การแปลงแผนภาพยูเอ็มแอลสเตทแมชชีนเป็นภาษาโพรเมลา

นางสาวปาณิสรา ดำจันทร์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2560

Transformation of UML State Machine Diagram to Promela

Miss Panisara Damjan

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Computer Science
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2017

หัวข้อวิทยานิพนธ์	การแปลงแผนภาพยูเอ็มแอลสเตทแมชชีนเป็นภาษาโพรเมลา
โดย	นางสาวปาณิสรา ดำจันทร์
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์

(รองศาสตราจารย์ ดร.สุพจน์ เตชวรสินสกุล)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ

(รองศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

(รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ)

.....กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.อาทิตย์ ทองทักษ์)

.....กรรมการภายนอกมหาวิทยาลัย

(ดร.เฉลิมทรัพย์ สังขวิจิตร)

ปาณิสรา ดำจันทร์ : การแปลงแผนภาพยูเอ็มแอลสเตทแมชชีนเป็นภาษาโพรเมลา (Transformation of UML State Machine Diagram to Promela) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: รศ. ดร.วิวัฒน์ วัฒนาวุฒิ, 127 หน้า.

การทำการทวนสอบเชิงรูปนัยโดยใช้วิธีการโมเดลเช็คกิงโดยเครื่องมือสปินนั้นต้องอาศัยแบบจำลองที่เป็นภาษาโพรเมลาซึ่งการทวนสอบเชิงรูปนัยนั้นสามารถทำได้ตั้งแต่ระยะต้นของกระบวนการพัฒนาซอฟต์แวร์โดยเฉพาะขั้นตอนการออกแบบ ปัจจุบันการออกแบบระบบได้มีการนำแผนภาพยูเอ็มแอลมาใช้โดยเฉพาะแผนภาพสเตทแมชชีนที่ช่วยอธิบายพฤติกรรมแบบพลวัตของระบบซอฟต์แวร์

วิทยานิพนธ์นี้จึงนำเสนอเครื่องมือในการแปลงแผนภาพสเตทแมชชีนที่มีการเขียนโอซีแอลไปเป็นภาษาโพรเมลา เพื่อเป็นทางเลือกหนึ่งสำหรับผู้ที่ต้องการทำการทวนสอบด้วยวิธีโมเดลเช็คกิงโดยเครื่องมือสปิน โดยวิทยานิพนธ์นี้สนใจสัญลักษณ์พื้นฐานของแผนภาพสเตทแมชชีน 5 สัญลักษณ์ด้วยกันคือ สัญลักษณ์เริ่มต้น สัญลักษณ์สิ้นสุด สัญลักษณ์สถานะ สัญลักษณ์ทางเลือก และสัญลักษณ์การเปลี่ยนสถานะ และมีแม่แบบในการแปลง 6 แม่แบบ สำหรับแปลงแผนภาพสเตทแมชชีนเป็นภาษาโพรเมลา ทั้งนี้เครื่องมือสามารถแปลงแผนภาพสเตทแมชชีนที่มีการเขียนโอซีแอลบนแผนภาพสเตทแมชชีนไปเป็นภาษาโพรเมลาได้ โดยเครื่องมือจะรับแผนภาพสเตทแมชชีนที่อยู่ในรูปแบบแฟ้มเอกสารเอกซ์เอ็มแอลเป็นข้อมูลนำเข้าในการแปลงและข้อมูลนำออกเป็นภาษาโพรเมลา

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา วิทยาศาสตร์คอมพิวเตอร์ ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

ปีการศึกษา 2560

5870242121 : MAJOR COMPUTER SCIENCE
KEYWORDS: UML STATE MACHINE DIAGRAMS, PROMELA, SPIN

PANISARA DAMJAN: Transformation of UML State Machine Diagram to Promela. ADVISOR: ASSOC. PROF. WIWAT VATANAWOOD, Ph.D., 127 pp.

Formal verification using model checking with SPIN needs a formal model written in Promela. However, the formal verification would be conducted in the early phase of software development process, especially in the design phase. Nowadays, the system design commonly uses UML diagrams, especially, the state machine diagram that describes the dynamic behavior of the software system.

This thesis proposes an automatic tool to translate a UML state machine diagram along with OCL into Promela code, in order to be an alternative model for verification in model checking method with SPIN. This thesis considers mainly on five elements of the diagram, including initial state, final state, state, choice, and transition. We provide five mapping rules and six mapping templates are provided for transforming a state machine to Promela code. The software tool is developed to perform the translation of the state machine diagram with OCL into Promela code. The input state machine is expected in XML format and the output is generated in Promela code.

Department: Computer Engineering Student's Signature .....
Field of Study: Computer Science Advisor's Signature .....
Academic Year: 2017

กิตติกรรมประกาศ

ขอกราบขอบพระคุณ รศ.ดร.วิวัฒน์ วัฒนาวุฒิ เป็นอย่างสูงสำหรับความเมตตากรุณาและความเอาใจใส่ ให้คำแนะนำทั้งการทำวิทยานิพนธ์และผลงานทางวิชาการ จากอาจารย์วิทยานิพนธ์ฉบับนี้ไม่อาจสำเร็จลงได้หากไม่ได้รับคำแนะนำ ความเอื้อเฟื้อ ความเอาใจใส่ และกำลังใจที่ดีเยี่ยมจากอาจารย์

ขอกราบขอบพระคุณ รศ.ดร.ธาราทิพย์ สุวรรณศาสตร์ ผศ.ดร.อาทิตย์ ทองทักษ์ และ ดร.เฉลิมทรัพย์ สังขวิจิตร กรรมการวิทยานิพนธ์ที่ให้คำแนะนำและเอาใจใส่ ส่งผลให้วิทยานิพนธ์ฉบับนี้สำเร็จลงได้นอกจากนี้ ผู้วิจัยได้รับความเอื้อเฟื้อที่ดีจากอาจารย์ผู้สอนในหลักสูตรและบุคลากรของคณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย และจากเพื่อนๆ นิสิต ผู้วิจัยต้องขอขอบคุณมา ณ ที่นี้ด้วย

และบุคคลที่ผู้วิจัยจะต้องยกย่องและต้องขอกราบขอบพระคุณท่านคือ บิดามารดาของผู้วิจัยที่เป็นกำลังใจและการให้การส่งเสริมสนับสนุนผู้วิจัยในทุกๆ ด้าน และขอขอบคุณพี่สาว พี่ชาย และหลานๆ ของผู้วิจัยซึ่งเป็นกำลังใจและคอยสนับสนุนมาโดยตลอด

สารบัญ

หน้า

บทคัดย่อภาษาไทย ..... ง
บทคัดย่อภาษาอังกฤษ ..... จ
กิตติกรรมประกาศ ..... ฉ
สารบัญ ..... ช
สารบัญตาราง ..... ฌ
สารบัญภาพ ..... ฎ
บทที่ 1 บทนำ 1
1.1 ที่มาและความสำคัญของปัญหา 1
1.2 วัตถุประสงค์ 1
1.3 ขอบเขตการดำเนินงาน 2
1.4 ขั้นตอนการดำเนินงาน 3
1.5 ผลความที่ตีพิมพ์จากงานวิจัย 3
บทที่ 2 ทฤษฎีที่เกี่ยวข้อง 4
2.1 ทฤษฎีที่เกี่ยวข้อง 4
2.2 งานวิจัยที่เกี่ยวข้อง 13
บทที่ 3 แนวคิดและรายละเอียดในการแปลง 16
3.1 แนวคิดและรายละเอียดในการแปลง 16
3.2 การรวมองค์ประกอบของภาษาโพรเมลา 26
3.3 การเพิ่มรายละเอียดในภาษาโพรเมลาเพื่อพร้อมสำหรับการทวนสอบ 35
บทที่ 4 การออกแบบและพัฒนาเครื่องมือ 37
4.1 การออกแบบเครื่องมือ 37
4.2 สภาพแวดล้อมในการพัฒนาระบบ 45

หน้า

4.3 ส่วนต่อประสานกับผู้ใช้ของเครื่องมือ 46
4.4 การใช้งานแอนท์เลอร์สร้างเลกเซอร์และพาสเซอร์ 49
บทที่ 5 การทดสอบเครื่องมือ 50
5.1 สภาพแวดล้อมที่ใช้ในการทดสอบ 50
5.2 การทดสอบ 50
5.3 ตัวอย่างการเพิ่มรายละเอียดในส่วนของโอเปอเรชัน 63
บทที่ 6 สรุปผลวิจัยและข้อเสนอแนะ 67
6.1 สรุปผลการวิจัย 67
6.2 ข้อเสนอแนะ 67
6.3 ประโยชน์ที่คาดว่าจะได้รับ 67
รายการอ้างอิง 68
ภาคผนวก ก ภาษาโพรเมลาที่ได้จากการแปลงและผลการจำลอง 69
ประวัติผู้เขียนวิทยานิพนธ์ 127

สารบัญตาราง

หน้า

ตารางที่ 2.1 ไวยากรณ์และตัวอย่างการเขียนเงื่อนไขก่อน 7
ตารางที่ 2.2 ไวยากรณ์และตัวอย่างการเขียนเงื่อนไขหลัง 7
ตารางที่ 2.3 ไวยากรณ์และตัวอย่างการเขียนค่ายืนยง 7
ตารางที่ 2.4 การประกาศ proctype 8
ตารางที่ 2.5 การประกาศ init 8
ตารางที่ 2.6 รูปแบบไวยากรณ์และแสดงตัวอย่างรูปแบบการเขียนช่องข้อความ 8
ตารางที่ 2.7 ชนิดข้อมูลของภาษาโพรเมลา 9
ตารางที่ 2.8 ตัวอย่างการเขียนคำสั่งลำดับออโตมิกซ์ 9
ตารางที่ 2.9 ตัวอย่างการเขียนคำสั่งโครงสร้างดีสเตป 10
ตารางที่ 2.10 ตัวอย่างการเขียนคำสั่งโครงสร้างทางเลือก 10
ตารางที่ 2.11 ตัวอย่างการเขียนคำสั่งข้ามแบบไม่มีเงื่อนไข 10
ตารางที่ 3.1 แม่แบบโครงสร้างภาษาโพรเมลาที่สอดคล้องกับสัญลักษณ์ของแผนภาพ
สเตทแมชชีน..... 32
ตารางที่ 4.1 รายละเอียดยูสเคสการนำเข้าแผนภาพสเตทแมชชีนเอกซ์เอ็มแอลไฟล์ 38
ตารางที่ 4.2 รายละเอียดยูสเคสการสร้างภาษาโพรเมลา..... 38
ตารางที่ 4.3 รายละเอียดยูสเคสการตรวจสอบแฟ้มเอกสารเอกซ์เอ็มแอล 39
ตารางที่ 4.4 รายละเอียดยูสเคสการสกัดและตีความ..... 39
ตารางที่ 4.5 รายละเอียดยูสเคสการแปลงแผนภาพสเตทแมชชีนเป็นภาษาโพรเมลา 39
ตารางที่ 4.6 รายละเอียดยูสเคสบันทึกไฟล์ 40
ตารางที่ 4.7 การสกัดแต่ละส่วนประกอบของแผนภาพของสเตทแมชชีน 44
ตารางที่ 4.8 การสกัดและเก็บข้อมูลของสัญลักษณ์การเปลี่ยนสถานะ..... 44
ตารางที่ 4.9 เทียบข้อมูลนำเข้าและการใช้แม่แบบแต่ละข้อสำหรับแปลงเป็นภาษาโพรเมลา..... 45

ตารางที่ 5.1 รายละเอียดส่วนประกอบของแผนภาพสเตทแมชชีนและโอซีแอลในแต่ละกรณีศึกษา..... 51

สารบัญภาพ

หน้า

ภาพที่ 2.1 สถานะเริ่มต้น 4
ภาพที่ 2.2 สถานะสิ้นสุด 4
ภาพที่ 2.3 สัญลักษณ์สถานะ 5
ภาพที่ 2.4 สัญลักษณ์ทางเลือก 5
ภาพที่ 2.5 สัญลักษณ์การเปลี่ยนสถานะ 5
ภาพที่ 2.6 ตัวอย่างแผนภาพสเตทแมชชีนของระบบการจองตั๋วหนัง 6
ภาพที่ 2.7 โครงสร้างของเครื่องมือสปิน 12
ภาพที่ 2.8 ตัวอย่างการใช้ไวยากรณ์ของอาร์เรย์เพื่อสร้างเลกเซอร์และพาสเซอร์จากแอนท์เลอร์ ... 13
ภาพที่ 3.1 กิจกรรมการทำงานของสัญลักษณ์สถานะ 18
ภาพที่ 3.2 แม่แบบโครงสร้างภาษาโพรเมลาส่วนของการแปลงสัญลักษณ์สถานะ 19
ภาพที่ 3.3 แม่แบบโครงสร้างภาษาโพรเมลาส่วนของการแปลงสัญลักษณ์เริ่มต้น 20
ภาพที่ 3.4 แม่แบบโครงสร้างภาษาโพรเมลาส่วนของการแปลงสัญลักษณ์สิ้นสุด 21
ภาพที่ 3.5 แม่แบบโครงสร้างภาษาโพรเมลาส่วนของการแปลงสัญลักษณ์การเปลี่ยนสถานะ 22
ภาพที่ 3.6 แม่แบบโครงสร้างภาษาโพรเมลาส่วนของการแปลงสัญลักษณ์สัญลักษณ์ทางเลือกที่มีเส้นการเปลี่ยนสถานะเข้า 1 เส้นและออก n เส้น 23
ภาพที่ 3.7 แม่แบบโครงสร้างภาษาโพรเมลาส่วนของการแปลงสัญลักษณ์สัญลักษณ์ทางเลือกที่มีเส้นการเปลี่ยนสถานะเข้า n เส้นและออก 1 เส้น 24
ภาพที่ 3.8ค่ายืนยงในรูปแบบโอซีแอลและภาษาโพรเมลา 25
ภาพที่ 3.9 ตัวอย่าง proctype ที่ได้จากการแปลงแอคชัน 26
ภาพที่ 3.10 ลำดับการรวมองค์ประกอบของภาษาโพรเมลาที่ได้จากการแปลง 27
ภาพที่ 3.11 ตัวอย่างจำลองแผนภาพสเตทแมชชีน 27
ภาพที่ 3.12 ภาษาโพรเมลที่ได้จากการแปลงสถานะเริ่มต้น 28

ภาพที่ 3.13 ภาษาโพรเมลาที่ได้จากการแปลงสัญลักษณ์สถานะ C 28
ภาพที่ 3.14 ภาษาโพรเมลาที่ได้จากการแปลงสัญลักษณ์ทางเลือก 29
ภาพที่ 3.15 ตัวอย่างบางส่วนที่ได้จากการเรียงองค์ประกอบของภาษาโพรเมลา 30
ภาพที่ 3.16 ตัวอย่างการเพิ่มรายละเอียดลงใน proctype 36
ภาพที่ 4.1 แผนภาพยูสเคสแสดงขอบเขตและการทำงานของเครื่องมือ 37
ภาพที่ 4.2 แผนภาพคลาสของเครื่องมือ 41
ภาพที่ 4.3 แผนภาพกิจกรรมของเครื่องมือ 41
ภาพที่ 4.4 แผนภาพของสเตทแมชชีนที่วาดด้วยเครื่องมือวิชวลพาราไดม์ 42
ภาพที่ 4.5แผนภาพของสเตทแมชชีนในรูปแบบแฟ้มเอกสารเอกซ์เอ็มแอล 43
ภาพที่ 4.6 การสกัดแต่ละส่วนประกอบของแผนภาพของสเตทแมชชีน 43
ภาพที่ 4.7 การสกัดแต่ละส่วนประกอบของสัญลักษณ์การเปลี่ยนสถานะ 44
ภาพที่ 4.8 ชุดเครื่องมือ IntelliJI DEA 2016.2 46
ภาพที่ 4.9 นำเข้าแฟ้มเอกสารแผนภาพสเตทแมชชีน 46
ภาพที่ 4.10 ที่อยู่ของแฟ้มเอกสาร 47
ภาพที่ 4.11 ภาษาโพรเมลาที่ได้จากการแปลง 47
ภาพที่ 4.12 ภาษาโพรเมลาที่ได้จากการแปลงส่วนของสัญลักษณ์สถานะ 48
ภาพที่ 4.13 การบันทึกภาษาโพรเมลา 48
ภาพที่ 4.14 หน้าจอการระบุตำแหน่งที่ต้องการบันทึกแฟ้มเอกสารภาษาโพรเมลา 49
ภาพที่ 5.1 แผนภาพสเตทแมชชีนระบบออเดอร์สินค้า 52
ภาพที่ 5.2 ตัวแปรที่เกิดจากเงื่อนไขก่อน เงื่อนไขหลัง เหตุการณ์และเงื่อนไขการ์ดของแผนภาพสเตทแมชชีนของระบบออเดอร์สินค้า 53
ภาพที่ 5.3 ภาษาโพรเมลาที่ได้จากการแปลงสถานะ create order 53
ภาพที่ 5.4 หน้าจอผลลัพธ์การจำลองภาษาโพรเมลาที่ได้จากการแปลงแผนภาพสเตทแมชชีนของระบบออเดอร์สินค้า 54

ภาพที่ 5.5 เส้นทางการเปลี่ยนสถานะที่ได้เมื่อเทียบกับภาษาโพรเมลาที่ได้จากการแปลงแผนภาพสเตทแมชชีนของระบบออเดอร์สินค้า 55

ภาพที่ 5.6 แผนภาพสเตทแมชชีนของระบบตรวจสอบบุคคลและยานพาหนะ 56

ภาพที่ 5.7 ภาษาโพรเมลาที่ได้จากการแปลงการเปลี่ยนสถานะจากสถานะหนึ่งไปหลายสถานะ 57

ภาพที่ 5.8 ภาษาโพรเมลาที่ได้จากการแปลงทางเลือกที่มีเส้นการเปลี่ยนสถานะเข้า 1 เส้นและออก n เส้น 57

ภาพที่ 5.9 ภาษาโพรเมลาที่ได้จากการแปลงทางเลือกที่มีเส้นการเปลี่ยนสถานะเข้า n เส้นและออก 1 เส้น 58

ภาพที่ 5.10 เส้นทางการเปลี่ยนสถานะที่ได้เมื่อเทียบกับลำดับการเอ็กซิคิวต์ภาษาโพรเมลาที่ได้จากการแปลงแผนภาพสเตทแมชชีนของระบบตรวจสอบบุคคลและยานพาหนะ 59

ภาพที่ 5.11 แผนภาพสเตทแมชชีนระบบซื้อของออนไลน์โดยจ่ายเงินด้วยบัตรเครดิต 60

ภาพที่ 5.12 ภาษาโพรเมลาที่ได้จากการแปลงสถานะpayment ที่มีการเขียนเงื่อนไขก่อน 60

ภาพที่ 5.13 ภาษาโพรเมลาที่ได้จากการแปลงค่ายืนยง 61

ภาพที่ 5.14 เงื่อนไขการ์ดและเหตุการณ์ที่ปรากฏบนเส้นเปลี่ยนสถานะของภาพที่ 5.11 62

ภาพที่ 5.15 ตัวอย่างการเพิ่มรายละเอียดในส่วนของโอเปอเรเตอร์ 64

ภาพที่ 5.16 การเปลี่ยนสถานะของแผนภาพระบบออเดอร์สินค้าหลังจากการเปลี่ยนค่าตัวแปรและเพิ่มรายละเอียดใน proctype Stateviewsummary 65

ภาพที่ 5.17 ค่าตัวแปรที่เปลี่ยนหลังเพิ่มส่วนโอเปอเรชัน 65

ภาพที่ 5.18 ผลการจำลองภาษาโพรเมลาที่ได้จากการแปลงแผนภาพสเตทแมชชีนของระบบออเดอร์สินค้าหลังเพิ่มโอเปอเรชัน 66

ภาพที่ ก.1 ภาษาโพรเมลาที่ได้จากการแปลงแผนภาพสเตทแมชชีนของระบบออเดอร์สินค้า 80

ภาพที่ ก.2 ผลการจำลองภาษาโพรเมลาที่ได้จากการแปลงแผนภาพสเตทแมชชีนของระบบออเดอร์สินค้า 91

ภาพที่ ก.3 ภาษาโพรเมลาที่ได้จากการแปลงแผนภาพสเตทแมชชีนของระบบตรวจสอบบุคคลและยานพาหนะ 106

ภาพที่ ก.4 ผลการจำลองภาษาโพรเมลาที่ได้จากการแปลงแผนภาพสเตทแมชชีนของระบบตรวจสอบบุคคลและยานพาหนะ 112

ภาพที่ ก.5 ภาษาโพรเมลาที่ได้จากการแปลงแผนภาพสเตทแมชชีนแผนภาพสเตทแมชชีนของระบบซื้อของออนไลน์และจ่ายเงินด้วยบัตรเคดิต 120

ภาพที่ ก.6 ผลการจำลองภาษาโพรเมลาที่ได้จากการแปลงแผนภาพสเตทแมชชีนของระบบออเดอร์สินค้า 126

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหา

ในการทวนสอบเชิงรูปนัย (Formal Verification) โดยใช้วิธีการโมเดลเช็คกิง (Model Checking) มีความจำเป็นต้องใช้แบบจำลองเชิงรูปนัย (Formal Model) เพื่อนำไปใช้ในการทวนสอบ ซึ่งเครื่องมือในการทำโมเดลเช็คกิงที่ได้รับความนิยมเครื่องมือหนึ่งคือเครื่องมือสปิน (SPIN : Simple Promela Interpreter) [1] แต่เครื่องมือสปินนั้นมีข้อจำกัดเรื่องภาษาที่ใช้คือเครื่องมือสปินรองรับเฉพาะภาษาโพรเมลาเท่านั้น จึงจำเป็นต้องเขียนแบบจำลองเชิงรูปนัยให้อยู่ในรูปแบบของภาษาโพรเมลาเพื่อให้รองรับกับเครื่องมือสปิน แต่ทั้งนี้การเขียนภาษาโพรเมลาจากระบบที่ต้องการทวนสอบเพื่อใช้เป็นแบบจำลองเชิงรูปนัยนั้นเป็นเรื่องที่ไม่ง่าย เนื่องจากโครงสร้างของภาษาโพรเมลาแตกต่างจากภาษาโปรแกรมทั่วไป แต่อย่างไรก็ตามในทางปฏิบัติระบบซอฟต์แวร์หรือฮาร์ดแวร์ใดๆ ก็ตาม สามารถเขียนอธิบายพฤติกรรมให้อยู่ในรูปแบบแผนภาพสเตทแมชชีน (State Machine Diagrams) ได้ จึงเป็นที่มาของงานวิจัยนี้ซึ่งนำเสนอการแปลงแผนภาพสเตทแมชชีนเป็นภาษาโพรเมลาเพื่อใช้เป็นแบบจำลองเชิงรูปนัยสำหรับการทำโมเดลเช็คกิงด้วยเครื่องมือสปิน

งานวิจัยนี้นำเสนอเครื่องมือช่วยแปลงแผนภาพสเตทแมชชีนที่ได้จากขั้นตอนการออกแบบไปเป็นภาษาโพรเมลา จากงานวิจัยที่ผ่านมามีผู้เสนอกฎในการแปลงแผนภาพสเตทแมชชีนไปเป็นภาษาโพรเมลาบ้างแล้ว [2, 3] แต่มีข้อจำกัดคือเมื่อทำการแปลงแล้วภาษาโพรเมลาที่ได้จากการแปลงยังมีรายละเอียดไม่เพียงพอต่อการนำไปใช้งานต่อ ดังนั้นในงานวิจัยนี้จึงได้มีการเพิ่มการเขียนโอซีแอลหรือภาษาที่ใช้ระบุเงื่อนไขบังคับสำหรับอ็อบเจกต์ (OCL : Object constraint language) เพื่อเขียนอธิบายเงื่อนไขก่อน (Precondition) เงื่อนไขหลัง (Postcondition) และค่ายืนยง (Invariant) บนแผนภาพสเตทแมชชีน ซึ่งจะทำให้ได้ภาษาโพรเมลาที่มีรายละเอียดมากขึ้นและเนื่องจากยังไม่มีเครื่องมือกึ่งอัตโนมัติที่ทำการแปลงแผนภาพสเตทแมชชีนที่มีการเขียนโอซีแอลอธิบายเงื่อนไขก่อน เงื่อนไขหลังและค่ายืนยงดังกล่าวและคาดว่างานวิจัยนี้จะมีประโยชน์สำหรับการนำมาใช้ทำการทวนสอบเชิงรูปนัยด้วยเครื่องมือสปิน

1.2 วัตถุประสงค์

1) ออกแบบกฎการแปลงแม่แบบโครงสร้างของภาษาโพรเมลาในการแปลงแผนภาพสเตทแมชชีนพร้อมคำอธิบายโอซีแอลบนแผนภาพไปเป็นภาษาโพรเมลา

2) พัฒนาเครื่องมือกึ่งอัตโนมัติในการนำกฎการแปลงแม่แบบโครงสร้างของภาษาโพรเมลาแปลงแผนภาพสเตทแมชชีนที่พร้อมคำอธิบายโอซีแอล

1.3 ขอบเขตการดำเนินงาน

1) แผนภาพสเตทแมชชีนต้องเป็นแผนภาพที่มีความสมบูรณ์โดยมีการระบุข้อมูลดังต่อไปนี้
   1.1) แผนภาพสเตทแมชชีนที่สนใจในงานวิจัยนี้ใช้มาตรฐานแบบยูเอ็มเอล 2.5 และส่วนประกอบของแผนภาพสเตทแมชชีนในงานวิจัยนี้จะครอบคลุม ส่วนประกอบบางส่วนของแผนภาพดังนี้คือ สัญลักษณ์สถานะเริ่มต้น สัญลักษณ์สถานะสิ้นสุด สัญลักษณ์สถานะ สัญลักษณ์ทางเลือกและสัญลักษณ์การเปลี่ยนสถานะ
   1.2) สัญลักษณ์สถานะต้องมีชื่อกำกับเพราะต้องนำไปใช้สร้างเป็นส่วนประกอบของภาษาโพรเมลา
   1.3) การเขียนโอซีแอลบนแผนภาพสเตทแมชชีนรองรับเฉพาะเงื่อนไขก่อน เงื่อนไขหลัง และค่ายืนยง เท่านั้น โดยการเขียนโอซีแอลนั้นสามารถเขียนอยู่ในสัญลักษณ์สถานะ
   1.4) เงื่อนไขการ์ดสามารถเขียนในรูปแบบของโอซีแอลได้
   1.5) ไม่รองรับคอมโพสิตสเตท (Composite state)
2) นำแม่แบบ (Template) โครงสร้างภาษาโพรเมลาที่ออกแบบไว้มาช่วยในการแปลงแผนภาพสเตทแมชชีนไปเป็นภาษาโพรเมลา โดยแม่แบบโครงสร้างภาษาโพรเมลาจะอยู่ในรูปแบบกฎการแปลง (Mapping rule)
3) พัฒนาเครื่องมือกึ่งอัตโนมัติในการแปลงแผนภาพสเตทแมชชีนที่มีการเขียนโอซีแอลไปเป็นภาษาโพรเมลาโดยเครื่องมือที่พัฒนาจะมีคุณสมบัติดังนี้
   3.1) เครื่องมือสามารถรับข้อมูลเป็นแผนภาพสเตทแมชชีนที่อยู่ในรูปแบบแฟ้มเอกสารเอกซ์เอ็มแอลและสามารถตรวจสอบได้ว่าเอกซ์เอ็มแอลที่รับเข้ามานั้นเป็นแผนภาพสเตทแมชชีนหรือไม่
   3.2) เครื่องมือสามารถอ่านแผนภาพสเตทแมชชีนที่อยู่ในรูปแบบเอกซ์เอ็มแอลไฟล์ได้โดยนำความสามารถของเครื่องมือแอนท์เลอร์มาใช้งาน
   3.3) เครื่องมือสามารถแปลงแผนภาพสเตทแมชชีนที่อยู่ในรูปแบบเอกซ์เอ็มแอลไฟล์ไปเป็นภาษาโพรเมลาได้โดยกฎการแปลงแม่แบบโครงสร้างของภาษาโพรเมลาที่ออกแบบขึ้นมาและสามารถแปลงเงื่อนไขก่อน เงื่อนไขหลังและค่ายืนยงที่ถูกเขียนอยู่ในรูปแบบของภาษามาตรฐาน โอซีแอลบนแผนภาพสเตทแมชชีนไปเป็นโพรเมลาได้
   3.4) เครื่องมือสามารถส่งออกผลลัพธ์ที่อยู่ในรูปแบบแฟ้มเอกสารนามสกุล .pml ได้

3.5) ตรวจสอบความถูกต้องของเครื่องมือที่พัฒนาด้วยการใช้กรณีศึกษาโดยเพิ่มรายละเอียดในไฟล์นามสกุล .pml ให้พร้อมใช้งานในการทวนสอบด้วยเครื่องมือสปินโดยใช้กรณีศึกษาที่ครอบคลุมสัญลักษณ์สถานะ สัญลักษณ์สถานะเริ่มต้น สัญลักษณ์สถานะสิ้นสุด สัญลักษณ์ทางเลือกและสัญลักษณ์การเปลี่ยนสถานะ

1.4 ขั้นตอนการดำเนินงาน

1) ศึกษาและทำความเข้าใจทฤษฎีการออกแบบแผนภาพสเตทแมชชีน
2) ศึกษาและทำความเข้าใจทฤษฎีโอซีแอล
3) ศึกษารูปแบบไวยากรณ์ของภาษาโพรเมลา
4) ออกแบบกฎการแปลงแม่แบบโครงสร้างของภาษาโพรเมลาสำหรับการแปลงแผนภาพสเตทแมชชีนที่มีการเขียนโอซีแอลบนแผนภาพไปเป็นภาษาโพรเมลา
5) ศึกษาและทำความเข้าใจทฤษฎีเอกซ์เอ็มแอล
6) ศึกษารูปแบบและวิธีการใช้งานเครื่องมือตรวจสอบแบบจำลองสปิน
7) วิเคราะห์และกำหนดความสามารถของเครื่องมือที่พัฒนา
8) พัฒนาเครื่องมือกึ่งอัตโนมัติสำหรับการแปลงแผนภาพสเตทแมชชีนไปเป็นภาษาโพรเมลา
9) ทดสอบเครื่องมือ
10) สรุปผลวิจัยและข้อเสนอแนะ
11) จัดทำวิทยานิพนธ์

1.5 ผลความที่ตีพิมพ์จากงานวิจัย

ส่วนหนึ่งของงานวิจัยนี้ได้รับการตีพิมพ์เป็นบทความวิชาการเรื่อง “Translating UML State Machine Diagram into Promela ” โดย ปาณิสรา ดำจันทร์ และวิวัฒน์ วัฒนาวุฒิ ในการประชุมวิชาการ “The 25th International MultiConference of Engineers and Computer Scientists (IMECS 2017)” ระหว่างวันที่ 15-17 มีนาคม 2560 ณ โรงแรม เดอะ รอยัล การ์เดน เมืองเกาลูน เขตบริหารพิเศษฮ่องกงแห่งสาธารณรัฐประชาชนจีน

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 แผนภาพสเตทแมชชีน [4]

1) แผนภาพสเตทแมชชีน คือ แผนภาพที่ใช้อธิบายสถานะและพฤติกรรมของวัตถุ (Object) ในระบบ รวมทั้งเหตุการณ์ต่างๆ ที่ทำให้วัตถุเปลี่ยนสถานะจากสถานะหนึ่งไปอีกสถานะหนึ่ง ณ ช่วงชีวิตของวัตถุ ในงานวิจัยนี้สนใจสัญลักษณ์ของแผนภาพสเตทแมชชีน 5 สัญลักษณ์ ดังต่อไปนี้

1.1) สัญลักษณ์สถานะเริ่มต้น (Initial state)

สถานะเริ่มต้นคือ จุดเริ่มต้นของวัตถุซึ่งแสดงให้เห็นถึงจุดเริ่มต้นของกิจกรรมต่างๆ ที่เกิดขึ้นกับวัตถุบนแผนภาพ โดยสัญลักษณ์จะมีลักษณะเป็นวงกลมทึบดังภาพที่ 2.1

ภาพที่ 2.1 สถานะเริ่มต้น [4]

1.2) สัญลักษณ์สถานะสิ้นสุด (Final state)

สถานะสิ้นสุดคือ จุดสิ้นสุดพฤติกรรมของวัตถุซึ่งแสดงให้เห็นถึงจุดสิ้นสุดของกิจกรรมต่างๆ ของวัตถุบนแผนภาพ โดยจะมีลักษณะวงกลม 2 วงซ้อนกันอยู่โดยวงกลมด้านในเป็นสีดำทึบ ดังภาพที่ 2.2

ภาพที่ 2.2 สถานะสิ้นสุด [4]

1.3) สัญลักษณ์สถานะ (State)

สัญลักษณ์สถานะคือ สถานะของวัตถุนั้นๆ ซึ่งแสดงให้เห็นถึงค่าของวัตถุหรือการที่วัตถุกำลังรอเหตุการณ์ใดเหตุการณ์หนึ่ง ณ สถานะนั้น โดยสัญลักษณ์จะมีลักษณะเป็นสี่เหลี่ยมมุมมน ดังแสดงในภาพที่ 2.3

ภาพที่ 2.3 สัญลักษณ์สถานะ [4]

1.4) สัญลักษณ์ทางเลือก (Choice)

สัญลักษณ์ทางเลือกคือ สถานะเทียมที่แสดงถึงการเลือกทำทางใดทางหนึ่งเมื่อมีการประเมินเงื่อนไขใดๆ แล้วพบว่าเงื่อนไขนั้นเป็นจริงซึ่งทำให้เกิดการเปลี่ยนสถานะโดยจะมีลักษณะเป็นสามเหลี่ยมเปียกปูนดังแสดงในภาพที่ 2.4

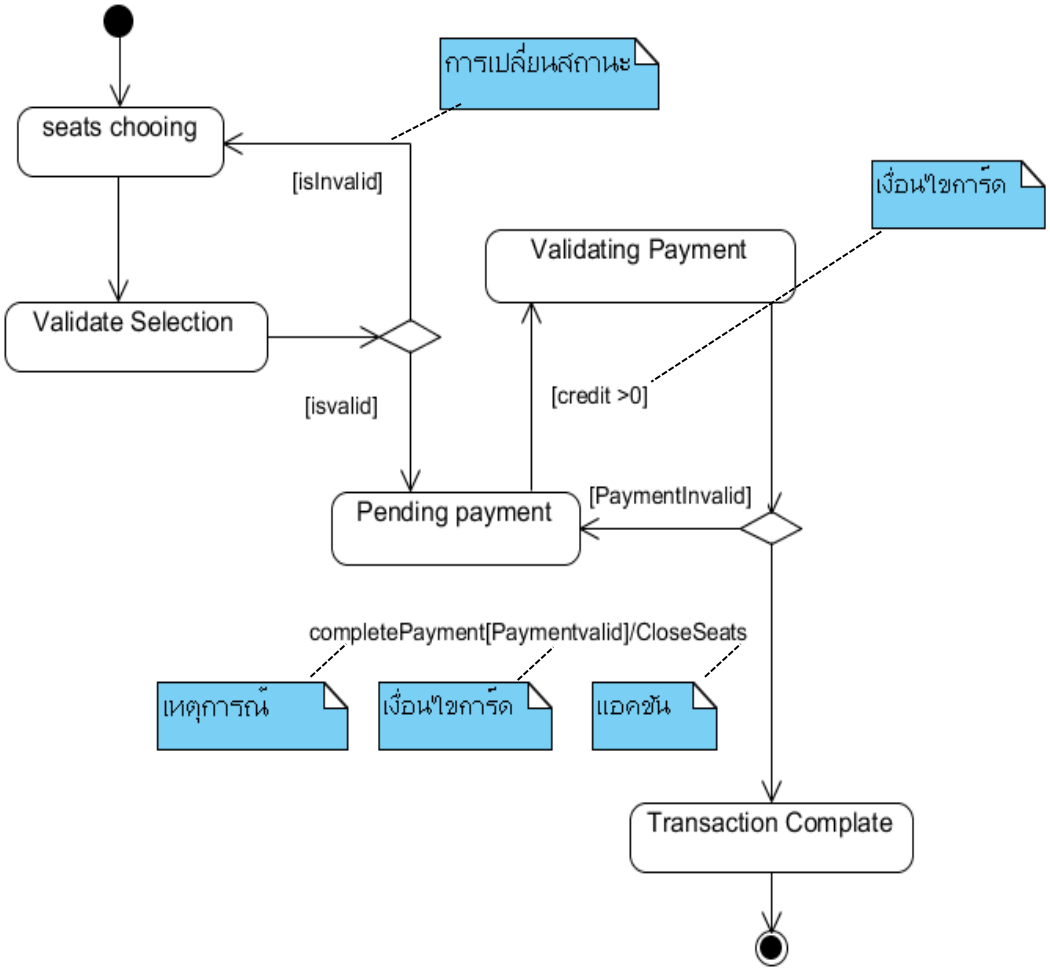
ภาพที่ 2.4 สัญลักษณ์ทางเลือก [4]

