การแปลงตรรกเชิงเวลาแบบเมตริกไปเป็นตรรกเชิงเวลาเชิงเส้นสำหรับโพรเมลา

น.ส.จุฑามาศ กะวิเศษ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2563

Transformation of Metric Temporal Logic into Linear Temporal Logic for Promela

Miss Jutamard Kawises

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Software Engineering
Department of Computer Engineering
FACULTY OF ENGINEERING
Chulalongkorn University
Academic Year 2020

หัวข้อวิทยานิพนธ์	การแปลงตรรกเชิงเวลาแบบเมตริกไปเป็นตรรกเชิงเวลาเชิงเส้นสำหรับโพรเมลา
โดย	น.ส.จุฑามาศ กะวิเศษ
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.สุพจน์ เตชวรสินสกุล)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(รองศาสตราจารย์ ดร.สมชาย ประสิทธิ์จูตระกูล)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ)

..... กรรมการ
(รองศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี)

..... กรรมการภายนอกมหาวิทยาลัย
(ดร.เดชานุชิต กตัญญูทวีทิพย์)

จุฑามาศ กะวิเศษ : การแปลงตรรกเชิงเวลาแบบเมตริกไปเป็นตรรกเชิงเวลาเชิงเส้นสำหรับโพรเมลา. (Transformation of Metric Temporal Logic into Linear Temporal Logic for Promela) อ.ที่ปรึกษาหลัก : รศ. ดร.วิวัฒน์ วัฒนาวุฒิ

การทวนสอบสำหรับระบบเรียลไทม์เป็นสิ่งสำคัญและหลีกเลี่ยงไม่ได้ โดยปกติแล้วระบบเรียลไทม์จะอยู่ในรูปแบบที่เป็นรูปแบบทางการและถูกตรวจสอบเพื่อให้สามารถรองรับคุณลักษณะที่สำคัญ ๆ ได้ แบบจำลองที่อยู่ในรูปแบบที่เป็นทางการนี้สามารถตรวจสอบคุณลักษณะที่สนใจด้วยใช้การตรวจสอบแบบจำลอง อย่างไรก็ตาม การตรวจสอบคุณลักษณะของแบบจำลองดังกล่าว ที่ได้ประสิทธิภาพสามารถเขียนแทนให้อยู่ในรูปแบบของตรรกเชิงเวลาแบบเมตริก หรือเอ็มทีแอล ซึ่งสามารถระบุช่วงเวลาที่สนใจได้ วิทยานิพนธ์ฉบับนี้ได้นำเสนอทางเลือกหนึ่งสำหรับการแปลงสมการในรูปบบแอลทีแอลไปเป็นสมการในรูปแบบของตรรกศาสตร์เชิงเส้น หรือแอลทีแอล ที่ทำงานร่วมกันกับภาษาโพรเมลา จากคุณลักษณะของเอ็มทีแอลที่ประกอบไปด้วย ตัวดำเนินการแบบตลอดไป ตัวดำเนินการแบบในที่สุด ตัวดำเนินการแบบถัดไป ตัวดำเนินการจนกระทั่งแบบเข้ม และตัวดำเนินการจนกระทั่งแบบอ่อน จากคุณะลักษณะทั้งหมดที่กล่าวมานั้นเป็นคุณลักษณะที่ได้นำมาศึกษาในวิทยานิพนธ์ฉบับนี้ กรณีศึกษาแบบจำลองระบบการบรรจุในแจ้งยอดหนี้บัตรเครดิตที่อยู่ในรูปแบบของไทม์แพททริเนทและถูกเขียนด้วยภาษาโพรเมลาซึ่งทำงานด้วยระบบสัญญาณนาฬิกาหลักและนาฬิกาย่อยเพื่อรองรับการทำงานกับเงื่อนไขด้านเวลาของกระบวนการในกรณีศึกษา การแปลงสมการในรูปแบบของเอ็มทีแอลพร้อมด้วยเงื่อนไขของเวลาไปเป็นสมการในรูปแบบของแอลทีแอลแสดงให้เห็นว่าการทวนสอบแบบจำลองกรณีศึกษาทำงานได้อย่างถูกต้องโดยทวนสอบแบบจำลองดังกล่าวด้วยเครื่องมือสปิน

สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่อนิสิต .....

ปีการศึกษา 2563 ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

5971002121 : MAJOR SOFTWARE ENGINEERING

KEYWORD: Time Petri nets, Metric temporal logic (MTL), Linear temporal logic (LTL), Promela, SPIN Tool

Jutamard Kawises : Transformation of Metric Temporal Logic into Linear Temporal Logic for Promela. Advisor: Assoc. Prof. Wiwat Vatanawood, Ph.D.

Formal verification for the real-time systems is crucial and inevitable. Typically, a real-time system is formalized and verified to ensure the critical properties. The formal model and its desirable properties may be checked using model checker. However, the target properties may be efficiently written in metric temporal logic (MTL) providing the time duration checking. In this thesis, we propose an alternative to transform the given MTL formula into the conventional linear temporal logic (LTL) formula in order to work along with Promela code. The following MTL operators - operators of Always, Eventually, Next, Strong Until, and Weak Until, are focused in this thesis. A case study of credit card billing system model is formalized into Timed Petri net and translated into Promela code with the multi-level clock system – Global clock and local clocks, to cope with the time constraints among the activities in the case study. The transformation of the MTL formula with time constraints into the corresponding LTL formula is demonstrated. The formal verification of the case study is correctly conducted using SPIN tool.

Field of Study: Software Engineering Student's Signature .....

Academic Year: 2020 Advisor's Signature .....

กิตติกรรมประกาศ

ขอกราบขอบพระคุณอาจารย์ที่ปรึกษาวิทยานิพนธ์ รศ.ดร. วิวัฒน์ วัฒนาวุฒิ ที่ให้คำปรึกษาในการจัดทำวิทยานิพนธ์เล่มนี้เป็นอย่างดี โดยเสียสละเวลาให้คำแนะนำและแก้ไข ซึ่งช่วยทำให้วิทยานิพนธ์สำเร็จลุล่วงไปได้ด้วยดี

ขอกราบขอบพระคุณคณะกรรมการทุกท่านที่ให้คำเสนอแนะในการปรับปรุงวิทยานิพนธ์เพื่อให้วิทยานิพนธ์เล่มนี้ถูกต้องและครบถ้วน

ขอขอบคุณทุกท่านที่ช่วยผลักดันให้ข้าพเจ้าสามารถจัดทำวิทยานิพนธ์เล่มนี้เสร็จสมบูรณ์

