การจัดรูปนัยและตรวจสอบความต้องกันของแผนภาพสถานะยูเอ็มแอล โดยใช้ซีอาร์อีและไพแคลคูลัส

นายศิริชัย จันทร์สมัคร

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2549

ISBN 974-14-2496-5

FORMALIZATION AND CONSISTENCY CHECKING FOR UML STATECHART DIAGRAM USING CRE AND π-CALCULUS

Mr. Sirichai Jansamak

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering Program in Computer Engineering

Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University

Academic Year 2006
ISBN 974-14-2496-5

หัวข้อวิทยานิพนธ์	การจัดรูปนัยและตรวจสอบความต้องกันของแผนภาพสถานะยูเอ็มแอล โดยใช้ซีอาร์อี และไพแคลคูลัส
โดย	นายศิริชัย จันทร์สมัคร
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษา	อาจารย์ ดร.อรรถสิทธิ์ สุรฤกษ์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์

(ศาสตราจารย์ ดร. ดิเรก ลาวัณย์ศิริ)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ

(รองศาสตราจารย์ ดร.วันชัย ริ้วไพบูลย์)

..... อาจารย์ที่ปรึกษา

(อาจารย์ ดร.อรรถสิทธิ์ สุรฤกษ์)

..... กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.อานนท์ รุ่งสว่าง)

..... กรรมการ

(อาจารย์ ดร.วิษณุ โคตรจรัส)

ศิริชัย จันทร์สมัคร: การจัดรูปนัยและตรวจสอบความต้องกันของแผนภาพสถานะยูเอ็มแอล โดยใช้ซีอาร์อี และไพแคลคูลัส. (FORMALIZATION AND CONSISTENCY CHECKING FOR UML STATECHART DIAGRAM USING CRE AND π-CALCULUS) อ. ที่ปรึกษา: อ. ดร.อรรถสิทธิ์ สุรฤกษ์, 73 หน้า. ISBN 974-14-2496-5.

ปัจจุบันแผนภาพสถานะยูเอ็มแอลถูกนำมาเป็นเครื่องมือในการพัฒนาซอฟต์แวร์กันอย่างแพร่หลายเพื่อช่วยในการอธิบายพฤติกรรมการทำงานของระบบ ซึ่งภายในระบบหนึ่งๆ นั้นจะประกอบไปด้วยวัตถุต่างๆ ที่ทำงานร่วมกัน โดยที่วัตถุเหล่านี้จะมีการติดต่อสื่อสารทั้งกับสิ่งแวดล้อมภายนอกระบบและติดต่อกับวัตถุต่างๆ ภายในระบบด้วยกันเอง ดังนั้นผู้ที่ออกแบบระบบจึงจำเป็นต้องบรรยายพฤติกรรมของแต่ละวัตถุให้มีความชัดเจนและสอดคล้องกัน แต่ในขณะเดียวกันเมื่อระบบที่ต้องการมีขนาดขยายใหญ่ยิ่งขึ้น ความยากและซับซ้อนของการออกแบบวัตถุก็จะขยายเพิ่มขึ้นเช่นกัน ดังนั้นการตรวจสอบความต้องกันของแผนภาพของแต่ละวัตถุจึงเข้ามามีส่วนสำคัญเพื่อช่วยให้ขั้นตอนของการออกแบบระบบมีประสิทธิภาพยิ่งขึ้น โดยเฉพาะอย่างยิ่งในระบบที่มีความเสี่ยงหรือต้องการความถูกต้องสูงการตรวจสอบความถูกต้องของระบบจึงเป็นสิ่งที่ละไม่ได้และทวีความสำคัญมากยิ่งขึ้น

ในการตรวจสอบความต้องกันของแผนภาพสถานะยูเอ็มแอลนั้น งานวิจัยชิ้นนี้ได้นำเสนอกฎและวิธีการแปลงแผนภาพไปเป็น ซีอาร์อี และไพแคลคูลัส ซึ่งเป็นภาษารูปนัยที่รองรับการบรรยายพฤติกรรมของระบบที่ทำงานพร้อมกันซึ่งมีความซับซ้อนในการทำงานสูง เพื่อเพิ่มความสามารถในการประยุกต์ใช้วิธีการทางรูปนัยต่างๆ นอกจากนั้นยังได้นำเสนอกฎการตรวจสอบความต้องกันระหว่างแผนภาพย่อยของแต่ละวัตถุซึ่งทำงานร่วมกันภายในระบบ คือ ตรวจสอบความเท่าเทียมกันของวัตถุ และพฤติกรรมของวัตถุที่ไม่สอดคล้องกันเมื่อทำงานร่วมกันในระบบ โดยผลของการตรวจสอบที่ได้จะแสดงให้เห็นถึงความผิดพลาดที่สามารถเกิดขึ้นได้ของระบบจากขั้นตอนของการออกแบบที่ไม่ถูกต้อง ซึ่งจะช่วยให้ผู้ออกแบบสามารถนำมาเป็นจุดอ้างอิงเพื่อการปรับปรุงหรือแก้ไขระบบนั้นๆ เพื่อให้ได้ระบบที่ประสิทธิภาพต่อไปในอนาคต

ภาควิชาวิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต.....
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ลายมือชื่ออาจารย์ที่ปรึกษา.....
ปีการศึกษา 2549

4570710021 : MAJOR COMPUTER ENGINEERING

KEY WORD: FORMALIZATION / UML STATECHART / CONSISTENCY CHECKING

SIRICHAI JANSAMAK: FORMALIZATION AND CONSISTENCY CHECKING FOR UML STATECHART DIAGRAM USING CRE AND π-CALCULUS. THESIS ADVISOR: ATHASIT SURARERKS, Ph.D., 73 pp. ISBN 974-14-2496-5.

UML Statechart diagram is a tool that is widely used in software development project to describe behaviors of system. A system consists of objects that work in the environment. Because these objects will communicate with both environment and other objects in the same system, system designers have to describe their behaviors precisely and consistently. When the size of the system increases, the complexity and effort to develop will go up as well, therefore consistency verification of each object becomes one of important tools to help the system design phase becoming efficient, especially for the real-time or control system of which correctness cannot be overlooked.

Regarding the consistency checking of UML Statechart diagram, this thesis proposes rules and methodology for transforming this diagram to CRE and π-Calculus, formal languages which support describing behaviors of complex concurrent system. This thesis also introduces some rules of consistency checking between UML Statechart diagram of each object working in the same system. Equality of objects and inconsistency behaviors of objects in the same system can be identified. The results of consistency checking will allow the system designer to see the problems that will occur by poor quality design phase. These will be the reference points that help designers to adjust or improve their design for high quality systems.

Department Computer Engineering Student's signature.....
Field of study Computer Engineering Advisor's signature.....
Academic year 2006

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้ด้วยความอนุเคราะห์ และความช่วยเหลืออย่างยิ่งจาก อ.ดร.อรรถสิทธิ์ สุรฤกษ์ อาจารย์ที่ปรึกษา ซึ่งเป็นผู้ให้ข้อคิด แนวทาง และคำปรึกษาตลอดจนเป็นผู้ตรวจทานแก้ไข จนทำให้วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วง จึงขอขอบพระคุณอาจารย์เป็นอย่างสูงที่ให้ความเมตตา ช่วยเหลือ รวมทั้งโอกาส และสิ่งที่ดีแก่ผู้วิจัยเสมอมา

ขอขอบพระคุณ รศ.ดร.วันชัย ริ้วไพบูลย์ อ.ดร.วิษณุ โคตรจรัส และผศ.ดร.อานนท์ รุ่งสว่าง คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ ที่ได้กรุณาให้คำแนะนำในการเขียนวิทยานิพนธ์ที่ดี ซึ่งวิทยานิพนธ์ฉบับนี้ไม่อาจสำเร็จได้หากไม่ได้รับความร่วมมือจากทุกท่าน ขอขอบคุณอาจารย์ในภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัยทุกท่าน และเพื่อนๆ ทุกคน ผู้ที่ให้คำแนะนำเพิ่มเติมกับผู้วิจัยตลอดมา ขอบคุณบริษัทอวานาด พี่ๆ และเพื่อนร่วมงานทุกท่านที่ช่วยเหลือและให้โอกาสผู้เขียนได้ทุ่มเทเวลากับวิทยานิพนธ์ฉบับนี้ และขอขอบคุณคนหนึ่งคนนั้นซึ่งเป็นกำลังใจให้เสมอ

สุดท้ายนี้ขอขอบพระคุณ บิดา มารดา และสมาชิกในครอบครัวทุกท่าน ที่เป็นกำลังใจสำคัญ แรงผลักดัน และช่วยเหลือในทุกๆ ด้าน จนผู้วิจัยสามารถทำวิทยานิพนธ์ได้สำเร็จลุล่วง

สารบัญ

สารบัญภาพ

ภาพที่ หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันระบบซอฟต์แวร์ได้มีการขยายตัวทั้งในเชิงขนาดและความซับซ้อนมากยิ่งขึ้น ซึ่งในระบบเหล่านั้นต่างก็ต้องการซอฟต์แวร์ที่มีความเชื่อถือได้โดยเฉพาะในระบบที่มีความเสี่ยงสูง แต่การสร้างซอฟต์แวร์ให้มีความเชื่อถือได้นั้นเป็นปัญหาที่ยุ่งยากของอุตสาหกรรมการผลิตซอฟต์แวร์ และจะเป็นการล่าช้าเกินไปหากจะรอให้พบข้อผิดพลาดของการใช้งานโดยผู้ใช้หลังจากซอฟต์แวร์ได้ออกจำหน่ายไปแล้ว การตรวจสอบความถูกต้อง (verification) เป็นวินัยอย่างหนึ่งของกรรมวิธีการพัฒนาซอฟต์แวร์ที่สำคัญที่สามารถช่วยทำให้การพัฒนาซอฟต์แวร์มีคุณภาพมากยิ่งขึ้น

ในการอธิบายความต้องการของระบบนั้น ภาษาธรรมชาติซึ่งเป็นภาษาที่เราใช้ในการสื่อสารกันในชีวิตประจำวันมักจะถูกนำมาใช้เป็นภาษาหลัก ดังนั้นความเข้าใจถึงความต้องการของระบบจึงขึ้นอยู่กับความหมายของคำและรูปประโยคที่นำมาใช้ในการอธิบาย จากความซับซ้อนและกำกวมของภาษาธรรมชาติในความต้องการของระบบดังกล่าวทำให้เป็นการยากที่เราจะทำความเข้าใจและตรวจสอบความถูกต้อง จนอาจก่อให้เกิดความคลาดเคลื่อนในการสื่อสารระหว่างผู้ร่วมงาน โดยเฉพาะอย่างยิ่งหากระบบมีการขยายตัวและมีความซับซ้อนมากยิ่งขึ้นจะทำให้มีผลต่อประสิทธิภาพของการพัฒนา และด้วยความเป็นอรูปนัย (informal) ของความต้องการระบบนี้เอง ขั้นตอนที่สามารถดำเนินการได้โดยอัตโนมัติก็เป็นไปได้ยากด้วยเช่นกัน ดังนั้นความต้องการทางด้านวิธีรูปนัย (formal methods) จึงเพิ่มมากยิ่งขึ้นโดยเฉพาะในส่วนของการพัฒนาระบบการประมวลผลแบบกระจาย (distributed system) ระบบที่มีการทำงานแบบพร้อมกัน (concurrent system) และระบบการทำงานแบบเรียลไทม์ (real-time system) ซึ่งการนำเอาวิธีรูปนัยเข้ามาใช้จะช่วยเพิ่มความชัดเจนของการอธิบาย ทำให้ความสามารถในการเข้าถึงและเข้าใจความต้องการของระบบมีมากขึ้น อีกทั้งทำให้เราสามารถสร้างและเพิ่มประสิทธิภาพของขั้นตอนการตรวจสอบแบบอัตโนมัติได้ด้วย ดังนั้นการใช้งานวิธีรูปนัยจึงเป็นส่วนสำคัญส่วนหนึ่งที่คาดหมายว่าจะนำไปสู่การพัฒนาซอฟต์แวร์ที่มีประสิทธิภาพและมีความเชื่อถือได้

แผนภาพยูเอ็มแอล (UML: Unified Modeling Language) [1, 11] ซึ่งเป็นมาตรฐานที่ดูแลและควบคุมโดยโอเอ็มจี (OMG: Object Management Group) เป็นภาษาที่นำมาใช้ในการวิเคราะห์และออกแบบพัฒนาซอฟต์แวร์ในระบบเชิงวัตถุ โดยในปัจจุบันนี้ได้มีการนำไปใช้งานกันอย่างแพร่หลายในอุตสาหกรรมการพัฒนาซอฟต์แวร์ แผนภาพยูเอ็มแอลนี้เองประกอบด้วย

แผนภาพทั้งหมด 9 แผนภาพสำหรับอธิบายโครงสร้างและพฤติกรรมของระบบในมุมมองต่างๆ กัน โดยจะช่วยให้ผู้อ่านในแต่ละกลุ่มสามารถเข้าใจถึงตัวระบบได้ง่ายยิ่งขึ้น

แผนภาพสถานะ (state diagram) [4] เป็นหนึ่งในแผนภาพยูเอ็มแอลที่มีการใช้งานมากในการอธิบายการเปลี่ยนแปลงพฤติกรรมของระบบ โดยแต่ละแผนภาพจะอธิบายการเปลี่ยนแปลงพฤติกรรมของวัตถุเมื่อมีเหตุการณ์ต่างๆ เกิดขึ้น แต่เนื่องจากความเป็นกึ่งรูปนัย (semi-formal) ของตัวแผนภาพสถานะนั้น ทำให้การตรวจสอบและวิเคราะห์ด้วยกระบวนการทางรูปนัยนั้นเป็นไปได้ยาก ดังนั้นในงานวิจัยนี้จึงมีแนวความคิดที่จะเสนอวิธีการแปลงแผนภาพสถานะไปเป็นภาษารูปนัย (formal language) เพื่อรองรับและเพิ่มความสามารถในการตรวจสอบและวิเคราะห์แผนภาพ งานวิจัยชิ้นนี้ให้ความสนใจในคุณสมบัติการเปลี่ยนแปลงพฤติกรรมของระบบ (dynamic behavior) และการทำงานแบบพร้อมกัน (concurrent) ดังนั้นภาษารูปนัยที่นำมาใช้เป็นภาษาเป้าหมายจึงต้องมีความสามารถรองรับการอธิบายคุณสมบัติดังกล่าวเป็นอย่างดี ภาษาที่ได้เลือกมาใช้เป็นภาษาที่อธิบายลำดับการเกิดขึ้นของกระบวนการต่างๆ ซึ่งในที่นี้ คือ ซีอาร์อี (CRE: Concurrent Regular Expression) [3] และไพแคลคูลัส (π-Calculus) [7]

1.2 วัตถุประสงค์ของการวิจัย

1.2.1 เพื่อกำหนดขั้นตอนและกฎในการแปลงแผนภาพสถานะไปเป็น ซีอาร์อี และไพแคลคูลัส

1.2.2 เพื่อสร้างกฎในการตรวจสอบความสัมพันธ์ระหว่างแผนภาพสถานะ

1.3 ขอบเขตของการวิจัย

1.3.1 เพื่อกำหนดขั้นตอนและกฎในการแปลงแผนภาพสถานะไปเป็น ซีอาร์อี และไพแคลคูลัส โดยจะต้องได้ขั้นตอนและกฎที่มีความสามารถดังนี้

1.3.1.1 สามารถแปลงองค์ประกอบในการสร้างแผนภาพสถานะดังต่อไปนี้

1) สถานะทั่วไป (simple state) ซึ่งครอบคลุมถึงเหตุการณ์การเข้าสู่สถานะ (entry) การออกจากสถานะ (exit) กิจกรรมที่ต้องทำ (do activity) และการเปลี่ยนแปลงภายใน (internal transition)
2) สถานะประกอบ (composite state) ซึ่งครอบคลุมทั้ง สถานะย่อยแบบต่อเนื่อง (sequential substates) และสถานะย่อยแบบทำงานพร้อมกัน (concurrent substates) เป็นส่วนประกอบภายใน

3) สถานะซิง (synch state)

4) สถานะเริ่มต้นและสิ้นสุด (initial & final pseudostate)

5) สถานะประวัติ (history pseudostate) (เฉพาะไพแคลคูลัส)

6) สถานะแยกและเชื่อมสำหรับการทำงานพร้อมกัน (fork & join pseudostate)

7) สถานะตัวต่อและตัวเลือก (junction & choice Point)

8) เส้นการเปลี่ยนแปลง (transition) ซึ่งครอบคลุมทั้งส่วนของเหตุการณ์ (event) เงื่อนไข (condition) และ การกระทำ (actions)

9) แผนภาพย่อย (submachine) (เฉพาะไพแคลคูลัส)

1.3.1.2 สามารถแปลงการเชื่อมต่อระหว่างเส้นการเปลี่ยนแปลงและสถานะในแบบต่างๆ ดังนี้

1) สถานะที่มีเส้นการเปลี่ยนแปลงเชื่อมต่อไปยังอีกสถานะหนึ่งเพียงหนึ่งเส้น

2) สถานะที่มีเส้นการเปลี่ยนแปลงเชื่อมต่อไปยังสถานะอื่นๆ หลายเส้น

3) สถานะที่มีเส้นการเปลี่ยนแปลงที่เชื่อมต่อไปยังสถานะย่อยที่อยู่ภายในสถานะประกอบ ซึ่งครอบคลุมทั้งสถานะย่อยแบบต่อเนื่องและสถานย่อยแบบทำงานพร้อมกัน (เฉพาะไพแคลคูลัส)

4) สถานะที่มีเส้นการเปลี่ยนแปลงจากสถานะย่อยผ่านสถานะประกอบไปยังสถานะอื่นๆ (เฉพาะไพแคลคูลัส)

1.3.2 เพื่อสร้างกฎในการตรวจสอบความสัมพันธ์ระหว่างแผนภาพสถานะ โดยแบ่งออกเป็น 2 ส่วน คือ

1.3.2.1 การตรวจสอบความเท่าเทียมกันในด้านพฤติกรรมของวัตถุ คือ การตรวจสอบพฤติกรรมที่วัตถุตอบสนองต่อระบบว่าเท่าเทียมกันหรือไม่ โดยจะเกิดขึ้นมากในกรณีของการพัฒนาซอฟต์แวร์แบบอาศัยส่วนประกอบ (component based software development) ซึ่งจะพิจารณาการแทนที่วัตถุเก่าด้วยวัตถุใหม่ที่มีคุณสมบัติการทำงานเหมือนกัน หรือในกรณีของการออกแบบระบบในแต่ละระดับของรายละเอียด โดยสามารถเปรียบเทียบคุณสมบัติของระบบในระดับรายละเอียดย่อยและระดับนามธรรมว่าตรงกันได้

1.3.2.2 การตรวจสอบพฤติกรรมการทำงานของวัตถุต่างๆ ที่เกิดขึ้นเมื่อทำงานร่วมกับวัตถุอื่นภายในระบบ โดยสามารถตรวจสอบถึงพฤติกรรมที่ได้รับการออกแบบไว้ของวัตถุแต่ไม่ได้ถูกใช้งานเมื่อทำงานร่วมกับวัตถุอื่นภายในระบบได้

1.4 ประโยชน์ที่คาดว่าจะได้รับ

ผลของงานวิจัยนี้จะทำให้ได้ขั้นตอน และกฎในการแปลงแผนภาพสถานะไปเป็นการบรรยายในภาษารูปนัย คือ ซีอาร์อี และไพแคลคูลัส ทำให้สามารถนำวิธีการตรวจสอบและพิสูจน์ทางรูปนัยมาช่วยในขั้นตอนพัฒนาซอฟต์แวร์ได้ ส่งผลให้สามารถสร้างซอฟต์แวร์ที่มีคุณภาพได้มากยิ่งขึ้น

1.5 วิธีดำเนินการวิจัย

1.5.1 ศึกษาและทำความเข้าในการออกแบบซอฟต์แวร์ด้วยแผนภาพสถานะ รวมถึงศึกษาตัวอย่างการเขียนแผนภาพสถานะที่เป็นที่นิยม

1.5.2 ศึกษางานวิจัยที่เกี่ยวข้องกับการแปลงแผนภาพสถานะไปเป็นข้อกำหนดแบบรูปนัย

1.5.3 ศึกษาข้อกำหนดรูปนัยที่เหมาะสมเพื่อนำมาใช้เป็นภาษาเป้าหมายของการแปลงแผนภาพสถานะ

1.5.4 ออกแบบขั้นตอน และกฎการแปลงจากแผนภาพสถานะไปเป็นข้อกำหนดรูปนัยที่กำหนด

1.5.5 ออกแบบวิธีการในการตรวจสอบความสัมพันธ์ของแผนภาพจากข้อกำหนดรูปนัยที่ได้จากขั้นตอนการแปลง

1.5.6 สรุปผลการวิจัยและจัดทำรายงานวิทยานิพนธ์

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 แนวคิดและทฤษฎี

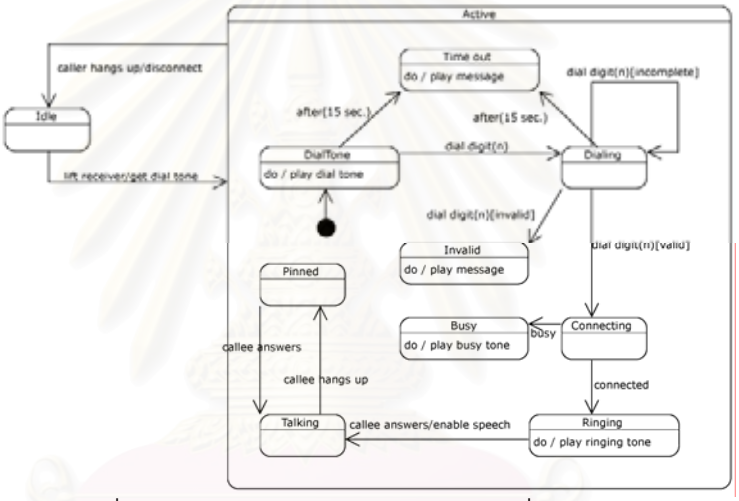
2.1.1 แผนภาพสถานะ (Statechart Diagram)

แผนภาพยูเอ็มแอลประกอบด้วยแผนภาพทั้งหมด 9 แผนภาพ คือ

- แผนภาพคลาส (class diagram) ใช้สำหรับแสดงข้อมูลคลาส ส่วนต่อประสาน (interface) และการเชื่อมต่อกัน ซึ่งเป็นแผนภาพสำคัญสำหรับการพัฒนาซอฟต์แวร์ในระบบเชิงวัตถุ สามารถนำไปเป็นต้นแบบโครงสร้างในการเขียนรหัสโปรแกรมได้
- แผนภาพวัตถุ (object diagram) ใช้สำหรับแสดงข้อมูลวัตถุ และความสัมพันธ์ระหว่างกัน
- แผนภาพยูสเคส (use case diagram) ใช้สำหรับแสดงข้อมูลยูสเคส (use cases) ผู้แสดง (actors) และความสัมพันธ์ระหว่างกัน ซึ่งช่วยอธิบายความต้องการของผู้ใช้ที่มีต่อระบบในระดับสูง
- แผนภาพลำดับ (sequence diagram) ใช้สำหรับแสดงการตอบโต้การทำงานระหว่างวัตถุภายในระบบ โดยอยู่ในมุมมองลำดับของเวลา
- แผนภาพความร่วมมือ (collaboration diagram) ใช้สำหรับแสดงการตอบโต้การทำงานระหว่างวัตถุภายในระบบเช่นเดียวกับแผนภาพลำดับ แต่จะแสดงในมุมมองของโครงสร้างความสัมพันธ์ระหว่างวัตถุ
- แผนภาพสถานะ (statechart diagram) ใช้สำหรับแสดงสถานะการเปลี่ยนแปลงของระบบ โดยจะประกอบไปด้วยสถานะ เส้นการเปลี่ยนแปลง เหตุการณ์ และการกระทำ
- แผนภาพกิจกรรม (activity diagram) ใช้สำหรับแสดงฟังก์ชันการทำงานต่างๆ ของระบบในลักษณะของลำดับการไหลของการทำงานหนึ่งๆ ไปยังอีกการทำงานหนึ่ง
- แผนภาพส่วนประกอบ (component diagram) ใช้สำหรับอธิบายโครงสร้างของส่วนประกอบต่างๆ ว่าเชื่อมต่อกันอย่างไร และการทำงานของส่วนประกอบแต่ละส่วนขึ้นอยู่กับส่วนประกอบส่วนอื่นๆ อย่างไร

- แผนภาพดิพลอยเมนท์ (deployment diagram) ใช้สำหรับอธิบายรูปแบบการจัดวางของระบบและส่วนประกอบต่างๆ ว่าต้องมีรูปแบบอย่างไรในการใช้งานจริง

ในงานวิจัยนี้เราได้ให้ความสนใจในแผนภาพสถานะ ซึ่งใช้สำหรับอธิบายพฤติกรรมของระบบ บอกสถานะทั้งหมดที่วัตถุสามารถเกิดขึ้นได้ และการเปลี่ยนแปลงสถานะของวัตถุเมื่อมีเหตุการณ์ต่างๆ เกิดขึ้น ซึ่งโดยทั่วไปแล้วแผนภาพสถานะแผนภาพหนึ่งจะใช้สำหรับอธิบายคลาสคลาสเดียว ปัจจุบันนี้มีการเขียนแผนภาพสถานะในหลายรูปแบบ แต่แผนภาพสถานะของยูเอ็มแอลนั้นมีพื้นฐานมาจากแผนภาพสถานะของเดวิด ฮาเรล [4] โดยรูปที่ 2.1 ข้างล่างนี้เป็นตัวอย่างระบบการสั่งซื้อสินค้าที่เขียนอธิบายด้วยแผนภาพสถานะ



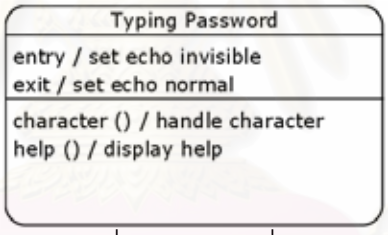
รูปที่ 2.1 แผนภาพสถานะแสดงระบบเครื่องโทรศัพท์

จากข้อกำหนดของแผนภาพนั้น ณ เวลาหนึ่งๆ ระบบจะต้องอยู่ในสถานะใดสถานะหนึ่งเท่านั้น หรือบางครั้งอยู่ในหลายสถานะในกรณีที่เป็นการทำงานแบบพร้อมกัน เมื่อมีเหตุการณ์ใดๆ เกิดขึ้นกับระบบแล้ว ระบบจะตอบรับต่อเหตุการณ์ที่เกิดขึ้นโดยการเปลี่ยนแปลงสถานะจากสถานะหนึ่งไปยังอีกสถานะหนึ่ง นอกจากนั้นเรายังอาจเพิ่มการอธิบายสิ่งที่ต้องกระทำ (actions) เข้าไปด้วยกับการเปลี่ยนแปลงที่เกิดขึ้น

แผนภาพสถานะยูเอ็มแอลประกอบด้วยสัญลักษณ์ที่ใช้สร้างแผนภาพดังต่อไปนี้

- สถานะทั่วไป (simple state) มีลักษณะเป็นสี่เหลี่ยมมุมโค้งดังรูปที่ 2.2 โดยภายในจะประกอบด้วยสองส่วน คือ
  - ชื่อสถานะ อยู่บริเวณตอนบนของสถานะ ซึ่งไม่สามารถมีสถานะที่มีชื่อเหมือนกันได้ภายในหนึ่งแผนภาพ

- การกระทำที่เกิดขึ้นภายในสถานะ อยู่บริเวณตอนล่างของสถานะ ส่วนนี้จะแสดงถึงสิ่งที่ต้องกระทำเมื่อระบบอยู่ในสถานะดังกล่าว ซึ่งสิ่งที่ต้องทำนี้สามารถระบุได้ทั้งหมดห้าชนิด คือ
  - *entry* ใช้สำหรับระบุสิ่งที่ต้องกระทำเมื่อระบบมีการเปลี่ยนแปลงเข้ามาสู่สถานะที่ระบุ
  - *exit* ใช้สำหรับระบุสิ่งที่ต้องกระทำเมื่อระบบมีการเปลี่ยนแปลงออกจากสถานะที่ระบุ
  - *do* ใช้สำหรับระบุถึงสิ่งที่ต้องกระทำอย่างต่อเนื่อง หรือจนเสร็จสมบูรณ์ในขณะที่ระบบอยู่ภายในสถานะที่ระบุ
  - *include* ใช้อ้างถึงแผนภาพย่อยของระบบ
  - ส่วนที่นอกเหนือจากสี่ชนิดข้างต้น คือ มีการระบุเหตุการณ์ เงื่อนไข และการกระทำ โดยเมื่อมีเหตุการณ์เกิดขึ้น และเงื่อนไขถูกต้อง จะกระทำการกระทำที่ระบุไว้ แต่สถานะจะไม่เปลี่ยนไปยังสถานะอื่น

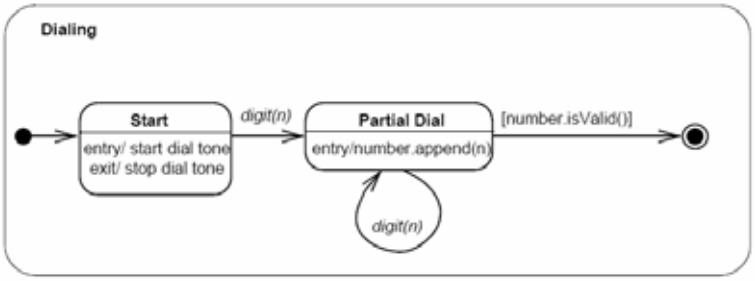


รูปที่ 2.2 สถานะทั่วไป

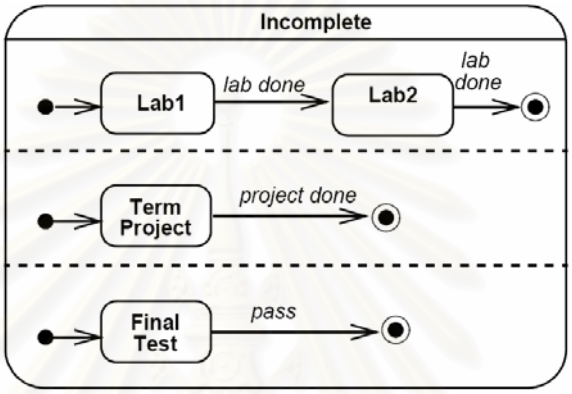
- สถานะสิ้นสุด (final state) มีลักษณะเป็นวงกลมทึบซึ่งมีวงกลมล้อมรอบอีกชั้น ดังรูปที่ 2.3 โดยใช้ระบุถึงการทำงานได้เสร็จเรียบร้อยแล้ว ซึ่งสถานะสิ้นสุดนี้ไม่สามารถมีเส้นการเปลี่ยนแปลงชี้ต่อไปยังสถานะอื่นได้

รูปที่ 2.3 สถานะเริ่มต้น

- สถานะประกอบ (composite state) มีลักษณะคล้ายกับสถานะทั่วไปแต่ภายในประกอบด้วยสถานะย่อยๆ เชื่อมต่อกัน หรือเป็นสถานะย่อยซึ่งทำงานพร้อมก็ได้ ดังรูปที่ 2.4 และรูปที่ 2.5

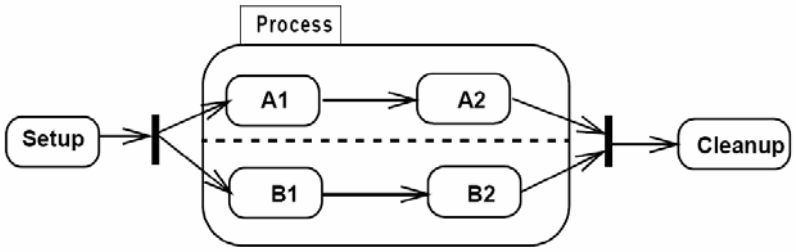


รูปที่ 2.4 สถานะประกอบแบบสถานะย่อยต่อเนื่อง



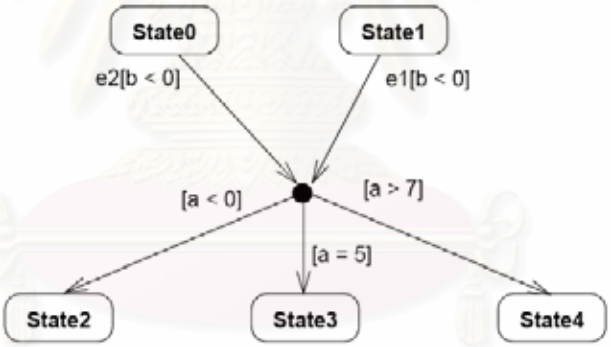
รูปที่ 2.5 สถานะประกอบแบบสถานย่อยทำงานพร้อมกัน