1.5) สัญลักษณ์การเปลี่ยนสถานะ (Transition)

สัญลักษณ์การเปลี่ยนสถานะคือ เส้นที่แสดงให้เห็นถึงความสัมพันธ์ระหว่างสถานะโดยมีเหตุการณ์ เงื่อนไขการ์ดและแอคชันเป็นลาเบลกำกับบนเส้นเพื่อบ่งบอกว่าก่อนที่วัตถุจะเปลี่ยนสถานะหนึ่งไปอีกสถานะหนึ่งได้นั้นต้องมีการประเมินลาเบลดังกล่าวก่อนถ้าพบว่าเป็นจริงถึงจะทำการเปลี่ยนสถานะได้ โดยจะมีลักษณะเป็นเส้นทึบและหัวลูกศรชี้ทางเดียว ดังแสดงในภาพที่ 2.5

ภาพที่ 2.5 สัญลักษณ์การเปลี่ยนสถานะ [4]

ภาพที่ 2.6 แสดงถึงตัวอย่างแผนภาพสเตทแมชชีนของระบบจองตั๋วหนังซึ่งประกอบไปด้วยสัญลักษณ์ต่างๆ ของแผนภาพสเตทแมชชีน เช่นสัญลักษณ์สถานะ สัญลักษณ์ทางเลือก สัญลักษณ์สถานะเริ่มต้น และสัญลักษณ์สถานะสิ้นสุด เป็นต้น



ภาพที่ 2.6 ตัวอย่างแผนภาพสเตทแมชชีนของระบบการจองตั๋วหนัง [5]

2.1.2 โอซีแอล

โอซีแอล [6] คือ ภาษารูปนัย (Formal language) ที่ถูกพัฒนาขึ้นโดยไอบีเอ็ม (IBM) เพื่อใช้อธิบายหรือกำหนดเงื่อนไขบังคับ (Constraint) เช่น เงื่อนไขก่อน เงื่อนไขหลังรวมถึงค่ายืนยงและข้อคำถาม (Query) บนแผนภาพยูเอ็มแอล ซึ่งในงานวิจัยนี้สนใจเงื่อนไขบังคับของโอซีแอลโดยสนใจเงื่อนไขก่อน เงื่อนไขหลังและค่ายืนยงซึ่งมีรายละเอียดดังต่อไปนี้

1) เงื่อนไขก่อนคือ เงื่อนไขที่ต้องตรวจสอบก่อนเข้าการทำงานซึ่งเงื่อนไขนี้ต้องเป็นจริงเท่านั้นถึงจะยอมให้ทำงานในลำดับต่อไปได้ เช่น ก่อนการถอนเงินจากเครื่องเอทีเอ็มต้องมีการตรวจสอบเงื่อนไขก่อนว่ายอดเงินในบัญชีต้องมีมากกว่าจำนวนเงินที่ต้องการถอนเงินหรือไม่ ซึ่งเรียกเงื่อนไขที่ตรวจสอบก่อนการถอนนี้ว่าเงื่อนไขก่อน โดยการเขียนเงื่อนไขก่อนเขียนตามไวยากรณ์ดังตารางที่ 2.1

ตารางที่ 2.1 ไวยากรณ์และตัวอย่างการเขียนเงื่อนไขก่อน [6]

ไวยากรณ์	ตัวอย่าง
context<classifier>::<operation>(<parameters>) pre[<constraint name>]:< actualpreconditions >	contextATM::withdraw(amount:Integet) pre:customer.account.balance >=amount

2) เงื่อนไขหลัง คือเงื่อนไขที่เป็นจริงเสมอเมื่อทำงานเสร็จสิ้นโดยหากทุกการทำงานใดๆ มีการกำหนดเงื่อนไขหลังจากการทำงานแล้วเสร็จต้องมีการตรวจสอบว่าเงื่อนไขหลังที่ระบุดังกล่าวเป็นจริงหรือไม่และเรียกเงื่อนที่ตรวจสอบนี้ว่าเงื่อนไขหลัง เช่น หลังการฝากเงินในบัญชีต้องมากกว่าศูนย์ โดยการเขียนเงื่อนไขหลังเขียนตามไวยากรณ์ดังตารางที่ 2.2

ตารางที่ 2.2 ไวยากรณ์และตัวอย่างการเขียนเงื่อนไขหลัง [6]

ไวยากรณ์	ตัวอย่าง
context <classifier>::<operation> (<parameters>) post[<constraint name>]:<actualpostconditions >	context ATM::deposit(amount:Integet) post: customer.account.balance > 0

3) ค่ายืนยง คือค่าที่ต้องเป็นจริงตั้งแต่เริ่มต้นการทำงานจนกระทั่งจบการทำงาน เช่นตัวอย่างค่ายืนยงของระบบบัญชีธนาคาร คือไม่ว่าจะฝากหรือถอนเงินก็ตาม ยอดเงินคงเหลือในบัญชีต้องมากกว่าหรือเท่ากับศูนย์ โดยการเขียนค่ายืนยงนั้นสามารถเขียนตามไวยากรณ์ดังตารางที่ 2.3

ตารางที่ 2.3 ไวยากรณ์และตัวอย่างการเขียนค่ายืนยง [6]

ไวยากรณ์	ตัวอย่าง
context <classifier> inv [<constraint name>]: < actual invariants >	context getBalance inv:self.balance >= 0

2.1.3 ภาษาโพรเมลา [1]

ภาษาโพรเมลา คือ ภาษาที่นำมาใช้ในการสร้างแบบจำลองหรือโมเดลสำหรับการทวนสอบ (Verification model) ระบบเป้าหมายจากพฤติกรรมของระบบ ดังนั้นจึงสามารถนำภาษาโพรเมลามาสร้างเป็นโมเดลของพฤติกรรมการทำงานของระบบที่ต้องการทวนสอบคุณสมบัติที่ต้องการตรวจสอบได้อย่างมีประสิทธิภาพโดยภาษาโพรเมลามีองค์ประกอบดังนี้

1) กระบวนการ (Process) คือ ตัวกำหนดพฤติกรรมของโมเดลที่ต้องการทวนสอบ โดยการกำหนดพฤติกรรมดังกล่าวจะสามารถแบ่งได้เป็นสองชนิดคือ กระบวนการ proctype และกระบวนการ init โดยการประกาศ proctype ทำได้โดยการประกาศชื่อตามหลังคำหลัก proctype ดังตัวอย่างในตารางที่ 2.4

ตารางที่ 2.4 การประกาศ proctype [1]

คำสั่งภาษาโพรเมลา	ความหมาย
1: proctype A() { 2: byte activeflag; 3: activeflag =1 4:}	การประกาศ proctypeชื่อ A โดยพฤติกรรมของ A คือมีการประกาศตัวแปรชื่อ activeflag เป็นตัวแปรโลคอลในบรรทัดที่ 2 และทำการให้ค่าตัวแปรดังกล่าวให้มีค่าเป็นหนึ่งในบรรทัดที่ 3

กระบวนการ init.เป็นกระบวนการที่เปรียบเสมือนโปรแกรมหลัก (Main program) ของภาษาโพรเมลา โดยมีหน้าที่ในการสั่งเอ็กซิคิวต์ (Execute) กระบวนการ proctype โดย init จะสามารถถูกประกาศได้เพียงที่เดียวในภาษาโพรเมลาซึ่ง การประกาศ init ทำได้ดังตารางที่ 2.5

ตารางที่ 2.5 การประกาศ init [1]

คำสั่งภาษาโพรเมลา	ความหมาย
1: init 2: {..... 3: ..... 4: run A(); 5:}	ประกาศ init โดยภายใน init บรรทัดที่ 4 หมายถึงการสั่งเอ็กซิคิวต์ proctype A()

2) ช่องข้อความ (Message channels) ใช้ในการรับส่งข้อมูลระหว่างกระบวนการโดยการเขียนช่องข้อความ นั้นสามารถเขียนตามรูปแบบไวยากรณ์และตัวอย่างรูปแบบการเขียนแสดงในตารางที่ 2.6

ตารางที่ 2.6 รูปแบบไวยากรณ์และแสดงตัวอย่างรูปแบบการเขียนช่องข้อความ [1]

รูปแบบไวยากรณ์	ตัวอย่าง
chan name = '[' const ']' of { typename [, typename]* }	chan a = [16] of { short }

จากตัวอย่างจะเห็นว่าเป็นการเขียนช่องข้อความชื่อ a ที่สามารถเก็บข้อความได้ 16 ช่อง ข้อความมีชนิดข้อมูลเป็น short

3) ดาต้าอ็อบเจกต์ (Data object) คือ ชนิดข้อมูล (Data types) โดยในภาษาโพรเมลานั้นมีชนิดข้อมูลเพียง 9 ชนิด เช่น ชนิดบูลีน ชนิดจำนวนเต็ม ดังแสดงในตารางที่ 2.7

ตารางที่ 2.7 ชนิดข้อมูลของภาษาโพรเมลา [1]

ชนิดของข้อมูล	ค่าของข้อมูล
bit	0,1
bool	true,false
byte	0..255
chan	1..255
mtype	1..255
pid	0.255
short	-215 ..215 -1
int	-231 ..231 -1
unsigned	31 2n -1

4) คอมพาวด์สเตทเมนท์ (Compound statement) คือการรวมคำสั่งหลายๆ คำสั่งเข้าไว้เป็นกลุ่มเดียวกันซึ่งโดยปกติมักรวมคำสั่งเหล่านี้ไว้ในวงเล็บปีกกาเปิดและวงเล็บปีกกาปิดหรือที่เรียกว่าบล็อก (Block) สำหรับในงานวิจัยนี้ผู้วิจัยใช้คอมพาวด์สเตทเมนท์ในภาษาโพรเมลาดังนี้

4.1) คำสั่งลำดับออโตมิกซ์ (Atomic sequence) เป็นการกำหนดให้มีการทำงานตามลำดับภายใต้คำสั่งลำดับออโตมิกซ์ โดยไม่มีการสอดแทรกการทำงานของกระบวนการอื่นตารางที่ 2.8 แสดงตัวอย่างการเขียนคำสั่งลำดับออโตมิกซ์

ตารางที่ 2.8 ตัวอย่างการเขียนคำสั่งลำดับออโตมิกซ์ [1]

คำสั่งลำดับออโตมิกซ์	คำอธิบาย

atomic {
run A ( );
run B ( );
}
	กำหนดให้เอ็กซิคิวต์คำสั่ง “run A( )” ให้เสร็จสิ้นก่อนแล้วค่อยเอ็กซิคิวต์คำสั่ง “run B ( );” ตามลำดับ

4.2) คำสั่งลำดับดีสเตป (d_step) เป็นคำสั่งที่ใช้เป็นการกำหนดให้มีการทำงานตามลำดับภายใต้คำสั่งดีสเตปโดยดีสเตป สามารถเขียนได้ในรูปแบบดังตารางที่ 2.9

ตารางที่ 2.9 ตัวอย่างการเขียนคำสั่งโครงสร้างดีสเตป [1]

คำสั่งลำดับดีสเตป	คำอธิบาย

d_step{
a=b
c=a
}
	เป็นการนำค่าตัวแปร b ให้กับตัวแปร a ก่อนที่จะนำค่าตัวแปร a ให้กับตัวแปร c โดยทำเป็นลำดับ จะไม่สามารถนำค่าตัวแปร a ให้กับตัวแปร c ได้ถ้าไม่ผ่านคำสั่ง a= b

4.3) คำสั่งโครงสร้างทางเลือก (Selections) คือการเลือกทำงานแบบทางใดทางหนึ่งโดยโครงสร้างทางเลือกในภาษาโพรเมลาสามารถเขียนได้ในรูปแบบดังตารางที่ 2.10

ตารางที่ 2.10 ตัวอย่างการเขียนคำสั่งโครงสร้างทางเลือก [1]

คำสั่งโครงสร้างทางเลือก	คำอธิบาย
If :: (a!= b) -> option1; :: (a==b) -> option2; fi	หมายถึงการเลือกทำอย่างใดอย่างหนึ่งหลังประเมินเงื่อนไข เช่น ถ้าเงื่อนไข a!= b เป็นจริง ให้ทำ option 1 และถ้าเงื่อนไข a==b เป็นจริง ให้ทำ option2 เป็นต้น

4.4) คำสั่งข้ามแบบไม่มีเงื่อนไข คือการกระโดดไปกระทำคำสั่งตามที่ได้ระบุไว้หลังสงวน “goto” สามารถแสดงตัวอย่างดังตารางที่ 2.11

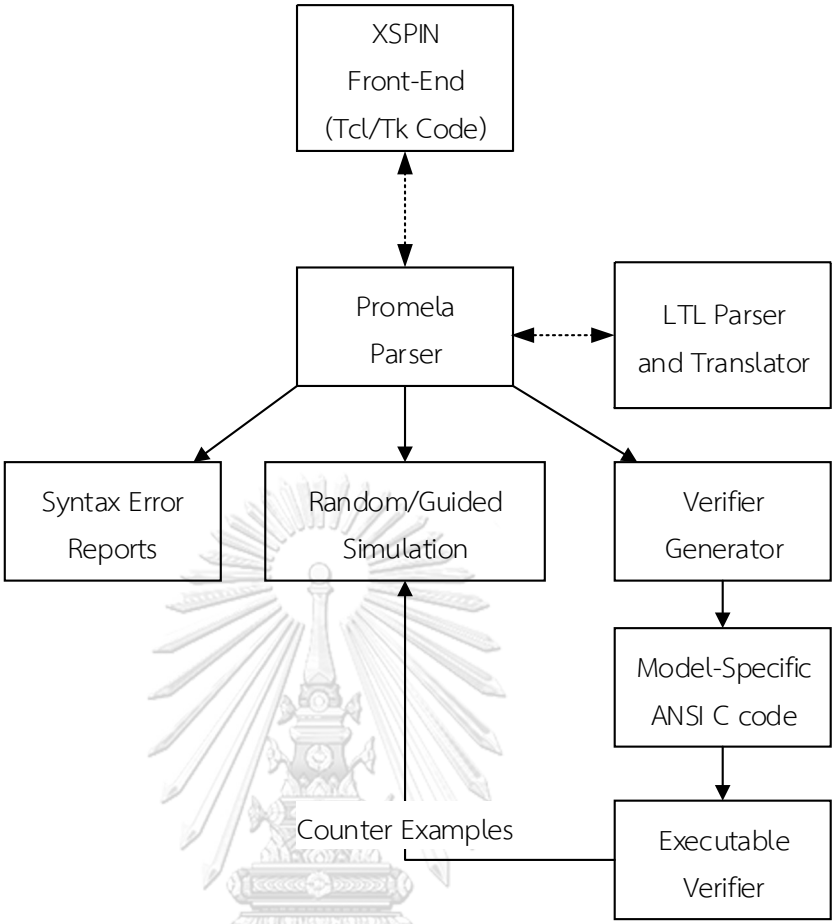
ตารางที่ 2.11 ตัวอย่างการเขียนคำสั่งข้ามแบบไม่มีเงื่อนไข [1]

คำสั่งข้ามแบบไม่มีเงื่อนไข	คำอธิบาย
L1: if :: a != b -> goto L1 :: a == b -> goto L2 fi; L2: a+c	ถ้าเลือกกระทำเงื่อนไขแรกจะข้ามไปทำลาเบล “L1” ถ้าเลือกกระทำเงื่อนไขหลังจะข้ามไปกระทำ ลาเบล “L2”

5) คำสงวน (Reserved keywords) ในภาษาโพรเมลานั้นมีคำสงวนที่ห้ามใช้ในการประกาศเป็นชื่อตัวแปรหรือใช้เป็นชื่อกระบวนการ ดังอย่างคำสงวนในภาษาโพรเมลา ได้แก่ active assert, atomic, bit, bool, break, byte, chan ,d_step, d_proctype, do, else, empty, enabled, fi, full, goto, hidden, if, init, len, mtype, nempty, never, nfull, od, of, pc_value, printf, priority, proctype, provided, run, short, skip, timeout, typedef, unless, unsigned, xr, xs

2.1.4 เครื่องมือสปิน [1]

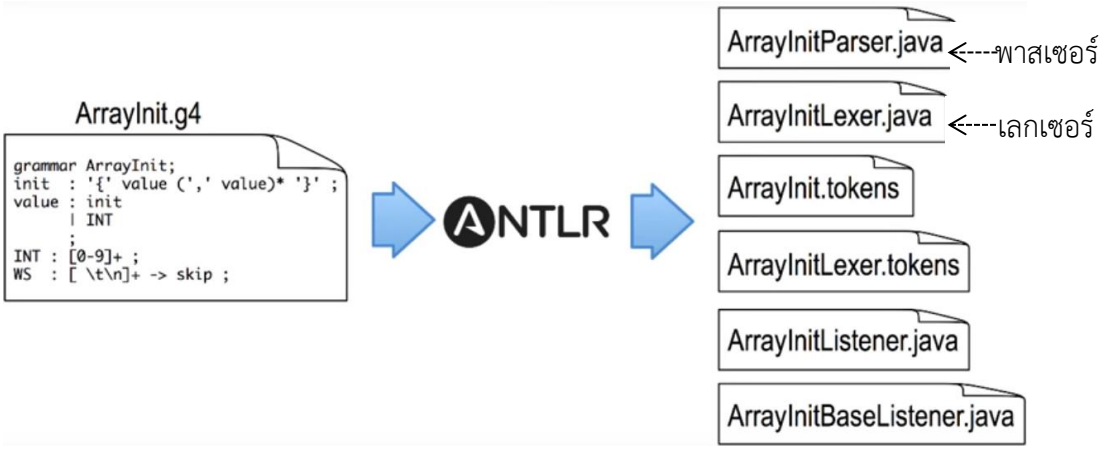
เครื่องมือสปินถูกพัฒนาขึ้น โดย Gerard J. Holmann และทีมเพื่อใช้เป็นเครื่องมือสำหรับการทวนสอบโมเดลของซอฟต์แวร์หรือฮาร์ดแวร์ ซึ่งในการทำการทวนสอบด้วยเครื่องมือสปินนั้น ผู้ใช้เครื่องมือต้องทำการสร้างโมเดลจากพฤติกรรมของระบบที่ต้องการทวนสอบ โดยโมเดลที่สร้างขึ้นต้องเขียนอธิบายให้อยู่ในรูปแบบของภาษาโพรเมลา โดยในส่วนของเครื่องมือสปินนั้นมีองค์ประกอบหลักอยู่ 3 ส่วน ดังภาพที่ 2.7 โดย ส่วนแรกคือ ส่วนของเอกซ์สปิน (XSPIN) ซึ่งเป็นส่วนต่อประสานกับผู้ใช้ (User Interface) ทำหน้าที่ช่วยเหลือผู้ใช้ในการสร้างภาษาโพรเมลา ส่วนที่สองคือส่วนอ่านภาษาโพรเมลาทำหน้าที่อ่านและตรวจสอบโครงสร้างของภาษาโพรเมลาซึ่งในส่วนการอ่านและการตรวจสอบโครงสร้างของภาษาโพรเมลานั้นจะมีการทำงานหลักดังนี้ คือตรวจสอบโครงสร้างของภาษาโพรเมลาแล้วทำรายงานผลข้อผิดพลาด และทำการจำลองการทำงานของภาษาโพรเมลาตามที่ระบุในโมเดลและทำการสร้างโปรแกรมสำหรับตรวจสอบโมเดลดังกล่าวและเมื่อทำการตรวจสอบโมเดลเสร็จสิ้น หากเกิดข้อผิดพลาดขึ้นเครื่องมือสปินจะแสดงข้อผิดพลาดนั้นให้ผู้ใช้ทราบโดยเรียกข้อผิดพลาดนั้นว่า “counter example” และส่วนที่สามคือส่วนของการอ่านและตีความสูตรตรรกะเวลาเชิงเส้น (LTL : Linear temporal logic) โดยส่วนของการอ่านสูตรตรรกะเวลาเชิงเส้น นั้นจะทำหน้าที่อ่านและตีความสูตรตรรกะเวลา เชิงเส้น ซึ่งเครื่องมือสปินจะมีโหมดการทำงานให้เลือกสองโหมดคือ โหมดแบบจำลอง (Simulation) กับโหมดทวนสอบ (Verification) โดยเครื่องมือสปินสามารถทวนสอบคุณสมบัติที่ต้องการได้เช่นคุณสมบัติความปลอดภัย (Safety Property) คุณสมบัติไลฟ์เนส (Liveness Property) โดยคุณสมบัติที่ต้องการทวนสอบด้วยเครื่องมือสปินนั้นจะถูกเขียนอธิบายด้วยสูตรตรรกะเวลาเชิงเส้น



ภาพที่ 2.7 โครงสร้างของเครื่องมือสปิน [1]

2.1.5 แอนท์เลอร์ [7]

แอนท์เลอร์คือ เครื่องมือที่ใช้สำหรับสร้างส่วนของการแยกแยะสัญลักษณ์ (Lexer-Lexical analyzer) หรือเลกเซอร์และส่วนของการตีความหมายของภาษาหรือที่เรียกว่าพาสเซอร์ (Parser) [5] ที่มีประสิทธิภาพ ซึ่งในการที่จะใช้แอนท์เลอร์สำหรับสร้างส่วนของเลกเซอร์กับพาสเซอร์นั้น ผู้ใช้จำเป็นต้องมีรายละเอียดของภาษาที่ต้องการจะสร้างเลกเซอร์กับพาสเซอร์และรายละเอียดของภาษานั้นต้องอยู่ในรูปแบบภาษารูปนัย มาเป็นข้อมูลนำเข้าให้แอนท์เลอร์ โดยแอนท์เลอร์จะเรียกภาษาทางการนี้ว่าแกรมมาและเมื่อนำแกรมมาเข้าสู่แอนท์เลอร์แล้ว แอนท์เลอร์ก็จะทำการสร้างเลกเซอร์กับพาสเซอร์ให้อย่างอัตโนมัติและสามารถนำไปใช้งานได้อย่างง่ายดาย ซึ่งจากภาพที่ 2.8 แสดงถึงการรับข้อมูลนำเข้าเป็นไวยากรณ์ของอาร์เรย์ โดยแอนท์เลอร์จะทำการสร้างเลกเซอร์และพาสเซอร์ให้โดยจากภาพที่ 2.8 ได้คลาสชื่อ ArrayInitParser.java และ ArrayInitLexsr.java ตามลำดับ



ภาพที่ 2.8 ตัวอย่างการใช้ไวยากรณ์ของอาร์เรย์เพื่อสร้างเลกเซอร์และพาสเซอร์จากแอนท์เลอร์ [7]

2.2 งานวิจัยที่เกี่ยวข้อง

2.2.1 งานวิจัยชื่อ Model Checking UML State Machines and Collaborations โดย Alexander Knapp ปี ค.ศ. 2001. [2]

งานวิจัยนี้เป็นการตรวจสอบความต้องกันของแผนภาพสเตทแมชชีนและแผนภาพคอลลาโบเรชัน (Collaborations Diagram) ซึ่งในงานวิจัยนี้ได้แสดงให้เห็นถึงการแปลงแผนภาพสเตทแมชชีนเป็นภาษาโพรเมลาโดยผู้วิจัยใช้เครื่องมือชื่อ “HUGO” ซึ่งเป็นเครื่องมือที่ได้พัฒนาขึ้น โดยในงานวิจัยนี้แสดงให้เห็นถึงการแปลงสถานะต่างๆ บนแผนภาพสเตทแมชชีนเก็บไว้ในตัวแปรอาร์เรย์และกำหนดให้ทุกๆ สถานะมีชุดคำสั่งภาษาโพรเมลาที่ใช้สำหรับจัดการกับเหตุการณ์ต่างๆ ที่เกิดขึ้นกับสถานะแต่ละสถานะ งานวิจัยนี้มุ่งเน้นว่าทุกสถานะบนแผนภาพจะต้องถูกเอ็กซิคิวต์ได้แต่ไม่สนใจการแปลงรายละเอียดภายในแผนภาพสเตทแมชชีน สำหรับในขั้นตอนการทวนสอบของงานวิจัยนี้ได้ใช้ข้อมูลจากแผนภาพคอลลาโบเรชันมาใช้ในการเขียนสูตรตรรกะเวลาเชิงเส้น เพื่อทวนสอบความต้องกันของแผนภาพดังกล่าว แต่ทั้งนี้ในงานวิจัยนี้ไม่ได้แสดงให้เห็นถึงกฎการแปลงของเครื่องมืออย่างละเอียดและภาษาโพรเมลาที่ได้จากการแปลงไม่มีรายละเอียดของโอเปอเรชันต่างๆ ของแต่ละสถานะของแผนภาพสเตทแมชชีน

งานวิจัยนี้มีประโยชน์สำหรับการศึกษาแนวทางในการแปลงแผนภาพสเตทแมชชีนเป็นภาษาโพรเมลา นอกจากนั้นใช้ประโยชน์ จากงานวิจัยนี้ในการกำหนดกฎการแปลงดังแสดงในแม่แบบโครงสร้างภาษาโพรเมลา ว่าด้วยเรื่องการประกาศตัวแปรของแต่ละสถานะเพื่อใช้ควบคุมลำดับการเอ็กซิคิวต์ของภาษาโพรเมลา

2.2.2 งานวิจัยชื่อ Automatic Verification of a Behavioural Subset of UML Statechart Diagrams Using the SPIN Model-checker โดย Diego Latella ปี ค.ศ. 1999. [8]

งานวิจัยนี้เป็นการทวนสอบพฤติกรรมของคอมโพสิตสเตทของแผนภาพสเตทแมชชีนโดยใช้เครื่องมือสปิน ซึ่งในงานวิจัยนี้ได้แสดงให้เห็นถึงการแปลงคอมโพสิตสเตทของแผนภาพสเตทแมชชีนไปเป็นภาษาโพรเมลาโดยพิจารณาจากจำนวนสถานะทั้งหมดและเหตุการณ์ที่ทำให้มีการเปลี่ยนจากสถานะหนึ่งไปอีกสถานะหนึ่งโดยเน้นการทวนสอบว่าทุกๆ สถานะในแผนภาพสเตทแมชชีนต้องสามารถทำการ เอ็กซิคิวต์ได้ทุกสถานะโดยไม่สนใจรายละเอียดของสถานะนั้นๆ ซึ่งในการแปลงผู้วิจัยได้ออกแบบกฎการแปลงดังนี้คือเหตุการณ์ที่เกิดขึ้นบนแผนภาพแล้วทำการแปลงให้อยู่ในรูปภาษาโพรเมลาดังเช่น

E = {e1, . . . , en} คือเซตของเหตุการณ์ซึ่งภาษาโพรเมลาที่ได้คือ

#define e1 1

.....

.....

#define en n

กำหนดลำดับของเหตุการณ์ที่เกิดขึ้นโดยการแทนด้วยเซตของตัวแปรบิต (Bit variable) โดยในงานวิจัยนี้กำหนดลำดับของเหตุการณ์ด้วยการประกาศตัวแปรชื่อ “Q” ตามด้วยชื่อเหตุการณ์ ซึ่งจะได้ภาษาโพรเมลาดังนี้ bit Qe1,...,Qen; สถานะแต่ละสถานะบนแผนภาพแทนด้วยเซตของ ตัวแปรบิต เช่น bit S1, . . Sz; ซึ่งหลังจากได้โพรเมลาดังตัวอย่างจากการใช้กฎข้อ 1 ถึงข้อ 3 แล้วนั้นในลำดับต่อไปเป็นการสร้างกระบวนการภาษาโพรเมลา ซึ่งในงานวิจัยไม่ได้ระบุกฎที่ชัดเจนในการแปลงสัญลักษณ์ต่างๆ ของแผนภาพสเตทแมชชีนไปเป็นภาษาโพรเมลาและพบว่าผลลัพธ์ภาษาโพรเมลาที่ได้เป็นลำดับของการเปลี่ยนสถานะหนึ่งไปอีกสถานะหนึ่งซึ่งยังไม่เพียงพอต่อการนำไปใช้ในการทวนสอบเนื่องจากการเพิ่มรายละเอียดลงไปในภาษาโพรเมลานั้นค่อนข้างยาก และไม่มีรายละเอียดของเงื่อนไขก่อน เงื่อนไขหลังและค่ายืนยง

งานวิจัยนี้มีประโยชน์สำหรับการศึกษาแนวทางในการแปลงแผนภาพสเตทแมชชีนเป็นภาษาโพรเมลาโดยศึกษาลำดับของการเปลี่ยนสถานะบนแผนภาพและการทำการเอ็กซิคิวต์สถานะแต่ละสถานะ รวมทั้งศึกษาผลลัพธ์ที่ได้จากการแปลงเพื่อหาแนวทางในการแปลงให้ได้ผลลัพธ์ที่สมบูรณ์มากพอสำหรับนำไปใช้ในการทวนสอบด้วยเครื่องมือสปิน

2.2.3 งานวิจัยชื่อ Implementing statecharts in PROMELA/SPIN โดย Erich Mikk ปี ค.ศ. 1997. [9]

ในงานวิจัยนี้ได้ใช้โมเดลอีเอชเอ (EHA: Extended hierarchical automata) ของงานวิจัย [10] เป็นตัวกลางในการแปลงแผนภาพสเตทแมชชีนเป็นภาษาโพรเมลาโดยที่อีเอชเอ จะประกอบไปด้วยเซ็ตของลำดับออโตมาตา (Equential automata) ที่ประกอบไปด้วยเซตของสถานะและเซตของทรานซิชันลาเบล (Transition labels) ซึ่งสามารถแสดงให้เห็นถึงการเปลี่ยนสถานะ โดยในงานวิจัยได้ทำการแปลงอีเอชเอไปเป็นภาษาโพรลาได้โครงสร้างทางเลือกเท่านั้นและไม่สามารถแปลงแอคชันและตัวแปรต่างๆ บนแผนภาพได้

งานวิจัยนี้มีประโยชน์สำหรับการศึกษาแนวทางในการแปลงแผนภาพสเตทแมชชีนเป็นภาษาโพรเมลาและศึกษาผลลัพธ์ที่ได้จากการแปลงเพื่อหาแนวทางในการแปลงให้ได้ผลลัพธ์ที่สมบูรณ์มากพอสำหรับนำไปใช้ในการทวนสอบด้วยเครื่องมือสปิน

2.2.4 งานวิจัยชื่อ Formalising UML state machines for model checking โดย Johan Lilius ปี ค.ศ. 1999. [11]

งานวิจัยนี้เป็นการเสนอวิธีการเขียนแผนภาพสเตทแมชชีนให้อยู่ในรูปแบบจำลองเชิงรูปนัยเพื่อนำมาใช้ประโยชน์ในการแปลงแผนภาพสเตทแมชชีนเป็นภาษาโพรเมลาซึ่งเป็นส่วนหนึ่งของการพัฒนาเครื่องมือ “vUML ” [3] โดยในงานวิจัยนี้มีสองส่วนการทำงานคือ ส่วนของการกำหนดโครงสร้างของแผนภาพสเตทแมชชีนและส่วนกำหนดความหมายของรายละเอียดของโอเปอเรชันบนแผนภาพเพื่อใช้ในการแปลงรายละเอียดของพฤติกรรมในแผนภาพสเตทแมชชีนเป็นภาษาโพรเมลา

งานวิจัยนี้มีประโยชน์สำหรับการศึกษาแนวทางในการแปลงแผนภาพสเตทแมชชีนเป็นภาษาโพรเมลาเพื่อหาแนวทางในการแปลงให้ได้ผลลัพธ์ที่สมบูรณ์มากพอสำหรับนำไปใช้ในการทวนสอบด้วยเครื่องมือสปิน

จากการศึกษางานวิจัยที่เกี่ยวข้องพบว่าในการแปลงแผนภาพสเตทแมชชีนเป็นภาษาโพรเมลานั้น แต่ละงานวิจัยมีข้อสังเกตที่เหมือนกันคือผู้วิจัยไม่ได้แสดงให้เห็นกฎในการแปลงสัญลักษณ์ของแผนภาพสเตทแมชชีนแต่ละสัญลักษณ์เป็นองค์ประกอบของภาษาโพรเมลาและผลลัพธ์ของภาษาโพรเมลาที่ได้ไม่มีรายละเอียดของเงื่อนไขก่อน เงื่อนไขหลังและค่ายืนยง

บทที่ 3
แนวคิดและรายละเอียดในการแปลง

ในบทนี้จะกล่าวถึงแนวคิดการแปลงแผนภาพสเตทแมชชีนที่มีการเขียนเงื่อนไขก่อน เงื่อนไขหลังและค่ายืนยงในรูปแบบของโอซีแอล การรวมองค์ประกอบของภาษาโพรเมลาที่ได้จากการแปลงและรายละเอียดของการแปลงแผนภาพสเตทแมชชีนเป็นภาษาโพรเมลา และจะกล่าวถึงขั้นตอนการเพิ่มรายละเอียดอื่นๆ ในภาษาโพรเมลาที่ได้จากการแปลงสำหรับกรณีที่ผู้ใช้งานต้องการเพิ่มข้อมูลอื่นๆ ลงไปในภาษาโพรเมลา

3.1 แนวคิดและรายละเอียดในการแปลง

เนื่องจากแผนภาพสเตทแมชชีนเป็นแผนภาพที่ใช้อธิบายถึงพฤติกรรมของระบบ โดยแผนภาพสเตทแมชชีนจะประกอบไปด้วย สถานะ การเปลี่ยนสถานะ และลาเบลที่ปรากฏบนเส้นการเปลี่ยนสถานะ โดยลาเบลดังกล่าวจะเขียนในรูปแบบ “เหตุการณ์ [เงื่อนไขการ์ด]/แอคชัน” ซึ่งการทำงานของแผนภาพสเตทแชชีนจะเริ่มการทำงานจากสถานะเริ่มต้นและมีการเปลี่ยนสถานะไปเรื่อยๆ จนถึงสถานะสิ้นสุด ซึ่งเป็นจุดจบการทำงานของแผนภาพสเตทแมชชีน ในระหว่างที่แผนภาพสเตทแมชชีนทำการเปลี่ยนสถานะนั้น ก่อนที่จะทำการเปลี่ยนสถานะต้องมีการตรวจสอบลาเบล ที่ปรากฏบนเส้นการเปลี่ยนสถานะก่อนที่จะทำการเปลี่ยนสถานะเสมอ

ส่วนภาษาโพรเมลาเป็นภาษาที่ใช้สำหรับการจำลองพฤติกรรมของระบบโดยภาษาโพรเมลาประกอบด้วยส่วนของ proctype, init และตัวแปรภาษาโพรเมลา โดย proctype แทนพฤติกรรมของระบบ และ init เป็นจุดเริ่มต้นการทำงานของภาษาโพรเมลา ดังนั้นเมื่อนำแผนภาพสเตทแมชชีนมาเทียบเคียงกับภาษาโพรเมลาจะพบว่าสามารถแทน สถานะ ด้วย proctype และแทนสถานะเริ่มต้นด้วย init ได้และในส่วนของการเปลี่ยนสถานะนั้นจะแทนด้วยชุดคำสั่งภาษาโพรเมลาซึ่งเป็นชุดคำสั่งที่ใช้สำหรับตรวจสอบการเปลี่ยนสถานะ โดยชุดคำสั่งดังกล่าวจะมีส่วนของเงื่อนไขการ์ด หรือเหตุการณ์รวมอยู่ด้วยซึ่งจะอธิบายในรายละเอียดของกฎการแปลงที่จะกล่าวในลำดับถัดไป ส่วนแอคชันที่ปรากฏบนแผนภาพสเตทแมชชีนนั้นจะทำการแปลงเป็น Dummy proctype เนื่องจากแอคชันหมายถึง กิจกรรมที่ทำระหว่างการเปลี่ยนสถานะซึ่งเมื่อเทียบกับส่วนประกอบของภาษาโพรเมลาแล้วสามารถแทนได้ด้วย proctype และสำหรับส่วนของตัวแปรต่างๆ ที่ปรากฏอยู่บนแผนภาพสเตทแมชชีนเช่น ตัวแปรที่เกิดจาก เหตุการณ์ เงื่อนไขการ์ด เงื่อนไขก่อน เงื่อนไขหลังและค่ายืนยงจะถูกประกาศเป็นตัวแปรในภาษาโพรเมลาโดยในงานวิจัยนี้จะประกาศ