จุฑามาศ กะวิเศษ

สารบัญ

หน้า

บทที่ 1
บทนำ

1.1. ที่มาและความสำคัญของปัญหา

เมื่อความก้าวหน้าของเทคโนโลยีที่เกิดขึ้นในยุคทองที่เรียกได้ว่าเป็นโลกของดิจิตอลโดยแท้จริง ไม่ว่าจะเกิดบนพื้นฐานของการแข่งขันที่สูงทางธุรกิจ การติดต่อสื่อสาร การคมนาคม การเงิน ตลาดหุ้น เป็นต้น ซอต์ฟแวร์ที่ถูกพัฒนาขึ้นจะต้องสามารถตอบสนองความต้องการของผู้ใช้งานได้อย่างมีประสิทธิภาพสูงสุด มีความแม่นยำและต้องสร้างความเชื่อมั่นให้กับผู้ใช้งาน ภายใต้การทำงานดังกล่าวของซอต์ฟแวร์จึงนำระบบเรียลไทม์ (Real-time System) เข้ามาช่วยสนับสนุนเพื่อสร้างผลประโยชน์และผลกำไรในการทำงาน ซึ่งการให้ความสำคัญกับบทบาทของกระบวนการการทำงานของซอต์ฟแวร์ต่าง ๆ ที่เกิดขึ้นในยุคนี้ ไม่สามารถจะปฏิเสธได้เลยว่าเทคโนโลยีหรือซอต์ฟแวร์ที่เกิดขึ้นนั้น สร้างเม็ดเงินมหาศาลให้กับธุรกิจ หรือการทำงานต่างๆ อย่างไรก็ตามเมื่อการดำเนินงานของกระบวนการใดๆ ในซอต์ฟแวร์บนระบบใดๆ เกิดหยุดชะงักหรือติดตาย ไม่สามารถแก้ไขได้ทันเวลา ไม่สามารถรองรับข้อผิดพลาดที่เกิดขึ้นได้ จะส่งผลกระทบเป็นวงกว้างต่อธุรกิจการทำงานอาจทำให้พลาดโอกาสของการทำเงิน หรือเสียหายต่อผลกำไรของงานได้ ดังนั้นการจะสร้างระบบซอต์ฟแวร์แบบเรียลไทม์ที่ดีได้นั้น จะต้องอาศัยการตรวจสอบความถูกต้องของกระบวนการการทำงานของซอต์ฟแวร์ที่จะส่งออกสู่ตลาด เพื่อสร้างความเชื่อมั่นให้กับผู้ใช้งาน และสร้างข้อได้เปรียบกับคู่แข่ง เพื่อเป็นรากฐานที่มั่นคงในการดำเนินธุรกิจ

กระบวนการของการดำเนินการในระบบเรียลไทม์หรือแบบจำลองใดๆ ที่จำเป็นต้องมีความถูกต้องแม่นยำ ภายใต้เวลาที่ถูกจำกัดของการดำเนินการ ซึ่งด้วยระยะเวลาดังกล่าว เมื่อเริ่มต้นกระบวนการของการทำงาน การกำหนดเวลาเริ่มต้นและสิ้นสุดในแต่ละกระบวนการจะถูกระบุไว้อย่างชัดเจน ซึ่งอาจจะเป็นลักษณะการทำงานเป็นแบบคู่ขนานหรือการทำงานแบบเส้นตรง ที่ไม่ว่าจะมีจุดสิ้นสุดของการทำงานแยกจากกัน หรือเป็นจุดสิ้นสุดของการทำงานจุดเดียวกันในระบบหรือแบบจำลองนั้นๆ จะทราบได้อย่างไรว่า ณ เวลาของจุดสิ้นสุดการทำงานในแต่ละกระบวนการการทำงานนั้นจะสิ้นสุดที่ ณ เวลาใด หรือ ณ จุดสิ้นสุดของการทำงานแบบคู่ขนาน มีเวลาสิ้นสุดของกระบวนการคู่ขนานนั้นๆ พร้อมกันหรือไม่ ทั้งนี้การตรวจสอบการทำงานของระบบใดๆ ให้สามารถทำงานได้อย่างมีประสิทธิภาพไม่เกิดความล่าช้าจากกระบวนการการทำงานย่อยๆ จำเป็นต้องได้รับการตรวจสอบอย่างละเอียดและครอบคลุมสเตจทั้งหมดที่เป็นไปได้ เพื่อให้นำไปสู่การพยากรณ์การทำงานในลำดับกระบวนการทำงานต่อไปได้ นอกจากนี้การตรวจสอบพบความผิดพลาดของการ

ดำเนินการในระบบหรือแบบจำลองใดๆ ก็จะช่วยให้สามารถนำข้อผิดพลาดดังกล่าว แก้ไขได้ทันถ่วงที เพื่อป้องกันความเสียหายที่อาจเกิดขึ้นได้

แบบจำลองระบบบรรจุใบแจ้งยอดหนี้บัตรเครดิตที่ได้ยกมาเป็นกรณีศึกษาในงานวิจัยนี้ เป็นระบบการทำงานแบบเรียลไทม์ซึ่งในสายพานการทำงานจะแบ่งออกเป็น 2 สายพาน เมื่อเริ่มกระบวนการทำงานระบบจะทำงานทั้งสองสายพานคู่ขนานกัน และสายพานจะมีการรวมกันก่อนที่จะจบกระบวนการ ดังนั้น ณ ช่วงเวลาที่เป็นจุดรวมกันทั้งสองสายพานจะต้องทำงานเสร็จทั้งสองสายพานจึงจะสามารถเริ่มกระบวนการถัดไปได้ จึงมีความจำเป็นต้องตรวจสอบความพร้อมและความถูกต้องของการทำงาน โดยจะต้องสามารถทำงานหนึ่งๆ ในเวลาที่จำกัดอย่างถูกต้อง เพื่อส่งต่อให้กระบวนการถัดไปได้ทันเวลาโดยไม่มีข้อบกพร่อง อย่างไรก็ตามจะมั่นใจได้อย่างไรว่าทั้งสองสายพานในกระบวนการของการทำงานของระบบจะสามารถทำงานได้อย่างถูกต้องและทันเวลา ไม่เกิดการหยุดชะงักระหว่างกระบวนการทำงานของแต่ละสายพาน หรือทำงานผิดไปจากที่ควรจะเป็น งานวิจัยนี้จึงได้นำการทวนสอบระบบเรียลไทม์เข้ามาช่วยทวนสอบเพื่อตรวจสอบความถูกต้องและความสามารถของระบบที่เป็นกรณีศึกษา

การทวนสอบระบบเรียลไทม์ในกรณีศึกษาทำได้โดยการสร้างแบบจำลองเชิงรูปนัยของระบบด้วยภาษาโพรเมลา และกำหนดคุณสมบัติที่ต้องการทวนสอบด้วยเงื่อนไขในรูปแบบแอลทีแอลที่ทวนสอบด้วยระบบเรียลไทม์ซึ่งสนใจเวลาตั้งแต่ต้นกระบวนการจนจบกระบวนการ โดยทั้งหมดนี้สามารถทำได้โดยใช้เครื่องมือสปิน อย่างไรก็ตาม เมื่อระบบเรียลไทม์มีปัจจัยเรื่องเวลาเข้ามาเกี่ยวข้อง กล่าวคือ มีการกำหนดช่วงเวลาและความจำเป็นของการทำงานร่วมกันภายใต้จังหวะของช่วงเวลาที่กำหนดเท่านั้น ทำให้การทวนสอบด้วยเงื่อนไขแบบแอลทีแอลไม่สามารถทำได้โดยตรง โดยทั่วไปมักนิยมใช้การเขียนเงื่อนไขในรูปแบบเอ็มทีแอลสำหรับการเขียนในภาษาโพรเมลา เพื่อกำหนดช่วงเวลาที่สนใจลงไป แต่อย่างไรก็ตามการที่ใช้เงื่อนไขในรูปแบบเอ็มทีแอลทวนสอบนั้นไม่สามารถทำได้ ดังนั้นเพื่อให้ภาษาโพรเมลาสามารถนำมาใช้งานได้ งานวิจัยนี้จะนำเสนอการแปลงเงื่อนไขในรูปแบบเอ็มทีแอลให้อยู่ในรูปแบบแอลทีแอล พร้อมกับส่วนการจัดการเรื่องการนับเวลาและกำหนดช่วงเวลาที่เขียนด้วยภาษาโพรเมลา เพื่อให้ระบบเรียลไทม์ที่มีปัจจัยเรื่องเวลาได้รับการทวนสอบด้วยเครื่องมือสปินได้

