใน $\{0, 1\}^*$ ที่ลงท้ายด้วย 11 นั่นคือ $L(M_5) = \{\omega \in \{0,1\}^* \mid \omega$ ลงท้ายด้วย $11\}$
จาก M_6 ในตัวอย่าง 4.2 และ ตัวอย่าง 4.3 จะเห็นว่า 10011 $\in L(M_6)$ เพราะ $(s, 10011) \mid\!-^*_{M_6} (f, \varepsilon)$ ภาษาที่ M_6 ยอมรับประกอบด้วยสตริงใน $\{0, 1\}^*$ ที่มี 10 และ 01 ในสตริงนั้นอย่างละไม่เกินหนึ่งครั้ง และ 10 ต้องมาก่อน 01 นั่นคือ $L(M_6) = \{\omega \in \{0,1\}^* \mid$ 10 และ 01 อยู่ใน ω อย่างละไม่เกินหนึ่งครั้ง และถ้ามีทั้ง 10 และ 01 แล้ว 10 ต้องมาก่อน $01\}$

บางครั้งการสร้างออโตมาตาจำกัดเชิงไม่กำหนดอาจง่ายกว่าการสร้างออโตมาตาจำกัดเชิงกำหนด ตัวอย่างต่อไปนี้แสดงออโตมาตาจำกัดเชิงไม่กำหนดที่ยอมรับที่กำหนดซึ่งจะง่ายกว่าการสร้างออโตมาตาจำกัดเชิงกำหนด

ตัวอย่าง 4.5 กำหนดอักขระ $\Sigma = \{0, 1\}$ ภาษา $L = \{\omega \in \{0,1\}^* \mid$ 1011 หรือ 1110 เป็นสตริงย่อยใน $\omega\}$ จากตัวอย่าง 3.8

ออโตมาตาจำกัดเชิงไม่กำหนด M_n ที่แสดงในรูป 4.5 ยอมรับภาษา L เครื่อง M_n ใช้การเปลี่ยนสถานะด้วยสตริงว่างที่สถานะ s เลือกว่าจะเริ่มตรวจสอบสตริงย่อยเมื่อไร และใช้การเปลี่ยนสถานะเชิงไม่กำหนดที่สถานะ s เลือกว่าจะเปลี่ยนไปยังสถานะ a หรือ p เพื่อตรวจสอบสตริงย่อย 1011 หรือ 1110 ทั้งนี้แนวคิดในการสร้าง M_n ง่ายกว่าการสร้างออโตมาตาจำกัดเชิงกำหนดในตัวอย่าง 3.8 ซึ่งต้องตรวจสอบจำส่วนย่อยของสตริงที่ต้องการหาทั้งหมด



รูป 4.5 ออโตมาตาจำกัดเชิงไม่กำหนดที่ยอมรับภาษา L

สิ่งที่ควรระวังในการสร้างออโตมาตาจำกัดเชิงไม่กำหนด คือ เราต้องแน่ใจว่าเมื่อออโตมาตาไม่ยอมรับสตริงใด ออโตมาตาต้องไม่มีทางใดเลยที่ทำให้เปลี่ยนสถานะไปจนจบการทำงานที่สถานะสิ้นสุดสำหรับสตริงนั้น

การเพิ่มการทำงานเชิงไม่กำหนดให้กับออโตมาตาจำกัดอาจทำให้การสร้างออโตมาตาจำกัดง่ายขึ้น แต่ยังไม่อาจสรุปได้ว่ามันจะทำให้ออโตมาตาจำกัดมีความสามารถเพิ่มขึ้นด้วยหรือไม่ หัวข้อต่อไปนี้เป็นการพิสูจน์ว่าออโตมาตาจำกัดเชิงกำหนดและออโตมาตาจำกัดเชิงไม่กำหนดมีความสามารถต่างกันหรือไม่

4.2 การจำลองการทำงานของออโตมาตาจำกัดเชิงไม่กำหนด

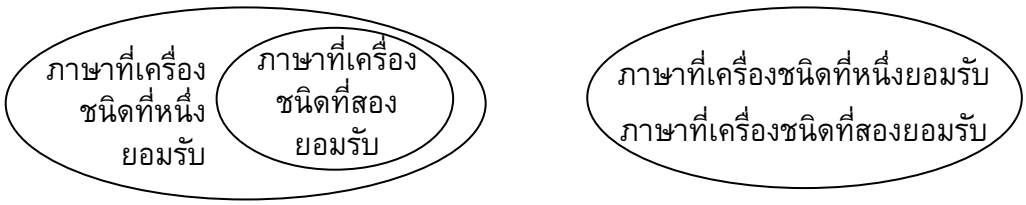
เนื่องจากการทำงานของโปรแกรมคอมพิวเตอร์เป็นแบบเชิงกำหนด คือ โปรแกรมต้องตัดสินใจว่าการทำงานคำสั่งถัดไปคือคำสั่งใดคำสั่งหนึ่งทางเดียวเท่านั้น ดังนั้นเราจึงต้องจำลองการทำงานเชิงไม่กำหนดโดยการทำงานเชิงกำหนด ในหัวข้อ 4.2.1 เราจะแสดงว่าออโตมาตาจำกัดเชิงกำหนดมีความสามารถเท่ากับออโตมาตาจำกัดเชิงไม่กำหนด จากนั้นเราใช้วิธีการที่พิสูจน์ในหัวข้อ 4.2.1 มาใช้ในการเขียนโปรแกรมจำลองการทำงานของออโตมาตาจำกัดเชิงไม่กำหนดในหัวข้อ 4.2.2

4.2.1 การเปรียบเทียบออโตมาตาจำกัดเชิงกำหนดและออโตมาตาจำกัดเชิงไม่กำหนด

หัวข้อนี้เป็นการเปรียบเทียบความสามารถของออโตมาตาจำกัดเชิงกำหนดและออโตมาตาจำกัดเชิงไม่กำหนด เราจะกล่าวว่าเครื่องชนิดที่หนึ่ง**มีความสามารถมากกว่า**เครื่องชนิดที่สอง ถ้า (1) เครื่องชนิดที่หนึ่งแก้ปัญหาทุกปัญหา (หรือยอมรับภาษาทุกภาษา) ที่เครื่องชนิดที่สองแก้ (หรือยอมรับ) ได้ (2) แต่มีปัญหาบางปัญหา (หรือภาษาบางภาษา) ที่เครื่องชนิดที่หนึ่งแก้ (หรือยอมรับ) ได้แต่เครื่องชนิดที่สองแก้ (หรือยอมรับ) ไม่ได้ดังแสดงในรูป 4.6 (a) เราจะกล่าวว่าเครื่องชนิดที่หนึ่ง**มีความสามารถเท่ากับ**ชนิดที่สอง ถ้า (1) เครื่องชนิดที่หนึ่งแก้ปัญหาทุกปัญหา (หรือยอมรับภาษาทุกภาษา) ที่เครื่องชนิดที่สองแก้ (หรือยอมรับ) ได้ และ (2) ในทางกลับกัน เครื่องชนิดที่สองแก้ปัญหาทุกปัญหา (หรือยอมรับภาษาทุกภาษา) ที่เครื่องชนิดที่หนึ่งแก้ (หรือยอมรับ) ได้ดังแสดงในรูป 4.6 (b) สังเกตว่าการเปรียบเทียบความสามารถของเครื่องที่กล่าวถึงในการศึกษาเกี่ยวกับออโตมาตานี้ใช้ความสามารถในการแก้ปัญหาเป็นตัววัดโดยไม่สนใจประสิทธิภาพ (เช่น เนื้อที่หรือเวลาที่ใช้)

ในการเปรียบเทียบออโตมาตาจำกัดเชิงกำหนดและออโตมาตาจำกัดเชิงไม่กำหนดนี้ ขั้นแรกเราจะพิจารณาสิ่งที่ออโตมาตาจำกัดเชิงกำหนดทำได้แล้วตรวจสอบว่ามีออโตมาตาจำกัดเชิงไม่กำหนดที่ทำสิ่งนั้นได้หรือไม่ นั่นคือเราพิจารณาทุกภาษาที่ออโตมาตาจำกัดเชิงกำหนดยอมรับ (หรืออาจพูดได้ว่า

พิจารณาออโตมาตาจำกัดเชิงกำหนดทุกเครื่อง) แล้วหาออโตมาตาจำกัดเชิงไม่กำหนดที่ยอมรับภาษานั้น ถ้าหาได้แล้วเราจะสรุปได้ว่าออโตมาตาจำกัดเชิงไม่กำหนดมีความสามารถไม่น้อยกว่าออโตมาตาจำกัดเชิงกำหนด สำหรับขั้นที่สอง เราจะพิสูจน์ว่าออโตมาตาจำกัดเชิงกำหนดมีความสามารถไม่น้อยกว่าออโตมาตาจำกัดเชิงไม่กำหนดโดยพิจารณาสิ่งที่ออโตมาตาจำกัดเชิงไม่กำหนดทำได้ (หรืออาจพูดได้ว่าพิจารณาออโตมาตาจำกัดเชิงไม่กำหนดทุกเครื่อง) แล้วตรวจสอบว่าออโตมาตาจำกัดเชิงกำหนดทำสิ่งนั้นได้หรือไม่ ซึ่งสามารถทำได้ในทำนองเดียวกับขั้นแรก



(a) เครื่องชนิดที่หนึ่งมีความสามารถมากกว่าเครื่องชนิดที่สอง

(b) เครื่องชนิดที่หนึ่งมีความสามารถเท่ากับเครื่องชนิดที่สอง

รูป 4.6 การเทียบความสามารถของเครื่องสองชนิด

ทฤษฎีต่อไปนี้พิสูจน์ว่าออโตมาตาจำกัดเชิงไม่กำหนดมีความสามารถไม่น้อยกว่าออโตมาตาจำกัดเชิงกำหนดโดยแสดงวิธีสร้างออโตมาตาจำกัดเชิงไม่กำหนดที่ทำงานเลียนแบบออโตมาตาจำกัดเชิงกำหนดใดๆ อย่างไรก็ตามในทฤษฎีนี้เราจะแสดงวิธีพิสูจน์แบบไม่เคร่งครัดเพราะวิธีสร้างเครื่องที่ใช้ในทฤษฎีนี้ง่ายมาก แต่เราจะแสดงวิธีพิสูจน์แบบเคร่งครัดสำหรับขั้นที่สองซึ่งซับซ้อนกว่า

ทฤษฎี 4.1 สำหรับออโตมาตาจำกัดเชิงกำหนด M_d ใดๆ มีออโตมาตาจำกัดเชิงไม่กำหนดที่ยอมรับ $L(M_d)$

พิสูจน์: กำหนด $M_d = (Q, \Sigma, \delta, s, F)$ เป็นออโตมาตาจำกัดเชิงกำหนดใดๆ เราจะสร้างออโตมาตาจำกัดเชิงไม่กำหนด M_n แล้วแสดงว่า $L(M_n) = L(M_d)$

เราจะให้ $M_n = (Q, \Sigma, \Delta, s, F)$ โดยที่ $\delta(q, a) = p \leftrightarrow \Delta(q, a) = \{p\}$ จาก Δ ใน M_n ที่กำหนดเราจะเห็นได้ว่าแผนภาพเปลี่ยนสถานะของ M_n และ M_d เหมือนกัน นั่นคือ M_n และ M_d ทำงานเหมือนกันทุกอย่าง ดังนั้นสำหรับสตริง ω ใดๆ ใน Σ^* $\omega \in L(M_n) \leftrightarrow \omega \in L(M_d)$ หรือ $L(M_n) = L(M_d)$ นั่นเอง

ขั้นต่อไปของการเปรียบเทียบออโตมาตาจำกัดเชิงกำหนดและและออโตมาตาจำกัดเชิงไม่กำหนดเป็นการพิสูจน์ว่าออโตมาตาจำกัดเชิงกำหนดมีความสามารถไม่น้อยกว่าออโตมาตาจำกัดเชิงไม่กำหนดโดยแสดงว่า มีออโตมาตาจำกัดเชิงกำหนดที่ทำงานเลียนแบบออโตมาตาจำกัดเชิงไม่กำหนด การพิสูจน์นี้คล้ายกับการพิสูจน์ทฤษฎี 4.1 แต่วิธีสร้างออโตมาตาจำกัดเชิงกำหนดที่ทำงานเลียนแบบออโตมาตาจำกัดเชิงไม่กำหนดจะยุ่งยากกว่า เราพิจารณาการเปลี่ยนสถานะเชิงไม่กำหนดสองแบบที่

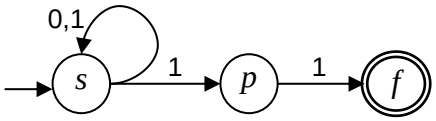
กล่าวมาแล้ว คือการเลือกเปลี่ยนสถานะได้มากกว่าหนึ่งทางและการเลือกเปลี่ยนสถานะด้วยสตริงว่าง

เมื่อออโตมาตาจำกัดเลือกเปลี่ยนสถานะได้มากกว่าหนึ่งทาง โครงแบบถัดไปมีได้มากกว่าหนึ่งโครงแบบ โครงแบบเหล่านี้จะมีสตริงที่ยังไม่ถูกอ่านบนเทปเหมือนกันแต่สถานะต่างกันดังแสดงในโครงแบบที่อยู่ในกล่องเดียวกันในรูป 4.3 อย่างไรก็ตามสถานะที่เป็นไปได้ในโครงแบบถัดไปมีจำนวนจำกัด คือไม่เกิน $2^{|Q|}$ ดังนั้นเราสามารถสร้างออโตมาตาจำกัดเชิงกำหนดที่ทำงานเลียนแบบออโตมาตาจำกัดเชิงไม่กำหนดโดยจำสถานะของเครื่องเชิงไม่กำหนดได้ดังแสดงในตัวอย่างต่อไปนี้

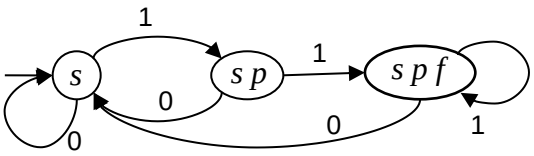
ตัวอย่าง 4.6 จากตัวอย่าง 4.1 ออโตมาตาจำกัด M_5 แสดงไว้ในรูป 4.7 (a) มีการตัดสินใจเชิงไม่กำหนดได้ เช่น เมื่ออ่านได้ 1 ที่สถานะ s เครื่องจะเปลี่ยนไปสู่สถานะ s หรือ q ดังนั้นออโตมาตาจำกัดเชิงกำหนดที่ทำงานเลียนแบบมันต้องจำสถานะทั้งสองนี้ไว้ สถานะที่ออโตมาตาจำกัดเชิงกำหนดต้องจำและการเปลี่ยนสถานะแสดงไว้ในรูป 4.7 (b) สัญลักษณ์ที่แสดงในแต่ละสถานะในรูป 4.7 (b) เป็นสถานะของ M_5 ที่ออโตมาตาจำกัดเชิงกำหนดต้องจำไว้ การทำงานของออโตมาตาจำกัดเชิงกำหนดจะเป็นดังนี้

- เมื่อเริ่มต้นทำงาน ออโตมาตาจำกัดเชิงกำหนดต้องจำว่า M_5 อยู่ในสถานะ s ดังนั้นออโตมาตาจำกัดเชิงกำหนดต้องจำสถานะ s ไว้
- จากสถานะ s ถ้า M_5 อ่านได้ 1 สถานะถัดไปของมันอาจเป็นสถานะ s หรือ p ก็ได้ (ซึ่งไม่อาจบอกได้จนกว่าเครื่องจะอ่านสตริงบนเทปจนหมด) ดังนั้นเมื่อออโตมาตาจำกัดเชิงกำหนดอ่านได้ 1 มันจะเปลี่ยนสถานะไปยังสถานะที่จำสถานะ s และ p ไว้
- จากสถานะ s ถ้า M_5 อ่านได้ 0 สถานะถัดไปของมันเป็นสถานะ s ได้เท่านั้น ดังนั้นเมื่อออโตมาตาจำกัดเชิงกำหนดอ่านได้ 0 มันจะเปลี่ยนสถานะไปยังสถานะ s
- จากสถานะ s และ p ถ้า M_5 อ่านได้ 1 สถานะถัดไปของมันเป็นสถานะ s หรือ p หรือ f ก็ได้ เพราะมันอาจเปลี่ยนสถานะจาก s เป็น s หรือ p ได้และอาจเปลี่ยนสถานะจาก p เป็น f ได้ ดังนั้นเมื่อออโตมาตาจำกัดเชิงกำหนดอ่านได้ 1 มันจะเปลี่ยนสถานะไปยังสถานะที่จำสถานะ s, p และ f ไว้
- จากสถานะ s และ p ถ้า M_5 อ่านได้ 0 สถานะถัดไปของมันเป็นสถานะ s ได้เท่านั้น เพราะมันอาจเปลี่ยนสถานะจาก s เป็น s ได้ แต่ไม่อาจเปลี่ยนสถานะจาก p ไปได้ ดังนั้นเมื่อออโตมาตาจำกัดเชิงกำหนดอ่านได้ 0 มันจะเปลี่ยนสถานะไปยังสถานะ s
- จากสถานะ s, p และ f ถ้า M_5 อ่านได้ 1 สถานะถัดไปของมันเป็นสถานะ s หรือ p หรือ f ก็ได้ เพราะมันอาจเปลี่ยนสถานะจาก s เป็น s หรือ p ได้และอาจเปลี่ยนสถานะจาก p เป็น f ได้ แต่เปลี่ยนสถานะจาก f ไปไม่ได้ ดังนั้นเมื่อออโตมาตาจำกัดเชิงกำหนดอ่านได้ 1 มันจะเปลี่ยนสถานะไปยังสถานะที่จำสถานะ s, p และ f ไว้

- จากสถานะ s, p และ f ถ้า M_5 อ่านได้ 0 สถานะถัดไปของมันเป็นสถานะ s เพราะมันอาจเปลี่ยนสถานะจาก s เป็น s แต่ไม่อาจเปลี่ยนสถานะจาก p หรือ f ไปได้ ดังนั้นเมื่อออโตมาตาจำกัดเชิงกำหนดอ่านได้ 0 มันจะเปลี่ยนสถานะไปยังสถานะที่จำสถานะ s, p และ f ไว้



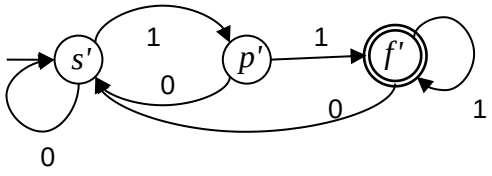
(a) การเปลี่ยนสถานะของออโตมาตาจำกัดเชิงไม่กำหนด M_5



(b) การเปลี่ยนสถานะของออโตมาตาจำกัดเชิงกำหนดที่เลียนแบบ M_5

รูป 4.7 การสร้างออโตมาตาจำกัดเชิงกำหนดที่เลียนแบบ M_5

จากที่อธิบายมาข้างต้น รูป 4.7 (b) แสดงการเปลี่ยนสถานะของออโตมาตาจำกัดเชิงกำหนดที่ทำงานเลียนแบบเครื่องเชิงไม่กำหนด M_5 คำถามต่อไปคือออโตมาตาเชิงกำหนดนี้จะยอมรับสตริงบนเทปที่สถานะใด ด้วยเหตุที่ออโตมาตาเชิงไม่กำหนดยอมรับสตริงบนเทปเมื่อมีการเปลี่ยนโครงแบบอย่างน้อยที่สุดหนึ่งทางที่ไปถึงสถานะสิ้นสุดได้ ดังนั้นออโตมาตาเชิงกำหนดจะยอมรับสตริงบนเทปเมื่อมันจบการทำงานในสถานะที่จำสถานะสิ้นสุดของออโตมาตาเชิงไม่กำหนดไว้อย่างน้อยหนึ่งสถานะ เช่น ในสถานะที่จำสถานะ s, p และ f ไว้ ดังนั้นออโตมาตาจำกัดเชิงกำหนดที่แสดงด้วยแผนภาพเปลี่ยนสถานะในรูป 4.8 เป็นเครื่องที่เลียนแบบการทำงานของ M_5



รูป 4.8 แผนภาพเปลี่ยนสถานะของออโตมาตาจำกัดเชิงกำหนดที่เลียนแบบ M_5

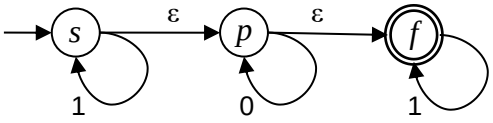
ตัวอย่างนี้แสดงให้เห็นว่าสิ่งที่ออโตมาตาจำกัดเชิงกำหนดต้องจำในแต่ละสถานะคือเซตย่อยของเซตของสถานะของออโตมาตาเชิงไม่กำหนด ดังนั้นสถานะที่เป็นไปได้ทั้งหมดมีจำนวนไม่เกิน 2^n เมื่อ n เป็นจำนวนสถานะของออโตมาตาเชิงไม่กำหนด เราจะกำหนดชื่อสถานะของออโตมาตาที่สร้างเลียนแบบออโตมาตาเชิงไม่กำหนดนี้เป็นเซตของสิ่งที่มันจำ สรุปคือถ้าให้ $M_n = (Q, \Sigma, \delta, s, F)$ เป็นออโตมาตาจำกัดเชิงไม่กำหนดที่ไม่มีการเปลี่ยนสถานะด้วยสตริงว่าง ออโตมาตาจำกัดเชิงกำหนดที่เลียนแบบการทำงานของ M_n คือ $M_d = (2^Q, \Sigma, \Delta, \{s\}, F')$ โดยที่สำหรับทุก $K \subseteq Q$ และ $a \in \Sigma$ $\Delta(K, a) = \cup_{k \in K} \delta(k, a)$ และ $F' = \{q \subseteq Q \mid q \cap F \neq \varnothing\}$

จำนวนเซตย่อยที่เป็นไปได้ของสถานะของ M_5 คือ $2^3 = 8$ แต่ออโตมาตาจำกัดเชิงกำหนดที่เราสร้างขึ้นมีเพียง 3 สถานะซึ่งจะน้อยกว่าจำนวนเซตย่อยที่เป็นไปได้ของสถานะของ M_5 เหตุที่เป็นเช่นนี้

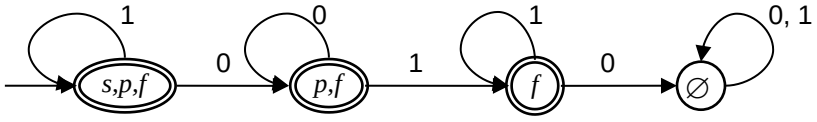
เพราะมีบางสถานะที่สร้างขึ้นมาเพื่อจำเซตของสถานะที่ไม่อาจเป็นไปได้จริงและออโตมาตาจำกัดเชิงกำหนดที่สร้างเลียนแบบขึ้นมาไม่อาจไปอยู่ในสถานะเหล่านี้ได้เลย ดังนั้นเราไม่จำเป็นต้องมีสถานะเหล่านี้ ตัวอย่างเช่น เมื่อ M_5 อยู่ในสถานะ f แล้วมันต้องอยู่ในสถานะ p ได้ด้วย ดังนั้นเครื่องที่เลียนแบบ M_5 ไม่อาจมีสถานะที่จำสถานะ s และ f โดยไม่รวมสถานะ p เช่นสถานะ $\{s, f\}$ และ $\{f\}$

ต่อมาเราจะพิจารณาการทำงานเชิงไม่กำหนดของออโตมาตาจำกัด M_6 ในตัวอย่าง 4.3 ที่มีการเปลี่ยนสถานะด้วยสตริงว่างดังแสดงในรูป 4.4 ตัวอย่างนี้แสดงให้เห็นว่าเมื่อเครื่องอยู่ที่สถานะ s แล้วมันอาจเปลี่ยน (หรืออาจไม่เปลี่ยน) ไปอยู่ในสถานะ p หรือ f โดยไม่อ่านสตริงบนเทปเลย (คือ $(s, \alpha) \vdash (p, \alpha) \vdash (f, \alpha)$ ได้) ในทำนองเดียวกันเมื่อเครื่องอยู่ที่สถานะ p แล้วมันอาจเปลี่ยนหรือไม่เปลี่ยนไปอยู่ในสถานะ f โดยไม่อ่านสตริงบนเทปก็ได้ (คือ $(p, \alpha) \vdash (f, \alpha)$) ดังนั้นเพื่อกำจัดการเปลี่ยนสถานะด้วยสตริงว่าง เครื่องต้องจำว่าถ้ามันอยู่ในสถานะ s แล้วมันอาจอยู่ในสถานะ p หรือ f ได้ด้วยและถ้ามันอยู่ในสถานะ p แล้วมันอาจอยู่ในสถานะ f ได้ด้วย (แต่ในทางกลับกันถ้าเครื่องอยู่ในสถานะ p แล้วมันอาจไม่อยู่ในสถานะ s ด้วยหรือถ้าเครื่องอยู่ในสถานะ f แล้วมันอาจไม่อยู่ในสถานะ s หรือ p ด้วย) ตัวอย่าง 4.7 แสดงการกำจัดการเปลี่ยนสถานะด้วยสตริงว่างในออโตมาตาจำกัด

ตัวอย่าง 4.7 จากตัวอย่าง 4.2 ออโตมาตาจำกัดเชิงไม่กำหนด M_6 ที่แสดงในรูป 4.9 (a) มีการเปลี่ยนสถานะด้วยสตริงว่างจากสถานะ s ไปยัง p และจากสถานะ p ไปยัง f เราสามารถกำจัดการเปลี่ยนสถานะด้วยสตริงว่างใน M_6 ได้โดยสร้างออโตมาตาจำกัดเชิงกำหนดที่แสดงในรูป 4.9 (b) ที่เลียนแบบการทำงานของ M_6 ในทำนองเดียวกันกับตัวอย่าง 4.6 ดังนี้



(a) ออโตมาตาจำกัดเชิงไม่กำหนด M_6



(b) ออโตมาตาจำกัดเชิงกำหนดที่เลียนแบบการทำงานของ M_6

รูป 4.9 การกำจัดการเปลี่ยนสถานะด้วยสตริงว่างในออโตมาตาจำกัดเชิงไม่กำหนด M_6

- เมื่อเริ่มต้นทำงาน ออโตมาตาจำกัดที่เลียนแบบต้องจำว่า M_6 อยู่ในสถานะ s และอาจอยู่ในสถานะ p และ f ได้จากการเปลี่ยนสถานะด้วยสตริงว่าง ดังนั้นเมื่อเริ่มต้นทำงาน เครื่องเลียนแบบต้องจำสถานะ s, p และ f ไว้
- จากสถานะ s, p และ f ถ้า M_6 อ่านได้ 1 สถานะถัดไปของ M_6 อาจเป็นสถานะ s หรือ f ก็ได้ (เพราะเมื่ออ่านได้ 1 มันอาจเปลี่ยนสถานะจาก s เป็น s และจาก f เป็น f แต่ไม่อาจเปลี่ยน

สถานะจาก p ไปได้) จากการเปลี่ยนสถานะด้วยสตริงว่าง เมื่อ M_6 อยู่ในสถานะ s มันอาจอยู่ในสถานะ p และ f ได้ด้วย และเมื่ออยู่ในสถานะ p M_6 อาจอยู่ในสถานะ f ได้ด้วย ดังนั้นเครื่องเลียนแบบต้องจำสถานะ s, p และ f ไว้

- จากสถานะ s, p และ f ถ้า M_6 อ่านได้ 0 สถานะถัดไปของ M_6 เป็นสถานะ p ได้เท่านั้น (เพราะมันอาจเปลี่ยนสถานะจาก p เป็น p ได้ แต่ไม่อาจเปลี่ยนสถานะจาก s และ f ไปได้) จากการเปลี่ยนสถานะด้วยสตริงว่าง เมื่อ M_6 อยู่ในสถานะ p มันอาจอยู่ในสถานะ f ได้ด้วย ดังนั้นเครื่องเลียนแบบต้องจำสถานะ p และ f ไว้
- จากสถานะ p และ f ถ้า M_6 อ่านได้ 1 สถานะถัดไปของ M_6 เป็นสถานะ f เท่านั้น (เพราะมันอาจเปลี่ยนสถานะจาก f เป็น f แต่ไม่อาจเปลี่ยนสถานะจาก p ไปได้) แต่ไม่มีการเปลี่ยนสถานะด้วยสตริงว่างจากสถานะ f ดังนั้นเครื่องเลียนแบบต้องจำสถานะ f เท่านั้น
- จากสถานะ p และ f ถ้า M_6 อ่านได้ 0 สถานะถัดไปของ M_6 เป็นสถานะ p ได้เท่านั้น (เพราะมันอาจเปลี่ยนสถานะจาก p เป็น p ได้ แต่ไม่อาจเปลี่ยนสถานะจาก f ไปได้) จากการเปลี่ยนสถานะด้วยสตริงว่าง เมื่อ M_6 อยู่ในสถานะ p มันอาจอยู่ในสถานะ f ได้ด้วย ดังนั้นเครื่องเลียนแบบต้องจำสถานะ p และ f ไว้
- จากสถานะ f ถ้า M_6 อ่านได้ 1 สถานะถัดไปของ M_6 เป็นสถานะ f เท่านั้น (เพราะมันอาจเปลี่ยนสถานะจาก f เป็น f ได้) ดังนั้นเครื่องเลียนแบบต้องจำสถานะ f ไว้
- จากสถานะ f ถ้า M_6 อ่านได้ 0 มันจะเปลี่ยนสถานะไม่ได้เลย ดังนั้นเครื่องเลียนแบบต้องจำว่าไม่มีสถานะที่เป็นไปได้

จากที่อธิบายมาข้างต้น ออโตมาตาจำกัดเชิงกำหนดในรูป 4.9 (b) ทำงานเลียนแบบออโตมาตาจำกัดเชิงไม่กำหนด M_6 ในรูป 4.9 (a) สถานะสิ้นสุดของออโตมาตาจำกัดเชิงกำหนดนี้ใช้แนวคิดเดียวกับที่กล่าวมาแล้ว คือ สถานะสิ้นสุดเป็นสถานะที่จำสถานะสิ้นสุดของออโตมาตาจำกัดเชิงไม่กำหนดอย่างน้อยหนึ่งสถานะ ดังนั้น สถานะสิ้นสุดของเครื่องนี้คือ $\{s, p, f\}$, $\{p, f\}$ และ $\{f\}$ ซึ่งจำสถานะสิ้นสุด f ของ M_6

จากตัวอย่าง 4.7 จะเห็นได้ว่าเราสามารถกำจัดการเปลี่ยนสถานะด้วยสตริงว่างจากสถานะ q ได้ โดยให้ออโตมาตาจำกัดที่เลียนแบบจำสถานะ q พร้อมไปกับสถานะที่เชื่อมกับสถานะ q โดยการเปลี่ยนสถานะด้วยสตริงว่าง เช่น จากออโตมาตา M_6 ในตัวอย่าง 4.7 มีการเปลี่ยนสถานะด้วยสตริงว่างจากสถานะ s ไปยัง p และยังเปลี่ยนสถานะด้วยสตริงว่างต่อจากสถานะ p ไปยัง f ได้ด้วย เซตของสถานะที่เชื่อมกับสถานะ q ด้วยการเปลี่ยนสถานะด้วยสตริงว่าง เรียกว่า **ส่วนปิดคลุมของสถานะ *q*** (closure of a state q) ส่วนปิดคลุมของสถานะและของเซตของสถานะนิยามได้ดังนี้

นิยาม 4.6 กำหนด $M = (Q, \Sigma, \delta, s, F)$ เป็นออโตมาตาจำกัดเชิงไม่กำหนด, q เป็นสถานะใน Q และ K เป็นเซตย่อยของ Q

ส่วนปิดคลุมของสถานะ q (เขียนแทนด้วย $E(q)$) คือ $\{p \in Q \mid (q, \varepsilon) \mid -^*_M (p, \varepsilon)\}$