ตัวแปรเป็น 2 ชนิดคือบูลีนและจำนวนเต็มเท่านั้นเนื่องจากบนแผนภาพสเตทแมชชีนและหลักการเขียนโอซีแอลไม่ได้แสดงถึงชนิดของข้อมูล

ในงานวิจัยนี้จะมีกฎการแปลง 6 ข้อ สำหรับแปลงสัญลักษณ์ของแผนภาพสเตทแมชชีนไปเป็นภาษาโพรเมลา โดยในแต่ละภาษาโพรเมลาจะมีแม่แบบโครงสร้างของภาษาโพรเมลาที่สอดคล้องกันโดยมีรายละเอียดของกฎแต่ละข้อดังนี้

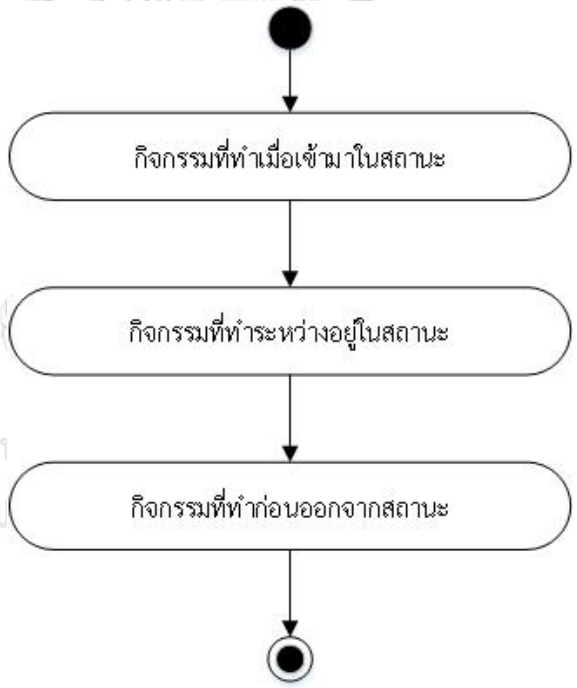
1) กฎการแปลงสัญลักษณ์สถานะเป็น proctype ในภาษาโพรเมลาโดย 1 สถานะเท่ากับ 1 proctype ซึ่งจากการทำงานของสัญลักษณ์สถานะแต่ละสถานะนั้นมีกิจกรรมภายในดังนี้คือ กิจกรรมที่ทำเมื่อเข้ามาในสถานะ (Entry Activity) กิจกรรมที่ทำในระหว่างอยู่ในสถานะ (Do activity) และกิจกรรมที่ทำก่อนออกจากสถานะ (Exit Activity) [4] ดังภาพที่ 3.1 แสดงถึงลำดับกิจกรรมในสถานะ โดยในการแปลงสถานะเป็น 1 proctype โดยจะมีแม่แบบภาษาโพรเมลาในการแปลง ดังภาพที่ 3.2 โดยมีรายละเอียดดังนี้ st_i คือ ชื่อสถานะ และจะปรากฏชื่อสถานะเป็นส่วนประกอบในแม่แบบภาษาโพรเมลา ซึ่งจากภาพจะเห็นว่าแม่แบบประกอบด้วย 2 ส่วน

- ส่วนแรกเป็นส่วนการประกาศตัวแปรควบคุมสถานะ เป็นการประกาศตัวแปรที่ใช้สำหรับการทำงานของ proctype ในภาษาโพรเมลา เช่น ตัวแปร mtype stateStatus$\boldsymbol{st_i}$ เป็นต้น โดยตัวแปรดังกล่าวเปรียบเสมือนตัวแปรที่คอยควบคุม การสั่งเอ็กซิคิวต์ proctype นั้นๆ เนื่องจากในการทำงานของเครื่องมือสปินเมื่อมีการสั่งเอ็กซิคิวต์ภาษาโพรเมลา สปินจะทำงานแบบพร้อมกัน (concurrent) ทำให้ลำดับในการสั่งเอ็กซิคิวต์คำสั่งโพรเมลาแต่ละบรรทัด ไม่ได้เรียงตามภาษาอื่นทั่วไป ผู้วิจัยจึงจำเป็นต้องกำหนดตัวแปรในส่วนนี้เพื่อควบคุมให้เครื่องมือสปินสั่งเอ็กซิคิวต์ proctype ที่ได้จากการแปลงให้สอดคล้องกับการเปลี่ยนสถานะของแผนภาพสเตทแมชชีน
- ส่วนที่ 2 คือส่วนของการแปลงสัญลักษณ์สถานะเป็น proctype โดยนำชื่อ สถานะมาเป็นส่วนประกอบของชื่อ proctype เช่นภาพ ที่ 3.2 ส่วนแม่แบบภาษาโพรเมลามี proctype ชื่อ “State$\boldsymbol{st_i}$” ซึ่งภายใน proctype State$\boldsymbol{st_i}$ จะประกอบด้วย 3 ส่วนหลักดังนี้
    a. ส่วนของเงื่อนไขก่อน เทียบได้กับส่วนของกิจกรรมที่ต้องทำเมื่อเข้ามาในสถานะ

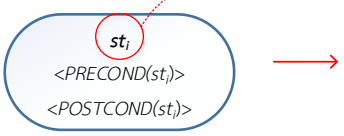
b. ส่วนของตัวดำเนินการ เทียบได้กับกิจกรรมที่ทำในระหว่างอยู่ในสถานะ

c. ส่วนของเงื่อนไขหลัง เทียบได้กับกิจกรรมที่ทำก่อนออกจากสถานะ

จากภาพที่ 3.2 จะเห็นว่าในส่วนของเงื่อนไขก่อนและส่วนของเงื่อนไขหลังมีการตรวจสอบเงื่อนไขด้วยโครงสร้างทางเลือก IF THEN ELSE โดยทุกๆ โครงสร้าง IF THEN ELSE ที่ปรากฏจะให้มีการตรวจสอบเงื่อนไขที่เป็นจริงเพียงเงื่อนไขเดียว เนื่องมาจากในการทำงานของโครงสร้างทางเลือกในภาษาโพรเมลานั้นการเลือกทำ จะเลือกเพียงแค่ทางเลือกเดียว โดยเลือกทำแบบการสุ่ม ดังนั้นเพื่อให้มีการตรวจสอบเงื่อนไขทุกเงื่อนไขและเพื่อป้องกันการเกิด Time out เนื่องจากกรณีที่สปินตรวจสอบเงื่อนไขในโครงสร้างทางเลือกแล้วพบว่าไม่มีเงื่อนไขใดเป็นจริง งานวิจัยนี้จึงออกแบบให้มีการเขียนโครงสร้างทางเลือกที่อยู่ในรูปแบบ “IF เงื่อนไข THEN ทำคำสั่งที่อยู่ภายใต้เงื่อนไขที่เป็นจริง ELSE print case else “



ภาพที่ 3.1 กิจกรรมการทำงานของสัญลักษณ์สถานะ



สัญลักษณ์สเตทแมชชีน

1

```
/*ประกาศ ประเภทข้อมูล mtype ให้มีค่าได้ 3 ค่า คือ Idel,
  running, done*/
1: mtype = {idle,running,done};

/*ประกาศตัวแปรชนิด mtype stateStatus st_i ไว้ควบคุมสถานะโดย
  ให้ค่าเริ่มต้นเท่า กับ Idel โดยหลักการตั้งชื่อให้ขึ้นต้นด้วยคำว่า
  stateStatus ตามด้วยชื่อสถานะ */
2: mtype stateStatus st_i = idle;

/*ประกาศตัวแปรชนิดบูลีนให้ค่าเริ่มต้นเป็นเท็จไว้สำหรับตรวจสอบ
  การผ่านและไม่ผ่านเงื่อนไขก่อนและเงื่อนไขหลังโดยหลักการตั้งชื่อให้
  ขึ้นต้นด้วยคำว่า precFail ตามด้วยชื่อสถานะกรณีเป็น เงื่อนไขก่อน
  และต้นด้วยคำว่า postFail ตามด้วยชื่อสถานะกรณีเป็น
  เงื่อนไขหลัง*/
3: bool precFail st_i = false, postFail st_i = false;
```

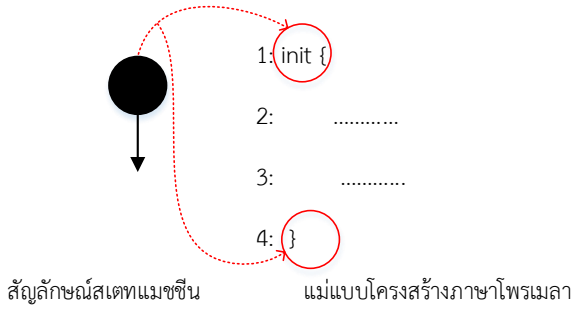
2

```
4: proctype State st_i () /*นำชื่อสถานะมาเป็นชื่อ proctype โดยหลักการตั้งชื่อให้
                           ขึ้นต้น ด้วย State ตามด้วยชื่อสถานะ*/
5: {
6:   d_step{ /*ภายใน proctype ครอบด้วยคำสั่ง d_step เพื่อเป็นการควบคุม
              ให้คำสั่งภายในปีกกาเปิดและปีกกาถูกเอ็กซิคิวต์ตามลำดับ*/
7:   stateStatus st_i = running; /*ให้ตัวแปร stateStatus st_i มีค่าเท่ากับ running*/

8:   precon:  /*ส่วนของเงื่อนไขก่อน*/
9:     if PRECOND(st_i) == true /*ตรวจสอบเงื่อนไขก่อนเท่ากับจริง โดยนำค่า
                                ของเงื่อนไขก่อนมาจากเงื่อนไขก่อนที่ปรากฏในสถานะ */
10:      then goto st_iOperation; /*ข้ามไปส่วนดำเนินการ*/

11:    else  print "case else" /*พริ้น case else*/

12:    if !PRECOND(st_i) == true /*ตรวจสอบเงื่อนไขก่อนเท่ากับเท็จ */

13:      then precFail st_i = true; Terminate =1; /*ให้ตัวแปรprecFail st_i
                เท่ากับจริง เมื่อเงื่อนไขก่อนเป็นเท็จและให้ Terminate เท่ากับ 1
                เพื่อให้สิ้นหยุดการเอ็กซิคิวต์ */

14:    else  print "case else"/*พริ้น case else*/

15:  st_iOperation: /*ส่วนดำเนินการหลักการตั้งชื่อโดยลาเบล คือนำชื่อสถานะ
                     ตาม ด้วยคำว่า Operation*/

16:        atomic {  /*คำสั่งลำดับ atomic*/

17:          stateStatus st_i = running; /*กำหนดให้ตัว
                                แปรstateStatus st_i เท่ากับ running*/

18:           /* ...Fill in details of operation..... */

19:              goto poscon; /*ข้ามไปทำส่วนของเงื่อนไขหลัง*/

20:           } /*end atomic*/

21:  postcon:/*ส่วนของเงื่อนไขหลัง*/

22:    if  POSTCOND(st_i)== true /*ตรวจสอบเงื่อนไขหลังเท่ากับจริง โดยนำ
                                  ค่าของเงื่อนไขหลังมาจากเงื่อนไขหลังที่ปรากฏในสถานะ */

23:      then stateStatus st_i = done;/*กำหนดให้ตัวแปร stateStatus st_i
                                  เท่ากับ done*/

24:    else  print "case else" /*พริ้น case else*/

25:    if  !POSTCOND(st_i)== true /*ตรวจสอบเงื่อนไขหลังเท่ากับเท็จ */

26:      then postFail st_i = true; Terminate =1;/*ให้ตัวแปรpostFail st_i
                เท่ากับ จริง เมื่อเงื่อนไขหลังเป็นเท็จและให้ Terminate เท่ากับ 1
                เพื่อให้สิ้นหยุดการเอ็กซิคิวต์ */

27:    else  print "case else" /*พริ้น case else*/

28:   }/*end d_step*/

29:} /*end proctype*/
```

แม่แบบโครงสร้างภาษาโพรเมลา

ภาพที่ 3.2 แม่แบบโครงสร้างภาษาโพรเมลาส่วนของการแปลงสัญลักษณ์สถานะ

2) การแปลงสัญลักษณ์เริ่มต้น แนวคิดการแปลงสัญลักษณ์เริ่มต้นจะทำการแปลงเป็นกรอบของ init ซึ่งเปรียบเหมือนจุดเริ่มต้นการทำงานของภาษาโพรเมลา ดังแสดงในภาพที่ 3.3



ภาพที่ 3.3 แม่แบบโครงสร้างภาษาโพรเมลาส่วนของการแปลงสัญลักษณ์เริ่มต้น

3) การแปลงสัญลักษณ์สิ้นสุดนั้นจะแปลงเป็น 3 proctype คือ

- *FinalState*
- *checkInvariantNormalTerminate*
- *checkInvariantAbNormalTerminate*

โดยแต่ละ proctype มีที่มาดังนี้ proctype *FinalState* คือ proctype ที่ทำหน้าที่แทนจุดจบของแผนภาพสเตทแมชชีนบนภาษาโพรเมลา โดยถ้าเครื่องมือสปินมีการเอ็กซิคิวต์ proctype นี้จะทำการเปลี่ยนค่าตัวแปร CompleteFinalState เท่ากับ 1 เพื่อเป็นจุดสิ้นสุดของโปรแกรม ส่วน proctype, *checkInvariantNormalTerminate* และ *checkInvariantAbNormalTerminate* ทำหน้าที่เปรียบเสมือนตัวเฝ้าสังเกต (Monitor) ค่าของตัวแปร CompleteFinalState และ Terminate โดยภายในจะเห็นว่ามีการใช้คำสั่ง assert เพื่อ assert ค่าตัวแปรดังกล่าว ถ้าพบว่าค่าของตัวแปรดังกล่าวเปลี่ยนไปเท่ากับ 1 ณ ช่วงเวลาใดเครื่องมือสปินจะหยุดการเอ็กซิคิวต์ภาษาโพรเมลา โดยคำสั่ง assert ในภาษาโพรเมลานั้นใช้สำหรับตรวจสอบดูค่าตัวแปรซึ่งหากค่าตัวแปรที่ทำการ assert เป็นเท็จสปินจะหยุดการเอ็กซิคิวต์ภาษาโพรเมลา ภาพที่ 3.4 แสดงถึงแม่แบบโครงสร้างภาษาโพรเมลาส่วนของการแปลงสัญลักษณ์สิ้นสุด

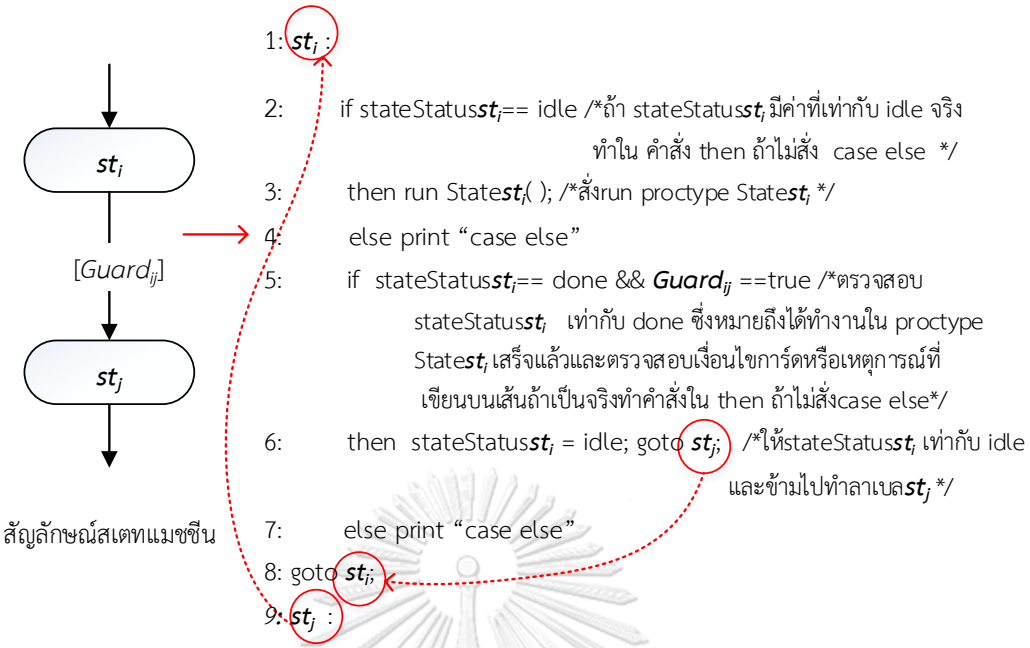
สัญลักษณ์สเตทแมชชีน

```
1: bit CompleteFinalState = 0;
2: bit Terminate = 0;
3: proctype FinalState( )
4: {  CompleteFinalState = 1; }
5: active protype checkInvariantNormalTerminate ()
6:{/*วน Loop ตรวจสอบค่าตัวแปรCompleteFinalState
     ถ้าไม่เท่ากับ 0  ให้สปินหยุดการทำงาน */
7: do
8: ::assert(CompleteFinalState ==0)
9: od;
10:}
11:active proctype checkInvariantAbNormalTerminate()
12:{/*วน Loop ตรวจสอบค่าตัวแปรTerminateถ้าไม่เท่ากับ 0
      ให้สปินหยุดการทำงาน */
13: do
14: ::assert(Terminate ==0)
15:od;
16:}
```

แม่แบบโครงสร้างภาษาโพรเมลา

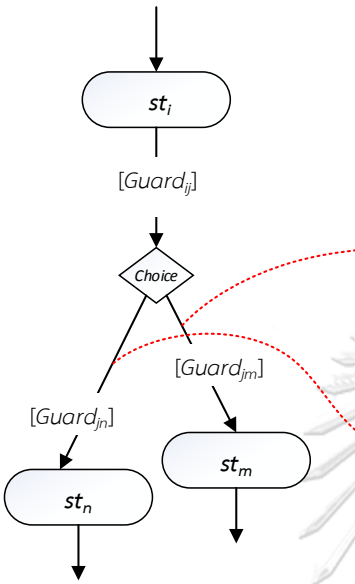
ภาพที่ 3.4 แม่แบบโครงสร้างภาษาโพรเมลาส่วนของการแปลงสัญลักษณ์สิ้นสุด

4) การแปลงสัญลักษณ์การเปลี่ยนสถานะนั้นจะแปลงเป็นโครงสร้างภาษาโพรเมลาดังภาพที่ 3.5 โดยมีแนวคิดในการแปลงมาจากการทำงานของแผนภาพสเตทแมชชีนทำงานโดยการเปลี่ยนสถานะจากสถานะหนึ่งไปอีกสถานะหนึ่งซึ่งการเปลี่ยนสถานะได้นั้นต้องทำกิจกรรมภายในสถานะนั้นๆเสร็จสิ้นและมีการตรวจสอบเงื่อนไขบางอย่างที่ปรากฏบนเส้นการเปลี่ยนสถานะจึงสามารถเปลี่ยนสถานะได้ และจากการทำงานดังกล่าวจึงมีการกำหนดให้มีการตรวจสอบเงื่อนไขดังกล่าว โดยโครงสร้าง IF THEN ELSE แรกเป็นการตรวจสอบว่า มีการสั่งเอ็กซิคิวต์ proctype ที่แทนสถานะก่อนที่จะมีการเปลี่ยนสถานะแล้วหรือไม่ ถ้ายังให้สั่งเอ็กซิคิวต์ proctype นั้นด้วยคำสั่ง run ตามด้วยชื่อ proctype ดังบรรทัดที่ 3 ในภาพที่ 3.5 และสำหรับ IF THEN ELSE ส่วนที่ 2 เป็นการตรวจสอบว่าได้ทำการเอ็กซิคิวต์ proctype ที่แทนสถานะก่อนที่จะเปลี่ยนสถานะเสร็จแล้วหรือไม่ ถ้าเสร็จให้ตรวจสอบเงื่อนไขการ์ดหรือเหตุการณ์ที่อยู่บนเส้นเปลี่ยนสถานะถ้าเป็นจริงให้ทำการเปลี่ยนสถานะ ซึ่งจากภาษาโพรเมลาที่ได้ผู้วิจัยใช้คำสั่งข้ามแบบไม่มีเงื่อนไข



ภาพที่ 3.5 แม่แบบโครงสร้างภาษาโพรเมลาส่วนของการแปลงสัญลักษณ์การเปลี่ยนสถานะ

5) การแปลงสัญลักษณ์ทางเลือกที่มีเส้นการเปลี่ยนสถานะเข้า 1 เส้นและออก n เส้น โดยจะแปลงเป็นโครงสร้าง IF THEN ELSE ภาพที่ 3.6 โดยเมื่อเจอสัญลักษณ์ทางเลือกให้แปลงเป็น โครงสร้างคำสั่ง IF THEN ELSE ตามรูปแบบ บรรทัดที่ 7 ถึงบรรทัดที่ 9 และบรรทัดที่ 10 ถึงบรรทัดที่ 12 โดยจำนวนโครงสร้าง IF THEN ELSE ที่ปรากฏจะเท่ากับจำนวนสัญลักษณ์การเปลี่ยนสถานะที่หัวลูกศรพุ่งออกจากสัญลักษณ์ทางเลือก โดยแนวคิดมาจากการมองสัญลักษณ์ทางเลือกเป็นสถานะเทียมที่อยู่ระหว่างสัญลักษณ์การเปลี่ยนสถานะโดยสนใจเส้นการเปลี่ยนสถานะออกจากสัญลักษณ์ทางเลือกและนำเงื่อนไขการ์ดหรือเหตุการณ์ที่ปรากฏบนเส้นสัญลักษณ์การเปลี่ยนมาเป็นเงื่อนไข เพื่อพิจารณาว่าจะเปลี่ยนสถานะไปที่สถานะใด



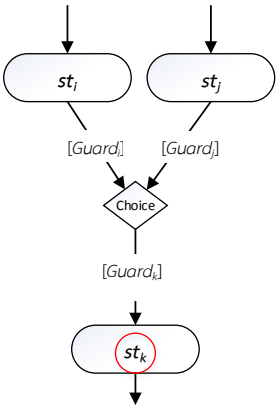
สัญลักษณ์สเตทแมชชีน

```
1: sti  :
2:      if stateStatusSti== idle  /*ถ้า stateStatusSti มีค่าที่เท่ากับ idle
                  เป็นจริงทำในคำสั่ง thenถ้าไม่สั่ง  case else  */
3:      then run stateSti();/*สั่งrun proctype StateSti */
4:      else print “case else”
5:      if  stateStatusSti== done && Guardij ==true /*ตรวจสอบ
            stateStatusSti  เท่ากับ done ซึ่งหมายถึงได้ทำงานใน
            proctype StateSti เสร็จแล้ว และตรวจสอบเงื่อนไขการ์ด
            หรือเหตุการณ์ที่เขียนบนเส้นถ้าเป็นจริงทำคำสั่งใน then
            ถ้าไม่สั่งcase else  */
6:      then
7:          if Guardjm ==true /*ตรวจสอบเงื่อนไขการ์ดหรือ
                  เหตุการณ์ที่ปรากฏบนเส้นถ้าเป็นจริงทำคำสั่ง
                  ใน then*/
8:          then stateStatusSti=idle; goto  stm
9:          else print “case else”
10:         if Guardjn ==true  /*ตรวจสอบเงื่อนไขการ์ดหรือ
                  เหตุการณ์ที่ปรากฏบนเส้นถ้าเป็นจริงทำคำสั่ง
                  ใน then */
11:         then stateStatusSti=idle; goto  stn
12:         else print “case else”
13:     else print “case else”
14: goto sti ;
15: stn :
16:     :
17: stm
18:     :
```

แม่แบบโครงสร้างภาษาโพรเมลา

ภาพที่ 3.6 แม่แบบโครงสร้างภาษาโพรเมลาส่วนของการแปลงสัญลักษณ์สัญลักษณ์ทางเลือกที่มีเส้นการเปลี่ยนสถานะเข้า 1 เส้นและออก n เส้น

6) การแปลงสัญลักษณ์ทางเลือกที่มีเส้นการเปลี่ยนสถานะเข้า n เส้นและออก 1 เส้น ทำโดยให้แปลงเป็นโครงสร้างคำสั่ง IF THEN ELSE ตามรูปแบบบรรทัดที่ 7 ถึงบรรทัดที่ 9 และบรรทัดที่ 18 ถึงบรรทัดที่ 20 ดังแสดงในภาพที่ 3.7 โดยแนวคิดมาจากการมองสัญลักษณ์ทางเลือกเป็นสถานะเทียมที่อยู่ระหว่างสัญลักษณ์การเปลี่ยนสถานะโดยสนใจเส้นการเปลี่ยนสถานะที่พุ่งออกจากสัญลักษณ์ทางเลือกและนำเงื่อนไขการ์ดหรือเหตุการณ์ที่ปรากฏบนเส้นสัญลักษณ์การเปลี่ยนมาเป็นเงื่อนไข เพื่อพิจารณาว่าจะเปลี่ยนสถานะไปที่สถานะใด



สัญลักษณ์สเตทแมชชีน

```
1:  st_i :
2:    if stateStatus st_i == idle  /*ถ้า stateStatus st_i มีค่าที่เท่ากับ idle
          ถ้า จริงทำในคำสั่ง thenถ้าเป็นเท็จให้ print case else  */
3:    then run state st_i();/*สั่งrun proctype State st_i */
4:    else print "case else"
5:    if stateStatus st_i == done  && Guard_i==true /*ตรวจสอบ
          stateStatus st_i  เท่ากับ done ซึ่งหมายถึงได้ทำงานใน
          proctype State st_i เสร็จแล้ว และตรวจสอบเงื่อนไขการ์ด
          หรือเหตุการณ์ที่เขียนบนเส้นถ้าเป็นจริงทำคำสั่งใน then
          ถ้าเป็นเท็จให้ print case else ในบรรทัดที่ 10 */
6:        then
7:            if Guard_k == true /*ตรวจสอบเงื่อนไขการ์ดหรือ
                  เหตุการณ์ที่ปรากฏบนเส้นถ้าเป็นจริงทำคำสั่ง
                  ในthen*/
8:             then stateStatus st_i = idle;  goto st_k;
9:             else print "case else"
10:   else print "case else"
11: goto st_i;

12:  st_j :
13:    if stateStatus st_j ==idle
14:          then run state st_j ();/*สั่งrun proctype State st_j */
15:   else print "case else"
16:    if stateStatus st_j==done && Guard_j==true /*ตรวจสอบ
          stateStatus st_j  เท่ากับ done ซึ่งหมายถึงได้ทำงานใน
          proctype State st_j เสร็จแล้ว และตรวจสอบ เงื่อนไขการ์ด
          หรือเหตุการณ์ที่เขียนบนเส้นถ้าเป็นจริงทำคำสั่งใน then
          ถ้าเป็นเท็จให้ print case else ในบรรทัดที่ 21 */
17:         then
18:           if Guard_k == true /*ตรวจสอบเงื่อนไขการ์ดหรือ
                  เหตุการณ์ที่ปรากฏบนเส้นถ้าเป็นจริงทำคำสั่ง
                  ในthen*/
19:            then stateStatus st_j = idle; goto st_k;
20:            else print "case else"
21: else print "case else"
22: goto st_j;

23: st_k:
24:   if stateStatus st_k == idle
25:       then run state st_k();
          :
```

แม่แบบภาษาโพรเมลา

ภาพที่ 3.7 แม่แบบโครงสร้างภาษาโพรเมลาส่วนของการแปลงสัญลักษณ์สัญลักษณ์ทางเลือกที่มีเส้นการเปลี่ยนสถานะเข้า n เส้นและออก 1 เส้น

จากรายละเอียดการแปลงสามารถสรุปเป็นแบบตารางแม่แบบภาษาโพรเมลาได้ดังตารางที่ 3.1 และหลังจากได้ภาษาโพรเมลาตามแนวคิดข้อ 3.1 ต่อไปเป็นส่วนของการประกาศตัวแปร

เนื่องจากในการทำงานของภาษาโพรเมลาจำเป็นต้องมีการประกาศตัวแปรเหมือนภาษาอื่น โดยหลังได้จากภาษาโพรเมลาตามข้อ 3.1 จะพบว่ามีตัวแปรที่เกิดจากเงื่อนไขก่อน เงื่อนไขหลังปรากฏในส่วนของ proctype และตัวแปรเงื่อนไขการ์ดและเหตุการณ์ปรากฏในกรอบของ init โดยตัวแปรเหล่านี้ต้องถูกประกาศตัวแปรในภาษาโพรเมลาเหมือนภาษาอื่น

การแปลงค่ายืนยง สำหรับการแปลงค่ายืนยงนั้นจะแปลงเป็นตัวแปรและทำการแปลงเป็น active proctype เนื่องจากค่ายืนยงคือค่าที่ต้องเป็นจริงตลอดดังนั้นจึงแปลงเป็น active proctype เพื่อให้มีหน้าที่เหมือนตัวมอนิเตอร์ภาษาโพรเมลาที่กำลังเอ็กซิคิวต์ ณ ขณะนั้น โดยมีรูปแบบในการแปลงดังภาพที่ 3.8 โดย 3.8 (a) คือ ค่ายืนยงที่ปรากฏบนแผนภาพสเตทแมชชีน 3.8 (b) คือ ตัวแปรที่ได้จากค่ายืนยงและ active proctype ที่ได้จากการแปลงค่ายืนยง โดยบรรทัดที่ 1 คือ ตัวแปรที่ได้จากการแปลง บรรทัดที่ 2 ถึงบรรทัดที่ 7 คือ ส่วนของ active proctype

inv:age > 0

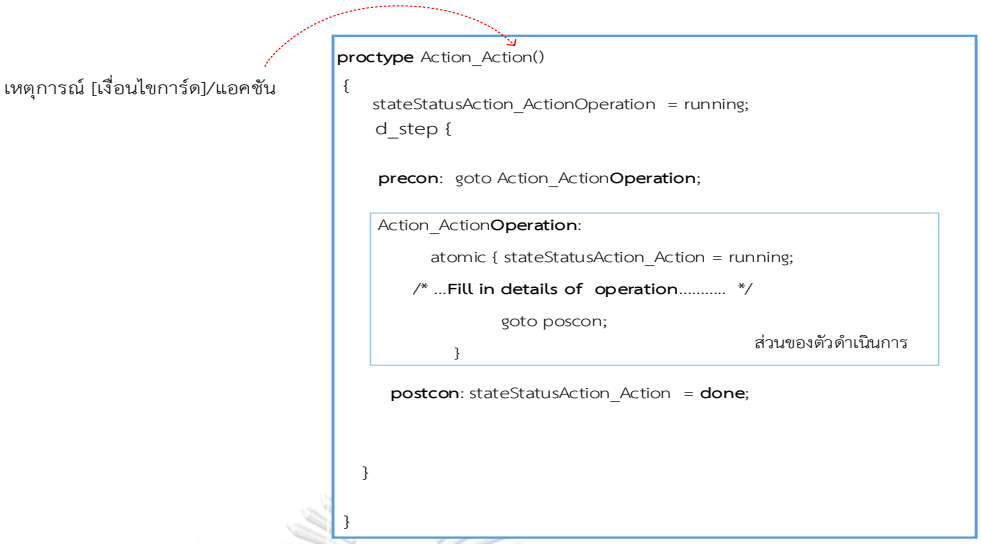
(a) รูปแบบโอซีแอล

```
1:  int age = 1
2: active proctype invage()
3: {
4: do
5: ::assert(age == 1)
6: od;
7: }
```

(b) รูปแบบภาษาโพรเมลา

ภาพที่ 3.8ค่ายืนยงในรูปแบบโอซีแอลและภาษาโพรเมลา

การแปลงแอคชันจะทำการแปลงเป็น Dummy proctype ซึ่งเป็นโครงของ proctype โดยภายในเนื้อหาของ proctype จะไม่ระบุรายละเอียดของกิจกรรมที่ทำเมื่อเข้ามาในสถานะหรือกิจกรรมที่ทำก่อนออกจากสถานะ ผู้ใช้ต้องทำการใส่รายละเอียดด้วยตัวเองหากต้องการทำกิจกรรมบางอย่างในแอคชันดังกล่าว



ภาพที่ 3.9ตัวอย่าง proctype ที่ได้จากการแปลงแอคชัน

โดยจากภาพที่ 3.9 ในการแปลงแอคชันเป็น Dummy proctype นั้นจะตั้งชื่อด้วย Action_ และตามด้วยชื่อแอคชันที่ปรากฏบนแผนภาพ และส่วนของการประกาศตัวแปรของ proctype นั้น ประกาศเช่นเดียวกับการแปลงสัญลักษณ์สถานะเพียงเปลี่ยนชื่อจากชื่อสถานะมาเป็นชื่อแอคชัน

3.2 การรวมองค์ประกอบของภาษาโพรเมลา

เนื่องจากส่วนประกอบของภาษาโพรเมลามีองค์ประกอบ 3 ส่วน หลักคือส่วนของ proctype, initและตัวแปร โดยหลังจากได้ภาษาโพรเมลาที่ได้จากกฎการแปลงแต่ละส่วนจากนั้นจะนำแต่ละส่วนมาประกอบรวมกันโดย 1 แฟ้มเอกสาร .pml จะได้ภาษาโพรเมลาที่มีลำดับส่วนของเนื้อหาดังภาพที่ 3.10 แสดงถึงลำดับการรวมองค์ประกอบของภาษาโพรเมลาที่ได้จากการแปลง และภาพที่ 3.11 แสดงตัวอย่างของแผนภาพสเตทแมชชีน ซึ่งหลังจากที่แปลงแผนภาพสเตทแมชชีนในภาพที่ 3.11 จะได้ชุดภาษาโพรเมลาที่แปลงได้จากแต่ละสัญลักษณ์ จากนั้นจะนำแต่ละภาษาโพรเมลามารวมเป็นองค์ประกอบของภาษาโพรเมลา

การประกาศตัวแปร /*ได้จากตัวแปรที่เกิดจากการแปลงสถานะเริ่มต้น เงื่อนไขก่อน เงื่อนไขหลัง ค่ายืนยง เหตุการณ์ เงื่อนไขการ์ดและตัวแปรที่เกิดจากการแปลงสถานะสิ้นสุด*/

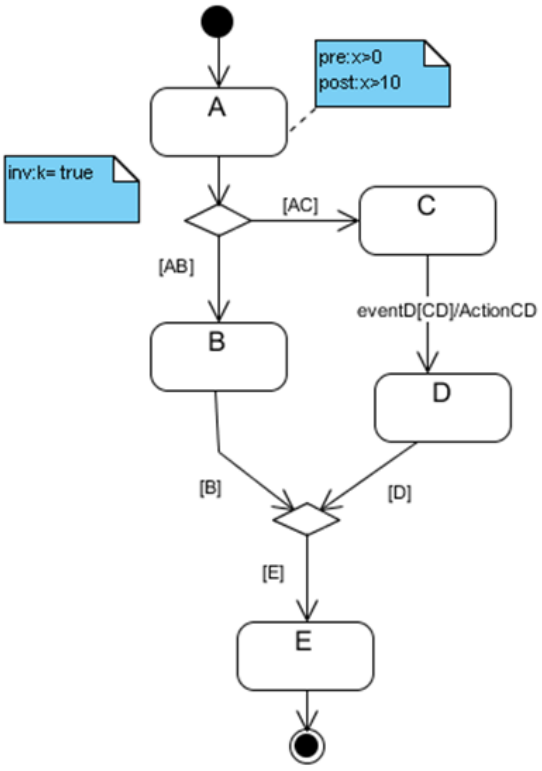
proctype ที่ได้จากการแปลงสถานะ /*ได้จากกฎการแปลงข้อที่ 1*/

proctype ที่ได้จากการแปลงสถานะสิ้นสุด /*ได้จากกฎการแปลงข้อที่ 3*/

init ที่ได้จากการแปลงสถานะเริ่มต้น โดยเนื้อหาภายใน init ได้จากการแปลงการเปลี่ยนสถานะและการแปลงทางเลือก /*ได้จากกฎการแปลงข้อที่ 2, 4, 5, 6*/

proctype ที่ได้จากการแปลงค่ายืนยง /*ได้จากการแปลงค่ายืนยง*/

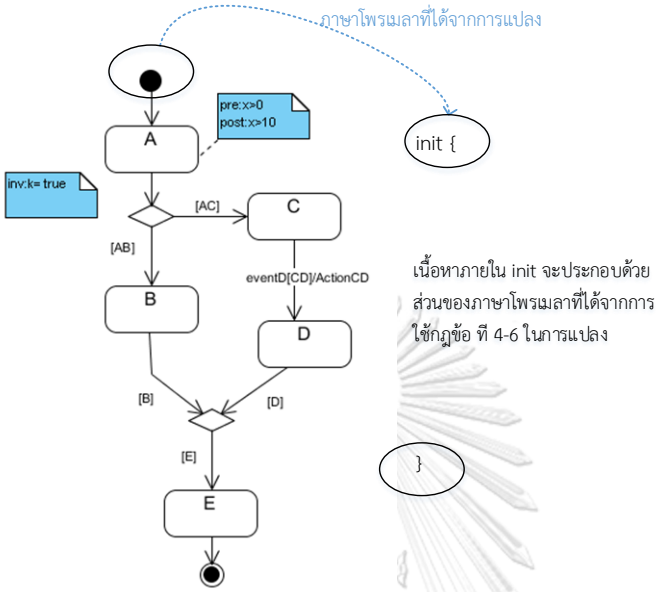
ภาพที่ 3.10 ลำดับการรวมองค์ประกอบของภาษาโพรเมลาที่ได้จากการแปลง