1.2. วัตถุประสงค์งานวิจัย

เพื่อนำเสนอการแปลงเงื่อนไขแบบเอ็มทีแอลให้อยู่ในรูปแบบแอลทีแอล พร้อมกับส่วนการจัดการเรื่องการนับเวลาและติดตามช่วงเวลาที่เขียนด้วยภาษาโพรเมลา

1.3. ขอบเขตงานวิจัย

1) แบบจำลองเขียนด้วยภาษาโพรเมลาและทวนสอบตรรกะเชิงเวลาด้วยโปรแกรมสปิน
2) พัฒนาการแปลงคุณลักษณะและข้อจำกัดด้านเวลาให้อยู่ในรูปแบบของภาษาโพรเมลา
3) การจัดการด้านเวลาสำหรับภาษาโพรเมลา จะมีการดำเนินการนับเวลาแบบย่อยและนับเวลาแบบหลัก
4) คุณลักษณะของ MTL ที่จะทำการแปลงมีดังนี้
   (1) ตัวดำเนินการแบบตลอดไป (Always - $\Box_{[t1, t2]} \varphi$)
   (2) ตัวดำเนินการแบบในที่สุด (Eventually - $\Diamond_{[t1, t2]} \varphi$)
   (3) ตัวดำเนินการแบบถัดไป (Next - $\bigcirc_{[t1, t2]} \varphi$)
   (4) ตัวดำเนินการจนกระทั่งแบบเข้ม (Strong Until - $\varphi \cup_{[t1, t2]} \psi$)
   (5) ตัวดำเนินการจนกระทั่งแบบอ่อน (Weak Until - $\varphi \omega_{[t1, t2]} \psi$)
5) กรณีศึกษาที่ใช้ในการประเมินผลจะครอบคลุมกรณีต่างๆ ในรูปไทม์เพททริเนทและกรณีศึกษาที่เป็นกระบวนการธุรกรรมเสมือนจริง

1.4. ประโยชน์ที่ได้รับ

1) ได้แบบจำลองที่ถูกสร้างขึ้นด้วยภาษาโพรเมลา สำหรับรูปแบบการจัดการด้วยเวลาแบบเรียลไทม์
2) การทวนสอบการแปลงคุณลักษณะของการจัดการด้วยเวลาแบบเรียลไทม์ด้วยกรณีศึกษา
3) ได้คุณลักษณะในรูปแบบของเอ็มทีแอลไปเป็นแอลทีแอล

1.5. ขั้นตอนการดำเนินการ

1) ศึกษาหลักการทำงานและโครงสร้างของตรรกศาสตร์เชิงเวลา
2) ศึกษาหลักการทำงานและคุณลักษณะของรูปแบบเอ็มทีแอลและรูปแบบแอลทีแอล
3) ศึกษาหลักการทำงานของโปรแกรมสปินและการพัฒนาด้วยภาษาโพรเมลา
4) ศึกษางานวิจัยที่เกี่ยวข้องกับตรรกศาสตร์เชิงเวลา โปรแกรมสปินและภาษาโพรเมลา
5) วิเคราะห์และออกแบบโครงสร้าง การทำงานของแบบจำลองสำหรับการบรรจุใบแจ้งยอดหนี้บัตรเครดิต
6) พัฒนาตัวนับเวลาของกระบวนการของการบรรจุใบแจ้งยอดบัตรเครดิต
7) สร้างการแปลงคุณลักษณะของแบบจำลองเพื่อการทวนสอบ
8) สร้างคุณลักษณะที่กำหนดขึ้นภายใต้ขอบเขตงานวิจัย เพื่อเปรียบเทียบกับแบบจำลองกรณีศึกษา

9) ทวนสอบแบบจำลองกับเงื่อนไขที่สอดคล้องและเงื่อนไขค้านด้วยโปรแกรมสปิน

10) สรุปผลการทวนสอบ

11) จัดทำรายงานการวิจัย และรูปเล่มวิทยานิพนธ์

1.6. บทความที่ตีพิมพ์จากงานวิจัย

ส่วนหนึ่งของวิทยานิพนธ์นี้ ได้รับการตีพิมพ์เป็นบทความวิชาการ ดังนี้

1) เรื่อง “Transformation of Metric Temporal Logic into Linear Temporal Logic for Promela” โดย จุฑามาศ กะวิเศษ และ วิวัฒน์ วัฒนาวุฒิ ในงานประชุมวิชาการ 20thIEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD2019) เมื่อวันที่ 8 - 11 มิถุนายน พ.ศ. 2562 ณ จังหวัดโทยามะ ประเทศญี่ปุ่น

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1. ตรรกศาสตร์เชิงเส้น หรือแอลทีแอล (Linear Temporal Logic : LTL)

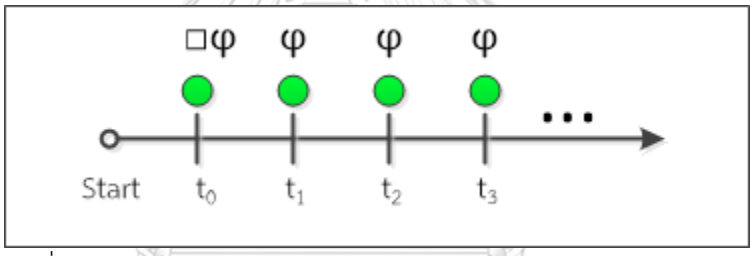
แอลทีแอลเป็นตรรกศาสตร์เชิงเส้นที่มีลำดับของสถานะ (State) ในแต่ละช่วงเวลาเรียงกันเป็นเส้นตรง ตัวดำเนินการทางตรรกศาสตร์ (Temporal Operators) มีดังนี้ [1, 2]

เมื่อกำหนดให้ φ คือเหตุการณ์ที่เกิดขึ้น ณ ช่วงเวลาหนึ่งๆ

ψ คือเหตุการณ์ที่ 2 ที่เกิดขึ้น ณ ช่วงเวลาหนึ่งๆ

2.1.1. ตัวดำเนินการแบบตลอดไป (Always)

ตัวดำเนินการแบบตลอดไป แทนด้วยสัญลักษณ์ $\Box$ สามารถเขียนแทนด้วยประโยค $\Box\varphi$ หมายถึงเหตุการณ์ φ ต้องเป็นจริงตลอด หลังจากเกิด $\Box\varphi$

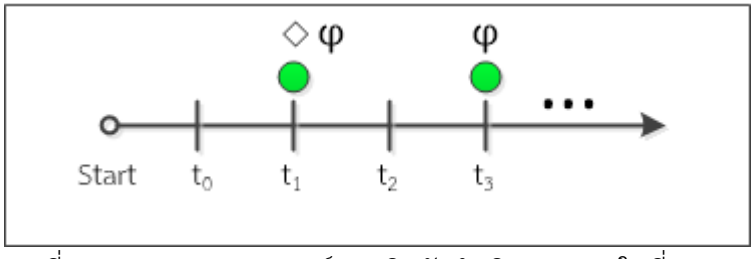


รูปที่ 2.1 แผนภาพเหตุการณ์การเกิดตัวดำเนินการแบบตลอดไป [2]

จากรูปที่ 2.1 หมายถึงเหตุการณ์ φ จะเป็นจริงเสมอ เมื่อเวลาใดๆ ที่เวลา t_1, t_2 และ t_3