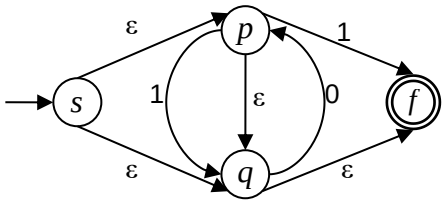
ส่วนปิดคลุมของเซต K (เขียนแทนด้วย $E(K)$) คือ $\{p \in Q \mid q \in K$ และ $(q, \varepsilon) \mid -^*_M (p, \varepsilon)\} = \cup_{q \in K} E(q)$

เราสามารถสรุปได้ว่า สำหรับออโตมาตาจำกัดเชิงไม่กำหนด $M_n = (Q, \Sigma, \delta, s, F)$ ออโตมาตาจำกัดเชิงกำหนดที่เลียนแบบการทำงานของ M_n คือ $M_d = (2^Q, \Sigma, \Delta, E(s), F')$ โดยที่สำหรับทุก $K \subseteq Q$ และ $a \in \Sigma$ $\Delta(K, a) = \cup_{k \in K} E(\delta(k, a))$ และ $F' = \{q \subseteq Q \mid q \cap F \neq \varnothing\}$ ตัวอย่างต่อไปนี้แสดงการสร้างออโตมาตาจำกัดเชิงกำหนดที่ทำงานเลียนแบบออโตมาตาจำกัดเชิงไม่กำหนด

ตัวอย่าง 4.8 กำหนดออโตมาตาจำกัดเชิงไม่กำหนด $M_7 = (\{s, p, q, f\}, \{0, 1\}, \delta, E(s), \{f\})$ ซึ่ง δ แสดงในรูป 4.10 เราสามารถสร้างออโตมาตาจำกัดเชิงกำหนดที่ทำงานเลียนแบบ M_7 ได้ดังนี้

เราหาส่วนปิดคลุมของแต่ละสถานะได้ดังแสดงในตารางในรูป 4.10 (b) จากนั้นเราเริ่มจากสถานะเริ่มต้น $E(s) = \{s, p, q, f\}$ และหา $\Delta(\{s, p, q, f\}, 0)$ และ $\Delta(\{s, p, q, f\}, 1)$ ได้ดังนี้

$$\begin{aligned}\Delta(\{s, p, q, f\}, 0) &= E(\delta(s, 0)) \cup E(\delta(p, 0)) \cup E(\delta(q, 0)) \cup E(\delta(f, 0)) \\ &= E(\varnothing) \cup E(\varnothing) \cup E(p) \cup E(\varnothing) \\ &= \varnothing \cup \varnothing \cup \{p, q, f\} \cup \varnothing \\ &= \{p, q, f\}\end{aligned}$$



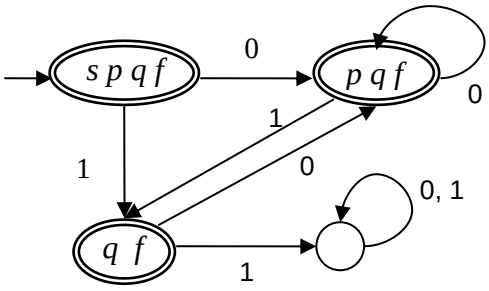
(a) ออโตมาตาจำกัด M_7

state	closure	input symbol	next state
s	s, p, q, f	0	$\varnothing$
		1	$\varnothing$
		ε	p, q
p	p, q, f	0	$\varnothing$
		1	q, f
		ε	q
q	q, f	0	p
		1	$\varnothing$
		ε	f
f	f	0	$\varnothing$
		1	$\varnothing$
		ε	$\varnothing$

(b) ตารางแสดงฟังก์ชันเปลี่ยนสถานะของ M_7

รูป 4.10 แผนภาพเปลี่ยนสถานะและตารางแสดงฟังก์ชันเปลี่ยนสถานะของออโตมาตา M_7

ทำนองเดียวกัน เราได้ $\Delta(\{s, p, q, f\}, 1) = \{q, f\}$ ดังนั้น เราได้สถานะใหม่ของ M_7 คือ $\{p, q, f\}$ และ $\{q, f\}$ และหาฟังก์ชันเปลี่ยนสถานะจากสองสถานะนี้ต่อไป เมื่อหาฟังก์ชันเปลี่ยนสถานะครบแล้วเราเลือกสถานะที่เกิดจริงได้ดังที่แสดงในรูป 4.11



State	input symbol	next state
s, p, q, f	0	*p, q, f*
	1	*q, f*
p, q, f	0	*p, q, f*
	1	*q, f*
q, f	0	*p, q, f*
	1	∅
∅	0	∅
	1	∅

รูป 4.11 ฟังก์ชันเปลี่ยนสถานะของออโตมาตาจำกัดเชิงกำหนดที่เลียนแบบ M_7

จากการหาส่วนปิดคลุมในตัวอย่าง 4.8 จะเห็นว่า ส่วนปิดคลุมสำหรับออโตมาตาจำกัดเชิงไม่กำหนด $M = (Q, \Sigma, \delta, s, F)$ มีสมบัติที่อธิบายได้ด้วยความสัมพันธ์แบบเวียนเกิดได้ดังนี้

สำหรับสถานะ $q \in Q$

$q \in E(q)$

ถ้า $p \in \delta(r, \varepsilon)$ และ $r \in E(q)$ แล้ว $p \in E(q)$

สำหรับเซตของสถานะ $K \subseteq Q$

ถ้า $k \in K$ แล้ว $k \in E(K)$

ถ้า $p \in \delta(r, \varepsilon)$ และ $r \in E(K)$ แล้ว $p \in E(K)$

เราจะใช้สมบัตินี้เขียนโปรแกรมหาส่วนปิดคลุมในหัวข้อ 4.2.2

จากการสร้างออโตมาตาจำกัดเชิงกำหนดที่เลียนแบบการทำงานของออโตมาตาจำกัดเชิงไม่กำหนดในตัวอย่าง 4.8 เราจะแสดงว่าเมื่อกำหนดออโตมาตาจำกัดเชิงไม่กำหนด M ให้ เราสามารถสร้างออโตมาตาจำกัดเชิงกำหนดที่ทำงานเลียนแบบ M ได้ ทฤษฎี 4.2 แสดงการพิสูจน์ว่าเครื่องทั้งสองยอมรับภาษาเดียวกันซึ่งซับซ้อนมากกว่าบทพิสูจน์ของทฤษฎี 4.1 ดังนั้นเราจะแสดงวิธีพิสูจน์แบบเข้มงวด

<u>ทฤษฎี 4.2</u> สำหรับออโตมาตาจำกัดเชิงไม่กำหนด M_n ใดๆ มีออโตมาตาจำกัดเชิงกำหนดที่ยอมรับ $L(M_n)$

พิสูจน์:

กำหนด $M_n = (Q, \Sigma, \delta, s, F)$ เป็นออโตมาตาจำกัดเชิงไม่กำหนดใดๆ เราจะสร้างออโตมาตาจำกัดเชิงกำหนด M_d ที่ทำงานเลียนแบบ M_n แล้วแสดงว่า $L(M_d) = L(M_n)$

เราจะให้ $M_d = (2^Q, \Sigma, \Delta, E(s), F')$ โดยที่ สำหรับทุก $K \subseteq Q$ และ $a \in \Sigma$ จะได้

$\Delta(K, a) = \cup_{k \in K} E(\delta(k, a))$ และ

$F' = \{q \subseteq Q \mid q \cap F \neq \varnothing\}$

จากนั้นต้องพิสูจน์ว่า $L(M_d) = L(M_n)$ โดยแสดงว่าสำหรับสตริง ω ใดๆ ใน Σ^* $\omega \in L(M_n)$

$\leftrightarrow$ $\omega \in L(M_d)$ เมื่อพิจารณานิยามของภาษาที่ยอมรับด้วยออโตมาตาจำกัดแล้วจะเห็นว่าต้องพิสูจน์ประโยคต่อไปนี้

สำหรับสตริง ω ใดๆ ใน Σ^* จะได้

$(s, \omega) \vdash^*_{M_n} (f, \varepsilon)$ เมื่อ $f \in F \leftrightarrow (E(s), \omega) \vdash^*_{M_d} (f', \varepsilon)$ เมื่อ $f' \in F'$

เราจะพิสูจน์โดยใช้วิธีอุปนัยบนจำนวนขั้นของการเปลี่ยนโครงแบบ ดังนั้นเราจะพิสูจน์รูปทั่วไปของประโยคข้างบนดังนี้

สำหรับสตริง ω ใดๆ ใน Σ^*, สถานะ q และ r ใดๆ ใน Q

$(q, \omega) \vdash^*_{M_n} (r, \varepsilon) \leftrightarrow (E(q), \omega) \vdash^*_{M_d} (R, \varepsilon)$ สำหรับบางสถานะ R ใน 2^Q ที่ $r \in R$

เราจะแยกพิสูจน์ประโยคข้างบนนี้เป็นสองส่วนคือ

สำหรับสตริง ω ใดๆ ใน Σ^*, สถานะ q และ r ใดๆ ใน Q และจำนวนเต็ม n ที่ไม่น้อยกว่า 0
ถ้า $(q, \omega) \vdash^*_{M_n} (r, \varepsilon)$ ใน n ขั้น **แล้ว** $(E(q), \omega) \vdash^*_{M_d} (R, \varepsilon)$ สำหรับบางสถานะ R ใน 2^Q ที่ $r \in R$

สำหรับสตริง ω ใดๆ ใน Σ^*, สถานะ q และ r ใดๆ ใน Q และจำนวนเต็ม n ที่ไม่น้อยกว่า 0
ถ้า $(E(q), \omega) \vdash^*_{M_d} (R, \varepsilon)$ ใน n ขั้น **แล้ว** $(q, \omega) \vdash^*_{M_n} (r, \varepsilon)$ สำหรับบางสถานะ R ใน 2^Q ที่ $r \in R$

ส่วนแรกเริ่มต้นด้วยการพิสูจน์ขั้นตอนฐานหลักโดยสมมุติให้ ω เป็นสตริงใดๆ ใน Σ^*, q และ r เป็นสถานะใดๆ ใน Q และ $(q, \omega) \vdash^*_{M_n} (r, \varepsilon)$ ใน 0 ขั้น

จากนิยาม 4.3 และ $(q, \omega) \vdash^*_{M_n} (r, \varepsilon)$ ใน 0 ขั้น

บอกได้ว่า $q = r$ (ทำให้ $E(q) = E(r)$ ด้วย) และ $\omega = \varepsilon$

ดังนั้น $(E(q), \omega) = (E(r), \varepsilon)$ และ $(E(q), \omega) \vdash^*_{M_d} (E(r), \varepsilon)$

นั่นคือจะมี $R = E(r)$ ซึ่ง $r \in R = E(r)$ ที่ทำให้ $(E(q), \omega) \vdash^*_{M_d} (R, \varepsilon)$

ต่อมาเราตั้งสมมุติฐานอุปนัยโดยให้ k เป็นจำนวนเต็มที่ไม่น้อยกว่า 0 และสมมุติว่า

สำหรับสตริง ω ใดๆ ใน Σ^* และสถานะ q และ r ใดๆ ใน Q
ถ้า $(q, \omega) \vdash^*_{M_n} (r, \varepsilon)$ ใน k ขั้น **แล้ว** $(E(q), \omega) \vdash^*_{M_d} (R, \varepsilon)$ สำหรับบางสถานะ R ใน 2^Q ที่ $r \in R$

จากนั้นเราจะพิสูจน์ขั้นตอนอุปนัยเพื่อสรุปว่า

สำหรับสตริง ω ใดๆ ใน Σ^* และสถานะ q และ r ใดๆ ใน Q
ถ้า $(q, \omega) \vdash^*_{M_n} (r, \varepsilon)$ ใน $k+1$ ขั้น **แล้ว** $(E(q), \omega) \vdash^*_{M_d} (R, \varepsilon)$ สำหรับบางสถานะ R ใน 2^Q ที่ $r \in R$

สมมุติให้ ω เป็นสตริงใดๆ ใน Σ^*, q และ r เป็นสถานะใดๆ ใน Q และ

$(q, \omega) \vdash^*_{M_n} (r, \varepsilon)$ ใน $k+1$ ขั้น (เนื่องจาก $k \geq 0 \quad k+1 \geq 1$)

จากนิยาม 4.3 บอกได้ว่ามีสถานะ p ที่อยู่ใน Q และสตริง α ที่อยู่ใน Σ^* ที่

$(q, \omega) \vdash^*_{M_n} (p, a)$ ใน k ขั้นและ $(p, a) \vdash_{M_n} (r, \varepsilon)$ ในหนึ่งขั้น โดยที่ $a \in \Sigma \cup \{\varepsilon\}$ ดังแสดงในรูป 4.12 (a)

จาก $(q, \omega) \vdash_{M_n} (p, a)$ ใน k ขั้นและสมมุติฐานอุปนัย เราบอกได้ว่า

$(E(q), \omega) \vdash^*_{M_d} (P, a)$ และ $p \in P$

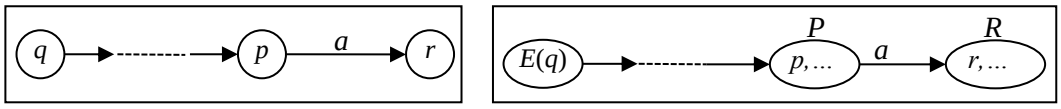
นอกจากนั้นเนื่องจาก $(p, a) \vdash_{M_n} (r, \varepsilon)$ บอกได้ว่า

$r \in E(\delta(p,a))$

จากการสร้าง M_d และ P เป็นเซตที่มี p เป็นสมาชิก บอกได้ว่า

$E(\delta(p,a)) \subseteq \Delta(P, a)$

ดังนั้นถ้า $(P, a) \vdash^*_{M_d} (R, \varepsilon)$ แล้ว $r \in R$ ดังแสดงในรูป 4.12 (b) ดังนั้นสรุปได้ว่า $(E(q), \omega) \vdash^*_{M_d} (P, a) \vdash^*_{M_d} (R, \varepsilon)$ และ $r \in R$



(a) M_n (b) M_d

รูป 4.12 การเปลี่ยนสถานะใน M_n เมื่อเทียบกับใน M_d

ส่วนที่สองเริ่มต้นด้วยการพิสูจน์ขั้นตอนฐานหลัก โดย

สมมุติให้ ω เป็นสตริงใด ๆ ใน Σ^*,

q และ r เป็นสถานะใดๆ ใน Q,

R เป็นสถานะใน 2^Q ที่ $r \in R$ และ

$(E(q), \omega) \vdash^*_{M_d} (R, \varepsilon)$ ใน 0 ขั้น

จาก $(E(q), \omega) \vdash^*_{M_d} (R, \varepsilon)$ ใน 0 ขั้นและการสร้าง M_d บอกได้ว่า

สำหรับ M_n มีสถานะ $r \in R$ ที่ $\delta(q, \varepsilon) = r$ นั่นคือ $(q, \omega) \vdash^*_{M_d} (r, \varepsilon)$

ต่อมาเราตั้งสมมุติฐานอุปนัยโดยให้ k เป็นจำนวนเต็มที่ไม่น้อยกว่า 0 และ สมมุติว่า

สำหรับสตริง ω ใดๆ ใน Σ^*, สถานะ q และ r ใดๆ ใน Q และบางสถานะ R ใน 2^Q ที่ $r \in R$
ถ้า $(E(q), \omega) \vdash^*_{M_d} (R, \varepsilon)$ ใน k ขั้นหรือน้อยกว่า **แล้ว** $(q, \omega) \vdash^*_{M_n} (r, \varepsilon)$

จากนั้นเราจะพิสูจน์ขั้นตอนอุปนัยเพื่อสรุปว่า

สำหรับสตริง ω ใดๆ ใน Σ^*, สถานะ q และ r ใดๆ ใน Q, บางสถานะ R ใน 2^Q ที่ $r \in R$
ถ้า $(E(q), \omega) \vdash^*_{M_d} (R, \varepsilon)$ ใน $k+1$ ขั้น **แล้ว** $(q, \omega) \vdash^*_{M_n} (r, \varepsilon)$

สมมุติให้ ω เป็นสตริงใดๆ ใน Σ^*,

q และ r เป็นสถานะใดๆ ใน Q และ

$(E(q), \omega) \vdash^*_{M_d} (R, \varepsilon)$ ใน $k+1$ ขั้น

(เนื่องจาก $k \geq 0$ $k+1 \geq 1$) จากนิยาม 4.3 บอกได้ว่ามีสถานะ P ใน 2^Q และสตริง α ใน Σ^* ที่

$(E(q), \omega) \vdash^*_{M_d} (P, a)$ ใน k ขั้นและ $(P, a) \vdash_{M_d} (R, \varepsilon)$ ในหนึ่งขั้น โดยที่ $a \in \Sigma$

จาก $(E(q), \omega) \vdash^*_{M_d} (P, a)$ ใน k ขั้นและสมมุติฐานอุปนัย บอกได้ว่า

$(q, \omega) \vdash^*_{M_n} (p, a)$ และ $p \in P$

นอกจากนั้น จาก $(P, a) \vdash_{M_d} (R, \varepsilon)$ บอกได้ว่ามี $r \in R$ ที่ $r \in E(\delta(p,a))$

ดังนั้น $(p, a) \vdash_{M_n} (r, \varepsilon)$ ดังนั้นสรุปได้ว่า $(q, \omega) \vdash^*_{M_n} (p, a) \vdash^*_{M_n} (r, \varepsilon)$ และ $r \in R$

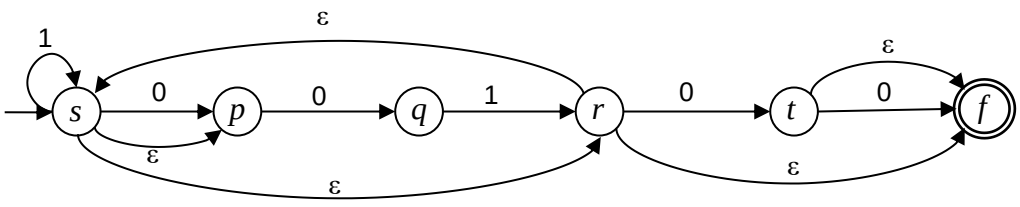
ดังนั้นสำหรับสตริง ω ใดๆ $\omega \in L(M_n) \leftrightarrow \omega \in L(M_d)$ หรือ $L(M_n) = L(M_d)$ นั่นเอง

จากทฤษฎี 4.1 และ ทฤษฎี 4.2 สรุปได้ว่า การเพิ่มการเปลี่ยนสถานะเชิงไม่กำหนดไม่ทำให้ออโตมาตาจำกัดมีความสามารถเพิ่มขึ้น นั่นคือออโตมาตาเชิงกำหนดและออโตมาตาเชิงไม่กำหนดมีความสามารถเท่ากัน ดังนั้นเมื่อต้องการพิสูจน์ว่าออโตมาตาจำกัดสามารถแก้ปัญหาใดได้เราอาจแสดงว่าออโตมาตาจำกัดเชิงไม่กำหนดสามารถแก้ปัญหานั้นได้ นอกจากนั้นเมื่อต้องการสร้างออโตมาตาจำกัดเชิงกำหนดเครื่องหนึ่ง เราอาจสร้างออโตมาตาจำกัดเชิงไม่กำหนดก่อนแล้วจึงสร้างออโตมาตาจำกัดเชิงกำหนดที่ทำงานเลียนแบบกัน

4.2.2 โปรแกรมจำลองการทำงานของออโตมาตาจำกัดเชิงไม่กำหนด

หัวข้อ 4.2.1 ได้แสดงว่าเราสามารถจำลองการทำงานของออโตมาตาจำกัดเชิงไม่กำหนดโดยใช้แนวคิดว่าให้โปรแกรมจำว่าออโตมาตาอาจอยู่ในสถานะใดได้บ้าง เราจะแสดงโปรแกรมจำลองการทำงานในตัวอย่าง 4.9

ตัวอย่าง 4.9 กำหนดออโตมาตาจำกัดเชิงไม่กำหนด M ที่แสดงในรูป 4.13 ยอมรับภาษา L = {$\omega \in$ {0, 1}* | ω มี 0 ติดกันได้ไม่เกินสองตัว}



รูป 4.13 ออโตมาตาจำกัดเชิงกำหนดที่ยอมรับสตริงที่มี 0 ติดกันได้ไม่เกินสองตัว

โปรแกรมจำลองการทำงานของออโตมาตาจำกัดเชิงไม่กำหนดที่แสดงในตัวอย่างนี้เก็บฟังก์ชัน

เปลี่ยนสถานะของออโตมาตาในโครงสร้างข้อมูลแบบ dictionary ในภาษาไพธอน ฟังก์ชันเปลี่ยนสถานะของออโตมาตาในตัวอย่างข้างบนเก็บในตัวแปร tFunc ดังในรูป 4.14

```
01: tFunc = {('s','0'):set(['p']),('s','1'):set(['s']),('s','e'):set(['p', 'r']),
02:          ('p','0'):set(['q']),('p','1'):set([]),   ('p','e'):set([]),
03:          ('q','0'):set([]),   ('q','1'):set(['r']),('q','e'):set([]),
04:          ('r','0'):set(['t']),('r','1'):set([]),   ('r','e'):set(['s', 'f']),
05:          ('t','0'):set(['f']),('t','1'):set([]),   ('t','e'):set(['f']),
06:          ('f','0'):set([]),   ('f','1'):set([]),   ('f','e'):set([]) }
```

รูป 4.14 ตัวแปรที่เก็บฟังก์ชันเปลี่ยนสถานะของออโตมาตาจำกัดเชิงไม่กำหนด

เราจะใช้นิยามของส่วนปิดคลุมที่ระบุว่าออโตมาตาอาจเปลี่ยนสถานะด้วยสตริงว่างจากสถานะหนึ่งไปอยู่ในสถานะใดในส่วนปิดคลุมได้ ฟังก์ชัน closure ที่แสดงในรูป 4.15 รับเซตของสถานะ แล้วคำนวณส่วนปิดคลุมของเซตของสถานะนั้น

```
01: def closure(states):
02:    global tFunc
03:    clsSet=states
04:    for s in states:
05:        nw=tFunc[(s,'e')]
06:        clsSet=clsSet.union(nw)
07:    if (not clsSet.issubset(states)):
08:       return states.union(closure(clsSet))
09:    else:
10:       return states
```

รูป 4.15 ฟังก์ชันคำนวณส่วนปิดคลุมของเซตของสถานะ

โปรแกรมนี้คำนวณส่วนปิดคลุมตามที่ระบุในหัวข้อ4.2.1 ว่า ถ้าสถานะ p อยู่ในเซต Q แล้ว p อยู่ในส่วนปิดคลุมของ Q ด้วย ในโปรแกรมนี้ ตัวแปร clsSet เก็บส่วนปิดคลุมของเซตของสถานะที่รับมาจากพารามิเตอร์ states

- บรรทัด 03 ให้ทุกสถานะใน states อยู่ในส่วนปิดคลุม
- บรรทัด 05-06 หาสถานะที่มีการเปลี่ยนสถานะด้วยสตริงว่างจากสถานะที่อยู่ในส่วนปิดคลุมไปถึงได้จากตัวแปร tfunc และวงวนในบรรทัด 04-06 ทำเช่นนี้กับทุกสถานะใน states
- บรรทัด 07-10 ตรวจสอบว่ามีสถานะใหม่เพิ่มมาในส่วนปิดคลุมที่หาได้ใหม่ (คือมีสถานะใน clsSet ที่ไม่อยู่ใน states) ถ้ามี จะให้หาส่วนปิดคลุมอีกรอบเพื่อหาสถานะที่มีการเปลี่ยนสถานะด้วยสตริงว่างจากสถานะที่หาได้ใหม่นี้ (บรรทัด 07-08) แต่ถ้าไม่มีสถานะใหม่เพิ่มขึ้นมาก็ไม่ต้องหาต่อ (บรรทัด 09-10)

ฟังก์ชัน closure ในรูป 4.14 หาส่วนปิดคลุมของสถานะของออโตมาตา M ในรูป 4.13 ได้ดังแสดงในตาราง 4.12

ตาราง 4.12 แสดงส่วนปิดคลุมของสถานะของออโตมาตา M

สถานะ	s	p	q	r	t	f
ส่วนปิดคลุม	{s, p, r, f}	{p}	{q}	{s, p, r, f}	{t, f}	{f}

เราสามารถจำลองการทำงานของออโตมาตาจำกัดเชิงไม่กำหนดได้ด้วยวิธีที่แสดงในตัวอย่าง 4.3 โปรแกรมในรูป 4.16 เป็นการจำลองการทำงานด้วยวิธีดังกล่าว ตัวแปร states เก็บสถานะทั้งหมดของออโตมาตา ตัวแปร startState เก็บสถานะเริ่มต้น ตัวแปร finalStates เก็บเซตของสถานะสิ้นสุด ตัวแปร tape เก็บสตริงที่อยู่บนเทปของออโตมาตา ตัวแปร sym เก็บสัญลักษณ์ที่อ่านจากเทป สถานะที่ออโตมาตาอาจอยู่ได้ในขณะหนึ่งๆ เก็บในตัวแปร states ตัวแปร setClosure เป็นโครงสร้างข้อมูลแบบ dictionary ที่เก็บส่วนปิดคลุมของแต่ละสถานะซึ่งคำนวณในบรรทัด 06-08

```
01: states=set(['s', 'f', 'p', 'q', 'r', 't',])     # SET of states
02: startState='s'                  # start state
03: finalStates=set(['f'])          # SET of final states
04: tape='1011'      # input tape
05: # ---------------------------------------- Find the closure of each state
06: setClosure={}
07: for state in states:
08:     setClosure[state]=closure(set([state]))
09: # ========================================================= START SIMULATION
10: # ------------------ At the beginning, the automata is in the start state
11: currentState=setClosure[startState]
12: # ------------------------------ Loop until the whole input tape is read
13: for sym in tape:
14:     nextState=set([])
15:     for st in currentState:
16:         for nxt in tFunc[(st,sym)]:
17: # ------------------- Get the closure of the next states
18:             nextState=nextState.union(setClosure[nxt])
19: # ------------------------------------------------- Change state
20:     currentState=nextState
21: if states.isdisjoint(finalStates): # If the machine stops in a final state,
22:     print('REJECT')               # accept the input on the tape.
23: else:                             # Otherwise,
24:     print('ACCEPT')               # reject the input string.
```

รูป 4.16 โปรแกรมจำลองการทำงานของออโตมาตาจำกัดเชิงไม่กำหนด

เมื่อเริ่มต้นทำงานออโตมาตาจะอยู่ในสถานะเริ่มต้นและสามารถเปลี่ยนสถานะด้วยสตริงว่างไปอยู่ในทุกสถานะที่อยู่ในส่วนปิดคลุมได้ ดังนั้นตัวแปร currentStates จึงเก็บส่วนปิดคลุมของสถานะเริ่มต้น (บรรทัด 11) วงวนในบรรทัด 13-20 เป็นการเปลี่ยนสถานะเมื่ออ่านสัญลักษณ์จากเทปทีละตัว (บรรทัด 13) บรรทัด 14-18 หาว่าออโตมาตาสามารถเปลี่ยนจากสถานะปัจจุบันที่เป็นไปได้ทั้งหมด (ใน

ตัวแปร currentStates) ไปอยู่ในสถานะใดบ้างเมื่ออ่านได้สัญลักษณ์ถัดไป (ในตัวแปร sym) เราหาสถานะถัดไปได้จากฟังก์ชันการเปลี่ยนสถานะ (ตัวแปร tFunc) ดังแสดงในบรรทัด 16 และนำส่วนปิดคลุมของสถานะเหล่านี้ที่เก็บในตัวแปร setClosure และเก็บรวมกันในตัวแปร nextState ในบรรทัด 18

เมื่ออ่านสัญลักษณ์บนเทปจนหมดแล้ว ถ้าสถานะสิ้นสุดสถานะหนึ่งอยู่ในสถานะในตัวแปร states ออโตมาตาจะยอมรับสตริงบนเทป (บรรทัด 21-22) ในทางตรงข้าม ออโตมาตาจะไม่ยอมรับสตริงนั้น (บรรทัด 23-24)

เราสามารถจำลองการทำงานของออโตมาตาจำกัดเชิงไม่กำหนดได้อีกวิธี โดยสร้างออโตมาตาจำกัดเชิงกำหนดที่เทียบเท่ากันดังแสดงในทฤษฎี 4.2 แล้วจำลองการทำงานของออโตมาตาจำกัดเชิงกำหนดโดยใช้โปรแกรมในรูป 3.7 การสร้างออโตมาตาจำกัดตามทฤษฎี 4.2 เริ่มด้วยการสร้างสถานะจากเซตกำลังของสถานะของออโตมาตาเชิงไม่กำหนด แล้วจึงสร้างฟังก์ชันเปลี่ยนสถานะโดยใช้ส่วนปิดคลุมของสถานะดังแสดงในตัวอย่าง 4.10

ตัวอย่าง 4.10 จากออโตมาตาในรูป 4.13 เราสร้างฟังก์ชัน mapState ในรูป 4.17 ซึ่งสร้างเลขแทนสถานะสำหรับเซตกำลังของสถานะของออโตมาตาเชิงไม่กำหนด และตัวแปร NFAtoDFA เก็บค่า {'s':1,'f':2,'p':4, 'q':8, 'r':16, 't':32} ที่ใช้ในการคำนวณเลขสำหรับสถานะของออโตมาตาจำกัดเชิงกำหนด

```
01: def mapState(states):
02:     global NFAtoDFA
03:     DFAstate=0
04:     for s in states:
05:         DFAstate += NFAtoDFA[s]
06:     return DFAstate
```

รูป 4.17 ฟังก์ชันซึ่งสร้างเลขแทนสถานะในออโตมาตาเชิงไม่กำหนด

ดังนั้น เมื่อสร้างสถานะในออโตมาตาจำกัดเชิงกำหนดแทนเซตของสถานะในออโตมาตาจำกัดเชิงไม่กำหนด เราแทนสถานะในออโตมาตาจำกัดเชิงกำหนดด้วยผลรวมของจำนวนเต็มที่เก็บคู่กับสถานะในตัวแปร NFAtoDFA เช่น สถานะในออโตมาตาจำกัดเชิงกำหนด {'f','p','r'} แทนด้วยสถานะ 22 (=2+4+16)

โปรแกรมในรูป 4.18 สร้างฟังก์ชันเปลี่ยนสถานะของออโตมาตาจำกัดเชิงกำหนด เราให้สถานะของออโตมาตาจำกัดเชิงกำหนดเป็นทุกเซตย่อยของสถานะทั้งหมดของออโตมาตาจำกัดเชิงไม่กำหนด (บรรทัด 03) และหาว่าถ้าอ่านสัญลักษณ์หนึ่งจากเทปแล้วจะเปลี่ยนไปยังสถานะใดได้บ้าง (บรรทัด 05-09) จากนั้นจึงหาส่วนปิดคลุมของสถานะเหล่านี้ (บรรทัด 10) จากนั้นจึงใช้ฟังก์ชัน mapState เพื่อคำนวณเลขที่ใช้เป็นชื่อสถานะ และเก็บค่าของฟังก์ชันการเปลี่ยนสถานะในตัวแปร DFAtransFunc (บรรทัด 12)

โปรแกรมในรูป 4.18 สร้างฟังก์ชันเปลี่ยนสถานะของออโตมาตาจำกัดเชิงกำหนดที่ทำงานเลียนแบบ M ในตัวอย่าง 4.9 ได้ดังแสดงในตาราง 4.13 ฟังก์ชัน powerset ในบรรทัด 03 เป็นฟังก์ชันที่สร้างเซตกำลังซึ่งไม่ได้แสดงไว้ที่นี่ ผู้อ่านที่สนใจอาจหาได้จากเอกสารภาษาไพธอน [4]

```
01: # ------------------------- each DFA state is a subset of all NFA states
02: DFAtransFunc={}                          # OUTPUT: DFA transition function
03: for NFAstate in powerset(states):
04: # ------------------------------------------------ each input symbol
05:    for sym in '01':
06:       nextState=set([])
07:       for st in NFAstate:
08: # ------------------- Get next state from the transition function
09:          for nxt in tFunc[(st,sym)]:
10:             nextState=nextState.union(setClosure[nxt])
11: # ---------------------- use mapState to create state number for DFA
12:       DFAtransFunc[(mapState(set(NFAstate)),sym)]=mapState(set(nextState))
```