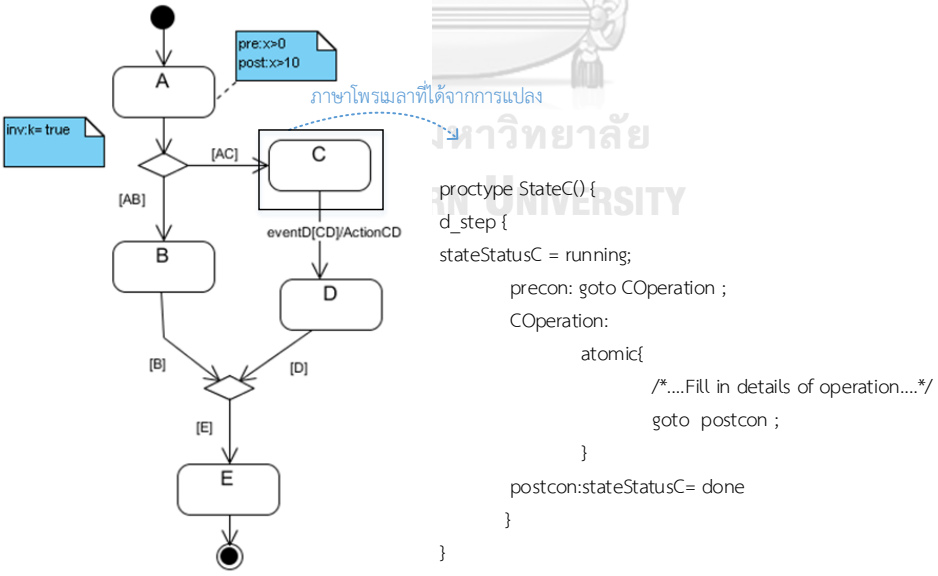
ภาพที่ 3.11 ตัวอย่างจำลองแผนภาพสเตทแมชชีน

โดยจากตัวอย่างภาพที่ 3.11 หลังจากที่ทำการแปลงแผนภาพแล้วจะได้ภาษาโพรเมลาที่แบ่งออกเป็นส่วนๆ ตามแม่แบบโครงสร้างภาษาโพรเมลาดังตัวอย่างในภาพที่ 3.12 เป็นตัวอย่างการแปลง

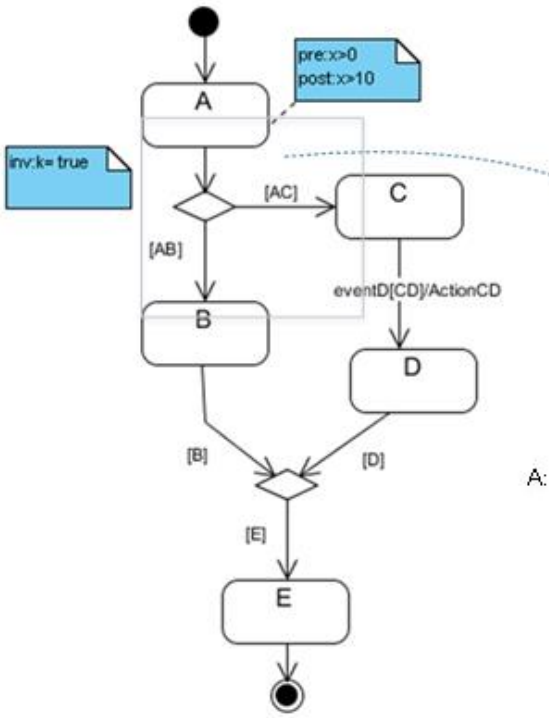
สัญลักษณ์สถานะเริ่มต้นของภาพที่ 3.11 ภาพที่ 3.13 แสดงภาษาโพรเมลาที่ได้จากการแปลงสัญลักษณ์สถานะของภาพที่ 3.11 ภาพที่ 3.14 แสดงภาษาโพรเมลาที่ได้จากการแปลงสัญลักษณ์ทางเลือกของภาพที่ 3.11



ภาพที่ 3.12 ภาษาโพรเมลที่ได้จากการแปลงสถานะเริ่มต้น



ภาพที่ 3.13 ภาษาโพรเมลาที่ได้จากการแปลงสัญลักษณ์สถานะ C



ภาษาโพรเมลาที่ได้จากการแปลง

```
A:
if
::(stateStatusA== idle)->run StateA( )
:: else -> printf("case else")
fi;
if
::(stateStatusA== done)->
        if
        ::(ab)-> stateStatusA=idle ;  goto B
        :: else -> printf("case else")
        fi;
        if
        ::(ac)-> stateStatusA=idle ;goto C
        :: else -> printf("case else")
        fi;
:: else -> printf("case else")
fi;
goto A;
```

ภาพที่ 3.14 ภาษาโพรเมลาที่ได้จากการแปลงสัญลักษณ์ทางเลือก

```
/*ตัวแปรได้จากกฎการแปลงสัญลักษณ์สถานะ ตัวแปร
ที่เกิดจากเงื่อนไขการ์ดและเหตุการณ์ เงื่อนไขก่อน
เงื่อนไขหลัง ค่ายืนยง การแปลงสถานะสิ้นสุด*/
mtype ={idle ,running ,done}
mtype stateStatusA = idle
 .....
.....
mtype stateStatusE= idle
bool preFailA= false
 .....
.....
bool postFailE= false
bit Terminate = 0 ;
bit CompleteFinalState = 0 ;
.....
.....
bool ab = true;
/*proctype ได้จากกฎการแปลงสถานะ*/
proctype StateA() {
d_step { stateStatusA = running;
 precon:
   if
   ::(x>0)->   goto AOperation ;
   :: else -> printf("case else")
   fi;
.....
.....
}
proctype StateB() {
.....
.....
}
.....
.....
```

ภาพที่ 3.15 ตัวอย่างบางส่วนที่ได้จากการเรียงองค์ประกอบของภาษาโพรเมลา

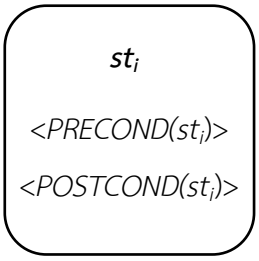
```
/*proctype ที่เกิดจากการแปลงสถานะสิ้นสุด*/
proctype FinalState() {
            CompleteFinalState = 1;
}
.....
.....
/*กรอบ init ที่ได้จากการแปลงสถานะเริ่มต้น*/
init {

/*ใช้กฎการแปลงข้อ 4, 5, 6*/

A:
    if
    ::(stateStatusA== idle)->run StateA( )
    :: else -> printf("case else")
    fi;
    .....
    .....
goto  A;
B:
    .....
    .....
}
/* proctype ที่เกิดจากการแปลงค่ายืนยง*/

active proctype invk()

{ do

 :: assert(k== true)

 od

}*/
```


ภาพที่ 3.15 ตัวอย่างบางส่วนที่ได้จากการเรียงองค์ประกอบของภาษาโพรเมลา (ต่อ)

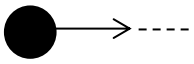
จากรายละเอียดการแปลงสามารถสรุปเป็นแบบตารางแม่แบบโครงสร้างภาษาโพรเมลาดังตารางที่ 3.1

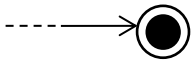
ตารางที่ 3.1แม่แบบโครงสร้างภาษาโพรเมลาที่สอดคล้องกับสัญลักษณ์ของแผนภาพสเตทแมชชีน

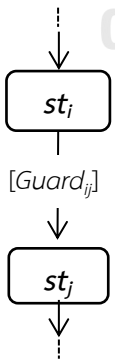
ข้อ	สัญลักษณ์แผนภาพ	โครงสร้างภาษาโพรเมลา

1		1: mtype = {idle,running,done};
		2: mtype stateStatusst_i= idle;
		3: bool precFailst_i= false, postFailst_i= false;
		4: **Proctype** Statest_i (){
		5: **d_step** {
		6: stateStatusst_i = running;
		7: **precon**:
		8: if ***PRECOND(st_i)*** == true
		9: then goto st_i**Operation**;
		10: else
		11: print “case else”
		12: if !***PRECOND(st_i)*** == true
		13: then precFailst_i= true; Terminate =1;
		14: else
		15: print “case else”
		16: st_i**Operation**:
		17: atomic { stateStatusst_i = running;
		18: /* ...**Fill in details of operation**..... */
		19: goto poscon;
		20: }
		21: **postcon**:
		22: if ***POSTCOND(st_i)***== true
		23: then stateStatusst_i = **done**;
		24: else
		25: print “case else”
		26: if !***POSTCOND(st_i)***== true
		27: then postFail = true; Terminate =1;
		28: else
		29: print “case else”
		30 } /* ...**end d_step**...*/
		31: }

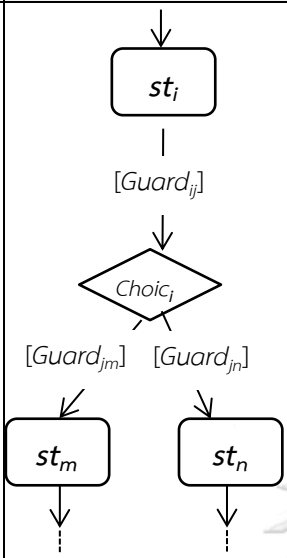
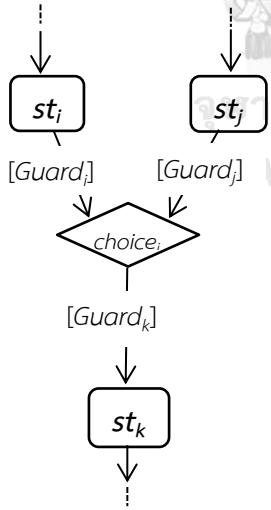
ตารางที่ 3.1 แม่แบบโครงสร้างภาษาโพรเมลาที่สอดคล้องกับสัญลักษณ์ของแผนภาพสเตทแมชชีน (ต่อ)

ข้อ	สัญลักษณ์แผนภาพ	โครงสร้างภาษาโพรเมลา
2		1: **init** { 2: ..... 3: ..... 4: }

3		1: bit CompleteFinalState = 0;
		2: bit Terminate = 0;
		3: proctype ***Finalst***$_i$( )
		4: { CompleteFinalState = 1; }
		5:active protype checkInvariantNormalTerminate ()
		6:{
		7: do
		8: ::assert(CompleteFinalState ==0)
		9: od;
		10:}
		11:active proctype checkInvariantAbNormalTerminate()
		12:{
		13: do
		14: ::assert(Terminate ==0)
		15:od;
		16:}

4		: 1: ***st***$_i$: 2: if stateStatus***st***$_i$== idle 3: then run state***st***$_i$( ); 4: else print “case else” 5: if stateStatus***st***$_i$== done && ***Guard***$_{ij}$ ==true 6: then stateStatus***st***$_i$ = idle; goto ***st***$_j$; 7: else print “case else” 8: goto ***st***$_i$; 9: ***st***$_j$: :

ตารางที่ 3.1 แม่แบบโครงสร้างภาษาโพรเมลาที่สอดคล้องกับสัญลักษณ์ของแผนภาพสเตทแมชชีน (ต่อ)

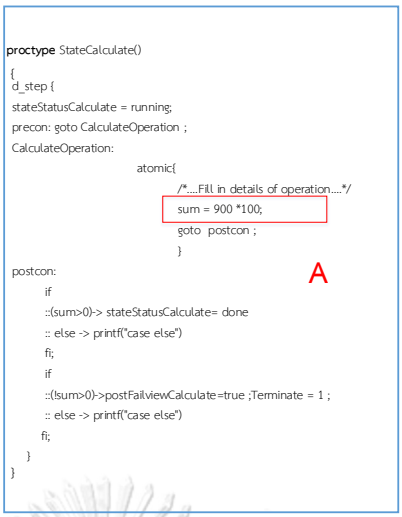
ข้อ	สัญลักษณ์แผนภาพ	โครงสร้างภาษาโพรเมลา
5		1: ***st***$_i$: 2: if stateStatus***st***$_i$== idle 3: then run state***st***$_i$( ); 4: else print “case else” 5: if stateStatus***st***$_i$== done && ***Guard***$_{ij}$ ==true 6: then if ***Guard***$_{jm}$ ==true then stateStatusA=idle; goto ***st***$_m$ else print “case else” 7: if ***Guard***$_{jn}$ ==true then stateStatusA=idle; goto ***st***$_n$ else print “case else” 8: else print “case else” 9: goto ***st***$_i$; 10: ***st***$_n$:
6		1: ***st***$_i$: 2: if stateStatus***st***$_i$== idle 3: then run state***st***$_i$(); 4: else print “case else” 5: if stateStatus***st***$_i$== done && ***Guard***$_i$==true 6: then 7: if ***Guard***$_k$ == true 8: then stateStatus***st***$_i$= idle; goto ***st***$_k$; 9: else print “case else” 10: else print “case else” 11: goto ***st***$_i$;

ตารางที่ 3.1 แม่แบบโครงสร้างภาษาโพรเมลาที่สอดคล้องกับสัญลักษณ์ของแผนภาพสเตทแมชชีน (ต่อ)

ข้อ	สัญลักษณ์แผนภาพ	โครงสร้างภาษาโพรเมลา
		12: st_j : 13: if stateStatusst_j ==idle 14: then run statest_j (); 15: else print “case else” 16: if stateStatusst_j==done && $Guard_j$==true 17: then 18: if $Guard_k$ == true 19: then stateStatusst_j = idle; goto st_k; 20: else print “case else” 21: else print “case else” 22: goto st_j; 23: st_k: 24: if stateStatusst_k== idle 25: then run statest_k(); :

3.3 การเพิ่มรายละเอียดในภาษาโพรเมลาเพื่อพร้อมสำหรับการทวนสอบ

เนื่องจากในสถานะแต่ละสถานะนั้นมีกิจกรรมที่ทำในระหว่างสถานะซึ่งจากแผนภาพสเตทแมชชีนไม่ได้ระบุเนื้อหารายละเอียดของกิจกรรมดังกล่าว ดังนั้นเมื่อได้ภาษาโพรเมลาที่ได้จากการแปลงแล้วหากถ้าผู้ใช้งานเครื่องมือ ต้องการนำภาษาโพรเมลาไปใช้งานในการทวนสอบโดยต้องการเพิ่มรายละเอียดในส่วนกิจกรรมดังกล่าว ผู้ใช้สามารถเพิ่มได้ในส่วนการดำเนินการ ยกตัวอย่างเช่น มีสถานะของการคำนวณราคาสินค้า โดยต้องการคำนวณค่าผลรวมของสินค้าเมื่ออยู่ในสถานะดังกล่าว ให้ผู้ใช้แทรกคำสั่งการคำนวณดังภาพที่ 3.17 แสดงตัวอย่างการเพิ่มรายละเอียดลงใน proctype จากภาพ A คือส่วนของการเพิ่มรายละเอียดโดยเป็นการคำนวนค่าให้กับตัวแปร sum



ภาพที่ 3.16 ตัวอย่างการเพิ่มรายละเอียดลงใน proctype

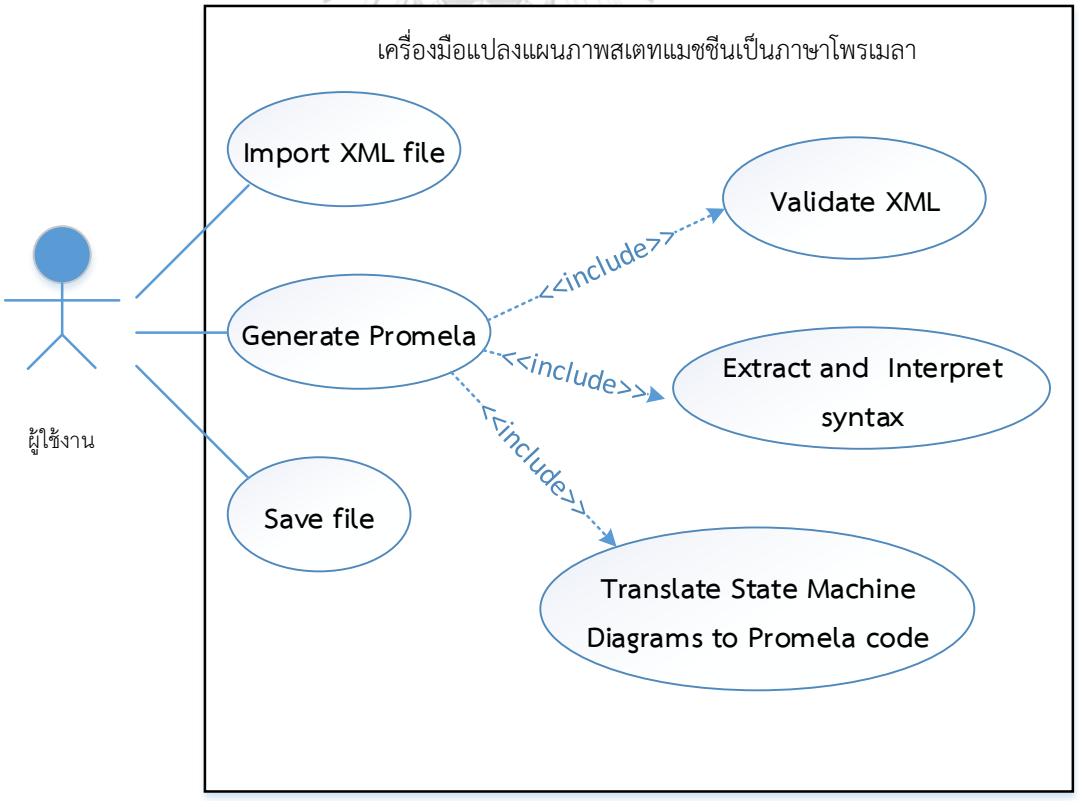
บทที่ 4
การออกแบบและพัฒนาเครื่องมือ

การวิเคราะห์และออกแบบเครื่องมือผู้วิจัยจะอธิบายด้วยแผนภาพยูสเคส (Use Case Diagram) แผนภาพคลาส แผนภาพกิจกรรม เพื่อแสดงให้เห็นถึงขอบเขตการทำงานและความสามารถของเครื่องมือรวมทั้งส่วนต่อประสานของเครื่องมือ

4.1 การออกแบบเครื่องมือ

เครื่องมือการแปลงแผนภาพสเตทแมชชีนเป็นภาษาโพรเมลา เป็นเครื่องมือที่สนับสนุนการทวนสอบด้วยเครื่องมือสปิน สำหรับการออกแบบเครื่องมือจะแบ่งเป็น 2 ส่วนดังนี้

1) แผนภาพยูสเคส
2) แผนภาพคลาส
3) แผนภาพกิจกรรม



ภาพที่ 4.1 แผนภาพยูสเคสแสดงขอบเขตและการทำงานของเครื่องมือ

ตารางที่ 4.1 รายละเอียดยูสเคสการนำเข้าแผนภาพสเตทแมชชีนเอกซ์เอ็มแอลไฟล์

Use Case ID:	UC_001
Use Case Name:	Import XML File
Actor:	ผู้ใช้งาน
Description:	นำเข้าแผนภาพสเตทแมชชีนในรูปแบบแฟ้มเอกสารเอกซ์เอ็มแอลเพื่อเป็นข้อมูลนำเข้าสำหรับการแปลงแผนภาพเป็นภาษาโพรเมลา
Precondition:	แฟ้มเอกสารเอกซ์เอ็มแอลต้องได้จากเครื่องมือ visual-paradigm
Postcondition:	-
Step:	1.ระบบแสดงหน้าจอให้เลือกแฟ้มเอกสาร 2.ผู้ใช้เลือกแฟ้มเอกสาร 3.ระบบอ่านแฟ้มเอกสารเข้ามาประมวลผลและจัดเก็บ
Exceptions conditions	-

ตารางที่ 4.2 รายละเอียดยูสเคสการสร้างภาษาโพรเมลา

Use Case ID:	UC_002
Use Case Name:	Genarate Promela
Actor:	ผู้ใช้งาน
Description:	ระบบสร้างภาษาโพรเมลาให้ผู้ใช้งาน
Precondition:	แผนภาพสเตทแมชชีนในรูปแบบแฟ้มเอกสารเอกซ์เอ็มแอลถูกนำเข้ามาในเครื่องมือแล้ว
Postcondition:	-
Step:	1. ระบบเรียกยูสเคสตรวจสอบแฟ้มเอกสารเอกซ์เอ็มแอล (Validate XML) ว่าใช่แฟ้มแผนภาพสเตทแมชชีนหรือไม่ 2. ระบบเรียกยูสเคสตัดคำและตีความไวยากรณ์ 3. ระบบเรียกยูสเคสการแปลงแผนภาพสเตทแมชชีนเป็นภาษาโพรเมลา
Exceptions conditions:	สถานการณ์แฟ้มเอกสารเอกซ์เอ็มแอลไม่ใช่แผนภาพสเตทแมชชีน ถ้าไม่ใช้จะแจ้งเตือนผู้ใช้

ตารางที่ 4.3 รายละเอียดยูสเคสการตรวจสอบแฟ้มเอกสารเอกซ์เอ็มแอล

Use Case ID:	UC_003
Use Case Name:	Validate XML
Actor:	ผู้ใช้งาน
Description:	ระบบทำการตรวจสอบแฟ้มเอกสารเอกซ์เอ็มแอลว่าถูกต้องตามไวยากรณ์ของแผนภาพสเตทแมชชีนหรือไม่
Precondition:	นำเข้าแผนภาพสเตทแมชชีนในรูปแบบแฟ้มแอกสารเอกซ์เอ็มแอล
Postcondition:	-
Step:	ระบบทำการตรวจสอบแฟ้มเอกสารเอกซ์เอ็มแอลถ้าพบว่าไม่ใช่ระบบจะแจ้งเตือนผู้ใช้
Exceptions conditions	-

ตารางที่ 4.4 รายละเอียดยูสเคสการสกัดและตีความ

Use Case ID:	UC_004
Use Case Name:	Extract and Interpret syntax
Actor:	ผู้ใช้
Description:	เครื่องมือทำการสกัดและตีความแผนภาพสเตทแมชชีนและเก็บรายละเอียดที่จำเป็นสำหรับการแปลงแผนภาพเป็นภาษาโพรเมลา
Precondition:	นำเข้าแผนภาพสเตทแมชชีนในรูปแบบแฟ้มแอกสารเอกซ์เอ็มแอล
Postcondition:	-
Step:	ระบบจะทำการสกัดและตีความแฟ้มแอกสารเอกซ์เอ็มแอล
Exceptions conditions	-

ตารางที่ 4.5 รายละเอียดยูสเคสการแปลงแผนภาพสเตทแมชชีนเป็นภาษาโพรเมลา

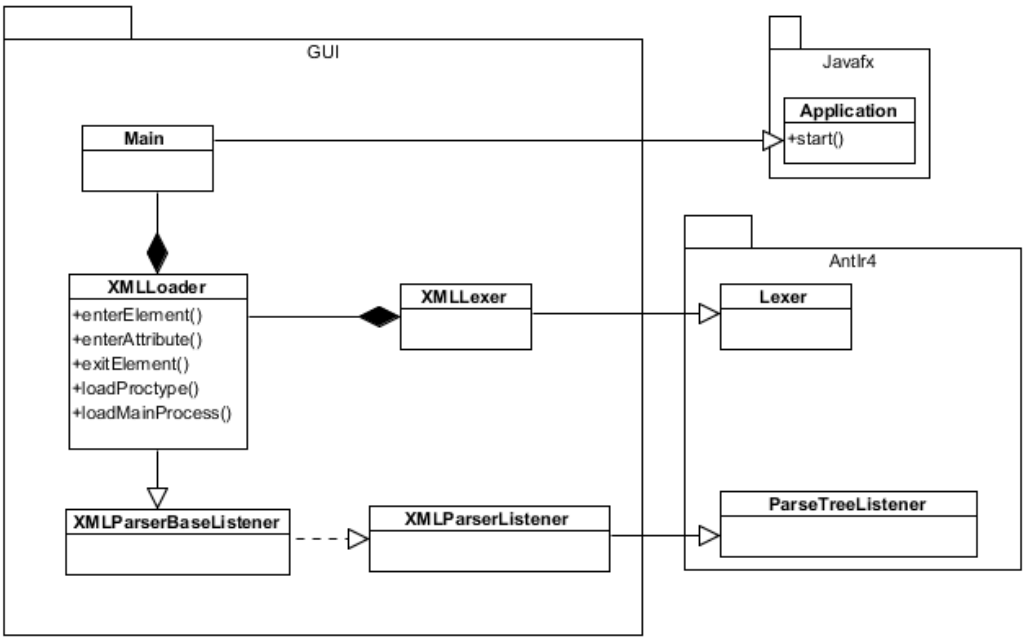
Use Case ID:	UC_005
Use Case Name:	Translate State Machine Diagrams to Promela code
Actor:	ผู้ใช้
Description:	ระบบนำข้อมูลที่ได้จากการสกัดและตีความมาทำการแปลงเป็นภาษาโพรเมลา
Precondition:	ระบบทำการสกัดและตีความแผนภาพสเตทแมชชีนและเก็บรายละเอียดที่จำเป็นสำหรับการแปลงเป็นภาษาโพรเมลา
Postcondition:	-
Step:	ระบบทำการแปลงแผนภาพสเตทแมชชีนเป็นภาษาโพรเมลา
Exceptions conditions	-

ตารางที่ 4.6 รายละเอียดยูสเคสบันทึกไฟล์

Use Case ID:	UC_003
Use Case Name:	Save file
Actor:	ผู้ใช้งาน
Description:	บันทึกแบบจำลองภาษาโพรเมลา
Precondition:	เมื่อทำการแปลงแผนภาพเรียบร้อยแล้ว
Postcondition:	-
Step:	ผู้ใช้งานตั้งชื่อและบันทึกนามสกุลไฟล์เป็น.pml
Exceptions conditions	-

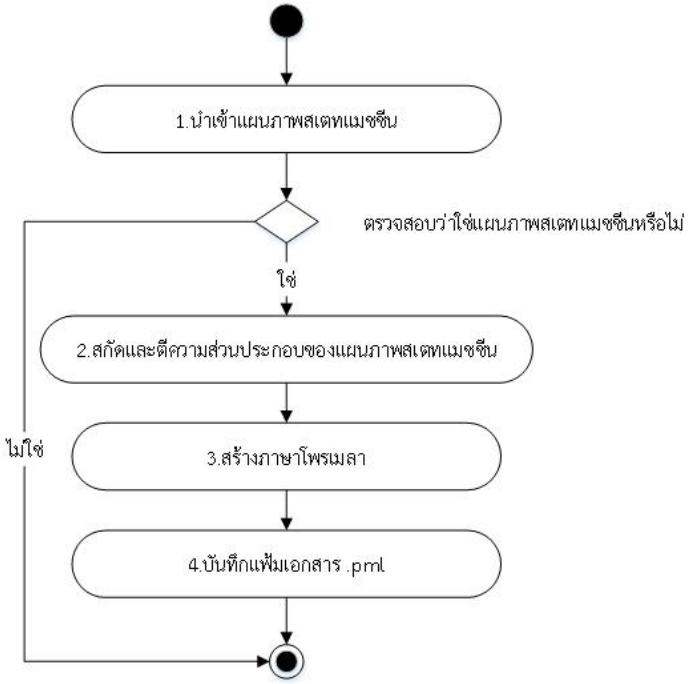
2) แผนภาพคลาส (Class diagram) สำหรับการพัฒนาเครื่องมือการแปลงมีการใช้งาน 3 แพ็กเกจ (Package) ดังนี้ แพ็กเกจ Javafx ใช้ในการจัดการส่วนต่อประสานผู้ใช้ แพ็กเกจ Antlr4 ใช้สำหรับในการตีความภาษาเอ็กซ์แอมแอล แพ็กเกจ GUI เป็นส่วนการทำงานหลักของโปรแกรมซึ่งมีรายละเอียดดังนี้

- คลาส Main ทำหน้าที่เป็นจุดเริ่มต้นการทำงาน
- คลาส XMLLexer ทำหน้าที่ในการอ่านเอกสารและทำการตัดคำ
- คลาส XMLPaserBaseListener กำหนดส่วนเรียกกลับเมื่อส่วนตีความได้และรับ Token ที่ได้จากการตัดคำจาก XMLLexer
- คลาส XMLLoader สืบทอดมาจากคลาส XMLPaserBaseListener โดยใน XMLLoader ประกอบไปด้วยส่วนในการสกัดข้อมูลและบันทึกข้อมูลไว้สำหรับใช้ในการแปลงเป็นภาษาโพรเมลาจากนั้นทำการแปลงข้อมูลเหล่านั้นไปเป็นภาษาโพรเมลา



ภาพที่ 4.2 แผนภาพคลาสของเครื่องมือ

3) แผนภาพกิจกรรม



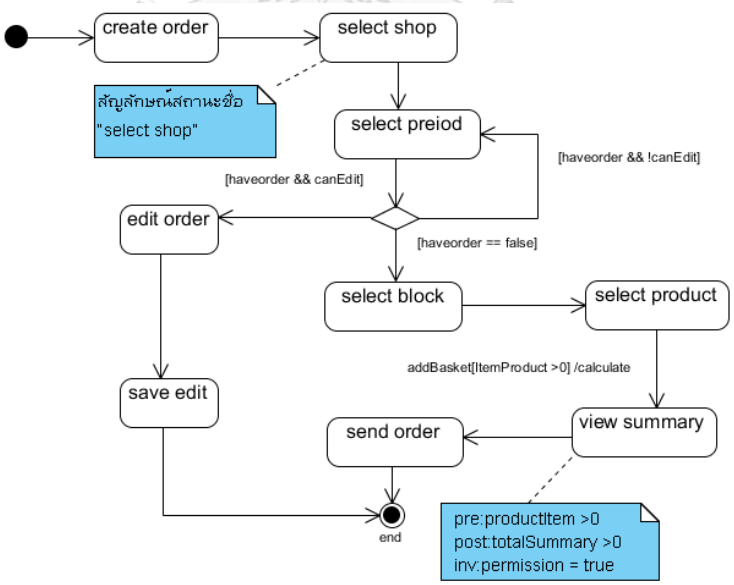
ภาพที่ 4.3 แผนภาพกิจกรรมของเครื่องมือ

สำหรับแผนภาพกิจกรรมนั้นจะประกอบด้วยกิจกรรมข้อมูลนำเข้าแผนภาพสเตทแมชชีน กิจกรรมสกัดและตีความองค์ประกอบของแผนภาพสเตทแมชชีนการและกิจกรรมการสร้างภาษาโพรเมลาโดยแต่ละกิจกรรมมีรายละเอียดดังนี้

1) ข้อมูลนำเข้าแผนภาพสเตทแมชชีน

ข้อมูลนำเข้าแผนภาพสเตทแมชชีนจะอยู่ในรูปแบบแฟ้มเอกสารเอกซ์เอ็มแอลโดยในงานวิจัยนี้ใช้เครื่องมือวิชวลพาราไดม์ รุ่น 14.0 (Visual Paradaigm version 14) สำหรับวาดแผนภาพสเตทแมชชีน และทำการแปลงแผนภาพดังกล่าวให้อยู่ในรูปแฟ้มเอกสารเอกซ์เอ็มแอลโดยแผนภาพสเตทแมชชีนที่นำเข้านั้นต้องมีความสมบูรณ์ในการระบุข้อมูลดังแสดงในภาพที่ 4.4 และภาพที่ 4.5 แสดงถึงแผนภาพสเตทแมชชีนจะอยู่ในรูปแบบแฟ้มเอกสารเอกซ์เอ็มแอล

1) สัญลักษณ์สถานะต้องมีชื่อกำกับและชื่อรองรับเฉพาะตัวอักขระเท่านั้น เช่น สถานะชื่อ “select shop” เป็นต้น
2) การเขียนโอซีแอลให้เขียนในรูปแบบตามดังนี้คือ ให้มีคำว่า “pre, post, inv” ตามด้วยเครื่องหมายทวิภาคนำหน้าและตามด้วยเงื่อนไขที่มีค่าความจริงเท่านั้น เช่น pre:productItem > 0, post:totalSummary >0, inv:permission = true ดังแสดงในรูปที่ 4.4



ภาพที่ 4.4 แผนภาพของสเตทแมชชีนที่วาดด้วยเครื่องมือวิชวลพาราไดม์

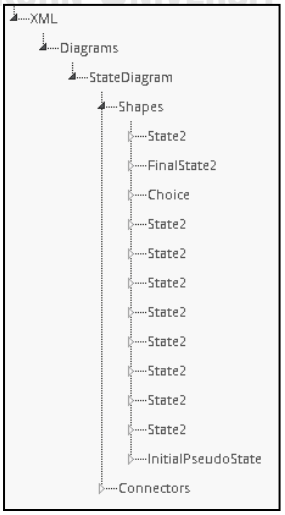
```
<Diagrams>
  <StateDiagram AlignToGrid="false" AutoFitShapesSize="false">
    <Shapes>
      <State2  Id="8jONRvqAUyn0Jgf4" MetaModelElement="8jONRvqAUyn0Jgf5" Name="select shop">
        <ElementFont Color="rgb(0, 0, 0)" Name="Dialog" Size="16" Style="0"/>
        <Line Cap="0" Color="rgb(0, 0, 0)" Transparency="0" Weight="1.0">
          <Stroke/>
        </Line>
        <Caption Height="21" InternalHeight="-2147483648" InternalWidth="-2147483648"/>
        <FillColor Color="rgb(255, 255, 255)" Style="1" Transparency="0" Type="1"/>
      </State2>
      <FinalState2 Id="qgCscvqAUyX45w2N" MetaModelElement="qgCscvqAUyX45w2O" Name="end">
        <ElementFont Color="rgb(0, 0, 0)" Name="Dialog" Size="11" Style="0"/>
        <Line Cap="0" Color="rgb(0, 0, 0)" Transparency="0" Weight="1.0">
          <Stroke/>
        </Line>
        <Caption Height="15" InternalHeight="-2147483648" InternalWidth="-2147483648"/>
        <FillColor Color="rgb(0, 0, 0)" Style="1" Transparency="0" Type="1"/>
      </FinalState2>
      <Choice Id="NTQEcvqAUyX45ww9" MetaModelElement="NTQEcvqAUyX45ww." Name="">
        <ElementFont Color="rgb(0, 0, 0)" Name="Dialog" Size="11" Style="0"/>
        <Line Cap="0" Color="rgb(0, 0, 0)" Transparency="0" Weight="1.0">
          <Stroke/>
        </Line>
        <Caption Height="0" InternalHeight="-2147483648" InternalWidth="-2147483648"/>
        <FillColor Color="rgb(255, 255, 255)" Style="1" Transparency="0" Type="1"/>
      </Choice>
```