2.1.2. ตัวดำเนินการแบบในที่สุด (Eventually)

ตัวดำเนินการแบบในที่สุด แทนด้วยสัญลักษณ์ $\Diamond$ สามารถเขียนแทนด้วยประโยค $\Diamond\varphi$ หมายถึงเหตุการณ์ φ ต้องเป็นจริง ณ เวลาใดๆ อย่างน้อย 1 ครั้ง หลังจากเกิด $\Diamond\varphi$

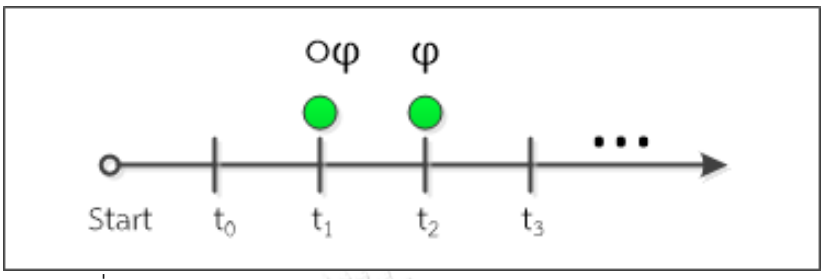


รูปที่ 2.2 แผนภาพเหตุการณ์การเกิดตัวดำเนินการแบบในที่สุด [2]

จากรูปที่ 2.2 หมายถึงเหตุการณ์ φ จะเป็นจริงในอนาคตอย่างน้อย 1 ครั้ง หลังเวลา t_1

2.1.3. ตัวดำเนินการแบบถัดไป (Next)

ตัวดำเนินการแบบถัดไป แทนด้วยสัญลักษณ์ $\bigcirc$ สามารถเขียนด้วยประโยค $\bigcirc\varphi$ หมายถึง เหตุการณ์ φ เป็นจริงหลังจากเกิด $\bigcirc\varphi$

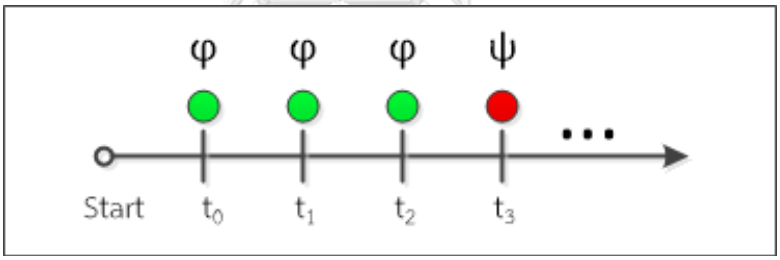


รูปที่ 2.3 แผนภาพเหตุการณ์การเกิดตัวดำเนินการแบบถัดไป [2]

จากรูปที่ 2.3 หมายถึงเหตุการณ์ φ จะเป็นจริงที่เวลา t_2 หลังจากเกิด $\bigcirc\varphi$

2.1.4. ตัวดำเนินการจนกระทั่งแบบเข้ม (Strong Until / Until)

ตัวดำเนินการจนกระทั่งแบบเข้ม แทนด้วยสัญลักษณ์ U สามารถเขียนเป็นประโยคได้เป็น $\varphi\ \mathrm{U}\ \psi$ หมายถึง เหตุการณ์ φ ต้องเหตุการณ์จะเป็นจริงได้ เมื่อ φ จะเป็นจริงเสมอ จนกระทั่ง ψ จะเป็นจริง ซึ่งจะทำให้ φ มีค่าความจริงเป็นค่าอะไรก็ได้

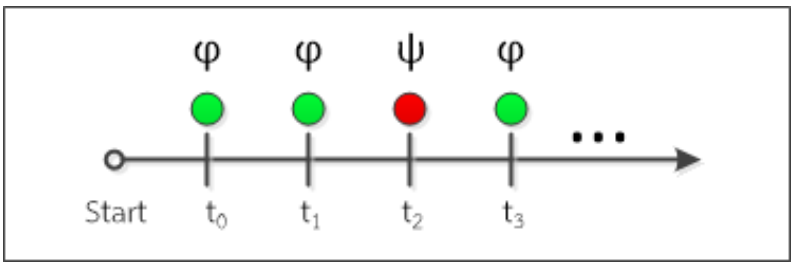


รูปที่ 2.4 แผนภาพเหตุการณ์การเกิดตัวดำเนินการจนกระทั่งแบบเข้ม [2]

จากรูปที่ 2.4 หมายถึงเหตุการณ์ φ เป็นจริง ที่เวลา t_0, t_1, t_2 และที่เวลา t_3 เหตุการณ์ ψ เป็นจริงทำให้ หลังจากเวลาที่ t_3 ค่าของเหตุการณ์ φ จะมีค่าความจริงเป็นอะไรก็ได้

2.1.5. ตัวดำเนินการจนกระทั่งแบบอ่อน (Weak Until/ Unless)

ตัวดำเนินการแบบจนกระทั่งแบบอ่อน แทนด้วยสัญลักษณ์ W สามารถขียนเป็นประโยคได้เป็น $\varphi\ \mathrm{W}\ \psi$ หมายถึงเหตุการณ์ที่ φ มีค่าเป็นจริงจนกระทั่ง ψ มีค่าเป็นจริง จะทำให้ φ เป็นจริงเสมอ



รูปที่ 2.5 แผนภาพเหตุการณ์การเกิดตัวดำเนินการแบบเป็นจริงในที่สุดแบบอ่อน [2]

จากรูปที่ 2.5 หมายถึงเหตุการณ์ φ จะเป็นจริงจนกระทั้งเหตุการณ์ ψ เป็นจริง จะทำให้เหตุการณ์ φ เป็นจริงตลอดกาล

2.2. ตรรกเชิงเวลาแบบเมตริก หรือเอ็มทีแอล (Metric Temporal Logic : MTL)

เอ็มทีแอล เป็นระบบข้อมูลจำเพาะที่โดดเด่นใช้สำหรับระบบเวลาจริง (Real time) และระบบไฮบริด (Hybrid Systems) ซึ่งขยายคุณลักษณะจากตรรกศาสตร์เชิงเส้นรวมกับช่วงเวลา (Time Interval) ซึ่งสามารถอ้างถึงคุณสมบัติความไม่ต่อเนื่องในเวลา (Discrete-timed) และรูปแบบที่เป็นไทม์แสตมป์ (Time stamped) [3]

นิยามของเอ็มทีแอล

กำหนดให้ $\mathcal{P}$ คือ เซตของประพจน์เดี่ยว (Atomic Propositions) โดย เอ็มทีแอลจะสร้างนิยามจาก $\mathcal{P}$ โดยใช้สัญลักษณ์ภายใต้ตัวดำเนินการ Until (U) สามารถเขียนได้ตามสมการที่ 2.1 [4]

$$\varphi ::= \rho | \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \, \mathrm{U}_{\mathrm{I}} \, \varphi \,, \qquad \text{สมการที่ 2.1}$$

เมื่อ $\rho \in \mathcal{P}$ และ $\mathrm{I} \subseteq (0, \infty)$ ซึ่งเป็นช่วงเวลาที่แท้จริงของจุดสิ้นสุด