รูป 4.18 โปรแกรมสร้างฟังก์ชันเปลี่ยนสถานะของออโตมาตาจำกัดเชิงกำหนดเลียนแบบออโตมาตาจำกัดเชิงไม่กำหนด

ตาราง 4.13 ฟังก์ชันเปลี่ยนสถานะของออโตมาตาจำกัดเชิงกำหนดที่เลียนแบบ M ในตัวอย่าง 4.9

State	Name (State)	input	Next state	Name (Next state)
{s, p, r, f}	53	0	{p, q, t, f}	60
{s, p, r, f}	53	1	{s, p, r, f}	53
{p, q, t, f}	60	0	{q, f}	10
{p, q, t, f}	60	1	{s, p, r, f}	53
{q, f}	10	0	{}	0
{q, f}	10	1	{s, p, r, f}	53
{}	0	0	{}	0
{}	0	1	{}	0

ออโตมาตานี้มี 64 สถานะ (จากขนาดของเซตกำลังของ {'s','f','p','q','r','t'}) ตารางนี้ไม่ได้แสดงสถานะที่เข้าถึงไม่ได้ จึงเหลือเพียง 4 สถานะนี้

หัวข้อต่อไปนี้อธิบายการสร้างออโตมาตาจำกัดจากออโตมาตาจำกัดที่มีอยู่แล้ว

4.3 การสร้างออโตมาตาจำกัดจากออโตมาตาย่อย

ในการประยุกต์ใช้ออโตมาตาจำกัด เรามักสร้างออโตมาตาสำหรับภาษาง่าย ๆ แล้วนำมาประกอบกันใช้สำหรับภาษาที่ซับซ้อนขึ้น ดังนั้นเราต้องพิจารณาว่าสามารถใช้การดำเนินการใดกับภาษาโดยที่ยังทำให้ผลลัพธ์เป็นภาษาที่มีออโตมาตาจำกัดยอมรับได้ นั่นหมายความว่าเราจะต้องพิสูจน์สมบัติ

ปิด (closure properties) ของคลาส (class) ของภาษาที่ยอมรับโดยออโตมาตาจำกัดภายใต้การดำเนินการที่สนใจ จากนั้นจะใช้วิธีสร้างออโตมาตาจากการพิสูจน์สมบัติปิดมาประยุกต์ใช้

ในหัวข้อนี้ เราพิจารณาการดำเนินการต่อไปนี้ คือ การรวม (union) การต่อกัน (concatenation) การเติมเต็ม (complementation) การซ้ำ (Kleene's star) และ การร่วม (intersection) อย่างไรก็ตาม คลาสของภาษาที่ยอมรับด้วยออโตมาตาจำกัดมีสมบัติปิดภายใต้การดำเนินการอื่นๆ ที่ไม่ได้กล่าวไว้ในที่นี้ด้วย

4.3.1 สมบัติปิดภายใต้การรวม

การรวมใช้สร้างภาษาที่มีสตริงมากกว่าหนึ่งรูปแบบและแต่ละรูปแบบสามารถอธิบายได้ง่าย เช่น คำอธิบาย (coment) ในภาษาซีอาจขึ้นต้นด้วย // และจบที่ท้ายบรรทัดด้วยสัญลักษณ์ newline หรือขึ้นต้นด้วย /* และปิดท้ายด้วย */ ดังนั้น เราสามารถสร้างออโตมาตา 2 เครื่องที่ยอมรับสตริง 2 รูปแบบนี้ แล้วจึงนำมาต่อกันเพื่อให้ยอมรับสตริง 2 รูปแบบนี้ ทฤษฎี 4.3 พิสูจน์สมบัติปิดภายใต้การรวมโดยแสดงว่า ถ้ามีออโตมาตา 2 เครื่องที่ยอมรับ 2 ภาษา แล้วเราสามารถนำออโตมาตา 2 เครื่องนี้มาประกอบกันเป็นออโตมาตาที่ยอมรับสตริงที่อยู่ในส่วนรวมของ 2 ภาษานี้ได้

<u>ทฤษฎี 4.3</u> คลาสของภาษาที่ยอมรับด้วยออโตมาตาจำกัดมีสมบัติปิดภายใต้การรวม

พิสูจน์

กำหนด $M_A = (Q_A, \Sigma, \delta_A, s_A, F_A)$ และ

$M_B = (Q_B, \Sigma, \delta_B, s_B, F_B)$

เป็นออโตมาตาจำกัดใดๆ โดยที่ M_A และ M_B ยอมรับภาษา L_A และ L_B ตามลำดับ

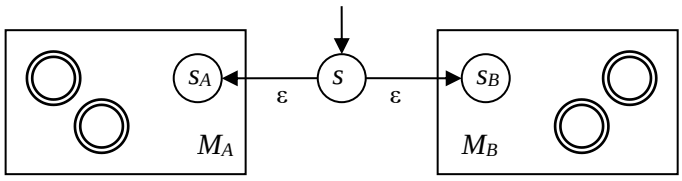
เราจะสร้างออโตมาตาจำกัดเชิงไม่กำหนด M ที่ยอมรับ $L_A \cup L_B$ ดังแสดงในรูป 4.19

ดังนั้น $M = (Q, \Sigma, \delta, s, F)$ โดยที่

$Q = Q_A \cup Q_B \cup \{s\}$,

$\delta = \delta_A \cup \delta_B \cup \{ (s, \varepsilon, \{s_A, s_B\}) \}$ และ

$F = F_A \cup F_B$



รูป 4.19 แผนภาพแสดงการสร้างออโตมาตาจำกัดที่ยอมรับ $L_A \cup L_B$

จากนั้นเราพิสูจน์ว่า M ยอมรับ $L_A \cup L_B$ โดยพิสูจน์ว่า

(1) สำหรับสตริง ω ใดๆ ใน Σ^* ถ้า ω อยู่ใน $L_A \cup L_B$ แล้ว M ยอมรับ ω และ

(2) สำหรับสตริง ω ใดๆ ใน Σ^* ถ้า ω ไม่อยู่ใน $L_A \cup L_B$ แล้ว M ไม่ยอมรับ ω

สำหรับ (1) ถ้าให้ ω เป็นสตริงใน $L_A \cup L_B$ แล้วเราต้องพิจารณาสองกรณีคือ

(1.1) ω อาจเป็นสตริงใน L_A (อยู่หรือไม่อยู่ใน L_B ก็ได้) หรือ

(1.2) ω อาจเป็นสตริงใน L_B (อยู่หรือไม่อยู่ใน L_A ก็ได้)

ในกรณี (1.1) กำหนด ω เป็นสตริงใดๆ ใน L_A

ดังนั้น จากนิยาม 4.5 เราบอกได้ว่า

$(s_A, \omega) \vdash^*_{M_A} (f_A, \varepsilon)$ โดยที่ f_A อยู่ใน F_A

เนื่องจาก $\delta_A \subseteq \delta$ เราบอกได้ว่า

$(s_A, \omega) \vdash^*_M (f_A, \varepsilon)$

นอกจากนั้น เนื่องจาก $s_A \in \delta(s, \varepsilon)$ ดังนั้น

$(s, \omega) \vdash_M (s_A, \omega)$

ดังนั้นจาก $(s, \omega) \vdash_M (s_A, \omega)$ และ $(s_A, \omega) \vdash^*_M (f_A, \varepsilon)$ สรุปได้ว่า

$(s, \omega) \vdash^*_M (f_A, \varepsilon)$

นั่นคือ M ยอมรับ ω ตามจากนิยาม 4.5

ในกรณี (1.2) กำหนด ω เป็นสตริงใดๆ ใน L_B

ดังนั้น จากนิยาม 4.5 เราบอกได้ว่า

$(s_B, \omega) \vdash^*_{M_B} (f_B, \varepsilon)$ โดยที่ f_B อยู่ใน F_B

เนื่องจาก $\delta_B \subseteq \delta$ เราบอกได้ว่า

$(s_B, \omega) \vdash^*_M (f_B, \varepsilon)$

นอกจากนั้น เนื่องจาก $s_B \in \delta(s, \varepsilon)$ ดังนั้น

$(s, \omega) \vdash_M (s_B, \omega)$

ดังนั้นจาก $(s, \omega) \vdash_M (s_B, \omega)$ และ $(s_B, \omega) \vdash^*_M (f_B, \varepsilon)$ สรุปได้ว่า

$(s, \omega) \vdash^*_M (f_B, \varepsilon)$

นั่นคือ M ยอมรับ ω ตามจากนิยาม 4.5

สำหรับ (2) กำหนด ω เป็นสตริงใดๆ ที่ไม่อยู่ใน $L_A \cup L_B$ การทำงานของ M ต้องเริ่มต้นจากโครงแบบ (s, ω) แล้ว M อาจเปลี่ยนจากสถานะ s ไปอยู่ในสถานะ s_A หรือ s_B เท่านั้น เนื่องจาก $\delta(s, \varepsilon) = \{s_A, s_B\}$ ดังนั้น เราต้องพิจารณาสองกรณี คือ

(2.1) $(s, \omega) \vdash_M (s_A, \omega)$ หรือ

(2.2) $(s, \omega) \vdash_M (s_B, \omega)$

ในกรณี (2.1) $(s, \omega) \vdash_M (s_A, \omega)$

เนื่องจาก ω ไม่อยู่ใน L_A จึงได้ว่า

ไม่มีสถานะ f_A ใน F_A ที่ $(s_A, \omega) \vdash^*_{M_A} (f_A, \varepsilon)$

จาก δ ของ M ซึ่งรวม δ_A ด้วย เราบอกได้ว่า

ไม่มีสถานะ f_A ใน F_A ที่ $(s_A, \omega) \vdash^*_M (f_A, \varepsilon)$

จาก δ ของ M ซึ่งไม่มีการเปลี่ยนสถานะจากสถานะใน Q_A ไปยังสถานะใน Q_B เราบอกได้ว่า

ไม่มีสถานะ f_B ใน F_B ที่ $(s_A, \omega) \vdash^*_M (f_B, \varepsilon)$

สรุปได้ว่าไม่มีสถานะ f ใน $F_A \cup F_B$ ที่ $(s_A, \omega) \vdash^*_M (f, \varepsilon)$

นั่นคือ M ไม่ยอมรับ ω ในกรณีนี้

ในกรณี (2.2) $(s, \omega) \vdash_M (s_B, \omega)$

เนื่องจาก ω ไม่อยู่ใน L_B จึงได้ว่า

ไม่มีสถานะ f_B ใน F_B ที่ $(s_B, \omega) \vdash^*_{M_B} (f_B, \varepsilon)$

จาก δ ของ M ซึ่งรวม δ_B ด้วย เราบอกได้ว่า

ไม่มีสถานะ f_B ใน F_B ที่ $(s_B, \omega) \vdash^*_M (f_B, \varepsilon)$

จาก δ ของ M ซึ่งไม่มีการเปลี่ยนสถานะจากสถานะใน Q_B ไปยังสถานะใน Q_A เราบอกได้ว่า

ไม่มีสถานะ f_A ใน F_A ที่ $(s_B, \omega) \vdash^*_M (f_A, \varepsilon)$

สรุปได้ว่าไม่มีสถานะ f ใน $F_A \cup F_B$ ที่ $(s_B, \omega) \vdash^*_M (f, \varepsilon)$

นั่นคือ M ไม่ยอมรับ ω ในกรณีนี้

จาก (1) และ (2) สรุปได้ว่า M ยอมรับ $L_A \cup L_B$ นั่นคือเราสามารถสร้างออโตมาตาจำกัดที่ยอมรับส่วนรวมของภาษาสองภาษาที่ยอมรับด้วยออโตมาตาจำกัดใดๆ ได้ ดังนั้น คลาสของภาษาที่ยอมรับด้วยออโตมาตาจำกัดมีสมบัติปิดภายใต้การรวม

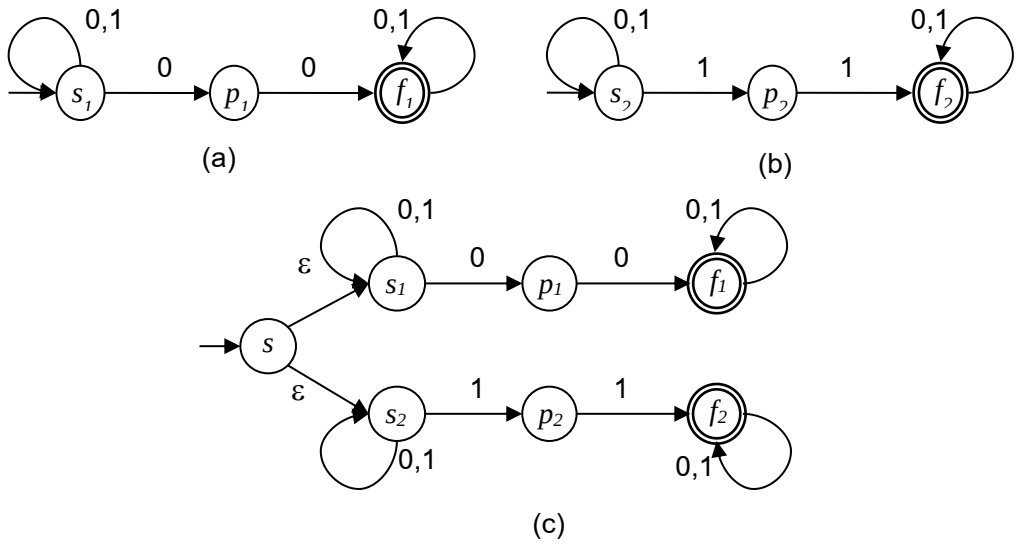
ในการพิสูจน์นี้เราให้อักขระของ 2 ภาษาเป็นอักขระชุดเดียวกัน แต่ถ้าภาษาทั้งสองมีอักขระต่างชุดกันเราก็สามารถแก้ปัญหาได้โดยสร้างอักขระชุดใหม่ที่เป็นส่วนรวมของ 2 ชุดเดิมแล้วใช้กับ 2 ภาษา

วิธีสร้างออโตมาตาที่ใช้ในทฤษฎี 4.3 นำไปเขียนโปรแกรมสร้างออโตมาตาจำกัดที่ยอมรับส่วนรวมของภาษาที่ยอมรับด้วยออโตมาตาจำกัดสองภาษาได้ นอกจากนั้น ทฤษฎี 4.3 นำมาใช้เพื่อพิสูจน์ว่าภาษาหนึ่งยอมรับด้วยออโตมาตาจำกัดได้ดังแสดงในตัวอย่าง 4.11

<u>ตัวอย่าง 4.11</u> กำหนดอักขระ $\Sigma = \{0, 1\}^*$ จงพิสูจน์ว่า $L = \{\omega \in \Sigma^* \mid 00$ หรือ 11 เป็นสตริงย่อยใน $\omega\}$ ยอมรับได้ด้วยออโตมาตาจำกัด

ถ้าให้ $L_1 = \{\omega \in \Sigma^* \mid 00$ เป็นสตริงย่อยใน $\omega\}$ และ $L_2 = \{\omega \in \Sigma^* \mid 11$ เป็นสตริงย่อยใน $\omega\}$ เนื่องจาก $L = L_1 \cup L_2$ และออโตมาตาจำกัดเชิงไม่กำหนดที่แสดงในรูป 4.20 (a) และ(b) ยอมรับ L_1 และ L_2 ตามลำดับ ดังนั้นเราสามารถสร้างออโตมาตาจำกัดที่ยอมรับ L ได้ด้วยวิธีที่แสดงในทฤษฎี 4.3

ออโตมาตาจำกัดที่ยอมรับ L แสดงในรูป 4.20 (c)



รูป 4.20 ออโตมาตาจำกัดที่ยอมรับ L

จะเห็นได้ว่าการใช้ความสามารถเชิงไม่กำหนดทำให้สร้างออโตมาตาจำกัดได้ง่ายขึ้นเมื่อเทียบกับการสร้างออโตมาตาจำกัดเชิงกำหนด

4.3.2 สมบัติปิดภายใต้การต่อกัน

การต่อกันใช้สร้างภาษาโดยนำสตริงรูปแบบต่าง ๆ มาต่อกัน เช่น ชื่อไฟล์เกิดจากการนำชื่อมาต่อกับสัญลักษณ์ . แล้วต่อด้วยนามสกุลของไฟล์ซึ่งเป็นตัวอักษร 3 ตัว ดังนั้น เราสามารถสร้างออโตมาตาจำกัดที่ยอมรับชื่อไฟล์จากออโตมาตาจำกัดที่ยอมรับชื่อ ออโตมาตาจำกัดที่ยอมรับสัญลักษณ์ . ออโตมาตาจำกัดที่ยอมรับนามสกุลของไฟล์ ทฤษฎี 4.4 พิสูจน์สมบัติปิดภายใต้การต่อกันโดยแสดงว่าถ้ามีออโตมาตาจำกัด 2 เครื่องที่ยอมรับ 2 ภาษา แล้วเราสามารถนำออโตมาตา 2 เครื่องนี้มาประกอบกันเป็นออโตมาตาจำกัดที่ยอมรับสตริงที่เกิดจากการต่อสตริงใน 2 ภาษานี้

<u>ทฤษฎี 4.4</u> คลาสของภาษาที่ยอมรับด้วยออโตมาตาจำกัดมีสมบัติปิดภายใต้การต่อกัน

พิสูจน์

กำหนด $M_A = (Q_A, \Sigma, \delta_A, s_A, F_A)$ และ

$M_B = (Q_B, \Sigma, \delta_B, s_B, F_B)$

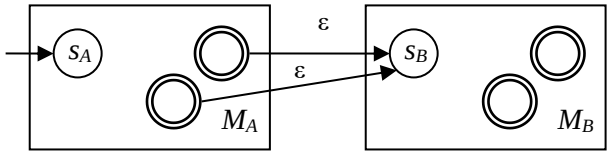
เป็นออโตมาตาจำกัดใดๆ โดยที่ M_A และ M_B ยอมรับภาษา L_A และ L_B ตามลำดับ

เราจะสร้างออโตมาตาจำกัดเชิงไม่กำหนด M ที่ยอมรับ $L_A \cdot L_B$ ดังแสดงในรูป 4.21

ดังนั้น $M = (Q, \Sigma, \delta, s_A, F_B)$ โดยที่

$Q = Q_A \cup Q_B$ และ

$$\delta = \delta_B \cup \delta_A - \{(q, \varepsilon, V) \mid q \in F_A, V \subseteq Q\} \cup \{(q, \varepsilon, V \cup \{s_B\}) \mid q \in F_A, V \subseteq Q$$



รูป 4.21 แผนภาพแสดงการสร้างออโตมาตาจำกัดที่ยอมรับ $L_A \cdot L_B$

จากนั้นเราพิสูจน์ว่า M ยอมรับ $L_A \cdot L_B$ โดยพิสูจน์ว่า

(1) สำหรับสตริง ω ใดๆ ใน Σ^* ถ้า ω อยู่ใน $L \cdot L_B$ แล้ว M ยอมรับ ω และ

(2) สำหรับสตริง ω ใดๆ ใน Σ^* ถ้า M ยอมรับ ω แล้ว ω อยู่ใน $L_A \cdot L_B$

สำหรับ (1) ถ้าให้ ω เป็นสตริงใน $L_A \cdot L_B$ แล้วจะต้องมีสตริง ω_1 ใน L_A และ ω_2 ใน L_B ที่ $\omega_1 \cdot \omega_2 = \omega$ เนื่องจาก ω_1 อยู่ใน L_A และ ω_2 อยู่ใน L_B เราบอกได้ว่า

$(s_A, \omega_1) \vdash^*_{M_A} (f_A, \varepsilon)$ โดยที่ f_A อยู่ใน F_A และ

$(s_B, \omega_2) \vdash^*_{M_B} (f_B, \varepsilon)$ โดยที่ f_B อยู่ใน F_B

เนื่องจาก $\delta_A \subseteq \delta$ และ $\delta_B \subseteq \delta$ เราบอกได้ว่า

$(s_A, \omega_1) \vdash^*_M (f_A, \varepsilon)$ และ $(s_B, \omega_2) \vdash^*_M (f_B, \varepsilon)$

จาก $(s_A, \omega_1) \vdash^*_M (f_A, \varepsilon)$ บอกได้ว่า

$(s_A, \omega_1 \cdot \omega_2) \vdash^*_M (f_A, \omega_2)$

เนื่องจาก f_A อยู่ใน F_A จึงมี s_B ใน $\delta(f_A, \varepsilon)$ ดังนั้น

$(f_A, \omega_2) \vdash^*_M (s_B, \omega_2)$

จาก $(s_A, \omega_1 \cdot \omega_2) \vdash^*_M (f_A, \omega_2)$, $(f_A, \omega_2) \vdash^*_M (s_B, \omega_2)$ และ $(s_B, \omega_2) \vdash^*_M (f_B, \varepsilon)$ จะได้ว่า

$(s_A, \omega_1 \cdot \omega_2) \vdash^*_M (f_B, \varepsilon)$

ดังนั้น M ยอมรับ ω

สำหรับ (2) กำหนด ω เป็นสตริงใดๆ ที่ยอมรับด้วย M ดังนั้น $(s_A, \omega) \vdash^*_M (f_B, \varepsilon)$

จาก δ ของ M ที่กำหนดขึ้น M เปลี่ยนจากสถานะที่อยู่ใน Q_A ไปอยู่ในสถานะที่อยู่ใน Q_B ด้วยการเปลี่ยนสถานะ $\delta(f, \varepsilon, V)$ เมื่อ $f \in F_A$ และ $s_B \in V$

ดังนั้นต้องมีสตริง ω_1 และ ω_2 ที่ $\omega = \omega_1 \cdot \omega_2$ และ

$(s_A, \omega_1 \cdot \omega_2) \vdash^*_M (f, \omega_2) \vdash_M (s_B, \omega_2) \vdash^*_M (f_B, \varepsilon)$

นอกจากนั้น $(s_A, \omega_1 \cdot \omega_2) \vdash^*_M (f, \omega_2)$ เกิดจากการเปลี่ยนสถานะโดยที่เกี่ยวข้องกับ δ_1 เท่านั้น และ $(s_B, \omega_2) \vdash^*_M (f_B, \varepsilon)$ เกิดจากการเปลี่ยนสถานะโดยที่เกี่ยวข้องกับ δ_B เท่านั้น ดังนั้น เราบอกได้ด้วยว่า

$(s_A, \omega_1 \cdot \omega_2) \vdash^*_{M_A} (f, \omega_2)$ และ $(s_B, \omega_2) \vdash^*_{M_B} (f_B, \varepsilon)$ ด้วย

จาก $(s_A, \omega_1 \cdot \omega_2) \vdash^*_{M_A} (f, \omega_2)$ บอกได้ว่า

$(s_A, \omega_1) \vdash^*_{M_A} (f, \varepsilon)$

เพราะออโตมาตาจำกัดทำงานโดยไม่อ่านสัญลักษณ์ที่หัวเทปยังไปไม่ถึง

ดังนั้นจาก $(s_A, \omega_1) \vdash^*_{M_A} (f, \varepsilon)$ และ $(s_B, \omega_2) \vdash^*_{M_B} (f_B, \varepsilon)$ เราทราบว่า ω_1 อยู่ใน L_A และ ω_2 อยู่ใน L_B เราจึงสรุปได้ว่ามีสตริง ω_1 ที่อยู่ใน L_A และ ω_2 ที่อยู่ใน L_B และ $\omega_1 \cdot \omega_2 = \omega$

ดังนั้น ω อยู่ใน $L_A \cdot L_B$

จาก (1) และ (2) สรุปได้ว่า M ยอมรับ $L_A \cdot L_B$ นั่นคือเราสามารถสร้างออโตมาตาจำกัดที่ยอมรับการต่อกันของภาษาสองภาษาที่ยอมรับด้วยออโตมาตาจำกัดใดๆ ได้ ดังนั้นคลาสของภาษาที่ยอมรับด้วยออโตมาตาจำกัดมีสมบัติปิดภายใต้การต่อกัน

ทำนองเดียวกับทฤษฎี 4.3 เราสามารถนำวิธีสร้างออโตมาตาที่ใช้ในทฤษฎี 4.4 ไปเขียนโปรแกรมสร้างออโตมาตาจำกัดที่ยอมรับส่วนต่อกันของภาษาที่ยอมรับด้วยออโตมาตาจำกัดสองภาษามาใช้ได้ นอกจากนั้นทฤษฎี 4.4 นำมาใช้เพื่อพิสูจน์ว่าภาษาหนึ่งยอมรับด้วยออโตมาตาจำกัดได้ดังแสดงในตัวอย่าง 4.12

<u>ตัวอย่าง 4.12</u> กำหนดอักขระ $\Sigma = \{0, 1\}^*$ จงพิสูจน์ว่า $L = \{\omega \in \Sigma^* \mid 00$ และ 11 เป็นสตริงย่อยใน ω โดยที่ 00 อยู่ทางซ้ายของ $11\}$ ยอมรับได้ด้วยออโตมาตาจำกัด

จาก $L_1 = \{\omega \in \Sigma^* \mid 00$ เป็นสตริงย่อยใน $\omega\}$ และ $L_2 = \{\omega \in \Sigma^* \mid 11$ เป็นสตริงย่อยใน $\omega\}$ ในตัวอย่าง 4.11 $L = L_1 \cdot L_2$ โดยที่ L_1 และ L_2 ยอมรับได้ด้วยออโตมาตาจำกัดดังแสดงในรูป 4.20 (a) และ (b) ตามลำดับ ดังนั้นจากทฤษฎี 4.4 L ยอมรับได้ด้วยออโตมาตาจำกัดและเราสามารถสร้างออโตมาตาจำกัดที่ยอมรับ L ได้ด้วยวิธีที่แสดงในทฤษฎี 4.4

4.3.3 สมบัติปิดภายใต้การเติมเต็ม

การเติมเต็มใช้สร้างภาษาจากส่วนเติมเต็มของอีกภาษา ตัวอย่างเช่น ถ้ามีออโตมาตาที่ยอมรับภาษาที่ประกอบด้วยสตริงที่มี α เป็นสตริงย่อย เราสามารถใช้ออโตมาตานี้มาสร้างออโตมาตาที่ยอมรับภาษาที่ประกอบด้วยสตริงที่ไม่มี α เป็นสตริงย่อยได้ ทฤษฎี 4.5 พิสูจน์สมบัติปิดภายใต้การเติมเต็มโดยแสดงว่า ถ้ามีออโตมาตาที่ยอมรับภาษาหนึ่ง แล้วเราสามารถนำออโตมาตาเครื่องนี้มาสร้างเป็นออโตมาตาที่ยอมรับส่วนเติมเต็มของภาษานี้ บทพิสูจน์นี้ต่างจากบทพิสูจน์ที่ผ่านมาคือเราต้องเริ่มต้นจากออโตมาตาจำกัดเชิงกำหนดเพื่อให้ออโตมาตาจำกัดที่สร้างขึ้นใหม่ทำงานได้ครบถ้วน

<u>ทฤษฎี 4.5</u> คลาสของภาษาที่ยอมรับด้วยออโตมาตาจำกัดมีสมบัติปิดภายใต้การเติมเต็ม

พิสูจน์

กำหนด $M = (Q, \Sigma, \delta, s, F)$ เป็นออโตมาตาจำกัดเชิงกำหนดใดๆ และ M ยอมรับภาษา L เราจะสร้างออโตมาตาจำกัด $\bar{M}$ ที่ยอมรับ $\bar{L}$ โดยให้ $\bar{M} = (Q, \Sigma, \delta, s, Q\text{-}F)$

จากนั้นเราพิสูจน์ว่า $\bar{M}$ ยอมรับ $\bar{L}$ โดยพิสูจน์ว่า

(1) สำหรับสตริง ω ใดๆ ใน Σ^* ถ้า ω อยู่ใน $\bar{L}$ แล้ว $\bar{M}$ ยอมรับ ω และ

(2) สำหรับสตริง ω ใดๆ ใน Σ^* ถ้า $\bar{M}$ ยอมรับ ω แล้ว ω อยู่ใน $\bar{L}$

สำหรับ (1) ถ้าให้ ω เป็นสตริงใน $\bar{L}$ เราบอกได้ว่า

$(s, \omega) \vdash^*_M (q, \varepsilon)$ โดยที่ q ไม่อยู่ใน F

นั่นคือ q อยู่ใน $Q\text{-}F$

เนื่องจาก δ ของ M คือ δ ของ $\bar{M}$ ด้วย ดังนั้นเราบอกได้ว่า

$(s, \omega) \vdash^*_{\bar{M}} (q, \varepsilon)$

เนื่องจาก q เป็นสถานะสิ้นสุดของ $\bar{M}$ ดังนั้น M ยอมรับ ω

สำหรับ (2) กำหนด ω เป็นสตริงใดๆ ที่ยอมรับด้วย $\bar{M}$ ดังนั้น

$(s, \omega) \vdash^*_{\bar{M}} (q, \varepsilon)$ โดยที่ q อยู่ใน $Q\text{-}F$

จาก δ ของ $\bar{M}$ ที่กำหนดขึ้น บอกได้ว่า

$(s, \omega) \vdash^*_M (q, \varepsilon)$ โดยที่ q ไม่อยู่ใน F

นั่นคือ M ไม่ยอมรับ ω

เนื่องจาก M ยอมรับ L ดังนั้น ω ไม่อยู่ใน L นั่นคือ ω อยู่ใน $\bar{L}$

จาก (1) และ (2) สรุปได้ว่า $\bar{M}$ ยอมรับ $\bar{L}$ นั่นคือเราสามารถสร้างออโตมาตาจำกัดที่ยอมรับส่วนเติมเต็มของภาษาที่ยอมรับด้วยออโตมาตาจำกัดใดๆ ได้ ดังนั้น คลาสของภาษาที่ยอมรับด้วยออโตมาตาจำกัดมีสมบัติปิดภายใต้การเติมเต็ม

ทำนองเดียวกับทฤษฎี 4.3 เราสามารถนำวิธีสร้างออโตมาตาจำกัดที่ยอมรับส่วนเติมเต็มของภาษาที่ยอมรับด้วยออโตมาตาจำกัดมาใช้สร้างออโตมาตาจำกัดได้ นอกจากนั้น ทฤษฎี 4.5 นำมาใช้เพื่อพิสูจน์ว่าภาษาหนึ่งยอมรับด้วยออโตมาตาจำกัดได้ดังแสดงในตัวอย่าง 4.13

<u>ตัวอย่าง 4.13</u> กำหนดอักขระ $\Sigma = \{0, 1\}^*$ จงพิสูจน์ว่า $L = \{\omega \in \Sigma^* \mid$ ไม่มี 00 เป็นสตริงย่อยใน $\omega\}$ ยอมรับได้ด้วยออโตมาตาจำกัด

ตัวอย่าง 3.6 แสดงออโตมาตาจำกัดเชิงกำหนด M_{d1} ในรูป 3.6 (a) ที่ยอมรับภาษา $L_1 = \{\omega \in \Sigma^* \mid$ 00 เป็นสตริงย่อยใน $\omega\}$

เนื่องจาก $L = \bar{L_1}$ ดังนั้นจากทฤษฎี 4.5 L ยอมรับได้ด้วยออโตมาตาจำกัดและเราสามารถสร้างออโตมาตาจำกัดที่ยอมรับ L จากออโตมาตาจำกัดเชิงกำหนด M_{d1} ได้ด้วยวิธีที่แสดงในทฤษฎี 4.5