- เงื่อนไข (guard) เงื่อนไขจะมีลักษณะเป็นนิพจน์ที่ให้ความหมายเป็นถูกหรือผิดที่แสดงอยู่ในเส้นการเปลี่ยนแปลง หากให้ความหมายเป็นถูกเส้นการเปลี่ยนแปลงนั้นจะถูกอนุญาตให้เกิดการเปลี่ยนแปลงสถานะไปยังสถานะปลายทางได้ หากให้ความหมายเป็นผิดเส้นการเปลี่ยนแปลงจะไม่อนุญาตให้เกิดการเปลี่ยนสถานะ
- สถานะปลอม (pseudo state) เป็นสถานะแบบพิเศษเพื่อใช้ในการช่วยอธิบายพฤติกรรมของระบบ ซึ่งประกอบด้วย
  - สถานะเริ่มต้น (initial state) ใช้ระบุตำแหน่งเริ่มต้นของแผนภาพสถานะ มีลักษณะเป็นวงกลมทึบดังรูปที่ 2.5 และมีเส้นการเปลี่ยนแปลงหนึ่งเส้นเชื่อมโยงไปยังสถานะเริ่มต้นของระบบ
  - สถานะแยก (fork state) ใช้สำหรับแยกการทำงานไปสู่การทำงานแบบพร้อมกัน มีลักษณะเป็นเส้นตรงที่มีเส้นการเปลี่ยนแปลงขาเข้าหนึ่งเส้นและขาออกหลายเส้นดังรูปที่ 2.6



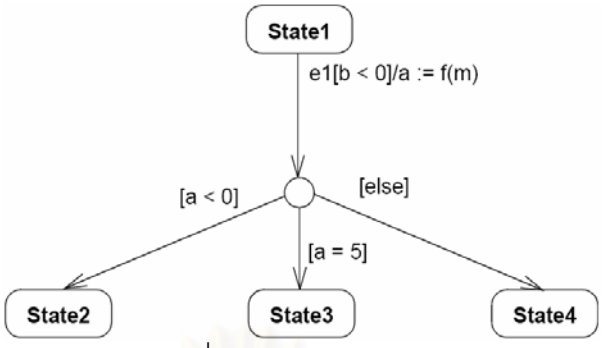
รูปที่ 2.6 สถานะแยกและเชื่อม

- สถานะเชื่อม (join state) ใช้สำหรับรวมเส้นการเปลี่ยนแปลงจากพื้นที่การทำงานพร้อมกันเข้าเป็นการทำงานเดียว มีลักษณะเป็นเส้นตรงที่มีเส้นการเปลี่ยนแปลงขาเข้าหลายเส้นและขาออกเพียงหนึ่งเส้นดังรูปที่ 2.6
- สถานะตัวต่อ (junction) ใช้สำหรับเป็นทางแยกหรือทางเชื่อมสำหรับเส้นการเปลี่ยนแปลงดังรูปที่ 2.7 โดยการเปลี่ยนแปลงสถานะจะพิจารณาเงื่อนไขในเส้นการเปลี่ยนแปลงทั้งหมดก่อนว่าถูกต้องหรือไม่ ถ้าถูกต้องการเปลี่ยนสถานะจะเปลี่ยนจากสถานะต้นทางไปยังสถานะปลายทางโดยไม่หยุดที่สถานะตัวต่อก่อน



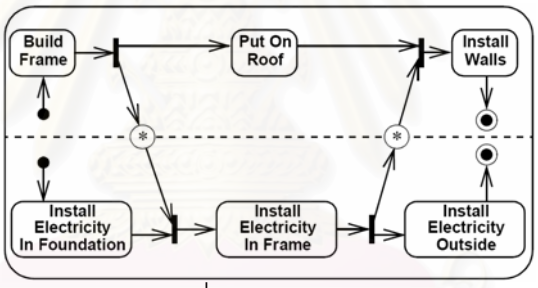
รูปที่ 2.7 สถานะตัวต่อ

- สถานะตัวเลือก (choice) ใช้สำหรับเป็นทางแยกหรือทางเชื่อมสำหรับเส้นการเปลี่ยนแปลงดังรูปที่ 2.8 โดยการพิจารณาเงื่อนไขในเส้นการเปลี่ยนแปลงหลังสถานะตัวเลือกจะถูกพิจารณาหลังจากสถานะของระบบได้เปลี่ยนมาอยู่ที่สถานะตัวเลือกเรียบร้อยแล้ว ในกรณีที่ตัวเลือกเส้นการเปลี่ยนแปลงหลังจากสถานะตัวเลือกนั้นไม่มีตัวใดมีเงื่อนไขถูกต้องและทำให้ไม่สามารถเปลี่ยนสถานะจากสถานะตัวเลือกไปยังสถานะปลายทางได้ จะถือว่าแผนภาพนั้นมีความบกพร่อง



รูปที่ 2.8 สถานะตัวเลือก

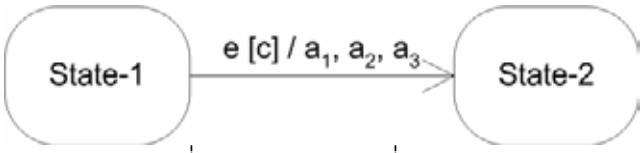
- สถานะซิง (synch state) เป็นสถานะแบบพิเศษที่หมายถึงการทำงานพร้อมกันระหว่างสองระบบที่ทำงานพร้อมกัน โดยจะมีลักษณะเป็นวงกลมวางทับอยู่บนเส้นแบ่งระหว่างระบบที่ทำงานพร้อมกันและมีจำนวนระบุภายในเพื่อแสดงความจุดังรูปที่ 2.9 จำนวนที่ระบุภายในนั้นอาจเป็นตัวเลขจำนวนบวก หรือดอกจันทน์ก็ได้ โดยดอกจันทน์จะหมายถึงความจุที่ไม่จำกัดจำนวน



รูปที่ 2.9 สถานะซิง

- เส้นการเปลี่ยนแปลง (transition) เส้นการเปลี่ยนแปลงจะมีลักษณะเป็นลูกศรเชื่อมต่อระหว่างสองสถานะ และมีคำอธิบายบนเส้นซึ่งประกอบด้วย เหตุการณ์, เงื่อนไข และสิ่งที่ต้องกระทำ ดังรูปที่ 2.10 ซึ่งอธิบายถึงว่า ถ้าเหตุการณ์ที่ระบุไว้เกิดขึ้นและเงื่อนไขเป็นจริง สถานะของระบบจะถูกเปลี่ยนจากสถานะต้นทางไปยังสถานะปลายทาง และประมวลผลการกระทำที่ระบุไว้ ลักษณะของคำอธิบายเส้นการเปลี่ยนแปลงเป็นดังนิพจน์ข้างล่าง

event – name '(' *comma – separated – parameter – list* ')' '[' *guard – condition* ']' '/' *action – ex pression*



รูปที่ 2.10 เส้นการเปลี่ยนแปลง

2.1.2 นิพจน์สม่ำเสมอ (Regular Expression)

นิพจน์สม่ำเสมอประกอบด้วยตัวอักษร (alphabet) อักษรว่าง (ε) และตัวดำเนินการทั้งหมด 3 ตัว คือ เชื่อมต่อ ($\cdot$) ทางเลือก ($+$) และคลีนสตาร์ (*) ซึ่งตัวดำเนินการแต่ละตัวเมื่อถูกนำมาใช้ในการเชื่อมต่อตัวอักษรมีความหมายดังนี้

- ตัวดำเนินการเชื่อมต่อ เมื่อเชื่อมระหว่างอักษรใดๆ ลำดับจะมีความสำคัญ คือ เหตุการณ์ที่อักษรที่อยู่ด้านซ้ายจะเกิดก่อนอักษรที่อยู่ด้านขวาของตัวดำเนินการ เช่น $a \cdot b = \{a \cdot b\}$ เราสามารถใช้ตัวเลขยกกำลังในกรณีที่ต้องการเชื่อมตัวอักษรตัวเดียวกันซ้ำกันหลายๆ ครั้งได้ เช่น $a^1 = \{a\}$, $a^2 = \{a \cdot a\}$ ในบางกรณีสามารถละเว้นการเขียนตัวดำเนินการ $\cdot$ ได้หากไม่ทำให้เกิดความสับสน เช่น $a \cdot b = ab$
- ตัวดำเนินการทางเลือก เมื่อเชื่อมระหว่างอักษรใดๆ หมายถึงเหตุการณ์ที่สามารถเกิดตัวอักษรใดก็ได้ เช่น $a + b = \{a,b\}$ ซึ่งตัวดำเนินการทางเลือกนี้มีคุณสมบัติของการสลับข้างได้ คือ $a + b = b + a$
- ตัวดำเนินการคลีนสตาร์ เมื่อใช้กับนิพจน์ใดๆ จะหมายถึงการรวมกันของการทำซ้ำของนิพจน์ตั้งแต่ 0 ครั้ง ไปเรื่อยๆ ไม่สิ้นสุด คือ $A^* = \bigcup_{i=0,1,\ldots} A^i$ ตัวอย่างเช่น

$$a^* = a^0 + a^1 + a^2 + \ldots = \{\varepsilon, a, aa, \ldots\}$$

2.1.3 ซีอาร์อี (CRE: Concurrent Regular Expression)

ซีอาร์อีเป็นส่วนเพิ่มเติมการทำงานของนิพจน์สม่ำเสมอ เพื่อช่วยให้นิพจน์สม่ำเสมอสามารถอธิบายระบบที่งานแบบพร้อมกันได้ โดยซีอาร์อีได้เพิ่มตัวดำเนินการเข้ามาอีก 4 ตัว คือ

- Interleaving ใช้สัญลักษณ์แทนคือ || โดยใช้สำหรับอธิบายการแทรกสลับกันของสองระบบที่ทำงานพร้อมกันอย่างอิสระ เช่น $a \,||\, b = \{ab, ba\}$ หรือ $ab \,||\, cd = \{abcd, acbd, cabd, cdab\}$ โดยถ้าหากเป็นการทำ interleaving ซ้ำ

ตัวเองจะใช้เครื่องหมายคล้ายกับยกกำลังแต่มีวงเล็บล้อมรอบเพื่อให้แตกต่างกัน คือ $A^{(2)} = A \| A$

- Alpha-closure ใช้สัญลักษณ์แทน คือ $^{\alpha}$ โดยความหมายคือใช้อธิบายการรวมกันของการทำ interleaving กับตัวเองตั้งแต่ 0 ครั้งไปเรื่อยๆ ไม่มีที่สิ้นสุด คือ $A^{\alpha} = \bigcup_{i=0,1,...} A^{(i)}$ ตัวอย่างเช่น

  $$(ab)^{\alpha} = (ab)^{(0)} + (ab)^{(1)} + (ab)^{2} + ... = \bigcup_{i=0,1,...} (ab)^{(i)}$$

- Synchronous Composition ใช้อธิบายการซิงโครนัสกันของระบบ 2 ระบบที่ทำงานพร้อมกัน เช่น $ab\,[\,]\,bc = \{abc\}$, $ab[\,]\,ac = \{abc, acb\}$ หรือ $ab[\,]\,ba = \{\,\}$ ในกรณีที่ระบบ 2 ระบบไม่มีส่วนที่ซิงโครนัสกันเราสามารถแทนที่ตัวดำเนินการด้วยตัวดำเนินการ interleaving ได้ เช่น $ab[\,]cd = ab \| cd$
- Renaming ใช้ช่วยสำหรับการเปลี่ยนชื่อตัวอักษรในนิพจน์

2.1.4 ไพแคลคูลัส (π-Calculus)

ไพแคลคูลัสเป็นแคลคูลัสที่อยู่บนพื้นฐานของการสื่อสารระหว่างกระบวนการ (process) การสื่อสารจะกระทำผ่านช่องที่ได้กำหนดไว้ โดยที่กระบวนการที่เป็นฝ่ายส่งค่า v ผ่านช่อง c จะใช้สัญลักษณ์แทน คือ $\bar{c}\langle v \rangle$ ซึ่งอาจกระทำการสื่อสารกับอีกกระบวนการหนึ่งซึ่งเป็นฝ่ายรับผ่านช่อง c โดยเก็บค่าไว้ใน w คือ $c(w) \cdot P$ และเปลี่ยนคุณสมบัติตัวเองเป็น P โดยความแตกต่างของไพแคลคูลัสจากแคลคูลัสของกระบวนอื่นๆ คือ สามารถสร้างช่องทางการสื่อสารใหม่ให้กับตัวกระบวนการได้โดยการรับค่าช่องทางการสื่อสารผ่านการสื่อสารกับกระบวนการอื่นๆ ซึ่งคุณสมบัติดังกล่าวนี้ส่งผลให้ความสามารถในการอธิบายคุณสมบัติต่างๆ ของระบบที่มีอยู่สูงขึ้น

วากยสัมพันธ์ของไพแคลคูลัสประกอบด้วยส่วนต่างๆ ดังนี้

- 0 หมายถึง ว่าง (Null) คือ กระบวนการที่ไม่มีการทำงาน
- $x(\vec{y}) \cdot P$ หมายถึง การรับค่า $y_1, y_2, ..., y_n$ จากช่อง x และเปลี่ยนคุณสมบัติตัวเองเป็น P กรณีของ $x(\vec{y}) \cdot 0$ สามารถเขียนในรูป $x(\vec{y})$ ได้โดยละเว้น 0 ไว้ในฐานที่เข้าใจ
- $\bar{x}\langle \vec{y} \rangle \cdot P$ หมายถึง การส่งค่า $y_1, y_2, ..., y_n$ ไปทางช่อง x และเปลี่ยนคุณสมบัติตัวเองเป็น P กรณีของ $\bar{x}\langle \vec{y} \rangle \cdot 0$ ก็เช่นกันสามารถเขียนในรูป $\bar{x}\langle \vec{y} \rangle$ ได้โดยละเว้น 0 ไว้ในฐานที่เข้าใจ
- $P \mid Q$ หมายถึง การทำงานพร้อมกันของกระบวนการ P และ Q

- $P+Q$ หมายถึง ทางเลือกเชิงไม่กำหนด (non-deterministic choice) ระหว่าง P หรือ Q
- $new\vec{x}$ หมายถึง การสร้างช่องสื่อสาร $x_1, x_2, ..., x_n$ ขึ้นมาใหม่ โดยช่องสื่อสารดังกล่าวจะถูกใช้งานเฉพาะในกลุ่มที่กำหนดเท่านั้น ระบบภายนอกจะไม่สามารถมองเห็นช่องสื่อสารดังกล่าว
- $!P$ หมายถึง การทำงานพร้อมกันจำนวนอนันต์ของ P คือ $!P = P|!P$
- $A(x_1, x_2, ..., x_n) = ...$ หมายถึง A ประกอบด้วย n พารามิเตอร์
- $[x=y]P$ หมายถึง ระบบจะมีคุณสมบัติเป็น P ก็ต่อเมื่อ $x=y$ ถ้าไม่จะเป็นว่าง
- τ หมายถึง การกระทำที่เกิดขึ้นภายใน ซึ่งเกิดจากการรับและส่งค่ากันเองภายในระบบ
- $P\{\vec{y}/\vec{x}\}$ หมายถึง การแทนที่ $\vec{x}$ ใน P ด้วย $\vec{y}$

ไพแคลคูลัสมีคุณสมบัติที่เรียกว่า ความสัมพันธ์ของการลดทอน (reduction relation) คือ ความสัมพันธ์ระหว่างกระบวนการ $P \longrightarrow Q$ โดยที่กระบวนการ P สามารถเปลี่ยนไปเป็นกระบวนการ Q ด้วยการประมวลผลเพียง 1 ขั้นตอน หรือการสื่อสารกันระหว่างตัวรับกับตัวส่ง ยกตัวอย่าง เช่น

- $a \cdot P \xrightarrow{a} P$ เป็นการประมวลผล a จากนั้นกระบวนการเปลี่ยนจาก $a \cdot P$ ไปเป็น P
- $\bar{a} \cdot P \xrightarrow{\bar{a}} P$ เป็นการประมวลผล $\bar{a}$ จากนั้นกระบวนการเปลี่ยนจาก $\bar{a} \cdot P$ ไปเป็น P
- $a \cdot P | \bar{a} \cdot Q \xrightarrow{a} P | \bar{a} \cdot Q$ เป็นการประมวลผล a จากนั้นกระบวนการเปลี่ยนจาก $a \cdot P | \bar{a} \cdot Q$ ไปเป็น $P | \bar{a} \cdot Q$
- $a \cdot P | \bar{a} \cdot Q \xrightarrow{\bar{a}} a \cdot P | Q$ เป็นการประมวลผล $\bar{a}$ จากนั้นกระบวนการเปลี่ยนจาก $a \cdot P | \bar{a} \cdot Q$ ไปเป็น $a \cdot P | Q$
- $a \cdot P | \bar{a} \cdot Q \xrightarrow{\tau} P | Q$ เป็นการประมวลผล τ คือ a และ $\bar{a}$ ถูกประมวลผลพร้อมกัน โดย $\bar{a}$ เป็นฝ่ายส่ง และ a เป็นฝ่ายรับ จากนั้นกระบวนการเปลี่ยนจาก $a \cdot P | \bar{a} \cdot Q$ ไปเป็น $P | Q$
- $new\, a(a \cdot P | \bar{a} \cdot Q) \xrightarrow{\tau} new\, a(P | Q)$ เป็นการประมวลผล τ คือ a และ $\bar{a}$ ถูกประมวลผลพร้อมกัน โดย $\bar{a}$ เป็นฝ่ายส่ง และ a เป็นฝ่ายรับ จากนั้นกระบวนการเปลี่ยนจาก $new\, a(a \cdot P | \bar{a} \cdot Q)$ ไปเป็น $new\, a(P | Q)$

จากตัวอย่างของการลดทอนจะเห็นได้ว่า $a \cdot P \mid \overline{a} \cdot Q$ สามารถเกิดการประมวลผลลดทอนได้ทั้งหมด 3 รูปแบบ คือ ประมวลผล a จะได้ $a \cdot P \mid \overline{a} \cdot Q \xrightarrow{a} P \mid \overline{a} \cdot Q$ ประมวลผล $\overline{a}$ จะได้ $a \cdot P \mid \overline{a} \cdot Q \xrightarrow{\overline{a}} a \cdot P \mid Q$ หรือประมวลผล τ จะได้ $a \cdot P \mid \overline{a} \cdot Q \xrightarrow{\tau} P \mid Q$ โดยการประมวลผลลดทอน a และ $\overline{a}$ นั้นเปรียบเสมือนระบบเกิดการสื่อสารคู่กับ $\overline{a}$ และ a ของสิ่งแวดล้อมภายนอกตามลำดับ แต่ τ นั้นเกิดจากการสื่อสารภายในของ a ใน $a \cdot P$ คู่กับ $\overline{a}$ ใน $\overline{a} \cdot Q$ หากเราต้องการให้ช่องการสื่อสาร a เกิดการสื่อสารกันเฉพาะภายในระบบเท่านั้นและไม่สามารถสื่อสารกับสิ่งแวดล้อมได้ เราสามารถซ่อนช่องสื่อสาร a จากระบบภายนอกได้โดยการเติม $newa$ คลุมระบบที่ต้องการ ดังตัวอย่าง $newa(a.P \mid \overline{a}.Q)$ จะหมายถึงช่องสัญญาณ a เป็นช่องสื่อสารที่รู้จักเฉพาะภายในระบบเท่านั้น ระบบภายนอกไม่สามารถสื่อสารกับ $a \cdot P \mid \overline{a} \cdot Q$ ผ่านช่องทาง a ได้

2.2 เอกสารและงานวิจัยที่เกี่ยวข้อง

2.2.1 “Formalizing sequence diagrams and state machines using Concurrent Regular Expression” โดย มิตซูทากะ โอกาซากิ, โทชิอากิ อาโอกิ และทาคูระ คาทายามะ [9]

งานวิจัยชิ้นนี้ได้เสนอวิธีการแปลงแผนภาพลำดับและแผนภาพสถานะไปเป็นซีอาร์อี และได้เสนอวิธีการสร้างแผนภาพสถานะจากแผนภาพลำดับ โดยการใช้ภาษาซีอาร์อีเป็นตัวกลางในการแปลง และได้เสนอวิธีการตรวจสอบความต้องกันระหว่างแผนภาพทั้งสองชนิดด้วยองค์ประกอบที่ถูกนำมาพิจารณาในส่วนของแผนภาพสถานะนั้นมีเพียงสถานะทั่วไปและเส้นการเปลี่ยนเท่านั้น ยังไม่ครอบคลุมองค์ประกอบของแผนภาพสถานะยูเอ็มแอลอีกจำนวนมาก

2.2.2 “Towards Formalizing UML State Diagrams in CSP” โดย มวน ยองและ ไมเคิล บัทเลอร์ [8]

งานวิจัยนี้ได้เสนอวิธีการแปลงแผนภาพสถานะไปเป็นซีเอสพี (CSP: Communicating and Sequential Processes) [10] ซึ่งเป็นแคลคูลัสของกระบวนการเช่นกัน โดยวิธีการแปลงนั้นได้เปรียบเทียบ สถานะของแผนภาพไปเป็นกระบวนการซีเอสพี และเหตุการณ์ของแผนภาพไปเป็นเหตุการณ์ของซีเอสพี และได้สร้างกฎการแปลงขึ้นมาสำหรับองค์ประกอบแต่ละตัวในแผนภาพและการเชื่อมต่อในแบบต่างๆ รวมถึงได้สร้างเครื่องมือในการแปลงแผนภาพสถานะไปเป็นภาษาซีเอสพีอย่างอัตโนมัติ ซึ่งภาษาซีเอสพีนี้สามารถนำเข้าไปยังเอฟดีอาร์ (FDR: Failures Divergence Refinement) [12] ซึ่งเป็นเครื่องมือตรวจสอบการแบ่งละเอียด (refinement checker) ได้องค์ประกอบในการสร้างแผนภาพสถานะที่งานวิจัยนี้ได้เสนอกฎการแปลงนั้นยังขาดในส่วนของ

สถานะประกอบที่สถานะย่อยภายในเป็นแบบทำงานพร้อมกัน ซึ่งมีความสำคัญมากในการใช้อธิบายระบบที่มีการทำงานแบบพร้อมกัน

2.2.3 “Extraction of π-Calculus specifications from UML sequence and state diagrams” โดย แคทรีนา คอเรนบลาท และคอแรโด พริอามี [5]

งานวิจัยนี้ได้เสนอวิธีการแปลงข้อกำหนดจากแผนภาพยูเอ็มแอลสถานะและลำดับไปเป็นไพแคลคูลัส โดยได้อาศัยข้อมูลจากแผนภาพลำดับเป็นตัวหลักและใช้ข้อมูลจากแผนภาพสถานะช่วยเสริมในส่วนรายละเอียด โดยองค์ประกอบของแผนภาพลำดับนั้นได้ถูกกำหนดวิธีการแปลงได้ครอบคลุมส่วนหลักๆ ทั้งหมด แต่ในส่วนของแผนภาพสถานะนั้นมีเพียงส่วนหลักสองส่วนเท่านั้น คือ สถานะ และเส้นการเปลี่ยนแปลง

2.2.4 “Symbolic Model Checking of UML Statechart Diagrams with an Integrated Approach” โดย วิตัส เอส ดับบิว แลม และจูเลียน แพดเจท [6]

งานวิจัยนี้ได้เสนอวิธีการแปลงแผนภาพสถานะยูเอ็มแอลไปเป็นไพแคลคูลัสอย่างเป็นระบบ ซึ่งส่วนการแปลงที่ได้รับนำเสนอนั้นจะครอบคลุมส่วนหลักๆ ขององค์ประกอบในการสร้างแผนภาพ และได้ทำการแปลงจากไพแคลคูคัสเป็นข้อมูลนำเข้าเพื่อใช้กับ NuSMV [2] ซึ่งเป็นเครื่องมือในการตรวจสอบต้นแบบ (model checker) ชนิดหนึ่ง แต่ทั้งนี้ผลที่ได้จากการแปลงแผนภาพไปเป็นไพแคลคูลัสไม่ได้อธิบายถึงความสัมพันธ์ระหว่างแผนภาพอื่นๆ ภายในระบบเดียวกัน ทำให้การตรวจสอบระบบสามารถทำได้เฉพาะภายในแผนภาพของแต่ละวัตถุเท่านั้น ไม่สามารถตรวจสอบภาพรวมการทำงานของระบบซึ่งมีวัตถุหลายๆ วัตถุทำงานด้วยกัน

บทที่ 3

การแปลงและตรวจสอบแผนภาพสถานะ

3.1 แนวคิดในการแปลงและตรวจสอบแผนภาพสถานะ

ในส่วนของเนื้อหางานวิจัยจะครอบคลุมสองส่วน คือ การแปลงแผนภาพสถานะไปเป็นข้อกำหนดแบบรูปนัย และการวิเคราะห์และตรวจสอบความสัมพันธ์ของแผนภาพสถานะจากข้อกำหนดแบบรูปนัยที่ได้จากขั้นตอนการแปลง ดังนี้

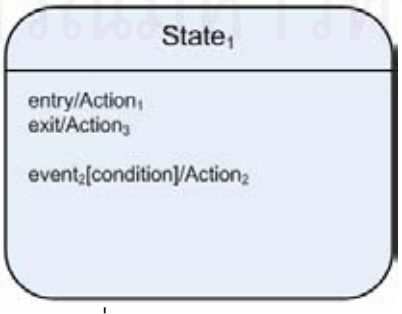
3.1.1 การแปลงแผนภาพสถานะไปเป็นข้อกำหนดรูปนัย

แผนภาพสถานะยูเอ็มแอลนั้นประกอบด้วยองค์ประกอบในการสร้างแผนภาพจำนวนมาก ซึ่งแต่ละองค์ประกอบก็มีความหมายการทำงานในตัวของมันเอง โดยองค์ประกอบแต่ละตัวยังสามารถนำมาเชื่อมต่อกับองค์ประกอบอื่นๆ ได้อีกหลายรูปแบบเพื่อให้เกิดเป็นแผนภาพสถานะ ซึ่งแต่ละแผนภาพจะอธิบายความหมายของพฤติกรรมที่เกิดขึ้นภายในระบบในรูปแบบลำดับของเหตุการณ์ต่างๆ ที่เกิดขึ้น ในลักษณะเดียวกันแต่ละนิพจน์ของแคลคูลัสของกระบวนการก็อธิบายถึงลำดับเหตุการณ์ต่างๆ ที่เกิดขึ้นเช่นกัน จึงมีความเป็นไปได้ในการใช้แคลคูลัสของกระบวนการนี้มาช่วยอธิบายแผนภาพสถานะยูเอ็มแอล ซึ่งในขั้นตอนการแปลงนั้นเราแบ่งกฎการแปลงออกเป็นทั้งหมด 2 ส่วน คือ การแปลงองค์ประกอบแต่ละส่วน และการแปลงรูปแบบการเชื่อมต่อชนิดต่างๆ

1) การแปลงองค์ประกอบแต่ละส่วน

ในเบื้องต้นนั้นองค์ประกอบหลักในการสร้างแผนภาพสถานะ คือ สถานะ และเส้นการเปลี่ยนแปลง

ในส่วนของสถานะนั้นสามารถกำหนดรายละเอียดภายในได้ดังรูปที่ 3.1



รูปที่ 3.1 ตัวอย่างสถานะ

รายละเอียดภายในนั้นจะเป็นการเปลี่ยนแปลงภายใน (internal transition) อยู่ในรูปแบบของ *เหตุการณ์ [เงื่อนไข] /การกระทำ* ซึ่งเมื่อมีเหตุการณ์ที่กำหนดเกิดขึ้นและเงื่อนไขตรงตามที่กำหนดจะทำการกระทำที่ได้กำหนดไว้ แต่สถานะจะยังคงอยู่ในสถานะเดิมไม่เปลี่ยนไปเป็นสถานะอื่น

โดยมีเหตุการณ์พิเศษอยู่สามชนิด คือ ***entry*** , ***exit*** และ ***do*** ซึ่งเหตุการณ์ ***entry*** จะเกิดเมื่อเกิดการเปลี่ยนสถานะมายังสถานะที่กำหนด ***exit*** จะเกิดเมื่อสถานะเปลี่ยนไปยังสถานะอื่น และ ***do*** ซึ่งทำให้เกิดการกระทำที่กำหนดไปเรื่อยๆ จนกว่าสถานะจะเปลี่ยนไปยังสถานะอื่นหรือการกระทำที่กำหนดเสร็จสิ้น

จะเห็นได้ว่าเหตุการณ์ต่างๆ ที่เกิดขึ้นนั้นจะมีลำดับของการเกิดพฤติกรรม ซึ่งหากเราทำการเปรียบเทียบกับแคลคูลัสของกระบวนการนั้นจะพบว่ามีการอธิบายลำดับของสิ่งต่างๆ ที่เกิดขึ้นเช่นกัน โดยเราสามารถกำหนดสถานะแต่ละตัวเป็นกระบวนการ และเหตุการณ์และการกระทำที่เกิดขึ้นเปรียบเสมือนลำดับของอักษรในแคลคูลัสของกระบวนการ จากรูปที่ 3.1 สามารถแทนด้วยนิพจน์ คือ $State_1 = entry \cdot Action_1 \cdot (event_2 \cdot conditionIsTrue \cdot Action_2)^* \cdot exit \cdot Action_3$

หากสถานะเป็นลักษณะของสถานะประกอบซึ่งมีสถานะภายในเป็นสถานะแบบลำดับก็สามารถเขียนนิพจน์ในลักษณะต่อเนื่องได้เช่นกัน แต่หากสถานะภายในเป็นสถานะแบบทำงานพร้อมกันจะต้องอาศัยตัวดำเนินที่สามารถอธิบายการทำงานแบบพร้อมกันด้วย

รูปที่ 3.2 ตัวอย่างเส้นการเปลี่ยนแปลง

ในส่วนของเส้นการเปลี่ยนแปลงดังรูปที่ 3.2 จะมีคำอธิบายประกอบเส้นการเปลี่ยนแปลงซึ่งอยู่ในรูปแบบ *เหตุการณ์ [เงื่อนไข] /การกระทำ* เช่นเดียวกับการเปลี่ยนแปลงภายในสถานะ โดยจากตัวอย่าง หากสถานะอยู่ที่ $State_A$ และมีเหตุการณ์ ***event*** เกิดขึ้นแล้วจะมีการพิจารณา ***condition*** ว่าถูกต้องหรือไม่ หากถูกต้องประมวลผล ***Action*** และสถานะจะเปลี่ยนจาก $State_A$ ไปเป็น $State_B$ แต่ในกรณีที่ ***condition*** ไม่ถูกต้อง จะไม่มีการประมวลผล ***Action*** และสถานะจะยังคงอยู่ที่ $State_A$ โดยไม่มีการเปลี่ยนแปลง จากพฤติกรรมดังกล่าวเราสามารถแทนด้วยนิพจน์คือ $State_A = event \cdot (ConditionIsTrue \cdot Action \cdot State_B + ConditionIsFalse \cdot State_A)$

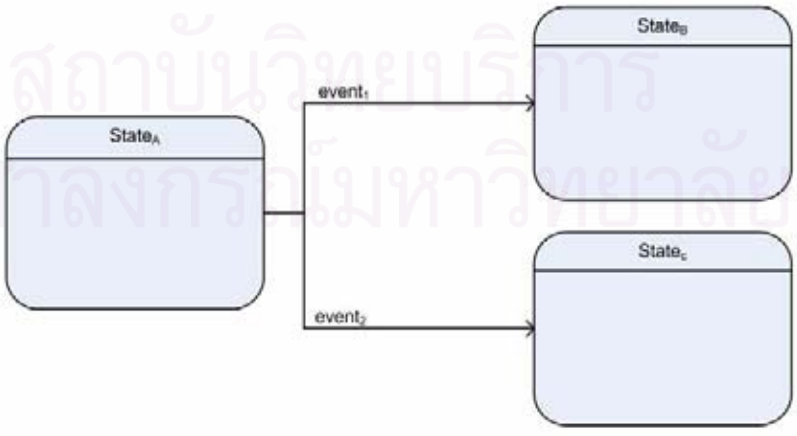
2) การแปลงรูปแบบการเชื่อมต่อชนิดต่างๆ

การเชื่อมต่อกันขององค์ประกอบจะแบ่งออกได้สองชนิด คือ การเชื่อมต่อแบบต่อเนื่องกันของเส้นการเปลี่ยนแปลง และการเชื่อมต่อแบบมีเส้นการเปลี่ยนแปลงออกจากสถานะมากกว่าหนึ่งเส้น โดยที่แบบต่อเนื่องเป็นดังรูปที่ 3.3