ภาพที่ 4.5 แผนภาพของสเตทแมชชีนในรูปแบบแฟ้มเอกสารเอกซ์เอ็มแอล

2) การสกัดและตีความประกอบของแผนภาพสเตทแมชชีน

การสกัดแต่ละส่วนประกอบของแผนภาพสเตทแมชชีนจากเอกสารเอกซ์เอ็มแอล เพื่อที่จะแปลงแต่ละส่วนประกอบของแผนภาพไปเป็นส่วนประกอบของภาษาโพรเมลาตามแม่แบบโครงสร้างภาษาโพรเมลาที่สร้างขึ้น โดยมีรายละเอียดดังนี้

2.1) ทำการอ่านแต่ละส่วน ของส่วนประกอบของแผนภาพสเตทแมชชีนจากแฟ้มเอกสารเอกซ์เอ็มแอล โดยอ่านที่โหนดลูกของโหนด Shapes ดังภาพที่ 4.6 ทั้งหมดและทำการเก็บข้อมูลของแต่ละโหนดไว้ดังแสดงในตารางที่ 4.7

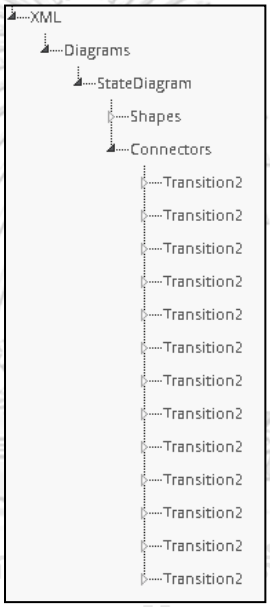


ภาพที่ 4.6 การสกัดแต่ละส่วนประกอบของแผนภาพของสเตทแมชชีน

ตารางที่ 4.7 การสกัดแต่ละส่วนประกอบของแผนภาพของสเตทแมชชีน

ชื่อสัญลักษณ์	**รูปแบบเอ็มแอล**	**เก็บข้อมูล**
สถานะเริ่มต้น	<InitialPseudoState>.....</InitialPseudoState>	เลขรหัส ชื่อ
สถานะสิ้นสุด	<<FinalState2>.....</FinalState2>	เลขรหัส ชื่อ
สถานะ	<State2>.....</State2>	เลขรหัส ชื่อ โอซีแอล
ทางเลือก	<Choice>.....</Choice>	เลขรหัส ชื่อ

2.2) ทำการอ่านแต่ละส่วนของเส้นการเปลี่ยนสถานะโดยอ่านที่โหนดลูกของโหนด Connectors ทั้งหมดดังภาพที่ 4.7 และทำการเก็บข้อมูลของแต่ละโหนดไว้ดังแสดงในตารางที่ 4.8



ภาพที่ 4.7 การสกัดแต่ละส่วนประกอบของสัญลักษณ์การเปลี่ยนสถานะ

ตารางที่ 4.8 การสกัดและเก็บข้อมูลของสัญลักษณ์การเปลี่ยนสถานะ

ชื่อสัญลักษณ์	**รูปแบบเอ็มแอล**	**เก็บข้อมูล**
การเปลี่ยนสถานะ	<Transition2>.....</Transition2>	เลขรหัส ชื่อ เชื่อมต่อจากรหัส เชื่อมไปยังรหัส

3) สร้างภาษาโพรเมลา

ในส่วนนี้เป็นการแปลงส่วนประกอบของแผนภาพสเตทแมชชีน ที่ได้จากขั้นตอนการสกัดและตีความ โดยใช้แม่แบบสำหรับการเทียบเคียงโครงสร้างของแผนภาพสเตทแมชชีนเป็นภาษาโพรเมลา

โดยเมื่อเจอข้อมูลในรูปแบบเอ็มแอลดังตารางที่ 4.9 ดังคอลัมน์ด้านซ้ายให้ทำการแปลงเป็นโครงสร้างภาษาโพรเมลาตามแม่แบบโครงสร้างภาษาโพรเมลาที่ออกแบบไว้ในตารางที่ 3.1

ตารางที่ 4.9 เทียบข้อมูลนำเข้าและการใช้แม่แบบแต่ละข้อสำหรับแปลงเป็นภาษาโพรเมลา

รูปแบบเอ็มแอล	**แม่แบบโครงสร้างภาษาโพรเมลา (ข้อ จากตารางที่ 3.1)**
<InitialPseudoState>.....</InitialPseudoState>	1
<FinalState2>.....</FinalState2>	2
<State2>.....</State2>	3
Transition2>.....</Transition2>	4
<Choice>.....</Choice>	5 ,6

หลังจากใช้แม่แบบโครงสร้างภาษาโพรเมลาเป็นตัวแบบในการแปลงภาษาโพรเมลาแล้วนั้นในลำดับต่อไปเป็นส่วนของการแปลงตัวแปรต่างๆ ที่ปรากฏบนแผนภาพสเตทแมชชีนเป็นภาษาโพรเมลา

4) หลังจากทำการสร้างภาษาโพรเมลาแล้วขั้นตอนสุดท้ายเป็นการบันทึกแฟ้มเอกสารนามสกุล .pml โดยให้ผู้ใช้ตั้งชื่อแฟ้มเอกสารและให้ตั้งนามสกุล.pml และทำการบันทึกเพื่อนำไปใช้งาน

4.2 สภาพแวดล้อมในการพัฒนาระบบ

สภาพแวดล้อมในการพัฒนาระบบแบ่งได้เป็น 2 ประเภท คือ ฮาร์ดแวร์ (Hardware) และซอฟต์แวร์ (Software) โดยมีรายละเอียดดังนี้

4.2.1 สภาพแวดล้อมในการพัฒนาเครื่องมือฮาร์ดแวร์

1) เครื่องคอมพิวเตอร์ 1 เครื่อง หน่วยประมวลผล Intel ® Core ™ i5 ความเร็ว 2.60 กิกะเฮิร์ซ (GHz)
2) หน่วยความจำสำรอง ขนาด 4.00 กิกะไบต์ (4.0 GB)
3) ฮาร์ดดิสก์ (Harddisk) 250 กิกะไบต์ (250 GB)

4.2.2 สภาพแวดล้อมในการพัฒนาเครื่องมือด้านซอฟต์แวร์

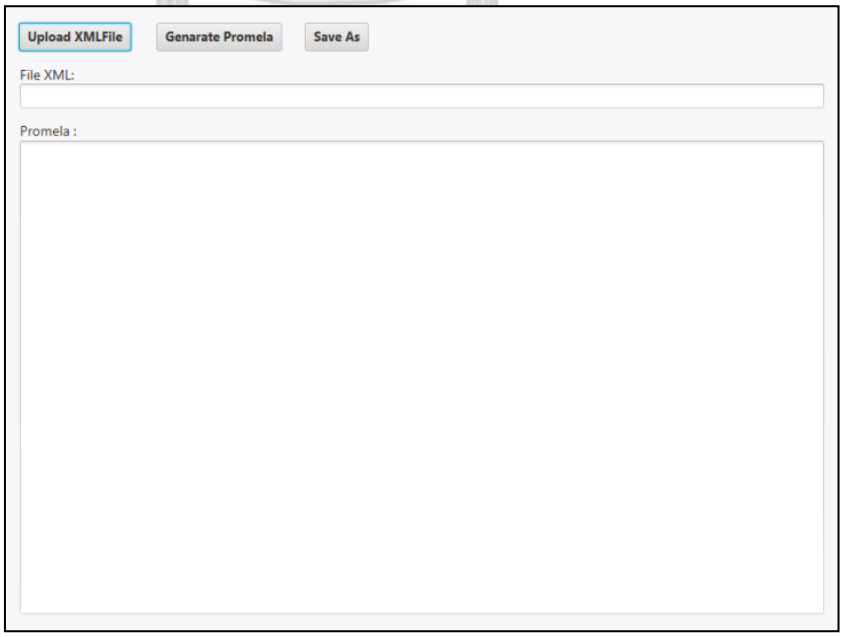
1) ระบบปฏิบัติการ Microsoft Windows 8
2) IntellijI DEA 2016.2
3) ANTLR v4 grammar plug-in
4) จาวา เจดีเค เวอร์ชัน 8.0 (JAVA JDK Version 8.0)

ภาพที่ 4.8 ชุดเครื่องมือ IntellijI DEA 2016.2

4.3 ส่วนต่อประสานกับผู้ใช้ของเครื่องมือ

การใช้งานเครื่องมือจะประกอบด้วย 3 ขั้นตอนหลัก ดังต่อไปนี้ คือ นำเข้าไฟล์แผนภาพสเตทแมชชีน การดำเนินการแปลงแผนภาพสเตทแมชชีนและการบันทึกภาษาโพรเมลาที่ได้

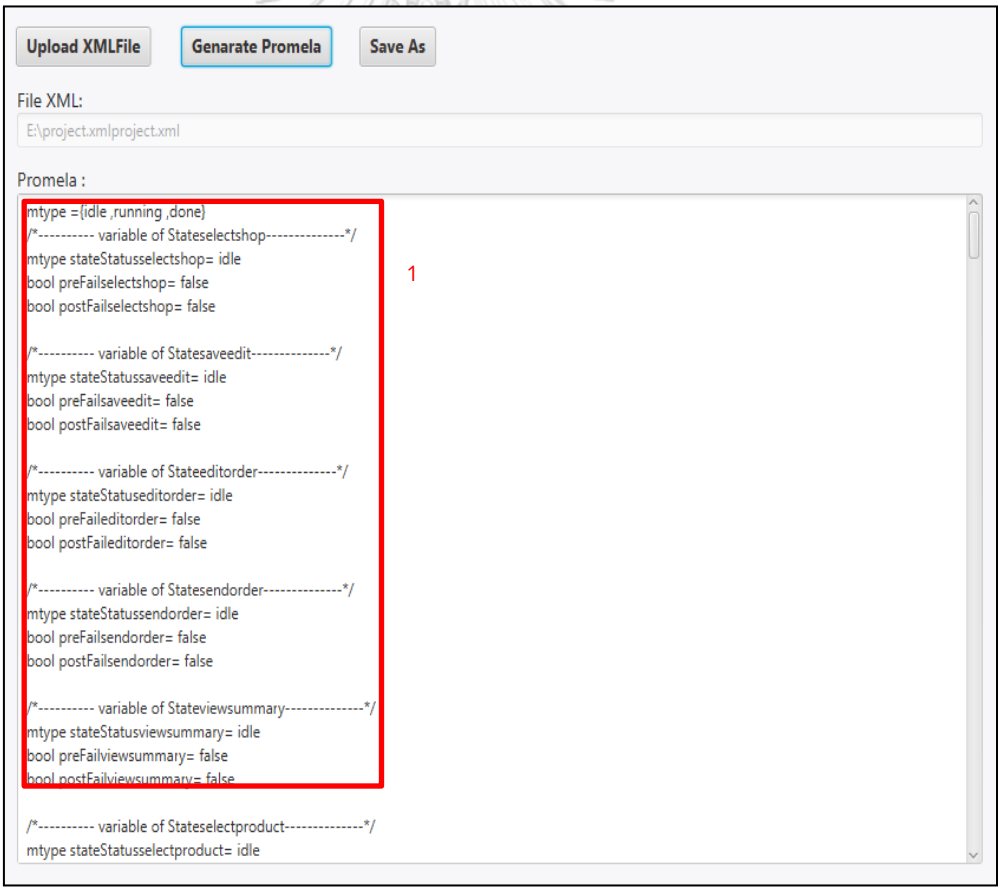
4.3.1 นำเข้าแผนภาพสเตทแมชชีนที่อยู่ในรูปแบบแฟ้มเอกสารเอกซ์เอ็มแอลโดยกดที่ปุ่ม Upload XMLFile ดังแสดงในภาพที่ 4.9 จากนั้นเลือกแฟ้มเอกสารแผนภาพสเตทแมชชีนที่อยู่ในรูปแบบแฟ้มเอกสารเอกซ์เอ็มแอล เมื่อทำการเลือกเสร็จหน้าจอที่กล่องข้อความจะปรากฏที่อยู่ของแฟ้มเอกสารที่ได้ทำการเลือกดังหมายเลข 1 ที่ปรากฏในภาพที่ 4.10



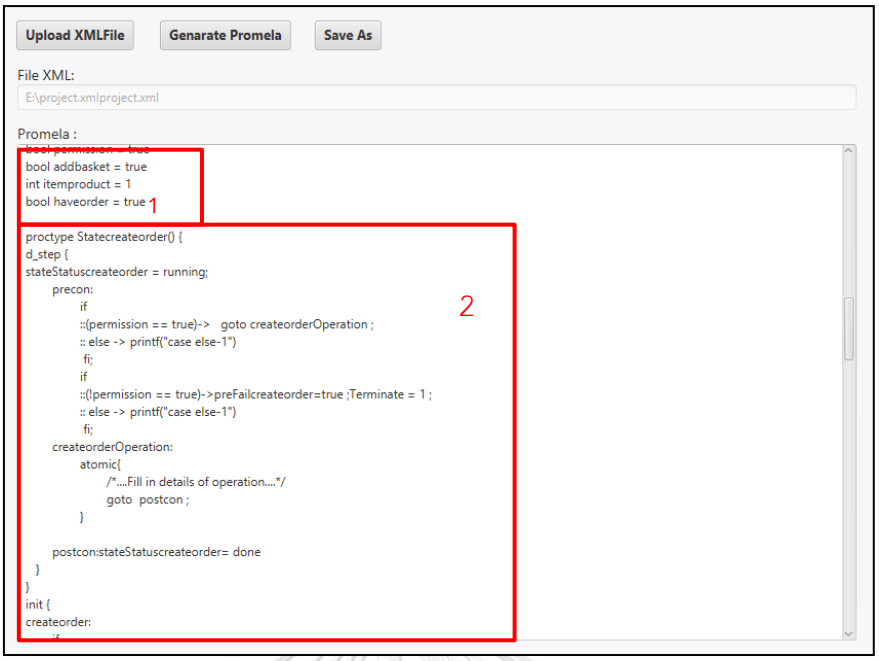
ภาพที่ 4.9 นำเข้าแฟ้มเอกสารแผนภาพสเตทแมชชีน

ภาพที่ 4.10 ที่อยู่ของแฟ้มเอกสาร

4.3.2 คลิกปุ่มสำหรับดำเนินการแปลง เพื่อสั่งให้โปรแกรมทำการแปลงแผนภาพสเตทแมชชีนไปเป็นแบบภาษาโพรเมลา ดังแสดงในภาพที่ 4.11 เมื่อทำการกดปุ่มเพื่อทำการแปลงแล้วจะปรากฏภาษาโพรเมลาที่ได้จากการแปลงดังรูปซึ่งสามารถให้ผู้ใช้แก้ไขภาษาโพรเมลาก่อนทำการบันทึกเป็นแฟ้มเอกสารนามสกุล.pml ได้ จะเห็นว่าส่วนแรกของภาษาโพรเมลาที่ได้เริ่มจากการประกาศตัวแปรของสถานะดังหมายเลข 1 ในภาพที่ 4.11 และลำดับต่อมาเป็นตัวแปรที่สร้างจากเงื่อนไขการ์ด เหตุการณ์และโอซีแอล ดังหมายเลข 1 ในภาพที่ 4.12 จากนั้นเป็นส่วนของ proctype ต่างๆ รวมถึง init ดังหมายเลย 2 ในภาพที่ 4.12

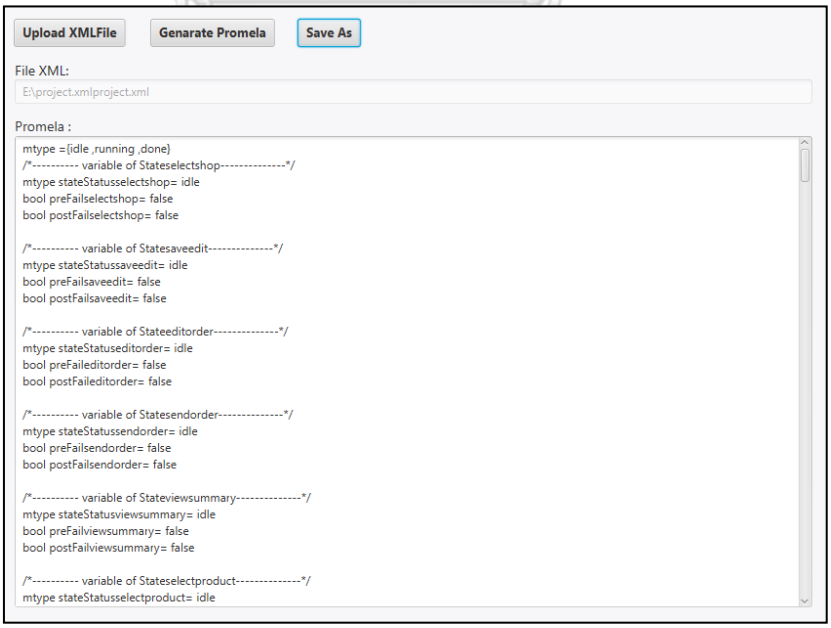


ภาพที่ 4.11 ภาษาโพรเมลาที่ได้จากการแปลง

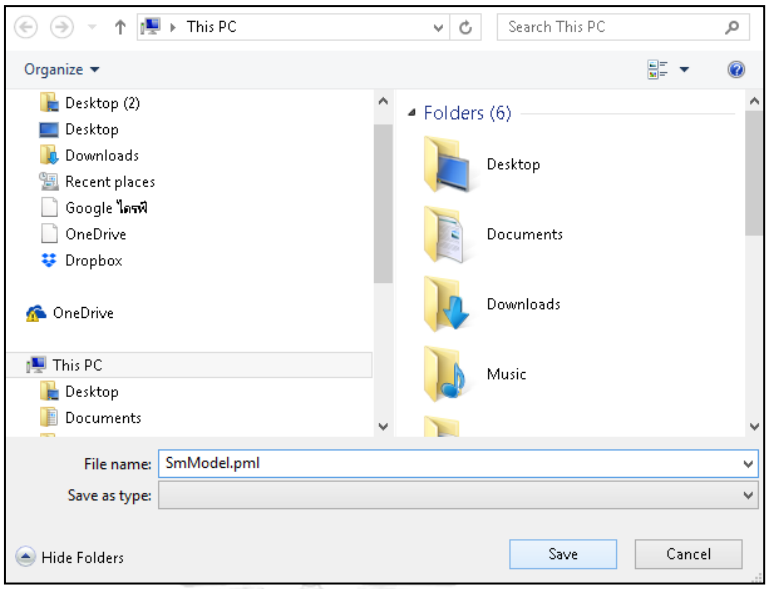


ภาพที่ 4.12 ภาษาโพรเมลาที่ได้จากการแปลงส่วนของสัญลักษณ์สถานะ

4.3.3 ขั้นตอนสุดท้ายเป็นการบันทึก ภาษาโพรเมลาสำหรับนำไปใช้ในการทวนสอบเชิงรูปนัยด้วยเครื่องมือสปิน ดังแสดงในภาพที่ 4.13 หลังจากทำการสร้างภาษาโพรเมลาแล้ว ผู้ใช้สามารถบันทึกภาษาโพรเมลาดังกล่าวเป็นแฟ้มเอกสารนามสกุล .pml เพื่อนำไปใช้งานต่อด้วยการนำเข้าเครื่องมือสปิน



ภาพที่ 4.13 การบันทึกภาษาโพรเมลา



ภาพที่ 4.14 หน้าจอการระบุตำแหน่งที่ต้องการบันทึกแฟ้มเอกสารภาษาโพรเมลา

4.4 การใช้งานแอนท์เลอร์สร้างเลกเซอร์และพาสเซอร์

เนื่องจากเครื่องมือการแปลง มีการใช้เลกเซอร์และพาสเซอร์ที่ได้จากเครื่องมือแอนท์เลอร์สำหรับสกัดและตีความส่วนประกอบของแผนภาพสเตทแมชชีน เพื่อแปลงแผนภาพสเตทแมชชีน เป็นภาษาโพรเมลา ดังนั้นในส่วนนี้จะกล่าวถึงการใช้งานแอนท์เลอร์สำหรับสร้างเลกเซอร์และพาสเซอร์ดังกล่าว โดยเริ่มจากผู้ใช้ต้องทำการติดตั้ง ANTLR v4 grammar plug-in บน เครื่องมือ Intellijl DEA จากนั้นนำเข้าไวยากรณ์เอกสารเอกซ์เอ็มแอล และให้ผู้ใช้ทำการสั่งแอนเลอร์ประมวลผลเพื่อสร้างเลกเซอร์และพาสเซอร์เพื่อสำหรับนำไปใช้งาน

บทที่ 5
การทดสอบเครื่องมือ

การทดสอบมีวัตถุประสงค์เพื่อทดสอบการทำงานของเครื่องมือและแม่แบบโครงสร้างภาษาโพรเมลาที่สร้างขึ้นว่าถูกต้องตรงตามที่ได้ทำการออกแบบไว้หรือไม่ โดยมีแนวทางในการทดสอบดังนี้

1) สร้างแผนภาพสเตทแมชชีนและส่งออกแผนภาพในรูปแบบแฟ้มเอกสารเอ็กซ์เอ็มแอล
2) เครื่องมือนำเข้าแฟ้มเอกสารเอ็กซ์เอ็มแอลจากนั้นทำการแปลงเป็นภาษาโพรเมลา
3) นำภาษาโพรเมลาที่ได้เข้าเครื่องมือสปิน
4) รวบรวมและสรุปผลการทดสอบ

5.1 สภาพแวดล้อมที่ใช้ในการทดสอบ

5.1.1 สภาพแวดล้อมในการพัฒนาเครื่องมือด้านฮาร์ดแวร์

1) เครื่องคอมพิวเตอร์ 1 เครื่อง หน่วยประมวลผล Intel ® Core TM i5 ความเร็ว 2.60 กิกะเฮิร์ซ (GHz)
2) หน่วยความจำสำรอง ขนาด 4.00 กิกะไบต์ (GB)
3) ฮาร์ดดิสก์ (Harddisk) 250 กิกะไบต์

5.12 สภาพแวดล้อมในการพัฒนาเครื่องมือด้านซอฟต์แวร์

1) ระบบปฏิบัติการ Microsoft Windows 8 เป็นระบบปฏิบัติการที่ใช้ในการพัฒนาเครื่องมือ
2) IntellijI DEA 2016.2.2 และ ANTLR v4 grammar plug-in
3) จาวาเจดีเค เวอร์ชัน 8.0 (JAVA JDK 8.0)

5.2 การทดสอบ

ในการทดสอบเครื่องมือและกฎในการแปลงแผนภาพสเตทแมชชีนไปเป็นภาษาโพรเมลานั้น จะใช้กรณีศึกษา 3 กรณีศึกษาคือ แผนภาพยูเอ็มแอลสเตทแมชชีน 3 แผนภาพ คือ แผนภาพสเตทแมชชีนของระบบออเดอร์สินค้า แผนภาพสเตทแมชชีนของระบบตรวจสอบบุคคลและยานพาหนะและแผนภาพสเตทแมชชีนของระบบซื้อของออนไลน์และจ่ายเงินด้วยบัตรเครดิต เพื่อทำการทดสอบให้ครอบคลุมส่วนประกอบของแผนภาพยูเอ็มแอลสเตทแมชชีนที่เครื่องมือรองรับ โดยสามารถสรุปรายละอียดส่วนประกอบของแผนภาพสเตทแมชชีนและโอซีแอลที่ครอบคลุมในแต่ละกรณีศึกษาดังตารางที่ 5.1

กรณีศึกษาที่ 1 ระบบออเดอร์สินค้า

กรณีศึกษาที่ 2 ระบบตรวจสอบบุคคลและยานพาหนะ

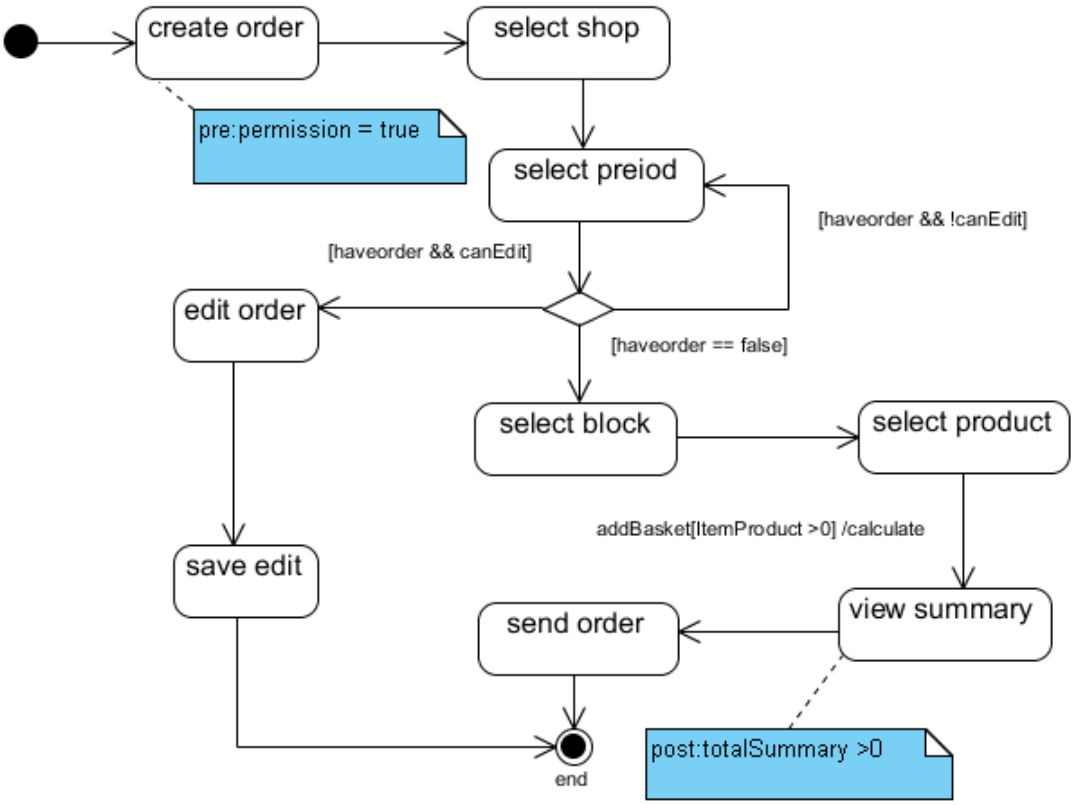
กรณีศึกษาที่ 3 ระบบซื้อของออนไลน์และจ่ายเงินด้วยบัตรเครดิต

ตารางที่ 5.1 รายละเอียดส่วนประกอบของแผนภาพสเตทแมชชีนและโอซีแอลในแต่ละกรณีศึกษา

	กรณีศึกษา		
แผนภาพสเตทแมชชีนที่มีการเขียนโอซีแอล	1	2	3
สถานะเริ่มต้น	✓	✓	✓
สถานะสิ้นสุด	✓	✓	✓
สถานะ	✓	✓	✓
ทางเลือกที่มีเส้นการเปลี่ยนสถานะเข้า 1เส้นและออก n เส้น	-	✓	✓
ทางเลือกที่มีเส้นการเปลี่ยนสถานะเข้า n เส้นและออก 1 เส้น	-	✓	
การเปลี่ยนสถานะ	✓	✓	✓
เงื่อนไขก่อน	✓	-	✓
เงื่อนไขหลัง	✓		✓
ค่ายืนยง	-	-	✓

5.3.1 แผนภาพระบบออเดอร์สินค้า

การทดสอบนี้มีวัตถุประสงค์ในการทดสอบเครื่องมือแปลงแผนภาพสเตทแมชชีนที่มีส่วนประกอบของแผนภาพสเตทแมชชีนที่งานวิจัยครอบคลุมคือ สัญลักษณ์สถานะเริ่มต้น สัญลักษณ์สถานะสิ้นสุด สัญลักษณ์สถานะ สัญลักษณ์การเปลี่ยนสถานะ เงื่อนไขก่อนและเงื่อนไขหลัง ไปเป็นภาษาโพรเมลา เพื่อทวนสอบผลลัพธ์ภาษาโพรเมลาที่ได้จากการแปลงสามารถทำงานได้สอดคล้องกับแผนภาพสเตทแมชชีนต้นทางหรือไม่และทำการทดสอบภาษาโพรเมลาด้วยเครื่องมือสปิน ภาพที่ 5.1 คือแผนภาพสเตทแมชชีนของระบบออเดอร์สินค้า



ภาพที่ 5.1 แผนภาพสเตทแมชชีนระบบออเดอร์สินค้า

ผลการทดสอบ

ส่วนของการทดสอบจะแบ่งส่วนของเนื้อหาของโพรเมลาที่ได้เป็นส่วน ๆ ดังนี้คือ ส่วนของการแปลงเงื่อนไขการ์ดและโอซีแอลเป็นตัวแปรในภาษาโพรเมลาและส่วนของการแปลงสัญลักษณ์ต่างๆ เป็นภาษาโพรเมลา จากการแปลงจะเห็นว่าได้ผลลัพธ์ครบตามที่ระบุบนแผนภาพสแตทแมชชีน ดังแสดงในภาคผนวกภาพที่ ก.1 บรรทัดที่ 1-61 เป็นส่วนของการประกาศตัวแปร ซึ่งยกตัวอย่างการแปลงส่วนของการประกาศตัวแปรที่ได้จากการแปลง เงื่อนไขการ์ดและโอซีแอลที่ปรากฏบนแผนภาพ ดังภาพที่ 5.2 แสดงตัวอย่างตัวแปร ที่ได้จากการแปลงแผนภาพสเตทแมชชีนของระบบออเดอร์สินค้า ซึ่งจากภาพที่ 5.1 พบว่ามีการเขียนโอซีแอลที่สัญลักษณ์สถานะ create order และ view summary ทำให้ได้ตัวแปรที่เกิดจากโอซีแอล 2 ตัวแปรคือ permission และ totalsummary ตามลำดับซึ่งแสดงดังบรรทัดที่ 1และ บรรทัดที่3 และมีตัวแปรที่เกิดจาก เหตุการณ์และเงื่อนไขการ์ด 4 ตัวแปร ซึ่งทำให้ได้ตัวแปรภาษาโพรเมลา ทั้งหมดดังภาพที่ 5.2

1	int totalsummary = 1	4	bool addbasket = true
2	bool canedit = true	5	int itemproduct = 1
3	bool permission = true	6	bool haveorder = true

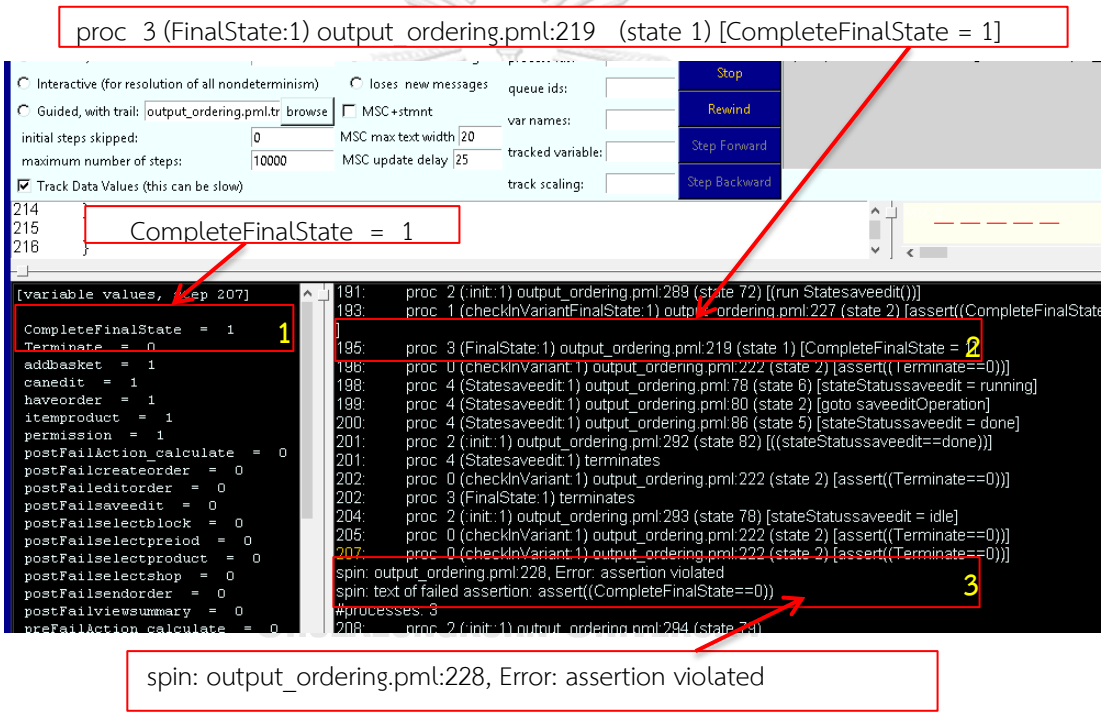
ภาพที่ 5.2 ตัวแปรที่เกิดจากเงื่อนไขก่อน เงื่อนไขหลัง เหตุการณ์และเงื่อนไขการ์ดของแผนภาพสเตทแมชชีนของระบบออเดอร์สินค้า

หลังจากแสดงให้เห็นตัวแปรที่ได้แล้วนั้น ในลำดับต่อไปเป็นส่วนของการแปลงสัญลักษณ์ของแผนภาพเป็นภาษาโพรเมลาโดยเริ่มที่การแปลงสัญลักษณ์สถานะเริ่มต้น สัญลักษณ์สถานะสิ้นสุด สัญลักษณ์สถานะ การแปลงสัญลักษณ์การเปลี่ยนสถานะ จากภาพที่ ก.1 ในภาคผนวกคือภาษาโพรเมลาที่ได้จากการแปลงซึ่งสามารถแปลงได้ถูกต้อง โดยขอยกตัวอย่างการแปลงสถานะ create order เครื่องมือสามารถแปลงได้ดังภาพที่ 5.3

```
1   proctype Statecreateorder() {
2   d_step {
3   stateStatuscreateorder = running;
4           precon:
5                   if
6                   ::(permission == true)->   goto createorderOperation ;
7                   :: else -> printf("case else")
8                    fi;
9                   if
10                  ::(!permission == true)->preFailcreateorder=true ;Terminate = 1 ;
11                  :: else -> printf("case else")
12                   fi;
13          createorderOperation:
14                  atomic{
15                          /*....Fill in details of operation....*/
16                          goto  postcon ;
17                  }
18          postcon:stateStatuscreateorder= done
19      }
21  }
```

ภาพที่ 5.3 ภาษาโพรเมลาที่ได้จากการแปลงสถานะ create order

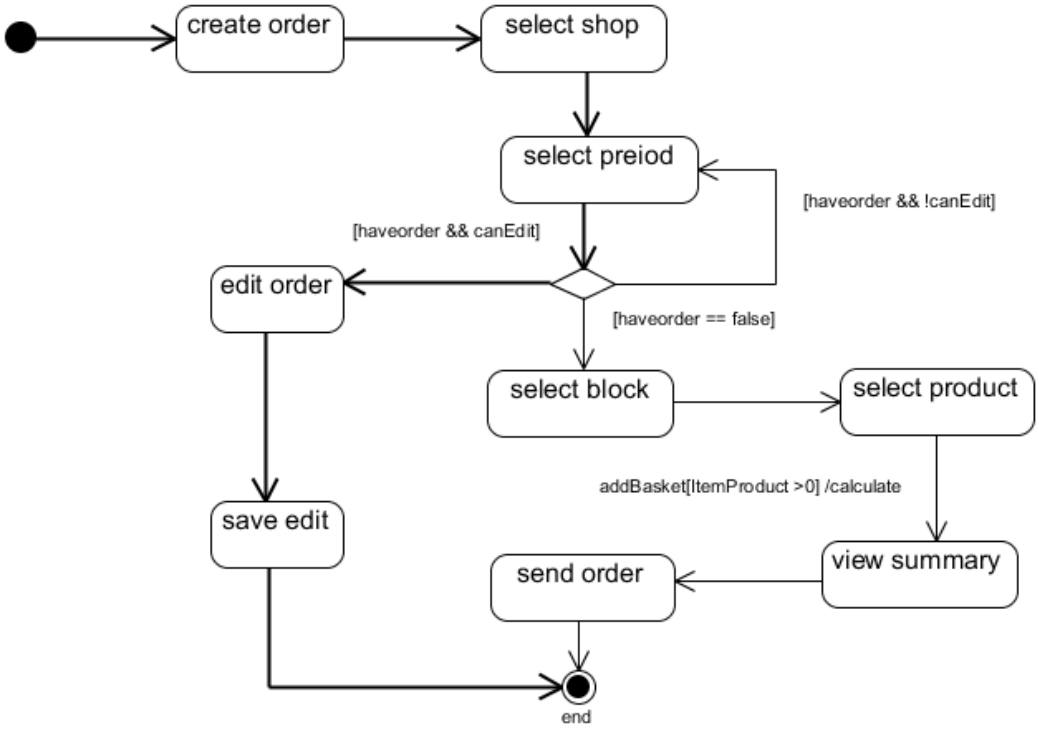
หลังจากแปลงแล้วเสร็จนำผลลัพธ์ภาษาโพรเมลาที่ได้ไปเอ็กซิคิวต์ด้วยเครื่องมือสปินและสั่งให้เครื่องมือสปินทำการจำลอง (simulation) เพื่อตรวจสอบว่าแปลงได้ถูกต้องและไม่ผิดหลักไวยากรณ์ของภาษาโพรเมลาและพิจารณาผลการจำลองพบว่าสามารถทำงานได้สอดคล้องกับแผนภาพสเตทแมชชีน โดยการตรวจสอบความสอดคล้องนั้นจะตรวจสอบว่าผลการจำลองที่เครื่องมือสปินจำลองให้นั้น เครื่องมือสปินสามารถเอ็กซิคิวต์ได้ถึงสถานะสิ้นสุดหรือไม่ ซึ่งจากแม่แบบการแปลงสถานะสิ้นสุดในที่นี้คือ FinalState หากสามารถเอ็กซิคิวต์ถึง FinalState แล้วนั้น ค่าของตัวแปร CompleteFinalState จะถูกกำหนดให้มีค่าเท่ากับหนึ่งซึ่งจากผลการจำลองพบว่าสามารถเอ็กซิคิวต์ได้ถึง FinalState ดังภาพที่ 5.4