อย่างไรก็ตามเราควรระวังด้วยว่าวิธีสร้างออโตมาตาจำกัดที่ใช้ในทฤษฎีนี้ต้องเริ่มสร้างจากออโตมาตาจำกัด**เชิงกำหนด**เท่านั้น

4.3.4 สมบัติปิดภายใต้การซ้ำ

การซ้ำใช้สร้างภาษาที่มีสตริงรูปแบบที่กำหนดซ้ำต่อ ๆ กัน เช่น ไฟล์ข้อมูลเกรดของนักเรียนซึ่งประกอบด้วยเลขประจำตัว และ เกรด ซ้ำต่อไปเรื่อย ๆ ดังนั้น เราสามารถสร้างออโตมาตาที่ยอมรับเลขประจำตัวและเกรดต่อกัน แล้วจึงนำมาต่อกันเพื่อให้ยอมรับสตริงรูปแบบนี้ซ้ำ ๆ ต่อกัน ทฤษฎี 4.6 พิสูจน์สมบัติปิดของคลาสของภาษาที่ยอมรับด้วยออโตมาตาจำกัดภายใต้การซ้ำ

ทฤษฎี 4.6 คลาสของภาษาที่ยอมรับด้วยออโตมาตาจำกัดมีสมบัติปิดภายใต้การซ้ำ

พิสูจน์

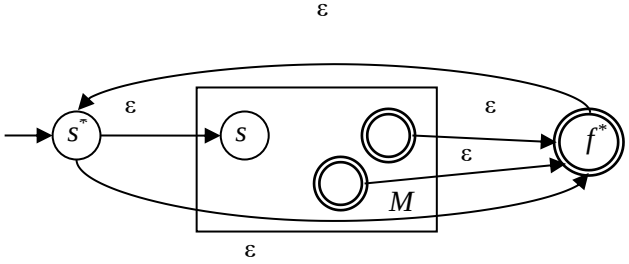
กำหนด $M = (Q, \Sigma, \delta, s, F)$ เป็นออโตมาตาจำกัดใดๆ และ M ยอมรับภาษา L

เราจะสร้างออโตมาตาจำกัดเชิงไม่กำหนด M^* ที่ยอมรับ L^* โดยให้

$$M^* = (Q \cup \{s^*, f^*\}, \Sigma, \delta^*, s^*, \{f^*\}) \text{ และ}$$

$$\delta^* = \{(s^*, \varepsilon, \{s, f^*\}), (f^*, \varepsilon, \{s^*\})\} \cup$$
$$\delta - \{(f, \varepsilon, V) | f \in F, V \subseteq Q\} \cup$$
$$\{(f, \varepsilon, V \cup \{f^*\}) | f \in F, V \subseteq Q\}$$

ดังแสดงในรูป 4.22



รูป 4.22 แผนภาพแสดงการสร้างออโตมาตาจำกัดที่ยอมรับ L^*

จากนั้นเราพิสูจน์ว่า M^* ยอมรับ L^* โดยพิสูจน์ว่า

(1) สำหรับสตริง ω ใดๆ ใน Σ^* ถ้า ω อยู่ใน L^* แล้ว M^* ยอมรับ ω และ

(2) สำหรับสตริง ω ใดๆ ใน Σ^* ถ้า M^* ยอมรับ ω แล้ว ω อยู่ใน L^*

สำหรับ (1) ถ้าให้ ω เป็นสตริงใน L^* และไม่เป็นสตริงว่าง แล้วจะต้องมีสตริง(ที่ไม่เป็นสตริงว่าง) $\omega_1, \omega_2, \ldots, \omega_n$ ใน L ที่ $\omega = \omega_1 \omega_2 \ldots \omega_n$

จาก $\omega_1, \omega_2, \ldots, \omega_n$ อยู่ใน L บอกได้ว่า

$$(s, \omega_1) \vdash^*_M (f_1, \varepsilon), (s, \omega_2) \vdash^*_M (f_2, \varepsilon), \ldots, \text{และ } (s, \omega_n) \vdash^*_M (f_n, \varepsilon)$$

โดยที่ $f_1, f_2, \ldots,$ และ f_n อยู่ใน F

นอกจากนั้นเราบอกได้ด้วยว่า

$(s, \omega_1\omega_2\ldots \omega_n) \vdash^*_M (f_1,\omega_2\ldots \omega_n)$,

$(s, \omega_2\ldots \omega_n) \vdash^*_M (f_2, \omega_3\ldots \omega_n),\ldots,$ และ

$(s, \omega_n) \vdash^*_M (f_n, \varepsilon)$

เนื่องจาก $\delta \subseteq \delta^*$ เราบอกได้ว่า

$(s, \omega_1\omega_2\ldots \omega_n) \vdash_{M^*} (f_1,\omega_2\ldots \omega_n)$,

$(s, \omega_2\ldots \omega_n) \vdash_{M^*} (f_2, \omega_3\ldots \omega_n),\ldots,$ และ

$(s, \omega_n) \vdash_{M^*} (f_n, \varepsilon)$

เนื่องจาก $\omega_1, \omega_2,\ldots, \omega_n$ ไม่เป็นสตริงว่างและจาก (s^*, ε, s) และ (s^*, ε, f^*) อยู่ใน δ^* เราบอกได้ว่า M^* มีการเปลี่ยนโครงแบบดังนี้

$(s^*, \omega_1 \omega_2\ldots \omega_n) \vdash_{M^*} (s, \omega_1 \omega_2\ldots \omega_n)$,

$(s^*, \omega_2\ldots \omega_n) \vdash_{M^*} (s, \omega_2\ldots \omega_n)$, … และ

$(s^*, \omega_n) \vdash_{M^*} (s, \omega_n)$

เนื่องจาก $\{(f, \varepsilon, f^*)| f\in F\}$ เป็นเซตย่อยของ δ^* เราบอกได้ว่า M^* มีการเปลี่ยนโครงแบบดังนี้

$(f_1, \omega_2\ldots \omega_n) \vdash_{M^*} (f^*, \omega_2\ldots \omega_n)$,

$(f_2, \omega_3\ldots \omega_n) \vdash_{M^*} (f^*, \omega_3\ldots \omega_n)$, … ,

$(f_{n-1}, \omega_n) \vdash_{M^*} (f^*, \omega_n)$ และ

$(f_n, \varepsilon) \vdash_{M^*} (f^*, \varepsilon)$

เนื่องจาก (f^*, ε, s^*) อยู่ใน δ^* เราบอกได้ว่า M^* มีการเปลี่ยนโครงแบบดังนี้

$(f^*, \omega_2\ldots \omega_n) \vdash_{M^*} (s^*, \omega_2\ldots \omega_n)$, … และ

$(f^*, \omega_n) \vdash_{M^*} (s^*, \omega_n)$

ดังนั้นเราบอกได้ว่า

$(s^*, \omega_1 \omega_2\ldots \omega_n) \vdash_{M^*} (s, \omega_1 \omega_2\ldots \omega_n)$
$\vdash^*_M (f_1,\omega_2\ldots \omega_n)$
$\vdash_{M^*} (f^*, \omega_2\ldots \omega_n)$
$\vdash_{M^*} (s^*, \omega_2\ldots \omega_n)$
$\vdash_{M^*} (s, \omega_2\ldots \omega_n)$
$\vdash^*_M (f_2, \omega_3\ldots \omega_n)$
$\vdash_{M^*} (f^*, \omega_3\ldots \omega_n)$
$\vdash_{M^*} \ldots$
$\vdash_{M^*} (s^*, \omega_n)$
$\vdash_{M^*} (s, \omega_n)$
$\vdash^*_M (f_n, \varepsilon)$
$\vdash_{M^*} (f^*, \varepsilon)$

นั่นคือ M^* ยอมรับ $\omega=\omega_1\,\omega_2\ldots\,\omega_n$

สำหรับ (2) กำหนด ω เป็นสตริงใดๆ ที่ยอมรับด้วย M^* ดังนั้น $(s^*, \omega) \vdash^*_{M^*} (f^*, \varepsilon)$ จาก δ^* ของ M^* ที่กำหนดขึ้น M^* เปลี่ยนจากสถานะที่อยู่ใน Q ไปอยู่ในสถานะ s^* หรือ f^* ด้วยการเปลี่ยนสถานะ $(f, \varepsilon, s^*)\in\delta$ เมื่อ $f\in F$ เท่านั้นและเปลี่ยนจากสถานะ s^* หรือ f^* ไปอยู่ในสถานะที่อยู่ใน Q ด้วยการเปลี่ยนสถานะ $(s^*, \varepsilon, s)\in\delta$ เท่านั้น

ดังนั้นต้องมีสตริง $\omega_1, \omega_2, \ldots$ และ ω_n ที่

$\omega = \omega_1\,\omega_2\ldots\,\omega_n$ และ

$(s^*, \omega_1\,\omega_2\ldots\,\omega_n) \vdash_{M^*} (s, \omega_1\,\omega_2\ldots\,\omega_n)$,
$(f_1,\omega_2\ldots\,\omega_n) \vdash_{M^*} (f^*, \omega_2\ldots\,\omega_n)$,
$(s^*, \omega_2\ldots\,\omega_n) \vdash_{M^*} (s, \omega_2\ldots\,\omega_n)$,
$(f_2, \omega_3\ldots\,\omega_n) \vdash_{M^*} (f^*, \omega_3\ldots\,\omega_n)$, …,
$(s^*, \omega_n) \vdash_{M^*} (s, \omega_n)$ และ
$(f_n, \varepsilon) \vdash_{M^*} (f^*, \varepsilon)$

นอกจากนั้น

$(s, \omega_1\,\omega_2\ldots\,\omega_n) \vdash^*_{M^*} (f_1,\omega_2\ldots\,\omega_n)$,
$(s, \omega_2\ldots\,\omega_n) \vdash^*_{M^*} (f_2, \omega_3\ldots\,\omega_n)$, … และ
$(s, \omega_n) \vdash^*_{M^*} (f_n, \varepsilon)$

เกิดจากการเปลี่ยนสถานะโดยที่เกี่ยวข้องกับ δ เท่านั้น ดังนั้น เราบอกได้ด้วยว่า

$(s, \omega_1\,\omega_2\ldots\,\omega_n) \vdash^*_{M} (f_1,\omega_2\ldots\,\omega_n)$,
$(s, \omega_2\ldots\,\omega_n) \vdash^*_{M} (f_2, \omega_3\ldots\,\omega_n)$, … และ
$(s, \omega_n) \vdash^*_{M} (f_n, \varepsilon)$

จาก $(s, \omega_1\,\omega_2\ldots\,\omega_n) \vdash^*_{M} (f_1,\omega_2\ldots\,\omega_n)$,
$(s, \omega_2\ldots\,\omega_n) \vdash^*_{M} (f_2, \omega_3\ldots\,\omega_n)$, … และ
$(s, \omega_n) \vdash^*_{M} (f_n, \varepsilon)$

เราบอกได้ว่า

$(s, \omega_1) \vdash^*_{M} (f_1,\varepsilon)$,
$(s, \omega_2) \vdash^*_{M} (f_2, \varepsilon)$, … และ
$(s, \omega_n) \vdash^*_{M} (f_n, \varepsilon)$

เพราะออโตมาตาจำกัดทำงานโดยไม่อ่านสัญลักษณ์ที่หัวเทปยังไปไม่ถึง

ดังนั้นเราทราบว่า $\omega_1, \omega_2,\ldots,$ และ ω_n อยู่ใน L

สรุปได้ว่ามีสตริง $\omega_1, \omega_2,\ldots,$ และ ω_n ที่อยู่ใน L ดังนั้น ω อยู่ใน L^*

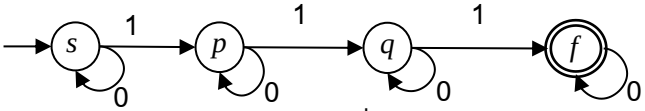
จาก (1) และ (2) สรุปได้ว่า M ยอมรับ L^* นั่นคือเราสามารถสร้างออโตมาตาจำกัดที่ยอมรับการซ้ำของภาษาที่ยอมรับด้วยออโตมาตาจำกัดใดๆ ได้ ดังนั้น คลาสของภาษาที่ยอมรับด้วยออโตมาตา

จำกัดมีสมบัติปิดภายใต้การซ้ำ

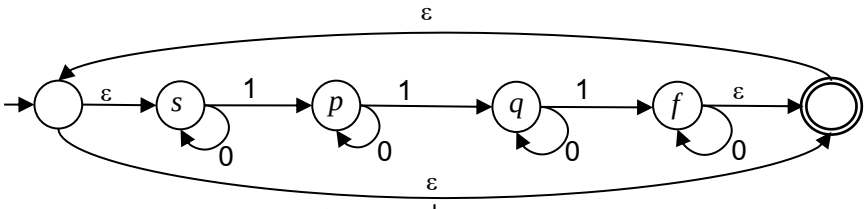
ทำนองเดียวกับทฤษฎี 4.3 เราสามารถนำวิธีสร้างออโตมาตาจำกัดที่ยอมรับภาษาที่เกิดจากการซ้ำของภาษาที่ยอมรับด้วยออโตมาตาจำกัดมาใช้ได้ นอกจากนั้นทฤษฎี 4.6 นำมาใช้เพื่อพิสูจน์ว่าภาษาหนึ่งยอมรับด้วยออโตมาตาจำกัดดังได้แสดงในตัวอย่าง 4.14

ตัวอย่าง 4.14 กำหนดอักขระ $\Sigma = \{0, 1\}^*$ จงพิสูจน์ว่า $L = \{\omega \in \Sigma^* \mid$ จำนวนของ 1 ใน ω เป็นพหุคูณของ 3 $\}$ ยอมรับได้ด้วยออโตมาตาจำกัด

เนื่องจาก $L = L_t^*$ โดยที่ $L_t = \{ \omega \in \Sigma^* \mid$ จำนวนของ 1 ใน ω เป็น 3 $\}$ และออโตมาตาจำกัดที่แสดงในรูป 4.23 (a) ยอมรับ L_t ดังนั้นจากทฤษฎี 4.6 L ยอมรับได้ด้วยออโตมาตาจำกัดและเราสามารถสร้างออโตมาตาจำกัดที่ยอมรับ L ได้ด้วยวิธีที่แสดงในทฤษฎี 4.6 ดังแสดงในรูป 4.23 (b)



(a) ออโตมาตาที่ยอมรับ L_t



(b) ออโตมาตาที่ยอมรับ $L = L_t^*$

รูป 4.23 ออโตมาตาที่ยอมรับภาษาที่สร้างจากการซ้ำของภาษาที่ยอมรับด้วยออโตมาตาจำกัด

4.3.5 สมบัติปิดภายใต้การร่วม

การร่วมเป็นตัวดำเนินการที่ใช้สร้างภาษาซึ่งสตริงในภาษานั้นเป็นไปตามรูปแบบมากกว่าหนึ่งรูปแบบร่วมกัน เช่น เราสามารถสร้างออโตมาตาจำกัดที่ยอมรับเลขฐานสองที่หารด้วย 6 และ 15 ลงตัวจากออโตมาตาที่ยอมรับเลขฐานสองที่หารด้วย 6 ลงตัวและออโตมาตาที่ยอมรับเลขฐานสองที่หารด้วย 15 ลงตัว ทฤษฎี 4.7 พิสูจน์สมบัติปิดภายใต้การร่วมของคลาสของภาษาที่ยอมรับด้วยออโตมาตาจำกัด

ทฤษฎี 4.7 คลาสของภาษาที่ยอมรับด้วยออโตมาตาจำกัดมีสมบัติปิดภายใต้การร่วม

พิสูจน์

กำหนด L_A และ L_B เป็นภาษาใดๆ ที่ยอมรับด้วยออโตมาตาจำกัด ดังนั้นจากทฤษฎี 4.5 $\overline{L_A}$ และ $\overline{L_B}$ เป็นภาษาที่ยอมรับด้วยออโตมาตาจำกัด จากทฤษฎี 4.3 $\overline{L_A} \cup \overline{L_B}$ เป็นภาษาที่ยอมรับด้วยออโตมาตาจำกัด ส่วนเติมเต็มของ $\overline{L_A} \cup \overline{L_B}$ คือ $L_A \cap L_B$ ดังนั้นจากทฤษฎี 4.5 $L_A \cap L_B$ เป็นภาษาที่

ยอมรับด้วยออโตมาตาจำกัด ดังนั้นคลาสของภาษาที่ยอมรับด้วยออโตมาตาจำกัดมีสมบัติปิดภายใต้การร่วม

บทพิสูจน์ของทฤษฎี 4.7 ต่างจากบทพิสูจน์ของทฤษฎี 4.3 - ทฤษฎี 4.6 คือบทพิสูจน์ของทฤษฎี 4.7 ไม่มีการสร้างออโตมาตาจำกัดที่ต้องการมาให้แต่ใช้ผลของทฤษฎี 4.3 และทฤษฎี 4.5 ดังนั้นเราไม่ได้วิธีสร้างออโตมาตาจำกัดที่ยอมรับภาษาที่เป็นส่วนร่วมของภาษาที่ยอมรับด้วยออโตมาตาจำกัดสองภาษาจากทฤษฎีนี้ อย่างไรก็ตามเราสามารถสร้างออโตมาตาจำกัดดังกล่าวได้โดยเริ่มจากออโตมาตาจำกัดเชิงกำหนดที่ต้องการสองเครื่อง แล้วสร้างออโตมาตาจำกัดใหม่ให้เลียนแบบการทำงานของเครื่องสองเครื่องพร้อมๆ กัน โดยให้เครื่องที่สร้างขึ้นใหม่นี้จำสถานะของออโตมาตาจำกัดสองเครื่องไปพร้อมกัน

ดังนั้น ถ้ากำหนด

$M_A = (Q_A, \Sigma, \delta_A, s_A, F_A)$ และ

$M_B = (Q_B, \Sigma, \delta_B, s_B, F_B)$

เป็นออโตมาตาจำกัดที่ยอมรับภาษา L_A และ L_B ตามลำดับ เราจะสร้างออโตมาตาจำกัดเชิงไม่กำหนด M ที่ยอมรับ $L_A \cap L_B$ ดังนี้

$M = (Q, \Sigma, \delta, s_A \times s_B, F_A \times F_B)$ โดยที่

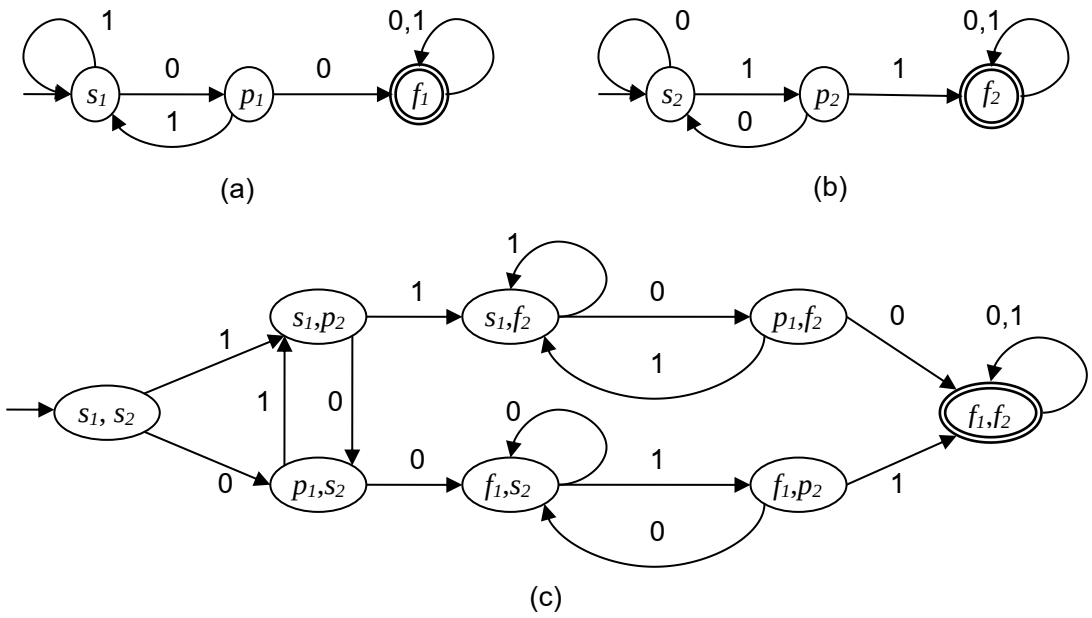
$Q = Q_A \times Q_B$ และ

$\delta((q_1, q_2), a)=(\delta_A(q_1,a), \delta_B(q_2, a))$ สำหรับสถานะ q_1 และ q_2 ใดๆ ใน Q_A และ Q_B ตามลำดับ และสัญลักษณ์ a ใดๆ ใน Σ ดังแสดงในตัวอย่าง 4.15

ตัวอย่าง 4.15 กำหนดอักขระ $\Sigma = \{0, 1\}^*$ จงสร้างออโตมาตาจำกัดที่ยอมรับ $L = \{\omega \in \Sigma^* \mid 00$ และ 11 เป็นสตริงย่อยของ $\omega\}$

จาก $L_1 = \{\omega \in \Sigma^* \mid 00$ เป็นสตริงย่อยใน $\omega\}$ และ $L_2 = \{\omega \in \Sigma^* \mid 11$ เป็นสตริงย่อยใน $\omega\}$ ในตัวอย่าง 4.11 ออโตมาตาจำกัดเชิงกำหนดในรูป 4.24 (a) และ (b) ยอมรับ L_1 และ L_2 ตามลำดับ จากทฤษฎี 4.7 สรุปได้ว่า L ยอมรับได้ด้วยออโตมาตาจำกัด

เนื่องจาก $L = L_1 \cap L_2$ เราสามารถสร้างออโตมาตาจำกัดที่ยอมรับ L ได้ดังแสดงในรูป 4.24 (c) สถานะทั้งหมดที่สร้างขึ้นอยู่ใน $\{s_1, p_1, f_1\} \times \{s_2, p_2, f_2\}$ แต่ไม่มีสถานะ (p_1, p_2) เพราะเป็นสถานะที่เกิดขึ้นจริงไม่ได้ ส่วนสถานะสิ้นสุดคือ (f_1, f_2) และฟังก์ชันเปลี่ยนสถานะสร้างจากการเปลี่ยนสถานะของ M_{d1} และ M_{d2} รวมกัน เช่น จากสถานะ (s_1, s_2) เมื่ออ่านได้สัญลักษณ์ 0 จะมีการเปลี่ยนสถานะเลียนแบบ M_{d1} คือไปอยู่ในสถานะ p_1 และเลียนแบบ M_{d2} คือไปอยู่ในสถานะ s_2 ดังนั้นเครื่องที่สร้างมานี้ต้องเปลี่ยนไปอยู่ในสถานะ (p_1, s_2)



รูป 4.24 การสร้างออโตมาตาจำกัดที่ยอมรับ $L_1 \cap L_2$

เราสามารถสร้างออโตมาตาจำกัดที่ยอมรับภาษาที่สร้างจากการรวมและการหาส่วนต่างของภาษาที่ยอมรับด้วยออโตมาตาจำกัดโดยสร้างผลคูณคาร์ทีเซียนของสถานะดังที่แสดงในตัวอย่าง 4.15 นี้ แต่กำหนดสถานะสิ้นสุดต่างไป

- ออโตมาตาที่รับส่วนรวมของสองภาษามีเซตของสถานะสิ้นสุดเป็น $(Q_A \times F_B) \cup (F_A \times Q_B)$
- สำหรับออโตมาตาที่รับส่วนต่างของสองภาษามีเซตของสถานะสิ้นสุดเป็น $(F_A \times (Q_B - F_B))$

ตัวอย่าง 4.16 แสดงการรวมออโตมาตาจำกัดด้วยวิธีดังกล่าว

ตัวอย่าง 4.16 กำหนดภาษา

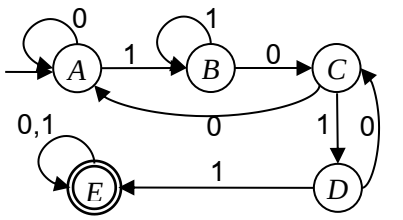
L_a = {ω∈Σ* | 1001 เป็นสตริงย่อยของ ω} และ

L_b = {ω∈Σ* | 1110 เป็นสตริงย่อยของ ω} บนอักขระ Σ = {0, 1}*

ซึ่งมีออโตมาตาจำกัด M_a และ M_b ที่แสดงในรูป 4.25 (a) และ (b) ยอมรับ L_a และ L_b ฟังก์ชันเปลี่ยนสถานะของ M_a และ M_b แสดงในตารางในรูป 4.25 (c) และ (d) ตามลำดับ

เราสามารถสร้างออโตมาตาจำกัดที่ยอมรับ $L_a \cup L_b$, $L_a \cap L_b$, $L_a - L_b$ และ $L_b - L_a$ โดยสร้างออโตมาตาเช่นเดียวกับในตัวอย่าง 4.15 แต่เปลี่ยนเซตของสถานะสิ้นสุดของออโตมาตาจำกัดตามภาษาที่ต้องการ

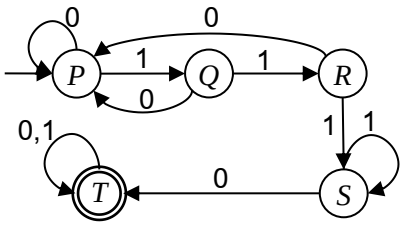
(a)



state	input	next state
A	0	*A*
A	1	*B*
B	0	*C*
B	1	*B*
C	0	*A*
C	1	*D*
D	0	*C*
D	1	*E*
E	0	*E*
E	1	*E*

(c)

(b)



state	input	next state
P	0	*P*
P	1	*Q*
Q	0	*P*
Q	1	*R*
R	0	*P*
R	1	*S*
S	0	*T*
S	1	*S*
T	0	*T*
T	1	*T*

(d)

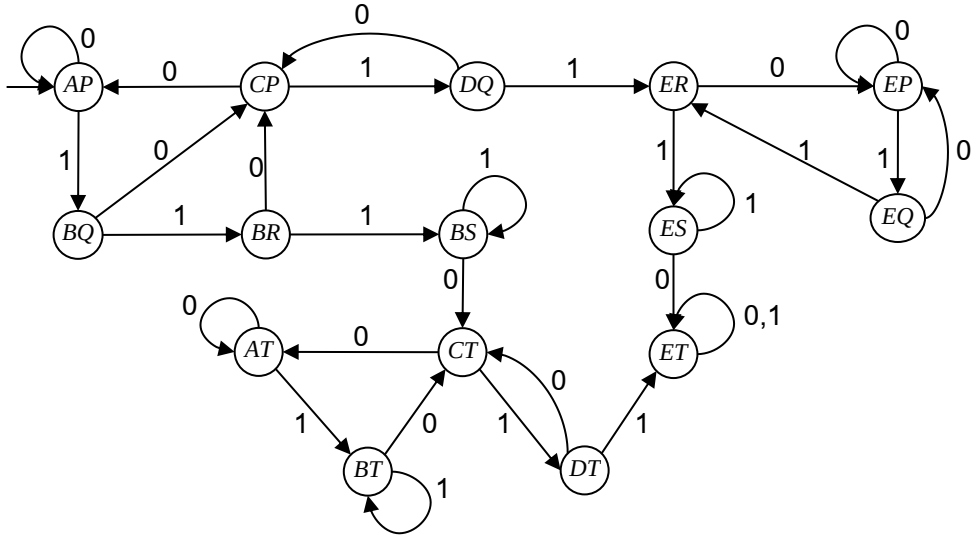
state	input	next state
AP	0	*AP*
AP	1	*BQ*
AQ	0	*AP*
AQ	1	*BR*
AR	0	*AP*
AR	1	*BS*
AS	0	*AT*
AS	1	*BS*
AT	0	*AT*
AT	1	*BT*
BP	0	*CP*
BP	1	*BQ*
BQ	0	*CP*
BQ	1	*BR*
BR	0	*CP*
BR	1	*BS*
BS	0	*CT*
BS	1	*BS*
BT	0	*CT*
BT	1	*BT*

state	input	next state
CP	0	*AP*
CP	1	*DQ*
CQ	0	*AP*
CQ	1	*DR*
CR	0	*AP*
CR	1	*DS*
CS	0	*AT*
CS	1	*DS*
CT	0	*AT*
CT	1	*DT*
DP	0	*CP*
DP	1	*EQ*
DQ	0	*CP*
DQ	1	*ER*
DR	0	*CP*
DR	1	*ES*
DS	0	*CT*
DS	1	*ES*
DT	0	*CT*
DT	1	*ET*

state	input	next state
EP	0	*EP*
EP	1	*EQ*
EQ	0	*EP*
EQ	1	*ER*
ER	0	*EP*
ER	1	*ES*
ES	0	*ET*
ES	1	*ES*
ET	0	*ET*
ET	1	*ET*

(e)

(f)



รูป 4.25 การสร้างออโตมาตาจำกัดที่ยอมรับ $L_a \cup L_b$, $L_a \cap L_b$, $L_a - L_b$ และ $L_b - L_a$

จากตารางฟังก์ชันเปลี่ยนสถานะของ M_a และ M_b เราสามารถสร้างตารางแสดงฟังก์ชันเปลี่ยนสถานะ ในรูป 4.25 (e) ตามวิธีที่ใช้ในตัวอย่าง 4.15 ชื่อของสถานะในออโตมาตาที่สร้างขึ้นใหม่นี้เป็นชื่อของสถานะในออโตมาตาสองเครื่องแรกรวมกัน เช่น สถานะ AP สร้างมาจากสถานะ A ใน M_a และสถานะ P ใน M_b เป็นต้น ดังนั้น ถ้าสตริงบนเทปทำให้ออโตมาตานี้อยู่ในสถานะ AP หมายความว่าสตริงนี้ทำให้ออโตมาตา M_a ไปอยู่ที่สถานะ A และออโตมาตา M_b ไปอยู่ที่สถานะ P แต่เมื่อออโตมาตาทั้งสองนี้ทำงานพร้อมกัน ออโตมาตาทั้งสองนี้ไม่สามารถอยู่ในสถานะบางสถานะพร้อมกันได้ ดังนั้นเมื่อสร้างออโตมาตาจากสองออโตมาตานี้จะมีบางสถานะที่ไม่มีทางเข้าถึงได้ สถานะเหล่านี้ เรียกว่าสถานะที่เข้าไม่ถึง (unreachable state) แสดงด้วยสถานะที่แรเงาในตารางในรูป 4.25 (e) สถานะ AQ เป็นสถานะที่เข้าไม่ถึงเพราะสตริงที่ทำให้ M_a ไปถึงสถานะ A ได้ต้องลงท้ายด้วย 0 อย่างน้อยหนึ่งตัว แต่สตริงที่ทำให้ M_b ไปถึงสถานะ Q ได้ต้องลงท้ายด้วย 1 ด้วยเหตุนี้จึงไม่มีสตริงใดที่ทำให้ไปถึงสถานะ A และ Q ได้พร้อมกัน ดังนั้นสถานะเหล่านี้จึงไม่จำเป็นสำหรับออโตมาตาเครื่องนี้ เรานำการเปลี่ยนสถานะที่จำเป็นในตารางนี้มาเขียนแผนภาพเปลี่ยนสถานะได้ดังแสดงในรูป 4.25 (f) โดยไม่ได้แสดงสถานะสิ้นสุด

- ออโตมาตานี้ยอมรับ $L_a \cap L_b$ เมื่อสถานะ ET เป็นสถานะสิ้นสุด
- ออโตมาตานี้ยอมรับ $L_a \cup L_b$ เมื่อ $EP, EQ, ER, ES, ET, AT, BT, CT$ และ DT เป็นสถานะสิ้นสุด
- ออโตมาตานี้ยอมรับ $L_a - L_b$ เมื่อสถานะ EP, EQ, ER และ ES เป็นสถานะสิ้นสุด
- ออโตมาตานี้ยอมรับ $L_b - L_a$ เมื่อสถานะ AT, BT, CT และ DT เป็นสถานะสิ้นสุด