รูปที่ 3.3 การเชื่อมต่อกันของเส้นการเปลี่ยนแปลงแบบต่อเนื่อง

ลักษณะการเชื่อมต่อกันในแบบดังกล่าวเราสามารถใช้รูปแบบการอธิบายสถานะในหัวข้อ 1) ได้เลย โดยอธิบายสถานะแต่ละตัวด้วยกระบวนการของแคลคูลัสของกระบวนการ ดังนั้นจากตัวอย่างจะได้นิพจน์ดังนี้ $State_A = event_1 \cdot State_B$ และ $State_B = event_2 \cdot State_C$ ซึ่งจะเห็นได้ว่าหากระบบจะเปลี่ยนสถานะจาก $State_A$ ไปยัง $State_C$ ได้นั้นจะต้องเกิดเหตุการณ์ $event_1$ และ $event_2$ ก่อนตามลำดับ ซึ่งเราสามารถอธิบายพฤติกรรมขององค์ประกอบที่เชื่อมต่อกันในลักษณะนี้ได้เป็น $State_A = event_1 \cdot event_2 \cdot State_C$ ซึ่งหาพิจารณาเฉพาะพฤติกรรมของระบบจะพบว่าระบบนี้สามารถเกิดพฤติกรรมได้เพียงรูปแบบเดียวคือ $event_1 \cdot event_2$

ลักษณะการเชื่อมต่ออีกแบบ คือ มีเส้นการเปลี่ยนแปลงมากกว่าหนึ่งเส้นออกจากสถานะดังรูปที่ 3.4



รูปที่ 3.4 การเชื่อมต่อแบบเส้นการเปลี่ยนแปลงออกมากกว่าหนึ่งเส้น

ลักษณะการเชื่อมต่อกันแบบนี้จะเห็นได้ว่าสถานะสามารถเปลี่ยนจาก $State_A$ ไปเป็นสถานะ $State_B$ หรือสถานะ $State_C$ ก็ได้โดยขึ้นอยู่กับเหตุการณ์ที่เกิดขึ้น ในลักษณะนี้เราอธิบายโดยใช้ตัวดำเนินการทางเลือกช่วย คือ $State_A = event_1 \cdot State_B + event_2 \cdot State_C$ ซึ่งหากพิจารณาเฉพาะพฤติกรรมของระบบจะพบว่าระบบนี้สามารถเกิดพฤติกรรมได้สองรูปแบบ คือ $event_1$ หรือ $event_2$

3.1.2 การตรวจสอบความสัมพันธ์ของแผนภาพสถานะ

ในงานวิจัยนี้ได้ทำการตรวจสอบความสัมพันธ์ของแผนภาพสถานะสองรูปแบบ คือ

1) การตรวจสอบความเท่ากันของพฤติกรรมของวัตถุ โดยเมื่อทำการแปลงแผนภาพสถานะที่ต้องการตรวจสอบให้อยู่ในรูปของแคลคูลัสของกระบวนการแล้ว เราจะทำการเปรียบเทียบรูปแบบพฤติกรรมในจุดที่สนใจของแต่ละแผนภาพว่าถูกต้องตรงกันครบหรือไม่ โดยละเว้นพฤติกรรมย่อยหรือพฤติกรรมเสริมอื่นๆ ภายในระบบ

2) การตรวจสอบพฤติกรรมของวัตถุในการทำงานร่วมกับวัตถุอื่นภายในระบบ โดยวิเคราะห์พฤติกรรมของวัตถุที่เราได้ทำการออกแบบไว้แบบเดี่ยวเปรียบเทียบกับพฤติกรรมของวัตถุเมื่อทำงานร่วมกันกับวัตถุอื่นภายในระบบ ซึ่งวิธีการตรวจสอบจะคล้ายๆ กับข้อ 1) คือเปรียบเทียบพฤติกรรมที่เป็นไปได้ทั้งหมดของวัตถุว่าเกิดขึ้นได้ทั้งหมดภายในระบบหรือไม่

3.2 การแปลงแผนภาพสถานะไปเป็นซีอาร์อี

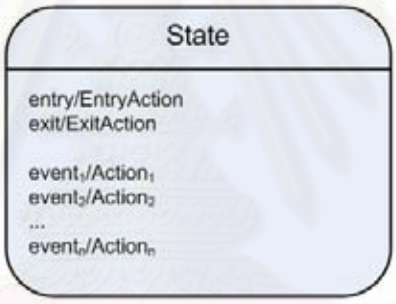
การแปลงพฤติกรรมต่างๆ ของระบบไปเป็นซีอาร์อีนั้น เรากำหนดให้ เหตุการณ์ เงื่อนไข และการกระทำ ที่เกิดขึ้นภายในระบบเปรียบเสมือนตัวอักษรของซีอาร์อี ซึ่งหากพฤติกรรมใดๆ มีลำดับของการเกิดขึ้นที่ต่อเนื่องกัน เราสามารถอธิบายพฤติกรรมนั้นได้ด้วยนิพจน์ของซีอาร์อีในลักษณะการเขียนพฤติกรรมนั้นๆ ต่อเนื่องกัน โดยเชื่อมระหว่างพฤติกรรมสองพฤติกรรมนั้นด้วยสัญลักษณ์ “ตัวดำเนินการเชื่อมต่อ” เช่น ในระบบหนึ่งๆ หากมีเหตุการณ์ *event* เกิดขึ้นแล้วจะต้องมีการกระทำ *action* เกิดขึ้นเพื่อตอบสนองต่อเหตุการณ์นั้น ดังนั้นลำดับพฤติกรรมของระบบที่เกิดขึ้น คือ *event* และ *action* ตามลำดับ ซึ่งเราทำการแทนพฤติกรรมลักษณะนี้ด้วยนิพจน์ของซีอาร์อี คือ $event \cdot action$ แต่หากในกรณีที่ระบบมีพฤติกรรมในลักษณะทางเลือก เราสามารถอธิบายพฤติกรรมนั้นได้ด้วยนิพจน์ของซีอาร์อีในลักษณะการเชื่อมต่อพฤติกรรมนั้นๆ ด้วยตัวดำเนินการทางเลือก เช่น ในระบบหนึ่งๆ ณ เวลาหนึ่ง หากมีเหตุการณ์ที่สามารถเกิดขึ้นกับระบบได้สองเหตุการณ์ คือ $event_A$ และ $event_B$ จะต้องมีการกระทำเพื่อตอบสนองเหตุการณ์

นั้นๆ คือ $action_A$ และ $action_B$ ตามลำดับ ดังนั้นเราจะได้ลำดับของพฤติกรรมที่สามารถเป็นไปได้สองแบบ คือ $event_A \cdot action_A$ หรือ $event_B \cdot action_B$ โดยที่พฤติกรรมของระบบจะเป็นแบบใดนั้นจะขึ้นอยู่กับเหตุการณ์ที่เกิดขึ้น ในลักษณะนี้เราแทนด้วยนิพจน์ที่ใช้ตัวดำเนินการทางเลือกเป็นตัวเชื่อมคือ $event_A \cdot action_A + event_B \cdot action_B$

จากที่ได้อธิบายให้เห็นข้างต้นเป็นเพียงพื้นฐานของการแปลงเท่านั้น แต่องค์ประกอบต่างๆ ที่ใช้สร้างแผนภาพสถานะยูเอ็มแอลนั้นมีพฤติกรรมที่ซับซ้อนทำให้ต้องสร้างกฎในการอธิบายเพื่อให้ครอบคลุมต่อพฤติกรรมต่างๆ ที่สามารถเกิดขึ้นได้ ซึ่งหัวข้อข้างล่างจะเป็นการนำเสนอกฎในการแปลงสำหรับองค์ประกอบต่างๆ ของแผนภาพ

3.2.1 องค์ประกอบต่างๆ ในการสร้างแผนภาพ

1) สถานะทั่วไป (simple state)



รูปที่ 3.5 สถานะทั่วไป

จากสถานะทั่วไปดังรูปที่ 3.5 เมื่อระบบเปลี่ยนแปลงมาสู่สถานะ *State* จะทำให้เกิดเหตุการณ์ *entry* และเกิดการกระทำ *EntryAction* ตามลำดับ ขณะที่ระบบคงอยู่ในสถานะ *State* นี้ หากมีเหตุการณ์ *ie* ใดๆ เกิดขึ้นภายใน ก็จะเกิดการกระทำ $Action_{ie}$ ควบคู่กันไปตามลำดับ ซึ่งขั้นตอนของเหตุการณ์และการกระทำนี้สามารถเกิดขึ้นซ้ำๆ ได้ จนกระทั่งระบบต้องการเปลี่ยนสถานะไปเป็นสถานะอื่น จะทำให้เกิดเหตุการณ์ *exit* และการกระทำ *ExitAction* ตามลำดับ ซึ่งจากพฤติกรรมดังกล่าว เราแทนด้วยนิพจน์ ดังนี้

$$entry \cdot EntryAction \cdot (\sum_{ie \in InternalEvent} ie \cdot Action_{ie})^* \cdot Exit \cdot Action$$

โดยที่

entry คือ เหตุการณ์ที่เกิดขึ้นเมื่อระบบเปลี่ยนสถานะ

EntryAction คือ การกระทำที่ตอบสนองเมื่อเกิดเหตุการณ์ *entry*

ie คือ เหตุการณ์ที่เกิดขึ้นภายในสถานะ ซึ่งไม่มีผลทำให้สถานะของระบบเปลี่ยนไป

$Action_{ie}$ คือ การกระทำที่ตอบสนองเมื่อเกิดเหตุการณ์ภายใน *ie*

exit คือ เหตุการณ์ที่เกิดขึ้นเมื่อระบบจะเปลี่ยนไปยังสถานะอื่น

ExitAction คือ การกระทำที่ตอบสนองเมื่อเกิดเหตุการณ์ *exit*

2) เส้นการเปลี่ยนแปลง (transition)

รูปที่ 3.6 เส้นการเปลี่ยนแปลง

จากเส้นการเปลี่ยนแปลงดังรูปที่ 3.6 เมื่อเกิดเหตุการณ์กระตุ้นขึ้นเพื่อให้ระบบเปลี่ยนสถานะ จะต้องพิจารณาเงื่อนไขภายในก่อน หากเงื่อนไขถูกต้องจะมีการกระทำการกระทำที่กำหนดไว้ และระบบเปลี่ยนไปยังสถานะที่ต้องการ ในส่วนของเงื่อนไขนั้น เราเลือกใช้สัญลักษณ์ฟังก์ชัน $g()$ แทนการตรวจสอบความถูกต้องของเงื่อนไข โดยหากเงื่อนไขไม่ถูกต้องเส้นการเปลี่ยนแปลงนั้นจะไม่ถูกกระทำ และระบบจะไม่มีการเปลี่ยนแปลงไปยังสถานะปลายทาง ซึ่งเราสามารถแทนด้วยนิพจน์ ดังนี้

$$event \cdot g(condition) \cdot Action$$

โดยที่

event คือ เหตุการณ์ของเส้นการเปลี่ยนแปลง ซึ่งเป็นตัวกระตุ้นให้ระบบเปลี่ยนสถานะ

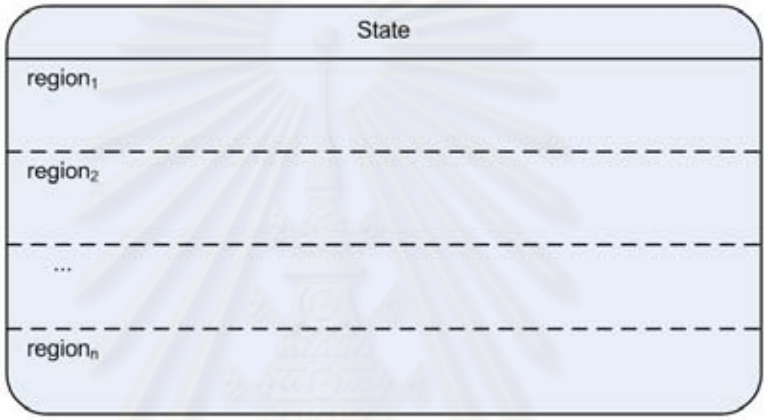
condition คือ เงื่อนไขของเส้นการเปลี่ยนแปลง ซึ่งเป็นตัวพิจารณาการเปลี่ยนแปลงสถานะว่ายอมให้เกิดขึ้นได้หรือไม่

$g()$ คือ การฟังก์ชันตรวจสอบเงื่อนไข ซึ่งหากผลที่ได้จากการประมวลผลเงื่อนไขภายในเป็นจริงจะอนุญาตให้มีการดำเนินการคำสั่งต่อไป

Action คือ การกระทำที่ตอบสนองเมื่อเกิดการเปลี่ยนแปลงสถานะ

3) สถานะประกอบ (composite state)

สถานะประกอบ จะมีสองรูปแบบ คือ สถานะประกอบที่มีสถานะย่อยแบบทำงานพร้อมกัน และสถานะประกอบที่มีสถานะย่อยแบบต่อเนื่อง ซึ่งสถานะประกอบที่มีสถานะย่อยแบบทำงานพร้อมกันนั้น จะประกอบด้วยพื้นที่ย่อยภายในสองพื้นที่ขึ้นไป ซึ่งมีการทำงานพร้อมกัน ส่วนสถานะประกอบที่มีสถานะย่อยแบบต่อเนื่องจะเปรียบเสมือนสถานะประกอบที่มีพื้นที่ย่อยภายในแค่เพียงพื้นที่เดียว ซึ่งการอธิบายจะใช้วิธีการเหมือนการอธิบายแผนภาพทั่วไป



รูปที่ 3.7 สถานะย่อยแบบทำงานพร้อมกัน

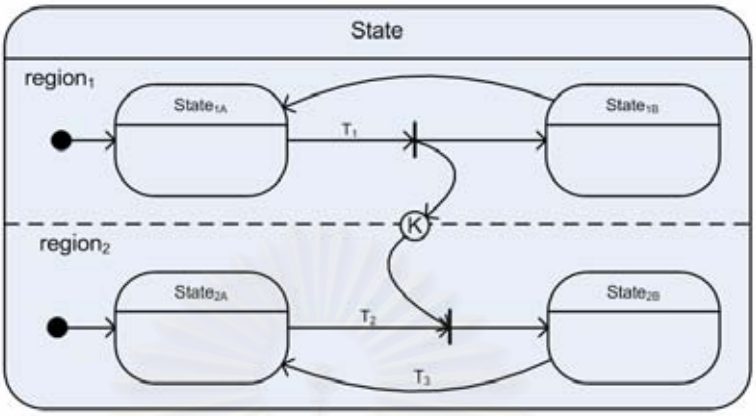
ในส่วนของสถานะย่อยแบบทำงานพร้อมกันดังรูปที่ 3.7 นั้น จะประกอบด้วยพื้นที่ย่อยๆ ภายในสถานะประกอบที่มีการทำงานในลักษณะพร้อมกัน โดยภายในพื้นที่ย่อยๆ ดังกล่าวจะประกอบไปด้วยสถานะที่เชื่อมต่อกันในรูปแบบต่างๆ เช่นกัน ซึ่งจากรูปเราสามารถแทนด้วยนิพจน์ดังนี้

$$region_1[]region_2[]...[]region_n$$

โดยที่

$region_i$ คือ พื้นที่ย่อยภายในสถานะประกอบ ซึ่งภายในจะประกอบด้วยองค์ประกอบต่างๆ เชื่อมต่อกันเสมือนแผนภาพย่อย ซึ่งสามารถอธิบายด้วยกฎวิธีการแปลงทั่วไป

4) สถานะซิง (synch state)



รูปที่ 3.8 สถานะซิง

จากสถานะซิงในรูปที่ 3.8 นั้น แสดงการพฤติกรรมการทำงานเชื่อมต่อกันระหว่างพื้นที่ย่อยภายในสถานะประกอบ โดยพื้นที่ $region_2$ จะมีการเปลี่ยนแปลงสถานะภายในได้จะขึ้นอยู่กับอีกพื้นที่ $region_1$ ซึ่งใน $region_2$ นั้น หากระบบต้องการเปลี่ยนสถานะจาก $State_{2A}$ ไปเป็นสถานะ $State_{2B}$ ที่สถานะซิงจะต้องมีโทเคนภายใน ซึ่งจำนวนโทเคนภายในจะเกิดขึ้นได้จากการเปลี่ยนแปลงของสถานะใน $region_1$ จาก $State_{1A}$ ไปยัง $State_{1B}$ ดังนั้นในการอธิบายจึงใช้รูปแบบของพื้นที่ย่อยๆ ที่มีความสัมพันธ์กันทำงานพร้อมกัน และมีการระบุข้อกำหนดของการเปลี่ยนแปลงสถานะสำหรับช่วงการเชื่อมต่อของสถานะซิง ดังนี้

$$region_1[]region_2[]constraint$$

โดยที่

$region_1$ คือ พื้นที่ย่อยภายในสถานะประกอบเกิดการวนรอบของเส้นเปลี่ยนแปลง T_1 ซึ่งสามารถอธิบายด้วยนิพจน์ คือ T_1^*

$region_2$ คือ พื้นที่ย่อยภายในสถานะประกอบเกิดการวนรอบของเส้นการแปลี่ยนแปลง T_2 และ T_3 ซึ่งสามารถอธิบายด้วยนิพจน์เช่นกัน คือ $(T_2 \cdot T_3)^*$

constraint คือ ข้อกำหนดที่ทำให้พื้นที่ย่อยสองส่วนทำงานสัมพันธ์กันอันเนื่องมาจากสถานะซิง นิพจน์ที่ได้จะขึ้นอยู่กับค่า K ที่กำหนดภายในสถานะซิง ในที่นี้แบ่งได้สองรูปแบบ คือ

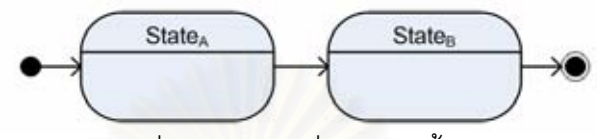
- ค่า K เป็นจำนวนเต็มบวกใดๆ

$$((T_1 \cdot T_1^* \cdot T_2 \cdot T_3)^*)^{(k)} \,||\, (T_2 \cdot T_1 \cdot T_1^* \cdot T_3)^*$$

- ค่า K เป็น *

$$((T_1 \cdot T_1^* \cdot T_2 \cdot T_3)^*)^\alpha \,||\, (T_2 \cdot T_1 \cdot T_1^* \cdot T_3)^*$$

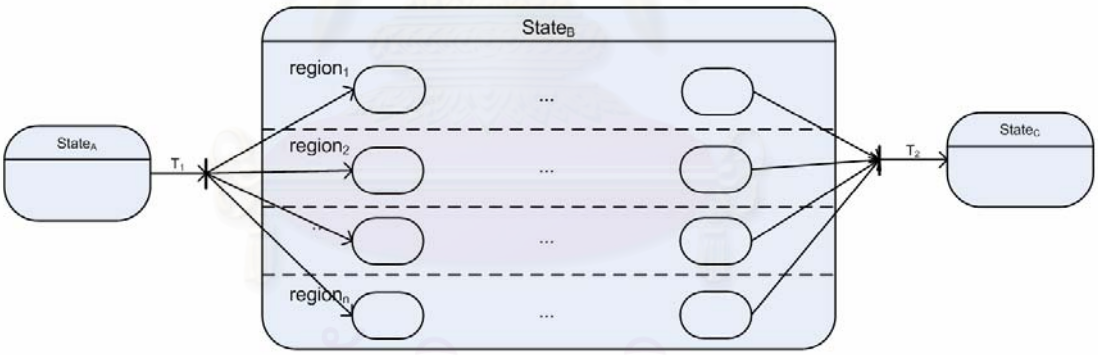
5) สถานะเริ่มต้นและสิ้นสุด (initial & final pseudostate)



รูปที่ 3.9 สถานะเริ่มต้นและสิ้นสุด

สถานะเริ่มและสถานะสิ้นสุดดังรูปที่ 3.9 ไม่มีการกระทำหรือเหตุการณ์ใดๆ ภายใน และเป็นเพียงสัญลักษณ์บอกจุดเริ่มต้นและสิ้นสุดของสถานะเท่านั้น ดังนั้นในที่นี้จึงไม่มีการอธิบายพฤติกรรมของสถานะดังกล่าวด้วยนิพจน์ใดๆ

6) สถานะแยกและเชื่อมสำหรับการทำงานพร้อมกัน (fork & join pseudostate)



รูปที่ 3.10 สถานะแยกและเชื่อม

จากรูปที่ 3.10 ซึ่งจำลองการทำงานของสถานะแยกและเชื่อม ซึ่งลักษณะการทำงานของระบบที่เชื่อมต่อจากสถานะแยกนั้นจะมีลักษณะเกิดขึ้นพร้อมกัน คือ มีสถานะที่ปัจจุบันได้มากกว่า 1 แต่หลังจากการทำงานผ่านสถานะเชื่อมแล้ว สถานะปัจจุบันที่มากกว่า 1 จะถูกรวมเหลือเพียงหนึ่ง เราสามารถอธิบายได้ด้วยนิพจน์ ดังนี้

$$T_1 \cdot (region_1 [] region_2 [] ... [] region_n) \cdot T_2$$

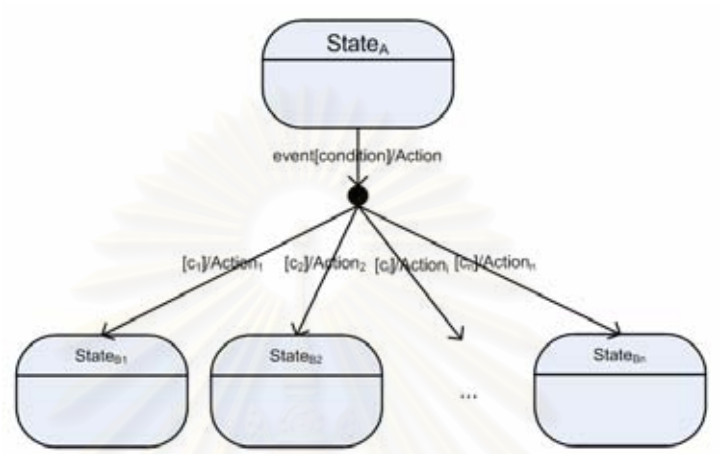
โดยที่

region คือ พื้นที่ย่อยภายในที่เชื่อมต่อจากสถานะแยก

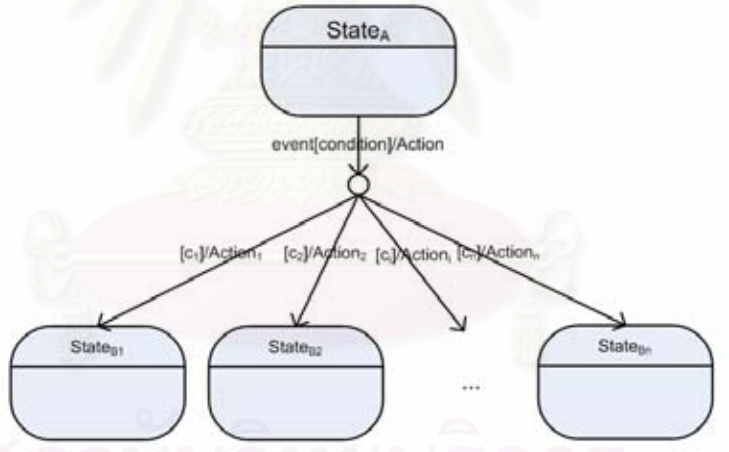
T_1 คือ เส้นการเปลี่ยนแปลงไปยังสถานะแยก

T_2 คือ เส้นการเปลี่ยนแปลงออกจากสถานะเชื่อม

7) สถานะตัวต่อและตัวเลือก (junction & choice point)



รูปที่ 3.11 สถานะตัวต่อ



รูปที่ 3.12 สถานะตัวเลือก

จากรูปที่ 3.11 และ 3.12 ของสถานะตัวต่อและสถานะตัวเลือก การทำงานจะมีลักษณะคล้ายคลึงกัน โดยความแตกต่างคือเงื่อนไขของเส้นการเปลี่ยนแปลงหลังจากสถานะตัวเลือกจะถูกพิจารณาหลังจากเส้นการเปลี่ยนจากสถานะต้นทางมายังสถานะเลือกซึ่งเงื่อนไขอาจมีผลกระทบจากการประมวลผลเส้นการเปลี่ยนแปลงดังกล่าว แต่ในส่วนของซีอาร์อีแล้วการกระทำต่างๆ จะเป็นผลต่อเนื่องจากเหตุการณ์ก่อนหน้าอยู่แล้ว ดังนั้นการอธิบายสถานะตัวต่อและตัวเลือกทั้งสองแบบนี้จึงไม่มีความแตกต่าง โดยสามารถใช้นิพจน์อ้างอิงได้ดังนี้

$$T_1 \cdot (\sum_{i=1}^{n} g(c_i) \cdot Action_i)$$

โดยที่

T_1 คือ นิพจน์แสดงเส้นเปลี่ยนแปลงจากสถานะต้นทางไปยังสถานะตัวต่อหรือตัวเลือก

c_i คือ เงื่อนไขของเส้นเปลี่ยนแปลงแต่ละเส้นซึ่งเชื่อมจากสถานะตัวต่อหรือตัวเลือกไปยังสถานะปลายทาง

$Action_i$ คือ การกระทำของเส้นเปลี่ยนแปลงแต่ละเส้นซึ่งเชื่อมจากสถานะตัวต่อหรือตัวเลือกไปยังสถานะปลายทาง

3.2.2 การเชื่อมต่อกันระหว่างองค์ประกอบประเภทต่างๆ

1) สถานะที่มีเส้นการเปลี่ยนแปลงออกหนึ่งเส้น

รูปที่ 3.13 สถานะที่มีเส้นการเปลี่ยนแปลงออกหนึ่งเส้น

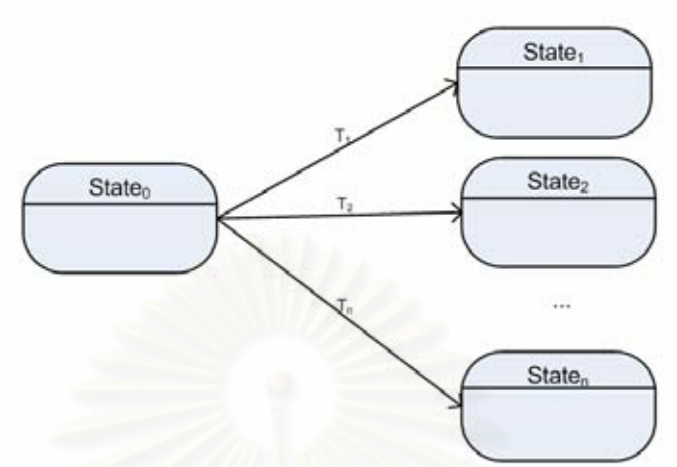
จากรูปที่ 3.13 หากมีเส้นการเปลี่ยนแปลงออกจากสถานะเพียงหนึ่งเส้นเราสามารถเขียนบรรยายระบบได้ด้วย T ซึ่งแทนนิพจน์ของสถานะดังกล่าว และหากมีการเชื่อมต่อกันของสถานะในลักษณะของลำดับ การบรรยายพฤติกรรมของระบบจะสามารถบรรยายในลักษณะของเหตุการณ์ที่เกิดต่อเนื่องกัน คือ $State_0 \cdot T_1 \cdot State_1 \cdot T_2 \cdot State_2 \cdot \ldots \cdot T_n State_n$ ซึ่งในกรณีที่สถานะต่างๆ ไม่มีพฤติกรรมภายใน เราสามารถละสัญลักษณ์ของแต่ละสถานะได้ โดยแทนด้วยนิพจน์ดังนี้

$$T_1 \cdot T_2 \cdot \ldots \cdot T_n$$

โดยที่

T_i คือ นิพจน์ของเส้นเปลี่ยนแปลงแต่ละเส้นที่เชื่อมต่อกัน

2) สถานะที่มีเส้นการเปลี่ยนแปลงออกมากกว่าหนึ่งเส้น



รูปที่ 3.14 สถานะที่มีเส้นเปลี่ยนแปลงออกมากกว่าหนึ่งเส้น

หากสถานะมีเส้นการเปลี่ยนแปลงออกจากสถานะมากกว่าหนึ่งเส้นแล้ว เราสามารถแทนได้ด้วยนิพจน์ $T_1 \cdot State_1 + T_2 \cdot State_2 + ... + T_n State_n$ ซึ่งในกรณีที่สถานะปลายทางต่างๆ ไม่มีเหตุการณ์ภายใน เราสามารถละสัญลักษณ์ของแต่ละสถานะได้ โดยแทนด้วยนิพจน์ ดังนี้

$$\sum_{i=1}^{n} T_i$$

โดยที่

T_i คือ นิพจน์ของเส้นเปลี่ยนแปลงแต่ละเส้นที่ออกจากสถานะต้นทาง

3.3 การแปลงแผนภาพสถานะไปเป็นไพแคลคูลัส

จากส่วนการแปลงแผนภาพสถานะไปเป็นซีอาร์อีนั้น พบว่าการแปลงให้ครอบคลุมส่วนประกอบต่างๆ ทั้งหมดในแผนภาพสถานะยูเอ็มแอลเป็นไปได้ยาก เนื่องจากความซับซ้อนของตัวองค์ประกอบ การเชื่อมต่อองค์ประกอบในลักษณะต่างๆ และข้อจำกัดในการอธิบายของตัวภาษาเอง ดังนั้นจึงได้นำเสนอไพแคลคูลัสขึ้นมาเพื่อเป็นภาษาปลายทาง ซึ่งรูปแบบการแปลงจะมีความคล้ายคลึงกับซีอาร์อี คือ มีการแปลงองค์ประกอบและพิจารณารูปแบบการเชื่อมต่อกัน แต่ได้เปลี่ยนแนวคิดของการแปลงใหม่ในส่วนของการเชื่อมต่อขององค์ประกอบ โดยจากเดิมนั้นจะสังเกตจากรูปแบบการเชื่อมต่อของส่วนประกอบต่างๆ ว่าเชื่อมต่อกันโดยลักษณะใด และทำการแทนการเชื่อมต่อนั้นโดยใช้สัญลักษณ์และนิพจน์ที่เหมาะสม แต่ในรูปแบบใหม่จะมองในลักษณะส่วนประกอบต่างๆ ในระดับเดียวกันมีการทำงานในลักษณะพร้อมกันทั้งหมด และองค์ประกอบแต่ละตัวมีสถานะเป็นสถานะไม่ทำงานและสถานะทำงาน โดยที่ในแผนภาพหนึ่งในระยะเวลาหนึ่งๆ

ที่ระดับเดียวกัน จะมีองค์ประกอบที่อยู่ในสถานะทำงานอยู่เพียงหนึ่งกลุ่ม คือ สถานะเพียงหนึ่งสถานะและเส้นการเปลี่ยนที่เชื่อมออกจากสถานะนั้น ซึ่งการเปลี่ยนสถานะขององค์ประกอบแต่ละส่วนจากสถานะไม่ทำงานไปสู่สถานะทำงานนั้น จะเกิดจากการส่งสัญญาณจากองค์ประกอบที่เชื่อมต่อกันภายในระบบ โดยไม่ได้เกิดจากสัญญาณภายนอก เว้นแต่สถานะเริ่มต้นในแผนภาพซึ่งจะต้องถูกกำหนดให้เป็นสถานะทำงานจากสัญญาณภายนอกเมื่อระบบเริ่มต้นทำงาน

เนื้อหาในหัวข้อนี้จะขอนำเสนอกฎการแปลงแผนภาพสถานะยูเอ็มแอลไปเป็นไพแคลคูลัส โดยแบ่งการแปลงออกเป็นสองส่วน คือ การแปลงองค์ประกอบต่างๆ ในการสร้างแผนภาพ และการแปลงลักษณะการเชื่อมต่อขององค์ประกอบในแบบต่างๆ ซึ่งนิพจน์ที่ได้จะมีความซับซ้อนสูง ดังนั้นผู้เขียนจึงนำเสนอกรณีศึกษาในบทถัดไปซึ่งประกอบไปด้วยขั้นตอนการแปลงอย่างละเอียด เพื่อให้ผู้อ่านสามารถทำความเข้าใจกฎต่างๆ และขั้นตอนการแปลงได้ง่ายยิ่งขึ้น

3.3.1 องค์ประกอบต่างๆ ในการสร้างแผนภาพ

1) สถานะทั่วไป (simple state)