กรณีมี 2 เหตุการณ์ที่เกิดขึ้น แทนด้วย φ_1 และ φ_2 การเกิด $\varphi_1 \, \mathrm{U}_{\mathrm{I}} \, \varphi_2$ จากการนิยามของเอ็มทีแอล จะได้ φ_2 จะเป็นจริงในบางครั้งจนกระทั่ง φ_1 เป็นจริง สร้างเป็นสมการฟังก์ชันได้เป็น [3]

$$f: \mathbb{R} \rightarrow 2^{P} \qquad \text{สมการที่ 2.2}$$

เมื่อให้ $f, r \in \mathbb{R}$ ดังนั้น เอ็มทีแอลสามารถสร้างความสัมพันธ์ ระหว่าง $f, r \vDash \varphi$ ได้ดังนี้

1) $f, r \vDash \rho$ iff $\rho \in f(r)$
2) $f, r \vDash \neg\varphi$ iff $f, r \nvDash \varphi$
3) $f, r \vDash \varphi_1 \wedge \varphi_2$ iff $f, r \vDash \varphi_1$ and $f, r \vDash \varphi_2$
4) $f, r \vDash \varphi_1 \, \mathrm{U}_{\mathrm{I}} \, \varphi_2$ iff there exists $t > r$ such that $t - r \in I$, $f, t \vDash \varphi_2$ and $f, u \vDash \varphi_1$ for all $u, r < u < t$

ตัวอย่างของเอ็มทีแอล [4]

$$\square(\text{alarm} \rightarrow (\Diamond_{[0,10]}\text{allclear} \vee \Diamond_{[10]}\text{shutdown}))$$

จากตัวอย่างความหมายคือ ทุกๆ ครั้งที่มีเหตุการณ์ alarm เกิดขึ้นแล้ว จะต้องมีเหตุการณ์ shutdown ณ เวลาที่ 10 $\Diamond_{[10]}$ เว้นเสียแต่ว่า จะเกิดเหตุการณ์ allclear $\Diamond_{[0,10]}$ ณ เวลาที่ 0 – 10 เป็นจริงอย่างน้อยหนึ่งครั้ง หรือเกิดขึ้นอย่างน้อยหนึ่งครั้ง

2.3. โปรแกรมสปิน (Simple Promela Interpreter – SPIN)

เป็นเครื่องมือสำหรับการตรวจสอบแบบจำลองซอฟต์แวร์ เพื่อติดตามและตรวจสอบความผิดพลาดในการออกแบบ เชิงตรรกะ บนพื้นฐานทฤษฎีออโตมาตา (Automata) ซึ่งสามารถเขียนการตรวจสอบด้วยภาษาที่เรียกว่า ภาษาโพรเมลา (Promela) อย่างไรก็ตามเครื่องมือสปินนี้ไม่ได้สร้างการตรวจสอบด้วยตนเอง แต่จะสร้างโค้ดด้วยภาษาซีขึ้นมาเพื่อช่วยตรวจสอบแบบจำลองซอฟต์แวร์ที่ต้องการ การใช้เทคนิคดังกล่าวช่วยให้ใช้หน่วยความจำได้อย่างมีประสิทธิภาพ [5-7]
โปรแกรมสปินสามารถทำงานใน 4 โหมดหลักๆ ดังนี้

1) แบบจำลอง (Simulator) สปินสามารถสร้างแบบจำลองต้นแบบอย่างรวดเร็วด้วยการจำลองแบบสุ่ม หรือแบบจำลองที่มีการตอบสนอง

2) เครื่องมือตรวจสอบอย่างละเอียด (Exhaustive Verifier) สปินมีความสามารถทวนสอบและพิสูจน์ความถูกต้องของความต้องการของผู้ใช้ โดยอาศัยวิธีรีดิวซ์คำสั่งบางส่วนเพื่อช่วยเพิ่มประสิทธิภาพในการค้นหาและทวนสอบ

3) ระบบประมาณค่าหลักฐาน (Proof Approximation System) สปินสามารถตรวจสอบในทุกๆ การดำเนินการในระบบที่มีขนาดใหญ่ซึ่งครอบคลุมสเตจสเปซ (State Space)

4) ไดร์ฟเวอร์สำหรับการตรวจสอบสวอร์ม (A Driver for swarm Verification) โดยใช้การประมวลผลแบบคู่ขนานกันเพื่อเพิ่มประสิทธิภาพและเพิ่มโอกาสในการตรวจสอบหาความผิดพลาดในโมเดลที่มีขนาดใหญ่

คุณสมบัติการตรวจสอบของสปิน สามารถตรวจสอบได้ดังต่อไปนี้

1) สภาวะหยุดชะงัก หรือสภาวะติดตาย (Deadlock)

2) ยืนยันการถูกละเมิด (Violated assertions)

3) โค้ดที่ไม่สามารถเข้าถึง (Unreachable code)

กรณีเกิดข้อผิดพลาด โปรแกรมจะตรวจพบร่องรอย เพื่อนำไปสู่สถานะหรือส่วนของตรรกะที่แสดงถึงความไม่ถูกต้องโดยในปัจจุบันโปรแกรมสปินถูกพัฒนาขึ้นที่เวอร์ชั่น 6.4.6 ในปี 2016 [5, 7]

2.4. การทวนสอบเชิงรูปนัย (Formal Verification) และการตรวจสอบแบบจำลอง (Model Checking)

การทวนสอบเชิงรูปนัยนิยมทำการทวนสอบด้วยวิธีที่แตกต่างกันออกไป โดยจุดประสงค์ของการทวนสอบเพื่อตรวจสอบว่าพฤติกรรมหรือคุณสมบัติของแบบจำลองว่าตรงตามข้อกำหนดความต้องการหรือไม่ และในทางกลับกันยังสามารถทวนสอบด้วยวิธีการหาความขัดแย้งของพฤติกรรมหรือคุณสมบัติของแบบจำลองได้อีกด้วย จากวิธีการทวนสอบสามารถทำได้ด้วยวิธีการคำนวณ (Deductive methods) ซึ่งใช้หลักการพิสูจน์ทางคณิตศาสตร์ และวิธีอัตโนมัติ (Automatic Methods) จะใช้วิธีการแจกแจงสถานะที่เป็นไปได้ทั้งหมดของระบบที่สามารถจะเกิดขึ้นได้ จากนั้นจึงตรวจสอบความถูกต้องของพฤติกรรมที่เกิดขึ้นในระบบว่าตรงตามข้อกำหนดความต้องการ ซึ่งการตรวจสอบดังกล่าวจะใช้เทคนิคที่เรียกว่าแบบบรูทฟอร์ซ (Brute-force) โดยเป็นที่มาของการทดสอบพฤติกรรมในทุกกรณีที่สามารถเกิดขึ้นได้ (Exhaustive Testing) แบบอัตโนมัติ ซึ่งกรณีที่ตรวจพบข้อผิดพลาดแบบจำลองจะทำการสร้างตัวอย่างค้าน (Counter example) [8]

2.5. ระบบเรียลไทม์ (Real-time System)

ระบบเรียลไทม์เป็นระบบปฏิบัติการจริงที่สามารถตอบสนองได้อย่างทันท่วงที และตลอดเวลาภายใต้ข้อจำกัดทางเวลา ซึ่งทำให้การจัดการเวลาและทรัพยากรที่มีสะดวกและรวดเร็วแม่นยำมากขึ้น ระบบเรียลไทม์สามารถแบ่งได้เป็น 2 ประเภท ได้แก่ [9, 10]

2.5.1. ฮาร์ดเรียลไทม์ซิสเต็ม (Hard Real-time System)

เป็นระบบที่ต้องสามารถตอบสนองได้ตลอดเวลา มีความแม่นยำค่อนข้างสูง และไม่สามารถหยุดทำงานได้ ซึ่งหากการทำงานส่วนใดส่วนหนึ่งในระบบล้มเหลว จะส่งผลให้ระบบล้มเหลวไปด้วย เช่น ระบบตลาดหุ้น ระบบธนาคาร เป็นต้น