จะเห็นว่าออโตมาตาที่สร้างด้วยวิธีนี้เป็นการจำลองการทำงานของออโตมาตาสองเครื่องพร้อมกันบนเทปเดียว

หัวข้อต่อไปนี้แสดงตัวอย่างการนำออโตมาตาจำกัดไปใช้ในงานต่างๆ

4.4 การประยุกต์ใช้ออโตมาตาจำกัดเชิงไม่กำหนด

การเปลี่ยนสถานะเชิงไม่กำหนดช่วยให้สร้างออโตมาตาได้ง่ายขึ้น จากการพิสูจน์สมบัติปิดในหัวข้อที่ผ่านมา เราใช้การเปลี่ยนสถานะด้วยสตริงว่างช่วยให้ประกอบออโตมาตาขึ้นจากออโตมาตาที่มีอยู่ได้ง่าย ในหัวข้อนี้ เราจะแสดงตัวอย่างการประยุกต์ใช้ออโตมาตาจำกัดเชิงไม่กำหนด หัวข้อแรกจะกล่าวถึงการใช้ออโตมาตาจำกัดเชิงไม่กำหนดแยกโทเคน (token) ในภาษาโปรแกรม (programming languages) โปรแกรมส่วนที่ทำหน้าที่นี้ในตัวแปลภาษา (compiler) เรียกว่าสแกนเนอร์ (scanner) [1], [9]

4.4.1 สแกนเนอร์

สแกนเนอร์เป็นส่วนหนึ่งของตัวแปลภาษาที่ทำหน้าที่แยกส่วนประกอบพื้นฐานของภาษาโปรแกรมที่เรียกว่าโทเคน ตัวอย่างเช่น สแกนเนอร์ในตัวแปลภาษาไพธอนจะอ่านโปรแกรมในรูป 4.26

(a) แล้วแยกเป็นโทเคนที่แสดงในรูป 4.26 (b)

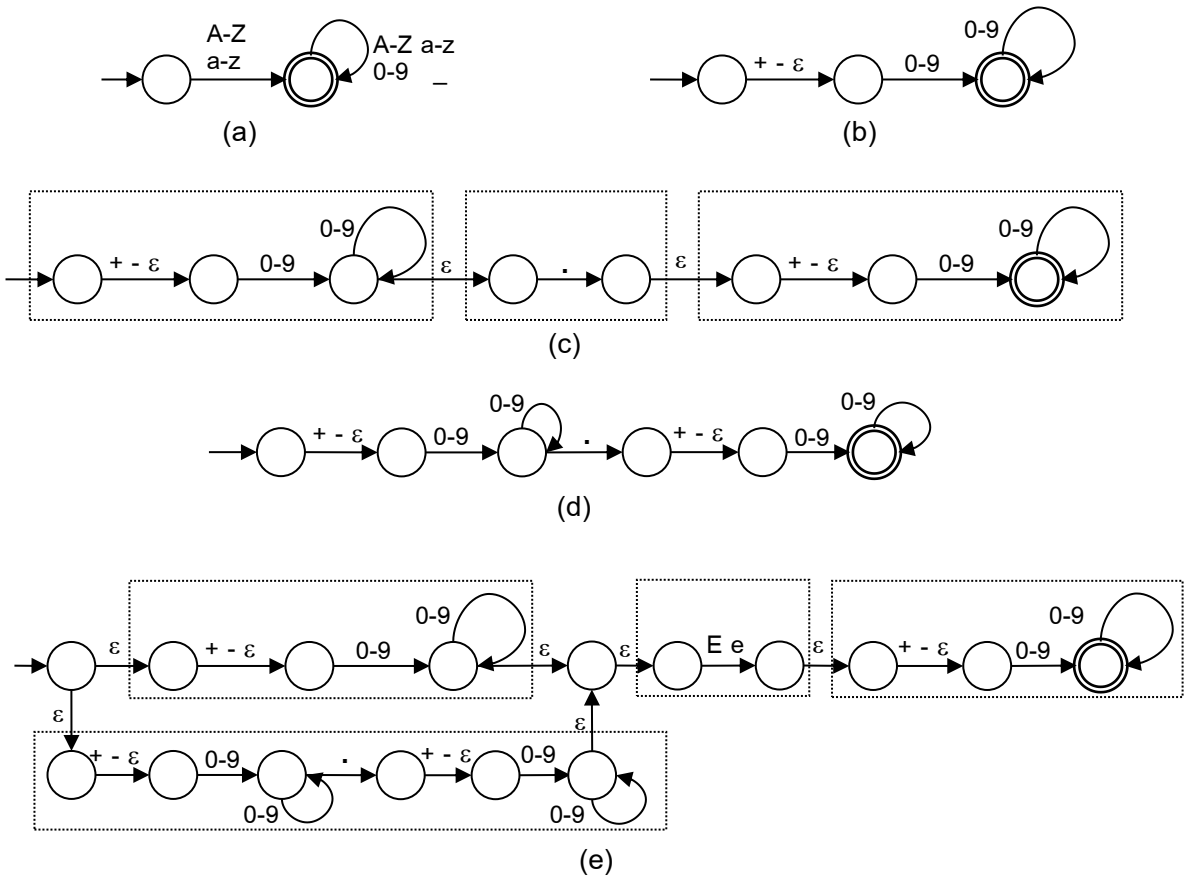
```
for temp in tempList:
    c=(temp-32)/1.8
    print("temp =",c,"c.")
```

(a)

```
keyword for          var temp          keyword in          var tempList
symbol :             var c             symbol =            symbol (
var temp             symbol -          int 32              symbol )
symbol /             float 1.8         var print           symbol (
string "temp ="      symbol ,          var c               symbol ,
string "c."          symbol )
```

(b)

รูป 4.26 ตัวอย่างโทเคนที่ได้จากโปรแกรมภาษาไพธอน



รูป 4.27 ออโตมาตาสำหรับตรวจสอบโทเคนชื่อตัวแปร จำนวนเต็ม และ จำนวนจริง

เราสามารถใช้ออโตมาตาจำกัดตรวจสอบรูปแบบของโทเคนที่ต้องการได้ เราจะแสดงตัวอย่างการสร้างสแกนเนอร์สำหรับโทเคนชื่อตัวแปร จำนวนเต็ม และ จำนวนจริง ออโตมาตาในรูป 4.27 (a) ตรวจสอบโทเคนชนิดชื่อตัวแปรซึ่งต้องขึ้นต้นด้วยตัวหนังสือ a-z และ A-Z แล้วต่อด้วยตัวหนังสือ ตัวเลข

และสัญลักษณ์ _ กี่ตัวก็ได้ ออโตมาตาในรูป 4.27 (b) ตรวจสอบโทเคนชนิดจำนวนเต็ม ซึ่งขึ้นต้นด้วยเครื่องหมายบวกหรือลบหรือไม่มีเครื่องหมายใด ๆ แล้วตามด้วยอักขระ 0 ถึง 9 อย่างน้อยหนึ่งตัว ออโตมาตาในรูป 4.27 (c) ตรวจสอบโทเคนชนิดจำนวนจริงที่ไม่มีส่วนยกกำลัง คือเป็นจำนวนเต็มที่ตามด้วยจุดทศนิยมและอาจมีเลขตามหลังจุดทศนิยมก็ได้ ออโตมาตานี้สร้างได้จากออโตมาตาที่รับจำนวนเต็มในรูป 4.27 (b) ต่อกับออโตมาตาที่รับจุดทศนิยม ต่อกับออโตมาตาที่รับจำนวนเต็ม เราสามารถลดขนาดของออโตมาตาในรูป 4.27 (c) ได้ดังแสดงในรูป 4.27 (d) ออโตมาตาในรูป 4.27 (e) ตรวจสอบโทเคนชนิดจำนวนจริงที่มีส่วนยกกำลัง คือเป็นจำนวนเต็มหรือจำนวนจริงที่ไม่มีส่วนยกกำลัง ตามด้วยตัวอักขระ e หรือ E และเลขยกกำลังซึ่งเป็นจำนวนเต็ม ออโตมาตานี้สร้างได้จากออโตมาตาที่รับจำนวนเต็มในรูป 4.27 (b) และ ออโตมาตาที่รับจำนวนจริงในรูป 4.27 (d) ที่เชื่อมกันด้วยการรวม แล้วเชื่อมด้วยการต่อกันกับออโตมาตาที่รับ e หรือ E และต่อกันกับออโตมาตาในรูป 4.27 (b)

แต่ภาษาโปรแกรมมีโทเคนหลายรูปแบบ เราจะสร้างออโตมาตาจำกัดเครื่องหนึ่งเพื่อตรวจสอบรูปแบบของโทเคนรูปแบบหนึ่ง จากนั้นจึงนำออโตมาตาทั้งหมดมาประกอบกันโดยใช้การรวม (union) ในการสร้างสแกนเนอร์ สำหรับโทเคนทั้งหมดนี้ เราสร้างออโตมาตาจำกัดเชิงไม่กำหนด โดยการนำออโตมาตาในรูป 4.27 (a) (b) (d) และ (e) มาต่อกันด้วยการรวม จากนั้นเราสามารถแปลงออโตมาตาที่ได้นี้เป็นออโตมาตาจำกัดเชิงกำหนด และเขียนโปรแกรมจำลองการทำงานของออโตมาตานี้ได้ อย่างไรก็ตามปัจจุบันมีโปรแกรมสร้างสแกนเนอร์โดยให้ผู้ใช้กำหนดรูปแบบของโทเคนด้วยนิพจน์ปกติซึ่งจะกล่าวถึงในบทถัดไป

บทที่ 5

ภาษาปกติ

บทที่ 5 ภาษาปกติ

ภาษาปกติ (regular language) เป็นภาษาที่แทนได้ด้วยนิพจน์ปกติ (regular expression) บทนี้จะนิยามภาษาปกติ หัวข้อ 5.1 นิยามนิพจน์ปกติที่ใช้แทนภาษาปกติ ต่อมาหัวข้อ 5.2 แสดงความสัมพันธ์ระหว่างนิพจน์ปกติและออโตมาตาจำกัด หัวข้อ 5.3 แสดงสมบัติปิดของคลาสของภาษาปกติ และ หัวข้อ 5.4 แสดงการประยุกต์ใช้นิพจน์ปกติ

5.1 นิพจน์ปกติ

นิพจน์ปกติใช้ตัวดำเนินการ (operator) และตัวถูกกระทำ (operand) เพื่ออธิบายการสร้างสตริงในภาษา ดังนั้นผลลัพธ์ของนิพจน์ปกติเป็นภาษาหรือเซตของสตริงนั่นเอง นิพจน์ปกติมีรูปแบบคล้ายนิพจน์ทางคณิตศาสตร์ แต่ตัวดำเนินการในนิพจน์ปกติประกอบด้วยการต่อกัน การรวม และการซ้ำ ส่วนตัวถูกกระทำจะเป็นสัญลักษณ์ในภาษา สตริงว่าง หรือ เซตว่าง รูปแบบของนิพจน์ปกติแสดงได้ดังนิยาม 5.1

นิยาม 5.1 กำหนด Σ เป็นอักขระ **นิพจน์ปกติ** (regular expression) บนอักขระ Σ เป็นสตริงบนอักขระ $\Sigma \cup \{\varepsilon, \cup, \varnothing, *, (,)\}$ โดยที่

1. $\varnothing$ เป็นนิพจน์ปกติ
2. ε เป็นนิพจน์ปกติ
3. ถ้า a เป็นสัญลักษณ์ใน Σ แล้ว a เป็นนิพจน์ปกติ
4. ถ้า α และ β เป็นนิพจน์ปกติแล้ว (α), $(\alpha\,\beta)$, $(\alpha \cup \beta)$, (α^*) เป็นนิพจน์ปกติ
5. สตริงอื่นๆนอกจากที่กำหนดใน 1-3 นี้ไม่เป็นนิพจน์ปกติ

สังเกตได้ว่านิพจน์ปกติตามนิยาม 5.1 นี้ต้องมีวงเล็บครอบสำหรับการกระทำทุกครั้งซึ่งทำให้นิพจน์ปกติที่ใช้ตัวดำเนินการหลายตัวยาวมาก ดังนั้นเราจะใส่วงเล็บในกรณีที่อาจทำให้เข้าใจลำดับการทำงานผิด ตัวอย่างต่อไปนี้แสดงนิพจน์ปกติตามนิยาม 5.1 ที่อธิบายส่วนประกอบในภาษาโปรแกรม เช่น เซตของชื่อตัวแปร เป็นต้น

ตัวอย่าง 5.1 ส่วนประกอบในภาษาโปรแกรมสามารถอธิบายด้วยนิพจน์ปกติ กำหนดอักขระ Σ = {A, B, ..., Z} $\cup$ {a, b, ..., z} $\cup$ {0, 1, ..., 9} $\cup$ {e, E, +, -, . , _ } และ ให้ α เป็นนิพจน์ปกติที่แทนเซตของ

ตัวอักษร และ η เป็นนิพจน์ปกติที่แทนเซตของตัวเลข นั่นคือ $\alpha = (A \cup B \cup ... \cup Z \cup a \cup b \cup ... \cup z)$ และ $\eta = (0 \cup 1 \cup ... \cup 9)$ ตัวอย่างนี้พิจารณา

- ชื่อตัวแปรที่ขึ้นต้นด้วยตัวอักษรหรือเครื่องหมาย _ ต่อด้วยตัวอักษร ตัวเลข หรือเครื่องหมาย _ กี่ตัวก็ได้ เช่น a_1
- จำนวนเต็มที่ขึ้นต้นด้วยเครื่องหมาย + หรือ - หรือไม่มีเครื่องหมายใด ๆ แล้วตามด้วยตัวเลขอย่างน้อยหนึ่งตัว เช่น -31, 82
- จำนวนจริงที่ขึ้นต้นด้วยเครื่องหมาย + หรือ - หรือไม่มีเครื่องหมายใด ๆ แล้วตามด้วยตัวเลขกี่ตัวก็ได้แล้วตามด้วยเครื่องหมาย . แล้วตามด้วยตัวเลขอย่างน้อยหนึ่งตัว ต่อด้วยเลขยกกำลังหรือไม่ก็ได้ เลขยกกำลังประกอบด้วยสัญลักษณ์ E หรือ e ต่อกับสตริงที่ใช้แทนจำนวนเต็ม เช่น -3.12e-21

นิพจน์ปกติสำหรับสตริงที่ใช้แทนชื่อตัวแปร จำนวนเต็ม และ จำนวนจริง เขียนได้ดังนี้

- $(\alpha \cup \_)\ (\alpha \cup \eta \cup \_)^*$ เป็นนิพจน์ปกติที่แทนเซตของชื่อตัวแปร
- $(+ \cup - \cup \varepsilon)\ \eta\eta^*$ เป็นนิพจน์ปกติที่แทนเซตของสตริงของจำนวนเต็ม
- $(+ \cup - \cup \varepsilon)\ \eta^* . \eta\eta^*\ (\varepsilon \cup (E \cup e)\ (+ \cup - \cup \varepsilon)\ \eta\eta^*)$ เป็นนิพจน์ปกติที่แทนเซตของสตริงของจำนวนจริง

นิยาม 5.1 กำหนดเพียงรูปแบบของนิพจน์ปกติโดยไม่ได้กำหนดว่านิพจน์ปกตินั้นแทนเซตของสตริงใด นิยาม 5.2 นี้กำหนดว่านิพจน์ปกติแทนเซตของสตริงใด

นิยาม 5.2 กำหนด Σ เป็นอักขระและ γ เป็นนิพจน์ปกติบนอักขระ Σ ภาษาที่อธิบายด้วยนิพจน์ปกติ γ สามารถเขียนแทนด้วย $L(\gamma)$ และนิยามได้ดังนี้

1. ถ้า $\gamma = \varnothing$ แล้ว $L(\gamma)$ เป็นเซตว่าง
2. ถ้า $\gamma = \varepsilon$ แล้ว $L(\gamma)$ เป็น $\{\varepsilon\}$
3. ถ้า a เป็นสัญลักษณ์ใน Σ และ $\gamma = a$ แล้ว $L(\gamma)$ เป็น $\{a\}$
4. ถ้า α และ β เป็นนิพจน์ปกติและ
   - 4.1 $\gamma = (\alpha\ \beta)$ แล้ว $L(\gamma)$ เป็น $L(\alpha) \cdot L(\beta)$
   - 4.2 $\gamma = (\alpha \cup \beta)$ แล้ว $L(\gamma)$ เป็น $L(\alpha) \cup L(\beta)$
   - 4.3 $\gamma = (\alpha\ ^*)$ แล้ว $L(\gamma)$ เป็น $L(\alpha)^*$

เราเรียกภาษาที่อธิบายได้ด้วยนิพจน์ปกติว่า **ภาษาปกติ** (regular language)

จากนิยาม 5.2 นิพจน์ปกติ $\varnothing$ แทนเซตว่าง นิพจน์ปกติ ε แทนเซตของสตริงว่าง และสัญลักษณ์ใด ๆ ในอักขระเป็นนิพจน์ปกติที่แทนเซตของสตริงที่มีสัญลักษณ์นั้นหนึ่งตัว จากนั้นนิพจน์ปกติที่สร้างโดยใช้ตัวดำเนินการหนึ่งใช้แทนเซตของสตริงที่สร้างด้วยตัวดำเนินการนั้นโดยที่นิพจน์ปกติ

$(\alpha\ \beta)$ แทนเซตของสตริงที่เกิดจากการต่อกันของสตริงในเซตที่แทนด้วย α และ β ทำนองเดียวกัน นิพจน์ปกติ $(\alpha \cup \beta)$ ใช้แทนเซตของสตริงที่เกิดจากการรวมเซตของสตริงที่แทนด้วย α และ β นิพจน์ปกติ $(\alpha\ ^*)$ ใช้แทนเซตของสตริงที่เกิดจากการเรียงซ้ำสตริงในเซตที่แทนด้วย α ตัวอย่าง 5.2 แสดงภาษาที่แทนด้วยนิพจน์ปกติในตัวอย่าง 5.1

<u>ตัวอย่าง 5.2</u> จากนิยาม 5.2 เราพิจารณาเซตของสตริงที่แทนด้วยนิพจน์ปกติจากตัวอย่าง 5.1

$L(\ (\alpha \cup \_\ )\ (\alpha \cup \eta \cup \_\ )^*\ )$

= {A, B, ..., Z, a, b, ..., z, _ } · { A, B, ..., Z, a, b, ..., z, _ , 0, 1, …, 9}*

= {ω | ω ขึ้นต้นด้วยตัวอักษรหรือเครื่องหมาย _ ต่อด้วยตัวอักษร ตัวเลข หรือเครื่องหมาย _ กี่ตัวก็ได้}

ดังนั้น $(\alpha \cup \_\ )\ (\alpha \cup \eta \cup \_\ )^*$ เป็นนิพจน์ปกติที่แทนเซตของชื่อตัวแปร

$L(\ (+ \cup - \cup \varepsilon)\ \eta\eta^*)$

= {+, – , ε} · {0, 1, …, 9} · … · {0, 1, …, 9}

= {ω | ω เป็นสตริงที่ประกอบด้วยตัวเลขอย่างน้อยหนึ่งตัวและอาจมีเครื่องหมาย + หรือ – ข้างหน้า}

ดังนั้น $(+ \cup - \cup \varepsilon)\ \eta\eta^*$ เป็นนิพจน์ปกติที่แทนเซตของสตริงของจำนวนเต็ม

$L(\ (+ \cup - \cup \varepsilon)\ \eta^*\ .\ \eta\eta^*\ (\varepsilon \cup (E \cup e)\ (+ \cup - \cup \varepsilon)\ \eta\eta^*)\ )$

= {+, –, ε} · {0, 1, …, 9} ·…· {0, 1, …, 9} · ({ε} ∪ { E, e} · {+, –, ε} · {0, 1, …, 9} ·…· {0, 1, …, 9}

= {ω | ω เป็นสตริงที่ขึ้นต้นด้วยเครื่องหมาย + หรือ – หรือไม่มีเครื่องหมายใด ๆ แล้วตามด้วยตัวเลขกี่ตัวก็ได้ ตามด้วยเครื่องหมาย . แล้วตามด้วย ตัวเลขอย่างน้อยหนึ่งตัว ต่อด้วยสัญลักษณ์ E หรือ e ต่อกับสตริงที่ใช้แทนจำนวนเต็ม ส่วนที่เป็นสัญลักษณ์ E หรือ e ต่อกับสตริงที่ใช้แทนจำนวนเต็มนี้อาจมีหรือไม่ก็ได้ นิพจน์ปกตินี้แทนสตริงของจำนวนจริงที่อาจมีส่วนยกกำลังหรือไม่ก็ได้ }

ดังนั้น $(+ \cup - \cup \varepsilon)\ \eta^*\ .\ \eta\eta^*\ (\varepsilon \cup (E \cup e)\ (+ \cup - \cup \varepsilon)\ \eta\eta^*)$ เป็นนิพจน์ปกติที่แทนเซตของสตริงของจำนวนจริง

ตัวอย่างต่อไปนี้แสดงการเขียนนิพจน์ปกติสำหรับสตริงที่ไม่มี 0 และ1 ติดกัน

<u>ตัวอย่าง 5.3</u> กำหนด Σ = {0, 1} เป็นอักขระ และ นิพจน์ปกติ β = ((01)* (0∪ε)) ∪ ((10)* (1∪ε)) แทนเซตของสตริงที่ไม่มี 0 ติดกันและไม่มี 1 ติดกัน ดังแสดงต่อไปนี้

$L(\beta)$ = $L(\ ((01)^*\ (0 \cup \varepsilon)) \cup ((10)^*\ (1 \cup \varepsilon))\ )$

= $L(\ ((01)^*\ (0 \cup \varepsilon))\ ) \cup L(\ ((10)^*\ (1 \cup \varepsilon))\ )$

= $(\ (L(01))^*\ L(0 \cup \varepsilon)\ ) \cup (\ (L(10))^*\ L(1 \cup \varepsilon)\ )$

= ({01}* {0, ε}) ∪ ({10}* {1, ε})

= ({$x \in \Sigma^*$| x ขึ้นต้นด้วย 0, ลงท้ายด้วย 1, มีความยาวเป็นคู่และใน x ไม่มี 0 ติดกันและไม่มี 1 ติดกัน } {0, ε}) ∪ ({ $x \in \Sigma^*$| x ขึ้นต้นด้วย 1 ลงท้ายด้วย 0 มีความยาวเป็นคู่และใน x ไม่มี 0 ติดกันและไม่มี 1 ติดกัน } {1, ε})

= { $x \in \Sigma^*$| x ขึ้นต้นด้วย 0 และใน x ไม่มี 0 ติดกันและไม่มี 1 ติดกัน } ∪ { $x \in \Sigma^*$| x ขึ้นต้นด้วย 1 และใน x ไม่มี 0 ติดกันและไม่มี 1 ติดกัน }

= { $x \in \Sigma^*$ | ใน x ไม่มี 0 ติดกันและไม่มี 1 ติดกัน }

ในหัวข้อ 5.2 เราจะแสดงว่าคลาสของภาษาปกติและคลาสของภาษาที่ยอมรับด้วยออโตมาตาจำกัดเป็นคลาสเดียวกัน

5.2 ภาษาปกติและออโตมาตาจำกัด

เราจะแสดงว่าคลาสของภาษาปกติและคลาสของภาษาที่ยอมรับด้วยออโตมาตาจำกัดเป็นคลาสเดียวกัน ในส่วนแรก เราแสดงว่ามีออโตมาตาจำกัดที่ยอมรับภาษาปกติในทฤษฎี 5.1 จากการพิสูจน์ทฤษฎีนี้ เราจะได้วิธีสร้างออโตมาตาจำกัดที่ยอมรับภาษาที่แทนด้วยนิพจน์ปกติที่กำหนด ในส่วนที่สอง เราแสดงว่ามีนิพจน์ปกติที่แทนภาษาที่ยอมรับด้วยออโตมาตาจำกัดในทฤษฎี 5.2 จากการพิสูจน์ทฤษฎีนี้ เราจะได้วิธีสร้างนิพจน์ปกติที่แทนภาษาที่ยอมรับด้วยออโตมาตาจำกัดที่กำหนด

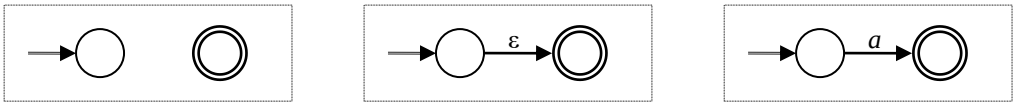
ทฤษฎี 5.1 ถ้า L เป็นภาษาปกติ แล้วต้องมีออโตมาตาจำกัดที่ยอมรับ L

พิสูจน์

กำหนด L เป็นภาษาปกติใดๆ บนอักขระ Σ จากนิยาม 5.1 จะได้ว่ามีนิพจน์ปกติ α ที่ $L = L(\alpha)$ เราจะพิสูจน์โดยวิธีอุปนัยบนโครงสร้าง (structural induction) ว่ามีออโตมาตาจำกัด M ที่ยอมรับ $L(\alpha)$ สำหรับนิพจน์ปกติ α ใดๆ ดังนั้นเราจะพิจารณาโครงสร้างของนิพจน์ปกติตามนิยาม 5.1 ขั้นตอนฐานหลักพิจารณานิพจน์ปกติที่สร้างโดยใช้ตัวดำเนินการ 0 ครั้ง คือ ∅, ε และ a โดยที่ $a \in \Sigma$ และขั้นตอนอุปนัยพิจารณานิพจน์ปกติซึ่งสร้างจากการต่อกัน การรวม และ การซ้ำ

ขั้นตอนฐานหลัก เราจะแสดงว่าเราสามารถสร้างออโตมาตาจำกัด M ที่ $L(M) = L(\alpha)$ สำหรับนิพจน์ปกติ α = ∅, ε และ a โดยที่ $a \in \Sigma$

- ถ้า α = ∅ แล้ว M เป็นออโตมาตาจำกัดที่แสดงในรูป 5.1 (a)
- ถ้า α = ε แล้ว M เป็นออโตมาตาจำกัดที่แสดงในรูป 5.1 (b)
- ถ้า α = a โดยที่ $a \in \Sigma$ แล้ว M เป็นออโตมาตาจำกัดที่แสดงในรูป 5.1 (c)



(a) (b) (c)

รูป 5.1 ออโตมาตาจำกัดที่ยอมรับ $\varnothing$, $\{\varepsilon\}$ และ $\{a\}$ โดยที่ $a \in \Sigma$

<u>ขั้นตอนอุปนัย</u> กำหนดสมมุติฐานอุปนัยว่ามีออโตมาตาจำกัด M ที่ $L(M) = L(\alpha)$ สำหรับนิพจน์ปกติ α ใดๆ ที่สร้างโดยใช้ตัวดำเนินการน้อยกว่า n ครั้ง เราจะพิสูจน์ว่ามีออโตมาตาจำกัดที่ยอมรับ $L(\beta)$ สำหรับนิพจน์ปกติ β ใดๆ ที่สร้างโดยใช้ตัวดำเนินการ n ครั้ง

กำหนดให้ β เป็นนิพจน์ปกติใดๆ ที่สร้างโดยใช้ตัวดำเนินการ n ครั้ง ดังนั้นจะต้องมีนิพจน์ปกติ α_1 และ α_2 ที่สร้างโดยใช้ตัวดำเนินการน้อยกว่า n ครั้งและ

(1) $\beta = (\alpha_1\alpha_2)$ หรือ

(2) $\beta = (\alpha_1 \cup \alpha_2)$ หรือ

(3) $\beta = (\alpha_1^*)$

จากสมมุติฐานอุปนัยจะได้ว่า มีออโตมาตาจำกัด $M_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$ และ $M_2 = (Q_2, \Sigma, \delta_2, s_1, F_2)$ ที่ $L(M_1) = L(\alpha_1)$ และ $L(M_2) = L(\alpha_2)$

ในกรณี (1) $\beta = (\alpha_1\alpha_2)$ จากทฤษฎี 4.4 เราสามารถสร้างออโตมาตาจำกัด M ซึ่ง

$$M = (Q_1 \cup Q_2, \Sigma, \delta_1 \cup \delta_2 \cup \{(q, \varepsilon, s_2) \mid q \in F_1\}, s_1, F_2)$$

ซึ่ง $L(M) = L(\alpha_1)\,L(\alpha_2)$ จากนิยาม 5.2 $L(\alpha_1)\,L(\alpha_2) = L(\alpha_1\alpha_2) = L(\beta)$ นั่นคือออโตมาตาจำกัด M ยอมรับ $L(\beta)$

ในกรณี (2) $\beta = (\alpha_1 \cup \alpha_2)$ จากทฤษฎี 4.3 เราสามารถสร้างออโตมาตาจำกัด M ซึ่ง

$$M = (Q_1 \cup Q_2 \cup \{s\}, \Sigma, \delta_1 \cup \delta_2 \cup \{(s, \varepsilon, s_1), (s, \varepsilon, s_2)\}, s, F_1 \cup F_2)$$

ซึ่ง $L(M) = L(\alpha_1) \cup L(\alpha_2)$ จากนิยาม 5.2 $L(\alpha_1) \cup L(\alpha_2) = L(\alpha_1 \cup \alpha_2) = L(\beta)$ นั่นคือมีออโตมาตาจำกัดที่ยอมรับ $L(\beta)$

ในกรณี (3) $\beta = (\alpha_1^*)$ จากทฤษฎี 4.6 เราสามารถสร้างออโตมาตาจำกัด M

$$M = (Q_1 \cup \{s, f\}, \Sigma, \delta_1 \cup \{(s, \varepsilon, s_1), (s, \varepsilon, f), (f, \varepsilon, s)\} \cup \{(q, \varepsilon, f) \mid q \in F_1\}, s, \{f\})$$

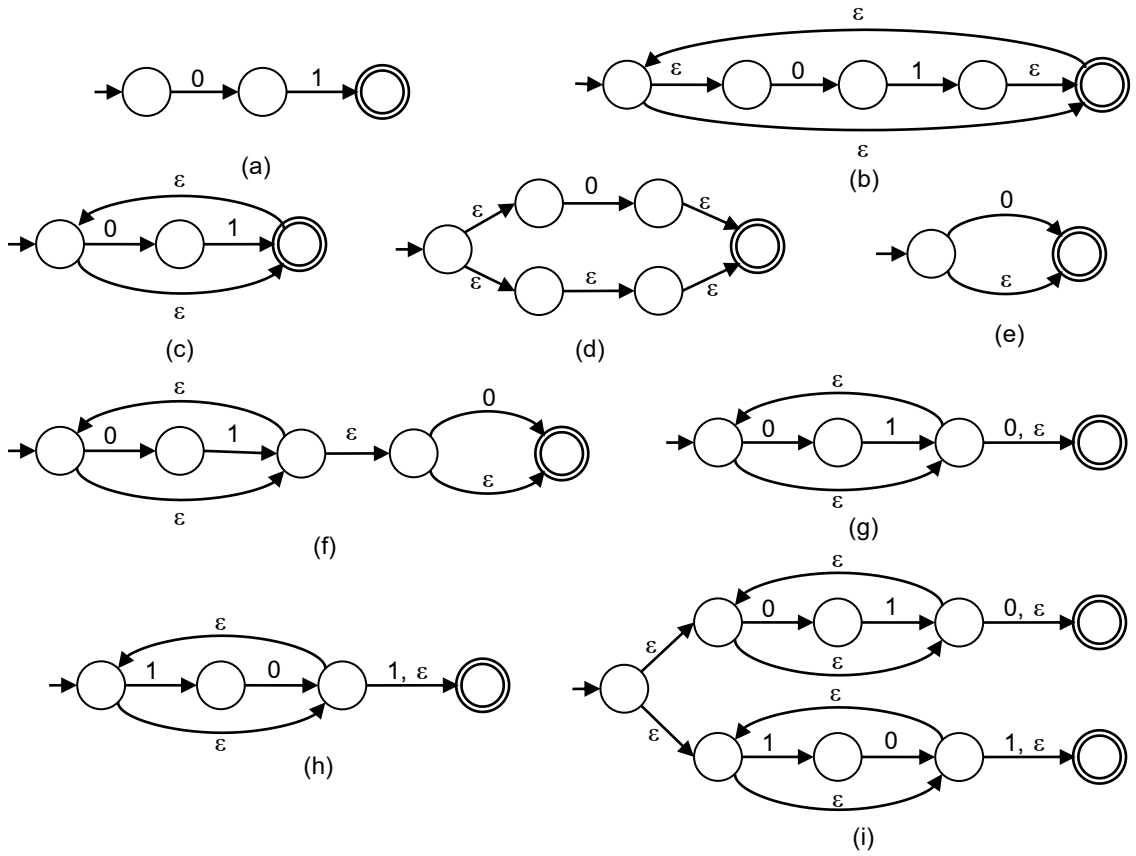
ซึ่ง $L(M) = L(\alpha_1)^*$ จากนิยาม 5.2 $(L(\alpha_1))^* = L(\alpha_1^*) = L(\beta)$ นั่นคือมีออโตมาตาจำกัดที่ยอมรับ $L(\beta)$