ในลักษณะของการแปลง กำหนดให้เมื่อระบบเริ่มต้น สถานะที่ต้องการแปลงจะมีค่าเป็นไม่ทำงาน และเมื่อระบบเปลี่ยนสถานะมายังสถานะดังกล่าว จะเกิดเหตุการณ์ $\overline{entry_S}$ และการกระทำที่ตอบสนองต่อเหตุการณ์นั้น ซึ่งจะมีการส่งสัญญาณ $\overline{setActiveState}\langle entry_S \rangle$ ไปแจ้งถึงสถานะปัจจุบันด้วยเพื่อช่วยให้สถานะประวัติสามารถบันทึกสถานะปัจจุบันของระบบไว้ได้ จากนั้นจะมีการส่งสัญญาณไปให้เส้นการเปลี่ยนแปลงที่เชื่อมออกจากสถานะอยู่ในสถานะทำงานเพื่อรอตอบสนองต่อเหตุการณ์ต่างๆ ที่เกิดขึ้น ซึ่งสถานะที่รอการตอบสนองนี้จะเรียกว่าสถานะทำงาน โดยหากระบบได้รับสัญญาณเพื่อจะเปลี่ยนสถานะไปยังสถานะอื่น จะมีการส่งสัญญาณ $\overline{exit}$ มายังสถานะนี้เพื่อให้สถานะนี้กลับไปสู่สถานะไม่ทำงานเช่นเดิม ซึ่งเมื่อสถานะประมวลผลการกระทำอันสืบเนื่องจากสัญญาณ $\overline{exit}$ ที่ได้รับแล้ว จะทำการส่งสัญญาณ $\overline{exitComplete}$ กลับไปแจ้งว่าได้เปลี่ยนสถานะของสถานะเรียบร้อยแล้ว จากแนวคิดดังกล่าวเราสามารถเขียนอธิบายได้ด้วยนิพจน์ดังนี้

$$S = entry_S \cdot \overline{setActiveState}\langle entry_S \rangle \cdot \overline{entryAction} \cdot \overline{activeT_S} \cdot S'$$

$$S' = \sum_{i=1}^{n} (ie_i \cdot \overline{a_i} \cdot S') + exit \cdot \overline{exitAction} \cdot \overline{exitComplete} \cdot S$$

โดยที่

S คือ สถานะทั่วไปที่ไม่ใช่สถานะทำงานซึ่งจะไม่ตอบสนองต่อเหตุการณ์ต่างๆ ที่เกิดขึ้นภายในระบบ

S' คือ สถานะทั่วไปซึ่งเป็นสถานะทำงานและคอยตอบสนองต่อเหตุการณ์ต่างๆ ที่เกิดขึ้นภายในระบบ

$entry_S$ คือ เหตุการณ์ที่เกิดขึ้นเมื่อระบบเปลี่ยนมายังสถานะ S

$\overline{setActiveState}\langle entry_S \rangle$ คือ การส่งสัญญาณแจ้งสถานะปัจจุบันของระบบ

$\overline{entryAction}$ คือ การกระทำที่ตอบสนองเมื่อเกิดเหตุการณ์ *entry* เข้าสู่สถานะ

$\overline{activeT_S}$ คือ การส่งสัญญาณให้เส้นการเปลี่ยนแปลงที่เชื่อมต่อกับสถานะนี้ทำงาน

ie_i คือ เหตุการณ์ที่เกิดขึ้นภายในสถานะ ซึ่งไม่มีผลทำให้สถานะของระบบเปลี่ยนไป

$\overline{a_i}$ คือ การกระทำที่ตอบสนองเมื่อเกิดเหตุการณ์ภายใน ie_i

exit คือ เหตุการณ์ที่เกิดขึ้นเมื่อระบบจะเปลี่ยนสถานะออกจากสถานะปัจจุบัน

$\overline{exitAction}$ คือ การกระทำที่ตอบสนองเมื่อเกิดเหตุการณ์ *exit* ออกจากสถานะ

$\overline{exitComplete}$ คือ การส่งสัญญาณแจ้งถึงการเปลี่ยนสถานะไปสู่สถานะไม่ทำงานเรียบร้อยแล้ว

2) เส้นการเปลี่ยนแปลง (transition)

การบรรยายเส้นการเปลี่ยนแปลงนั้น ในแนวคิดของการแปลงนี้จะใช้นิพจน์กลุ่มเดียวอธิบายเส้นการเปลี่ยนแปลงทั้งหมดที่ออกจากสถานะเดียวกัน โดยไม่ได้อธิบายแบ่งแยกจากกันในแต่ละเส้น โดยกลุ่มของเส้นการเปลี่ยนแปลงจะมีสถานะทำงานและไม่ทำงานเช่นกัน โดยเริ่มแรกกลุ่มของเส้นการเปลี่ยนแปลงดังกล่าวจะอยู่ในสถานะไม่ทำงาน แต่เมื่อระบบเปลี่ยนสถานะมาสู่สถานะที่กลุ่มของเส้นการเปลี่ยนแปลงนี้เชื่อมต่ออยู่ จะมีการส่งสัญญาณ $\overline{activeT_S}$ เพื่อกระตุ้นให้เส้นการเปลี่ยนแปลงเปลี่ยนไปสู่สถานะทำงาน และรอตอบรับต่อเหตุการณ์ต่างๆ ที่เกิดขึ้น หากมีเหตุการณ์ที่เกิดขึ้นและระบบต้องเปลี่ยนสถานะไปยังสถานะอื่นแล้ว เส้นการเปลี่ยนแปลงก็จะถูกเปลี่ยนกลับไปยังสถานะไม่ทำงาน พร้อมทั้งส่งสัญญาณ $\overline{exit}$ ไปยังสถานะปัจจุบันเพื่อให้สถานะดังกล่าวเปลี่ยนกลับไปสู่สถานะไม่ทำงานเช่นกัน แต่ในกรณีที่เส้นการเปลี่ยนแปลงซึ่งมีสถานะทำงานและอยู่ในสถานะประกอบ แล้วเกิดเหตุการณ์ที่สถานะปัจจุบันต้องเปลี่ยนจากสถานะประกอบไปเป็นสถานะอื่น เส้นการเปลี่ยนแปลงจึงต้องรองรับเหตุการณ์เพื่อเปลี่ยนสถานะไปเป็นสถานะไม่ทำงานด้วยเมื่อเกิดเหตุการณ์ *inActiveT* จากแนวคิดดังกล่าวเราสามารถเขียนอธิบายได้ด้วยนิพจน์ดังนี้

$$T_S = activeT_S \cdot T_S'$$

$$T'_S = \sum_{i=1}^{n}(e_i \cdot ([c_i == True] \cdot \overline{a_i} \cdot \overline{exit} \cdot exitComplete \cdot \overline{entry_{T_{Si}}} \cdot T_S + [c_i == False] \cdot T'_S)) + inActiveT \cdot \overline{exit} \cdot exitComplete \cdot T_S$$

โดยที่

T_S คือ กลุ่มของเส้นการเปลี่ยนแปลงทั่วไปที่เชื่อมต่อกับสถานะ S

T'_S คือ กลุ่มของเส้นการเปลี่ยนที่เชื่อมต่อกับสถานะ S ซึ่งเป็นสถานะปัจจุบัน

$activeT_S$ คือ เหตุการณ์ที่ทำให้เส้นการเปลี่ยนแปลงที่เชื่อมต่อกับสถานะ S ทำงาน

e_i คือ เหตุการณ์ของแต่ละเส้นการเปลี่ยนแปลงที่เชื่อมต่อกับสถานะ S

c_i คือ เงื่อนไขของแต่ละเส้นการเปลี่ยนแปลงที่เชื่อมต่อกับสถานะ S

$\overline{a_i}$ คือ การกระทำของแต่ละเส้นการเปลี่ยนแปลงที่เชื่อมต่อกับสถานะ S

$\overline{exit}$ คือ การส่งสัญญาณไปยังสถานะปัจจุบันเพื่อแจ้งให้ทราบว่าระบบจะมีการเปลี่ยนสถานะไปยังสถานะอื่น

$exitComplete$ คือ เหตุการณ์ตอบกลับหลังจากสถานะทำงานปัจจุบันเปลี่ยนไปสู่สถานะไม่ทำงานเรียบร้อยแล้ว

$\overline{entry_{T_{Si}}}$ คือ การส่งสัญญาณไปยังสถานะปลายทางของแต่ละเส้นการเปลี่ยนแปลงเพื่อให้สถานะนั้นเป็นสถานะทำงาน

$inActiveT$ คือ เหตุการณ์เพื่อแจ้งให้เส้นการเปลี่ยนแปลงเปลี่ยนสถานะเป็นไม่ทำงาน

3) สถานะประกอบ (composite state)

รูปแบบการอธิบายของสถานะประกอบจะคล้ายกับสถานะทั่วไป แต่เมื่อสถานะประกอบเปลี่ยนเป็นสถานะทำงานแล้ว พื้นที่ภายในก็จะต้องเปลี่ยนเป็นสถานะทำงานด้วย และเมื่อเกิดเหตุการณ์ที่ระบบจะเปลี่ยนออกจากสถานะประกอบ ก็จะต้องเกิดเหตุการณ์ $\overline{activeT}$ เกิดขึ้นที่สถานะย่อยภายในด้วย ดังนิพจน์ต่อไปนี้

$$CS = entry_{CS} \cdot \overline{setActiveState}\langle entry_{CS} \rangle \cdot \overline{entryAction} \cdot \overline{activeT_{CS}} \cdot (CS' \mid R_1 \mid R_2 \mid ... \mid R_n)$$

$$CS' = \sum_{i=1}^{n}(ie_i \cdot \overline{a_i} \cdot CS') + exit \cdot \overline{exitAction} \cdot \overline{exitComplete} \cdot CS$$

$$R_i = composition\ of\ \ all\ components\ in\ region$$

โดยที่

CS คือ สถานะประกอบที่อยู่ในสถานะไม่ทำงาน

CS' คือ สถานะประกอบที่อยู่ในสถานะทำงาน

R_i คือ พื้นที่ย่อยภายในสถานะประกอบแต่ละพื้นที่ ซึ่งการอธิบายองค์ประกอบย่อยภายในพื้นที่ย่อยนั้นจะใช้หลักการเดียวกัน

$entry_{CS}$ คือ เหตุการณ์ที่เกิดขึ้นเมื่อระบบเปลี่ยนมายังสถานะประกอบนี้

$\overline{setActiveState\langle entry_S \rangle}$ คือ การส่งสัญญาณแจ้งสถานะปัจจุบันของระบบ

$\overline{entryAction}$ คือ การกระทำที่ตอบสนองเมื่อเกิดเหตุการณ์ $entry_{CS}$ เข้าสู่สถานะ

$\overline{activeT_{CS}}$ คือ การส่งสัญญาณให้เส้นการเปลี่ยนแปลงที่เชื่อมต่อกับสถานะนี้ทำงาน

ie_i คือ เหตุการณ์ที่เกิดขึ้นภายในสถานะ ซึ่งไม่มีผลทำให้สถานะของระบบเปลี่ยนไป

$\overline{a_i}$ คือ การกระทำที่ตอบสนองเมื่อเกิดเหตุการณ์ภายใน ie_i

$exit$ คือ เหตุการณ์ที่เกิดขึ้นเมื่อระบบจะเปลี่ยนสถานะออกจากสถานะปัจจุบัน

$\overline{exit_{R_i}}$ คือ การส่งสัญญาณไปยังสถานะปัจจุบันในพื้นที่ย่อยต่างๆ ภายในสถานะประกอบเพื่อแจ้งให้ทราบว่าระบบจะมีการเปลี่ยนแปลงสถานะไปยังสถานะอื่น

$exitComplete_{R_i}$ คือ เหตุการณ์ตอบกลับหลังจากสถานะทำงานปัจจุบันในแต่ละพื้นที่ย่อยเปลี่ยนไปสู่สถานะไม่ทำงานเรียบร้อยแล้ว

$\overline{exitAction}$ คือ การกระทำที่ตอบสนองเมื่อเกิดเหตุการณ์ $exit$ ออกจากสถานะ

$\overline{exitComplete}$ คือ การส่งสัญญาณแจ้งถึงการเปลี่ยนสถานะไปสู่สถานะไม่ทำงานเรียบร้อยแล้ว

4) สถานะซิง (synch state)

เมื่อมีการเปลี่ยนแปลงสถานะเข้าสู่สถานะซิงแล้วจำนวนโทเคนในสถานะจะเพิ่มขึ้น และเมื่อมีการใช้งานโทเคนในสถานะซิงแล้วจำนวนโทเคนในสถานะก็จะลดลงเช่นกัน ในลักษณะการบรรยายนั้นจะใช้การแตกกระบวนการออกเพื่อรอรับการเพิ่มโทเคนต่อไป ซึ่งตัวแปร i ใน SS_i จะเป็นตัวบอกถึงจำนวนโทเคนที่อยู่ในปัจจุบัน ในกรณีที่ค่าจำนวนโทเคนมากที่สุดที่จัดเก็บได้ถูกระบุไว้ในค่า k แล้ว เมื่อมีเหตุการณ์เปลี่ยนมายังสถานะซิงเพื่อเพิ่มโทเคนแล้ว จำนวนโทเคนจะไม่คงที่และไม่เพิ่มขึ้น จากแนวคิดดังกล่าวเราสามารถบรรยายสถานะซิงได้ด้วยนิพจน์ดังนี้

$$SS = entry_{SS} \cdot (\overline{activeT_{SS}} \cdot SS' \mid SS_1) + decToken \cdot SS$$

$$SS_i = entry_{SS} \cdot (\overline{activeT_{SS}} \cdot SS' \mid SS_{i+1}) + decToken \cdot SS_{i-1}$$

$$SS_k = entry_{SS} \cdot SS_k + decToken \cdot SS_{k-1}$$

$$SS' = exit \cdot \overline{exitComplete} \mid \overline{decToken}$$

โดยที่

SS คือ สถานะซิงที่ไม่ใช่สถานะทำงาน

SS' คือ สถานซิงที่อยู่ในสถานะทำงาน

$entry_{SS}$ คือ เหตุการณ์ที่เกิดขึ้นเมื่อระบบเปลี่ยนมายังสถานะซิง SS

$\overline{activeT_{SS}}$ คือ การส่งสัญญาณให้เส้นการเปลี่ยนแปลงที่เชื่อมต่อกับสถานะซิงนี้ทำงาน

$decToken$ คือ เหตุการณ์ที่เกิดขึ้นเมื่อระบบมีการใช้งานโทเคน

$exit$ คือ เหตุการณ์ที่เกิดขึ้นเมื่อระบบจะเปลี่ยนสถานะออกจากสถานะปัจจุบัน

$\overline{exitComplete}$ คือ การส่งสัญญาณแจ้งถึงการเปลี่ยนสถานะไปสู่สถานะไม่ทำงานเรียบร้อยแล้ว

$\overline{decToken}$ คือ การส่งสัญญาณใช้งานโทเคน

5) สถานะเริ่มต้นและสิ้นสุด (initial & final pseudostate)

การอธิบายพฤติกรรมของสถานะเริ่มต้นจะคล้ายคลึงกับสถานะทั่วไป เพียงแต่สถานะเริ่มต้นจะไม่มีการกระทำตอบสนองเมื่อเข้าสู่สถานะและออกจากสถานะ และจะรอรับสัญญาณแจ้งสถานะปัจจุบันของระบบ $\overline{setActiveState}\langle entry_S \rangle$ แต่จะไม่บันทึกสถานะนั้นไว้ และในส่วนของสถานะสิ้นสุดจะไม่มีสถานะทำงานหรือไม่ทำงาน และไม่มีเส้นการเปลี่ยนแปลงออกจากสถานะ จากแนวคิดดังกล่าวสามารถอธิบายพฤติกรรมได้ด้วยนิพจน์ดังนี้

$$I = entry_I \cdot \overline{activeT_I} \cdot I' + setActiveState(entry_S) \cdot I$$

$$I' = exit \cdot \overline{exitComplete} \cdot I$$

$$F = entry_F \cdot F$$

โดยที่

I คือ สถานะเริ่มต้นที่ไม่ทำงาน

I' คือ สถานะเริ่มต้นที่ทำงาน

$entry_I$ คือ เหตุการณ์ที่เกิดขึ้นเมื่อระบบเปลี่ยนมายังสถานะเริ่มต้น

$\overline{activeT_I}$ คือ การส่งสัญญาณให้เส้นการเปลี่ยนแปลงที่เชื่อมต่อกับสถานะนี้ทำงาน

$setActiveState(entry_S)$ คือ เหตุการณ์ที่เกิดขึ้นเพื่อแจ้งสถานะปัจจุบันของระบบ

$exit$ คือ เหตุการณ์ที่เกิดขึ้นเมื่อระบบจะเปลี่ยนสถานะออกจากสถานะปัจจุบัน

$\overline{exitComplete}$ คือ การส่งสัญญาณแจ้งถึงการเปลี่ยนสถานะไปสู่สถานะไม่ทำงานเรียบร้อยแล้ว

F คือ สถานะสิ้นสุด

$entry_F$ คือ เหตุการณ์ที่เกิดขึ้นเมื่อระบบเปลี่ยนมายังสถานะสิ้นสุด

6) สถานะประวัติ (history pseudostate)

จากรูปที่ 3.1 เราสามารถอธิบายด้วยนิพจน์ ดังนี้

$$H = entry_I \cdot \overline{activeT_H} \cdot H' + setActiveState(entry_S) \cdot Hx(entry_S)$$

$$Hx(entry_S) = entry_I \cdot \overline{entry_S} \cdot H + setActiveState(entry_{NS}) \cdot Hx(entry_{NS})$$

$$H' = exit \cdot \overline{exitComplete} \cdot H$$

โดยที่

H คือ สถานะประวัติที่ยังไม่ได้ถูกบันทึกสถานะปัจจุบันและอยู่ในสถานะไม่ทำงาน

H' คือ สถานะประวัติที่ยังไม่ได้ถูกบันทึกสถานะปัจจุบัน และอยู่ในสถานะทำงาน

Hx คือ สถานะประวัติที่ถูกบันทึกสถานะปัจจุบันไว้ผ่านตัวแปร $entry_S$

$entry_I$ คือ เหตุการณ์ที่เกิดขึ้นเมื่อระบบเปลี่ยนมายังสถานะประวัติ

$\overline{activeT_H}$ คือ การส่งสัญญาณให้เส้นการเปลี่ยนแปลงที่เชื่อมต่อกับสถานะประวัตินี้

$\overline{entry_S}$ คือ การส่งสัญญาณไปยังสถานะที่ถูกบันทึกไว้ให้เป็นสถานะทำงาน

$setActiveState(entry_S)$ คือ เหตุการณ์ที่เกิดขึ้นเพื่อแจ้งสถานะปัจจุบันของระบบ

$exit$ คือ เหตุการณ์ที่เกิดขึ้นเมื่อระบบจะเปลี่ยนสถานะออกจากสถานะประวัติ

$\overline{exitComplete}$ คือ การส่งสัญญาณแจ้งถึงการเปลี่ยนสถานะประวัติไปสู่สถานะไม่ทำงานเรียบร้อยแล้ว

7) สถานะแยกและเชื่อมสำหรับการทำงานพร้อมกัน (fork & join pseudostate)

ในแผนภาพสถานะยูเอ็มแอลนั้นจะมีการใช้งานสถานะแยกและสถานะเชื่อมเฉพาะเมื่อมีการใช้งานร่วมกับสถานะประกอบเท่านั้นดังรูปที่ 3.10 ซึ่งหากทำการพิจารณารูปแบบการเชื่อมต่อดังกล่าว จะเปรียบได้เสมือนการเชื่อมต่อเส้นการเปลี่ยนแปลงไปยังสถานะประกอบโดยตรงแต่จะทำการกำหนดสถานะทำงานเริ่มต้นของแต่ละพื้นที่ย่อยภายในด้วย โดยที่การทำงาน

จะไม่ได้เริ่มต้นที่สถานะเริ่มต้นของแต่ละพื้นที่ย่อยเช่นเดิม ซึ่งเราจะอธิบายสถานะแยกและเส้นการเปลี่ยนแปลงที่ออกจากสถานะแยกรวมกันด้วยนิพจน์เดียว แต่ในส่วนของสถานะเชื่อมนั้นจะแยกอธิบายเส้นการเปลี่ยนแปลงที่เข้าสู่สถานะเชื่อมกับสถานะเชื่อมออกจากกัน โดยการอธิบายเส้นการเปลี่ยนแปลงจะเป็นดังข้อ 2) ส่วนสถานะเชื่อมนั้นจะมีการรอรับสัญญาณจากเส้นการเปลี่ยนแปลงทั้งหมดก่อน จึงจะสั่งให้เส้นการเปลี่ยนแปลงที่ออกจากสถานะเชื่อมเปลี่ยนเป็นสถานะทำงาน จากแนวคิดดังกล่าวเราสามารถอธิบายได้ด้วยนิพจน์ดังนี้

$$FK = entry_{FX} \cdot \overline{entry_{CS}} \cdot (\overline{entry_{S_{R_1}}} \mid \overline{entry_{S_{R_2}}} \mid ... \mid \overline{entry_{S_{R_n}}}) \cdot FK$$

$$J = (entry_{J_1} \mid entry_{J_2} \mid ... \mid entry_{J_n}) \cdot \overline{exit} \cdot exitComplete \cdot \overline{activeT_J} \cdot J'$$

$$J' = exit \cdot \overline{exitComplete} \cdot J$$

โดยที่

FK คือ สถานะแยก

$entry_{FX}$ คือ เหตุการณ์ที่เกิดขึ้นเมื่อระบบเปลี่ยนมายังสถานะแยก

$\overline{entry_{CS}}$ คือ การส่งสัญญาณไปยังสถานะประกอบให้เปลี่ยนเป็นสถานะทำงาน

$\overline{entry_{S_{R_i}}}$ คือ การส่งสัญญาณไปยังสถานะย่อยในพื้นที่ย่อยแต่ละพื้นที่ให้เปลี่ยนเป็นสถานะทำงาน

J คือ สถานะเชื่อมที่อยู่ในสถานะไม่ทำงาน

J' คือ สถานะเชื่อมที่อยู่ในสถานะทำงาน

$entry_{J_i}$ คือ เหตุการณ์ที่เกิดขึ้นเมื่อระบบเปลี่ยนมายังสถานะเชื่อมจากแต่ละพื้นที่ย่อย

$\overline{exit}$ คือ การส่งสัญญาณไปยังสถานะประกอบเพื่อแจ้งให้ทราบว่าระบบจะมีการเปลี่ยนสถานะไปยังสถานะอื่น

$exitComplete$ คือ เหตุการณ์ตอบกลับหลังจากสถานะทำงานประกอบเปลี่ยนไปสู่สถานะไม่ทำงานเรียบร้อยแล้ว

$\overline{activeT_J}$ คือ การส่งสัญญาณให้เส้นการเปลี่ยนแปลงที่เชื่อมต่อกับสถานะเชื่อมนี้ทำงาน

$exit$ คือ เหตุการณ์ที่เกิดขึ้นเมื่อระบบจะเปลี่ยนสถานะออกจากสถานะเชื่อม

$\overline{exitComplete}$ คือ การส่งสัญญาณแจ้งถึงการเปลี่ยนสถานะของสถานะเชื่อมไปสู่สถานะไม่ทำงานเรียบร้อยแล้ว

8) สถานะตัวต่อและตัวเลือก (junction & choice point)

การอธิบายสถานะตัวต่อและตัวเลือกจะพิจารณาเหมือนกับสถานะทั่วไป ยกเว้นแต่ไม่มีเหตุการณ์และการกระทำต่างๆ ภายใน ซึ่งเราสามารถอธิบายได้ด้วยนิพจน์ดังนี้

$$CJ = entry_{CJ} \cdot \overline{activeT_{CJ}} \cdot CJ'$$

$$CJ' = exit \cdot \overline{exitComplete} \cdot CJ$$

โดยที่

CJ คือ สถานะตัวต่อหรือตัวเลือกที่อยู่ในสถานะไม่ทำงาน

CJ' คือ สถานะตัวต่อหรือตัวเลือกที่อยู่ในสถานะทำงาน

$entry_{CJ}$ คือ เหตุการณ์ที่เกิดขึ้นเมื่อระบบเปลี่ยนมายังสถานะ CJ

$\overline{activeT_{CJ}}$ คือ การส่งสัญญาณให้เส้นการเปลี่ยนแปลงที่เชื่อมต่อกับสถานะ CJ ทำงาน

$exit$ คือ เหตุการณ์ที่เกิดขึ้นเมื่อระบบจะเปลี่ยนสถานะออกจากสถานะ CJ

$\overline{exitComplete}$ คือ การส่งสัญญาณแจ้งถึงการเปลี่ยนสถานะไปสู่สถานะไม่ทำงานเรียบร้อยแล้ว

9) แผนภาพย่อย (submachine)

เนื่องจากแผนภาพย่อยมีความหมายเดียวกันกับสถานะประกอบ เพียงแต่แยกการอธิบายสถานะภายในออกไปเป็นแผนภาพใหม่เท่านั้น ดังนั้นการอธิบายจึงใช้รูปรูปแบบเดียวกับสถานะประกอบในหัวข้อ 3) แต่จะมีพื้นที่ย่อยภายในเพียงหนึ่งพื้นที่

3.3.2 การเชื่อมต่อกันระหว่างองค์ประกอบประเภทต่างๆ

จากแนวคิดในการอธิบายแผนภาพโดยมององค์ประกอบต่างๆ ภายในทำงานร่วมกันแบบพร้อมกันนั้น ทำให้ลักษณะการเชื่อมต่อแบบต่างๆ ของสถานะกับเส้นการเปลี่ยนแปลงในแบบต่างๆ ที่อยู่ในระดับเดียวกันนั้นไม่มีความแตกต่างกันในการอธิบาย โดยเราสามารถใช้ตัวดำเนินการ | เชื่อมองค์ประกอบต่างๆ เข้าด้วยกันได้

1) สถานะที่มีเส้นการเปลี่ยนแปลงออกหนึ่งเส้น

จากรูปที่ 3.13 เราสามารถอธิบายการเชื่อมต่อในลักษณะดังกล่าวได้ดังนี้

$$S_0 \mid T_1 \mid S_1 \mid T_2 \mid ... \mid S_n$$

โดยที่

S_i คือ สถานะต่างๆ ที่เชื่อมต่อกัน

T_i คือ เส้นการเปลี่ยนแปลงที่เชื่อมออกจากสถานะแต่ละสถานะ

2) สถานะที่มีเส้นการเปลี่ยนแปลงออกมากกว่าหนึ่งเส้น

จากรูปที่ 3.14 เส้นการเปลี่ยนแปลงทั้งหมดที่ออกจากสถานะเดียวกันจะถูกอธิบายด้วยนิพจน์แสดงเส้นการเปลี่ยนแปลงเดียวดังนี้

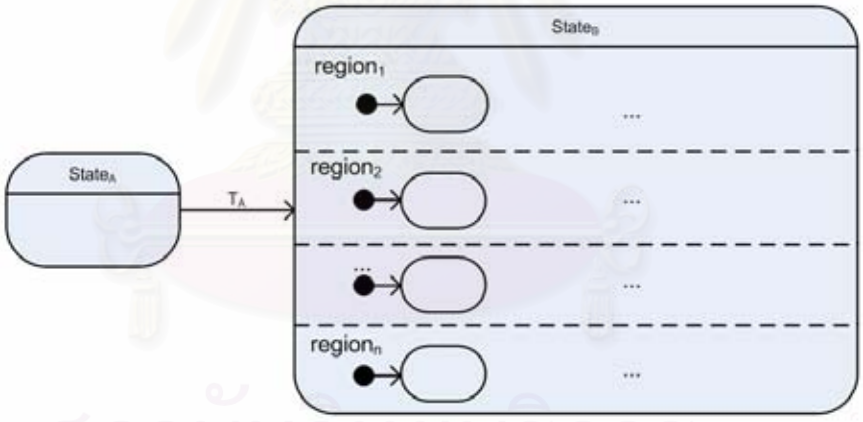
$$S \mid T_S$$

โดยที่

S คือ สถานะต้นทาง

T_S คือ เส้นการเปลี่ยนแปลงที่เชื่อมออกจากสถานะ S

3) สถานะที่มีเส้นการเปลี่ยนแปลงเชื่อมต่อไปยังสถานะประกอบ



รูปที่ 3.15 สถานะที่มีเส้นการเปลี่ยนแปลงเชื่อมต่อไปยังสถานะประกอบ

เส้นการเปลี่ยนแปลงที่ชี้ไปยังสถานะประกอบดังรูปที่ 3.15 นั้น จะพบว่ามีลักษณะคล้ายการอธิบายสถานะแยกในหัวข้อที่ผ่านมา แต่จะไม่มีเส้นการเปลี่ยนแปลงชี้ไปยังสถานะย่อยในแต่ละพื้นที่ย่อย อย่างไรก็ตามความหมายของระบบจะเปรียบเหมือนมีเส้นการเปลี่ยนแปลงชี้ไปยังสถานะเริ่มต้นแทน จากแนวคิดดังกล่าวเราสามารถอธิบายเส้นการเปลี่ยนแปลงดังกล่าวได้ดังนี้

$$T_A = activeT_A \cdot T'_A$$

$$T_A' = e \cdot ([c == True] \cdot \bar{a} \cdot \overline{exit} \cdot exitComplete \cdot \overline{entry_B} \cdot (\overline{entry_{I_{R_1}}} \mid \overline{entry_{I_{R_2}}} \mid ... \mid \overline{entry_{I_{R_n}}}) \cdot T_A + [c_i == False] \cdot T_A')$$

โดยที่

T_A คือ เส้นการเปลี่ยนแปลงที่เชื่อมต่อไปยังสถานะประกอบที่อยู่ในสถานะไม่ทำงาน

$T_A{}'$ คือ เส้นการเปลี่ยนแปลงที่เชื่อมต่อไปยังสถานะประกอบที่อยู่ในสถานะทำงาน

e คือ เหตุการณ์ของเส้นการเปลี่ยนแปลงที่เชื่อมต่อไปยังสถานะประกอบ

c คือ เงื่อนไขของเส้นการเปลี่ยนแปลงที่เชื่อมต่อไปยังสถานะประกอบ

$\bar{a}$ คือ การกระทำของเส้นการเปลี่ยนแปลงที่เชื่อมต่อไปยังสถานะประกอบ

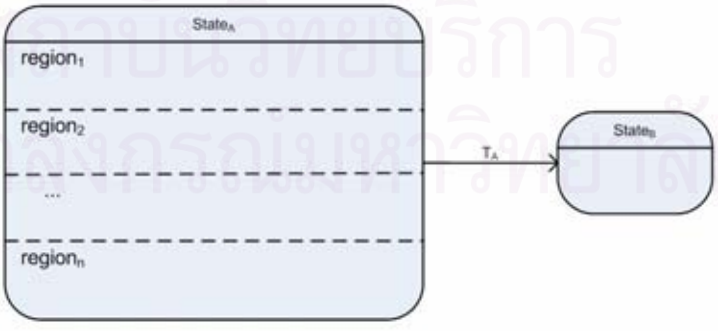
$\overline{exit}$ คือ การส่งสัญญาณไปยังสถานะปัจจุบันเพื่อแจ้งให้ทราบว่าระบบจะมีการเปลี่ยนสถานะไปยังสถานะประกอบ

$exitComplete$ คือ เหตุการณ์ตอบกลับหลังจากสถานะทำงานปัจจุบันเปลี่ยนไปสู่สถานะไม่ทำงานเรียบร้อยแล้ว

$\overline{entry_B}$ คือ การส่งสัญญาณไปยังสถานะประกอบปลายทางเพื่อให้เปลี่ยนเป็นสถานะทำงาน

$\overline{entry_{I_{R_i}}}$ คือ การส่งสัญญาณไปยังสถานะเริ่มต้นในพื้นที่ย่อยแต่ละพื้นที่ให้เปลี่ยนเป็นสถานะทำงาน

4) สถานะที่มีเส้นการเปลี่ยนแปลงเชื่อมออกจากสถานะประกอบ



รูปที่ 3.16 สถานะที่มีเส้นการเปลี่ยนแปลงออกจากสถานะประกอบ

เมื่อมีเหตุการณ์เพื่อเปลี่ยนสถานะปัจจุบันของระบบจากสถานะประกอบไปยังสถานะอื่น ดังรูปที่ 3.16 นั้น การอธิบายเส้นการเปลี่ยนแปลงจะต้องเพิ่มการส่งสัญญาณไปยังเส้นการ

เปลี่ยนแปลงภายในพื้นที่ย่อยแต่ละพื้นที่เพื่อให้เปลี่ยนสถานะเป็นไม่ทำงานด้วย จากแนวคิดดังกล่าวเราสามารถอธิบายเส้นการเปลี่ยนแปลงดังกล่าวได้ดังนี้

$$T_A = activeT_A \cdot T'_A$$

$$T'_A = e \cdot ([c == True] \cdot \bar{a} \cdot (\overline{inactiveT_{R_1}} \mid \overline{inactiveT_{R_2}} \mid ... \mid \overline{inactiveT_{R_n}}) \cdot \overline{exit} \cdot exitComplete \cdot \overline{entry_B} \cdot T_A + [c == False] \cdot T'_A)$$

โดยที่

T_A คือ เส้นการเปลี่ยนแปลงที่เชื่อมต่อกับสถานะประกอบไปยังสถานะอื่นซึ่งอยู่ในสถานะไม่ทำงาน