ภาพที่ 5.4 หน้าจอผลลัพธ์การจำลองภาษาโพรเมลาที่ได้จากการแปลงแผนภาพสเตทแมชชีนของระบบออเดอร์สินค้า

จากจากภาพที่ 5.4 หมายเลข 1 แสดงให้เห็นว่าค่าของตัวแปร CompleteFinalState มีค่าเท่ากับหนึ่ง ซึ่งเกิดจากการเอ็กซิคิวต์คำสั่งหมายเลข 2 ทำให้เครื่องมือสปินหยุดการจำลองดังที่แสดงให้เห็นในหมายเลข 3 และภาคผนวกภาพที่ ก.2 แสดงผลการจำลองการเอ็กซิคิวต์ภาษาโพรเมลาที่ได้จากการแปลง โดยจะเห็นการเอ็กซิคิวต์จาก proctype ที่แทนสัญลักษณ์สถานะเริ่มต้นถึง proctype ที่แทนสัญลักษณ์สถานะสิ้นสุด ซึ่งเห็นว่าเมื่อนำผลลัพธ์ภาษาโพรเมลามาเทียบกับแผนภาพสเตทแมชชีนแล้วพบว่าผลของการเปลี่ยนสถานะของแผนภาพสเตทแมชชีนดังกล่าว กับผล

ของการจำลองของเครื่องมือสปินนั้นสอดคล้องกัน โดยในภาพที่ 5.5 คือผลการเปลี่ยนสถานะที่ได้เมื่อเทียบกับภาษาโพรเมลาที่ได้จากการแปลงแผนภาพสเตทแมชชีนของระบบออเดอร์สินค้า



ภาพที่ 5.5 เส้นทางการเปลี่ยนสถานะที่ได้เมื่อเทียบกับภาษาโพรเมลาที่ได้จากการแปลงแผนภาพสเตทแมชชีนของระบบออเดอร์สินค้า

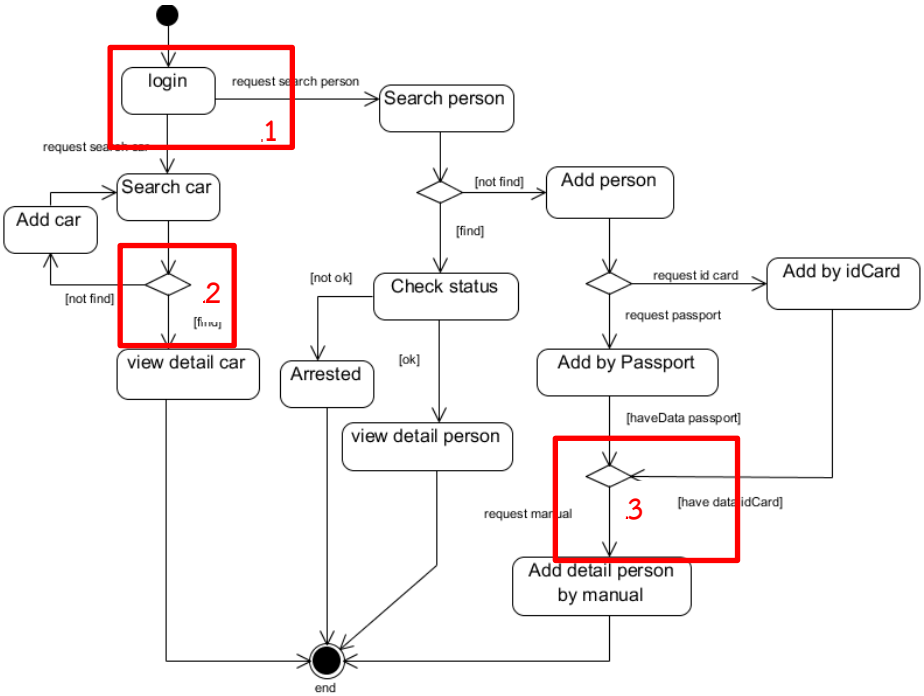
5.3.2 แผนภาพระบบตรวจสอบบุคคลและยานพาหนะ

การทดสอบนี้มีวัตถุประสงค์ในการทดสอบแม่แบบโครงสร้างภาษาโพรเมลาสำหรับการสัญลักษณ์การเปลี่ยนสถานะและสัญลักษณ์ทางเลือกโดยการทดสอบจะทำการวาดแผนภาพสเตทแมชชีนของระบบตรวจสอบบุคคลและยานพาหนะ จากนั้นทำการแปลงโดยใช้เครื่องมือที่พัฒนาจากนั้นนำภาษาโพรเมลาที่ได้เข้าเครื่องมือสปิน และทำการทวนสอบ ภาพที่ 5.6 คือรูปแผนภาพสเตทแมชชีนของระบบตรวจสอบบุคคลและยานพาหนะหมายเลข 1-3 คือส่วนที่ต้องการทดสอบโดย มีรายละเอียดดังนี้

1) หมายเลข 1 ในภาพที่ 5.6 แสดงถึงการเปลี่ยนสถานะจากสถานะหนึ่งไปหลายสถานะ

2) หมายเลข 2 ในภาพที่ 5.6 แสดงถึงทางเลือกที่มีสัญลักษณ์ทางเลือกที่มีเส้นการเปลี่ยนสถานะเข้า 1 เส้นและออก n เส้น

3) หมายเลข 3 ในภาพที่ 5.6 แสดงถึงทางเลือกที่มีสัญลักษณ์ทางเลือกที่มีเส้นการเปลี่ยนสถานะเข้า n เส้นและออก 1 เส้น



ภาพที่ 5.6 แผนภาพสเตทแมชชีนของระบบตรวจสอบบุคคลและยานพาหนะ

ผลการทดสอบ

ผลการทดสอบพบว่าหลังจากทำการแปลงแผนภาพแล้วได้ภาษาโพรเมลาที่ได้สอดคล้องตามแม่แบบโครงสร้างภาษาโพรเมลาดัง ภาพที่ 5.7, 5.8 และ5.9 โดยภาพที่ 5.7 แสดงการเปลี่ยนสถานะจากสถานะหนึ่งไปหลายสถานะ ภาพที่ 5.8 แสดงถึงสัญลักษณ์ทางเลือกที่มีเส้นการเปลี่ยนสถานะเข้า 1 เส้นและออก n เส้น และภาพที่ 5.9 แสดงถึงสัญลักษณ์ทางเลือกที่มีเส้นการเปลี่ยนสถานะเข้า n เส้นและออก 1 เส้นซึ่งจากภาษาโพรเมลาที่ได้จากการแปลงนั้นพบว่าถูกต้องตามแม่แบบโครงสร้างภาษาโพรเมลาที่ได้ออกแบบไว้และเมื่อนำไปเอ็กซิคิวต์ด้วยเครื่องเมื่อสปินสามารถเอ็กซิคิวต์ proctype ได้จนถึง proctype FinalState ดังภาพที่ 5.10

```
1  login:
2                   if
3                   ::(stateStatuslogin== idle)->run Statelogin( )
4                   :: else -> printf("case else")
5                    fi;
6                   if
7                  ::(stateStatuslogin== done &&requestsearchperson)->stateStatuslogin=idle ;
8                   goto  Searchperson
9                   :: else -> printf("case else")
10                   fi;
11                  if
12                  ::(stateStatuslogin== done &&requestsearchcar)->stateStatuslogin=idle ;
13                  goto  Searchcar
14                  :: else -> printf("case else")
15                   fi;
16        goto login;
```

ภาพที่ 5.7 ภาษาโพรเมลาที่ได้จากการแปลงการเปลี่ยนสถานะจากสถานะหนึ่งไปหลายสถานะ

```
1         Searchcar:
2                   if
3                   ::(stateStatusSearchcar== idle)->run StateSearchcar( )
4                   :: else -> printf("case else")
5                   fi;
6                   if
7                   ::(stateStatusSearchcar== done)->
8                             if
9                             ::(find)-> stateStatusSearchcar=idle ;goto  viewdetailcar
11                            :: else -> printf("case else")
12                            fi;
13                           if
14                            ::(notfind)-> stateStatusSearchcar=idle ;goto  Addcar
16                            :: else -> printf("case else")
17                            fi;
18                  :: else -> printf("case else")
19                  fi;
20                  goto  Searchcar;
```

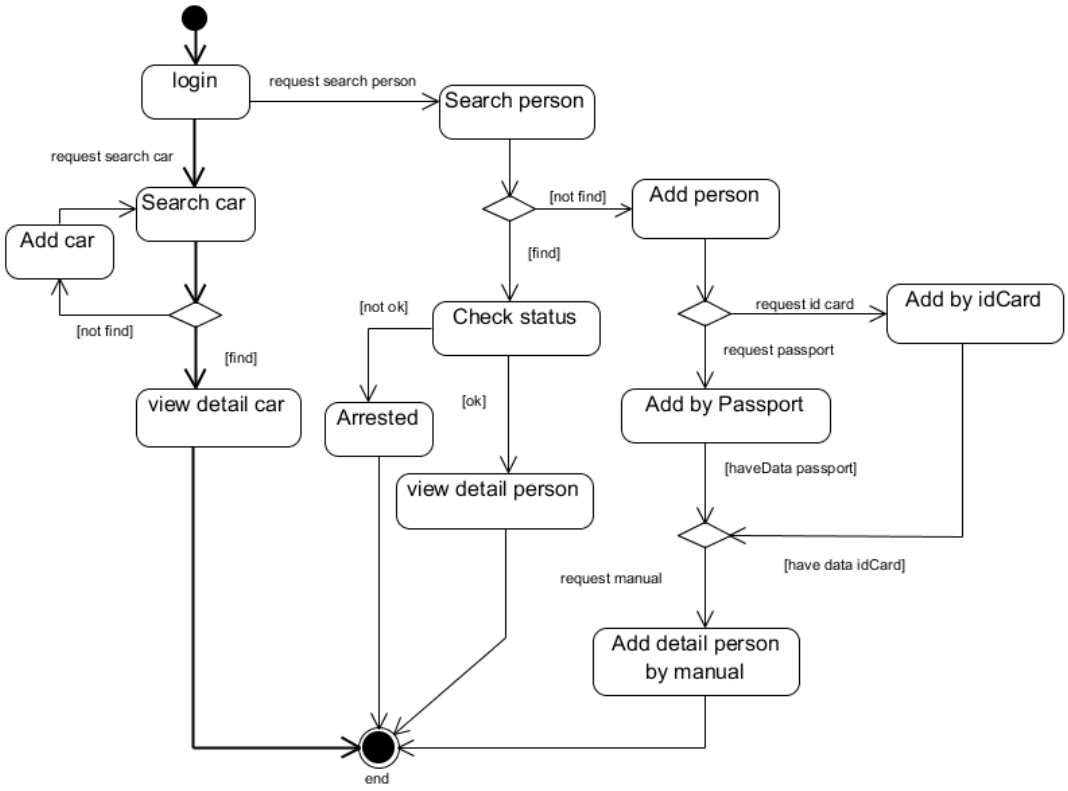
ภาพที่ 5.8 ภาษาโพรเมลาที่ได้จากการแปลงทางเลือกที่มีเส้นการเปลี่ยนสถานะเข้า 1 เส้นและออก n เส้น

```
1       AddbyPassport:
2               if
3               ::(stateStatusAddbyPassport== idle)->run StateAddbyPassport( )
4               :: else -> printf("case else")
5                fi;
6               if
7               ::(stateStatusAddbyPassport== done &&havedatapassport)->
8                       if
9                       ::(requestmanual)-> stateStatusAddbyPassport=idle ;
10                        goto  Adddetailpersonbymanual
11                      :: else -> printf("case else")
12                      fi;
13              :: else -> printf("case else")
14              fi;
15      goto  AddbyPassport;
16      AddbyidCard:
17              if
18              ::(stateStatusAddbyidCard== idle)->run StateAddbyidCard( )
19              :: else -> printf("case else")
20               fi;
21              if
22              ::(stateStatusAddbyidCard== done &&havedataidcard)->
23                      if
24                      ::(requestmanual)-> stateStatusAddbyidCard=idle ;
25                        goto  Adddetailpersonbymanual
26                      :: else -> printf("case else")
27                      fi;
28              :: else -> printf("case else")
29      fi;
30 goto  AddbyidCard;
```

ภาพที่ 5.9 ภาษาโพรเมลาที่ได้จากการแปลงทางเลือกที่มีเส้นการเปลี่ยนสถานะเข้า n เส้นและออก 1 เส้น

เมื่อนำภาษาโพรเมลาได้ไปจำลองได้ผลจากผลการจำลองดังภาพที่ ก.4 ในภาคผนวก จะเห็นว่าผลการจำลองสถานะที่เปลี่ยนไปแต่ละสถานะนั้นจากสถานะเริ่มต้นถึงสถานะสิ้นสุดบนแผนภาพ

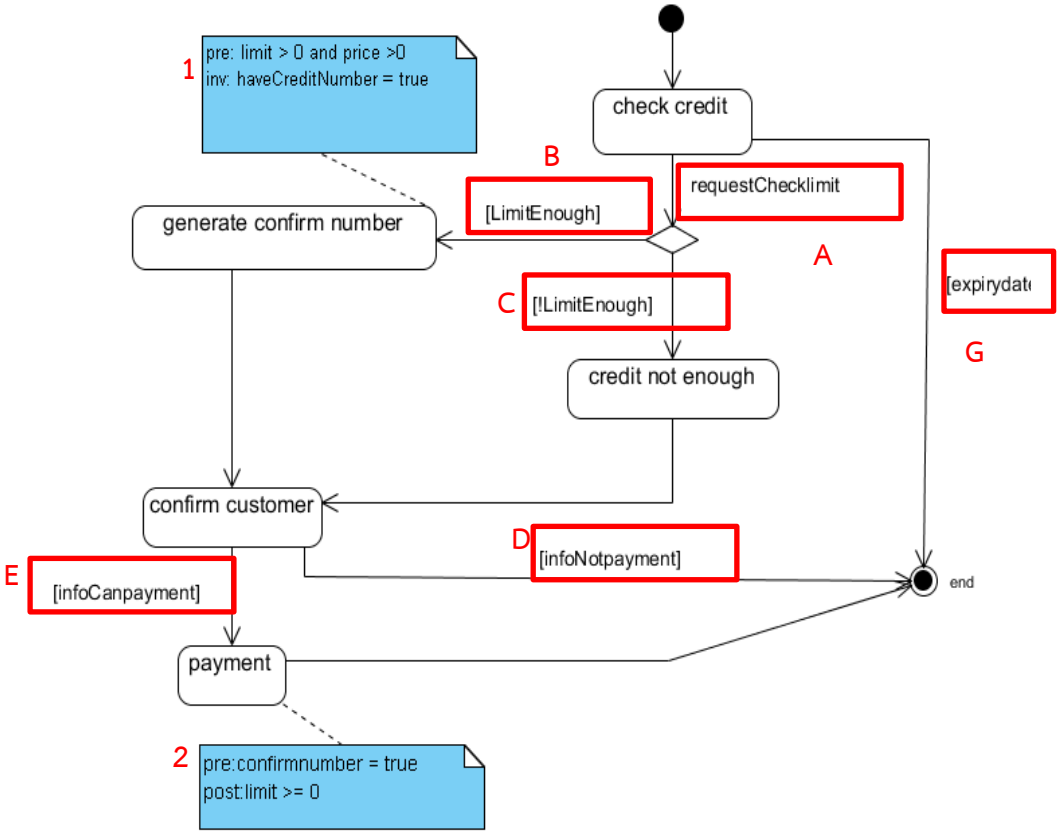
สเตทแมชชีนที่สอดคล้องกับภาษาโพรเมลาที่ได้คือเส้นทึบสีดำดังภาพที่ 5.10 บ่งบอกถึงภาษาโพรเมลาที่แปลงมาได้นั้นเมื่อนำมาเอ็กซิคิวต์ด้วยเครื่องมือสปินแล้วนั้นสามารถทำงานได้โดยไม่ผิดไวยากรณ์ของภาษาโพรเมลาและสามารถเอ็กซิคิวต์ได้ถึง proctype FinalState



ภาพที่ 5.10 เส้นทางการเปลี่ยนสถานะที่ได้เมื่อเทียบกับลำดับการเอ็กซิคิวต์ภาษาโพรเมลาที่ได้จากการแปลงแผนภาพสเตทแมชชีนของระบบตรวจสอบบุคคลและยานพาหนะ

5.3.3 แผนภาพระบบซื้อของออนไลน์โดยจ่ายเงินด้วยบัตรเครดิต

การทดสอบนี้มีวัตถุประสงค์สำหรับทดสอบตัวแปรเงื่อนไขการ์ดและโอซีแอลบนแผนภาพสแตทแมชชีนซึ่งต้องการตรวจสอบว่าหลังจากการแปลงแผนภาพแล้วตัวแปรการ์ดและโอซีแอลบนแผนภาพนั้นปรากฏบนภาษาโพรเมลาครบถ้วนและสอดคล้องกับแผนภาพต้นทางหรือไม่ โดยแผนภาพระบบซื้อของออนไลน์โดยจ่ายเงินด้วยบัตรเครดิต ดังภาพที่ 5.11 โดยหมายเลข 1 แสดงถึงเงื่อนไขก่อนและค่ายืนยง หมายเลข 2 แสดงถึงเงื่อนไขก่อนและเงื่อนไขหลัง B, C, D, E ,G แสดงถึงเงื่อนไขการ์ด และ A แสดงถึงเหตุการณ์



ภาพที่ 5.11 แผนภาพสเตทแมชชีนระบบซื้อของออนไลน์โดยจ่ายเงินด้วยบัตรเครดิต

ผลการทดสอบ

จากผลการทดสอบจะเห็นว่าเครื่องมือสามารถแปลงแผนภาพสเตทแมชชีนเป็นภาษาโพรเมลาได้ครบตามที่ต้องการทดสอบโดยจากภาพที่ 5.12 แสดงให้เห็นถึงการแปลงเงื่อนไขก่อนและเงื่อนไขหลังที่เขียนอยู่บนสถานะ payment ได้อย่างถูกต้อง ภาพที่ 5.13 แสดงภาษาโพรเมลาที่ได้จากการแปลงค่ายืนยง และภาพที่ 5.14 แสดงภาษาโพรเมลาที่ได้จากการแปลงเงื่อนไขการ์ด จะเห็นได้ว่าทั้ง ภาพที่ 5.12, 5.13 และภาพที่ 5.14 แสดงให้เห็นว่าภาษาโพรเมลาที่ได้จากการแปลงนั้นเครื่องมือสามารถแปลงมาได้อย่างครบถ้วน

```
1       proctype Statepayment() {
2       d_step {
3       stateStatuspayment = running;
4               precon:
```

ภาพที่ 5.12 ภาษาโพรเมลาที่ได้จากการแปลงสถานะpayment ที่มีการเขียนเงื่อนไขก่อนและเงื่อนไขหลัง

```
5                               if
6                               ::(confirmnumber == true)->   goto paymentOperation ;
7                               :: else -> printf("case else")
8                                fi;
9                               if
10                              ::(!confirmnumber == true)->preFailpayment=true ;Terminate = 1 ;
11                              :: else -> printf("case else")
12                               fi;
13                  paymentOperation:
14                              atomic{
15                                      /*....Fill in details of operation....*/
16                                      goto  postcon ;
17                              }
18                  postcon:
19                              if
20                              ::(limit >= 0)-> stateStatuspayment= done
21                              :: else -> printf("case else")
22                               fi;
23                              if
24                              ::(!limit >= 0)->postFailpayment=true ;Terminate = 1 ;
25                              :: else -> printf("case else")
26                               fi;
27              }
28      }
```

ภาพที่ 5.12 ภาษาโพรเมลาที่ได้จากการแปลงสถานะpayment ที่มีการเขียนเงื่อนไขก่อนและเงื่อนไขหลัง (ต่อ)

```
1       active proctype invhavecreditnumber()
2        { do
3           ::assert(havecreditnumber== true)
4           od;
5       }
```

ภาพที่ 5.13 ภาษาโพรเมลาที่ได้จากการแปลงค่ายืนยง

```
1       init {
2       checkcredit:
3               if
4               ::(stateStatuscheckcredit== idle)->run Statecheckcredit( )
5               :: else -> printf("case else")
6                fi;
7               if                                                    A
8               ::(stateStatuscheckcredit== done &&requestchecklimit)- >
9                       if
10                C     ::(!limitenough)-> stateStatuscheckcredit=idle ;
11                      goto  creditnotenough
12                      :: else -> printf("case else")
13                      fi;
14                      if
15                      limitenough)-> stateStatuscheckcredit=idle ;
                  B
16                      goto  generateconfirmnumber
17                      :: else -> printf("case else")
18                      fi;
19              :: else -> printf("case else")
20              fi;
21              if                                                    G
22              ::(stateStatuscheckcredit== done &&expirydate)->stateStatuscheckcredit=idle ;
23              run  FinalState()
24              :: else -> printf("case else")
25               fi;
26              goto  checkcredit;
```


ภาพที่ 5.14 เงื่อนไขการ์ดและเหตุการณ์ที่ปรากฏบนเส้นเปลี่ยนสถานะของภาพที่ 5.11

```
1       confirmcustomer:
2               if
3               ::(stateStatusconfirmcustomer== idle)->run Stateconfirmcustomer( )
4               :: else -> printf("case else")
5                fi;
6               if                                                          E
7               ::(stateStatusconfirmcustomer== done &&infocanpayment)
8                  ->stateStatusconfirmcustomer=idle ;
9               goto  payment
10              :: else -> printf("case else")
11               fi;
12              if                                                  D
13              ::(stateStatusconfirmcustomer== done &&infonotpayment)
14                 ->stateStatusconfirmcustomer=idle ; run  FinalState()
15              :: else -> printf("case else")
16             fi;
17      goto confirmcustomer;

18      payment:
19              if
20              ::(stateStatuspayment== idle)->run Statepayment( )
21              :: else -> printf("case else")
22              fi;
23              if
24              ::(stateStatuspayment== done)->stateStatuspayment=idle ; run FinalState()
25              :: else -> printf("case else")
26              fi;
27      goto  payment;
28      }
```

ภาพที่ 5.14 เงื่อนไขการ์ดและเหตุการณ์ที่ปรากฏบนเส้นเปลี่ยนสถานะของภาพที่ 5.11 (ต่อ)

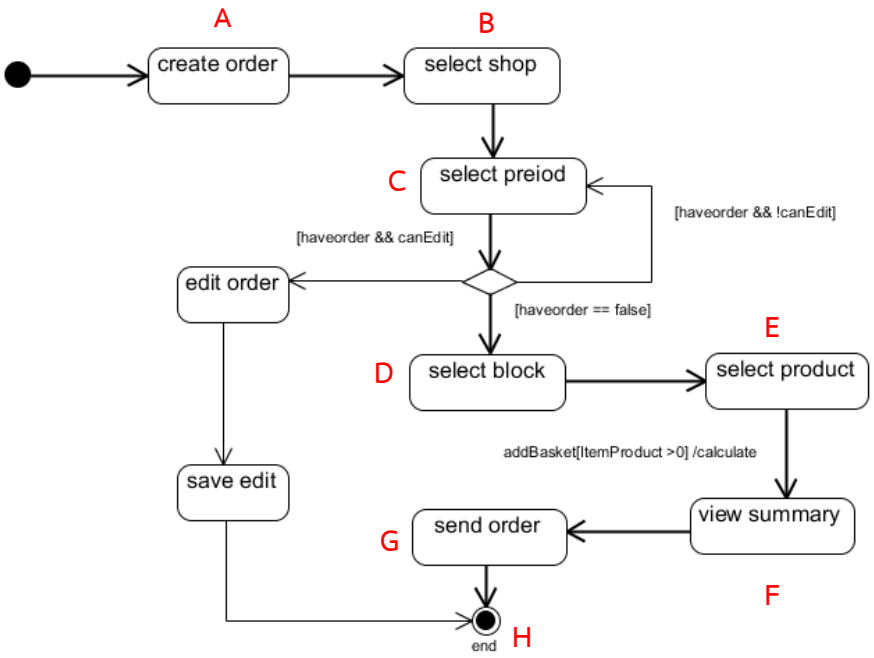
5.3 ตัวอย่างการเพิ่มรายละเอียดในส่วนของโอเปอเรชัน

หลังจากทดสอบความถูกต้องของการแปลงแผนภาพสเตทแมชชีนเป็นภาษาโพรเมลาดังกล่าวตามกรณีศึกษาแล้วนั้น ในส่วนนี้จะแสดงตัวอย่างการเพิ่มรายละเอียดบนภาษาโพรเมลาที่ได้

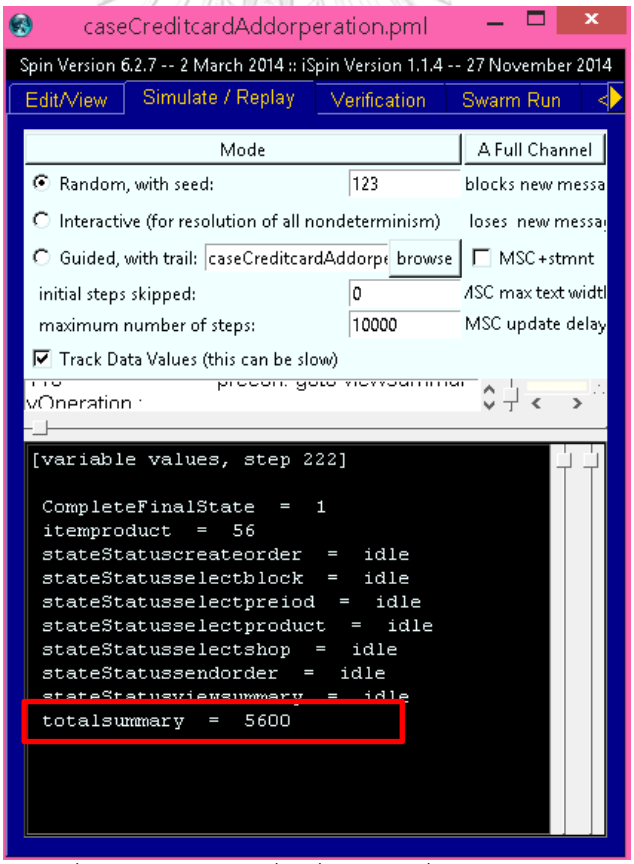
โดยใช้ภาษาโพรเมลาที่ได้จากการแปลงแผนภาพสเตทแมชชีนของระบบออเดอร์สินค้า โดยจะทำการเพิ่มข้อมูลรายละเอียดดังภาพที่ 5.15 โดยจากตัวอย่างบรรทัดที่ 8 เป็นการกำหนดค่าให้กับตัวแปร itemproduct และบรรทัดที่ 9 เป็นการคำนวณค่าให้กับตัวแปร totalsummary และก่อนการเอ็กซ์คิวต์ได้ทำการเปลี่ยนค่าตัวแปร haveorder เท่ากับ false เพื่อดูเส้นทางการเปลี่ยนสถานะซึ่งเมื่อสั่งเอ็กซิคิวต์ เส้นทางการเปลี่ยนสถานะของแผนภาพสเตทแมชชีนที่ได้ ต้องเป็นตามเส้นสีดำทึบดังภาพที่ 5.16 และค่าตัวแปร totalsummary ต้องมีค่าเท่ากับ 5600 ดังภาพที่ 5.17 โดยภาพที่ 5.16 แสดงถึงการเปลี่ยนสถานะของแผนภาพระบบออเดอร์สินค้าที่สอดคล้องกับการเอ็กซิคิวต์ภาษาโพรเมลาที่ได้จากการแปลงแผนภาพสเตทแมชชีนของระบบออเดอร์สินค้า หลังจากการทดสอบเปลี่ยนค่าตัวแปรและเพิ่มรายละเอียดใน proctype Stateviewsummary โดยดูจากลำดับการเอ็กซิคิวต์ภาษาโพรเมลาในภาพที่ 5.18 ซึ่งเมื่อเทียบกับเส้นทางการเปลี่ยนสถานะของแผนภาพสเตทแมชชีนบนแผนภาพนั้นสอดคล้องกันดัง Path :A-B-C-D-E-F-G-H บนแผนภาพที่ 5.16 และภาพที่ 5.18

```
1   proctype Stateviewsummary() {
2   d_step {
3   stateStatusviewsummary = running;
4           precon: goto viewsummaryOperation ;
5           viewsummaryOperation:
6                   atomic{
7                       /*....Fill in details of operation....*/
8                       itemproduct = 56;
9                       totalsummary = itemproduct *100;
10                      goto  postcon ;
11                  }
12          postcon:
13                  if
14                  ::(totalsummary >0)-> stateStatusviewsummary= done
15                  :: else -> printf("case else")
15                   fi;
16                  if
17                  ::(!totalsummary >0)->postFailviewsummary=true ;Terminate = 1 ;
18                  :: else -> printf("case else")
19                   fi;
20          }
21  }
```

ภาพที่ 5.15 ตัวอย่างการเพิ่มรายละเอียดในส่วนของโอเปอเรเตอร์



ภาพที่ 5.16 การเปลี่ยนสถานะของแผนภาพระบบออเดอร์สินค้าหลังจากการเปลี่ยนค่าตัวแปรและเพิ่มรายละเอียดใน proctype Stateviewsummary



ภาพที่ 5.17 ค่าตัวแปรที่เปลี่ยนหลังเพิ่มส่วนโอเปอเรชัน

จะเห็นว่าหลังการเพิ่มส่วนโอเปอเรชันใน proctype แล้วเมื่อเครื่องมือสปินทำการเอ็กซิคิวต์ทำให้ค่าของตัวแปรเปลี่ยนดังภาพที่ 5.17

```
 4:    proc  2 (:init::1) creates proc  3 (Statecreateorder)
 5:    proc  2 (:init::1) caseCreditcardAddorperation.pml:231 (state 2)    [(run Statecreateorder())]
```
A

```
:
60:    proc  2 (:init::1) creates proc  3 (Stateselectshop)
61:    proc  2 (:init::1) caseCreditcardAddorperation.pml:297 (state 86)    [(run Stateselectshop())]
```
B

```
:
87:    proc  2 (:init::1) creates proc  3 (Stateselectpreiod)
87:    proc  2 (:init::1) caseCreditcardAddorperation.pml:308 (state 100)    [(run Stateselectpreiod())]
```
C

```
:
114:    proc  2 (:init::1) creates proc  3 (Stateselectblock)
114:    proc  2 (:init::1) caseCreditcardAddorperation.pml:242 (state 16)    [(run Stateselectblock())]
```
D

```
:
146:    proc  2 (:init::1) creates proc  3 (Stateselectproduct)
146:    proc  2 (:init::1) caseCreditcardAddorperation.pml:253 (state 30)    [(run Stateselectproduct())]
```
E

```
:
156:    proc  2 (:init::1) creates proc  3 (Stateviewsummary)
156:    proc  2 (:init::1) caseCreditcardAddorperation.pml:264 (state 44)    [(run Stateviewsummary())]
```
F

```
:
179:    proc  2 (:init::1) creates proc  3 (Statesendorder)
179:    proc  2 (:init::1) caseCreditcardAddorperation.pml:275 (state 58)    [(run Statesendorder())]
```
G

```
:
194:    proc  2 (:init::1) creates proc  3 (FinalState)
194:    proc  2 (:init::1) caseCreditcardAddorperation.pml:280 (state 65)    [(run FinalState())]
```
H

ภาพที่ 5.18 ผลการจำลองภาษาโพรเมลาที่ได้จากการแปลงแผนภาพสเตทแมชชีนของระบบออเดอร์สินค้าหลังเพิ่มโอเปอเรชัน

บทที่ 6
สรุปผลวิจัยและข้อเสนอแนะ

6.1 สรุปผลการวิจัย

งานวิจัยนี้ได้นำเสนอแม่แบบโครงสร้างภาษาโพรเมลาและได้พัฒนาและออกแบบเครื่องมือที่ใช้สำหรับการแปลงแผนภาพสเตทแมชชีนที่มีการเขียนโอซีแอลไปเป็นภาษาโพรเมลาเพื่อเป็นทางเลือกหนึ่งสำหรับผู้ที่ต้องการทวนสอบระบบด้วยเครื่องมือสปิน

สำหรับการทวนสอบแม่แบบโครงสร้างภาษาโพรเมลาและเครื่องมือที่พัฒนานั้นใช้กรณีศึกษา 3 กรณี ซึ่งแต่ละกรณีเน้นการแปลงองค์ประกอบของแผนภาพสเตทแมชชีนที่มีโอซีแอลไปเป็นภาษาโพรเมลาในด้านความครบถ้วนของตัวแปรต่างๆ ที่ปรากฏบนแผนภาพและสามารถนำภาษโพรเมลาที่ได้จากการแปลงไปเอ็กซิคิวต์กับเครื่องมือสปินได้ในเบื้องต้นและผู้ใช้สามารถเพิ่มรายละเอียดต่างๆ เพิ่มเติมในส่วนที่กำหนดไว้ให้ได้

ผลที่ได้จากงานวิจัยนี้คือได้เครื่องมือการแปลงแผนภาพสเตมแมชชีนและแม่แบบในการแปลงแผนภาพสเตทแมชชีนเป็นภาษาโพรเมลา โดยผลลัพธ์ที่เครื่องมือแปลงได้สามารถนำไปทวนสอบด้วยเครื่องมือสปินได้โดยไม่มีข้อผิดพลาดด้านไวยากรณ์และโครงสร้างการทำงานของแผนภาพสเตทแมชชีน

6.2 ข้อเสนอแนะ

เครื่องมือในการแปลงแผนภาพสเตทแมชชีนที่มีการเขียนโอซีแอลยังสามารถพัฒนาเพิ่มเติมได้ในส่วนของการแปลงสัญลักษณ์ให้ครอบคลุมสัญลักษณ์อื่นๆ ในแผนภาพสเตทแมชชีน

6.3 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้แม่แบบโครงสร้างภาษาโพรเมลาสำหรับใช้ในการแปลงแผนภาพสเตทแมชชีนที่มีการเขียนโอซีแอลไปเป็นภาษาโพรเมลา
2) ได้เครื่องมือกึ่งอัตโนมัติในการแปลงแผนภาพสเตทแมชชีนที่มีการเขียนโอซีแอลไปเป็นโพรเมลา

รายการอ้างอิง

[1] Holzmann, G., *Spin model checker, the: primer and reference manual*. 2003: Addison-Wesley Professional.

[2] Schäfer, T., A. Knapp, and S. Merz, *Model checking UML state machines and collaborations.* Electronic Notes in Theoretical Computer Science, 2001. 55(3): p. 357-369.

[3] Lilius, J. and I.P. Paltor. *vUML: A tool for verifying UML models*. in *Automated Software Engineering, 1999. 14th IEEE International Conference on.* 1999. IEEE.

[4] Files, A.N.M.C., *OMG Unified Modeling Language TM (OMG UML).*

[5] *Booking a Movie Ticket Online*. 2011.