ดังนั้นเราสรุปได้ว่า สำหรับภาษาปกติ L ใดๆ จะมีออโตมาตาจำกัดที่ยอมรับ L

ทฤษฎี 5.1 แสดงวิธีสร้างออโตมาตาจำกัดเล็กๆ ที่ยอมรับสตริงที่แทนด้วยนิพจน์ปกติพื้นฐาน ได้แก่ สตริงว่าง สัญลักษณ์ในอักขระ หรือ ไม่รับสตริงใดๆ เลย และนำออโตมาตาจำกัดเล็กๆ นี้มาประกอบกัน ตัวอย่างต่อไปนี้แสดงการสร้างออโตมาตาจำกัดตามวิธีที่ใช้ในการพิสูจน์ทฤษฎี 5.1

ตัวอย่าง 5.4 เราสามารถสร้างออโตมาตาจำกัดที่ยอมรับสตริงที่แทนด้วยนิพจน์ปกติ ((01)* (0∪ε)) ∪ ((10)* (1∪ε)) ในตัวอย่าง 5.3 ได้ดังแสดงในรูป 5.2 เราจะพิจารณา ((01)* (0∪ε)) ก่อน

รูป 5.2 (a) แสดงออโตมาตาที่ยอมรับสตริงที่แทนด้วยนิพจน์ปกติ (01) ซึ่งได้จากการต่อกันของออโตมาตาที่รับ 0 และออโตมาตาที่รับ 1 รูป 5.2 (b) แสดงออโตมาตาที่ยอมรับสตริงที่แทนด้วยนิพจน์ปกติ (01)* ซึ่งสร้างจากการเพิ่มการเปลี่ยนสถานะด้วยสตริงว่างในออโตมาตาในรูป 5.2 (a) ออโตมาตานี้สามารถปรับให้เล็กลงได้ดังรูป 5.2 (c) รูป 5.2 (d) แสดงออโตมาตาที่ยอมรับสตริงที่แทนด้วยนิพจน์ปกติ (0∪ε) ซึ่งสามารถปรับให้เล็กลงได้ดังรูป 5.2 (e) รูป 5.2 (f) แสดงออโตมาตาที่ยอมรับสตริงที่แทนด้วยนิพจน์ปกติ ((01)* (0∪ε)) ซึ่งสร้างจากการเพิ่มการต่อออโตมาตาในรูป 5.2 (d) และ (e) ออโตมาตานี้สามารถสามารถปรับให้เล็กลงได้ดังรูป 5.2 (g)



รูป 5.2 การสร้างออโตมาตาที่ยอมรับสตริงที่แทนด้วยนิพจน์ปกติ ((01)* (0∪ε)) ∪ ((10)* (1∪ε))

ทำนองเดียวกัน เราสามารถสร้างออโตมาตาที่ยอมรับสตริงที่แทนด้วยนิพจน์ปกติ ((10)* (1∪ε)) ได้ดังรูป 5.2 (h) จากนั้น เราเชื่อมออโตมาตาในรูป 5.2 (g) และ (h) ด้วยการรวมดังแสดงในรูป 5.2 (i)

ซึ่งเป็นออโตมาตาที่ยอมรับสตริงที่แทนด้วยนิพจน์ปกติ ((01)* (0∪ε)) ∪ ((10)* (1∪ε)

จากนี้ เราจะพิจารณาส่วนที่สองของการพิสูจน์ว่าภาษาที่ยอมรับได้ด้วยออโตมาตาจำกัดว่าเป็นภาษาปกติโดยสร้างนิพจน์ปกติจากออโตมาตาจำกัดในทฤษฎี 5.2

ทฤษฎี 5.2 สำหรับภาษา L ใดๆ ถ้ามีออโตมาตาจำกัดที่ยอมรับ L แล้ว L เป็นภาษาปกติ

พิสูจน์

กำหนด L เป็นภาษาใดๆ บนอักขระ Σ และ $M = (Q, \Sigma, \delta, q_1, F)$ เป็นออโตมาตาจำกัดที่ยอมรับ L เราจะต้องพิสูจน์ว่ามีนิพจน์ปกติ α ที่ $L(\alpha) = L(M)$

จากนิยาม 4.4 สตริงที่ M ยอมรับคือสตริง ω ที่ $(q_1, \omega) \vdash^*_M (f, \varepsilon)$ โดยที่ $f \in F$

ถ้าให้ $Q = \{q_1, q_2, \ldots, q_n\}$ และ

$r(i, j, k)$ แทนเซตของสตริงที่ทำให้ M เปลี่ยนจากสถานะ q_i ไปยังสถานะ q_j โดยที่*ผ่าน* สถานะ $q_1, q_2, \ldots, q_k$ แต่ไม่*ผ่าน* q_i และ q_j เมื่อ $1 \le i, j, k \le n$ และ $0 \le k \le n$

ดังนั้น $L(M) = \cup_{q_j \in F} r(1, j, n)$

$L(M)$ สร้างจาก $r(1, j, n)$ ดังนั้นเราจะสร้างนิพจน์ปกติเพื่ออธิบาย $r(i, j, k)$ เมื่อ $1 \le i, j, k \le n$

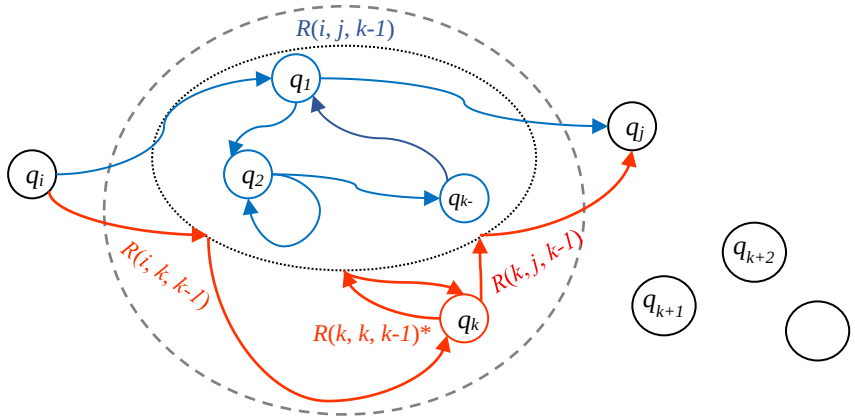
กำหนดให้ $R(i, j, k)$ แทนนิพจน์ปกติที่อธิบาย $r(i, j, k)$

จาก $r(i, j, k)$ ที่กำหนดไว้ เราจะได้ว่า

$r(i, j, k) = r(i, j, k\text{-}1) \cup r(i, k, k\text{-}1)\ r(k, k, k\text{-}1)^*\ r(k, j, k\text{-}1)$ เมื่อ $1 \le i, j, k \le n$

จากที่แสดงในรูป 5.3 เราจะได้ว่า

$R(i, j, k) = R(i, j, k\text{-}1) \cup R(i, k, k\text{-}1)\ R(k, k, k\text{-}1)^*\ R(k, j, k\text{-}1)$ เมื่อ $1 \le i, j, k \le n$



รูป 5.3 การสร้างนิพจน์ปกติ $R(i, j, k)$

นอกจากนั้นเรากำหนดให้ $r(i, j, 0)$ แทนเซตของสตริงที่ทำให้ M เปลี่ยนจากสถานะ q_i ไปยังสถานะ q_j โดยที่ผ่านไม่ผ่านสถานะใดเลย ดังนั้น

$r(i, j, 0) = \{a \in \Sigma \mid q_j \in \delta(q_i, a)\}$ และ

$r(i, i, 0) = \{a \in \Sigma \mid q_i \in \delta(q_i, a)\} \cup \{\varepsilon\}$

ซึ่งทำให้ได้

$R(i, j, 0) = \cup_{q_j \in \delta(q_i, a)} (a)$ และ $R(i, i, 0) = \cup_{q_i \in \delta(q_i, a)} (a) \cup \varepsilon$

จาก $R(i, j, k) = R(i, j, k\text{-}1) \cup R(i, k, k\text{-}1)\ R(k, k, k\text{-}1)^*\ R(k, j, k\text{-}1)$ และ $R(i, j, 0) = \cup_{q_j \in \delta(q_i, a)} a$ เราจะพิสูจน์ว่า $R(i, j, k)$ เป็นนิพจน์ปกติโดยใช้วิธีอุปนัย

<u>ขั้นตอนฐานหลัก</u>

กำหนด $k = 0$ เราจะเห็นว่า สำหรับ $1 \le i, j \le n$ $R(i, j, k) = a_1 \cup a_2 \cup \ldots \cup a_m$ เมื่อ $i \neq j$ และ $R(i, j, k) = a_1 \cup a_2 \cup \ldots \cup a_m \cup \varepsilon$ เมื่อ $i = j$ ดังนั้น $R(i, j, k)$ จึงเป็นนิพจน์ปกติ

<u>ขั้นตอนอุปนัย</u>

กำหนดสมมุติฐานอุปนัยว่าสำหรับ $k < m$ ใด ๆ $R(i, j, k)$ เป็นนิพจน์ปกติ สำหรับ $1 \le i, j \le n$ เราจะพิสูจน์ว่า $R(i, j, m)$ เป็นนิพจน์ปกติ จากสมมุติฐานอุปนัย เราจะได้ว่า $R(i, j, m\text{-}1)$ เป็นนิพจน์ปกติ สำหรับ $1 \le i, j \le n$ เนื่องจาก $R(i, j, m) = R(i, j, m\text{-}1) \cup R(i, m, m\text{-}1)\ R(m, m, m\text{-}1)^*\ R(m, j, m\text{-}1)$ ดังนั้น $R(i, j, m)$ เป็นนิพจน์ปกติด้วย

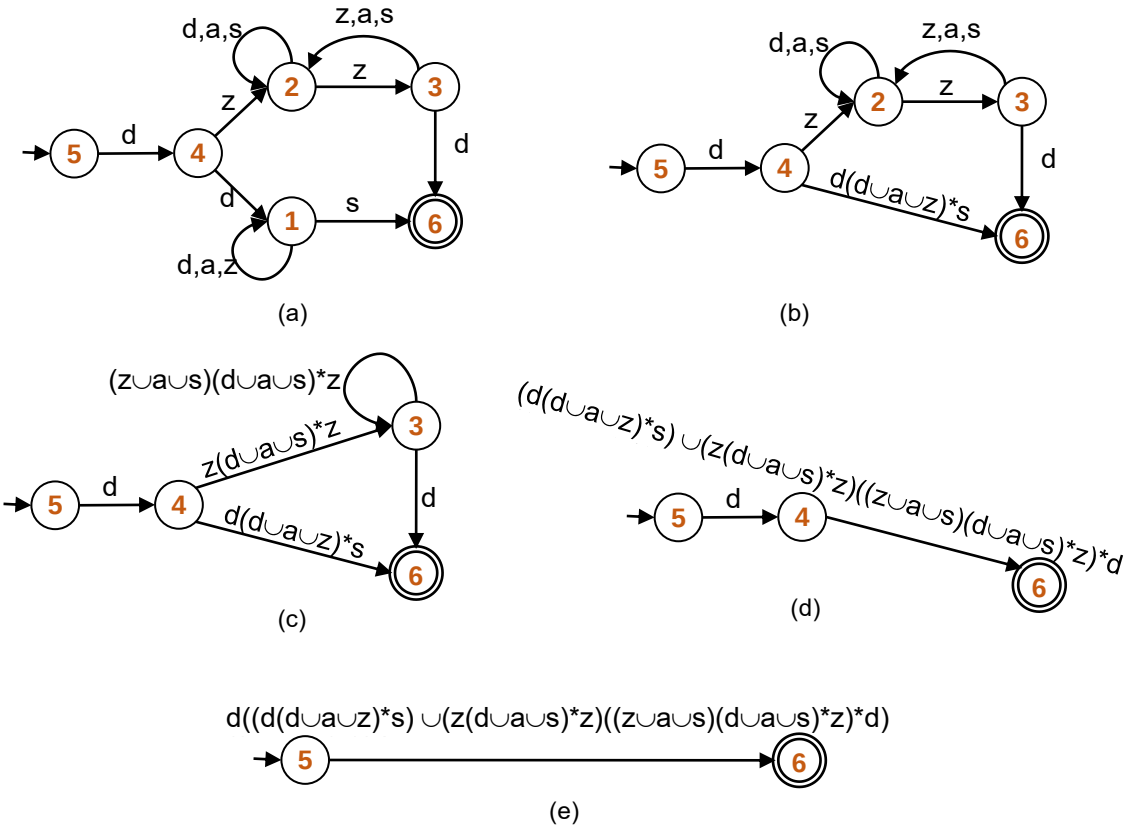
ดังนั้นเราสรุปได้ว่ามีนิพจน์ปกติ $\alpha = \cup_{q_j \in F} R(1, j, n)$ ที่ทำให้ $L(\alpha) = L(M)$ ดังนั้น $L(M)$ เป็นภาษาปกติ

ทฤษฎี 5.2 แสดงวิธีสร้างนิพจน์ปกติที่แทนสตริงที่ยอมรับด้วยออโตมาตาจำกัดที่กำหนดซึ่งสามารถทำได้โดยคำนวณหา $R(i, j, k)$ จาก $R(i, j, k) = R(i, j, k\text{-}1) \cup R(i, k, k\text{-}1)\ R(k, k, k\text{-}1)^*\ R(k, j, k\text{-}1)$ โดยเริ่มจาก k = 0, 1, 2,... ซึ่งวิธีนี้เหมาะสำหรับนำไปพัฒนาโปรแกรม แต่เราสามารถมองเป็นการยุบสถานะในแผนภาพเปลี่ยนสถานะโดยเรียงจากสถานะ $q_1, q_2, \ldots$ ตามลำดับ และ $R(i, j, 1)$ ได้จากการยุบสถานะ q_1 เป็นต้น ตัวอย่างต่อไปนี้แสดงการสร้างนิพจน์ปกติตามวิธีที่ใช้ในการพิสูจน์ทฤษฎี 5.2 ควบคู่กับการยุบสถานะ

<u>ตัวอย่าง 5.5</u> เราสามารถสร้างนิพจน์ปกติที่แทนสตริงที่ยอมรับด้วยออโตมาตาจำกัดในรูป 5.4 (a) ตามวิธีที่ใช้ในการพิสูจน์ทฤษฎี 5.2 ในออโตมาตานี้ d, z, s, a แทน /, *, newline, อักขระอื่น ๆ ตามลำดับ ออโตมาตานี้ยอมรับคอมเมนท์ 2 แบบในภาษาโปรแกรม คือ /* ... */ และ // ... newline ในตัวอย่างนี้ เราใช้สถานะ i แทนสถานะ q_i และใช้ $R(i, j, k)$ เป็นนิพจน์ปกติแทนเซตของสตริงที่ทำให้ออโตมาตาเปลี่ยนจากสถานะ i ไปยังสถานะ j โดยที่*ผ่าน*สถานะ $1, 2, \ldots, k$ แต่ไม่ผ่านสถานะ i และ j

$R(i, j, 0)$ เป็นนิพจน์ปกติแทนเซตของสตริงที่ทำให้ออโตมาตาเปลี่ยนจากสถานะ i ไปยังสถานะ j โดยที่ผ่านสถานะ 0 ได้ เนื่องจากเราไม่มีสถานะ 0 การเปลี่ยนสถานะที่สนใจนี้ต้องไม่ผ่านสถานะใดเลย ดังนั้น สตริงที่แทนด้วยนิพจน์ปกติ $R(i, j, 0)$ เป็นสตริงที่แสดงในแผนภาพเปลี่ยนสถานะจากสถานะ i ไปยังสถานะ j โดยตรง ดังนั้น แผนภาพเปลี่ยนสถานะในรูป 5.4 (a) จึงแสดง $R(i, j, 0)$

นอกจากนั้น จะเห็นว่า ถ้าไม่มีการเปลี่ยนจากสถานะ i ไปยังสถานะ j ได้แล้ว $R(i, j, 0) = \varnothing$



รูป 5.4 การสร้างนิพจน์ปกติที่แทนสตริงที่ยอมรับด้วยออโตมาตาจำกัด

ทำนองเดียวกัน สำหรับสถานะ i และ j ใด ๆ $R(i, j, 1)$ เป็นนิพจน์ปกติแทนเซตของสตริงที่ทำให้ออโตมาตาเปลี่ยนจากสถานะ i ไปยังสถานะ j โดยที่*ผ่าน*สถานะ 1 ได้เพียงสถานะเดียว เมื่อยุบสถานะ 1 จากรูป 5.4 (a) จะเห็นว่า การเปลี่ยนจากสถานะ 4 ไปยังสถานะ 6 โดยที่ผ่านสถานะ 1 ได้เพียงสถานะเดียวต้องใช้สตริง d(d∪a∪z)*s ที่ทำให้เปลี่ยนจากสถานะ 4 ไปยังสถานะ 1 และซ้ำอยู่ที่สถานะ 1 แล้วเปลี่ยนไปยังสถานะ 6 ดังนั้น

$R(4, 6, 1)$ = d(d∪a∪z)*s

แต่การเปลี่ยนสถานะอื่นไม่มีการเปลี่ยนแปลงจากการยุบสถานะ 1 หากพิจารณาตามการพิสูจน์ทฤษฎี 5.2

$R(4, 6, 1) = R(4, 6, 0) \cup R(4, 1, 0)\, R(1, 1, 0)^*\, R(1, 6, 0)$

$R(i, j, 0)$ คือนิพจน์ที่แสดงในการเปลี่ยนสถานะในรูป 5.4 (a) และ

$R(4, 6, 0) = \varnothing$, $R(4, 1, 0)$ = d

$R(1, 1, 0)$ = (d∪a∪z)

$R(1, 6, 0)$ = s

เราจะได้ $R(4, 6, 1)$ = d(d∪a∪z)*s เช่นกัน

เมื่อ $i \neq 4$ และ $j \neq 6$ เราได้

$R(i, 1, 0) = \varnothing$ หรือ $R(1, j, 0) = \varnothing$

ดังนั้น เมื่อ $i \neq 4$ และ $j \neq 6$

$R(i, j, 1) = R(i, j, 0) \cup R(i, 1, 0)\, R(1, 1, 0)^*\, R(1, j, 0) = R(i, j, 0) \cup \varnothing = R(i, j, 0)$

รูป 5.4 (b) แสดง $R(i, j, 1)$

ขั้นต่อมา เราคำนวณ $R(i, j, 2)$ จาก $R(i, j, 1)$ ซึ่งเป็นการยุบสถานะ 2 จากรูป 5.4 (b) ดังนั้น การเปลี่ยนสถานะจาก 4 ไปยังสถานะ 3 โดยที่ผ่านสถานะ 2 ได้เพียงสถานะเดียวต้องใช้สตริง z(d∪a∪s)*z ที่ทำให้เปลี่ยนจากสถานะ 4 ไปยังสถานะ 2 ด้วย z และซ้ำอยู่ที่สถานะ 2 ด้วย (d∪a∪s) แล้วเปลี่ยนไปยังสถานะ 3 ด้วย z

ดังนั้น $R(4, 3, 2)$ = z(d∪a∪s)*z

(ระวังว่าเราไม่อนุญาตให้เปลี่ยนจากสถานะ 3 ไปยังสถานะ 2 อีกครั้ง เพราะเราไม่อนุญาตให้ผ่านสถานะ 3)

นอกจากนั้น การเปลี่ยนสถานะจาก 3 ไปยังสถานะ 3 โดยที่ผ่านสถานะ 2 ได้เพียงสถานะเดียวต้องใช้สตริง (z∪a∪s)(d∪a∪s)*z ที่ทำให้เปลี่ยนจากสถานะ 3 ไปยังสถานะ 2 ด้วย (z∪a∪s) และซ้ำอยู่ที่สถานะ 2 ด้วย (d∪a∪s) แล้วเปลี่ยนไปยังสถานะ 3 ด้วย z

ดังนั้น $R(3, 3, 2)$ = (z∪a∪s)(d∪a∪s)*z

แต่การเปลี่ยนสถานะอื่นไม่มีการเปลี่ยนแปลงจากการยุบสถานะ 2 หากพิจารณาตามการพิสูจน์ทฤษฎี 5.2 จะได้ว่า

$R(4, 3, 2) = R(4, 3, 1) \cup R(4, 2, 1)\, R(2, 2, 1)^*\, R(2, 3, 1)$ และ

$R(3, 3, 2) = R(3, 3, 1) \cup R(3, 2, 1)\, R(2, 2, 1)^*\, R(2, 3, 1)$

เราใช้ $R(i, j, 1)$ ในรูป 5.4 (b) จะได้

$R(4, 3, 2)$ = z(d∪a∪s)*z และ

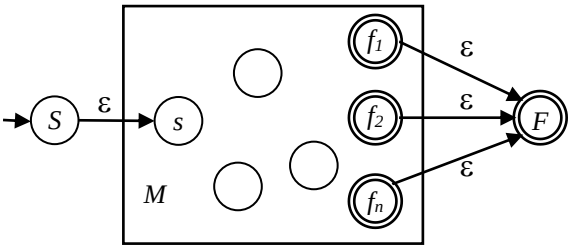
$R(3, 3, 2)$ = (z∪a∪s)(d∪a∪s)*z

รูป 5.4 (c) แสดง $R(i, j, 2)$

เมื่อทำตามขั้นตอนนี้ต่อไป จะได้ $R(i, j, 3)$ และ $R(i, j, 4)$ ดังรูป 5.4 (d) และ (e) ตามลำดับ และ $R(5, 6, 4)$ = d((d(d∪a∪z)*s) ∪(z(d∪a∪s)*z)((z∪a∪s)(d∪a∪s)*z)*d) เป็นนิพจน์ปกติแทนเซตของสตริงที่ยอมรับด้วยออโตมาตานี้

สิ่งที่ต้องการจากการสร้างนิพจน์ปกติด้วยการยุบสถานะที่แสดงในตัวอย่าง 5.5 คือนิพจน์ปกติของสตริงที่ทำให้ออโตมาตาเปลี่ยนสถานะจากสถานะเริ่มต้นและไปหยุดที่สถานะสิ้นสุด ดังนั้นหลังจากยุบสถานะแล้ว เราต้องการให้เหลือเพียงสถานะเริ่มต้นและสิ้นสุด แต่หากมีการเปลี่ยนสถานะผ่านสถานะเริ่มต้นและสถานะสิ้นสุดได้ก็จำเป็นต้องพิจารณาด้วย ดังนั้นเพื่อหลีกเลี่ยงการยุบสถานะเริ่มต้นและ

สิ้นสุด เราอาจเพิ่มสถานะใหม่เข้าไปให้เป็นสถานะเริ่มต้นและสถานะสิ้นสุดใหม่และเชื่อมสถานะใหม่นี้กับออโตมาตาเดิม นั่นคือเพิ่มการเปลี่ยนสถานะด้วยสตริงว่างจากสถานะเริ่มต้นใหม่ไปยังสถานะเริ่มต้นเดิม และจากสถานะสิ้นสุดเดิมทั้งหมดไปยังสถานะสิ้นสุดใหม่ที่มีเพียงสถานะเดียวดังแสดงในรูป 5.5 ด้วยวิธีนี้ จะไม่มีการเปลี่ยนสถานะใดที่ผ่านสถานะเริ่มต้นและสถานะสิ้นสุดที่เพิ่มเข้าไปใหม่ได้ เราจึงไม่ต้องยุบสถานะเริ่มต้นและสถานะสิ้นสุด นอกจากนั้น หากลำดับการยุบสถานะต่างกันแล้ว อาจทำให้ได้นิพจน์ปกติต่างกัน



รูป 5.5 การเปลี่ยนสถานะเริ่มต้นและสถานะสิ้นสุดก่อนยุบสถานะ

5.3 สมบัติปิดของคลาสของภาษาปกติ

จากหัวข้อ 5.2 เราได้พิสูจน์ไปแล้วว่าคลาสของภาษาปกติคือคลาสของภาษาที่ยอมรับได้ด้วยออโตมาตาจำกัดด้วย ดังนั้นจึงสามารถสรุปได้ด้วยว่าคลาสของภาษาปกติมีสมบัติปิดภายใต้ (1) การรวม (2) การต่อกัน (3) การเติมเต็ม (4) การซ้ำ และ (5) การร่วม ตามทฤษฎี 4.3 - 4.7 อย่างไรก็ตาม เราสามารถพิสูจน์สมบัติปิดของคลาสของภาษาปกติภายใต้การรวม การต่อกัน และการซ้ำได้โดยใช้นิยาม 5.1 และ 5.2 ได้ดังนี้

ทฤษฎี 5.3 คลาสของภาษาปกติมีสมบัติปิดภายใต้การรวม การต่อกัน และการซ้ำ

พิสูจน์

กำหนด L_A และ L_B เป็นภาษาปกติ ดังนั้นจะมีนิพจน์ปกติ α_A และ α_B ที่ $L_A = L(\alpha_A)$ และ $L_B = L_B(\alpha_B)$ ดังนั้นจากนิยาม 5.1 และ 5.2 $L_A \cup L_B$, $L_A L_B$ และ L_A^* อธิบายได้ด้วยนิพจน์ปกติ $(\alpha_A \cup \alpha_B)$, $(\alpha_A \alpha_B)$ และ (α_B^*) ตามลำดับ นั่นคือคลาสของภาษาปกติมีสมบัติปิดภายใต้การรวม การต่อกัน และการซ้ำ

5.4 การประยุกต์ใช้นิพจน์ปกติ

การทำงานเกี่ยวกับข้อความส่วนใหญ่ต้องมีการแยกข้อความเป็นส่วน ๆ ตามที่ต้องการ เรามักใช้รูปแบบของข้อความเพื่อกำหนดความแตกต่างของข้อความแต่ละส่วน เช่น แฟ้มสมุดโทรศัพท์

ประกอบด้วยข้อความที่เป็นชื่อนามสกุลประกอบด้วยตัวอักษรอย่างน้อยหนึ่งตัว แล้วเว้นวรรค แล้วต่อด้วยตัวอักษรอีกอย่างน้อยหนึ่งตัว เบอร์โทรศัพท์ประกอบด้วยตัวเลข 9 หรือ 10 ตัวติดกันโดยไม่มีการเว้นวรรค รูปแบบเหล่านี้สามารถอธิบายได้ด้วยนิพจน์ปกติ [14] ดังนั้นภาษาโปรแกรมหลายภาษาจะอนุญาตให้ระบุส่วนที่ต้องการหาจากข้อความด้วยนิพจน์ปกติ นอกจากนั้นภาษาโปรแกรมเหล่านี้ยังเพิ่มกลุ่มของสัญลักษณ์ที่ต้องการใช้บ่อย และเพิ่มตัวดำเนินการจากตัวดำเนินการพื้นฐานที่ได้กล่าวไปในบทนี้เพื่ออำนวยความสะดวกแก่ผู้เขียนโปรแกรม [4]

ในหัวข้อนี้เราจะแสดงตัวอย่างของนิพจน์ปกติที่ใช้ในภาษาไพธอนที่มีโมดูล re สำหรับนิพจน์ปกติ ผู้ใช้ต้องใช้คำสั่ง `import re` เพื่อใช้งานโมดูลนี้ ตาราง 5.1 แสดงนิพจน์ปกติที่แทนสัญลักษณ์บางชุดในภาษาปกติ เช่น ตัวเลขทุกตัว ตัวหนังสือทุกตัว การกำหนดนิพจน์ปกติเพิ่มมาเพื่อแทนสัญลักษณ์บางชุดที่ใช้บ่อยช่วยให้เขียนนิพจน์ปกติสั้นลงได้ ตาราง 5.2 แสดงตัวดำเนินการที่ใช้ในนิพจน์ปกติซึ่งมีตัวดำเนินการเพิ่มมาจากที่ระบุในนิยาม 5.1 เช่น การระบุจำนวนครั้งของการซ้ำได้ ในตาราง 5.1 และตาราง 5.2 เราให้ `x`, `y`, `z` แทนตัวอักขระ `r`, `s`, `t` เป็นนิพจน์ปกติใด และ `i`, `j` แทนจำนวนเต็มบวก

ตาราง 5.1 นิพจน์ปกติในภาษาไพธอนที่แทนสัญลักษณ์ในภาษาปกติ

รูปแบบในภาษาไพธอน	ความหมาย
`x`	ตัวอักขระนั้น
`\b`, `\B`	สตริงว่างที่ต้นคำ,สตริงว่างที่อื่นนอกจากต้นคำ
`\d`	ตัวเลข 0, 1, …, 9
`\D`	ตัวอักขระที่ไม่ใช่ตัวเลข 0, 1, …, 9
`\s`	ตัวอักขระที่เป็นช่องว่าง เช่น tab, space
`.`	ตัวอักขระที่ไม่ใช่อักขระขึ้นบรรทัดใหม่ (newline)

ตาราง 5.2 นิพจน์ปกติในภาษาไพธอนที่แทนตัวดำเนินการในภาษาปกติ

รูปแบบในภาษาไพธอน	ความหมาย
`r s t`	การต่อกัน
`[xyz]`	การรวม
`[r | s | t]`	การรวม
`r*`	การซ้ำกี่ครั้งก็ได้
`r+`	การซ้ำอย่างน้อยหนึ่งครั้ง
`r?`	`r` หรือสตริงว่าง
`r{i}`	การซ้ำ `i` ครั้ง
`r{i, j}`	การซ้ำไม่น้อยกว่า `i` ครั้งและไม่เกิน `j` ครั้ง

จากโปรแกรมในรูป 5.6 บรรทัด 04 แสดงตัวแปร `pattern` ที่เก็บสตริงมีใช้อธิบายนิพจน์ปกติที่

ต้องการหาด้วยเมธอด search ในบรรทัด 05 สตริงในตัวแปร pattern ระบุกลุ่ม (group) ในนิพจน์ปกติด้วย (…) จากค่า '(\d{7})\s([A-Z][a-z]*)\s([A-Z][a-z]*)\s(\d\.\d+)' ในตัวแปร pattern ซึ่งมีความหมายดังนี้

```
01: import re
02:
03: text='5534567 John Smith 3.21'
04: pattern='(\d{7})\s([A-Z][a-z])*)\s([A-Z][a-z]*)\s(\d\.\d+)'
05: re_obj = re.search(pattern,text)
06: id = re_obj.group(1)
07: fname=re_obj.group(2)
08: lname=re_obj.group(3)
09: grade=re_obj.group(4)
10: print(lname, fname, grade)
```

รูป 5.6 โปรแกรมภาษาไพธอนแสดงตัวอย่างการใช้นิพจน์ปกติ

กลุ่มที่ 1 คือ (\d{7}) แทนนิพจน์ปกติ (0∪1∪2∪…∪9)^7 หรือ (0∪1∪2∪…∪9) (0∪1∪2∪…∪9) (0∪1∪2∪ … ∪9) (0∪1∪2∪ … ∪9) (0∪1∪2∪ … ∪9) (0∪1∪2∪ … ∪9) ซึ่งเป็นสตริงที่ประกอบด้วยตัวเลขต่อกัน 7 ตัว