T'_A คือ เส้นการเปลี่ยนที่เชื่อมต่อกับสถานะประกอบซึ่งอยู่ในสถานะทำงาน

$activeT_A$ คือ เหตุการณ์ที่ทำให้เส้นการเปลี่ยนแปลงที่เชื่อมต่อกับสถานะประกอบทำงาน

e คือ เหตุการณ์ของเส้นการเปลี่ยนแปลงที่เชื่อมต่อกับสถานะประกอบ

c คือ เงื่อนไขของแต่ละเส้นการเปลี่ยนแปลงที่เชื่อมต่อกับสถานะประกอบ

$\bar{a}$ คือ การกระทำของแต่ละเส้นการเปลี่ยนแปลงที่เชื่อมต่อกับสถานะประกอบ

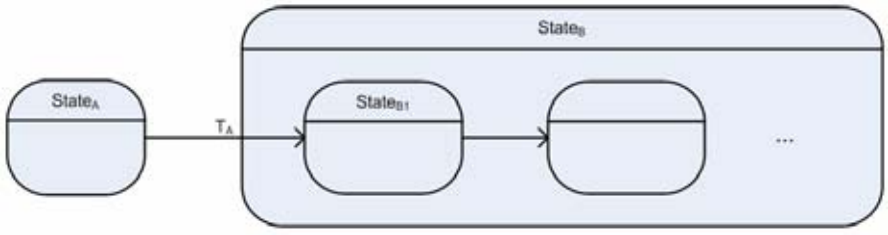
$\overline{inactiveT_{R_i}}$ คือ การส่งสัญญาณไปยังเส้นการเปลี่ยนแปลงที่ทำงานอยู่ภายในพื้นที่ย่อยแต่ละพื้นที่ให้เปลี่ยนสถานะตัวเองและสถานะที่เชื่อมต่อไปเป็นสถานะไม่ทำงาน

$\overline{exit}$ คือ การส่งสัญญาณไปยังสถานะปัจจุบันเพื่อแจ้งให้ทราบว่าระบบจะมีการเปลี่ยนสถานะไปยังสถานะอื่น

$exitComplete$ คือ เหตุการณ์ตอบกลับหลังจากสถานะทำงานปัจจุบันเปลี่ยนไปสู่สถานะไม่ทำงานเรียบร้อยแล้ว

$\overline{entry_B}$ คือ การส่งสัญญาณไปยังสถานะปลายทางของแต่ละเส้นการเปลี่ยนแปลงที่ออกจากสถานะประกอบเพื่อให้สถานะนั้นเปลี่ยนเป็นสถานะทำงาน

5) สถานะที่มีเส้นการเปลี่ยนแปลงเชื่อมต่อไปยังสถานะย่อยภายในสถานะประกอบ



รูปที่ 3.17 สถานะที่มีเส้นการเปลี่ยนแปลงเชื่อมต่อไปยังสถานะย่อยภายในสถานะประกอบ

ลักษณะของเส้นการเปลี่ยนแปลงเชื่อมต่อไปยังสถานะย่อมภายในสถานะรวมนั้น แบ่งได้สองรูปแบบ คือ แบบที่หนึ่ง สถานะรวมมีพื้นที่ย่อยภายในมากกว่าหนึ่งพื้นที่ ซึ่งรูปแบบดังกล่าวเส้นการเปลี่ยนแปลงจะต้องลากผ่านสถานะแยกดังรูปที่ 3.10 ซึ่งการอธิบายจะใช้นิพจน์ในรูปแบบที่ได้กล่าวถึงในหัวข้อสถานะแยก และรูปแบบที่สอง สถานะรวมมีพื้นที่ย่อยภายในเพียงหนึ่งพื้นที่ดังรูปที่ 3.17 ซึ่งการอธิบายจะเหมือนหัวข้อที่ 3) เพียงแต่กำหนดสถานะเริ่มต้นให้เป็นสถานะย่อยปลายทางของเส้นการเปลี่ยนแปลงแทนที่สถานะเริ่มต้น ซึ่งจากแนวคิดดังกล่าวเราสามารถอธิบายเส้นการเปลี่ยนแปลงได้ดังนี้

$$T_A = activeT_A \cdot T'_A$$

$$T'_A = e \cdot ([c == True] \cdot \bar{a} \cdot \overline{exit} \cdot exitComplete \cdot \overline{entry_B} \cdot \overline{entry_{B_1}} \cdot T_A + [c_i == False] \cdot T'_A)$$

โดยที่

T_A คือ เส้นการเปลี่ยนแปลงที่เชื่อมต่อไปยังสถานะรวมที่อยู่ในสถานะไม่ทำงาน

$T_A{}'$ คือ เส้นการเปลี่ยนแปลงที่เชื่อมต่อไปยังสถานะรวมที่อยู่ในสถานะทำงาน

e คือ เหตุการณ์ของเส้นการเปลี่ยนแปลงที่เชื่อมต่อไปยังสถานะรวม

c คือ เงื่อนไขของเส้นการเปลี่ยนแปลงที่เชื่อมต่อไปยังสถานะรวม

$\bar{a}$ คือ การกระทำของเส้นการเปลี่ยนแปลงที่เชื่อมต่อไปยังสถานะรวม

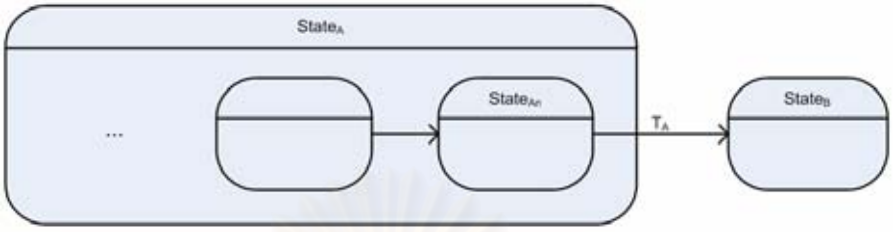
$\overline{exit}$ คือ การส่งสัญญาณไปยังสถานะปัจจุบันเพื่อแจ้งให้ทราบว่าระบบจะมีการเปลี่ยนสถานะไปยังสถานะประกอบ

$exitComplete$ คือ เหตุการณ์ตอบกลับหลังจากสถานะทำงานปัจจุบันเปลี่ยนไปสู่สถานะไม่ทำงานเรียบร้อยแล้ว

$\overline{entry_B}$ คือ การส่งสัญญาณไปยังสถานะประกอบปลายทางเพื่อให้เปลี่ยนเป็นสถานะทำงาน

$\overline{entry_{B_1}}$ คือ การส่งสัญญาณไปยังสถานะย่อยปลายทางของเส้นการเปลี่ยนแปลงในพื้นที่ย่อยที่ให้เปลี่ยนเป็นสถานะทำงาน

6) สถานะย่อยที่มีเส้นการเปลี่ยนแปลงเชื่อมโยงไปยังสถานะอื่นๆ ภายนอกสถานะประกอบที่สถานะย่อยนั้นอยู่ภายใน



รูปที่ 3.18 สถานะที่มีเส้นการเปลี่ยนแปลงเชื่อมโยงไปยังสถานะอื่นๆ ภายนอกสถานะประกอบ

ลักษณะของเส้นการเปลี่ยนแปลงเชื่อมต่อไปยังสถานะอื่นๆ ภายนอกสถานะรวมนั้น แบ่งได้สองรูปแบบ คือ แบบที่หนึ่ง สถานะย่อยนั้นอยู่ในพื้นที่ที่อยู่ในสถานะรวมที่มีพื้นที่ย่อยภายในมากกว่าหนึ่งพื้นที่ ซึ่งรูปแบบดังกล่าวเส้นการเปลี่ยนแปลงจะต้องชี้ไปยังสถานะเชื่อมดังรูปที่ 3.10 ซึ่งการอธิบายจะใช้นิพจน์ในรูปแบบที่ได้กล่าวถึงในหัวข้อสถานะเชื่อม และรูปแบบที่สอง สถานะรวมมีพื้นที่ย่อยภายในเพียงหนึ่งพื้นที่ดังรูปที่ 3.18 ซึ่งการอธิบายจะคล้ายกับหัวข้อที่ 4) คือ พิจารณาเสมือนเส้นการเปลี่ยนแปลงนั้นเชื่อมโยงออกจากสถานะประกอบเอง แต่เส้นการเปลี่ยนแปลงจะถูกเปลี่ยนให้เป็นสถานะทำงานก็ต่อเมื่อมีการเปลี่ยนสถานะของระบบมายังสถานะย่อยที่เส้นการเปลี่ยนแปลงนั้นเชื่อมโยงอยู่เท่านั้น

$$T_A = activeT_A \cdot T'_A$$

$$T'_A = e \cdot ([c == True] \cdot \bar{a} \cdot \cdot \overline{exit_A} \cdot exitComplete \cdot \overline{exit} \cdot exitComplete \cdot \overline{entry_B} \cdot T_A + [c == False] \cdot T'_A)$$

โดยที่

T_A คือ เส้นการเปลี่ยนแปลงที่เชื่อมต่อกับสถานะย่อยในสถานะประกอบไปยังสถานะอื่นซึ่งอยู่ในสถานะไม่ทำงาน

T'_A คือ เส้นการเปลี่ยนที่เชื่อมต่อกับสถานะย่อยในสถานะประกอบซึ่งอยู่ในสถานะทำงาน

$activeT_A$ คือ เหตุการณ์ที่ทำให้เส้นการเปลี่ยนแปลง T_A ทำงาน

e คือ เหตุการณ์ของเส้นการเปลี่ยนแปลง T_A

c คือ เงื่อนไขของแต่ละเส้นการเปลี่ยนแปลง T_A

$\bar{a}$ คือ การกระทำของแต่ละเส้นการเปลี่ยนแปลง T_A

$\overline{exit_A}$ คือ การส่งสัญญาณไปยังสถานะปัจจุบันในพื้นที่ย่อยของสถานะประกอบเพื่อแจ้งให้ทราบว่าระบบจะมีการเปลี่ยนสถานะไปยังสถานะอื่น

$\overline{exit}$ คือ การส่งสัญญาณไปยังสถานะปัจจุบันเพื่อแจ้งให้ทราบว่าระบบจะมีการเปลี่ยนสถานะไปยังสถานะอื่น

exitComplete คือ เหตุการณ์ตอบกลับหลังจากสถานะทำงานปัจจุบันเปลี่ยนไปสู่สถานะไม่ทำงานเรียบร้อยแล้ว

$\overline{entry_B}$ คือ การส่งสัญญาณไปยังสถานะปลายทางของแต่ละเส้นการเปลี่ยนแปลงที่ออกจากสถานะประกอบเพื่อให้สถานะนั้นเปลี่ยนเป็นสถานะทำงาน

3.4 การตรวจสอบความสัมพันธ์ระหว่างแผนภาพสถานะ

ในหัวข้อนี้การตรวจสอบความสัมพันธ์ระหว่างแผนภาพสถานะจะบรรยายในลักษณะของทฤษฎี ซึ่งสามารถนำไปประยุกต์ในการตรวจสอบแผนภาพสถานะที่ถูกแปลงไปอยู่ในรูปของซีอาร์อีหรือไพแคลคูลัสได้โดยอาศัยแนวคิดเดียวกัน ซึ่งกรณีศึกษาที่จะได้นำเสนอในบทถัดไปจะแสดงให้เห็นถึงการประยุกต์ใช้กฎการตรวจสอบในรูปแบบของไพแคลคูลัส

3.4.1 การตรวจสอบความเท่าเทียมกันในด้านพฤติกรรมของวัตถุ

ในการพัฒนาซอฟต์แวร์ในระบบขนาดใหญ่นั้น โดยส่วนมากมักมีการพัฒนาในรูปแบบของส่วนประกอบย่อยๆ หลายๆ ส่วนประกอบกัน ซึ่งในระยะยาวนั้นส่วนประกอบต่างๆ เหล่านี้อาจต้องมีการเปลี่ยนแปลงหรือแทนที่ด้วยส่วนประกอบใหม่เพื่อเพิ่มความสามารถของระบบเดิมที่มีอยู่ แต่อย่างไรก็ตามในบางครั้งระบบก็อาจต้องคงความสามารถเดิมที่มีอยู่แล้วไว้ ดังนั้นจึงต้องมีการพิจารณาถึงส่วนประกอบใหม่ที่เข้ามาแทนที่ว่าสามารถตอบสนองต่อความต้องการพื้นฐานเดิมที่มีอยู่แล้วหรือไม่ ดังนั้นจากหลักการดังกล่าวเราจะทำการตรวจสอบความครอบคลุมของพฤติกรรมการตอบสนองของส่วนประกอบใหม่เทียบกับส่วนประกอบเดิม ซึ่งส่วนประกอบใหม่นั้นต้องตอบสนองต่อสิ่งแวดล้อมจากเหตุการณ์ต่างๆ ที่เกิดขึ้นเหมือนกับส่วนประกอบเดิม

ตัวอย่าง เช่น ให้ระบบหนึ่งๆ ประกอบด้วยส่วนประกอบ A ซึ่งมีพฤติกรรมการทำงาน $\overline{a} \cdot \overline{b} \cdot \overline{c}$ และมีการเปลี่ยนแปลงระบบโดยการแทนที่ส่วนประกอบ A ด้วย ส่วนประกอบ B ซึ่งภายในประกอบด้วยส่วนประกอบ X และ Y ทำงานด้วยกัน คือ $X \mid Y$ โดยที่ $X = \overline{a} \cdot m \cdot \overline{b} \cdot n$ และ $Y = \overline{m} \cdot \overline{n} \cdot \overline{c}$ ซึ่งจะได้พฤติกรรมการทำงานร่วมกัน คือ $\overline{a} \cdot m \cdot \overline{b} \cdot n \cdot \overline{c}$ แต่ m และ n เป็นการสื่อสารกันภายในระหว่าง X และ Y ไม่ใช่การสื่อสารกับสิ่งแวดล้อมภายนอก จึงไม่จำเป็นต้องสนใจพฤติกรรมดังกล่าว ซึ่งจะทำให้ได้พฤติกรรมที่ตอบสนองต่อสิ่งแวดล้อม คือ $\overline{a} \cdot \overline{b} \cdot \overline{c}$ ซึ่งพบว่าจะเหมือนกับส่วนประกอบ A ทำให้เราสรุปได้ว่า เราสามารถแทนที่การทำงานของส่วนประกอบ A ด้วยส่วนประกอบ B

3.4.2 การตรวจสอบพฤติกรรมการทำงานของวัตถุที่เกิดขึ้นเมื่อทำงานร่วมกับวัตถุอื่นภายในระบบ

ในการออกแบบระบบด้วยแผนภาพสถานะนั้น จะมีการออกแบบการตอบสนองของวัตถุแต่ละวัตถุแยกจากกัน แต่เมื่อระบบมีการทำงานจริงวัตถุดังกล่าวจะมีการทำงานเกี่ยวเนื่องกัน ซึ่งการสื่อสารที่เกิดขึ้นจะมีทั้งระหว่างภายนอกระบบกับวัตถุภายใน และระหว่างวัตถุภายในระบบเอง ในบางครั้งในระบบที่มีขนาดใหญ่หรือมีพฤติกรรมการทำงานที่ซับซ้อน การตรวจสอบพฤติกรรมต่างๆ ที่เกิดขึ้นย่อมเป็นไปได้ยากตามกัน งานวิจัยนี้จึงนำเสนอวิธีการตรวจสอบพฤติกรรมที่เกิดขึ้นดังกล่าวในส่วนหนึ่ง คือ การตรวจสอบพฤติกรรมบางส่วนของวัตถุที่ถูกออกแบบไว้ แต่เมื่อทำงานภายในระบบกลับไม่มีพฤติกรรมนั้นๆ เกิดขึ้น ซึ่งการตรวจสอบดังกล่าวจะนำเสนอให้เห็นถึงข้อบกพร่องของการออกแบบได้ 2 รูปแบบ คือ มีการออกแบบวัตถุให้มีความสามารถเกินความจำเป็น หรือไม่ได้มีการออกแบบวัถตุที่ทำงานร่วมกันให้สนับสนุนการทำงานวัตถุนั้นๆ

ตัวอย่าง เช่น ให้ระบบหนึ่งๆ ประกอบด้วย ส่วนประกอบ X ซึ่งถูกออกแบบให้มีพฤติกรรมการทำงานเป็น $a \cdot \bar{x} + b \cdot \bar{y}$ ซึ่งจะทำงานร่วมกันกับส่วนประกอบ Y ซึ่งมีพฤติกรรมการทำงานเป็น $\bar{a} + \bar{b}$ โดยมี a และ b เป็นการสื่อสารภายในระหว่าง X และ Y ซึ่งเมื่อสองส่วนประกอบระบบทำงานร่วมกันจะได้พฤติกรรมที่สื่อสารกับสิ่งแวดล้อมเป็น $\bar{x} + \bar{y}$ แต่หากเราทำการแก้ไขระบบใหม่โดยให้ $Y = \bar{a}$ จะได้ว่าพฤติกรรมของระบบที่สื่อสารต่อสิ่งแวดล้อมจะเป็น $\bar{x}$ เท่านั้น ซึ่งเมื่อเทียบกับ X ที่ถูกออกแบบให้มีพฤติกรรมสื่อสารกับสิ่งแวดล้อมเป็น $\bar{x} + \bar{y}$ จะพบว่าพฤติกรรม $\bar{y}$ ไม่สามารถเกิดขึ้นได้ เนื่องจากการออกแบบที่ผิดพลาดของส่วนประกอบ Y

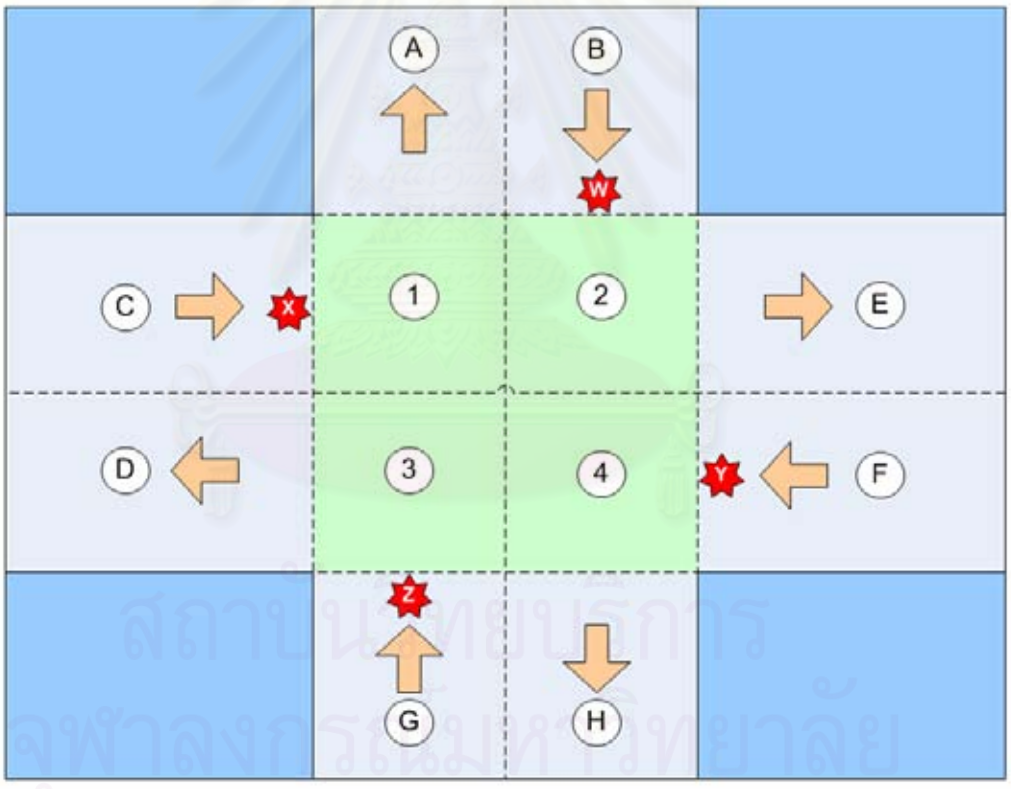
บทที่ 4

กรณีศึกษาการแปลงและตรวจสอบแผนภาพสถานะ

ระบบควบคุมสัญญาณไฟจราจร

4.1 ความเป็นมาของระบบควบคุมสัญญาณไฟจราจร

ในเมืองขนาดใหญ่ที่มีการจราจรหนาแน่น การบริหารและจัดการที่มีความสำคัญอย่างหนึ่งก็คือ การควบคุมสัญญาณไฟจราจร ซึ่งวัตถุประสงค์หลักของระบบ คือ เพื่อให้ผู้ขับขี่ยานพาหนะในแต่ละทิศทางสามารถเดินทางได้โดยปลอดภัย และการจราจรดำเนินไปด้วยความคล่องแคล่ว



รูปที่ 4.1 แผนภาพสัญญาณไฟจราจร

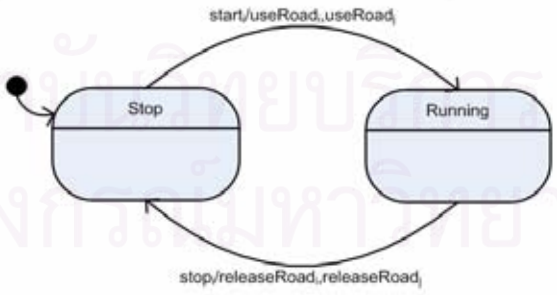
จากรูปที่ 4.1 เป็นแบบจำลองการควบคุมสัญญาณไฟจราจรซึ่งจะถูกนำมาใช้เป็นตัวอย่างในงานวิจัยนี้ โดยกำหนดให้ถนนมีถนนสองเส้นตัดกัน และแต่ละเส้นมีสองเลนเพื่อให้รถวิ่งสวนทางกันได้ ซึ่งรถแต่ละคันสามารถวิ่งได้ในทางตรงเท่านั้น ไม่สามารถเลี้ยวซ้ายหรือขวาได้

จากการวิเคราะห์ระบบ พบว่ารถที่วิ่งมาจากถนนในแต่ละทิศทางจะวิ่งผ่านพื้นที่ตัดกันตรงกลาง เช่น หากรถจากตำแหน่ง C ต้องการเดินทางไปยังตำแหน่ง E จะต้องวิ่งผ่านพื้นที่ 1 และ 2 หรือ หากรถจากตำแหน่ง G ต้องการเดินทางไปยังตำแหน่ง A จะต้องวิ่งผ่านพื้นที่ 3 และ 1 เป็นต้น จะเห็นได้ว่าการวิ่งของรถในแต่ละทิศทางจะมีการใช้งานพื้นที่ตรงกลางร่วมกัน เช่น รถที่วิ่งจาก C ไป E จะใช้พื้นที่ 1 ร่วมกับรถที่วิ่งจาก G ไป A เป็นต้น ซึ่งหากมีการปล่อยให้รถที่ใช้พื้นที่กลางร่วมกันวิ่งพร้อมกันก็จะทำให้เกิดการชนกันได้ ดังนั้นจึงต้องมีการติดตั้งสัญญาณไฟจราจรเพื่อควบคุมการวิ่งของรถในแต่ละทิศทาง เพื่อระบุช่วงเวลาที่วิ่งได้และช่วงเวลาที่ต้องหยุดรอ เช่น หากสัญญาณไฟ X เป็นเขียวแล้ว สัญญาณไฟ Z จะต้องเป็นแดง และเป็นเขียวพร้อมกันไม่ได้ เป็นต้น

หากพิจารณาระบบในอีกมุมมองหนึ่งจะพบว่าระบบมีความคล้ายคลึงกับ ระบบของผู้บริโภคและทรัพยากรที่จำกัด คือ ในระบบหนึ่งๆ จะประกอบด้วยผู้บริโภคจำนวนหนึ่งและทรัพยากรอีกจำนวนหนึ่ง ซึ่งในระยะเวลาหนึ่งๆ หากทรัพยากรถูกใช้งานโดยผู้บริโภคใดแล้ว ผู้บริโภคอื่นๆ จะไม่สามารถใช้งานทรัพยากรนั้นได้ ต้องรอให้ผู้บริโภคหยุดใช้งานทรัพยากรก่อน ซึ่งหากเปรียบเทียบกับระบบควบคุมสัญญาณไฟจราจร จะเปรียบถนนเป็นทรัพยากร และรถยนต์เป็นผู้บริโภค โดยมีสัญญาณไฟจราจรเป็นตัวสั่งงานว่าเมื่อใดรถยนต์จะใช้งานถนนได้

ในมุมมองของระบบเชิงวัตถุ จะประกอบด้วยสามวัตถุ คือ รถ ถนน และสัญญาณไฟจราจร ซึ่งเราสามารถเขียนบรรยายด้วยแผนภาพสถานะยูเอ็มแอลตามข้อกำหนดของระบบที่ได้อธิบายไว้ในเบื้องต้นได้ดังนี้

4.1.1 แผนภาพสถานะของรถ (Car)

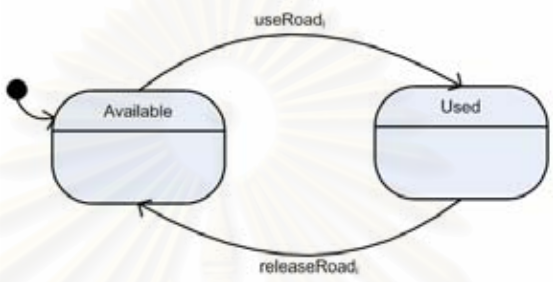


รูปที่ 4.2 แผนภาพสถานะของรถ

จากรูปที่ 4.2 เป็นแผนภาพแสดงสถานะของรถที่วิ่งในแต่ละเส้นทาง ซึ่งมีสถานะอยู่ 2 สถานะ คือ ***Stop*** สำหรับสถานะที่รถในเส้นทางนั้นหยุดวิ่ง และ ***Running*** สำหรับสถานะที่รถในเส้นทางนั้นวิ่ง ในตอนเริ่มต้นระบบนั้นรถจะอยู่ในสถานะ ***Stop*** จนกระทั่งมีสัญญาณ ***start*** จากไฟจราจรให้รถเริ่มวิ่งได้ และเมื่อรถเริ่มวิ่งก็จะส่งสัญญาณการใช้ถนน ***useRoad*** ไปยังถนนที่รถ

ต้องใช้ในการวิ่ง จากนั้นรถจะเปลี่ยนสถานะไปสู่ ***Running*** และในแนวทางเดียวกันรถจะสามารถเปลี่ยนสถานะกลับไปเป็น ***Stop*** เมื่อได้รับสัญญาณ ***stop*** จากไฟจราจร และเมื่อรถหยุดวิ่งก็จะส่งสัญญาณเลิกใช้ถนน ***releaseRoad*** ไปยังถนนที่ตัวรถใช้งานอยู่ เพื่อให้รถในเส้นทางอื่นๆ สามารถใช้งานถนนดังกล่าวได้

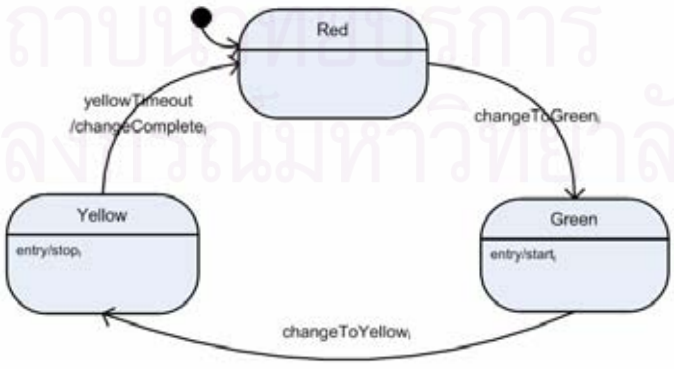
4.1.2 แผนภาพสถานะของถนน (Road)



รูปที่ 4.3 แผนภาพสถานะของถนน

จากรูปที่ 4.3 เป็นแผนภาพสถานะของถนนส่วนกลางที่เป็นทางแยก โดยมีสถานะทั้งหมด 2 สถานะ คือ ***Available*** แสดงสถานะที่ถนนว่างไม่มีรถวิ่งผ่าน และ ***Used*** แสดงสถานะที่ถนนไม่ว่างเนื่องจากมีรถวิ่งผ่าน โดยตอนเริ่มต้นระบบนั้นถนนจะอยู่ที่สถานะ ***Available*** ซึ่งถนนจะเปลี่ยนสถานะจาก ***Available*** ไปเป็น ***Used*** หากได้รับสัญญาณขอใช้ถนน ***useRoad*** จากรถ และในทำนองเดียวกันถนนสามารถเปลี่ยนสถานะจาก ***Used*** ไปเป็น ***Available*** เมื่อได้รับสัญญาณเลิกใช้ถนน ***releaseRoad*** จากตัวรถ

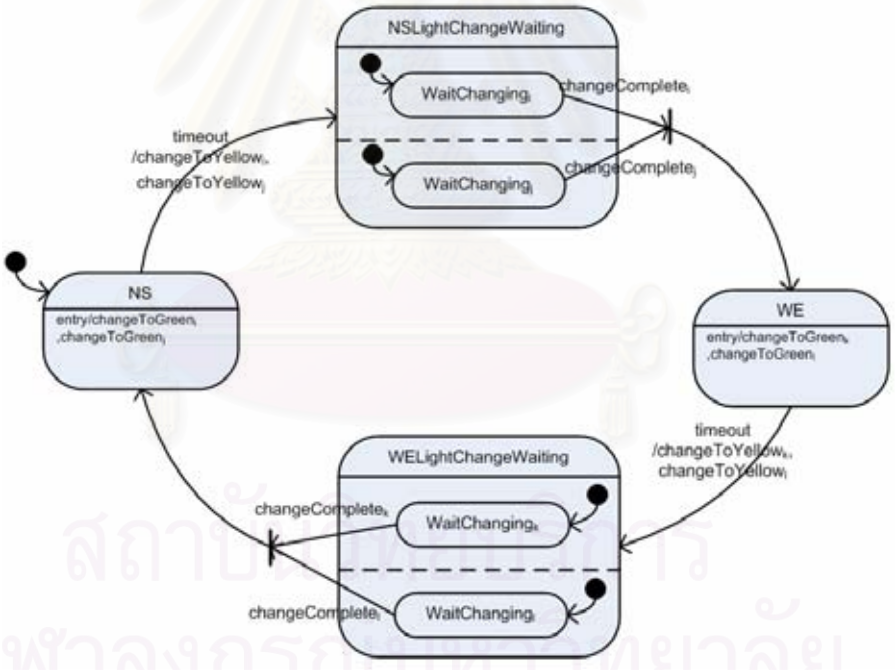
4.1.3 แผนภาพสถานะของสัญญาณไฟจราจร (Traffic Light)



รูปที่ 4.4 แผนภาพสถานะของสัญญาณไฟจราจร

จากรูปที่ 4.4 เป็นแผนภาพสถานะของสัญญาณไฟจราจร โดยสัญญาณไฟจราจรจะมีสถานะหลักสามสถานะ คือ ***Red*** สำหรับสัญญาณไฟแดง ***Green*** สำหรับสัญญาณไฟเขียว และ ***Yellow*** สำหรับสัญญาณไฟเหลือง โดยเริ่มต้นระบบสัญญาณไฟจะอยู่ที่สถานะ ***Red*** ซึ่งสัญญาณไฟจะเปลี่ยนสถานะจาก ***Red*** ไปเป็น ***Green*** เมื่อได้รับสัญญาณ ***changeToGreen*** จากตัวควบคุม และในขณะที่เปลี่ยนเป็นไฟเขียวก็จะมีการส่งสัญญาณ ***start*** ไปยังรถในเส้นทางที่สัญญาณไฟจราจรควบคุมอยู่ให้เริ่มวิ่งได้ และเมื่อระบบได้รับสัญญาณ ***changeToYellow*** จากตัวควบคุม สัญญาณไฟจะเปลี่ยนไปเป็นไฟเหลือง และส่งสัญญาณ ***stop*** ไปยังรถในเส้นทางที่สัญญาณไฟจราจรควบคุมอยู่ให้หยุดวิ่ง จากนั้นเมื่อหมดเวลาไฟเหลือง คือ ระบบได้รับสัญญาณ ***yellowTimeout*** จะทำให้สัญญาณไฟจะเปลี่ยนสถานะกลับไปเป็นแดง พร้อมทั้งส่งสัญญาณ ***changeComplete*** กลับไปแจ้งให้ตัวควบคุมทราบ

4.1.4 แผนภาพสถานะของตัวควบคุมสัญญาณไฟจราจร (Controller)