2.5.2. ซอฟต์เรียลไทม์ซิสเต็ม (Soft Real-time System)

เป็นระบบที่สามารถตอบสนองได้ในเวลาที่จำกัด (Less Restrictive Type) แต่ในขณะเดียวกันก็สามารถรอให้งานอื่นทำงานจนสิ้นสุดได้เช่นกัน หรือในอีกมุมหนึ่งสามารถกล่าวได้ว่า ระบบจะสามารถตอบสนองได้ในเวลาที่จำกัดภายใต้ความสามารถของการประมวลผลที่มีอยู่ที่ทำให้การทำงานเป็นไปได้ เช่น การประมวลผลของระบบคอมพิวเตอร์ ซึ่งสามารถตอบสนองได้ในเวลาที่จำกัดภายใต้การประมวลผลของซีพียูที่สามารถจัดสรรให้ทำงานได้ เป็นต้น

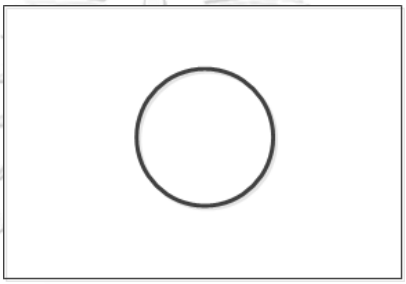
2.6. เพททริเนท (Petri-nets: PNs)

เพททริเนท (Petri-nets) เป็นการใช้การแสดงผลโดยอาศัยเส้น และสร้างรูปแบบทางคณิตศาสตร์เพื่อใช้อธิบายและวิเคราะห์กระบวนการต่างๆ ของระบบใดๆ ซึ่งสามารถแสดงได้ทั้ง การทำงานแบบคู่ขนาน การกระจายตัวของข้อมูล และการวิเคราะห์ประสิทธิภาพทั้งหมดของระบบ [11] กราฟเพททริเนทประกอบไปด้วย 4 ส่วน

2.6.1. ส่วนประกอบของกราฟเพททริเนท

1) เพลส (Places)

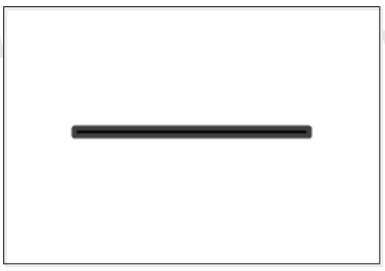
เพลสอธิบายเงื่อนไขของการเกิดเหตุการณ์หนึ่งๆ แทนด้วยรูปวงกลมโปรง และสัญลักษณ์ทางกราฟ ดังรูปที่ 2.6



รูปที่ 2.6 สัญลักษณ์ทางกราฟสำหรับเพลส [11]

2) ทรานซิชัน (Transitions)

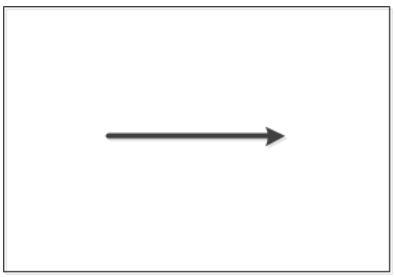
ทรานซิชันเป็นเหตุการณ์หรือการกระทำที่จะทำให้เสตจมีการเปลี่ยนแปลง แทนด้วยรูปสีเหลี่ยมผืนผ้าทึบ และสัญลักษณ์ทางกราฟ ดังรูปที่ 2.7



รูปที่ 2.7 สัญลักษณ์ทางกราฟของทรานซิชัน [11]

3) ไดเรคอาร์ค (Directed Arcs)

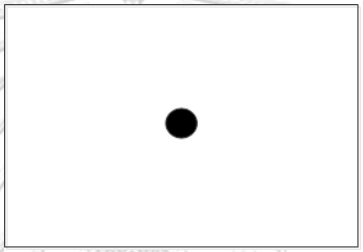
ไดเรคอาร์ค เป็นตัวบอกทิศทางของกระบวนการการทำงาน โดยเป็นเส้นที่มีหัวลูกศรไว้สำหรับแสดงทิศทางการทำงาน ซึ่งเป็นสัญลักษณ์เชื่อมจากเพลสไปยังทรานซิชัน หรือจากทรานซิชันไปยังเพลส แทนด้วยเส้นลูกศร และสัญลักษณ์ทางกราฟ ดังรูปที่ 2.8



รูปที่ 2.8 สัญลักษณ์ทางกราฟของไดเรคอาร์ค [11]

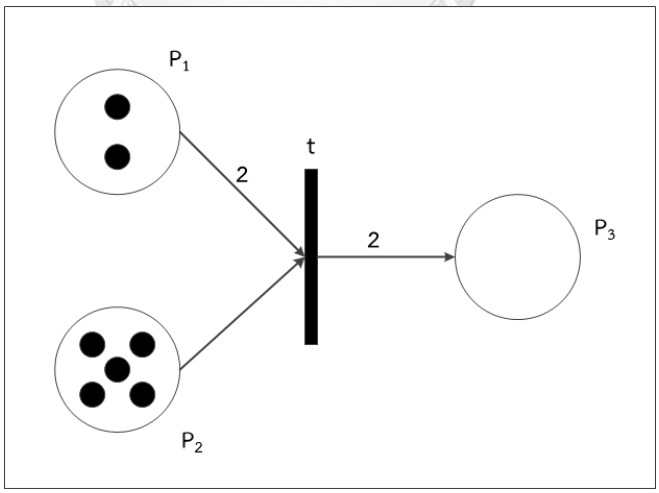
4) โทเคน (Tokens)

โทเคน เป็นตัวกระตุ้นในแต่ละเพลส เพื่อทำให้เพลสนั้นเป็นจริง ซึ่งโทเคนจะต้องอยู่ในเพลส และโทเคนจะเปลี่ยนเพลสได้ด้วยพฤติกรรมที่เรียกว่าการยิง (Fire) เขียนแทนด้วยจุดทึบ และสัญลักษณ์ทางกราฟ ดังรูปที่ 2.9



รูปที่ 2.9 สัญลักษณ์ทางกราฟของโทเคน [11]

ตัวอย่างเพททริเนท



รูปที่ 2.10 ตัวอย่างเพททริเนท [12]

เพททริเนทประกอบด้วย 5 องค์ประกอบ (5-Tuple) สามารถเขียนเป็นสมการได้ ดังนี้

$$S(\mathbb{Z}) = (P, T, F, W, m0) \qquad \text{สมการที่ 2.3}$$

จากสมการที่ 2.3 เมื่อกำหนด

$S(\mathbb{Z})$ เรียกว่า สเกเลตัน (Skeleton) ของ $\mathbb{Z}$

P แทน เซตของเพลส

T แทน ทรานซิชัน โดยที่ $P \cap T = \emptyset$

F แทน อินพุทฟังก์ชันหรือทิศทางการทำงาน (Flow Relation) ของแบบจำลองในระบบใดๆ โดยที่ $F \subseteq (P \times T) \cup (T \times P)$

W แทน ค่าน้ำหนักของทิศทางการทำงาน

m_0 คือจุดตั้งต้นของโทเคน

จากรูปที่ 2.10 และสมการที่ 2.3 สามารถเขียนความสัมพันธ์จากกราฟเพททริเนทและสมการได้ดังนี้