กลุ่มที่ 2 และ 3 คือ ([A-Z][a-z]*) แทนนิพจน์ปกติ (A∪B∪…∪Z) (a∪b∪…∪z)* ซึ่งเป็นสตริงที่ประกอบด้วยตัวอักษรตัวใหญ่หนึ่งตัวตามด้วยตัวอักษรตัวเล็กกี่ตัวก็ได้

กลุ่มที่ 4 คือ (\d\.\d+) แทนนิพจน์ปกติ (0∪1∪2∪…∪9). (0∪1∪2∪…∪9) (0∪1∪2∪…∪9)* ซึ่งเป็นสตริงที่ประกอบด้วยตัวเลขหนึ่งตัว ตามด้วยจุด แล้วต่อด้วยตัวเลขอย่างน้อยหนึ่งตัว

ส่วน \s เป็นสตริงที่เป็นช่องว่าง แต่ไม่มี (…) ครอบจึงไม่นับเป็นกลุ่ม ดังนั้น ในโปรแกรมนี้ re_obj.group(1) จึงได้ '5534567' ไปเก็บในตัวแปร id, re_obj.group(2) จึงได้ 'John' ไปเก็บในตัวแปร fname, re_obj.group(3) จึงได้ 'Smith' ไปเก็บในตัวแปร lname, และ re_obj.group(4) จึงได้ '3.21' ไปเก็บในตัวแปร grade เมื่อ print(lname, fname, grade) จะได้ผลลัพธ์เป็น Smith John 3.21

บทที่ 6

ออโตมาตาชนิดอื่น ๆ

บทที่ 6 ออโตมาตาชนิดอื่น ๆ

จากที่ศึกษาไปในบทที่ผ่านมาแล้วจะเห็นว่าออโตมาตาจำกัดมีข้อจำกัดในการทำงาน เราจะศึกษาว่ามีภาษาใดที่ออโตมาตาจำกัดไม่สามารถตรวจสอบได้ ในหัวข้อ 6.1 เรียกว่าเป็นภาษาไม่ปกติ (non-regular language) จากนั้นหัวข้อ 6.2 จะอธิบายออโตมาตาอีกชนิดหนึ่งที่เรียกว่าออโตมาตาแบบพูชดาวน์ (pushdown automata) ที่เป็นตัวแบบทางคณิตศาสตร์ คล้ายกับออโตมาตาจำกัดแต่มีสแตก (stack) สำหรับเก็บข้อมูลเพิ่มขึ้น ต่อมาในหัวข้อ 6.3 จะอธิบายเครื่องจักรทูวริง (Turing machines) ที่เป็นตัวแบบทางคณิตศาสตร์ซึ่งใช้อธิบายการทำงานของคอมพิวเตอร์ที่ใช้กันอยู่ในปัจจุบันนี้

6.1 ภาษาไม่ปกติ

ภาษาไม่ปกติ (Non-regular Languages) คือ ภาษาที่ไม่มีออโตมาตาจำกัดยอมรับได้ หรือ ไม่มีนิพจน์ปกติที่อธิบายภาษานั้นได้ เราไม่สามารถบอกว่าไม่มีออโตมาตาจำกัดที่ยอมรับภาษาเมื่อเราไม่สามารถสร้างออโตมาตาจำกัดได้ เพราะเราอาจหาไม่ได้ ทั้งที่มีออโตมาตาที่ยอมรับภาษานั้นก็ได้ อย่างไรก็ตามเราสามารถใช้บทตั้งปั๊มปิง (pumping lemma) ที่จะกล่าวต่อไปนี้ในการพิสูจน์ว่าไม่มีออโตมาตาจำกัดที่รับภาษานั้น แต่เราจะไม่แสดงการพิสูจน์บทตั้งปั๊มปิงในที่นี่

<u>บทตั้งปั๊มปิง (pumping lemma)</u>

ถ้า L เป็นภาษาปกติแล้วจะมีจำนวนเต็ม $n \geq 0$ ที่ทำให้สำหรับสตริง x ใดๆ ใน L ที่มีความยาวมากกว่าหรือเท่ากับ n จะมีสตริง u, v และ w ซึ่ง

(1) $x = uvw$,

(2) $|uv| \leq n$,

(3) $|v| > 0$, *และ*

(4) $\forall\, k \geq 0\ uv^k w \in L$

บทตั้งนี้บอกว่าถ้าสตริงที่อยู่ในภาษาปกติมีความยาวมากกว่าค่าหนึ่ง (สมมติให้เป็นจำนวนสถานะที่ใช้ในออโตมาตาจำกัดเพื่อยอมรับภาษานี้) ออโตมาตาจำกัดที่ยอมรับสตริงนี้ได้ต้องมีส่วนของสตริงที่ทำให้ออโตมาตานี้วิ่งผ่านสถานะชุดหนึ่งวนซ้ำอยู่ ทั้งนี้เป็นเพราะเมื่อออโตมาตาจำกัดเชิงกำหนดทำงานบนสตริงที่มีความยาว n จะต้องมีการเปลี่ยนสถานะ n ครั้ง หากออโตมาตามีน้อยกว่า n สถานะจะมีบางส่วนของสตริงที่ทำให้เปลี่ยนสถานะวนซ้ำอยู่ในสถานะชุดหนึ่ง ดังนั้นสตริงย่อยชุดนั้นจึงมีการ

ซ้ำอยู่ในสตริงดังแสดงในสตริงย่อย v ในสตริง x ในบทตั้งนี้

เราไม่สามารถนำบทตั้งนี้ไปใช้แสดงว่าภาษาหนึ่งไม่เป็นภาษาปกติโดยตรงได้ แต่จะใช้ประพจน์แย้งสลับที่ (contrapositive) ของบทตั้งนี้ที่ว่า

ถ้า สำหรับจำนวนเต็ม $n \geq 0$ จำนวนใดๆ จะมีสตริง x ใน L ที่มีความยาวมากกว่าหรือเท่ากับ n และสตริง u, v และ w ใดๆ ซึ่ง

(1) $x \neq uvw$,

(2) $|uv| > n$,

(3) $|v| \leq 0$, *หรือ*

(4) $\exists k \geq 0\ uv^k w \notin L$ **แล้ว** L ไม่เป็นภาษาปกติ

จากประพจน์แย้งสลับที่นี้เราสามารถพิสูจน์ว่าภาษา L ไม่เป็นภาษาปกติได้ดังนี้

1. กำหนด n เป็นจำนวนเต็มใดๆ โดยที่ $n \geq 0$
2. เลือกสตริง x ใน L สตริงหนึ่งที่มีความยาวมากกว่าหรือเท่ากับ n
3. หาทุกวิธีที่แบ่ง x เป็น 3 ส่วนคือ u, v, และ w (นั่นคือ $x=uvw$) โดยที่ $v \neq \varepsilon$ และ $|uv| \leq n$
4. สำหรับ u, v, และ w แต่ละชุดที่หามาจากข้อ 3 เราต้องหาค่า k ค่าหนึ่งที่ทำให้ $uv^k w$ ไม่อยู่ใน L
5. ถ้าเราหาค่า k ในข้อ 4 ได้สำเร็จแล้วเราสรุปได้ว่า L ไม่เป็นภาษาปกติ

จากขั้นตอนที่กล่าวมานี้เราจะสรุปได้ว่าภาษานั้นไม่เป็นภาษาปกติถ้าเงื่อนไขตรงตามข้อ 1 ถึงข้อ 4 ครบทุกข้อ อย่างไรก็ตาม หากเราไม่สามารถแสดงเงื่อนไขข้อ1 ถึง4 ได้ก็ยังไม่สามารถสรุปได้ว่าภาษานั้นเป็นภาษาปกติ ตัวอย่างต่อไปนี้แสดงการพิสูจน์ว่าภาษาที่กำหนดไม่เป็นภาษาปกติ

<u>ตัวอย่าง 6.1</u>: จงพิสูจน์ว่า $L_h = \{0^i 1^j \mid i \geq 0\}$ ไม่เป็นภาษาปกติ

(1) ให้ n เป็นจำนวนเต็มใดๆที่มากกว่าหรือเท่ากับ 0

(2) เราเลือกสตริง $x = 0^n 1^n$ จาก L_h (x มีความยาวมากกว่าหรือเท่ากับ n)

(3) เราแบ่ง x เป็น u, v, และ w โดยที่ $v \neq \varepsilon$ และ $|uv| \leq n$ ซึ่งมีวิธีแบ่งได้เพียงวิธีเดียวคือ $u = 0^i$ โดยที่ $0 \leq i < n$, $v = 0^j$ โดยที่ $0 < j \leq n$, และ $w = 0^l 1^n$ โดยที่ $i+j+l = n$
เราไม่พิจารณากรณีที่เหลือ คือ $v = 0^x 1^y$, $v = 1^x 0^y$, และ $v = 1^x$ ด้วยเหตุผลดังนี้
- ถ้า $v = 0^x 1^y$ หรือ $1^x 0^y$ แล้ว $uv^k w$ ไม่อยู่ใน L_h เพราะไม่อยู่ในรูปของ $0^i 1^j$
- ถ้า $v = 1^x$ แล้ว $|uv| > n$

(4) จาก $uv^k w = 0^i 0^{jk} 0^l 1^n$ จะเห็นได้ว่า $uv^k w$ ไม่อยู่ใน L_h เมื่อ $k > 1$ เพราะ $i+jk+l = n+(j-1)k \neq n$ เมื่อ $k > 1$

(5) ดังนั้นเราสรุปได้ว่า L_h ไม่เป็นภาษาปกติ

จากตัวอย่าง 6.1 จะเห็นว่ามีข้อควรระวังในการพิสูจน์ดังนี้ ในขั้นตอน (2) เราต้องเลือกสตริงหนึ่งซึ่งต้อง*อยู่ในภาษา*ด้วย ในขั้นตอน (3) เราต้องแบ่งสตริงนั้นให้*ครบทุกวิธี* และในขั้นตอน (4) เมื่อซ้ำส่วนของสตริงย่อย แล้วสตริงที่ได้ต้อง*ยังคงอยู่ในภาษาในทุกกรณี*ของสตริงย่อยจากขั้นตอน (3) ตัวอย่าง 6.2 แสดงการพิสูจน์สำหรับอีกภาษาหนึ่ง ให้สังเกตว่าเมื่อเลือกแบ่งสตริงเป็น 3 ส่วน สตริง 2 ส่วนแรกต้องยาวไม่เกิน n ดังนั้นสตริงส่วนที่ซ้ำ คือ v จะไม่สามารถมีสัญลักษณ์ *1* ได้เลย

ตัวอย่าง 6.2: จงพิสูจน์ว่า $L_m = \{0^i\ 1\ 0^i \mid i \geq 0\}$ ไม่เป็นภาษาปกติ

(1) กำหนด n เป็นจำนวนเต็มใดๆที่มากกว่าหรือเท่ากับ 0

(2) เราเลือกสตริง $x = 0^n 1\ 0^n$ จาก L_m (x มีความยาวมากกว่าหรือเท่ากับ n)

(3) เราแบ่ง x เป็น u, v, และ w โดยที่ $v \neq \varepsilon$ และ $|uv| \leq n$ ซึ่งมีวิธีแบ่งได้เพียงวิธีเดียวคือ $u = 0^i$ โดยที่ $0 \leq i < n$, $v = 0^j$ โดยที่ $0 < j \leq n$, และ $w = 0^l 1 0^n$ โดยที่ $i + j + l = n$ $(0 \leq l < n)$

(4) จาก u, v, และ w ที่กำหนด $uv^k w = 0^i 0^{jk} 0^l 1 0^n$ เราจะเห็นได้ว่า $uv^k w$ ไม่อยู่ใน L_m เมื่อ $k > 1$ เพราะ $i + jk + l = n + (j-1)k \neq n$ เมื่อ $k > 1$

(5) ดังนั้นเราสรุปได้ว่า L_m ไม่เป็นภาษาปกติ

ตัวอย่าง 6.3 แสดงการพิสูจน์ภาษาไม่ปกติอีกภาษาหนึ่ง

ตัวอย่าง 6.3: จงพิสูจน์ว่า $L_d = \{0^i\ 1^{2i} \mid i \geq 0\}$ ไม่เป็นภาษาปกติ

(1) กำหนด n เป็นจำนวนเต็มใดๆที่มากกว่าหรือเท่ากับ 0

(2) เราเลือกสตริง $x = 0^n\ 1^{2n}$ จาก L_d (x มีความยาวมากกว่าหรือเท่ากับ n)

(3) เราแบ่ง x เป็น u, v, และ w โดยที่ $v \neq \varepsilon$ และ $|uv| \leq n$ ซึ่งมีวิธีแบ่งได้เพียงวิธีเดียวคือ $u = 0^i$ โดยที่ $0 \leq i < n$, $v = 0^j$ โดยที่ $0 < j \leq n$, และ $w = 0^l\ 1^{2n}$ โดยที่ $i+j+l = n$ $(0 \leq l < n)$

(4) จาก u, v, และ w ที่กำหนด $uv^k w = 0^i 0^{jk} 0^l 1^{2n}$ เราจะเห็นได้ว่า $uv^k w$ ไม่อยู่ใน L_d เมื่อ $k > 1$ เพราะ $i + jk + l = n + (j-1)k \neq n$ เมื่อ $k > 1$

(5) ดังนั้นเราสรุปได้ว่า L_d ไม่เป็นภาษาปกติ

เราสามารถใช้สมบัติปิดของคลาสของภาษาปกติเพื่อพิสูจน์ว่าภาษาหนึ่งไม่เป็นภาษาปกติโดยการใช้ประพจน์แย้งสลับที่ดังนี้

ถ้า L_A เป็นภาษาปกติ แต่ $L_A \cup L_B$, $L_A \cap L_B$ หรือ $L_A \cdot L_B$ ไม่เป็นภาษาปกติแล้ว L_B ไม่เป็นภาษาปกติ

ถ้า $\overline{L}_A$ หรือ L_A^* ไม่เป็นภาษาปกติแล้ว L_A ไม่เป็นภาษาปกติ

ในกรณีแรก เราเลือกให้ L_A เป็นภาษาปกติเพื่อที่จะสรุปว่า L_B ไม่เป็นภาษาปกติ นั่นคือหากเราต้องการพิสูจน์ว่า L_B ไม่เป็นภาษาปกติเราก็จะเลือกภาษาปกติ L_A ที่ทำการรวม การร่วม หรือ การต่อกันกับ L_B แล้วได้ภาษาที่ไม่ปกติที่พิสูจน์ได้ง่ายกว่า

ในกรณีที่สอง จากสมบัติปิดภายใต้การเติมเต็มและการซ้ำของภาษาปกติเราสามารถบอกได้ว่าถ้าส่วนเติมเต็มของภาษาหนึ่งไม่เป็นภาษาปกติแล้วภาษานั้นไม่เป็นภาษาปกติด้วย เราสามารถสรุปในทำนองเดียวกันสำหรับการซ้ำได้ด้วย

ดังนั้นถ้าเราเลือก L_B ที่เป็นภาษาปกติแล้วเราสามารถสรุปได้ว่า L_A ไม่เป็นภาษาปกติโดยพิสูจน์ว่า $L_A \cup L_B$, $L_A \cap L_B$ หรือ $L_A \cdot L_B$ ไม่เป็นภาษาปกติดังแสดงในตัวอย่างต่อไปนี้

ตัวอย่าง 6.4: จงพิสูจน์ว่า $L_c = \{\omega \in \{0, 1\}^* | \text{ จำนวนของ 0 ใน } \omega = \text{ จำนวนของ 1 ใน } \omega\}$ ไม่เป็นภาษาปกติ

เราจะพิจารณา $L_h = \{0^i 1^i | \ i \geq 0\}$ จากตัวอย่าง 6.1 ซึ่งไม่เป็นภาษาปกติ แต่ $L_h = L_c \cap L(0^*1^*)$ และ $L(0^*1^*)$ เป็นภาษาปกติ ดังนั้น L_c ไม่เป็นภาษาปกติโดยสมบัติปิดของคลาสของภาษาปกติภายใต้การร่วม

จากตัวอย่าง 6.4 ถ้าพิจารณา L_c จะเห็นว่า การเลือกแบ่งสตริงในภาษานี้ทำได้หลายแบบมาก ซึ่งทำให้การพิสูจน์ยุ่งยาก แต่เมื่อเลือกภาษา $L(0^*1^*)$ มาทำการร่วมกับ L_c แล้ว ได้ภาษา L_h ซึ่งทำให้พิสูจน์ได้ง่ายขึ้น ในทำนองเดียวกัน ในตัวอย่าง 6.5 เราเลือกภาษา $L(0^*10^*)$ มาทำการร่วมกับภาษา L_r เช่นกัน

ตัวอย่าง 6.5: จงพิสูจน์ว่า $L_r = \{\omega \in \{0, 1\}^* | \ \omega = \omega^r\}$ ไม่เป็นภาษาปกติ

เราจะพิจารณา $L_m = \{0^i 1\ 0^i | \ i \geq 0\}$ จากตัวอย่าง 6.2 ซึ่งไม่เป็นภาษาปกติ แต่ $L_m = L_r \cap L(0^*10^*)$ และ $L(0^*10^*)$ เป็นภาษาปกติ ดังนั้น L_r ไม่เป็นภาษาปกติโดยสมบัติปิดของคลาสของภาษาปกติภายใต้การร่วม

ตัวอย่าง 6.6 แสดงการพิสูจน์โดยใช้สมบัติปิดภายใต้การเติมเต็ม

ตัวอย่าง 6.6: จงพิสูจน์ว่า $L_x = \{\omega \in \{0,1\}^* | \text{ จำนวนของ 0 ใน } \omega \neq \text{ จำนวนของ 1 ใน } \omega\}$ ไม่เป็นภาษาปกติ

เราจะพิจารณา $L_c = \{\omega \in \{0, 1\}^* | \text{ จำนวนของ } 0 \text{ ใน } \omega = \text{ จำนวนของ } 1 \text{ ใน } \omega\}$ จากตัวอย่าง 6.4 ซึ่งไม่เป็นภาษาปกติ แต่ $L_x = \overline{L_c}$ ดังนั้น L_x ไม่เป็นภาษาปกติโดยสมบัติปิดของคลาสของภาษาปกติภายใต้การเติมเต็ม

ออโตมาตาจำกัดไม่สามารถตรวจสอบภาษาไม่ปกติได้ แต่ก็ยังมีออโตมาตาอีกหลายแบบที่มี

ความสามารถมากกว่าออโตมาตาจำกัด ในหัวข้อ 6.2 เราจะอธิบายออโตมาตาแบบพูชดาวน์ ซึ่งสามารถตรวจสอบภาษาที่กล่าวถึงในตัวอย่าง 6.1 ถึง 6.6 ได้

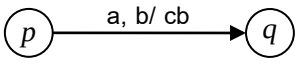
6.2 ออโตมาตาพูชดาวน์

ออโตมาตาพูชดาวน์เป็นตัวแบบทางคณิตศาสตร์ชนิดหนึ่งที่คล้ายกับออโตมาตาจำกัด แต่มีสแตกเพิ่มขึ้น หากเปรียบเทียบกับออโตมาตาจำกัดซึ่งสามารถจำสิ่งที่ต้องการได้ด้วยสถานะที่ต่างกันเท่านั้น ออโตมาตาพูชดาวน์สามารถใช้สแตกเพื่อเก็บสิ่งที่ต้องการจำได้ด้วย ดังนั้นจึงมีความสามารถมากกว่าออโตมาตาจำกัด ในระหว่างทำงานออโตมาตาพูชดาวน์สามารถใส่สัญลักษณ์ลงบนสแตก หรือดึงสัญลักษณ์ออกจากบนสแตกได้ ดังนั้นออโตมาตาพูชดาวน์ตรวจสอบภาษาที่ไม่เป็นภาษาปกติ เช่น ภาษาที่ประกอบด้วยสตริงที่มีจำนวน 0 และ 1 เท่ากัน ออโตมาตาพูชดาวน์จะตรวจสอบสตริงในลักษณะนี้ได้โดยเก็บสัญลักษณ์ 0 ลงในสแตกทุกครั้งที่อ่านเจอ สัญลักษณ์ 0 บนเทป และดึงสัญลักษณ์ 0 ออกจากสแตกทุกครั้งที่อ่านเจอสัญลักษณ์ 1 บนเทป ออโตมาตานี้จะยอมรับสตริงที่อยู่บนเทปเมื่ออ่านสัญลักษณ์บนเทปหมดและสแตกว่าง

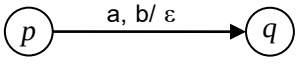
6.2.1 โครงสร้างและการทำงานของออโตมาตาพูชดาวน์

ออโตมาตาพูชดาวน์มีโครงสร้างต่างไปจากออโตมาตาจำกัด คือ ต้องมีสัญลักษณ์ที่ใช้กับสแตก และการเปลี่ยนสถานะก็ต้องเกี่ยวข้องกับสแตกด้วย สัญลักษณ์ที่ใช้ในสแตกอาจซ้ำกับสัญลักษณ์ที่อยู่บนเทปหรือไม่ก็ได้ นอกจากนั้นยังต้องมีสัญลักษณ์พิเศษที่แสดงว่าเป็นตัวล่างสุดในสแตก

การเปลี่ยนสถานะของออโตมาตาพูชดาวน์ขึ้นกับสถานะปัจจุบัน, สัญลักษณ์ที่อ่านได้จากเทป และ สัญลักษณ์ที่อยู่บนสุดของสแตก เมื่อเปลี่ยนสถานะแล้วออโตมาตาสามารถใส่สัญลักษณ์เพิ่มในสแตกหรือดึงสัญลักษณ์ตัวบนสุดของสแตกออกได้ด้วย รูป 6.1 แสดงสัญลักษณ์ที่ใช้ในแผนภาพเปลี่ยนสถานะสำหรับออโตมาตาพูชดาวน์



(a) เมื่ออ่านได้สัญลักษณ์ a บนเทปและ b บนสแตก แล้วใส่สัญลักษณ์ c ลงในสแตก



(b) เมื่ออ่านได้สัญลักษณ์ a บนเทปและ b บนสแตก แล้วดึงสัญลักษณ์ b ออกจากสแตก

รูป 6.1 การแสดงการเปลี่ยนค่าในสแตกที่ใช้ในแผนภาพเปลี่ยนสถานะสำหรับออโตมาตาพูชดาวน์

ออโตมาตาพูชดาวน์ทำงานเสร็จเมื่ออ่านสัญลักษณ์บนเทปหมด เมื่อทำงานเสร็จแล้ว ออโตมาตานั้นยอมรับสตริงที่อ่านไปหากจบการทำงานที่สถานะสิ้นสุดสถานะหนึ่งและสแตกว่าง แต่จะไม่ยอมรับสตริงนั้นหากจบการทำงานในสถานะที่ไม่ใช่สถานะสิ้นสุดหรือสแตกยังไม่ว่าง

โครงแบบของออโตมาตาพูชดาวน์ ประกอบด้วยสถานะ สตริงบนเทปที่ยังไม่ได้อ่าน และ สตริงของสัญลักษณ์ที่อยู่ในสแตก เช่น (q, 11010, 001Z) บอกว่าออโตมาตาอยู่ในสถานะ q และมีสตริง 11010 อยู่ ทางขวาของหัวเทปและสแตกมีสัญลักษณ์เรียงจากบนไปล่างเป็น 001 โดยใช้ Z แทนก้นของสแตก เนื่องจากออโตมาตาพูชดาวน์ไม่ขยับหัวเทปไปทางซ้ายเช่นเดียวกับออโตมาตาจำกัด สัญลักษณ์ที่อ่านไปแล้วจึงไม่มีผลกับการทำงานในขั้นต่อ ๆ ไปเช่นกัน

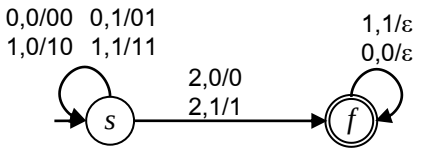
ออโตมาตาพูชดาวน์มักใช้ตรวจสอบสตริงที่มีการจับคู่สัญลักษณ์ซ้อนกันเป็นชั้นๆ เช่น วงเล็บที่ซ้อนกัน ดังนั้นจึงมีการนำออโตมาตาพูชดาวน์ประยุกต์ใช้ในการแปลภาษาคอมพิวเตอร์

6.2.2 ออโตมาตาพูชดาวน์เชิงกำหนดและออโตมาตาพูชดาวน์เชิงไม่กำหนด

ในทำนองเดียวกันกับออโตมาตาจำกัด ออโตมาตาพูชดาวน์มีการทำงานได้สองลักษณะ คือ การทำงานเชิงกำหนดและการทำงานเชิงไม่กำหนดตามที่กำหนดไว้ในฟังก์ชันเปลี่ยนสถานะ ฟังก์ชันเปลี่ยนสถานะของออโตมาตาทั้งสองแบบรับค่าสถานะปัจจุบัน สัญลักษณ์ที่อ่านได้บนเทป และ สัญลักษณ์ที่อยู่บนสุดของสแตก

สำหรับออโตมาตาพูชดาวน์เชิงกำหนด (deterministic pushdown automata) ฟังก์ชันนี้จะให้ค่าเป็นสถานะถัดไปและการเปลี่ยนค่าบนสแตก ซึ่งอาจเป็นการใส่ค่าเพิ่มบนสแตกหรือการดึงค่าออกมาจากสแตก สถานะและการเปลี่ยนค่าบนสแตกนี้จะมีได้เพียงแบบเดียว ตัวอย่าง เช่น เมื่อออโตมาตาอยู่ในสถานะ q อ่านได้สัญลักษณ์ 1 บนเทปและสัญลักษณ์ 0 บนสแตกและฟังก์ชันเปลี่ยนสถานะบอกว่าสถานะถัดไปคือสถานะ r และให้ใส่สัญลักษณ์ 1 เพิ่มบนสแตก จะไม่สามารถบอกว่าให้สถานะไปเป็นสถานะ t ได้ด้วย หรือให้ลบสัญลักษณ์ 0 บนสแตกออกด้วย ตัวอย่าง 6.7 ต่อไปนี้แสดงออโตมาตาพูชดาวน์เชิงกำหนดที่ยอมรับภาษา L_m = $\{\omega 2\omega^r \mid \omega \in \{0, 1\}^*\}$

ตัวอย่าง 6.7: ออโตมาตาพูชดาวน์เชิงกำหนดที่แสดงในรูป 6.2 ยอมรับภาษา L_m = $\{\omega 2\omega^r \mid \omega \in \{0, 1\}^*\}$



รูป 6.2 ออโตมาตาพูชดาวน์เชิงกำหนดที่ยอมรับ L_m

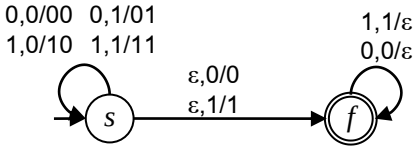
ที่สถานะ s ออโตมาตานี้ใส่สัญลักษณ์ 0 หรือ 1 ที่อ่านได้ลงในสแตก โดยยังไม่เปลี่ยนสถานะจนกว่าจะอ่านได้สัญลักษณ์ 2 จากเทปจึงเปลี่ยนไปยังสถานะ f นั่นคือสตริงส่วนหน้าที่อยู่ก่อนสัญลักษณ์ 2 จะถูกใส่ลงในสแตก

ที่สถานะ f ออโตมาตานี้จะดึงสัญลักษณ์บนสแตกออกถ้าเป็นสัญลักษณ์เดียวกับที่อ่านได้จากเทป ดังนั้นหากสแตกว่างเมื่ออ่านสัญลักษณ์สุดท้ายแล้ว หมายความว่า การตรวจสอบสตริงครึ่งแรกกับครึ่งหลังเสร็จสิ้นและส่วนหลังเป็นส่วนผกผันของส่วนแรก แต่หากไม่สามารถอ่านสตริงบนเทปจนหมด หมายความว่า พบบางตำแหน่งของสตริงครึ่งหลังที่ไม่ตรงกับส่วนผกผันของสตริงครึ่งแรก และเมื่อไม่สามารถอ่านเทปต่อได้จนจบก็ถือว่าออโตมาตาไม่ยอมรับสตริงนั้น

การเปลี่ยนสถานะทั้งหมดในออโตมาตานี้เป็นการเปลี่ยนสถานะเชิงกำหนด นั่นหมายความว่าเมื่อออโตมาตาอยู่ในสถานะใดๆ แล้วอ่านได้สัญลักษณ์บนเทปและบนสแตกชุดหนึ่ง จะเปลี่ยนสถานะและทำงานกับสแตกได้เพียงแบบเดียว

สำหรับออโตมาตาพูชดาวน์เชิงไม่กำหนด (nondeterministic pushdown automata) ฟังก์ชันเปลี่ยนสถานะจะให้ค่าสถานะถัดไปได้มากกว่าหนึ่งสถานะ และ ระบุการเปลี่ยนค่าบนสแตกได้มากกว่าหนึ่งแบบ ตัวอย่าง เช่น เมื่อออโตมาตาอยู่ในสถานะ q อ่านได้สัญลักษณ์ 1 บนเทปและสัญลักษณ์ 0 บนสแตก ฟังก์ชันเปลี่ยนสถานะอาจบอกว่าสถานะถัดไปคือสถานะ q หรือสถานะ r และให้ดึงสัญลักษณ์ 0 ออกจากสแตกหรือใส่สัญลักษณ์ 1 เพิ่มบนสแตก ก็ได้ นอกจากนั้นยังมีการเปลี่ยนสถานะด้วยสตริงว่าง ซึ่งทำให้ออโตมาตาอาจเลือกอยู่ที่สถานะเดิมหรือเลือกเปลี่ยนสถานะไปโดยใช้สตริงว่างก็ได้ ดังนั้นออโตมาตาเชิงไม่กำหนดนี้จะตัดสินใจเลือกเปลี่ยนสถานะในทางที่ทำให้ยอมรับสตริงที่อยู่บนเทปได้โดยอัตโนมัติ ตัวอย่าง 6.8 แสดงออโตมาตาพูชดาวน์เชิงไม่กำหนดที่ยอมรับภาษา L_p = $\{\omega\omega^r \mid \omega \in \{0, 1\}^*\}$

ตัวอย่าง 6.8: ออโตมาตาพูชดาวน์เชิงไม่กำหนดที่แสดงในรูป 6.3 ยอมรับภาษา L_p = $\{\omega\omega^r \mid \omega \in \{0, 1\}^*\}$



รูป 6.3 ออโตมาตาพูชดาวน์เชิงไม่กำหนดที่ ยอมรับภาษา L_p

ออโตมาตานี้คล้ายกับออโตมาตาในรูป 6.2 แต่ออโตมาตานี้มีการเปลี่ยนสถานะเชิงไม่กำหนด เมื่อออโตมาตาอยู่ที่สถานะ s มันอาจเปลี่ยนสถานะไปยังสถานะ f โดยไม่อ่านสัญลักษณ์ที่ได้บนเทปเลย การเปลี่ยนสถานะเช่นนี้เป็นการเปลี่ยนสถานะเชิงไม่กำหนดแบบหนึ่ง ออโตมาตานี้จะทำงานได้ถูกต้อง

เพราะมันใช้ความสามารถเชิงไม่กำหนดเพื่อเลือกเปลี่ยนสถานะจาก s ไปยัง f เมื่ออ่านครึ่งแรกของสตริงบนเทปจบพอดี

ถึงแม้ว่าออโตมาตาจำกัดเชิงกำหนดและออโตมาตาจำกัดเชิงไม่กำหนดจะมีความสามารถเท่ากัน แต่ออโตมาตาพูชดาวน์เชิงไม่กำหนดมีความสามารถมากกว่าออโตมาตาพูชดาวน์เชิงกำหนด นั่นคือมีภาษาที่ยอมรับได้ด้วยออโตมาตาพูชดาวน์เชิงไม่กำหนด แต่ไม่มีออโตมาตาพูชดาวน์เชิงกำหนดที่ยอมรับได้ ตัวอย่างเช่นภาษา L_p ยอมรับได้ด้วยออโตมาตาพูชดาวน์เชิงไม่กำหนดดังแสดงในตัวอย่าง 6.8 แต่ไม่มีออโตมาตาพูชดาวน์เชิงกำหนดที่ยอมรับภาษานี้ได้