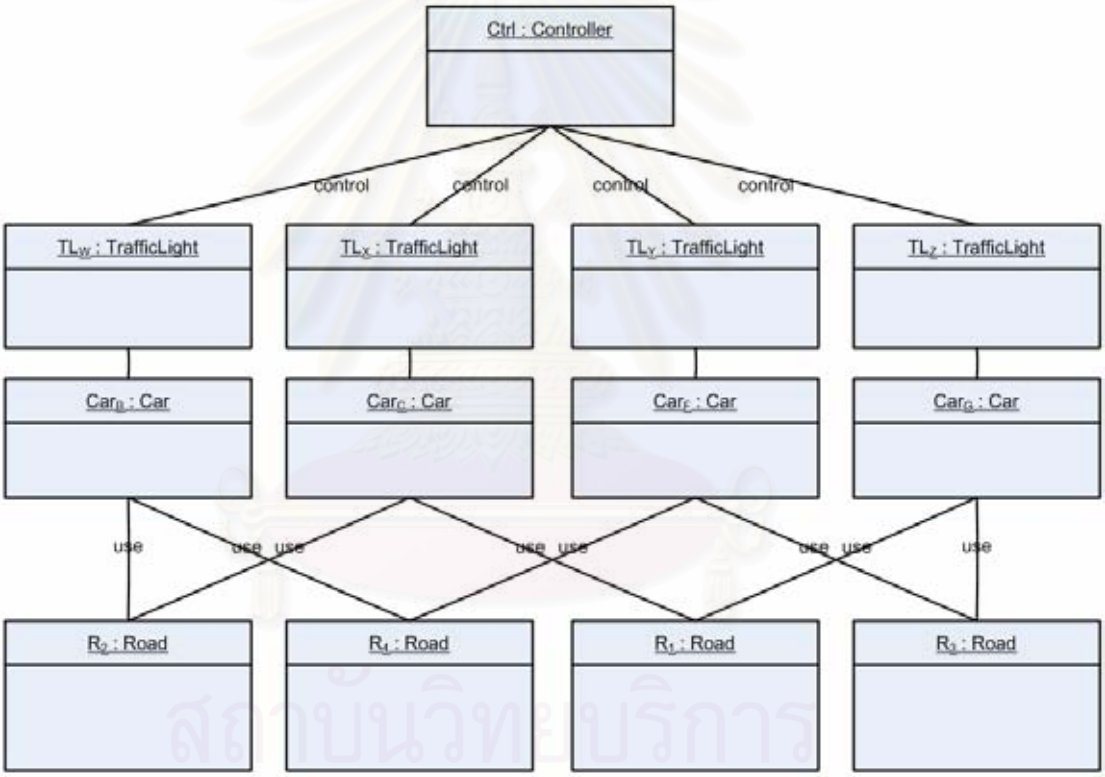


รูปที่ 4.5 แผนภาพสถานะของตัวควบคุมสัญญาณไฟจราจร

จากรูปที่ 4.5 เป็นแผนภาพสถานะของตัวควบคุมสัญญาณไฟจราจร ซึ่งในตอนเริ่มต้นระบบจะอยู่ที่สถานะ ***NS*** ซึ่งจะควบคุมสัญญาณไฟให้รถที่วิ่งในทิศทางตะวันออกไปตะวันตกหรือตะวันตกไปตะวันออกวิ่งได้เท่านั้น หากถึงเวลาที่กำหนดตัวควบคุมจะทำการส่งสัญญาณเปลี่ยนไฟ ***changeToYellow*** ไปยังรถในเส้นทางตะวันออกตะวันตก เพื่อให้เปลี่ยนสัญญาณไฟ จากนั้นจึงรอสัญญาณตอบกลับว่าเปลี่ยนแปลงเรียบร้อยแล้ว จึงเปลี่ยนสถานะตัวควบคุมไปอยู่ที่สถานะ

WE และส่งสัญญาณ *changeToGreen* ไปยังรถในเส้นทางเหนือใต้ ในแนวทางเดียวกันเมื่อถึงเวลาที่กำหนดตัวควบคุมจะทำการส่งสัญญาณเปลี่ยนไฟ *changeToYellow* ไปยังรถในเส้นทางเหนือใต้ และเปลี่ยนสถานะกลับไปเป็น *NS* พร้อมทั้งส่งสัญญาณ *changeToGreen* ไปยังรถในเส้นทางตะวันออกตะวันตก

แผนภาพสถานะทั้งหมดที่ได้นำเสนอข้างต้นแสดงพฤติกรรมต่างๆ ของวัตถุแต่ละชนิด หากพิจารณาจากรูปที่ 4.1 นั้น ระบบจะประกอบด้วยวัตถุ คือ ถนนสี่พื้นที่, รถสี่เส้นทาง, ไฟจราจรสี่สัญญาณ และตัวควบคุมสัญญาณหนึ่งตัว ทำงานร่วมกัน ซึ่งเราสามารถอธิบายความสัมพันธ์ของวัตถุแต่ละวัตถุโดยใช้แผนภาพวัตถุยูเอ็มแอลดังรูปที่ 4.6



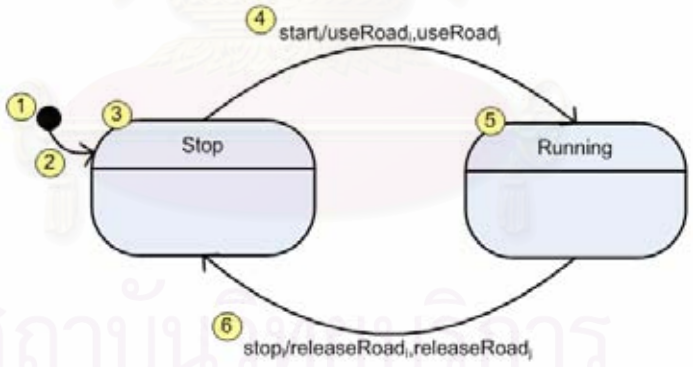
รูปที่ 4.6 แผนภาพวัตถุของระบบควบคุมสัญญาณไฟจราจร

4.2 การแปลงแผนภาพสถานะไปเป็นไพแคลคูลัส

ในหัวข้อนี้เราจะทำการแปลงแผนภาพสถานะยูเอ็มแอลของระบบควบคุมสัญญาณไฟจราจรที่ได้นำเสนอในหัวข้อ 4.1 โดยภาษาที่ได้เลือกมาเป็นภาษาปลายทางในกรณีศึกษานี้ คือ ไพแคลคูลัส เนื่องจากแผนภาพสถานะที่ถูกออกแบบมีความซับซ้อนในส่วนของเส้นการเปลี่ยนแปลงที่เชื่อมต่อกับสถานะย่อยภายในสถานะประกอบ ซึ่งในการแปลงเป็นซีอาร์อีไม่ได้ครอบคลุมการแปลงในส่วนนี้ แต่อย่างไรก็ตามหากแผนภาพสถานะถูกออกแบบและอยู่ในรูปลักษณะที่กฎการแปลงที่ได้นำเสนอในบทที่ 3 ครอบคลุมทั้งหมดก็จะสามารถแปลงและตรวจสอบความต้องกันของระบบได้ด้วยขั้นตอนเดียวกัน

ในการแปลงแผนภาพสถานะนี้เราจะตัดบางส่วนในนิพจน์ที่ไม่ต้องการทิ้งไปเนื่องจากในแผนภาพไม่ได้ระบุพฤติกรรมนั้นๆ ไว้ เช่น ในกรณีที่สถานะไม่มีการกระทำตอบสนองต่อเหตุการณ์เปลี่ยนเข้าสู่สถานะหรือออกจากสถานะก็จะตัดนิพจน์ส่วนนั้นทิ้ง หรือแผนภาพไม่ได้อยู่ภายใต้สถานะประกอบก็จะละนิพจน์ที่สนับสนุนการอธิบายพฤติกรรมของสถานะประกอบออก หรือนิพจน์ที่สนับสนุนสถานะประวัติ เป็นต้น

4.2.1 การแปลงแผนภาพสถานะของรถ



รูปที่ 4.7 การแปลงแผนภาพสถานะของรถ

จากแผนภาพสถานะของรถในรูปที่ 4.7 เราทำการแปลงโดยใช้สัญลักษณ์ Car_{ij} แทนรถที่วิ่งโดยใช้พื้นที่ส่วนกลาง i และ j ซึ่งเราสามารถแปลงองค์ประกอบย่อยต่างๆ ภายในแผนภาพได้ดังนี้

1) สถานะเริ่มต้น สามารถแปลงโดยใช้กฎข้อ 5) ในส่วนของสถานะเริ่มต้น

$$I_{Car_{ij}} = entry_{I_{Car_{ij}}} \cdot \overline{activeT_{I_{Car_{ij}}}} \cdot I'_{Car_{ij}}$$

$$I'_{Car_{ij}} = exit_{Car_{ij}} \cdot \overline{exitComplete_{Car_{ij}}} \cdot I_{Car_{ij}}$$

2) เส้นการเปลี่ยนแปลงจากสถานะเริ่มต้น สามารถแปลงโดยใช้กฎข้อ 2) ของการแปลงเส้นการเปลี่ยนแปลง

$$T_{I_{Car_{ij}}} = activeT_{I_{Car_{ij}}} \cdot T'_{I_{Car_{ij}}}$$

$$T'_{I_{Car_{ij}}} = \overline{exit_{Car_{ij}}} \cdot exitComplete_{Car_{ij}} \cdot \overline{entry_{Stop_{car_{ij}}}} \cdot T_{I_{Car_{ij}}}$$

3) สถานะ *Stop* สามารถแปลงโดยใช้กฎข้อที่ 1) ของการแปลงสถานะทั่วไป จะได้นิพจน์ดังนี้

$$Stop_{Car_{ij}} = entry_{Stop_{Car_{ij}}} \cdot \overline{activeT_{Stop_{Car_{ij}}}} \cdot Stop'_{Car_{ij}}$$

$$Stop'_{Car_{ij}} = exit_{Car_{ij}} \cdot \overline{exitComplete_{Car_{ij}}} \cdot Stop_{Car_{ij}}$$

4) เส้นการเปลี่ยนแปลงจากสถานะ *Stop* สามารถแปลงโดยใช้กฎข้อที่ 2) ของการแปลงเส้นการเปลี่ยนแปลง จะได้นิพจน์ดังนี้

$$T_{Stop_{Car_{ij}}} = activeT_{Stop_{Car_{ij}}} \cdot T'_{Stop_{Car_{ij}}}$$

$$T'_{Stop_{Car_{ij}}} = start_{Car_{ij}} \cdot \overline{useRoad_{R_i}} \cdot \overline{useRoad_{R_j}} \cdot \overline{exit_{Car_{ij}}} \cdot exitComplete_{Car_{ij}} \cdot \overline{entry_{Running_{Car_{ij}}}} \cdot T_{Stop_{Car_{ij}}}$$

5) สถานะ *Running* สามารถแปลงโดยใช้กฎข้อที่ 1) ของการแปลงสถานะทั่วไป จะได้นิพจน์ดังนี้

$$Running_{Car_{ij}} = entry_{Running_{Car_{ij}}} \cdot \overline{activeT_{Running_{Car_{ij}}}} \cdot Running'_{Car_{ij}}$$

$$Running'_{Car_{ij}} = exit_{Car_{ij}} \cdot \overline{exitComplete_{Car_{ij}}} \cdot Running_{Car_{ij}}$$

6) เส้นการเปลี่ยนแปลงจากสถานะ *Running* สามารถแปลงโดยใช้กฎข้อที่ 2) ของการแปลงเส้นการเปลี่ยนแปลง จะได้นิพจน์ดังนี้

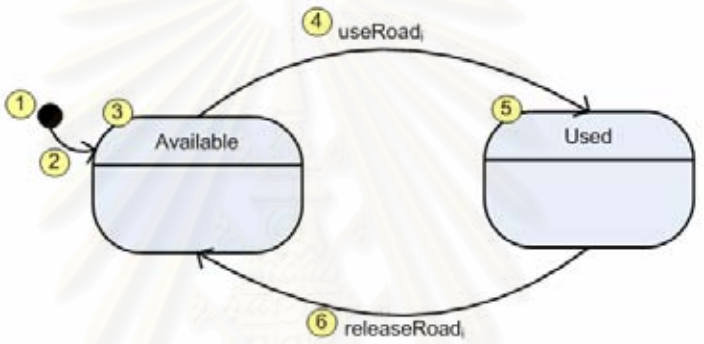
$$T_{Running_{Car_{ij}}} = activeT_{Running_{Car_{ij}}} \cdot T'_{Running_{Car_{ij}}}$$

$$T'_{Running_{Car_{ij}}} = stop_{Car_{ij}} \cdot \overline{releaseRoad_{R_i}} \cdot \overline{releaseRoad_{R_j}} \cdot \overline{exit_{Car_{ij}}} \cdot \overline{entry_{Stop_{Car_{ij}}}} \cdot T_{Running_{Car_{ij}}}$$

จากองค์ประกอบย่อยทั้งหมด เราสามารถอธิบายพฤติกรรมของรถได้ด้วยการอธิบายการทำงานร่วมกันขององค์ประกอบทั้งหมดโดยใช้ตัวดำเนินการทำงานพร้อมกัน และซ่อนการสื่อสารระหว่างองค์ประกอบต่างๆ ภายในด้วยกันเองด้วยตัวดำเนินการสร้างช่องใหม่ ได้ด้วยนิพจน์ดังนี้

$$Car_{ij} = (new\, activeT_{I_{Car_{ij}}}, exit_{Car_{ij}}, exitComplete_{Car_{ij}}, entry_{Stop_{Car_{ij}}}$$
$$, activeT_{Stop_{Car_{ij}}}, entry_{Running_{Car_{ij}}}, activeT_{Running_{Car_{ij}}})$$
$$(I_{Car_{ij}} | T_{I_{Car_{ij}}} | Stop_{Car_{ij}} | T_{Stop_{Car_{ij}}} | Running_{Car_{ij}} | T_{Running_{Car_{ij}}})$$

4.2.2 การแปลงแผนภาพสถานะของถนน



รูปที่ 4.8 การแปลงแผนภาพสถานะของถนน

จากแผนภาพสถานะของถนนในรูปที่ 4.8 เราทำการแปลงโดยใช้สัญลักษณ์ R_i แทนพื้นที่ส่วนกลาง i ซึ่งเราสามารถแปลงองค์ประกอบย่อยต่างๆ ภายในแผนภาพได้ดังนี้

1) สถานะเริ่มต้น สามารถแปลงโดยใช้กฎข้อ 5) ในส่วนของสถานะเริ่มต้น จะได้นิพจน์ดังนี้

$$I_{R_i} = entry_{I_{R_i}} \cdot \overline{activeT_{I_{R_i}}} \cdot I'_{R_i}$$
$$I'_{R_i} = exit_{R_i} \cdot \overline{exitComplete_{R_i}} \cdot I_{R_i}$$

2) เส้นการเปลี่ยนแปลงจากสถานะเริ่มต้น สามารถแปลงโดยใช้กฎข้อ 2) ของการแปลงเส้นการเปลี่ยนแปลง จะได้นิพจน์ดังนี้

$$T_{I_{R_i}} = activeT_{I_{R_i}} \cdot T'_{I_{R_i}}$$
$$T'_{I_{R_i}} = \overline{exit_{R_i}} \cdot exitComplete_{R_i} \cdot \overline{entry_{Available_{R_i}}} \cdot T_{I_{R_i}}$$

3) สถานะ *Available* สามารถแปลงโดยใช้กฎข้อที่ 1) ของการแปลงสถานะทั่วไป จะได้นิพจน์ดังนี้

$$Available_{R_i} = entry_{Available_{R_i}} \cdot \overline{activeT_{Available_{R_i}}} \cdot Available'_{R_i}$$

$$Available'_{R_i} = exit_{R_i} \cdot \overline{exitComplete_{R_i}} \cdot Available_{R_i}$$

4) เส้นการเปลี่ยนแปลงจากสถานะ *Available* สามารถแปลงโดยใช้กฎข้อที่ 2) ของการแปลงเส้นการเปลี่ยนแปลง จะได้นิพจน์ดังนี้

$$T_{Available_{R_i}} = activeT_{Available_{R_i}} \cdot T'_{Available_{R_i}}$$

$$T'_{Available_{R_i}} = useRoad_{R_i} \cdot \overline{exit_{R_i}} \cdot exitComplete_{R_i} \cdot \overline{entry_{Used_{R_i}}} \cdot T_{Available_{R_i}}$$

5) สถานะ *Used* สามารถแปลงโดยใช้กฎข้อที่ 1) ของการแปลงสถานะทั่วไป จะได้นิพจน์ดังนี้

$$Used_{R_i} = entry_{Used_{R_i}} \cdot \overline{activeT_{Used_{R_i}}} \cdot Used'_{R_i}$$

$$Used'_{R_i} = exit_{R_i} \cdot \overline{exitComplete_{R_i}} \cdot Used_{R_i}$$

6) เส้นการเปลี่ยนแปลงจากสถานะ *Used* สามารถแปลงโดยใช้กฎข้อที่ 2) ของการแปลงเส้นการเปลี่ยนแปลง จะได้นิพจน์ดังนี้

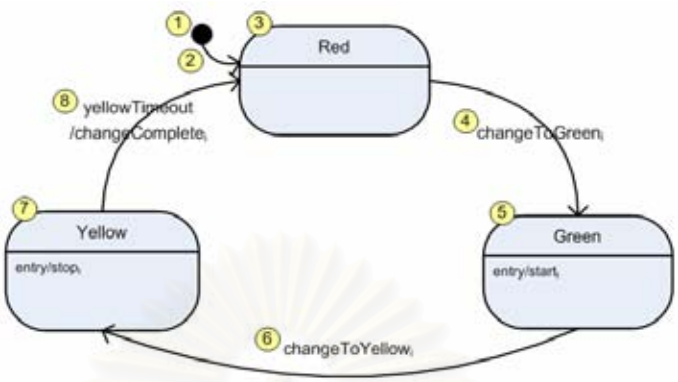
$$T_{Used_{R_i}} = activeT_{Used_{R_i}} \cdot T'_{Used_{R_i}}$$

$$T'_{Used_{R_i}} = releaseRoad_{R_i} \cdot \overline{exit_{R_i}} \cdot \overline{entry_{Available_{R_i}}} \cdot T_{Used_{R_i}}$$

จากองค์ประกอบย่อยทั้งหมด เราสามารถอธิบายพฤติกรรมของถนนได้ด้วยการอธิบายการทำงานร่วมกันขององค์ประกอบทั้งหมดโดยใช้ตัวดำเนินการทำงานพร้อมกัน และซ่อนการสื่อสารระหว่างองค์ประกอบต่างๆ ภายในด้วยกันเองด้วยตัวดำเนินการสร้างช่องใหม่ ได้ด้วยนิพจน์ดังนี้

$$\begin{aligned} R_i = (&newactiveT_{I_{R_i}}, exit_{R_i}, exitComplete_{R_i} \cdot entry_{Available_{R_i}}, activeT_{Available_{R_i}} \\ &, entry_{Used_{R_i}}, activeT_{Used_{R_i}}) \\ &(I_{R_i} \mid T_{I_{R_i}} \mid Available_{R_i} \mid T_{Available_{R_i}} \mid Used_{R_i} \mid T_{Used_{R_i}}) \end{aligned}$$

4.2.3 การแปลงแผนภาพสถานะของสัญญาณไฟจราจร



รูปที่ 4.9 การแปลงแผนภาพสถานะของสัญญาณไฟจราจร

จากแผนภาพสถานะของรถในรูปที่ 4.9 เราทำการแปลงโดยใช้สัญลักษณ์ TL_{ij} แทนไฟจราจรที่ควบคุมรถที่จะวิ่งผ่านพื้นที่ส่วนกลาง i และ j ซึ่งเราสามารถแปลงองค์ประกอบย่อยต่างๆ ภายในแผนภาพได้ดังนี้

1) สถานะเริ่มต้น สามารถแปลงโดยใช้กฎข้อ 5) ในส่วนของสถานะเริ่มต้น จะได้นิพจน์ดังนี้

$$I_{TL_{ij}} = entry_{I_{TL_{ij}}} \cdot \overline{activeT_{I_{TL_{ij}}}} \cdot I'_{TL_{ij}}$$

$$I'_{TL_{ij}} = exit_{TL_{ij}} \cdot \overline{exitComplete_{TL_{ij}}} \cdot I_{TL_{ij}}$$

2) เส้นการเปลี่ยนแปลงจากสถานะเริ่มต้น สามารถแปลงโดยใช้ชักฎข้อ 2) ของการแปลงเส้นการเปลี่ยนแปลง จะได้นิพจน์ดังนี้

$$T_{I_{TL_{ij}}} = activeT_{I_{TL_{ij}}} \cdot T'_{I_{TL_{ij}}}$$

$$T'_{I_{TL_{ij}}} = \overline{exit_{TL_{ij}}} \cdot exitComplete_{TL_{ij}} \cdot \overline{entry_{Red_{TL_{ij}}}} \cdot T_{I_{TL_{ij}}}$$

3) สถานะ Red สามารถแปลงโดยใช้กฎข้อที่ 1) ของการแปลงสถานะทั่วไป จะได้นิพจน์ดังนี้

$$Red_{TL_{ij}} = entry_{Red_{TL_{ij}}} \cdot \overline{activeT_{Red_{TL_{ij}}}} \cdot Red'_{TL_{ij}}$$

$$Red'_{TL_{ij}} = exit_{TL_{ij}} \cdot \overline{exitComplete_{TL_{ij}}} \cdot Red_{TL_{ij}}$$

4) เส้นการเปลี่ยนแปลงจากสถานะ ***Red*** สามารถแปลงโดยใช้กฎข้อที่ 2) ของการแปลงเส้นการเปลี่ยนแปลง จะได้นิพจน์ดังนี้

$$T_{Red_{TL_{ij}}} = activeT_{Red_{TL_{ij}}} \cdot T'_{Red_{TL_{ij}}}$$

$$T'_{Red_{TL_{ij}}} = changeToGreen_{TL_{ij}} \cdot \overline{exit_{TL_{ij}}} \cdot exitComplete_{TL_{ij}} \cdot \overline{entry_{Green_{TL_{ij}}}} \cdot T_{Red_{TL_{ij}}}$$

5) สถานะ ***Green*** สามารถแปลงโดยใช้กฎข้อที่ 1) ของการแปลงสถานะทั่วไป จะได้นิพจน์ดังนี้

$$Green_{TL_{ij}} = entry_{Green_{TL_{ij}}} \cdot \overline{start_{Car_{ij}}} \cdot \overline{activeT_{Green_{TL_{ij}}}} \cdot Green'_{TL_{ij}}$$

$$Green'_{TL_{ij}} = exit_{TL_{ij}} \cdot \overline{exitComplete_{TL_{ij}}} \cdot Green_{TL_{ij}}$$

6) เส้นการเปลี่ยนแปลงจากสถานะ ***Green*** สามารถแปลงโดยใช้กฎข้อที่ 2) ของการแปลงเส้นการเปลี่ยนแปลง จะได้นิพจน์ดังนี้

$$T_{Green_{TL_{ij}}} = activeT_{Green_{TL_{ij}}} \cdot T'_{Green_{TL_{ij}}}$$

$$T'_{Green_{TL_{ij}}} = changeToYellow_{TL_{ij}} \cdot \overline{exit_{TL_{ij}}} \cdot exitComplete_{TL_{ij}} \cdot \overline{entry_{Yellow_{TL_{ij}}}} \cdot T_{Green_{TL_{ij}}}$$

7) สถานะ ***Yellow*** สามารถแปลงโดยใช้กฎข้อที่ 1) ของการแปลงสถานะทั่วไป จะได้นิพจน์ดังนี้

$$Yellow_{TL_{ij}} = entry_{Yellow_{TL_{ij}}} \cdot \overline{stop_{Car_{ij}}} \cdot \overline{activeT_{Yellow_{TL_{ij}}}} \cdot Yellow'_{TL_{ij}}$$

$$Yellow'_{TL_{ij}} = exit_{TL_{ij}} \cdot Yellow_{TL_{ij}}$$

8) เส้นการเปลี่ยนแปลงจากสถานะ ***Yellow*** สามารถแปลงโดยใช้กฎข้อที่ 2) ของการแปลงเส้นการเปลี่ยนแปลง จะได้นิพจน์ดังนี้

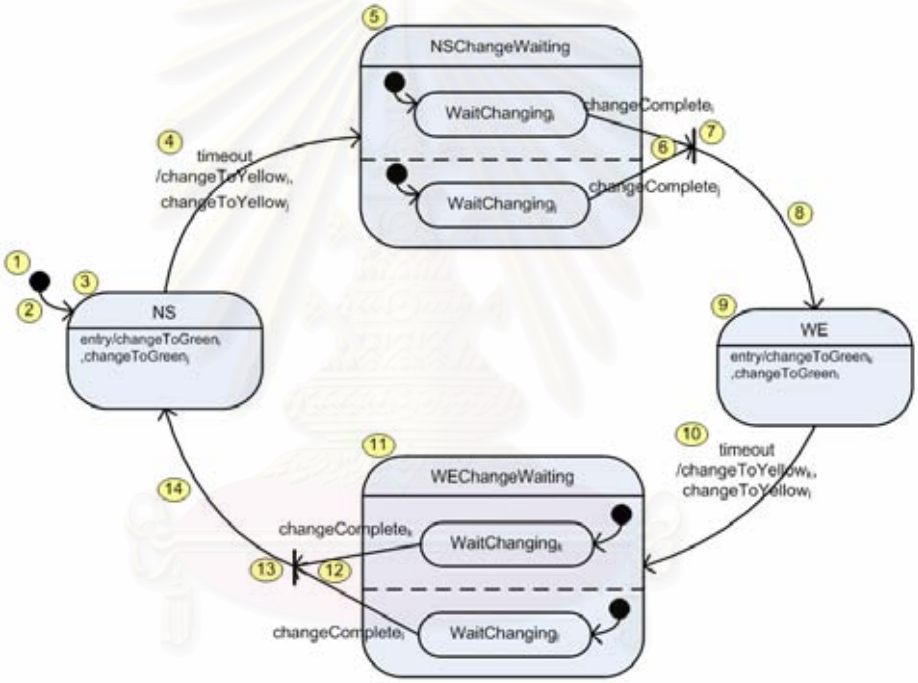
$$T_{Yellow_{TL_{ij}}} = activeT_{Yellow_{TL_{ij}}} \cdot T'_{Yellow_{TL_{ij}}}$$

$$T'_{Yellow_{TL_{ij}}} = yellowTimeout_{TL_{ij}} \cdot \overline{changComplete_{TL_{ij}}} \cdot \overline{exit_{TL_{ij}}} \cdot \overline{entry_{Red_{TL_{ij}}}} \cdot T_{Yellow_{TL_{ij}}}$$

จากองค์ประกอบย่อยทั้งหมด เราสามารถอธิบายพฤติกรรมของรถได้ด้วยการอธิบายการทำงานร่วมกันขององค์ประกอบทั้งหมดโดยใช้ตัวดำเนินการทำงานพร้อมกัน และซ่อนการสื่อสารระหว่างองค์ประกอบต่างๆ ภายในด้วยกันเองด้วยตัวดำเนินการสร้างช่องใหม่ ได้ด้วยนิพจน์ดังนี้

$$TL_{ij} = (new\ activeT_{I_{TL_{ij}}}, exit_{TL_{ij}}, exitComplete_{TL_{ij}}, entry_{Red_{TL_{ij}}} \cdot activeT_{Red_{TL_{ij}}}$$
$$entry_{Green_{TL_{ij}}}, activeT_{Green_{TL_{ij}}}, entry_{Yellow_{TL_{ij}}}, activeT_{Yellow_{TL_{ij}}})$$
$$(I_{TL_{ij}} \mid T_{I_{TL_{ij}}} \mid Red_{TL_{ij}} \mid T_{Red_{TL_{ij}}} \mid Green_{TL_{ij}} \mid T_{Green_{TL_{ij}}} \mid Yellow_{TL_{ij}} \mid T_{Yellow_{TL_{ij}}})$$

4.2.4 การแปลงแผนภาพตัวควบคุมสัญญาณไฟจราจร



รูปที่ 4.10 การแปลงแผนภาพสถานะของตัวควบคุมสัญญาณไฟจราจร

จากแผนภาพสถานะของตัวควบคุมสัญญาณไฟจราจรในรูปที่ 4.10 เราทำการแปลงโดยใช้สัญลักษณ์ $Ctrl_{ijkl}$ แทนตัวควบคุมสัญญาณไฟจราจรซึ่งควบคุมสัญญาณไฟที่ควบคุมรถที่วิ่งผ่านพื้นที่ส่วนกลาง i j k และ l ซึ่งเราสามารถแปลงองค์ประกอบย่อยต่างๆ ภายในแผนภาพได้ดังนี้

1) สถานะเริ่มต้น สามารถแปลงโดยใช้กฎข้อ 5) ในส่วนของสถานะเริ่มต้น จะได้นิพจน์ดังนี้

$$I_{Ctrl_{ijkl}} = entry_{I_{Ctrl_{ijkl}}} \cdot \overline{activeT_{I_{Ctrl_{ijkl}}}} \cdot I'_{Ctrl_{ijkl}}$$

$$I'_{Ctrl_{ijkl}} = exit_{Ctrl_{ijkl}} \cdot \overline{exitComplete_{Ctrl_{ijkl}}} \cdot I_{Ctrl_{ijkl}}$$

2) เส้นการเปลี่ยนแปลงจากสถานะเริ่มต้น สามารถแปลงโดยใช้ข้อกฎข้อ 2) ของการแปลงเส้นการเปลี่ยนแปลง จะได้นิพจน์ดังนี้

$$T_{I_{Ctrl_{ijkl}}} = activeT_{I_{Ctrl_{ijkl}}} \cdot T'_{I_{Ctrl_{ijkl}}}$$

$$T'_{I_{Ctrl_{ijkl}}} = \overline{exit_{Ctrl_{ijkl}}} \cdot exitComplete_{Ctrl_{ijkl}} \cdot \overline{entry_{NS_{Ctrl_{ijkl}}}} \cdot T_{I_{Ctrl_{ijkl}}}$$

3) สถานะ NS สามารถแปลงโดยใช้กฎข้อที่ 1) ของการแปลงสถานะทั่วไป จะได้นิพจน์ดังนี้

$$NS_{Ctrl_{ijkl}} = entry_{NS_{Ctrl_{ijkl}}} \cdot \overline{changeToGreen_{TL_{ij}}} \cdot \overline{changeToGreen_{TL_{lk}}} \cdot \overline{activeT_{NS_{Ctrl_{ijkl}}}} \cdot NS'_{Ctrl_{ijkl}}$$

$$NS'_{Ctrl_{ijkl}} = exit_{Ctrl_{ijkl}} \cdot \overline{exitComplete_{Ctrl_{ijkl}}} \cdot NS_{Ctrl_{ijkl}}$$

4) เส้นการเปลี่ยนแปลงจากสถานะ NS สามารถแปลงโดยใช้กฎข้อที่ 3) ของการแปลงรูปแบบเส้นการเปลี่ยนแปลงที่เชื่อมต่อไปยังสถานะประกอบจะได้นิพจน์ดังนี้

$$T_{NS_{Ctrl_{ijkl}}} = activeT_{NS_{Ctrl_{ijkl}}} \cdot T'_{NS_{Ctrl_{ijkl}}}$$

$$\begin{aligned} T'_{NS_{Ctrl_{ijkl}}} = {} & timeout_{Ctrl_{ijkl}} \cdot \overline{changeToYellow_{TL_{ij}}} \cdot \overline{changeToYellow_{TL_{kl}}} \\ & \cdot \overline{exit_{Ctrl_{ijkl}}} \cdot exitComplete_{Ctrl_{ijkl}} \cdot \overline{entry_{NSChangeWaiting_{Ctrl_{ijkl}}}} \\ & \cdot (\overline{entry_{I_{ij_{NS}}}} \mid \overline{entry_{I_{lk_{NS}}}}) \cdot T_{NS_{Ctrl_{ijkl}}} \end{aligned}$$

5) สถานะประกอบ $NSChangeWaiting$ สามารถแปลงโดยใช้กฎข้อที่ 3) ของการแปลงสถานะประกอบ จะได้นิพจน์ดังนี้

$$\begin{aligned} NSChangeWaiting_{Ctrl_{ijkl}} = {} & entry_{NSChangeWaiting_{Ctrl_{ijkl}}} \cdot \overline{activeT_{NSChangeWaiting_{Ctrl_{ijkl}}}} \\ & \cdot (NSChangeWaiting' \mid R1_{NS_{Ctrl_{ijkl}}} \mid R2_{NS_{Ctrl_{ijkl}}}) \end{aligned}$$

$$NSChangeWaiting'_{Ctrl_{ijkl}} = exit_{Ctrl_{ijkl}} \cdot \overline{exitComplete_{Ctrl_{ijkl}}} \cdot NSChangeWaiting_{Ctrl_{ijkl}}$$