[6] Group, O.M. *Object constraint language version 2.4*. 2014.

[7] Parr, T., *The definitive ANTLR 4 reference*. 2013: Pragmatic Bookshelf.

[8] Latella, D., I. Majzik, and M. Massink, *Automatic verification of a behavioural subset of UML statechart diagrams using the SPIN model-checker.* Formal aspects of computing, 1999. 11(6): p. 637-664.

[9] Mikk, E., et al. *Implementing statecharts in PROMELA/SPIN*. in *Industrial Strength Formal Specification Techniques, 1998. Proceedings. 2nd IEEE Workshop on*. 1998. IEEE.

[10] Mikk, E., Y. Lakhnechi, and M. Siegel. *Hierarchical automata as model for statecharts*. in *Annual Asian Computing Science Conference*. 1997. Springer.

[11] Lilius, J. and I.P. Paltor. *Formalising UML state machines for model checking*. in *International Conference on the Unified Modeling Language*. 1999. Springer.

ภาคผนวก ก ภาษาโพรเมลาที่ได้จากการแปลงและผลการจำลอง

```
1   mtype ={idle ,running ,done}
2   /*---------- variable of Stateselectshop----------*/
3   mtype stateStatusselectshop= idle
4   bool preFailselectshop= false
5   bool postFailselectshop= false
6
7   /*---------- variable of Statesaveedit----------*/
8   mtype stateStatussaveedit= idle
9   bool preFailsaveedit= false
10  bool postFailsaveedit= false
11
12  /*---------- variable of Stateeditorder----------*/
13  mtype stateStatuseditorder= idle
14  bool preFaileditorder= false
15  bool postFaileditorder= false
16
17  /*---------- variable of Statesendorder----------*/
18  mtype stateStatussendorder= idle
19  bool preFailsendorder= false
20  bool postFailsendorder= false
21
22  /*---------- variable of Stateviewsummary----------*/
23  mtype stateStatusviewsummary= idle
24  bool preFailviewsummary= false
25  bool postFailviewsummary= false
26
27  /*---------- variable of Stateselectproduct----------*/
28  mtype stateStatusselectproduct= idle
29  bool preFailselectproduct= false
30  bool postFailselectproduct= false
32  /*---------- variable of Stateselectblock----------*/
```

```
33      mtype stateStatusselectblock= idle
34      bool preFailselectblock= false
35      bool postFailselectblock= false
36
37      /*---------- variable of Stateselectpreiod----------*/
38      mtype stateStatusselectpreiod= idle
39      bool preFailselectpreiod= false
40      bool postFailselectpreiod= false
41
42      /*---------- variable of Statecreateorder----------*/
43      mtype stateStatuscreateorder= idle
44      bool preFailcreateorder= false
45      bool postFailcreateorder= false
46
47      /*---------- variable of StateAction_calculate----------*/
48      mtype stateStatusAction_calculate= idle
49      bool preFailAction_calculate= false
50      bool postFailAction_calculate= false
51
52      bit Terminate = 0 ; /*abnormal terminate */
53      bit CompleteFinalState = 0 ;
54
55      bool requestorder = true
56      int totalsummary = 1
57      bool canedit = true
58      bool permission = true
59      bool addbasket = true
60      int itemproduct = 1
61      bool haveorder = true
62
63      proctype Stateselectshop() {
```

```
64      d_step {
65      stateStatusselectshop = running;
66              precon: goto selectshopOperation ;
67              selectshopOperation:
68                      atomic{
69                              /*....Fill in details of operation....*/
70                              goto  postcon ;
71                      }
72              postcon:stateStatusselectshop= done
73              }
74      }
75
76      proctype Statesaveedit() {
77      d_step {
78      stateStatussaveedit = running;
79              precon: goto saveeditOperation ;
80              saveeditOperation:
81                      atomic{
82                              /*....Fill in details of operation....*/
83                              goto  postcon ;
84                      }
85              postcon:stateStatussaveedit= done
86              }
87      }
88
89      proctype Stateeditorder() {
90      d_step {
91      stateStatuseditorder = running;
92              precon: goto editorderOperation ;
93              editorderOperation:
94                      atomic{
```

```
95                              /*....Fill in details of operation....*/
96                              goto  postcon ;
97                      }
98              postcon:stateStatuseditorder= done
99                }
100     }
101
102     proctype Statesendorder() {
103     d_step {
104     stateStatussendorder = running;
105             precon: goto sendorderOperation ;
106             sendorderOperation:
107                     atomic{
108                             /*....Fill in details of operation....*/
109                             goto  postcon ;
110                     }
111             postcon:stateStatussendorder= done
112             }
113     }
114
115     proctype Stateviewsummary() {
116     d_step {
117     stateStatusviewsummary = running;
118             precon: goto viewsummaryOperation ;
119             viewsummaryOperation:
120                     atomic{
121                             /*....Fill in details of operation....*/
122                             goto  postcon ;
123                              }
124             postcon:
125                     if
```

```
126                     ::(totalsummary >0)-> stateStatusviewsummary= done
127                     :: else -> printf("case else")
128                      fi;
129                     if
130                     ::(!totalsummary >0)->postFailviewsummary=true ;Terminate =
1 ;
131                     :: else -> printf("case else")
132                      fi;
133             }
134     }
135
136     proctype Stateselectproduct() {
137     d_step {
138     stateStatusselectproduct = running;
139             precon: goto selectproductOperation ;
140             selectproductOperation:
141                     atomic{
142                             /*....Fill in details of operation....*/
143                             goto  postcon ;
144                     }
145             postcon:stateStatusselectproduct= done
146             }
147     }
148
149     proctype Stateselectblock() {
150     d_step {
151     stateStatusselectblock = running;
152             precon: goto selectblockOperation ;
153             selectblockOperation:
154                     atomic{
155                             /*....Fill in details of operation....*/
```

```
156                             goto  postcon ;
157                     }
158             postcon:stateStatusselectblock= done
159               }
160     }
161
162     proctype Stateselectpreiod() {
163     d_step {
164     stateStatusselectpreiod = running;
165             precon: goto selectpreiodOperation ;
166             selectpreiodOperation:
167                     atomic{
168                             /*....Fill in details of operation....*/
169                             goto  postcon ;
170                     }
171             postcon:stateStatusselectpreiod= done
172           }
173
174     }
175
176     proctype Statecreateorder() {
177     d_step {
178     stateStatuscreateorder = running;
179             precon:
180               if
181               ::(permission == true)->   goto createorderOperation ;
182               :: else -> printf("case else")
183               fi;
184               if
185               ::(!permission == true)->preFailcreateorder=true ;Terminate = 1 ;
186               :: else -> printf("case else")
```

```
187                     fi;
188             createorderOperation:
189                     atomic{
190                             /*....Fill in details of operation....*/
191                             goto  postcon ;
192                     }
193
194             postcon:stateStatuscreateorder= done
195         }
196
197     }
198
199     proctype StateAction_calculate() {
200     d_step {
201     stateStatusAction_calculate = running;
202             precon: goto Action_calculateOperation ;
203             Action_calculateOperation:
204                     atomic{
205                             /*....Fill in details of operation....*/
206                             goto  postcon ;
207                     }
208             postcon:stateStatusAction_calculate= done
209         }
210
211     }
213     proctype FinalState() {
214              CompleteFinalState = 1;
215     }
216     active proctype checkInVariant() {
217             do
218             :: assert(Terminate==0);
```

```
219             od ;
220     }
221     active proctype checkInVariantFinalState() {
222             do
223             :: assert(CompleteFinalState==0);
224             od ;
225     }
226     init {
227     createorder:
228             if
229             ::(stateStatuscreateorder== idle)->run Statecreateorder( )
230             :: else -> printf("case else")
231             fi;
232             if
233             ::(stateStatuscreateorder== done)->stateStatuscreateorder=idle ;
234             goto  selectshop
235             :: else -> printf("case else")
236             fi;
237     goto  createorder;
238     selectblock:
239             if
240             ::(stateStatusselectblock== idle)->run Stateselectblock( )
241             :: else -> printf("case else")
242             fi;
243             if
244             ::(stateStatusselectblock== done)->stateStatusselectblock=idle ;
245             goto  selectproduct
246             :: else -> printf("case else")
247             fi;
248     goto  selectblock;
249     selectproduct:
```

```
250             if
251             ::(stateStatusselectproduct== idle)->run Stateselectproduct( )
252             :: else -> printf("case else")
253              fi;
254             if
255             ::(stateStatusselectproduct== done &&itemproduct >0 )
                 ->stateStatusselectproduct=idle ;
256             goto viewsummary;
257             :: else -> printf("case else")
258              fi;
259             goto selectproduct;
260     viewsummary:
261             if
262             ::(stateStatusviewsummary== idle)->run Stateviewsummary( )
263             :: else -> printf("case else")
264             fi;
265             if
266             ::(stateStatusviewsummary== done)->stateStatusviewsummary=idle ;
267             goto  sendorder
268             :: else -> printf("case else")
269             fi;
270     goto  viewsummary;
271     sendorder:
272             if
273             ::(stateStatussendorder== idle)->run Statesendorder( )
274             :: else -> printf("case else")
275             fi;
276             if
277             ::(stateStatussendorder== done)->stateStatussendorder=idle ;
278             run FinalState()
279             :: else -> printf("case else")
```

```
280             fi;
281     goto  sendorder;
282     saveedit:
283             if
284             ::(stateStatussaveedit== idle)->run Statesaveedit( )
285             :: else -> printf("case else")
286             fi;
287             if
288             ::(stateStatussaveedit== done)->stateStatussaveedit=idle ;
289             run FinalState()
290             :: else -> printf("case else")
291             fi;
292     goto  saveedit;
293     selectshop:
294             if
295             ::(stateStatusselectshop== idle)->run Stateselectshop( )
296             :: else -> printf("case else")
297             fi;
298             if
299             ::(stateStatusselectshop== done)->stateStatusselectshop=idle ;
300             goto  selectpreiod
301             :: else -> printf("case else")
302             fi;
303     goto  selectshop;
304     selectpreiod:
305             if
306             ::(stateStatusselectpreiod== idle)->run Stateselectpreiod( )
307             :: else -> printf("case else")
308             fi;
309             if
310             ::(stateStatusselectpreiod== done)->
```

```
311                     if
312                     ::(haveorder==false)-> stateStatusselectpreiod=idle ;
313                       goto  selectblock
314                     :: else -> printf("case else")
315                     fi;
316                     if
317                     ::(haveorder&&canedit==false)-> stateStatusselectpreiod=idle ;
318                       goto  selectpreiod
319                     :: else -> printf("case else")
320                     fi;
321                     if
322                     ::(haveorder&&canedit)-> stateStatusselectpreiod=idle ;
323                       goto  editorder
324                     :: else -> printf("case else")
325                     fi;
326             :: else -> printf("case else")
327             fi;
328             goto  selectpreiod;
329     editorder:
330             if
331             ::(stateStatuseditorder== idle)->run Stateeditorder( )
332             :: else -> printf("case else")
333             fi;
334             if
335             ::(stateStatuseditorder== done)->stateStatuseditorder=idle ;
336             goto  saveedit
337             :: else -> printf("case else")
338             fi;
339     goto  editorder;
340     }
```

ภาพที่ ก.1 ภาษาโพรเมลาที่ได้จากการแปลงแผนภาพสเตทแมชชีนของระบบออเดอร์สินค้า

```
  spin: caseOrdering.pml:68, warning, atomic inside d_step (ignored)
spin: caseOrdering.pml:81, warning, atomic inside d_step (ignored)
spin: caseOrdering.pml:94, warning, atomic inside d_step (ignored)
spin: caseOrdering.pml:107, warning, atomic inside d_step (ignored)
spin: caseOrdering.pml:120, warning, atomic inside d_step (ignored)
spin: caseOrdering.pml:141, warning, atomic inside d_step (ignored)
spin: caseOrdering.pml:154, warning, atomic inside d_step (ignored)
spin: caseOrdering.pml:167, warning, atomic inside d_step (ignored)
spin: caseOrdering.pml:189, warning, atomic inside d_step (ignored)
spin: caseOrdering.pml:204, warning, atomic inside d_step (ignored)
  0:      proc  - (:root:) creates proc  0 (checkInVariant)
  0:      proc  - (:root:) creates proc  1 (checkInVariantFinalState)
  0:      proc  - (:root:) creates proc  2 (:init:)
spin: caseOrdering.pml:0, warning, global, 'bit   preFailselectshop' variable is never
used (other than in print stmnts)
spin: caseOrdering.pml:0, warning, global, 'bit   postFailselectshop' variable is never
used (other than in print stmnts)
spin: caseOrdering.pml:0, warning, global, 'bit   preFailsaveedit' variable is never used
(other than in print stmnts)
spin: caseOrdering.pml:0, warning, global, 'bit   postFailsaveedit' variable is never
used (other than in print stmnts)
spin: caseOrdering.pml:0, warning, global, 'bit   preFaileditorder' variable is never used
(other than in print stmnts)
spin: caseOrdering.pml:0, warning, global, 'bit   postFaileditorder' variable is never
used (other than in print stmnts)
spin: caseOrdering.pml:0, warning, global, 'bit   preFailsendorder' variable is never
used (other than in print stmnts)
spin: caseOrdering.pml:0, warning, global, 'bit   postFailsendorder' variable is never
used (other than in print stmnts)
spin: caseOrdering.pml:0, warning, global, 'bit   preFailviewsummary' variable is never
used (other than in print stmnts)
```

```
spin: caseOrdering.pml:0, warning, global, 'bit   postFailviewsummary' variable is never
used (other than in print stmnts)
spin: caseOrdering.pml:0, warning, global, 'bit   preFailselectproduct' variable is never
used (other than in print stmnts)
spin: caseOrdering.pml:0, warning, global, 'bit   postFailselectproduct' variable is
never used (other than in print stmnts)
spin: caseOrdering.pml:0, warning, global, 'bit   preFailselectblock' variable is never
used (other than in print stmnts)
spin: caseOrdering.pml:0, warning, global, 'bit   postFailselectblock' variable is never
used (other than in print stmnts)
spin: caseOrdering.pml:0, warning, global, 'bit   preFailselectpreiod' variable is never
used (other than in print stmnts)
spin: caseOrdering.pml:0, warning, global, 'bit   postFailselectpreiod' variable is never
used (other than in print stmnts)
spin: caseOrdering.pml:0, warning, global, 'bit   preFailcreateorder' variable is never
used (other than in print stmnts)
spin: caseOrdering.pml:0, warning, global, 'bit   postFailcreateorder' variable is never
used (other than in print stmnts)
spin: caseOrdering.pml:0, warning, global, 'mtype stateStatusAction_calculate' variable
is never used (other than in print stmnts)
spin: caseOrdering.pml:0, warning, global, 'bit   preFailAction_calculate' variable is
never used (other than in print stmnts)
spin: caseOrdering.pml:0, warning, global, 'bit   postFailAction_calculate' variable is
never used (other than in print stmnts)
spin: caseOrdering.pml:0, warning, global, 'bit   requestorder' variable is never used
(other than in print stmnts)
spin: caseOrdering.pml:0, warning, global, 'bit   addbasket' variable is never used
(other than in print stmnts)
  1:    proc  2 (:init::1) caseOrdering.pml:228 (state 5)
        [((stateStatuscreateorder==idle))]
  2:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
```

```
        [assert((CompleteFinalState==0))]
  3:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
Starting Statecreateorder with pid 3
  4:    proc  2 (:init::1) creates proc  3 (Statecreateorder)
  4:    proc  2 (:init::1) caseOrdering.pml:229 (state 2)       [(run Statecreateorder())]
  6:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
  9:    proc  2 (:init::1) caseOrdering.pml:232 (state 12)      [else]
 10:    proc  3 (Statecreateorder:1) caseOrdering.pml:177 (state 18)
        [stateStatuscreateorder = running]
 11:    proc  3 (Statecreateorder:1) caseOrdering.pml:180 (state 6)  [((permission==1))]
 12:    proc  3 (Statecreateorder:1) caseOrdering.pml:189 (state 16)          [goto
postcon]
 13:    proc  3 (Statecreateorder:1) caseOrdering.pml:194 (state 17)
        [stateStatuscreateorder = done]
case else 15:   proc  2 (:init::1) caseOrdering.pml:235 (state 11)      [printf('case else')]
 15:    proc  3 (Statecreateorder:1) terminates
 16:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
 18:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
 20:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
 22:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
 24:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
 25:    proc  2 (:init::1) caseOrdering.pml:237 (state 14)      [goto createorder]
 27:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
```

```
 29:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
 30:    proc  2 (:init::1) caseOrdering.pml:228 (state 5)        [else]
case else 31:   proc  2 (:init::1) caseOrdering.pml:230 (state 4)        [printf('case else')]
 35:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
 37:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
 38:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
 40:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
 41:    proc  2 (:init::1) caseOrdering.pml:232 (state 12)
        [((stateStatuscreateorder==done))]
 43:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
 46:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
 47:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
 49:    proc  2 (:init::1) caseOrdering.pml:233 (state 8) [stateStatuscreateorder =
idle]
 51:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
 52:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
 55:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
 57:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
 58:    proc  2 (:init::1) caseOrdering.pml:234 (state 9)        [goto selectshop]
```

```
 59:    proc  2 (:init::1) caseOrdering.pml:294 (state 89)
        [((stateStatusselectshop==idle))]
Starting Stateselectshop with pid 3
 60:    proc  2 (:init::1) creates proc  3 (Stateselectshop)
 60:    proc  2 (:init::1) caseOrdering.pml:295 (state 86)      [(run Stateselectshop())]
 61:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
 63:    proc  3 (Stateselectshop:1) caseOrdering.pml:64 (state 6)
        [stateStatusselectshop = running]
 64:    proc  3 (Stateselectshop:1) caseOrdering.pml:66 (state 2)    [goto
selectshopOperation]
 65:    proc  3 (Stateselectshop:1) caseOrdering.pml:72 (state 5)
        [stateStatusselectshop = done]
 66:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
 69:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
 71:    proc  2 (:init::1) caseOrdering.pml:298 (state 96)
        [((stateStatusselectshop==done))]
 72:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
 75:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
 75:    proc  3 (Stateselectshop:1) terminates
 76:    proc  2 (:init::1) caseOrdering.pml:299 (state 92)      [stateStatusselectshop =
idle]
 78:    proc  2 (:init::1) caseOrdering.pml:300 (state 93)      [goto selectpreiod]
 79:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
 80:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
```

```
 82:    proc  2 (:init::1) caseOrdering.pml:305 (state 103)
        [((stateStatusselectpreiod==idle))]
 83:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
 86:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
Starting Stateselectpreiod with pid 3
 87:    proc  2 (:init::1) creates proc  3 (Stateselectpreiod)
 87:    proc  2 (:init::1) caseOrdering.pml:306 (state 100)    [(run Stateselectpreiod())]
 90:    proc  2 (:init::1) caseOrdering.pml:309 (state 129)    [else]
 91:    proc  3 (Stateselectpreiod:1) caseOrdering.pml:163 (state 6)
        [stateStatusselectpreiod = running]
 92:    proc  3 (Stateselectpreiod:1) caseOrdering.pml:165 (state 2) [goto
selectpreiodOperation]
 93:    proc  3 (Stateselectpreiod:1) caseOrdering.pml:171 (state 5)
        [stateStatusselectpreiod = done]
 94:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
 96:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
 97:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
case else 98:   proc  2 (:init::1) caseOrdering.pml:326 (state 128)    [printf('case else')]
101:    proc  2 (:init::1) caseOrdering.pml:328 (state 131)    [goto selectpreiod]
102:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
103:    proc  3 (Stateselectpreiod:1) terminates
104:    proc  2 (:init::1) caseOrdering.pml:305 (state 103)    [else]
case else105:  proc  2 (:init::1) caseOrdering.pml:307 (state 102)    [printf('case else')]
107:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
```

```
109:    proc  2 (:init::1) caseOrdering.pml:309 (state 129)
        [((stateStatusselectpreiod==done))]
110:    proc  2 (:init::1) caseOrdering.pml:311 (state 111)    [else]
case else111:  proc  2 (:init::1) caseOrdering.pml:314 (state 110)    [printf('case else')]
113:    proc  2 (:init::1) caseOrdering.pml:316 (state 118)    [else]
case else114:  proc  2 (:init::1) caseOrdering.pml:319 (state 117)    [printf('case else')]
115:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
117:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
119:    proc  2 (:init::1) caseOrdering.pml:321 (state 125)    [((haveorder&&canedit))]
121:    proc  2 (:init::1) caseOrdering.pml:322 (state 121)    [stateStatusselectpreiod =
idle]
122:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
125:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
126:    proc  2 (:init::1) caseOrdering.pml:323 (state 122)    [goto editorder]
128:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
129:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
131:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
133:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
134:    proc  2 (:init::1) caseOrdering.pml:330 (state 136)
        [((stateStatuseditorder==idle))]
Starting Stateeditorder with pid 3
135:    proc  2 (:init::1) creates proc  3 (Stateeditorder)
135:    proc  2 (:init::1) caseOrdering.pml:331 (state 133)    [(run Stateeditorder())]
```

```
136:    proc  3 (Stateeditorder:1) caseOrdering.pml:90 (state 6)
        [stateStatuseditorder = running]
137:    proc  3 (Stateeditorder:1) caseOrdering.pml:92 (state 2)      [goto
editorderOperation]
138:    proc  3 (Stateeditorder:1) caseOrdering.pml:98 (state 5)
        [stateStatuseditorder = done]
140:    proc  3 (Stateeditorder:1) terminates
142:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
143:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
144:    proc  2 (:init::1) caseOrdering.pml:334 (state 143)
        [((stateStatuseditorder==done))]
146:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
147:    proc  2 (:init::1) caseOrdering.pml:335 (state 139)     [stateStatuseditorder =
idle]
148:    proc  2 (:init::1) caseOrdering.pml:336 (state 140)     [goto saveedit]
151:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
153:    proc  2 (:init::1) caseOrdering.pml:283 (state 75)
        [((stateStatussaveedit==idle))]
154:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
155:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
Starting Statesaveedit with pid 3
158:    proc  2 (:init::1) creates proc  3 (Statesaveedit)
158:    proc  2 (:init::1) caseOrdering.pml:284 (state 72)      [(run Statesaveedit())]
159:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
```

```
160:    proc  3 (Statesaveedit:1) caseOrdering.pml:77 (state 6)
        [stateStatussaveedit = running]
161:    proc  3 (Statesaveedit:1) caseOrdering.pml:79 (state 2)        [goto
saveeditOperation]
162:    proc  3 (Statesaveedit:1) caseOrdering.pml:85 (state 5)
        [stateStatussaveedit = done]
162:    proc  3 (Statesaveedit:1) terminates
163:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
165:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
167:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
170:    proc  2 (:init::1) caseOrdering.pml:287 (state 82)
        [((stateStatussaveedit==done))]
172:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
174:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
176:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
178:    proc  2 (:init::1) caseOrdering.pml:288 (state 78)       [stateStatussaveedit =
idle]
179:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
181:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
183:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
Starting FinalState with pid 3
184:    proc  2 (:init::1) creates proc  3 (FinalState)
```

```
184:    proc  2 (:init::1) caseOrdering.pml:289 (state 79)      [(run FinalState())]
187:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
188:    proc  2 (:init::1) caseOrdering.pml:292 (state 84)      [goto saveedit]
189:    proc  2 (:init::1) caseOrdering.pml:283 (state 75)
        [((stateStatussaveedit==idle))]
Starting Statesaveedit with pid 4
191:    proc  2 (:init::1) creates proc  4 (Statesaveedit)
191:    proc  2 (:init::1) caseOrdering.pml:284 (state 72)      [(run Statesaveedit())]
193:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
        [assert((CompleteFinalState==0))]
195:    proc  3 (FinalState:1) caseOrdering.pml:214 (state 1) [CompleteFinalState = 1]
196:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
198:    proc  4 (Statesaveedit:1) caseOrdering.pml:77 (state 6)
        [stateStatussaveedit = running]
199:    proc  4 (Statesaveedit:1) caseOrdering.pml:79 (state 2)      [goto
saveeditOperation]
200:    proc  4 (Statesaveedit:1) caseOrdering.pml:85 (state 5)
        [stateStatussaveedit = done]
201:    proc  2 (:init::1) caseOrdering.pml:287 (state 82)
        [((stateStatussaveedit==done))]
201:    proc  4 (Statesaveedit:1) terminates
202:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
202:    proc  3 (FinalState:1) terminates
204:    proc  2 (:init::1) caseOrdering.pml:288 (state 78)      [stateStatussaveedit =
idle]
205:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
        [assert((Terminate==0))]
207:    proc  0 (checkInVariant:1) caseOrdering.pml:217 (state 2)
```

```
        [assert((Terminate==0))]
spin: caseOrdering.pml:223, Error: assertion violated
spin: text of failed assertion: assert((CompleteFinalState==0))
#processes: 3
208:    proc  2 (:init::1) caseOrdering.pml:289 (state 79)
208:    proc  1 (checkInVariantFinalState:1) caseOrdering.pml:222 (state 2)
208:    proc  0 (checkInVariant:1) caseOrdering.pml:220 (state 3)
10 processes created
```

ภาพที่ ก.2 ผลการจำลองภาษาโพรเมลาที่ได้จากการแปลงแผนภาพสเตทแมชชีนของระบบออเดอร์สินค้า

```
1       mtype ={idle ,running ,done}
2       /*---------- variable of StateAdddetailpersonbymanual----------*/
3       mtype stateStatusAdddetailpersonbymanual= idle
4       bool preFailAdddetailpersonbymanual= false
5       bool postFailAdddetailpersonbymanual= false
6
7       /*---------- variable of StateAddcar----------*/
8       mtype stateStatusAddcar= idle
9       bool preFailAddcar= false
10      bool postFailAddcar= false
11
12      /*---------- variable of Stateviewdetailcar----------*/
13      mtype stateStatusviewdetailcar= idle
14      bool preFailviewdetailcar= false
15      bool postFailviewdetailcar= false
16
17      /*---------- variable of StateAddbyPassport----------*/
```

```
18      mtype stateStatusAddbyPassport= idle
19      bool preFailAddbyPassport= false
20      bool postFailAddbyPassport= false
21
22      /*---------- variable of StateAddbyidCard----------*/
23      mtype stateStatusAddbyidCard= idle
24      bool preFailAddbyidCard= false
25      bool postFailAddbyidCard= false
26
27      /*---------- variable of StateAddperson----------*/
28      mtype stateStatusAddperson= idle
29      bool preFailAddperson= false
30      bool postFailAddperson= false
31
32      /*---------- variable of StateArrested----------*/
33      mtype stateStatusArrested= idle
34      bool preFailArrested= false
35      bool postFailArrested= false
36
37      /*---------- variable of Stateviewdetailperson----------*/
38      mtype stateStatusviewdetailperson= idle
39      bool preFailviewdetailperson= false
40      bool postFailviewdetailperson= false
41
42      /*---------- variable of StateCheckstatus----------*/
43      mtype stateStatusCheckstatus= idle
44      bool preFailCheckstatus= false
45      bool postFailCheckstatus= false
46
47      /*---------- variable of StateSearchcar----------*/
48      mtype stateStatusSearchcar= idle
```

```
49      bool preFailSearchcar= false
50      bool postFailSearchcar= false
51
52      /*---------- variable of StateSearchperson----------*/
53      mtype stateStatusSearchperson= idle
54      bool preFailSearchperson= false
55      bool postFailSearchperson= false
56
57      /*---------- variable of Statelogin----------*/
58      mtype stateStatuslogin= idle
59      bool preFaillogin= false
60      bool postFaillogin= false
61
62      bit Terminate = 0 ; /*abnormal terminate */
63      bit CompleteFinalState = 0 ;
64
65      bool requestmanual = true
66      bool notfind = true
67      bool havedataidcard = true
68      bool notok = true
69      bool insertidcard = true
70      bool requestidcard = true
71      bool requestsearchperson = true
72      bool find = true
73      bool requestsearchcar = true
74      bool ok = true
75      bool requestpassport = true
76      bool havedatapassport = true
77      bool inputdataperson = true
78
79      proctype StateAdddetailpersonbymanual() {
```

```
80      d_step {
81      stateStatusAdddetailpersonbymanual = running;
82              precon: goto AdddetailpersonbymanualOperation ;
83              AdddetailpersonbymanualOperation:
84                      atomic{
85                              /*....Fill in details of operation....*/
86                              goto  postcon ;
87                      }
88              postcon:stateStatusAdddetailpersonbymanual= done
89      }
90
91      }
92
93      proctype StateAddcar() {
94      d_step {
95      stateStatusAddcar = running;
96              precon: goto AddcarOperation ;
97              AddcarOperation:
98                      atomic{
99                              /*....Fill in details of operation....*/
100                             goto  postcon ;
101                     }
102             postcon:stateStatusAddcar= done
103     }
104
105     }
106
107     proctype Stateviewdetailcar() {
108     d_step {
109     stateStatusviewdetailcar = running;
110             precon: goto viewdetailcarOperation ;
```

```
111             viewdetailcarOperation:
112                     atomic{
113                             /*....Fill in details of operation....*/
114                             goto  postcon ;
115                     }
116             postcon:stateStatusviewdetailcar= done
117     }
118
119     }
120
121     proctype StateAddbyPassport() {
122     d_step {
123     stateStatusAddbyPassport = running;
124             precon: goto AddbyPassportOperation ;
125             AddbyPassportOperation:
126                     atomic{
127                             /*....Fill in details of operation....*/
128                             goto  postcon ;
129                     }
130             postcon:stateStatusAddbyPassport= done
131     }
132
133     }
134
135     proctype StateAddbyidCard() {
136
137     d_step {
138
139     stateStatusAddbyidCard = running;
140
141             precon:
```

```
142                     if
143                     ::(insertidcard == true)->   goto AddbyidCardOperation ;
144                     :: else -> printf("case else")
145                      fi;
146                     if
147                     ::(!insertidcard == true)->preFailAddbyidCard=true
;Terminate = 1 ;
148                     :: else -> printf("case else")
149                      fi;
150             AddbyidCardOperation:
151                     atomic{
152                             /*....Fill in details of operation....*/
153
154                             goto  postcon ;
155                     }
156
157             postcon:
158                     if
159                     ::(inputdataperson == true)-> stateStatusAddbyidCard=
done
160                     :: else -> printf("case else")
161                      fi;
162                     if
163                     ::(!inputdataperson == true)->postFailAddbyidCard=true
;Terminate = 1 ;
164                     :: else -> printf("case else")
165                      fi;
166     }
167
168     }
169
```

```
170    proctype StateAddperson() {
171    d_step {
172    stateStatusAddperson = running;
173            precon: goto AddpersonOperation ;
174            AddpersonOperation:
175                    atomic{
176                            /*....Fill in details of operation....*/
177                            goto  postcon ;
178                    }
179            postcon:stateStatusAddperson= done
180    }
181
182    }
183
184    proctype StateArrested() {
185    d_step {
186    stateStatusArrested = running;
187            precon: goto ArrestedOperation ;
188            ArrestedOperation:
189                    atomic{
190                            /*....Fill in details of operation....*/
191                            goto  postcon ;
192                    }
193            postcon:stateStatusArrested= done
194    }
195
196    }
197
198    proctype Stateviewdetailperson() {
199    d_step {
200    stateStatusviewdetailperson = running;
```

```
201             precon: goto viewdetailpersonOperation ;
202             viewdetailpersonOperation:
203                     atomic{
204                             /*....Fill in details of operation....*/
205                             goto  postcon ;
206                     }
207             postcon:stateStatusviewdetailperson= done
208     }
209
210     }
211
212     proctype StateCheckstatus() {
213     d_step {
214     stateStatusCheckstatus = running;
215             precon: goto CheckstatusOperation ;
216             CheckstatusOperation:
217                     atomic{
218                             /*....Fill in details of operation....*/
219                             goto  postcon ;
220                     }
221             postcon:stateStatusCheckstatus= done
222     }
223
224     }
225
226     proctype StateSearchcar() {
227     d_step {
228     stateStatusSearchcar = running;
229             precon: goto SearchcarOperation ;
230             SearchcarOperation:
231                     atomic{
```

```
232                             /*....Fill in details of operation....*/
233                             goto  postcon ;
234                     }
235             postcon:stateStatusSearchcar= done
236     }
237
238     }
239
240     proctype StateSearchperson() {
241     d_step {
242     stateStatusSearchperson = running;
243             precon: goto SearchpersonOperation ;
244             SearchpersonOperation:
245                     atomic{
246                             /*....Fill in details of operation....*/
247                             goto  postcon ;
248                     }
249             postcon:stateStatusSearchperson= done
250     }
251
252     }
253
254     proctype Statelogin() {
255     d_step {
256     stateStatuslogin = running;
257             precon: goto loginOperation ;
258             loginOperation:
259                     atomic{
260                             /*....Fill in details of operation....*/
261                             goto  postcon ;
262                     }
```

```
263             postcon:stateStatuslogin= done
264     }
265
266     }
267
268     proctype FinalState() {
269              CompleteFinalState = 1;
270     }
271     active proctype checkInVariant() {
272             do
273             :: assert(Terminate==0);
274             od ;
275     }
276     active proctype checkInVariantFinalState() {
277             do
278             :: assert(CompleteFinalState==0);
279             od ;
280     }
281     init {
282     login:
283             if
284             ::(stateStatuslogin== idle)->run Statelogin( )
285             :: else -> printf("case else")
286              fi;
287             if
288             ::(stateStatuslogin== done &&requestsearchperson)-
>stateStatuslogin=idle ;
289             goto  Searchperson
290             :: else -> printf("case else")
291              fi;
292             if
```

```
293             ::(stateStatuslogin== done &&requestsearchcar)-
>stateStatuslogin=idle ;
294             goto  Searchcar
295             :: else -> printf("case else")
296              fi;
297     goto login;
298     Searchperson:
299             if
300             ::(stateStatusSearchperson== idle)->run StateSearchperson( )
301             :: else -> printf("case else")
302             fi;
303             if
304             ::(stateStatusSearchperson== done)->
305                     if
306                     ::(find)-> stateStatusSearchperson=idle ;
307                       goto  Checkstatus
308                     :: else -> printf("case else")
309                     fi;
310                     if
311                     ::(notfind)-> stateStatusSearchperson=idle ;
312                       goto  Addperson
313                     :: else -> printf("case else")
314                     fi;
315             :: else -> printf("case else")
316             fi;
317             goto  Searchperson;
318     Searchcar:
319             if
320             ::(stateStatusSearchcar== idle)->run StateSearchcar( )
321             :: else -> printf("case else")
322             fi;
```

```
323             if
324             ::(stateStatusSearchcar== done)->
325                     if
326                     ::(find)-> stateStatusSearchcar=idle ;
327                       goto  viewdetailcar
328                     :: else -> printf("case else")
329                     fi;
330                     if
331                     ::(notfind)-> stateStatusSearchcar=idle ;
332                       goto  Addcar
333                     :: else -> printf("case else")
334                     fi;
335             :: else -> printf("case else")
336             fi;
337             goto  Searchcar;
338     viewdetailperson:
339             if
340             ::(stateStatusviewdetailperson== idle)->run Stateviewdetailperson(
)
341             :: else -> printf("case else")
342             fi;
343             if
344             ::(stateStatusviewdetailperson== done)-
>stateStatusviewdetailperson=idle ;
345             run FinalState()
346             :: else -> printf("case else")
347             fi;
348     goto  viewdetailperson;
349     Checkstatus:
350             if
351             ::(stateStatusCheckstatus== idle)->run StateCheckstatus( )
```

```
352             :: else -> printf("case else")
353              fi;
354             if
355             ::(stateStatusCheckstatus== done &&ok)-
>stateStatusCheckstatus=idle ;
356             goto  viewdetailperson
357             :: else -> printf("case else")
358              fi;
359             if
360             ::(stateStatusCheckstatus== done &&notok)-
>stateStatusCheckstatus=idle ;
361             goto  Arrested
362             :: else -> printf("case else")
363              fi;
364     goto Checkstatus;
365     Arrested:
366             if
367             ::(stateStatusArrested== idle)->run StateArrested( )
368             :: else -> printf("case else")
369             fi;
370             if
371             ::(stateStatusArrested== done)->stateStatusArrested=idle ;
372             run FinalState()
373             :: else -> printf("case else")
374             fi;
375     goto  Arrested;
376     Addperson:
377             if
378             ::(stateStatusAddperson== idle)->run StateAddperson( )
379             :: else -> printf("case else")
380             fi;
```

```
381             if
382             ::(stateStatusAddperson== done)->
383                     if
384                     ::(requestidcard)-> stateStatusAddperson=idle ;
385                       goto  AddbyidCard
386                     :: else -> printf("case else")
387                     fi;
388                     if
389                     ::(requestpassport)-> stateStatusAddperson=idle ;
390                       goto  AddbyPassport
391                     :: else -> printf("case else")
392                     fi;
393             :: else -> printf("case else")
394             fi;
395             goto  Addperson;
396     AddbyidCard:
397             if
398             ::(stateStatusAddbyidCard== idle)->run StateAddbyidCard( )
399             :: else -> printf("case else")
400              fi;
401             if
402             ::(stateStatusAddbyidCard== done &&havedataidcard)->
403                     if
404                     ::(requestmanual)-> stateStatusAddbyidCard=idle ;
405                       goto  Adddetailpersonbymanual
406                     :: else -> printf("case else")
407                     fi;
408             :: else -> printf("case else")
409             fi;
410             goto  AddbyidCard;
411     AddbyPassport:
```

```
412             if
413             ::(stateStatusAddbyPassport== idle)->run StateAddbyPassport( )
414             :: else -> printf("case else")
415              fi;
416             if
417             ::(stateStatusAddbyPassport== done &&havedatapassport)->
418                     if
419                     ::(requestmanual)-> stateStatusAddbyPassport=idle ;
420                       goto  Adddetailpersonbymanual
421                     :: else -> printf("case else")
422                     fi;
423             :: else -> printf("case else")
424             fi;
425             goto  AddbyPassport;
426     viewdetailcar:
427             if
428             ::(stateStatusviewdetailcar== idle)->run Stateviewdetailcar( )
429             :: else -> printf("case else")
430             fi;
431             if
432             ::(stateStatusviewdetailcar== done)->stateStatusviewdetailcar=idle
;
433             run FinalState()
434             :: else -> printf("case else")
435             fi;
436     goto  viewdetailcar;
437     Addcar:
438             if
439             ::(stateStatusAddcar== idle)->run StateAddcar( )
440             :: else -> printf("case else")
441             fi;
```

```
442             if
443             ::(stateStatusAddcar== done)->stateStatusAddcar=idle ;
444             goto  Searchcar
445             :: else -> printf("case else")
446             fi;
447     goto  Addcar;
448     Adddetailpersonbymanual:
449             if
450             ::(stateStatusAdddetailpersonbymanual== idle)->run
StateAdddetailpersonbymanual( )
451             :: else -> printf("case else")
452             fi;
453             if
454             ::(stateStatusAdddetailpersonbymanual== done)-
>stateStatusAdddetailpersonbymanual=idle ;
455             run FinalState()
456             :: else -> printf("case else")
457             fi;
458     goto  Adddetailpersonbymanual;
459     }
1
```