$P = \{P_1, P_2, P_3\}$

$T = \{t\}$

$F = \{ (P_1, t), (P_2, t), (t, P_3)\}$

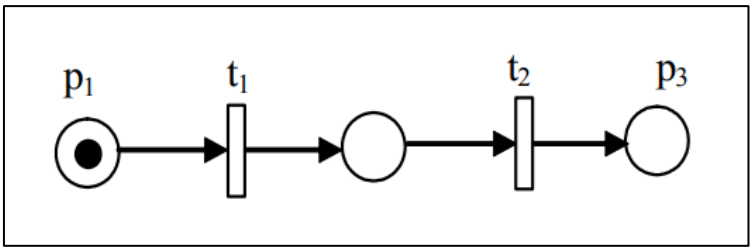
$W = \{ (P_1, t) \rightarrow 2, \{ (P_2, t) \rightarrow 1, (t, P_3) \rightarrow 2\}$

$m_0 = \{ P_1 \rightarrow 2, P_2 \rightarrow 5, P_3 \rightarrow 0\}$

2.6.2. ลักษณะพฤติกรรมของกราฟเพททริเนท

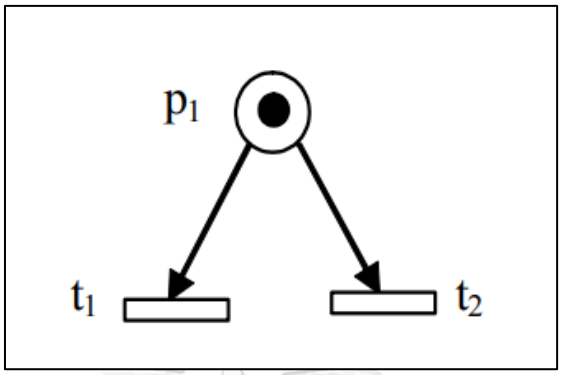
ลักษณะพฤติกรรมของเพททริเนทที่ถูกดำเนินการด้วยโทเคนหรือการดำเนินกิจกรรมของแบบจำลองใดๆ มีดังนี้

1) แบบลำดับ (Sequential Execution) เป็นการดำเนินกิจกรรมที่เป็นลำดับ จากรูปที่ 2.11 การยิงที่ t_2 จะเกิดขึ้นได้ก็ต่อเมื่อมีการยิงจาก t_1 แล้วเท่านั้น



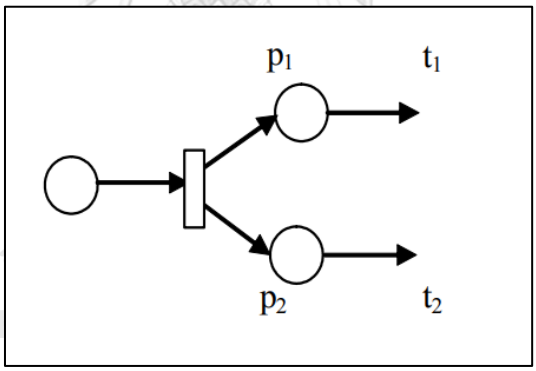
รูปที่ 2.11 พฤติกรรมแบบลำดับ [12]

2) แบบขัดแย้ง (Conflict) เป็นการดำเนินการที่โทเคนที่ตำแหน่งเพลสใดๆ สามารถยิงไปได้ทั้งสองทิศทาง จากรูปที่ 2.12 ที่ตำแหน่งของเพลส P_1 สามารถยิงโทเคนไปได้ทั้ง t_1 และ t_2 ซึ่งทำให้เมื่อโทเคนยิงออกไปทางใดทางหนึ่งแล้วอาจจะเกิดความขัดแย้งของระบบได้



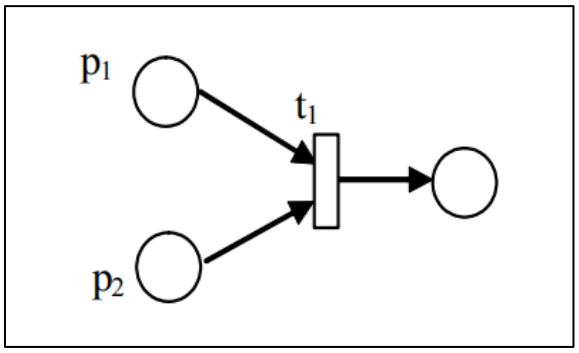
รูปที่ 2.12 พฤติกรรมแบบขัดแย้ง [12]

3) แบบทางแยก (Concurrency) เป็นพฤติกรรมที่ทรานซิชันสองทรานซิชันหรือมากกว่าสามารถเกิดขึ้นพร้อมกันได้เมื่อมีการยิงเพลสใดๆ มากกว่าสองตำแหน่งด้วยกัน จากรูปที่ 2.13 จะพบว่าที่ทรานซิชัน t_1 และ t_2 มีโอกาสที่จะเกิดขึ้นพร้อมกัน



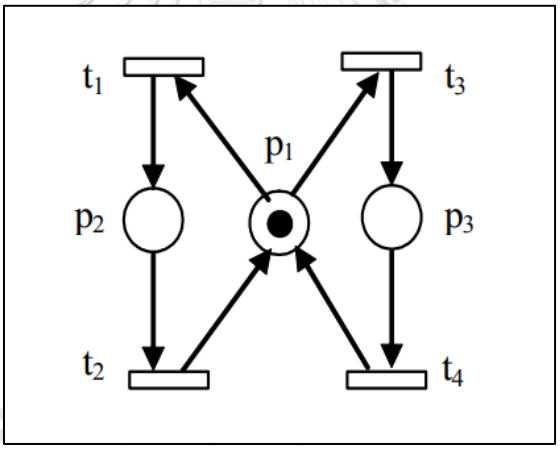
รูปที่ 2.13 พฤติกรรมแบบทางแยก [12]

4) แบบประสานเวลา (Synchronization) เป็นพฤติกรรมที่เป็นปกติสำหรับระบบไดนามิก ซึ่งเกิดจากการที่ทรานซิชันต้องการทรัพยากรมากกว่า 1 ตัว จากรูปที่ 2.14 ที่ทรานซิชัน t_1 ต้องการทรัพยากรจากเพลส P_1 และ P_2 จึงจะสามารถเปลี่ยนสถานะและดำเนินกิจกรรมต่อไปได้



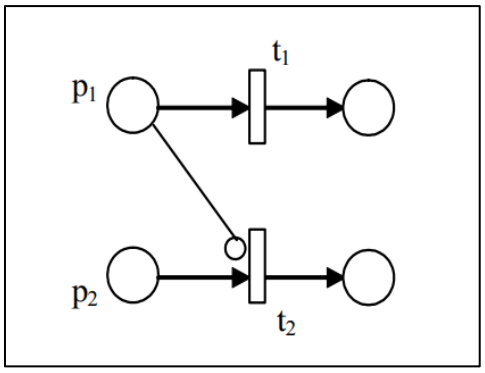
รูปที่ 2.14 พฤติกรรมแบบประสานเวลา [12]

5) แบบทำงานร่วมกัน (Mutually Exclusive) เป็นพฤติกรรมที่จะต้องประกอบด้วยกระบวนการ 2 กระบวนการที่ต้องทำงานร่วมกันในเวลาเดียวกัน และจะต้องใช้ทรัพยากรร่วมกัน จากรูปที่ 2.15 จะพบว่าที่ตำแหน่งทรานซิชัน t_1 และ t_3 ต้องการโทเคนที่ตำแหน่งเพลส P_1 ด้วยกันทั้งคู่ และในขณะเดียวกัน ตำแหน่งทรานซิชัน t_2 และ t_4 ก็จะต้องส่งทรัพยากรไปยังตำแหน่งเพลส P_1 เช่นกัน