โดยที่

$$\begin{aligned} R1_{NS_{Ctrl_{ijkl}}} = {} & (new\ activeT_{I_{ij_{NS}}}, exit_{ij_{NS}}, exitComplete_{ij_{NS}}, entry_{WaitChanging_{ij_{NS}}}) \\ & (I_{ij_{NS}} \mid T_{I_{ij_{NS}}} \mid WaitChanging_{ij_{NS}}) \end{aligned}$$

$$I_{ij_{NS}} = entry_{I_{ij_{NS}}} \cdot \overline{activeT_{I_{ij_{NS}}}} \cdot I'_{ij_{NS}}$$

$$I'_{ij_{NS}} = exit_{ij_{NS}} \cdot \overline{exitComplete_{ij_{NS}}} \cdot I_{ij_{NS}}$$
$$T_{I_{ij_{NS}}} = activeT_{I_{ij_{NS}}} \cdot T'_{I_{ij_{NS}}}$$
$$T'_{I_{ij_{NS}}} = \overline{exit_{ij_{NS}}} \cdot exitComplete_{ij_{NS}} \cdot \overline{entry_{WaitChanging_{ij_{NS}}}} \cdot T_{I_{ij_{NS}}}$$
$$WaitChanging_{ij_{NS}} = entry_{WaitChanging_{ij_{NS}}} \cdot \overline{activeT_{WaitChanging_{ij_{NS}}}} \cdot WaitChanging'_{ij_{NS}}$$
$$WaitChanging'_{ij_{NS}} = exit_{ij_{NS}} \cdot \overline{exitComplete_{ij_{NS}}} \cdot WaitChanging_{ij_{NS}}$$
$$R2_{NS_{Ctrl_{ijkl}}} = (new\ activeT_{I_{lk_{NS}}}, exit_{lk_{NS}}, exitComplete_{lk_{NS}}, entry_{WaitChanging_{lk_{NS}}})$$
$$(I_{lk_{NS}} \mid T_{I_{lk_{NS}}} \mid WaitChanging_{lk_{NS}})$$
$$I_{lk_{NS}} = entry_{I_{lk_{NS}}} \cdot \overline{activeT_{I_{lk_{NS}}}} \cdot I'_{lk_{NS}}$$
$$I'_{lk_{NS}} = exit_{lk_{NS}} \cdot \overline{exitComplete_{lk_{NS}}} \cdot I_{lk_{NS}}$$
$$T_{I_{lk_{NS}}} = activeT_{I_{lk_{NS}}} \cdot T'_{I_{lk_{NS}}}$$
$$T'_{I_{lk_{NS}}} = \overline{exit_{lk_{NS}}} \cdot exitComplete_{lk_{NS}} \cdot \overline{entry_{WaitChanging_{lk_{NS}}}} \cdot T_{I_{lk_{NS}}}$$
$$WaitChanging_{lk_{NS}} = entry_{WaitChanging_{lk_{NS}}} \cdot \overline{activeT_{WaitChanging_{lk_{NS}}}} \cdot WaitChanging'_{lk_{NS}}$$
$$WaitChanging'_{lk_{NS}} = exit_{lk_{NS}} \cdot \overline{exitComplete_{lk_{NS}}} \cdot WaitChanging_{lk_{NS}}$$

6) เส้นการเปลี่ยนแปลงจากสถานะย่อยภายในสถานะประกอบ *NSChangeWaiting* สามารถแปลงโดยใช้กฎข้อที่ 2) ของการแปลงรูปแบบเส้นการเปลี่ยนแปลงที่เชื่อมต่อไปยังสถานะประกอบจะได้นิพจน์ดังนี้

$$T_{WaitChanging_{R1_{NS}}} = activeT_{WaitChanging_{R1_{NS}}} \cdot T'_{WaitingChanging_{R1_{NS}}}$$
$$T'_{WaitingChanging_{R1_{NS}}} = changComplete_{TL_{ij}} \cdot \overline{exit_{Ctrl_{ijkl}}} \cdot exitComplete_{Ctrl_{ijkl}} \cdot \overline{entry_{J_{R1_{NS}}}} \cdot T_{NS_{Ctrl_{ijkl}}}$$
$$T_{WaitChanging_{R2_{NS}}} = activeT_{WaitChanging_{R2_{NS}}} \cdot T'_{WaitingChanging_{R2_{NS}}}$$
$$T'_{WaitingChanging_{R2_{NS}}} = changComplete_{TL_{lk}} \cdot \overline{exit_{Ctrl_{ijkl}}} \cdot exitComplete_{Ctrl_{ijkl}} \cdot \overline{entry_{J_{R2_{NS}}}} \cdot T_{NS_{Ctrl_{ijkl}}}$$

7) สถานะเชื่อมจากสถานะประกอบ *NSChangeWaiting* สามารถแปลงโดยใช้กฎข้อที่ 3) ของการแปลงสถานะทั่วไป จะได้นิพจน์ดังนี้

$$J_{NS_{Ctrl_{ijkl}}} = (entry_{J_{R1_{NS}}} \mid entry_{J_{R2_{NS}}}) \cdot \overline{exit_{Ctrl_{ijkl}}} \cdot exitComplete_{Ctrl_{ijkl}} \cdot \overline{activeT_{J_{NS_{ijkl}}}} \cdot J'_{NS_{Ctrl_{ijkl}}}$$

$$J'_{NS_{Ctrl_{ijkl}}} = exit_{Ctrl_{ijkl}} \cdot \overline{exitComplete_{Ctrl_{ijkl}}} \cdot J_{NS_{Ctrl_{ijkl}}}$$

8) เส้นการเปลี่ยนแปลงจากสถานะเชื่อม $J_{NS_{Ctrl_{ijkl}}}$ สามารถแปลงโดยใช้กฎข้อที่ 3) ของการแปลงรูปแบบเส้นการเปลี่ยนแปลงที่เชื่อมต่อไปยังสถานะประกอบจะได้นิพจน์ดังนี้

$$T_{J_{NS_{ijkl}}} = activeT_{J_{NS_{ijkl}}} \cdot T'_{J_{NS_{ijkl}}}$$

$$T'_{NS_{Ctrl_{ijkl}}} = timeout_{Ctrl_{ijkl}} \cdot \overline{changeToYellow_{TL_{ij}}} \cdot \overline{changeToYellow_{TL_{kl}}} \cdot \overline{exit_{Ctrl_{ijkl}}} \cdot exitComplete_{Ctrl_{ijkl}} \cdot \overline{entry_{WE_{Ctrl_{ijkl}}}} \cdot T_{NS_{Ctrl_{ijkl}}}$$

9) สถานะ *WE* สามารถแปลงโดยใช้กฎข้อที่ 1) ของการแปลงสถานะทั่วไปจะได้นิพจน์ดังนี้

$$WE_{Ctrl_{ijkl}} = entry_{WE_{Ctrl_{ijkl}}} \cdot \overline{changeToGreen_{TL_{ki}}} \cdot \overline{changeToGreen_{TL_{jl}}} \cdot \overline{activeT_{WE_{Ctrl_{ijkl}}}} \cdot WE'_{Ctrl_{ijkl}}$$

$$WE'_{Ctrl_{ijkl}} = exit_{Ctrl_{ijkl}} \cdot \overline{exitComplete_{Ctrl_{ijkl}}} \cdot WE_{Ctrl_{ijkl}}$$

10) เส้นการเปลี่ยนแปลงจากสถานะ *WE* สามารถแปลงโดยใช้กฎข้อที่ 3) ของการแปลงรูปแบบเส้นการเปลี่ยนแปลงที่เชื่อมต่อไปยังสถานะประกอบจะได้นิพจน์ดังนี้

$$T_{WE_{Ctrl_{ijkl}}} = activeT_{WE_{Ctrl_{ijkl}}} \cdot T'_{WE_{Ctrl_{ijkl}}}$$

$$T'_{WE_{Ctrl_{ijkl}}} = timeout_{Ctrl_{ijkl}} \cdot \overline{changeToYellow_{TL_{ki}}} \cdot \overline{changeToYellow_{TL_{jl}}} \cdot \overline{exit_{Ctrl_{ijkl}}} \cdot exitComplete_{Ctrl_{ijkl}} \cdot \overline{entry_{WEChangeWaiting_{Ctrl_{ijkl}}}} \cdot (\overline{entry_{I_{ki_{WE}}}} \mid \overline{entry_{I_{jl_{WE}}}}) \cdot T_{WE_{Ctrl_{ijkl}}}$$

11) สถานะประกอบ *WELightChangeWaiting* สามารถแปลงโดยใช้กฎข้อที่ 3) ของการแปลงสถานะประกอบ จะได้นิพจน์ดังนี้

$$WEChangeWaiting_{Ctrl_{ijkl}} = entry_{WEChangeWaiting_{Ctrl_{ijkl}}} \cdot \overline{activeT_{WEChangeWaiting_{Ctrl_{ijkl}}}} \cdot (WEChangeWaiting' \mid R1_{WE_{Ctrl_{ijkl}}} \mid R2_{WE_{Ctrl_{ijkl}}})$$

$WEChangeWaiting'_{Ctrl_{ijkl}} = exit_{Ctrl_{ijkl}} \cdot \overline{exitComplete_{Ctrl_{ijkl}}} \cdot WEChangeWaiting_{Ctrl_{ijkl}}$ โดยที่

$$R1_{WE_{Ctrl_{ijkl}}} = (new\ activeT_{I_{ki_{WE}}}, exit_{ki_{WE}}, exitComplete_{ki_{WE}}, entry_{WaitChanging_{ki_{WE}}}) (I_{ki_{WE}} \mid T_{I_{ki_{WE}}} \mid WaitChanging_{ki_{WE}})$$

$$I_{ki_{WE}} = entry_{I_{ki_{WE}}} \cdot \overline{activeT_{I_{ki_{WE}}}} \cdot I'_{ki_{WE}}$$

$$I'_{ki_{WE}} = exit_{ki_{WE}} \cdot \overline{exitComplete_{ki_{WE}}} \cdot I_{ki_{WE}}$$
$$T_{I_{ki_{WE}}} = activeT_{I_{ki_{WE}}} \cdot T'_{I_{ki_{WE}}}$$
$$T'_{I_{ki_{WE}}} = \overline{exit_{ki_{WE}}} \cdot exitComplete_{ki_{WE}} \cdot \overline{entry_{WaitChanging_{ki_{WE}}}} \cdot T_{I_{ki_{WE}}}$$
$$WaitChanging_{ki_{WE}} = entry_{WaitChanging_{ki_{WE}}} \cdot \overline{activeT_{WaitChanging_{ki_{WE}}}} \cdot WaitChanging'_{ki_{WE}}$$
$$WaitChanging'_{ki_{WE}} = exit_{ki_{WE}} \cdot \overline{exitComplete_{ki_{WE}}} \cdot WaitChanging_{ki_{WE}}$$
$$R2_{WE_{Ctrl_{ijkl}}} = (new\ activeT_{I_{jl_{WE}}}, exit_{jl_{WE}}, exitComplete_{jl_{WE}}, entry_{WaitChanging_{jl_{WE}}})$$
$$(I_{jl_{WE}} \mid T_{I_{jl_{WE}}} \mid WaitChanging_{jl_{WE}})$$
$$I_{lk_{NS}} = entry_{I_{lk_{NS}}} \cdot \overline{activeT_{I_{lk_{NS}}}} \cdot I'_{lk_{NS}}$$
$$I'_{jl_{WE}} = exit_{jl_{WE}} \cdot \overline{exitComplete_{jl_{WE}}} \cdot I_{jl_{WE}}$$
$$T_{I_{jl_{WE}}} = activeT_{I_{jl_{WE}}} \cdot T'_{I_{jl_{WE}}}$$
$$T'_{I_{jl_{WE}}} = \overline{exit_{jl_{WE}}} \cdot exitComplete_{jl_{WE}} \cdot \overline{entry_{WaitChanging_{jl_{WE}}}} \cdot T_{I_{jl_{WE}}}$$
$$WaitChanging_{jl_{WE}} = entry_{WaitChanging_{jl_{WE}}} \cdot \overline{activeT_{WaitChanging_{jl_{WE}}}} \cdot WaitChanging'_{jl_{WE}}$$
$$WaitChanging'_{jl_{WE}} = exit_{jl_{WE}} \cdot \overline{exitComplete_{jl_{WE}}} \cdot WaitChanging_{jl_{WE}}$$

12) เส้นการเปลี่ยนแปลงจากสถานะย่อยภายในสถานะประกอบ *WEChangeWaiting* สามารถแปลงโดยใช้กฎข้อที่ 2) ของการแปลงรูปแบบเส้นการเปลี่ยนแปลงที่เชื่อมต่อไปยังสถานะประกอบจะได้นิพจน์ดังนี้

$$T_{WaitChanging_{R1_{WE}}} = activeT_{WaitChanging_{R1_{WE}}} \cdot T'_{WaitingChanging_{R1_{WE}}}$$
$$T'_{WaitingChanging_{R1_{WE}}} = changComplete_{TL_{ki}} \cdot \overline{exit_{Ctrl_{ijkl}}} \cdot exitComplete_{Ctrl_{ijkl}} \cdot \overline{entry_{J_{R1_{WE}}}} \cdot T_{WE_{Ctrl_{ijkl}}}$$
$$T_{WaitChanging_{R2_{WE}}} = activeT_{WaitChanging_{R2_{WE}}} \cdot T'_{WaitingChanging_{R2_{WE}}}$$
$$T'_{WaitingChanging_{R2_{WE}}} = changComplete_{TL_{jl}} \cdot \overline{exit_{Ctrl_{ijkl}}} \cdot exitComplete_{Ctrl_{ijkl}} \cdot \overline{entry_{J_{R2_{WE}}}} \cdot T_{WE_{Ctrl_{ijkl}}}$$

13) สถานะเชื่อมจากสถานะประกอบ *WEChangeWaiting* สามารถแปลงโดยใช้กฎข้อที่ 3) ของการแปลงสถานะทั่วไป จะได้นิพจน์ดังนี้

$$J_{WE_{Ctrl_{ijkl}}} = (entry_{J_{R1_{WE}}} \mid entry_{J_{R2_{WE}}}) \cdot \overline{exit_{Ctrl_{ijkl}}} \cdot exitComplete_{Ctrl_{ijkl}} \cdot \overline{activeT_{J_{WE_{ijkl}}}} \cdot J'_{WE_{Ctrl_{ijkl}}}$$

$$J'_{WE_{Ctrl_{ijkl}}} = exit_{Ctrl_{ijkl}} \cdot \overline{exitComplete_{Ctrl_{ijkl}}} \cdot J_{WE_{Ctrl_{ijkl}}}$$

14) เส้นการเปลี่ยนแปลงจากสถานะเชื่อม $J_{WE_{Ctrl_{ijkl}}}$ สามารถแปลงโดยใช้กฎข้อที่ 3) ของการแปลงรูปแบบเส้นการเปลี่ยนแปลงที่เชื่อมต่อไปยังสถานะประกอบจะได้นิพจน์ดังนี้

$$T_{J_{WE_{ijkl}}} = activeT_{J_{WE_{ijkl}}} \cdot T'_{J_{WE_{ijkl}}}$$

$$\begin{aligned} T'_{J_{WE_{ijkl}}} = {} & timeout_{Ctrl_{ijkl}} \cdot \overline{changeToYellow_{TL_{ki}}} \cdot \overline{changeToYellow_{TL_{jl}}} \\ & \cdot \overline{exit_{Ctrl_{ijkl}}} \cdot exitComplete_{Ctrl_{ijkl}} \cdot \overline{entry_{NSLightChangeWaiting_{Ctrl_{ijkl}}}} \\ & \cdot (\overline{entry_{I_{ki_{NS}}}} \mid \overline{entry_{I_{jl_{NS}}}}) \cdot T_{J_{WE_{ijkl}}} \end{aligned}$$

จากองค์ประกอบย่อยทั้งหมด เราสามารถอธิบายพฤติกรรมของสัญญาณไฟจราจรได้ด้วยการอธิบายการทำงานร่วมกันขององค์ประกอบทั้งหมดได้ด้วยนิพจน์ดังนี้

$$\begin{aligned} Ctrl_{ijkl} = (& new\ activeT_{I_{Ctrl_{ijkl}}}, exit_{Ctrl_{ijkl}}, exitComplete_{Ctrl_{ijkl}} \\ & , entry_{NS_{Ctrl_{ijkl}}}, activeT_{NS_{Ctrl_{ijkl}}}, entry_{NSChangeWaiting_{Ctrl_{ijkl}}}, activeT_{NSChangeWaiting_{Ctrl_{ijkl}}} \\ & , activeT_{WaitChanging_{R1_{NS}}}, activeT_{WaitChanging_{R2_{NS}}}, entry_{J_{R1_{NS}}}, entry_{J_{R2_{NS}}}, activeT_{J_{NS_{ijkl}}} \\ & , entry_{WE_{Ctrl_{ijkl}}}, activeT_{WE_{Ctrl_{ijkl}}}, entry_{WEChangeWaiting_{Ctrl_{ijkl}}}, activeT_{WEChangeWaiting_{Ctrl_{ijkl}}} \\ & , activeT_{WaitChanging_{R1_{WE}}}, activeT_{WaitChanging_{R2_{WE}}}, entry_{J_{R1_{WE}}}, entry_{J_{R2_{WE}}}, activeT_{J_{WE_{ijkl}}}) \\ & (I_{Ctrl_{ijkl}} \mid T_{I_{Ctrl_{ijkl}}} \\ & \mid NS_{Ctrl_{ijkl}} \mid T_{NS_{Ctrl_{ijkl}}} \mid NSChangeWaiting_{Ctrl_{ijkl}} \\ & \mid T_{WaitChanging_{R1_{NS}}} \mid T_{WaitChanging_{R2_{NS}}} \mid J_{NS_{Ctrl_{ijkl}}} \mid T_{J_{NS_{ijkl}}} \\ & \mid WE_{Ctrl_{ijkl}} \mid T_{WE_{Ctrl_{ijkl}}} \mid WEChangeWaiting_{Ctrl_{ijkl}} \\ & \mid T_{WaitChanging_{R1_{WE}}} \mid T_{WaitChanging_{R2_{WE}}} \mid J_{WE_{Ctrl_{ijkl}}} \mid T_{J_{WE_{ijkl}}}) \end{aligned}$$

จากนิพจน์ที่ได้จากการแปลงแผนภาพย่อยของวัตถุทั้งหมดในระบบ เราสามารถอธิบายพฤติกรรมของระบบควบคุมสัญญาณไฟจราจรทั้งหมด ดังนี้

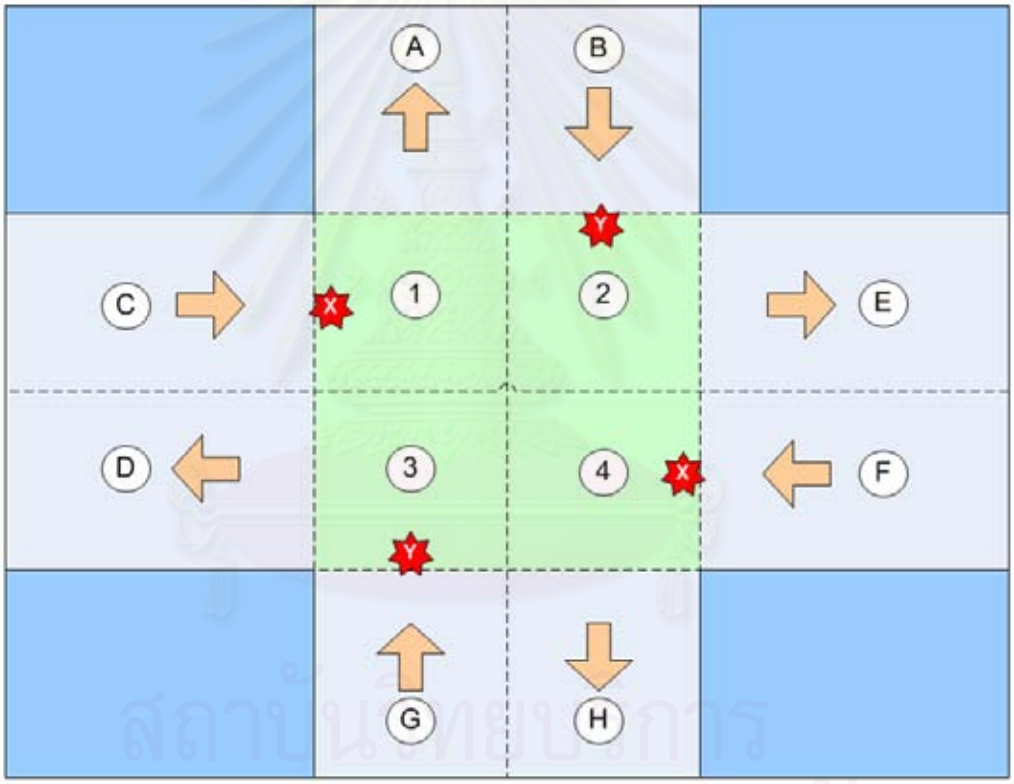
$$\begin{aligned} System_{1234} = {} & Ctrl_{1234} \mid TL_{12} \mid TL_{43} \mid TL_{31} \mid TL_{24} \\ & \mid Car_{12} \mid Car_{43} \mid Car_{31} \mid Car_{24} \mid R_1 \mid R_2 \mid R_3 \mid R_4 \end{aligned}$$

4.3 การตรวจสอบความสัมพันธ์ของวัตถุภายในระบบ

การตรวจสอบความสัมพันธ์ของวัตถุที่จะแสดงให้เห็นในระบบสัญญาณไฟจราจรแบ่งออกเป็นสองแบบ คือ

4.3.1 การตรวจสอบความเท่าเทียมกันในด้านพฤติกรรมของวัตถุต่างๆ ในระบบ

ในตัวอย่างนี้เราจะแสดงให้เห็นถึงการปรับปรุงการทำงานของระบบไฟจราจร โดยการแทนที่ส่วนประกอบบางส่วนโดยส่วนประกอบใหม่ ซึ่งผลของพฤติกรรมตอบสนองของระบบที่ได้จากการทำงานของส่วนประกอบใหม่จะต้องไม่แตกต่างไปจากระบบเดิม

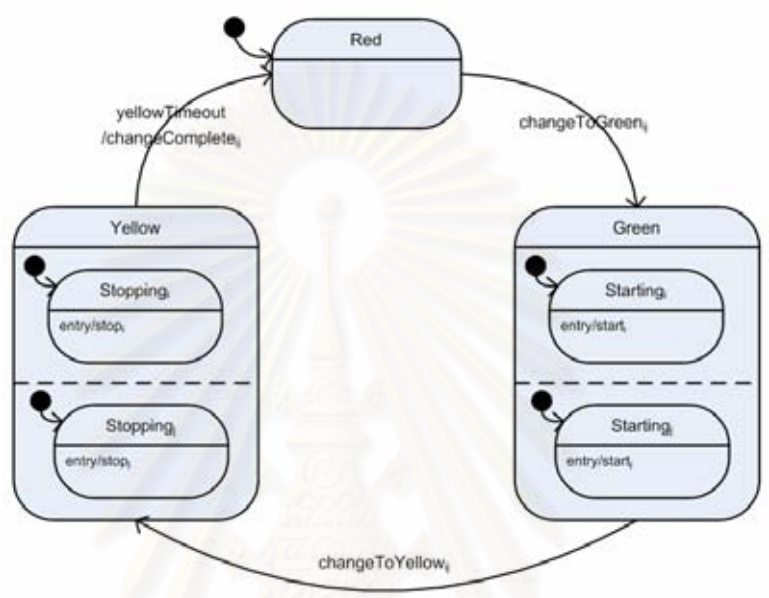


รูปที่ 4.11 แผนภาพสัญญาณไฟจราจรแบบใช้สัญญาณไฟร่วม

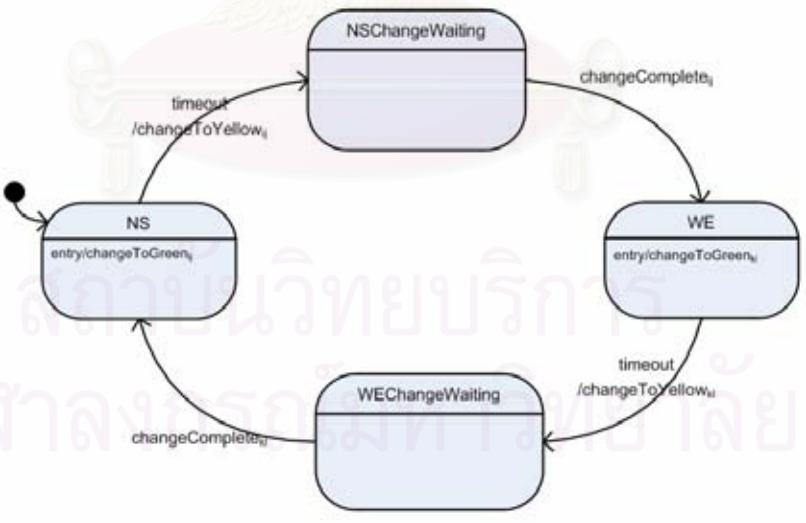
จากระบบควบคุมสัญญาณไฟจราจรที่ออกแบบไว้เบื้องต้นในรูปที่ 4.1 นั้น จะเห็นว่ารถในเส้นทางที่วิ่งตรงกันข้ามกัน เช่น C ไป E และ F ไป D เป็นต้น สามารถวิ่งได้พร้อมกันเนื่องจากไม่ได้ใช้พื้นที่ถนนส่วนกลางร่วมกัน ดังนั้นจึงเป็นไปได้ว่า เราสามารถยุบสัญญาณไฟจราจรลงเหลือเพียงสองจุดจากทั้งหมดจุด โดยสัญญาณไฟหนึ่งตัว จะเป็นตัวบอกให้รถที่วิ่งสวนทางกันวิ่งได้พร้อมกัน คือ C ไป E วิ่งได้พร้อม F ไป D และ G ไป A วิ่งได้พร้อมกับ B ไป H ดังรูปที่ 4.11 จะ

เหลือสัญญาณไฟเพียงสองตัว คือ X และ Y โดยในที่นี้ผู้วิจัยวาดภาพเป็นสี่จุด เพื่อให้สังเกตได้ง่ายว่าเป็นตัวบอกสัญญาณของรถทิศทางใด

จากระบบที่เปลี่ยนแปลงดังกล่าวเราจำเป็นจะต้องออกแบบวัตถุใหม่สองตัว คือ สัญญาณไฟจราจร และตัวควบคุมสัญญาณไฟจราจร ซึ่งจะได้ดังรูปที่ 4.12 และ 4.13



รูปที่ 4.12 แผนภาพสถานะของสัญญาณไฟจราจรแบบใหม่



รูปที่ 4.13 แผนภาพสถานะของตัวควบคุมสัญญาณไฟจราจรแบบใหม่

ซึ่งผลจากการแปลงแผนภาพสถานะของสัญญาณไฟจราจรรูปที่ 4.12 จะได้ ดังนี้

$$TL_{ijkl} = (new\ activeT_{I_{TL_{ijkl}}}, exit_{TL_{ijkl}}, exitComplete_{TL_{ijkl}}, entry_{Red_{TL_{ijkl}}} \cdot activeT_{Red_{TL_{ijkl}}}$$
$$entry_{Green_{TL_{ijkl}}}, activeT_{Green_{TL_{ijkl}}}, entry_{Yellow_{TL_{ijkl}}}, activeT_{Yellow_{TL_{ijkl}}})$$
$$(I_{TL_{ijkl}} \mid T_{I_{TL_{ijkl}}} \mid Red_{TL_{ijkl}} \mid T_{Red_{TL_{ijkl}}} \mid Green_{TL_{ijkl}} \mid T_{Green_{TL_{ijkl}}} \mid Yellow_{TL_{ijkl}} \mid T_{Yellow_{TL_{ijkl}}})$$

โดยที่องค์ประกอบย่อยภายในที่นิพจน์เปลี่ยนแปลง คือ

$$I_{TL_{ijkl}} = entry_{I_{TL_{ijkl}}} \cdot \overline{activeT_{I_{TL_{ijkl}}}} \cdot I'_{TL_{ijkl}}$$
$$I'_{TL_{ijkl}} = exit_{TL_{ijkl}} \cdot \overline{exitComplete_{TL_{ijkl}}} \cdot I_{TL_{ijkl}}$$
$$T_{I_{TL_{ijkl}}} = activeT_{I_{TL_{ijkl}}} \cdot T'_{I_{TL_{ijkl}}}$$
$$T'_{I_{TL_{ijkl}}} = \overline{exit_{TL_{ijkl}}} \cdot exitComplete_{TL_{ijkl}} \cdot \overline{entry_{Red_{TL_{ijkl}}}} \cdot T_{I_{TL_{ijkl}}}$$
$$Red_{TL_{ijkl}} = entry_{Red_{TL_{ijkl}}} \cdot \overline{activeT_{Red_{TL_{ijkl}}}} \cdot Red'_{TL_{ijkl}}$$
$$Red'_{TL_{ijkl}} = exit_{TL_{ijkl}} \cdot \overline{exitComplete_{TL_{ijkl}}} \cdot Red_{TL_{ijkl}}$$
$$T_{Red_{TL_{ijkl}}} = activeT_{Red_{TL_{ijkl}}} \cdot T'_{Red_{TL_{ijkl}}}$$
$$T'_{Red_{TL_{ijkl}}} = changeToGreen_{TL_{ijkl}} \cdot \overline{exit_{Ctrl_{ijkl}}} \cdot exitComplete_{Ctrl_{ijkl}} \cdot \overline{entry_{Green_{TL_{ijkl}}}}$$
$$\cdot (\overline{entry_{I_{iGreen}}} \mid \overline{entry_{I_{jGreen}}}) \cdot T_{Red_{TL_{ijkl}}}$$
$$Green_{TL_{ijkl}} = entry_{Green_{TL_{ijkl}}} \cdot \overline{activeT_{Green_{TL_{ijkl}}}}$$
$$\cdot (Green' \mid R1_{Green_{TL_{ijkl}}} \mid R2_{Green_{TL_{ijkl}}}) \cdot Green'_{TL_{ijkl}}$$
$$Green'_{TL_{ijkl}} = exit_{TL_{ijkl}} \cdot \overline{exitComplete_{TL_{ijkl}}} \cdot Green_{Ctrl_{ijkl}}$$
$$R1_{Green_{TL_{ijkl}}} = (new\ activeT_{I_{ij_{Green}}}, exit_{ij_{Green}}, exitComplete_{ij_{Green}}, entry_{Starting_{ij_{Green}}})$$
$$(I_{ij_{Green}} \mid T_{I_{ij_{Green}}} \mid Starting_{ij_{Green}})$$
$$Starting_{ij_{Green}} = entry_{Starting_{ij_{Green}}} \cdot \overline{start_{Car_{ij}}} \cdot Starting'_{ij_{Green}}$$
$$Starting'_{ij_{Green}} = exit_{ij_{Green}} \cdot \overline{exitComplete_{ij_{Green}}} \cdot Starting_{ij_{Green}}$$
$$+ inActiveT_{R1_{Green}} \cdot Starting_{ij_{Green}}$$
$$R2_{Green_{TL_{ijkl}}} = (new\ activeT_{I_{lk_{Green}}}, exit_{lk_{Green}}, exitComplete_{lk_{Green}}, entry_{Starting_{lk_{Green}}})$$
$$(I_{lk_{Green}} \mid T_{I_{lk_{Green}}} \mid Starting_{lk_{Green}})$$
$$Starting_{lk_{Green}} = entry_{Starting_{lk_{Green}}} \cdot \overline{start_{Car_{lk}}} \cdot Starting'_{lk_{Green}}$$
$$Starting'_{lk_{Green}} = exit_{lk_{Green}} \cdot \overline{exitComplete_{lk_{Green}}} \cdot Starting_{lk_{Green}}$$
$$+ inActiveT_{R2_{Green}} \cdot Starting_{lk_{Green}}$$
$$T_{Green_{TL_{ijkl}}} = activeT_{Green_{TL_{ijkl}}} \cdot T'_{Green_{TL_{ijkl}}}$$
$$T'_{Green_{TL_{ijkl}}} = changeToYellow_{TL_{ijkl}} \cdot (\overline{inactiveT_{R1_{Green}}} \mid \overline{inactiveT_{R1_{Green}}}) \cdot \overline{exit_{TL_{ijkl}}}$$
$$\cdot exitComplete_{TL_{ijkl}} \cdot \overline{entry_{Yellow_{TL_{ijkl}}}} \cdot (\overline{entry_{I_{iYellow}}} \mid \overline{entry_{I_{jYellow}}}) \cdot T_{Green_{TL_{ijkl}}}$$

$$Yellow_{TL_{ijkl}} = entry_{Yellow_{TL_{ijkl}}} \cdot \overline{activeT_{Yellow_{TL_{ijkl}}}} \cdot (Yellow' \mid R1_{Yellow_{TL_{ijkl}}} \mid R2_{Yellow_{TL_{ijkl}}}) \cdot Yellow'_{TL_{ijkl}}$$

$$Yellow'_{TL_{ijkl}} = exit_{TL_{ijkl}} \cdot \overline{exitComplete_{TL_{ijkl}}} \cdot Yellow_{TL_{ijkl}}$$

$$R1_{Yellow_{TL_{ijkl}}} = (new\ activeT_{I_{i_{Yellow}}}, exit_{i_{Yellow}}, exitComplete_{i_{Yellow}}, entry_{Stopping_{i_{Yellow}}}) (I_{i_{Yellow}} \mid T_{I_{i_{Yellow}}} \mid Stopping_{i_{Yellow}})$$

$$Stopping_{ij_{Yellow}} = entry_{Stopping_{ij_{Yellow}}} \cdot \overline{stop_{Car_{ij}}} \cdot Stopping'_{ij_{Yellow}}$$

$$Stopping'_{ij_{Yellow}} = exit_{ij_{Yellow}} \cdot \overline{exitComplete_{ij_{Yellow}}} \cdot Starting_{ij_{Yellow}} + inActiveT_{R1_{Yellow}} \cdot Stopping_{ij_{Yellow}}$$

$$R2_{Yellow_{TL_{ijkl}}} = (new\ activeT_{I_{lk_{Yellow}}}, exit_{lk_{Yellow}}, exitComplete_{lk_{Yellow}}, entry_{Starting_{lk_{Yellow}}}) (I_{lk_{Yellow}} \mid T_{I_{lk_{Yellow}}} \mid Stopping_{lk_{Yellow}})$$

$$Stopping_{lk_{Yellow}} = entry_{Stopping_{lk_{Yellow}}} \cdot \overline{stop_{Car_{lk}}} \cdot Stopping'_{lk_{Yellow}}$$

$$Stopping'_{lk_{Yellow}} = exit_{lk_{Yellow}} \cdot \overline{exitComplete_{lk_{Yellow}}} \cdot Starting_{lk_{Yellow}} + inActiveT_{R2_{Yellow}} \cdot Stopping_{lk_{Yellow}}$$

$$T_{Yellow_{TL_{ijkl}}} = activeT_{Yellow_{TL_{ijkl}}} \cdot T'_{Yellow_{TL_{ijkl}}}$$

$$T'_{Yellow_{TL_{ijkl}}} = yellowTimeout_{TL_{ijkl}} \cdot \overline{changeComplete_{TL_{ijkl}}} \cdot (\overline{inactiveT_{R1_{Yellow}}} \mid \overline{inactiveT_{R1_{Yellow}}}) \cdot \overline{exit_{TL_{ijkl}}} \cdot exitComplete_{TL_{ijkl}} \cdot \overline{entry_{Red_{TL_{ijkl}}}} \cdot T_{Yellow_{TL_{ijkl}}}$$

$$T'_{Green_{TL_{ijkl}}} = changeToYellow_{TL_{ijkl}} \cdot (\overline{inactiveT_{R1_{Green}}} \mid \overline{inactiveT_{R1_{Green}}}) \cdot \overline{exit_{TL_{ijkl}}} \cdot exitComplete_{TL_{ijkl}} \cdot \overline{entry_{Yellow_{TL_{ijkl}}}} \cdot (\overline{entry_{I_{i_{Yellow}}}} \mid \overline{entry_{I_{j_{Yellow}}}}) \cdot T_{Green_{TL_{ijkl}}}$$