ภาพที่ ก.3 ภาษาโพรเมลาที่ได้จากการแปลงแผนภาพสเตทแมชชีนของระบบตรวจสอบบุคคลและยานพาหนะ

```
spin: casePersonCar.pml:84, warning, atomic inside d_step (ignored)
spin: casePersonCar.pml:98, warning, atomic inside d_step (ignored)
spin: casePersonCar.pml:112, warning, atomic inside d_step (ignored)
spin: casePersonCar.pml:126, warning, atomic inside d_step (ignored)
spin: casePersonCar.pml:151, warning, atomic inside d_step (ignored)
spin: casePersonCar.pml:175, warning, atomic inside d_step (ignored)
```

```
spin: casePersonCar.pml:189, warning, atomic inside d_step (ignored)
spin: casePersonCar.pml:203, warning, atomic inside d_step (ignored)
spin: casePersonCar.pml:217, warning, atomic inside d_step (ignored)
spin: casePersonCar.pml:231, warning, atomic inside d_step (ignored)
spin: casePersonCar.pml:245, warning, atomic inside d_step (ignored)
spin: casePersonCar.pml:259, warning, atomic inside d_step (ignored)
  0:    proc  - (:root:) creates proc  0 (checkInVariant)
  0:    proc  - (:root:) creates proc  1 (checkInVariantFinalState)
  0:    proc  - (:root:) creates proc  2 (:init:)
spin: casePersonCar.pml:0, warning, global, 'bit   preFailAdddetailpersonbymanual'
variable is never used (other than in print stmnts)
spin: casePersonCar.pml:0, warning, global, 'bit   postFailAdddetailpersonbymanual'
variable is never used (other than in print stmnts)
spin: casePersonCar.pml:0, warning, global, 'bit   preFailAddcar' variable is never used
(other than in print stmnts)
spin: casePersonCar.pml:0, warning, global, 'bit   postFailAddcar' variable is never
used (other than in print stmnts)
spin: casePersonCar.pml:0, warning, global, 'bit   preFailviewdetailcar' variable is never
used (other than in print stmnts)
spin: casePersonCar.pml:0, warning, global, 'bit   postFailviewdetailcar' variable is
never used (other than in print stmnts)
spin: casePersonCar.pml:0, warning, global, 'bit   preFailAddbyPassport' variable is
never used (other than in print stmnts)
spin: casePersonCar.pml:0, warning, global, 'bit   postFailAddbyPassport' variable is
never used (other than in print stmnts)
spin: casePersonCar.pml:0, warning, global, 'bit   preFailAddbyidCard' variable is never
used (other than in print stmnts)
spin: casePersonCar.pml:0, warning, global, 'bit   postFailAddbyidCard' variable is
never used (other than in print stmnts)
spin: casePersonCar.pml:0, warning, global, 'bit   preFailAddperson' variable is never
used (other than in print stmnts)
```

```
spin: casePersonCar.pml:0, warning, global, 'bit   postFailAddperson' variable is never
used (other than in print stmnts)
spin: casePersonCar.pml:0, warning, global, 'bit   preFailArrested' variable is never
used (other than in print stmnts)
spin: casePersonCar.pml:0, warning, global, 'bit   postFailArrested' variable is never
used (other than in print stmnts)
spin: casePersonCar.pml:0, warning, global, 'bit   preFailviewdetailperson' variable is
never used (other than in print stmnts)
spin: casePersonCar.pml:0, warning, global, 'bit   postFailviewdetailperson' variable is
never used (other than in print stmnts)
spin: casePersonCar.pml:0, warning, global, 'bit   preFailCheckstatus' variable is never
used (other than in print stmnts)
spin: casePersonCar.pml:0, warning, global, 'bit   postFailCheckstatus' variable is never
used (other than in print stmnts)
spin: casePersonCar.pml:0, warning, global, 'bit   preFailSearchcar' variable is never
used (other than in print stmnts)
spin: casePersonCar.pml:0, warning, global, 'bit   postFailSearchcar' variable is never
used (other than in print stmnts)
spin: casePersonCar.pml:0, warning, global, 'bit   preFailSearchperson' variable is
never used (other than in print stmnts)
spin: casePersonCar.pml:0, warning, global, 'bit   postFailSearchperson' variable is
never used (other than in print stmnts)
spin: casePersonCar.pml:0, warning, global, 'bit   preFaillogin' variable is never used
(other than in print stmnts)
spin: casePersonCar.pml:0, warning, global, 'bit   postFaillogin' variable is never used
(other than in print stmnts)
  1:    proc  2 (:init::1) casePersonCar.pml:283 (state 5)      [((stateStatuslogin==idle))]
  2:    proc  1 (checkInVariantFinalState:1) casePersonCar.pml:277 (state 2)
        [assert((CompleteFinalState==0))]
  3:    proc  0 (checkInVariant:1) casePersonCar.pml:272 (state 2)
        [assert((Terminate==0))]
```

```
Starting Statelogin with pid 3
  4:    proc  2 (:init::1) creates proc  3 (Statelogin)
  4:    proc  2 (:init::1) casePersonCar.pml:284 (state 2)      [(run Statelogin())]
  6:    proc  1 (checkInVariantFinalState:1) casePersonCar.pml:277 (state 2)
        [assert((CompleteFinalState==0))]
  9:    proc  2 (:init::1) casePersonCar.pml:287 (state 12)     [else]
 10:    proc  3 (Statelogin:1) casePersonCar.pml:255 (state 6)          [stateStatuslogin =
running]
 11:    proc  3 (Statelogin:1) casePersonCar.pml:257 (state 2)          [goto
loginOperation]
 12:    proc  3 (Statelogin:1) casePersonCar.pml:263 (state 5)          [stateStatuslogin =
done]
case else 14:   proc  2 (:init::1) casePersonCar.pml:290 (state 11)     [printf('case else')]
 14:    proc  3 (Statelogin:1) terminates
 15:    proc  1 (checkInVariantFinalState:1) casePersonCar.pml:277 (state 2)
        [assert((CompleteFinalState==0))]
 17:    proc  1 (checkInVariantFinalState:1) casePersonCar.pml:277 (state 2)
        [assert((CompleteFinalState==0))]
 19:    proc  0 (checkInVariant:1) casePersonCar.pml:272 (state 2)
        [assert((Terminate==0))]
 21:    proc  1 (checkInVariantFinalState:1) casePersonCar.pml:277 (state 2)
        [assert((CompleteFinalState==0))]
 23:    proc  0 (checkInVariant:1) casePersonCar.pml:272 (state 2)
        [assert((Terminate==0))]
 24:    proc  2 (:init::1) casePersonCar.pml:292 (state 19)
        [(((stateStatuslogin==done)&&requestsearchcar))]
 25:    proc  2 (:init::1) casePersonCar.pml:293 (state 15)     [stateStatuslogin = idle]
 27:    proc  2 (:init::1) casePersonCar.pml:294 (state 16)     [goto Searchcar]
 28:    proc  2 (:init::1) casePersonCar.pml:319 (state 52)
        [((stateStatusSearchcar==idle))]
 30:    proc  1 (checkInVariantFinalState:1) casePersonCar.pml:277 (state 2)
```

```
        [assert((CompleteFinalState==0))]
Starting StateSearchcar with pid 3
 31:    proc  2 (:init::1) creates proc  3 (StateSearchcar)
 31:    proc  2 (:init::1) casePersonCar.pml:320 (state 49)    [(run StateSearchcar())]
 33:    proc  3 (StateSearchcar:1) casePersonCar.pml:227 (state 6)
        [stateStatusSearchcar = running]
 34:    proc  3 (StateSearchcar:1) casePersonCar.pml:229 (state 2)  [goto
SearchcarOperation]
 35:    proc  3 (StateSearchcar:1) casePersonCar.pml:235 (state 5)
        [stateStatusSearchcar = done]
 36:    proc  0 (checkInVariant:1) casePersonCar.pml:272 (state 2)
        [assert((Terminate==0))]
 36:    proc  3 (StateSearchcar:1) terminates
 37:    proc  2 (:init::1) casePersonCar.pml:323 (state 71)
        [((stateStatusSearchcar==done))]
 40:    proc  0 (checkInVariant:1) casePersonCar.pml:272 (state 2)
        [assert((Terminate==0))]
 41:    proc  2 (:init::1) casePersonCar.pml:325 (state 60)    [(find)]
 42:    proc  1 (checkInVariantFinalState:1) casePersonCar.pml:277 (state 2)
        [assert((CompleteFinalState==0))]
 44:    proc  2 (:init::1) casePersonCar.pml:326 (state 56)  [stateStatusSearchcar =
idle]
 45:    proc  2 (:init::1) casePersonCar.pml:327 (state 57)    [goto viewdetailcar]
 46:    proc  1 (checkInVariantFinalState:1) casePersonCar.pml:277 (state 2)
        [assert((CompleteFinalState==0))]
 48:    proc  2 (:init::1) casePersonCar.pml:427 (state 191)
        [((stateStatusviewdetailcar==idle))]
 49:    proc  1 (checkInVariantFinalState:1) casePersonCar.pml:277 (state 2)
        [assert((CompleteFinalState==0))]
 51:    proc  0 (checkInVariant:1) casePersonCar.pml:272 (state 2)
        [assert((Terminate==0))]
```

```
Starting Stateviewdetailcar with pid 3
 52:    proc  2 (:init::1) creates proc  3 (Stateviewdetailcar)
 52:    proc  2 (:init::1) casePersonCar.pml:428 (state 188)  [(run Stateviewdetailcar())]
 53:    proc  3 (Stateviewdetailcar:1) casePersonCar.pml:108 (state 6)
        [stateStatusviewdetailcar = running]
 54:    proc  3 (Stateviewdetailcar:1) casePersonCar.pml:110 (state 2)      [goto
viewdetailcarOperation]
 55:    proc  3 (Stateviewdetailcar:1) casePersonCar.pml:116 (state 5)
        [stateStatusviewdetailcar = done]
 57:    proc  0 (checkInVariant:1) casePersonCar.pml:272 (state 2)
        [assert((Terminate==0))]
 60:    proc  2 (:init::1) casePersonCar.pml:431 (state 198)
        [((stateStatusviewdetailcar==done))]
 61:    proc  0 (checkInVariant:1) casePersonCar.pml:272 (state 2)
        [assert((Terminate==0))]
 64:    proc  0 (checkInVariant:1) casePersonCar.pml:272 (state 2)
        [assert((Terminate==0))]
 64:    proc  3 (Stateviewdetailcar:1) terminates
 65:    proc  1 (checkInVariantFinalState:1) casePersonCar.pml:277 (state 2)
        [assert((CompleteFinalState==0))]
 66:    proc  2 (:init::1) casePersonCar.pml:432 (state 194)  [stateStatusviewdetailcar
= idle]
Starting FinalState with pid 3
 69:    proc  2 (:init::1) creates proc  3 (FinalState)
 69:    proc  2 (:init::1) casePersonCar.pml:433 (state 195)  [(run FinalState())]
 70:    proc  1 (checkInVariantFinalState:1) casePersonCar.pml:277 (state 2)
        [assert((CompleteFinalState==0))]
 71:    proc  3 (FinalState:1) casePersonCar.pml:269 (state 1)
        [CompleteFinalState = 1]
 72:    proc  3 (FinalState:1) terminates
 74:    proc  2 (:init::1) casePersonCar.pml:436 (state 200)  [goto viewdetailcar]
```

```
 75:    proc  2 (:init::1) casePersonCar.pml:427 (state 191)
        [((stateStatusviewdetailcar==idle))]
spin: casePersonCar.pml:278, Error: assertion violated
spin: text of failed assertion: assert((CompleteFinalState==0))
#processes: 3
 76:    proc  2 (:init::1) casePersonCar.pml:428 (state 188)
 76:    proc  1 (checkInVariantFinalState:1) casePersonCar.pml:277 (state 2)
 76:    proc  0 (checkInVariant:1) casePersonCar.pml:272 (state 2)
7 processes created
```

ภาพที่ ก.4 ผลการจำลองภาษาโพรเมลาที่ได้จากการแปลงแผนภาพสเตทแมชชีนของระบบตรวจสอบบุคคลและยานพาหนะ

```
1       mtype ={idle ,running ,done}
2       /*---------- variable of Statepayment----------*/
3       mtype stateStatuspayment= idle
4       bool preFailpayment= false
5       bool postFailpayment= false
6
7       /*---------- variable of Stateconfirmcustomer----------*/
8       mtype stateStatusconfirmcustomer= idle
9       bool preFailconfirmcustomer= false
10      bool postFailconfirmcustomer= false
11
12      /*---------- variable of Stategenerateconfirmnumber----------*/
13      mtype stateStatusgenerateconfirmnumber= idle
14      bool preFailgenerateconfirmnumber= false
15      bool postFailgenerateconfirmnumber= false
16
```

```
17      /*---------- variable of Statecreditnotenough----------*/
18      mtype stateStatuscreditnotenough= idle
19      bool preFailcreditnotenough= false
20      bool postFailcreditnotenough= false
21
22      /*---------- variable of Statecheckcredit----------*/
23      mtype stateStatuscheckcredit= idle
24      bool preFailcheckcredit= false
25      bool postFailcheckcredit= false
26
27      bit Terminate = 0 ; /*abnormal terminate */
28      bit CompleteFinalState = 0 ;
29      bool expirydate = true
30      bool havecreditnumber = true
31      bool limitenough = true
32      bool infonotpayment = true
33      int price = 1
34      bool requestchecklimit = true
35      bool confirmnumber = true
36      int limit = 1
37      bool infocanpayment = true
38
39      proctype Statepayment() {
40
41      d_step {
42
43      stateStatuspayment = running;
44
45              precon:
46                      if
47                      ::(confirmnumber == true)->   goto paymentOperation ;
```

```
48                      :: else -> printf("case else")
49                       fi;
50                      if
51                      ::(!confirmnumber == true)->preFailpayment=true ;Terminate =
1 ;
52                      :: else -> printf("case else")
53                       fi;
54              paymentOperation:
55                      atomic{
56                              /*....Fill in details of operation....*/
57
58                              goto  postcon ;
59                      }
60
61              postcon:
62                      if
63                      ::(limit >= 0)-> stateStatuspayment= done
64                      :: else -> printf("case else")
65                       fi;
66                      if
67                      ::(!limit >= 0)->postFailpayment=true ;Terminate = 1 ;
68                      :: else -> printf("case else")
69                       fi;
70      }
71
72      }
73
74      proctype Stateconfirmcustomer() {
75      d_step {
76      stateStatusconfirmcustomer = running;
77              precon: goto confirmcustomerOperation ;
```

```
78              confirmcustomerOperation:
79                      atomic{
80                              /*....Fill in details of operation....*/
81                              goto  postcon ;
82                      }
83              postcon:stateStatusconfirmcustomer= done
84      }
85
86      }
87
88      proctype Stategenerateconfirmnumber() {
89      d_step {
90
91      stateStatusgenerateconfirmnumber = running;
92
93              precon:
94                      if
95                      ::( limit > 0 && price >0   )->   goto
generateconfirmnumberOperation ;
96                      :: else -> printf("case else")
97                       fi;
98                      if
99                      ::(! limit > 0 && price >0   )-
>preFailgenerateconfirmnumber=true ;Terminate = 1 ;
100                     :: else -> printf("case else")
101                      fi;
102             generateconfirmnumberOperation:
103                     atomic{
104                             /*....Fill in details of operation....*/
105                             goto  postcon ;
106                     }
```

```
107
108             postcon:stateStatusgenerateconfirmnumber= done
109     }
110
111     }
112
113     proctype Statecreditnotenough() {
114     d_step {
115     stateStatuscreditnotenough = running;
116             precon: goto creditnotenoughOperation ;
117             creditnotenoughOperation:
118                     atomic{
119                             /*....Fill in details of operation....*/
120                             goto  postcon ;
121                     }
122             postcon:stateStatuscreditnotenough= done
123     }
124
125     }
126
127     proctype Statecheckcredit() {
128     d_step {
129     stateStatuscheckcredit = running;
130             precon: goto checkcreditOperation ;
131             checkcreditOperation:
132                     atomic{
133                             /*....Fill in details of operation....*/
134                             goto  postcon ;
135                     }
136             postcon:stateStatuscheckcredit= done
137     }
```

```
138
139     }
140
141     proctype FinalState() {
142                 CompleteFinalState = 1;
143     }
144     active proctype checkInVariant() {
145                 do
146                 :: assert(Terminate==0);
147                 od ;
148     }
149     active proctype checkInVariantFinalState() {
150                 do
151                 :: assert(CompleteFinalState==0);
152                 od ;
153     }
154     init {
155     checkcredit:
156                 if
157                 ::(stateStatuscheckcredit== idle)->run Statecheckcredit( )
158                 :: else -> printf("case else")
159                  fi;
160                 if
161                 ::(stateStatuscheckcredit== done &&requestchecklimit)->
162                             if
163                             ::(!limitenough)-> stateStatuscheckcredit=idle ;  goto
creditnotenough
164                             :: else -> printf("case else")
165                              fi;
166                             if
167                             ::(limitenough)-> stateStatuscheckcredit=idle ;  goto
```

```
generateconfirmnumber
168                         :: else -> printf("case else")
169                          fi;
170             :: else -> printf("case else")
171              fi;
172             if
173             ::(stateStatuscheckcredit== done &&expirydate)
                  ->stateStatuscheckcredit=idle ;
174             run  FinalState()
175             :: else -> printf("case else")
176              fi;
177     goto checkcredit;
178     creditnotenough:
179             if
180             ::(stateStatuscreditnotenough== idle)->run Statecreditnotenough( )
181             :: else -> printf("case else")
182             fi;
183             if
184             ::(stateStatuscreditnotenough== done)
                  ->stateStatuscreditnotenough=idle ;
185             goto  confirmcustomer
186             :: else -> printf("case else")
187             fi;
188     goto  creditnotenough;
189     generateconfirmnumber:
190             if
191             ::(stateStatusgenerateconfirmnumber== idle)->run
Stategenerateconfirmnumber( )
192             :: else -> printf("case else")
193             fi;
194             if
```

```
195             ::(stateStatusgenerateconfirmnumber== done)
                ->stateStatusgenerateconfirmnumber=idle ;
196             goto  confirmcustomer
197             :: else -> printf("case else")
198             fi;
199     goto  generateconfirmnumber;
200     confirmcustomer:
201             if
202             ::(stateStatusconfirmcustomer== idle)->run Stateconfirmcustomer( )
203             :: else -> printf("case else")
204              fi;
205             if
206             ::(stateStatusconfirmcustomer== done &&infocanpayment)
                ->stateStatusconfirmcustomer=idle ;
207             goto  payment
208             :: else -> printf("case else")
209              fi;
210             if
211             ::(stateStatusconfirmcustomer== done &&infonotpayment)
                 ->stateStatusconfirmcustomer=idle ;
212             run  FinalState()
213             :: else -> printf("case else")
214              fi;
215     goto confirmcustomer;
216     payment:
217             if
218             ::(stateStatuspayment== idle)->run Statepayment( )
219             :: else -> printf("case else")
220             fi;
221             if
222             ::(stateStatuspayment== done)->stateStatuspayment=idle ;
```

```
223                run FinalState()
224                :: else -> printf("case else")
225                fi;
226      goto  payment;
227      }
228      active proctype invhavecreditnumber()
229      { do
230       :: assert( havecreditnumber == true  )
231       od
232      }
```

ภาพที่ ก.5 ภาษาโพรเมลาที่ได้จากการแปลงแผนภาพสเตทแมชชีนแผนภาพสเตทแมชชีนของระบบซื้อของออนไลน์และจ่ายเงินด้วยบัตรเคดิต

```
  spin: cadit_Test.pml:55, warning, atomic inside d_step (ignored)
spin: cadit_Test.pml:79, warning, atomic inside d_step (ignored)
spin: cadit_Test.pml:103, warning, atomic inside d_step (ignored)
spin: cadit_Test.pml:118, warning, atomic inside d_step (ignored)
spin: cadit_Test.pml:132, warning, atomic inside d_step (ignored)
  0:      proc  - (:root:) creates proc  0 (checkInVariant)
  0:      proc  - (:root:) creates proc  1 (checkInVariantFinalState)
  0:      proc  - (:root:) creates proc  2 (:init:)
  0:      proc  - (:root:) creates proc  3 (invhavecreditnumber)
spin: cadit_Test.pml:0, warning, global, 'bit   preFailpayment' variable is never used
(other than in print stmnts)
spin: cadit_Test.pml:0, warning, global, 'bit   postFailpayment' variable is never used
(other than in print stmnts)
spin: cadit_Test.pml:0, warning, global, 'bit   preFailconfirmcustomer' variable is never
used (other than in print stmnts)
spin: cadit_Test.pml:0, warning, global, 'bit   postFailconfirmcustomer' variable is
```

```
never used (other than in print stmnts)
spin: cadit_Test.pml:0, warning, global, 'bit   preFailgenerateconfirmnumber' variable
is never used (other than in print stmnts)
spin: cadit_Test.pml:0, warning, global, 'bit   postFailgenerateconfirmnumber' variable
is never used (other than in print stmnts)
spin: cadit_Test.pml:0, warning, global, 'bit   preFailcreditnotenough' variable is never
used (other than in print stmnts)
spin: cadit_Test.pml:0, warning, global, 'bit   postFailcreditnotenough' variable is
never used (other than in print stmnts)
spin: cadit_Test.pml:0, warning, global, 'bit   preFailcheckcredit' variable is never used
(other than in print stmnts)
spin: cadit_Test.pml:0, warning, global, 'bit   postFailcheckcredit' variable is never
used (other than in print stmnts)
  1:    proc  2 (:init::1) cadit_Test.pml:156 (state 5)  [((stateStatuscheckcredit==idle))]
  2:    proc  3 (invhavecreditnumber:1) cadit_Test.pml:229 (state 2)
        [assert((havecreditnumber==1))]
  3:    proc  0 (checkInVariant:1) cadit_Test.pml:145 (state 2)
        [assert((Terminate==0))]
Starting Statecheckcredit with pid 4
  5:    proc  2 (:init::1) creates proc  4 (Statecheckcredit)
  5:    proc  2 (:init::1) cadit_Test.pml:157 (state 2)  [(run Statecheckcredit())]
  6:    proc  4 (Statecheckcredit:1) cadit_Test.pml:128 (state 6)
        [stateStatuscheckcredit = running]
  7:    proc  4 (Statecheckcredit:1) cadit_Test.pml:130 (state 2)      [goto
checkcreditOperation]
  8:    proc  4 (Statecheckcredit:1) cadit_Test.pml:136 (state 5)
        [stateStatuscheckcredit = done]
  9:    proc  4 (Statecheckcredit:1) terminates
 10:    proc  1 (checkInVariantFinalState:1) cadit_Test.pml:150 (state 2)
        [assert((CompleteFinalState==0))]
 11:    proc  3 (invhavecreditnumber:1) cadit_Test.pml:229 (state 2)
```

```
        [assert((havecreditnumber==1))]
 15:    proc  0 (checkInVariant:1) cadit_Test.pml:145 (state 2)
        [assert((Terminate==0))]
 17:    proc  1 (checkInVariantFinalState:1) cadit_Test.pml:150 (state 2)
        [assert((CompleteFinalState==0))]
 18:    proc  0 (checkInVariant:1) cadit_Test.pml:145 (state 2)
        [assert((Terminate==0))]
 20:    proc  1 (checkInVariantFinalState:1) cadit_Test.pml:150 (state 2)
        [assert((CompleteFinalState==0))]
 21:    proc  2 (:init::1) cadit_Test.pml:160 (state 24)
        [(((stateStatuscheckcredit==done)&&requestchecklimit))]
 24:    proc  1 (checkInVariantFinalState:1) cadit_Test.pml:150 (state 2)
        [assert((CompleteFinalState==0))]
 26:    proc  2 (:init::1) cadit_Test.pml:162 (state 13)[else]
case else 27:   proc  2 (:init::1) cadit_Test.pml:164 (state 12)[printf('case else')]
 28:    proc  0 (checkInVariant:1) cadit_Test.pml:145 (state 2)
        [assert((Terminate==0))]
 29:    proc  1 (checkInVariantFinalState:1) cadit_Test.pml:150 (state 2)
        [assert((CompleteFinalState==0))]
 31:    proc  2 (:init::1) cadit_Test.pml:166 (state 20)[(limitenough)]
 32:    proc  3 (invhavecreditnumber:1) cadit_Test.pml:229 (state 2)
        [assert((havecreditnumber==1))]
 33:    proc  2 (:init::1) cadit_Test.pml:167 (state 16)[stateStatuscheckcredit = idle]
 35:    proc  3 (invhavecreditnumber:1) cadit_Test.pml:229 (state 2)
        [assert((havecreditnumber==1))]
 36:    proc  2 (:init::1) cadit_Test.pml:167 (state 17)[goto generateconfirmnumber]
 39:    proc  0 (checkInVariant:1) cadit_Test.pml:145 (state 2)
        [assert((Terminate==0))]
 40:    proc  1 (checkInVariantFinalState:1) cadit_Test.pml:150 (state 2)
        [assert((CompleteFinalState==0))]
 42:    proc  0 (checkInVariant:1) cadit_Test.pml:145 (state 2)
```

```
        [assert((Terminate==0))]
 44:    proc  2 (:init::1) cadit_Test.pml:190 (state 52)
        [((stateStatusgenerateconfirmnumber==idle))]
Starting Stategenerateconfirmnumber with pid 4
 45:    proc  2 (:init::1) creates proc  4 (Stategenerateconfirmnumber)
 45:    proc  2 (:init::1) cadit_Test.pml:191 (state 49)[(run
Stategenerateconfirmnumber())]
 48:    proc  4 (Stategenerateconfirmnumber:1) cadit_Test.pml:89 (state 18)
        [stateStatusgenerateconfirmnumber = running]
 49:    proc  4 (Stategenerateconfirmnumber:1) cadit_Test.pml:94 (state 6)
        [((((limit>0)&&(price>0))))]
 50:    proc  4 (Stategenerateconfirmnumber:1) cadit_Test.pml:103 (state 16)
        [goto postcon]
 51:    proc  4 (Stategenerateconfirmnumber:1) cadit_Test.pml:108 (state 17)
        [stateStatusgenerateconfirmnumber = done]
 53:    proc  1 (checkInVariantFinalState:1) cadit_Test.pml:150 (state 2)
        [assert((CompleteFinalState==0))]
 54:    proc  2 (:init::1) cadit_Test.pml:194 (state 59)
        [((stateStatusgenerateconfirmnumber==done))]
 55:    proc  2 (:init::1) cadit_Test.pml:195 (state 55)
        [stateStatusgenerateconfirmnumber = idle]
 56:    proc  2 (:init::1) cadit_Test.pml:196 (state 56)[goto confirmcustomer]
 56:    proc  4 (Stategenerateconfirmnumber:1) terminates
 57:    proc  3 (invhavecreditnumber:1) cadit_Test.pml:229 (state 2)
        [assert((havecreditnumber==1))]
 59:    proc  2 (:init::1) cadit_Test.pml:201 (state 66)
        [((stateStatusconfirmcustomer==idle))]
 61:    proc  1 (checkInVariantFinalState:1) cadit_Test.pml:150 (state 2)
        [assert((CompleteFinalState==0))]
 62:    proc  0 (checkInVariant:1) cadit_Test.pml:145 (state 2)
        [assert((Terminate==0))]
```

```
 63:    proc  3 (invhavecreditnumber:1) cadit_Test.pml:229 (state 2)
        [assert((havecreditnumber==1))]
 66:    proc  1 (checkInVariantFinalState:1) cadit_Test.pml:150 (state 2)
        [assert((CompleteFinalState==0))]
Starting Stateconfirmcustomer with pid 4
 68:    proc  2 (:init::1) creates proc  4 (Stateconfirmcustomer)
 68:    proc  2 (:init::1) cadit_Test.pml:202 (state 63)[(run Stateconfirmcustomer())]
 69:    proc  0 (checkInVariant:1) cadit_Test.pml:145 (state 2)
        [assert((Terminate==0))]
 71:    proc  2 (:init::1) cadit_Test.pml:205 (state 73)[else]
case else 72:   proc  2 (:init::1) cadit_Test.pml:208 (state 72)[printf('case else')]
 74:    proc  4 (Stateconfirmcustomer:1) cadit_Test.pml:75 (state 6)
        [stateStatusconfirmcustomer = running]
 75:    proc  4 (Stateconfirmcustomer:1) cadit_Test.pml:77 (state 2)        [goto
confirmcustomerOperation]
 76:    proc  4 (Stateconfirmcustomer:1) cadit_Test.pml:83 (state 5)
        [stateStatusconfirmcustomer = done]
 77:    proc  1 (checkInVariantFinalState:1) cadit_Test.pml:150 (state 2)
        [assert((CompleteFinalState==0))]
 80:    proc  2 (:init::1) cadit_Test.pml:210 (state 80)
        [(((stateStatusconfirmcustomer==done)&&infonotpayment))]
 82:    proc  0 (checkInVariant:1) cadit_Test.pml:145 (state 2)
        [assert((Terminate==0))]
 83:    proc  1 (checkInVariantFinalState:1) cadit_Test.pml:150 (state 2)
        [assert((CompleteFinalState==0))]
 83:    proc  4 (Stateconfirmcustomer:1) terminates
 84:    proc  2 (:init::1) cadit_Test.pml:211 (state 76)[stateStatusconfirmcustomer =
idle]
 87:    proc  1 (checkInVariantFinalState:1) cadit_Test.pml:150 (state 2)
        [assert((CompleteFinalState==0))]
Starting FinalState with pid 4
```

```
 89:    proc  2 (:init::1) creates proc  4 (FinalState)
 89:    proc  2 (:init::1) cadit_Test.pml:212 (state 77)[(run FinalState())]
 91:    proc  0 (checkInVariant:1) cadit_Test.pml:145 (state 2)
        [assert((Terminate==0))]
 92:    proc  1 (checkInVariantFinalState:1) cadit_Test.pml:150 (state 2)
        [assert((CompleteFinalState==0))]
 93:    proc  2 (:init::1) cadit_Test.pml:215 (state 82)[goto confirmcustomer]
 96:    proc  3 (invhavecreditnumber:1) cadit_Test.pml:229 (state 2)
        [assert((havecreditnumber==1))]
 98:    proc  3 (invhavecreditnumber:1) cadit_Test.pml:229 (state 2)
        [assert((havecreditnumber==1))]
 99:    proc  1 (checkInVariantFinalState:1) cadit_Test.pml:150 (state 2)
        [assert((CompleteFinalState==0))]
101:    proc  3 (invhavecreditnumber:1) cadit_Test.pml:229 (state 2)
        [assert((havecreditnumber==1))]
103:    proc  2 (:init::1) cadit_Test.pml:201 (state 66)
        [((stateStatusconfirmcustomer==idle))]
104:    proc  4 (FinalState:1) cadit_Test.pml:142 (state 1)     [CompleteFinalState = 1]
104:    proc  4 (FinalState:1) terminates
105:    proc  3 (invhavecreditnumber:1) cadit_Test.pml:229 (state 2)
        [assert((havecreditnumber==1))]
Starting Stateconfirmcustomer with pid 4
106:    proc  2 (:init::1) creates proc  4 (Stateconfirmcustomer)
106:    proc  2 (:init::1) cadit_Test.pml:202 (state 63)[(run Stateconfirmcustomer())]
108:    proc  0 (checkInVariant:1) cadit_Test.pml:145 (state 2)
        [assert((Terminate==0))]
109:    proc  3 (invhavecreditnumber:1) cadit_Test.pml:229 (state 2)
        [assert((havecreditnumber==1))]
111:    proc  4 (Stateconfirmcustomer:1) cadit_Test.pml:75 (state 6)
        [stateStatusconfirmcustomer = running]
112:    proc  4 (Stateconfirmcustomer:1) cadit_Test.pml:77 (state 2)          [goto
```

```
confirmcustomerOperation]
113:    proc  4 (Stateconfirmcustomer:1) cadit_Test.pml:83 (state 5)
        [stateStatusconfirmcustomer = done]
115:    proc  0 (checkInVariant:1) cadit_Test.pml:145 (state 2)
        [assert((Terminate==0))]
118:    proc  0 (checkInVariant:1) cadit_Test.pml:145 (state 2)
        [assert((Terminate==0))]
119:    proc  3 (invhavecreditnumber:1) cadit_Test.pml:229 (state 2)
        [assert((havecreditnumber==1))]
spin: cadit_Test.pml:151, Error: assertion violated
spin: text of failed assertion: assert((CompleteFinalState==0))
#processes: 5
122:    proc  4 (Stateconfirmcustomer:1) cadit_Test.pml:86 (state 7)
122:    proc  3 (invhavecreditnumber:1) cadit_Test.pml:229 (state 2)
122:    proc  2 (:init::1) cadit_Test.pml:205 (state 73)
122:    proc  1 (checkInVariantFinalState:1) cadit_Test.pml:150 (state 2)
122:    proc  0 (checkInVariant:1) cadit_Test.pml:148 (state 3)
9 processes create
```

ภาพที่ ก.6 ผลการจำลองภาษาโพรเมลาที่ได้จากการแปลงแผนภาพสเตทแมชชีนของระบบออเดอร์สินค้า

ประวัติผู้เขียนวิทยานิพนธ์

นางสาวปาณิสรา ดำจันทร์ สำเร็จการศึกษาปริญญาตรีวิทยาศาสตร์บัณฑิต คณะวิทยาศาสตร์ สาขาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยรามคำแหง ปีการศึกษา 2552 ทำงานใน ตำแหน่งโปรแกรมเมอร์และวิศวกรซอฟต์แวร์ระหว่างปี 2550 – 2558 และเข้าศึกษาต่อสาขาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ปี 2558