6.2.3 ภาษาไม่พึ่งบริบท (Context-free Languages)

จากที่ผ่านมาเราอธิบายภาษาโดยใช้ออโตมาตาซึ่งตรวจสอบว่าสตริงใดอยู่ในภาษา เรายังสามารถอธิบายภาษาได้โดยใช้ไวยากรณ์ (grammar) ระบุกฎการสร้างสตริงในภาษา ไวยากรณ์ชนิดหนึ่งที่เกี่ยวข้องกับออโตมาตาพูชดาวน์ เรียกว่า ไวยากรณ์ไม่พึ่งบริบท (context-free grammar) [11]

ไวยากรณ์ไม่พึ่งบริบทประกอบด้วยชุดของกฎ กฎแต่ละข้อจะอยู่ในรูปแบบที่บอกว่าหน่วยหนึ่งของภาษานั้นประกอบด้วยหน่วยย่อยอย่างไรบ้าง กฎแต่ละข้อนี้กำหนดวิธีที่อนุญาตให้สร้างหน่วยนั้นในภาษาขึ้นมา เช่น กฎ $A \rightarrow xyz$ อธิบายว่าหน่วย A ในภาษาสร้างได้จากการนำหน่วยย่อย x, y และ z มาต่อกัน ในไวยากรณ์หนึ่ง ๆ อาจมีกฎมากกว่าหนึ่งกฎสำหรับสร้างหน่วยเดียวกัน นอกจากนั้นกฎของการสร้างแต่ละหน่วยต้องไม่ขึ้นกับบริบทที่หน่วยนั้นปรากฏอยู่ เช่น กฎ $A \rightarrow xyz$ บอกว่าหน่วย A สร้างได้จากหน่วยย่อย xyz ไม่ว่าจะปรากฏอยู่ที่ใด นั่นคือเราไม่สามารถระบุ ว่าหน่วย A สร้างจากหน่วยย่อย x, y, z เมื่อ A อยู่ระหว่าง b สองตัว ด้วยกฎ $b\,A\,b \rightarrow b\,xyz\,b$

คลาสของภาษาที่ยอมรับด้วยออโตมาตาพูชดาวน์เชิงไม่กำหนด เป็นคลาสเดียวกับภาษาไม่พึ่งบริบท เราสามารถพิสูจน์ได้โดย (1) แสดงว่าสามารถสร้างไวยากรณ์ไม่พึ่งบริบทสำหรับภาษาทุกภาษาที่ยอมรับด้วยออโตมาตาพูชดาวน์เชิงไม่กำหนด และ (2) แสดงว่าสามารถสร้างออโตมาตาพูชดาวน์เชิงไม่กำหนดที่ยอมรับภาษาไม่พึ่งบริบททุกภาษา [13] วิธีสร้างออโตมาตาจากไวยากรณ์ไม่พึ่งบริบทที่ใช้ในการพิสูจน์ส่วนที่ 2 นี้ใช้ในขั้นตอนวิธีแจงส่วน (parsing algorithm) ซึ่งเป็นขั้นตอนในการสร้างตัวแปลภาษา (compiler) ขั้นหนึ่ง

6.2.4 ภาษาที่ไม่เป็นภาษาไม่พึ่งบริบท

ในทำนองเดียวกันกับออโตมาตาจำกัด ความสามารถของออโตมาตาพูชดาวน์มีขีดจำกัดเช่นกัน หากลองพิจารณาภาษา $L_D = \{\omega\omega \mid \omega \in \{0, 1\}^*\}$ ที่ประกอบด้วยสตริงเดียวกันต่อกันสองครั้ง เช่น 011011 หากต้องการสร้างออโตมาตาพูชดาวน์เชิงไม่กำหนดที่ยอมรับภาษานี้ ออโตมาตานั้นต้องเก็บครึ่งแรกของสตริงลงในสแตก โดยใช้ความสามารถเชิงไม่กำหนดเปลี่ยนสถานะที่ตำแหน่งกึ่งกลางของ

สตริง หลังจากนั้นจะต้องเทียบครึ่งหลังของสตริงกับสตริงในสแตก แต่สตริงในสแตกเก็บย้อนทางกับสตริงที่เหลือบนเทป จึงไม่สามารถอ่านเปรียบเทียบกันทีละตัวได้ ดังนั้น เพื่อทำให้ลำดับการเรียงสัญลักษณ์ที่ยังไม่ได้อ่านจากเทปเหมือนกับสัญลักษณ์ที่อ่านไปแล้ว เราต้องใช้สแตกอีกหนึ่งตัวเพื่อเก็บสัญลักษณ์ที่เหลือ แล้วจึงเปรียบเทียบสตริงที่อยู่ในสแตกทั้งสอง แต่ออโตมาตาพูชดาวน์มีเพียงสแตกเดียว เราจึงไม่สามารถหาออโตมาตาพูชดาวน์เชิงไม่กำหนดที่ยอมรับภาษานี้ได้

วิธีที่เราใช้อธิบายมานี้เป็นวิธีแบบไม่เคร่งครัดที่สามารถทำให้เข้าใจได้ว่าภาษาใดไม่มีออโตมาตาพูชดาวน์ยอมรับ แต่เราสามารถใช้วิธี แบบเคร่งครัดพิสูจน์ได้เช่นกันวิธีนี้ใช้ทฤษฎีที่คล้ายกับบทตั้งปั๊มปิงสำหรับภาษาปกติที่ผ่านมา ทฤษฎีนี้เรียกว่าบทตั้งปั๊มปิงสำหรับภาษาที่ไม่ขึ้นกับบริบท [11] บทตั้งปั๊มปิงสำหรับภาษาไม่พึ่งบริบทคล้ายกับบทตั้งปั๊มปิงสำหรับภาษาปกติแต่จะต้องแบ่งสตริงเพื่อให้มีสตริงย่อยสองสตริงที่มีการซ้ำ และตรวจสอบว่าเมื่อมีการซ้ำแล้วสตริงที่ได้ยังอยู่ในภาษาเดิมหรือไม่

ภาษาต่อไปนี้ไม่เป็นภาษาไม่พึ่งบริบท

$L = \{0^n 1^n 0^n \mid n \geq 0\}$

$L = \{0^n 1^{n!} \mid n \geq 0\}$

$L = \{\omega \in \{0, 1, 2\}^* \mid$ จำนวนของ 0, 1และ 2 ใน ω เท่ากัน$\}$

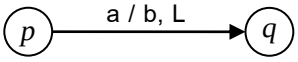
ในหัวข้อถัดไปเราจะอธิบายเครื่องจักรทูวริงซึ่งเป็นตัวแบบทางคณิตศาสตร์ที่สามารถใช้อธิบายการทำงานของคอมพิวเตอร์ที่ใช้กันอยู่ในปัจจุบัน

6.3 เครื่องจักรทูวริง

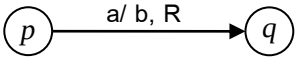
เครื่องจักรทูวริงเป็นตัวแบบทางคณิตศาสตร์ที่ใช้อธิบายการทำงานของคอมพิวเตอร์ที่ใช้กันในปัจจุบัน ตัวแบบนี้นิยามขึ้นโดยอลัน ทูวริง ในปี ค.ศ. 1936 ซึ่งในยุคนั้นยังไม่มีคอมพิวเตอร์สำหรับงานทั่วไป (General-purpose computers) ใช้ดังเช่นปัจจุบัน

6.3.1 โครงสร้างและการทำงานของเครื่องจักรทูวริง

เครื่องจักรทูวริงประกอบด้วยหน่วยควบคุมซึ่งมีสถานะและเทปที่เก็บข้อมูลเหมือนกับออโตมาตาจำกัด [12] แต่หัวเทปของเครื่องจักรทูวริงสามารถเขียนสัญลักษณ์ลงบนเทปและเลื่อนไปได้ทั้งทางซ้ายและทางขวา ดังนั้นเครื่องจักรทูวริงสามารถอ่านสตริงที่อยู่บนเทปกี่ครั้งก็ได้ นอกจากนั้นยังเขียนสิ่งที่ต้องการจำเก็บไว้บนเทปได้อีกด้วย การเปลี่ยนสถานะขึ้นอยู่กับสถานะปัจจุบันและสัญลักษณ์ที่อ่านได้บนเทปในขณะนั้นเช่นเดียวกับออโตมาตาจำกัด แต่เมื่อมีการเปลี่ยนสถานะแต่ละครั้งเครื่องจักรทูวริงสามารถตัดสินใจว่าจะเขียนสัญลักษณ์ใดลงบนเทปและเลื่อนหัวเทปไปทางซ้ายหรือขวา การเปลี่ยนสถานะของเครื่องจักรทูวริงเขียนเป็นแผนภาพเปลี่ยนสถานะได้โดยใช้สัญลักษณ์ดังแสดงในรูป 6.4



(a) เมื่ออ่านได้สัญลักษณ์ a บนเทปแล้วเขียน b ทับบนเทปและเลื่อนหัวเทปไปทางซ้าย



(b) เมื่ออ่านได้สัญลักษณ์ a บนเทปแล้วเขียน b ทับบนเทปและเลื่อนหัวเทปไปทางขวา

รูป 6.4 การแสดงการเลื่อนหัวเทปของเครื่องจักรทูวริงในแผนภาพเปลี่ยนสถานะ

โครงแบบของเครื่องจักรทูวริงประกอบด้วยสถานะและสตริงทั้งหมดบนเทปประกอบกับตำแหน่งของหัวเทปบนสตริงนั้น ตัวอย่างเช่น โครงแบบ (q, 100110) บอกว่าเครื่องจักรอยู่ที่สถานะ q และสตริงที่อยู่บนเทปเป็น 100110 โดยหัวเทปจะหยุดที่ช่องที่ 5 บนเทปซึ่งแสดงด้วยตำแหน่งที่ขีดเส้นใต้ในสตริง ดังนั้น ถ้าให้เครื่องจักรทูวริงมีฟังก์ชันเปลี่ยนสถานะ $\delta(q, 1) = (r, 0, L)$ ซึ่งหมายความว่าเมื่อเครื่องอยู่ที่สถานะ q และอ่านได้สัญลักษณ์ 1 แล้วจะเปลี่ยนไปอยู่ที่สถานะ r เขียนสัญลักษณ์ 0 และขยับหัวเทปไปทางซ้าย เมื่อเครื่องจากทูวริงมีโครงแบบเป็น (q, 100110) แล้ว จะเปลี่ยนเป็นโครงแบบถัดไปคือ (r, 100110)

เมื่อเปรียบเทียบการทำงานหนึ่งขั้นของเครื่องจักรทูวริงกับคำสั่งระดับพื้นฐานของคอมพิวเตอร์ที่ใช้กันในยุคแรก ๆ แล้ว การทำงานของเครื่องจักรทูวริงจะง่ายกว่ามาก การสร้างตัวแบบทางคณิตศาสตร์ที่ง่ายเช่นนี้ทำให้นำไปใช้พิสูจน์ความสามารถของเครื่องจักรทูวริงได้สะดวก

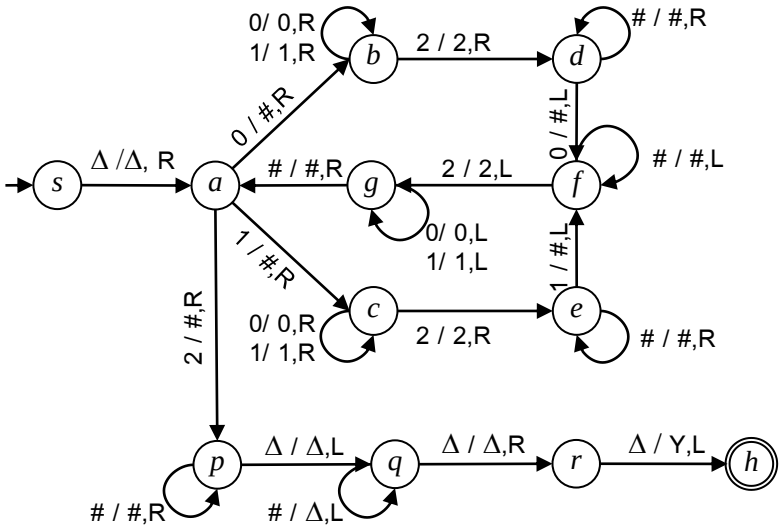
ข้อแตกต่างระหว่างเครื่องจักรทูวริงกับออโตมาตาจำกัดอีกประเด็นหนึ่งอยู่ที่สถานะสิ้นสุด เครื่องจักรทูวริงจบการทำงานเมื่อเข้าสู่สถานะที่เรียกว่าสถานะหยุด (halt state) ซึ่งต่างจากสถานะสิ้นสุดของออโตมาตาจำกัดคือเมื่อเครื่องเข้าสู่สถานะหยุดแล้วจะหยุดทำงานทันที นั่นคือไม่เปลี่ยนสถานะออกจากสถานะนี้ ในขณะที่ออโตมาตาจำกัดอาจเปลี่ยนสถานะออกจากสถานะสิ้นสุดได้ แต่จะจบการทำงานเมื่ออ่านสัญลักษณ์บนเทปจนหมด

6.3.2 เครื่องจักรทูวริงเชิงกำหนดและเครื่องจักรทูวริงเชิงไม่กำหนด

เครื่องจักรทูวริงเชิงกำหนด (deterministic Turing machine) เป็นเครื่องจักรที่มีการเปลี่ยนสถานะเชิงกำหนด นั่นคือเมื่อเครื่องอยู่ที่สถานะหนึ่งและอ่านได้สัญลักษณ์หนึ่งบนเทปแล้วจะมีทางเลือกให้เปลี่ยนไปยังสถานะถัดไป เขียนสัญลักษณ์บนเทป รวมถึงเลื่อนตรวจเทปไปในทางที่กำหนดได้แบบเดียว ตัวอย่าง 6.9 แสดงเครื่องจักรทูวริงเชิงกำหนด ที่ยอมรับภาษา $L_{D2} = \{w2w \mid w \in \{0, 1\}^*\}$ จะเห็นได้ว่าการเปลี่ยนสถานะที่แสดงในเครื่องจักรเป็นการเปลี่ยนสถานะเชิงกำหนดทั้งหมด

ตัวอย่าง 6.9: เครื่องจักรทูวริงเชิงกำหนดที่แสดงในรูป 6.5 ยอมรับภาษา $L_{D2} = \{\omega 2\omega \mid \omega \in \{0, 1\}^*\}$

เราสามารถอธิบายรูปแบบของสตริงที่จะยอมรับเป็น $w_1 2 w_2$ โดยที่ w_1 และ w_2 เป็นสตริงที่เหมือนกัน เครื่องจักรทูวริงนี้ทำงานโดยตรวจสอบสตริงส่วนแรก w_1 กับสตริงส่วนหลัง w_2



รูป 6.5 เครื่องจักรทูวริงเชิงกำหนดที่ยอมรับภาษา $L_{D2} = \{\omega 2 \omega \mid \omega \in \{0, 1\}^*\}$

ที่สถานะ s เครื่องจะเลื่อนหัวเทปไปทางขวาหนึ่งช่องเพื่ออ่านสัญลักษณ์แรกของ w_1

ที่สถานะ a เครื่องจะอ่านสัญลักษณ์แรกในส่วน w_1 แล้วจำว่าได้ 0 หรือ 1 ถ้าอยู่ที่สถานะ b คือ จำว่าอ่านได้ 0 ถ้าอยู่ที่สถานะ c คือ จำว่าอ่านได้ 1 ในการอ่านนี้จะลบสัญลักษณ์ที่อ่านแล้วโดยการเขียนสัญลักษณ์ # ทับ

จากนั้น สถานะ b และ c เป็นการอ่านข้ามส่วนของ w_1 แล้วจะเปลี่ยนไปอยู่ที่สถานะ d หรือ e เมื่ออ่านได้สัญลักษณ์ 2 และจะอ่านข้ามส่วนที่ลบไปแล้วของ w_2 ดังนั้นหากพบสัญลักษณ์แรกของ w_2 ตรงกับสัญลักษณ์แรกของ w_1 ที่อ่านมาแล้ว จะเปลี่ยนไปอยู่ที่สถานะ f

ที่สถานะ f เครื่องเลื่อนหัวเทปข้ามส่วนของ w_2 ที่ลบไปแล้วจนเจอสัญลักษณ์ 2 แล้วเปลี่ยนไปอยู่ที่สถานะ g

ที่สถานะ g เครื่องเลื่อนหัวเทปไปเพื่อหาสัญลักษณ์แรกของ w_1 ที่ยังไม่ได้ลบ

หากตรวจสอบทุกสัญลักษณ์ของ w_1 แล้ว ที่สถานะ a จะอ่านได้สัญลักษณ์ 2 ดังนั้นเมื่ออ่านได้สัญลักษณ์ 2 ที่สถานะ a เครื่องจะเปลี่ยนสถานะไปยังสถานะ p ซึ่งจะตรวจสอบว่าอ่านสัญลักษณ์ในส่วน w_2 แล้วทั้งหมดหรือไม่ นั่นคือ ที่สถานะ q ในกรณีที่ยังตรวจสอบส่วน w_2 ไม่ครบแล้ว (w_2 ยาวกว่า w_1) จะอ่านได้สัญลักษณ์ 0 หรือ 1 ที่สถานะนี้และไม่สามาถทำงานต่อไปได้ ดังนั้นจึงไม่ยอมรับสตริงนี้ ในกรณีที่ตรวจสอบส่วน w_2 ครบแล้ว ทางขวามือของเทปเหลือแต่สัญลักษณ์ # จนไปถึง Δ และเปลี่ยนสถานะไปยังสถานะ r แล้วเขียนคำตอบด้วยสัญลักษณ์ Y แล้วเข้าสู่สถานะหยุด

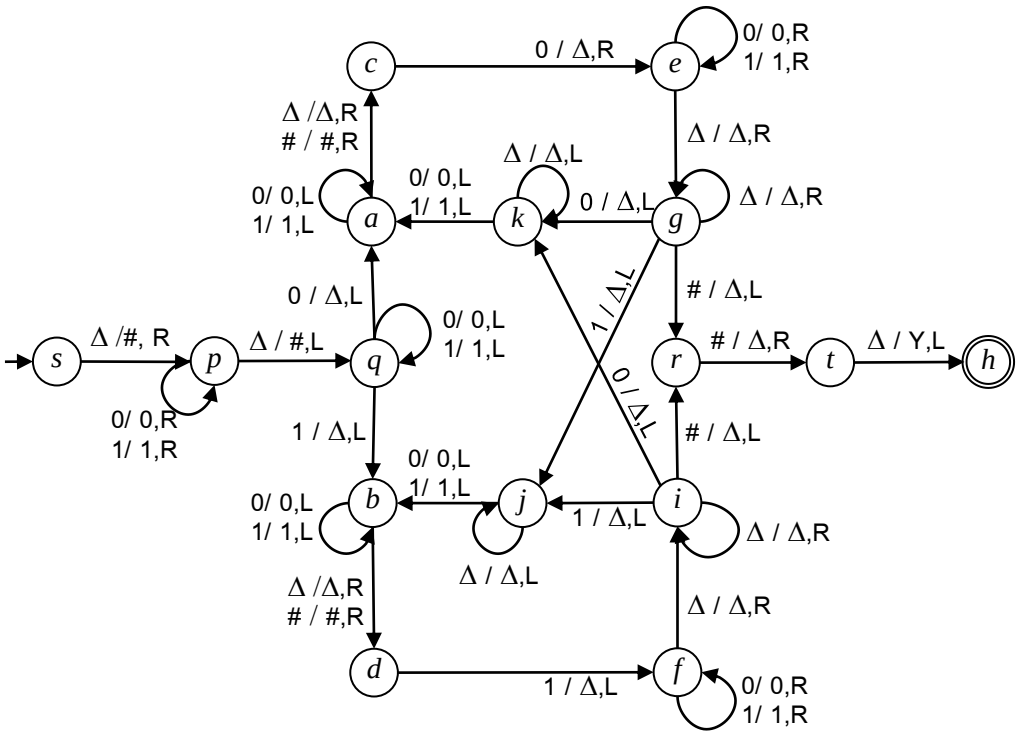
เครื่องจักรทูวริงเครื่องนี้อาจหยุดทำงานเนื่องจากไม่มีการเปลี่ยนสถานะที่เป็นไปได้สำหรับสัญลักษณ์ที่อ่านได้ในบางสถานะ เช่น ที่สถานะ d เมื่ออ่านได้สัญลักษณ์ 1 จะไม่มีการเปลี่ยนสถานะ

ออกไปจากสถานะ d ได้ ดังนั้นเมื่อสตริงที่อยู่บนเทปไม่อยู่ในภาษาที่กำหนดเครื่องอาจหยุดทำงานโดยไม่ไปถึงสถานะหยุด แต่ถ้าสตริงที่อยู่บนเทปอยู่ในภาษาที่กำหนดเครื่องต้องไปถึงสถานะหยุดและเขียนคำตอบให้เสมอ

เครื่องจักรทูวริงเชิงไม่กำหนด (nondeterministic Turing machine) เป็นเครื่องจักรที่มีการเปลี่ยนสถานะเชิงไม่กำหนดได้ นั่นคือมีการเปลี่ยนสถานะโดยไม่ขึ้นกับสัญลักษณ์ที่อ่านได้บนเทป (คือการเปลี่ยนสถานะด้วยสตริงว่าง) และ เมื่อ เครื่องอยู่ที่สถานะหนึ่งและอ่านได้สัญลักษณ์หนึ่งแล้วเครื่องมีทางเลือกเปลี่ยนสถานะมากกว่าหนึ่งทาง ดังนั้นฟังก์ชันเปลี่ยนสถานะของเครื่องจักรทูวริงเชิงไม่กำหนดเป็นฟังก์ชันจากสถานะและสัญลักษณ์ที่อ่านได้บนเทปซึ่งอาจเป็นสตริงว่าง ไปยังเซตของสถานะ, สัญลักษณ์ที่เขียนบนเทป และ ทิศทางที่เลื่อนหัวเทป ตัวอย่าง 6.10 แสดงเครื่องจักรทูวริงเชิงไม่กำหนดที่ยอมรับภาษา $L_D = \{\omega\omega \mid \omega \in \{0, 1\}^+\}$

ตัวอย่าง 6.10: เครื่องจักรทูวริงเชิงไม่กำหนดที่แสดงในรูป 6.6 ยอมรับภาษา $L_D = \{\omega\omega \mid \omega \in \{0, 1\}^+\}$

เราสามารถอธิบายรูปแบบของสตริงที่จะยอมรับเป็น w_1w_2 โดยที่ w_1 และ w_2 เป็นสตริงที่เหมือนกัน เครื่องจักรทูวริงนี้ทำงานโดยตรวจสอบสตริงส่วนแรก w_1 กับสตริงส่วนหลัง w_2



รูป 6.6 เครื่องจักรทูวริงเชิงไม่กำหนดที่ยอมรับภาษา L_D

ที่สถานะ s และ p เครื่องจะเขียนสัญลักษณ์ # ปิดหัวท้ายของสตริงบนเทป

ที่สถานะ q เครื่องใช้ความสามารถเชิงไม่กำหนดเพื่อตัดสินใจว่าเมื่อขยับหัวเทปไปทางขวาแล้ว

จะหยุดที่สัญลักษณ์แรกของ w_2 เมื่อไร นอกจากนั้นเครื่องจะเปลี่ยนสถานะไปยังสถานะ a หรือ b เพื่อจำว่าสัญลักษณ์ที่อ่านมาของส่วน w_2 เป็น 0 หรือ 1 ตามลำดับ

ที่สถานะ a เครื่องจะเลื่อนหัวเทปไปสัญลักษณ์แรกใน w_1

ที่สถานะ c เครื่องจะตรวจสอบว่าสัญลักษณ์แรกของอยู่ w_1 เป็น 0 เช่นเดียวกับสัญลักษณ์แรกในส่วน w_2 ที่อ่านไปแล้ว จากนั้นที่สถานะ e เครื่องจะอ่านข้ามส่วน w_1

ที่สถานะ g เครื่องจะเลื่อนหัวเทปข้ามส่วนที่ลบไปแล้วของ w_2 เพื่อหาสัญลักษณ์แรกในส่วน w_2

หากอ่านได้สัญลักษณ์ 0 ที่สถานะนี้เครื่องจะเปลี่ยนสถานะไปยังสถานะ k ซึ่งจะเลื่อนหัวเทปข้ามส่วนของ w_2 ที่ลบไปแล้วเพื่อหาส่วนท้ายของ w_1 เมื่อเจอส่วนท้ายของ w_1 จะเปลี่ยนสถานะไปยังสถานะ a

หากอ่านได้สัญลักษณ์ 1 ที่สถานะนี้ เครื่องจะเปลี่ยนสถานะไปยังสถานะ j ซึ่งจะทำหน้าที่คล้ายสถานะ k ที่กล่าวมาแล้ว

สังเกตว่าครึ่งบนของแผนภาพเปลี่ยนสถานะนี้จะทำงานคล้ายกับครึ่งล่าง แต่ครึ่งบนเป็นการตรวจสอบเมื่อเจอสัญลักษณ์ 0 แต่ครึ่งล่างเป็นการตรวจสอบเมื่อเจอสัญลักษณ์ 1

ที่สถานะ g และ i หากเลื่อนหัวเทปไปแล้วอ่านได้สัญลักษณ์ # หมายความว่าตรวจสอบสตริงทั้งหมดถูกต้องแล้วจึงเปลี่ยนสถานะไปยังสถานะ r ซึ่งเลื่อนหัวเทปไปซ้ายสุดเพื่อเขียนคำตอบแล้วไปยังสถานะหยุด

การเพิ่มความสามารถเชิงไม่กำหนดให้เครื่องจักรทูวริงทำให้สร้างเครื่องจักรทูวริงสำหรับทำงานที่กำหนดได้ง่ายขึ้นในทำนองเดียวกับออโตมาตาจำกัด อย่างไรก็ตามการเพิ่มความสามารถเชิงไม่กำหนดนี้ไม่ทำให้ความสามารถของเครื่องจักรทูวริงเพิ่มขึ้นดังแสดงได้จากการสร้างเครื่องจักรทูวริงเชิงกำหนดเลียนแบบเครื่องจักรทูวริงเชิงไม่กำหนด ตัวอย่าง เช่น เครื่องจักรทูวริงในรูป 6.6 ที่ใช้ความสามารถเชิงไม่กำหนดช่วยเดาจุดเริ่มต้นของครึ่งหลังของสตริงบนเทป เราอาจหาจุดแบ่งนี้ได้โดยใช้ความสามารถเชิงกำหนดเช่นกัน ผู้อ่านอาจลองสร้างเครื่องดังกล่าวเองเพื่อฝึกหัด

การสร้างเครื่องจักรทูวริงเชิงกำหนด T_s ที่เลียนแบบการทำงานของเครื่องจักรทูวริงไม่กำหนด T_x ทำได้โดยให้เครื่องจักรทูวริงเชิงกำหนดเขียนโครงแบบที่เป็นไปได้ของ T_x ต่อไปเรื่อย ๆ นั่นคือต้องสร้างเครื่องจักรทูวริงเชิงกำหนด T_n ที่อ่านโครงแบบที่บันทึกไว้บนเทปแล้วเขียนโครงแบบถัดไปซึ่งระบุไว้ด้วยฟังก์ชันเปลี่ยนสถานะของ T_x เมื่อมีการเปลี่ยนสถานะเชิงไม่กำหนดทำให้สามารถมีโครงแบบถัดไปได้มากกว่าหนึ่งโครงแบบ T_n ต้องเขียนโครงแบบที่เป็นไปได้ให้ครบ ดังนั้นเครื่องจักรทูวริง T_s สร้างมาจาก T_n โดยให้ T_n เขียนโครงแบบถัดไปจนกระทั่งได้โครงแบบที่อยู่ในสถานะหยุด เมื่อถึงโครงแบบนี้แล้วจะได้คำตอบที่อยู่บนเทปของ T_x ตามที่ระบุในโครงแบบ หาก T_s เขียนโครงแบบใหม่ต่อไปเรื่อย ๆ โดยไม่หยุดทำงาน หมายความว่า T_x เปลี่ยนโครงแบบไปได้เรื่อยๆ หรือไม่หยุดทำงานเช่นกัน

6.3.3 การประยุกต์ใช้เครื่องจักรทูวริง

อลัน ทูวริง นิยามเครื่องจักรทูวริงขึ้นมาเพื่อแสดงข้อจำกัดของคอมพิวเตอร์ ข้อจำกัดนี้แสดงโดยการพิสูจน์ว่าไม่มีเครื่องจักรทูวริงที่บอกได้ว่าเครื่องจักรทูวริงที่กำหนดให้อีกเครื่องจะหยุดทำงานเมื่อได้รับสตริงที่กำหนดบนเทปหรือไม่โดยใช้ข้อขัดแย้ง เราสามารถบอกได้ว่าเครื่องจักรทูวริงเครื่องหนึ่งจะหยุดทำงานได้โดยจำลองการทำงานของมันไปจนหยุดทำงาน แต่หากเราต้องการบอกว่าเครื่องจักรทูวริงเครื่องหนึ่งไม่หยุดทำงาน เราไม่สามารถบอกไว้ว่าต้องรอให้มันทำงานไปนานเท่าไร ดังนั้นจึงไม่สามารถสร้างเครื่องจักรทูวริงให้ตรวจสอบได้

นอกจากนั้น เรายังใช้เครื่องจักรทูวริงเพื่อวัดความซับซ้อนของปัญหาซึ่งเป็นหัวข้อที่เป็นพื้นฐานสำหรับการศึกษาการวิเคราะห์ขั้นตอนวิธี (Analysis of algorithms) อีกด้วย

บรรณานุกรม

[1] A.V. Aho, R. Sethi and J.D. Ullman, *Compilers: Principles, Techniques, and Tools.* Addison-Wesley, 1986.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to Algorithms. The MIT Press, 3rd ed., 2009.

[3] M. Davis, R. Sigal and E. J. Weyuker, *Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science.* Morgan Kaufmann, 2nd ed., 1994.

[4] J. V. Guttag, *Introduction to Computation and Programming Using Python.* The MIT Press, 2013

[5] M. T. Goodrich and R. Tomassia, *Data Structures and Algorithms in Java.* John Wiley & Sons, 4th ed., 2006.

[6] J. E. Hopcroft, R. Motwani and J. D Ullman, *Introduction to Automata Theory, Languages, and Computation*. Pearson Education, 2nd ed., 2008.

[7] R. H. Katz and G. Borriello, *Contemporary Logic Design.* Prentice Hall , 2nd ed., 2004.

[8] H. R. Lewis and C. H. Papadimitriou, *Elements of the Theory of Computation*. Prentice-Hall, 2nd ed., 1998.

[9] K. C. Louden and K. A. Lambert, *Programming Languages: Principles and Practice.* Cengage, 3rd ed., 2012.

[10] M. M. Mano, C. R. Kime, and T. Martin, *Logic and Computer Design: Fundamentals.* Pearson Education, 5th ed., 2016.

[11] J. Martin, *Introduction to Languages and the Theory of Computation.* McGraw-Hill Education, 4th ed., 2010.

[12] K. Rosen, Discrete Mathematics and Its Applications. McGraw-Hill, 7th ed., 2011.

[13] M. Sipser, *Introduction to the Theory of Computation.* Cengage Learning, 3rd ed., 2013.

[14] A. Watt, *Beginning Regular Expressions: Programmer to Programmer.* Wiley, 2005.

[15] T. Weilkiens, *Systems Engineering with SysML/UML: Modeling, Analysis, Design.* Morgan Kaufmann, 1st ed., 2008.

ดัชนี