และผลจากการแปลงแผนภาพสถานะของตัวควบคุมจากรูปที่ 4.9 จะได้ ดังนี้

$$Ctrl_{ijkl} = (new\ activeT_{I_{Ctrl_{ijkl}}}, exit_{Ctrl_{ijkl}}, exitComplete_{Ctrl_{ijkl}}, entry_{NS_{Ctrl_{ijkl}}}, activeT_{NS_{Ctrl_{ijkl}}}, entry_{WE_{Ctrl_{ijkl}}}, activeT_{WE_{Ctrl_{ijkl}}}) (I_{Ctrl_{ijkl}} \mid T_{I_{Ctrl_{ijkl}}} \mid NS_{Ctrl_{ijkl}} \mid T_{NS_{Ctrl_{ijkl}}} \mid NSChangeWaiting_{Ctrl_{ijkl}} \mid T_{NSChangeWaiting_{Ctrl_{ijkl}}} \mid WE_{Ctrl_{ijkl}} \mid T_{WE_{Ctrl_{ijkl}}} \mid WEChangeWaiting_{Ctrl_{ijkl}} \mid T_{WEChangeWaiting_{Ctrl_{ijkl}}})$$

โดยที่องค์ประกอบย่อยภายในที่นิพจน์เปลี่ยนแปลง คือ

$$NS_{Ctrl_{ijkl}} = entry_{NS_{Ctrl_{ijkl}}} \cdot \overline{changeToGreen_{TL_{ijkl}}} \cdot \overline{activeT_{NS_{Ctrl_{ijkl}}}} \cdot NS'_{Ctrl_{ijkl}}$$

$$T'_{NS_{Ctrl_{ijkl}}} = timeout_{Ctrl_{ijkl}} \cdot \overline{changeToYellow_{TL_{ijkl}}} \cdot \overline{exit_{Ctrl_{ijkl}}} \cdot exitComplete_{Ctrl_{ijkl}} \cdot \overline{entry_{NSChangeWaiting_{Ctrl_{ijkl}}}} \cdot T_{NS_{Ctrl_{ijkl}}}$$

$$NSChangeWaiting_{Ctrl_{ijkl}} = entry_{NSChangeWaiting_{Ctrl_{ijkl}}} \cdot \overline{activeT_{NSChangeWaiting_{Ctrl_{ijkl}}}} \cdot NSChangeWaiting'_{Ctrl_{ijkl}}$$

$$T'_{NSChangeWaiting_{Ctrl_{ijkl}}} = changeComplete_{Ctrl_{ijkl}} \cdot \overline{exit_{Ctrl_{ijkl}}} \cdot exitComplete_{Ctrl_{ijkl}} \cdot \overline{entry_{WE_{Ctrl_{ijkl}}}} \cdot T_{NSChangeWaiting_{Ctrl_{ijkl}}}$$

$$WE_{Ctrl_{ijkl}} = entry_{WE_{Ctrl_{ijkl}}} \cdot \overline{changeToGreen_{TL_{jlik}}} \cdot \overline{activeT_{WE_{Ctrl_{ijkl}}}} \cdot WE'_{Ctrl_{ijkl}}$$

$$T'_{WE_{Ctrl_{ijkl}}} = timeout_{Ctrl_{ijkl}} \cdot \overline{changeToYellow_{TL_{jlik}}} \cdot \overline{exit_{Ctrl_{ijkl}}} \cdot exitComplete_{Ctrl_{ijkl}} \cdot \overline{entry_{WEChangeWaiting_{Ctrl_{ijkl}}}} \cdot T_{WE_{Ctrl_{ijkl}}}$$

$$WEChangeWaiting_{Ctrl_{ijkl}} = entry_{WEChangeWaiting_{Ctrl_{ijkl}}} \cdot \overline{activeT_{WEChangeWaiting_{Ctrl_{ijkl}}}} \cdot WEChangeWaiting'_{Ctrl_{ijkl}}$$

$$T'_{WEChangeWaiting_{Ctrl_{ijkl}}} = changeComplete_{Ctrl_{jlik}} \cdot \overline{exit_{Ctrl_{ijkl}}} \cdot exitComplete_{Ctrl_{ijkl}} \cdot \overline{entry_{WE_{Ctrl_{ijkl}}}} \cdot T_{WEChangeWaiting_{Ctrl_{ijkl}}}$$

เราสามารถอธิบายพฤติกรรมของระบบควบคุมสัญญาณไฟจราจรทั้งหมดที่ถูกแก้ไขโดยยุบสัญญาณไฟจราจรเหลือเพียงสองสัญญาณ ดังนี้

$$System_{1234} = Ctrl_{1234} \,|\, TL_{1243} \,|\, TL_{3124} \,|\, Car_{12} \,|\, Car_{43} \,|\, Car_{31} \,|\, Car_{24} \,|\, R_1 \,|\, R_2 \,|\, R_3 \,|\, R_4$$

การตรวจสอบความเท่าเทียมกันของพฤติกรรมจะพิจารณาจากองค์ประกอบของระบบที่มีการเปลี่ยนแปลง ซึ่งในที่นี้เมื่อพิจารณาจากระบบควบคุมสัญญาณไฟจราจรเดิมที่ได้ถูกออกแบบไว้เปรียบเทียบกับระบบที่ได้รับการปรับปรุง พบว่าองค์ประกอบที่ได้รับการเปลี่ยนแปลง คือ สัญญาณไฟจราจร และตัวควบคุมสัญญาณไฟจราจร คือ $Ctrl_{1234} \,|\, TL_{12} \,|\, TL_{43} \,|\, TL_{31} \,|\, TL_{24}$ ในระบบเดิม เปลี่ยนเป็น $Ctrl_{1234} \,|\, TL_{1243} \,|\, TL_{3124}$ ในระบบใหม่

ในการพิจารณาความเท่าเทียมกันของระบบนั้น เราจะพิจารณาพฤติกรรมที่แต่ละระบบตอบสนองต่อสิ่งแวดล้อมภายนอกเท่านั้น ดังนั้นจึงต้องทำการซ่อนพฤติกรรมที่มีการสื่อสารภายในระหว่างสัญญาณไฟจราจรกับตัวควบคุมสัญญาณไฟจราจร ดังนิพจน์ด้านล่าง โดยให้ $CtrlTL_{1243}$ แทนการทำงานของสัญญาณไฟจราจรและตัวควบคุมสัญญาณไฟจราจรของระบบเดิม และให้ New_CtrlTL_{1243} แทนการทำงานของสัญญาณไฟจราจรและตัวควบคุมสัญญาณไฟจราจรของระบบที่ได้ถูกออกแบบใหม่

$$CtrlTL_{1234} = (new\ changeToYellow_{TL_{12}}, changeToYellow_{TL_{43}}, changeToYellow_{TL_{31}}, changeToYellow_{TL_{24}}, changeToGreen_{TL_{12}}, changeToGreen_{TL_{43}}, changeToGreen_{TL_{31}}, changeToGreen_{TL_{24}})$$
$$(Ctrl_{1234} \mid TL_{12} \mid TL_{43} \mid TL_{31} \mid TL_{24})$$

$$New\_CtrlTL_{1234} = (new\ changeToYellow_{TL_{1234}}, changeToYellow_{TL_{3224}}, changeToGreen_{TL_{1234}}, changeToGreen_{TL_{3224}})$$
$$(Ctrl_{1234} \mid TL_{1243} \mid TL_{3124})$$

ซึ่งผลจากการประมวลผลการทำงานพฤติกรรมของ $CtrlTL_{1243}$ นั้นจะได้กลุ่มของพฤติกรรมที่สามารถเกิดขึ้นได้ต่อสิ่งแวดล้อมดังนี้

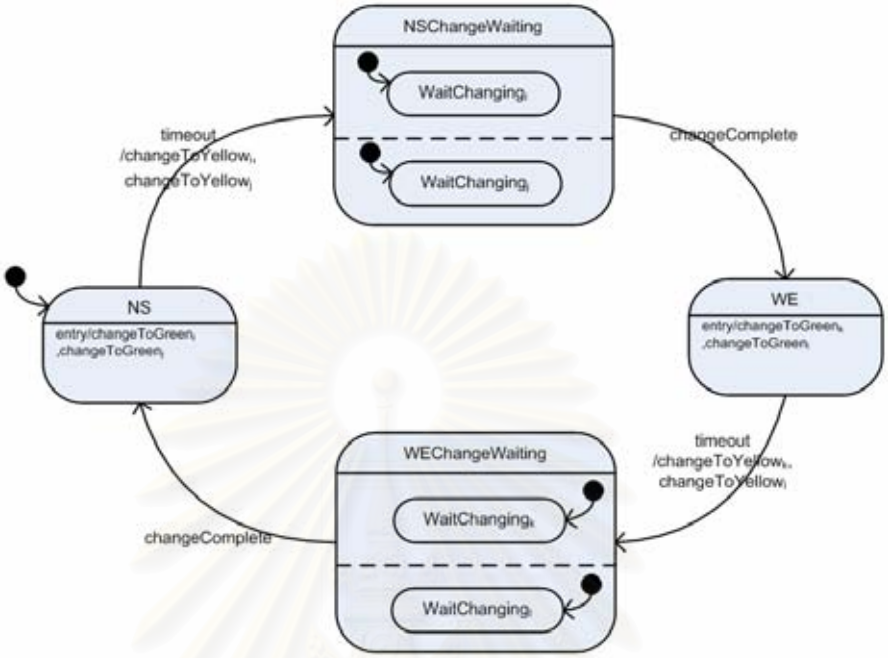
$$(start_{Car_{12}} \mid start_{Car_{43}}) \cdot (stop_{Car_{12}} \mid stop_{Car_{43}}) \cdot (start_{Car_{31}} \mid start_{Car_{24}}) \cdot (stop_{Car_{31}} \mid stop_{Car_{24}})$$

และกลุ่มของพฤติกรรมที่สามารถเกิดขึ้นได้ต่อสิ่งแวดล้อมของ New_CtrlTL_{1243} คือ

$$(start_{Car_{12}} \mid start_{Car_{43}}) \cdot (stop_{Car_{12}} \mid stop_{Car_{43}}) \cdot (start_{Car_{31}} \mid start_{Car_{24}}) \cdot (stop_{Car_{31}} \mid stop_{Car_{24}})$$

ซึ่งพบว่าผลที่ได้จากพฤติกรรมของสัญญาณไฟจราจรกับตัวควบคุมสัญญาณไฟจราจรของระบบใหม่จะเท่ากับพฤติกรรมของสัญญาณไฟจราจรกับตัวควบคุมสัญญาณไฟจราจรของระบบเดิม

4.3.2 การตรวจสอบพฤติกรรมการทำงานของวัตถุเมื่อทำงานร่วมกับวัตถุอื่นภายในระบบ



รูปที่ 4.14 แผนภาพสถานะตัวควบคุมสัญญาณไฟจราจรที่ออกแบบผิดพลาด

ในตัวอย่างนี้จะเป็นการแสดงให้เห็นถึงผลจากการออกแบบที่ผิดพลาดของระบบ โดยจะขอยกตัวอย่างการออกแบบที่ผิดพลาดของตัวควบคุมสัญญาณไฟจราจร โดยทำการเปลี่ยนเส้นการเปลี่ยนแปลงออกจากสถานะประกอบ *NSChangeWaiting* และ *WEChangeWaiting* จากเดิมที่เชื่อมโยงออกจากสถานะย่อยในแต่ละพื้นที่ย่อย ไปเป็นเชื่อมโยงจากสถานะประกอบแทนดังรูปที่ 4.14 ซึ่งหากผู้ออกแบบมองแผนภาพเพียงแผนภาพเดียวโดยไม่ได้เปรียบเทียบการทำงานร่วมกับแผนภาพอื่น จะพบว่าแผนภาพนี้มีการออกแบบที่ถูกต้อง แต่เมื่อตัวควบคุมสัญญาณไฟจราจรทำงานจริงร่วมกับสองสัญญาณไฟจราจรจะพบข้อผิดพลาดคือ สัญญาณส่งกลับจากสัญญาณไฟจราจรเพื่อแจ้งการเปลี่ยนสัญญาณไฟเสร็จเรียบร้อย คือ *changeComplete* จะมีสองสัญญาณ แต่ตัวควบคุมสัญญาณไฟจราจรจะมีส่วนที่รอรับเพียงส่วนเดียว ซึ่งผลของการทำงานจะทำให้เกิดการทำงานที่ชะงัก (deadlock) ของระบบ

จากแผนภาพสถานะของตัวควบคุมสัญญาณไฟจราจรในรูปที่ 4.14 จะได้นิพจน์ดังนี้

$$
\begin{aligned}
Ctrl_{ijkl} = (&new\ activeT_{I_{Ctrl_{ijkl}}}, exit_{Ctrl_{ijkl}}, exitComplete_{Ctrl_{ijkl}} \\
&, entry_{NS_{Ctrl_{ijkl}}}, activeT_{NS_{Ctrl_{ijkl}}}, entry_{NSChangeWaiting_{Ctrl_{ijkl}}}, activeT_{NSChangeWaiting_{Ctrl_{ijkl}}} \\
&, entry_{WE_{Ctrl_{ijkl}}}, activeT_{WE_{Ctrl_{ijkl}}}, entry_{WEChangeWaiting_{Ctrl_{ijkl}}}, activeT_{WEChangeWaiting_{Ctrl_{ijkl}}}) \\
&(I_{Ctrl_{ijkl}} | T_{I_{Ctrl_{ijkl}}} \\
&| NS_{Ctrl_{ijkl}} | T_{NS_{Ctrl_{ijkl}}} | NSChangeWaiting_{Ctrl_{ijkl}} | T_{NSChangWaitng_{Ctrl_{ijkl}}} \\
&| WE_{Ctrl_{ijkl}} | T_{WE_{Ctrl_{ijkl}}} | WEChangeWaiting_{Ctrl_{ijkl}} | T_{WEChangeWaiting_{Ctrl_{ijkl}}})
\end{aligned}
$$

โดยที่องค์ประกอบย่อยภายในที่นิพจน์เปลี่ยนแปลง คือ

$$
\begin{aligned}
T_{NSChangeWaiting_{Ctrl_{ijkl}}} &= activeT_{NSChangeWaiting_{Ctrl_{ijkl}}} \cdot T'_{NSChangeWaiting_{Ctrl_{ijkl}}} \\
T'_{NSChangeWaiting_{Ctrl_{ijkl}}} &= changeComplete_{Ctrl_{ijkl}} \cdot \overline{exit_{Ctrl_{ijkl}}} \cdot exitComplete_{Ctrl_{ijkl}} \\
&\quad \cdot \overline{entry_{WE_{Ctrl_{ijkl}}}} \cdot T_{NSChangeWaiting_{Ctrl_{ijkl}}} \\
T_{WEChangeWaiting_{Ctrl_{ijkl}}} &= activeT_{WEChangeWaiting_{Ctrl_{ijkl}}} \cdot T'_{WEChangeWaiting_{Ctrl_{ijkl}}} \\
T'_{WEChangeWaiting_{Ctrl_{ijkl}}} &= changeComplete_{Ctrl_{ijkl}} \cdot \overline{exit_{Ctrl_{ijkl}}} \cdot exitComplete_{Ctrl_{ijkl}} \\
&\quad \cdot \overline{entry_{NS_{Ctrl_{ijkl}}}} \cdot T_{WEChangeWaiting_{Ctrl_{ijkl}}}
\end{aligned}
$$

จากนิพจน์อธิบายตัวควบคุมสัญญาณไฟจราจรที่ผิดพลาด หากเราพิจารณาพฤติกรรมที่สามารถเกิดขึ้นได้ต่อสิ่งแวดล้อมของตัวควบคุมสัญญาณไฟจราจร จะได้ดังนี้

$$
\begin{aligned}
&(changeToGreen_{TL_{12}} \cdot changeToGreen_{TL_{43}}) \cdot (changeToYellow_{TL_{12}} \cdot changeToYellow_{TL_{43}}) \\
&\cdot (changeComplete) \\
&\cdot (changeToGreen_{TL_{31}} \cdot changeToGreen_{TL_{24}}) \cdot (changeToYellow_{TL_{31}} \cdot changeToYellow_{TL_{24}}) \\
&\cdot (changeComplete) \\
&\cdot (changeToGreen_{TL_{12}} \cdot changeToGreen_{TL_{43}}) \cdot ...
\end{aligned}
$$

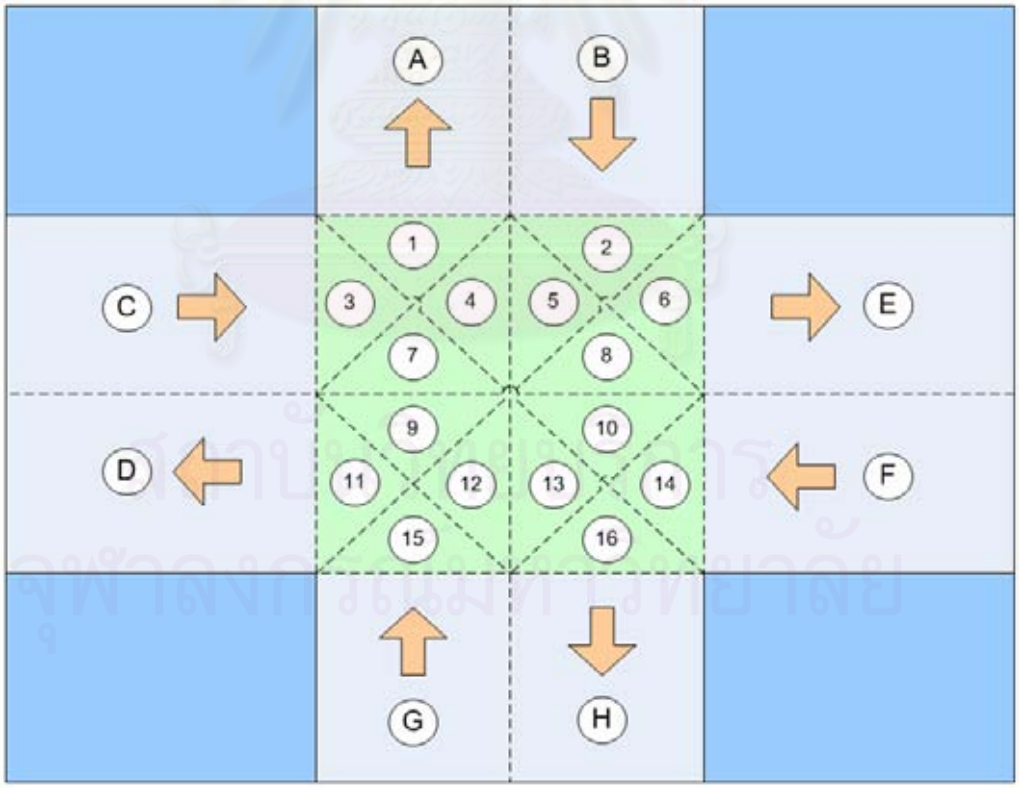
แต่เมื่อพิจารณาพฤติกรรมการทำงานของตัวควบคุมสัญญาณไฟจราจรควบคู่กับสิ่งสัญญาณไฟจราจรพบว่าระบบเกิดการทำงานชะงักขึ้น เมื่อเข้าสู่รอบการเปลี่ยนแปลงไฟในรอบที่สอง ดังพฤติกรรมด้านล่าง

$$
\begin{aligned}
&(changeToGreen_{TL_{12}} \cdot changeToGreen_{TL_{43}}) \cdot (changeToYellow_{TL_{12}} \cdot changeToYellow_{TL_{43}}) \\
&\cdot (changeComplete) \\
&\cdot (changeToGreen_{TL_{31}} \cdot changeToGreen_{TL_{24}}) \cdot (changeToYellow_{TL_{31}} \cdot changeToYellow_{TL_{24}}) \\
&\cdot (changeComplete) \\
&\cdot (changeToGreen_{TL_{12}}
\end{aligned}
$$

ซึ่งผลที่ได้ คือ พฤติกรรมการทำงานของตัวควบคุมสัญญาณไฟจราจรเมื่อทำงานแยกเดี่ยวไม่เท่ากับพฤติกรรมการทำงานของตัวควบคุมสัญญาณไฟจราจรเมื่อทำงานร่วมกับสัญญาณไฟจราจร ซึ่งเป็นส่วนช่วยให้ผู้ออกแบบระบบสามารถตรวจพบข้อผิดพลาดในการออกแบบได้

4.4 สรุปผลการแปลงและตรวจสอบ

จากกรณีศึกษาของการแปลงแผนภาพสถานะยูเอ็มแอลไปเป็นไพแคลคูลัสนั้น พบว่าขั้นตอนการแปลงโดยการพิจารณาส่วนของการแปลงเป็นจุดๆ ร่วมกับองค์ประกอบที่เชื่อมต่อติดกันสามารถทำได้โดยง่าย และในส่วนของการตรวจสอบความสัมพันธ์ของแผนภาพนั้น พบว่าเมื่อเราทำการแทนส่วนประกอบหนึ่งๆ ด้วยส่วนประกอบใหม่ เราสามารถตรวจสอบพฤติกรรมของระบบใหม่เทียบกับพฤติกรรมของระบบเดิมที่มีอยู่แล้วได้ ซึ่งจะเป็นส่วนช่วยให้สามารถตรวจสอบความถูกต้องในการออกแบบระบบใหม่เพื่อแทนที่ระบบเดิมได้ และในการตรวจสอบพฤติกรรมของวัตถุเมื่อทำงานร่วมกับวัตถุอื่นๆ ภายในระบบจะพบว่ามีความสามารถช่วยค้นหาจุดบกพร่องในการออกแบบได้



รูปที่ 4.15 แผนภาพสัญญาณไฟจราจรที่มีความซับซ้อน

ระบบสัญญาณไฟจราจรที่ได้นำเสนอในกรณีศึกษานี้เป็นเพียงตัวอย่างระบบจราจรอย่างง่ายเท่านั้น ซึ่งในการใช้งานจริงในชีวิตประจำวันนั้นระบบจะมีความซับซ้อนกว่านี้เป็นอย่างมาก

เช่น ในตัวอย่างรูปที่ 4.15 ซึ่งรถสามารถเดินทางได้หลายเส้นทาง คือ ทางตรง เลี้ยวซ้าย เลี้ยวขวา เลี้ยวกลับ ซึ่งในกรณีนี้การใช้พื้นที่ถนนส่วนกลางจะแตกต่างไป จำเป็นต้องแบ่งพื้นที่ให้ละเอียดยิ่งขึ้น กำหนดการใช้ถนนของแต่ละทิศทางให้ถูกต้อง ต้องเพิ่มสัญญาณไฟจราจรเพื่อให้สัญญาณการเดินรถในทิศทางต่างๆ รวมถึงออกแบบตัวควบคุมสัญญาณไฟให้ถูกต้อง เป็นต้น แต่อย่างไรก็ตามในระบบที่มีความซับซ้อนดังกล่าว หากเราสามารถอธิบายพฤติกรรมการทำงานของแต่ละวัตถุให้อยู่ในรูปของแผนภาพสถานะแล้ว เราก็สามารถแปลงแผนภาพดังกล่าวเป็นไพแคลคูลัสจากกฎการแปลง และตรวจสอบการออกแบบจากกฎการตรวจสอบที่ผู้วิจัยได้นำเสนอเช่นกัน

บทที่ 5

สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

งานวิจัยชิ้นนี้ได้นำเสนอวิธีการแปลงแผนภาพสถานะยูเอ็มแอลไปเป็นภาษารูปนัย คือ ซีอาร์อี และไพแคลคูลัส ซึ่งการแปลงแผนภาพไปเป็นซีอาร์อีจะมีนิพจน์ที่มีความซับซ้อนน้อย เหมาะกับการอธิบายระบบเบื้องต้นและผู้ใช้งานสามารถทำความเข้าใจได้ง่าย โดยในส่วนของไพแคลคูลัสนั้นจะสามารถแปลงแผนภาพได้ครอบคลุมองค์ประกอบต่างๆ มากกว่า ซึ่งจะทำให้สนับสนุนการออกแบบระบบที่มีความซับซ้อนและขนาดใหญ่ได้ดี แต่เพื่อที่จะให้กฎการแปลงรองรับควาามซับซ้อนของพฤติกรรมต่างๆ ของแต่ละองค์ประกอบได้นั้นทำให้นิพจน์ที่เป็นผลจากการแปลงค่อนข้างมีความซับซ้อนเพิ่มขึ้นเช่นกัน อย่างไรก็ตามผลที่ได้จากแปลงซึ่งอยู่ในรูปภาษาแบบรูปนัยก็มีคุณสมบัติที่เอื้ออำนวยต่อการตรวจสอบต่างๆ ซึ่งงานวิจัยชิ้นนี้ก็ได้นำเสนอแนวคิดการตรวจสอบแผนภาพยูเอ็มแอลเบื้องต้นด้วย คือ การตรวจสอบความเท่าเทียมกันของวัตถุซึ่งใช้ในการทดแทนกันของส่วนประกอบในการพัฒนาซอฟต์แวร์แบบอาศัยส่วนประกอบ และการตรวจสอบพฤติกรรมของวัตถุต่างๆ ซึ่งทำงานร่วมกันภายในระบบเพื่อตรวจสอบความสอดคล้องกันในการออกแบบพฤติกรรมของวัตถุแต่ละตัว

ในส่วนของการประยุกต์ใช้กับระบบต่างๆ งานวิจัยชิ้นนี้ได้นำเสนอตัวอย่างเทียบกับระบบควบคุมสัญญาณไฟจราจรซึ่งถือได้ว่าเป็นระบบที่มีความสำคัญและต้องการความถูกต้องสูง จากนั้นได้นำเสนอขั้นตอนการแปลงระบบดังกล่าวโดยละเอียด และสุดท้ายได้นำกฎการตรวจสอบที่ได้เสนอมาตรวจสอบความถูกต้องของระบบที่ได้รับการออกแบบ ซึ่งทั้งนี้ได้แสดงให้เห็นถึงผลที่ได้จากระบบที่ออกแบบมาไม่มีความสอดคล้องกันด้วย

จากกฎการแปลงแผนภาพและวิธีการตรวจสอบ พร้อมทั้งการนำเสนอตัวอย่างการใช้งานพบว่างานดังกล่าวสามารถนำไปใช้เพื่อช่วยให้การออกแบบและพัฒนาระบบมีความถูกต้องมากยิ่งขึ้น

5.2 ข้อเสนอแนะ

ในขั้นตอนการแปลงแผนภาพสถานะยูเอ็มแอลไปเป็นไพแคลคูลัสนั้นจะเห็นได้ว่ามีการพิจารณาที่เป็นระบบ และง่ายในการแทนที่องค์ประกอบแต่ละส่วนด้วยนิพจน์ต่างๆ แต่อย่างไรก็ดีนิพจน์ที่นำมาแทนที่ในบางองค์ประกอบนั้นก็มีความยาวทำให้ยากในการจดจำ อีกทั้งหากระบบ

ขยายใหญ่ขึ้น ปริมาณของนิพจน์ก็จะมากขึ้นเช่นกัน ดังนั้นวิธีการแปลงแบบอัตโนมัติจึงน่าจะเป็นแนวทางที่จะช่วยให้ขั้นตอนดังกล่าวสามารถกระทำได้อย่างรวดเร็ว และลดความผิดภาพจากการดำเนินการด้วยมนุษย์เอง

5.3 ผลงานที่ตีพิมพ์จากวิทยานิพนธ์

ส่วนหนึ่งของวิทยานิพนธ์นี้ได้รับการตีพิมพ์เป็นผลงานวิชาการในหัวข้อเรื่องดังต่อไปนี้

1) “Formalization of Concurrent UML Statechart Models Using Concurrent Regular Expressions” โดย ศิริชัย จันทร์สมัคร และอรรถสิทธิ์ สุรฤกษ์ ในงานประชุมวิชาการ National Computer Science and Engineering Conference (NCSEC2003)

2) “Formalization of UML Statechart Models Using Concurrent Regular Expressions” โดย ศิริชัย จันทร์สมัคร และอรรถสิทธิ์ สุรฤกษ์ ในงานประชุมวิชาการนานาชาติ Proceedings of the 27th Australian Computer Science Conference (ACSC2004)

รายการอ้างอิง

[1] G. Booch, J. Rumbaugh and I. Jacobson. “The Unified Modeling Language User Guide.” Addison Wesley, (1999).

[2] R. Cavada, A. Cimatti, E. Olivetti, M. Pistore, and M. Roveri. “NuSMV 2.1 User Manual”. http://nusmv.irst.itc.it, (2004).

[3] Vijay K. Garg, M.T. Ragunath. “Concurrent regular expressions and their relationship to Preti nets.” Theoretical Computer Science 96, (1992).

[4] David Harel. “Statecharts: A Visual Formalism for Complex Systems.” In Science of Computer Programming, (1987).

[5] Korenblat, K. and Priami, C. “Extraction of π-calculus specifications from UML sequence and state diagrams.”, Technical Report DIT-03-007, Department of Information and Communication Technology, University of Trento, (2003).

[6] Lam, V.S.W. Padget, J. “Symbolic model checking of UML statechart diagrams with an integrated approach.” 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, (2004).

[7] Milner, R., Parrow, J. and Walker, D. “A Calculus of Mobile Process (Part I and II)”, Information and Computation, (1992).

[8] Muan Yong Ng, Michael Butler. “Towards Formalizing UML State Diagrams in CSP.” First International Conference on Software Engineering and Formal Methods (SEFM'03), (2003).

[9] Mitsutaka Okazaki, Toshiaki Aoki, Takuya Katayama. “Formalizing sequence diagrams and state machines using Concurrent Regular Expression.” Proceedings of 2nd International Workshop on Scenarios and State Machines: Models, Algorithms, and Tools (SCESM’03), (2003).

[10] A.W. Roscoe. “The Theory and Practice of Concurrency.” Addison Wesley, (1997).

[11] OMG (The Object Management Group, Inc). “Unified Modeling Language Specification, Version 1.5”, http://www.omg.org/, (2003).

[12] The Formal Systems Website, Vendor for the FDR2. http://www.fsel.com/.

ประวัติผู้เขียนวิทยานิพนธ์

นายศิริชัย จันทร์สมัคร เกิดเมื่อวันที่ 21 สิงหาคม พ.ศ. 2519 เรียนจบการศึกษาระดับมัธยมศึกษาตอนปลายจากโรงเรียนเบญจะมะมหาราช อ.เมือง จ.อุบลราชธานี เข้ารับการศึกษาต่อที่มหาวิทยาลัยขอนแก่น ในคณะวิศวกรรมศาสตร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ สาขาวิชาวิศวกรรมคอมพิวเตอร์ และสำเร็จการศึกษาในระดับปริญญาบัณฑิตในปี พ.ศ. 2542 ปัจจุบันเป็นพนักงานประจำตำแหน่งที่ปรึกษาอาวุโส ที่บริษัท อวานาด (ประเทศไทย) จำกัด