รูปที่ 2.15 พฤติกรรมแบบ Mutually Exclusive [12]

6) แบบลำดับความสำคัญ (Priorities) เป็นพฤติกรรมที่ใช้สำหรับการควบคุมการทำงานของกระบวนการในแบบจำลอง จากรูปที่ 2.16 ที่ตำแหน่งทรานซิชัน t_2 จะไม่สามารถทำงานหรือดำเนินกิจกรรมต่อได้ จนกว่าที่เพลส P_1 จะปล่อยโทเคนมาให้



รูปที่ 2.16 พฤติกรรมแบบลำดับความสำคัญ [12]

2.7. ไทม์เพททริเนท (Timed Petri-nets: TPN)

ไทม์เพททริเนท (Timed Petri-nets) เป็นรูปแบบที่เป็นส่วนต่อขยายมาจากเพททริเนท ใช้อธิบายระบบไดนามิค (Dynamic System) ซึ่งจะระบุเวลาเริ่มและเวลาของการดำเนินการในแต่ละกระบวนการย่อยเพื่อเป็นตัวเปลี่ยนสถานะของโทเคนจากเพลสปัจจุบันไปยังเพลสตัวถัดไปในลำดับการทำงานของระบบ โดยที่ค่าขอบเวลาเริ่มต้น และขอบเวลาสิ้นสุดจะต้องเป็นค่าเดียวกัน ซึ่งมีนิยามที่สำคัญ 3 อย่าง ได้แก่ โครงสร้างโทโพโลจี (The Topological Structure), การกำหนดค่าเลเบล (The Labeling of The Structure) และกฎการยิง (Firing Rules) ประกอบด้วย 6 องค์ประกอบ (6-Tuple) สามารถเขียนเป็นสมการ ดังนี้[12]

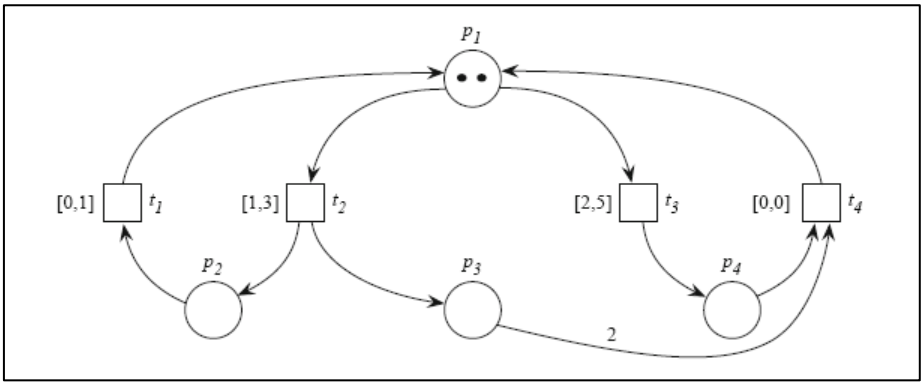
$$S(\mathbb{Z}) = (P, T, F, W, m_0, SI) \qquad \text{สมการที่ 2.4}$$

จากสมการที่ 2.4 เมื่อกำหนด

$S(\mathbb{Z}) = (P, T, F, W, m_0)$ เป็นสมการของเพททริเนท

SI เป็นฟังก์ชัน เรียกว่า ฟังก์ชันสเตติกอินเทอร์วอล (Static Interval Function)

ตัวอย่างไทม์เพททริเนท



รูปที่ 2.17 ตัวอย่างไทม์เพททริเนท [11]

2.8. งานวิจัยที่เกี่ยวข้อง

2.8.1. Translating Basic Metric Temporal Logic Formulas into Promela., Punwess Sukvanich, Arthit Thongtak, Wiwat Vatanawood. 2016 [13]

งานวิจัยดังกล่าวนำเสนอการแปลงคุณลักษณะในรูปแบบของเอ็มทีแอลพื้นฐานไปยังรูปแบบของแอลทีแอลด้วยภาษาโพรเมลา ซึ่งทำการแปลงคุณลักษณะของการดำเนินการ 2 ตัวดำเนินการ คือ Always และ Eventually ณ ช่วงเวลาใดๆ เพื่อสร้างแบบจำลองทวนสอบกับระบบนาฬิกาแบบเรียลไทม์ ที่ใช้หลักการนาฬิกาหลัก (Global clock) ซึ่งมีพฤติกรรมเป็นลูปโดยไม่มีที่สิ้นสุด ที่มีพฤติกรรมคล้ายกับระบบเรียลไทม์ และทำการเดินของนาฬิกาจนครบรอบนาฬิกา ที่เรียกว่า TotalTickPerLoop และจะทำการเริ่มต้นนาฬิกาดังกล่าวเมื่อนาฬิกาเดินครบรอบ โดยทำการแบ่งออกเป็น 3 ส่วนหลักๆ ได้แก่

1) เอ็มทีแอลพารามิเตอร์ (MTL Parameters) ซึ่งจะประกอบด้วย 6 พารามิเตอร์ ดังรูปที่ 2.18

```
1 typedef MTLevent{
2    bool    Event;           /*Event*/
3    bool    Reaction;        /*Corresponding Event*/
4    bool    EventFlag;       /*Event flag */
5    int     EventCounter;    /*Event Counter */
6    int     pTime1;          /*Left Boundary*/
7    int     pTime2;};        /*Right Boundary*/
```

รูปที่ 2.18 Promela Definition of MTL Parameters [13]

2) เอ็มทีแอลเคาน์ติ้ง (MTL Counting) เป็นการกำหนดจุดเริ่มต้นและนับจำนวนเหตุการณ์ที่เกิดขึ้นและรับรู้ถึงเหตุการณ์ที่หมดเวลาอีกด้วย ดังรูปที่ 2.19


```
1 inline MTLCounter(CauseEvent){
2    if:: (CauseEvent.Event) ->
3       if::!CauseEvent.EventFlag->CauseEvent.EventCounter = 0;
4          CauseEvent.EventFlag = true;
5          ::else->skip;
6       fi;
7       ::(!CauseEvent.Event&&CauseEvent.EventFlag)->
8          if::((CauseEvent.EventCounter<=CauseEvent.pTime2)
9             &&(CauseEvent.EventCounter!=expired))->
10            CauseEvent.EventCounter++;
11            :: else->CauseEvent.EventCounter=expired;
12            CauseEvent.EventFlag = false;
13         fi;
14   ::else->skip;
15 fi;}
```

รูปที่ 2.19 ตัวอย่างเอ็มทีแอลเคาน์ติ้งในภาษาโพรเมลา [13]

3) การแปลงแอลทีแอล (Translated LTL) กำหนดเป็น 4 รูปแบบสำหรับการแปลงจากรูปแบบของเอ็มทีแอลไปยังรูปแบบของแอลทีแอล ดังรูปที่ 2.20

$\Diamond_{(t1,t2)}p$

```
#define MTL_Fpp(effect) <>(effect.Reaction&&(effect.EventCounter>
 effect.pTime1)&&(effect.EventCounter<effect.pTime2))
```

$\Diamond_{[t1,t2)}p$

```
#define MTL_Fss(effect) <>(effect.Reaction&&(effect.EventCounter>=
 effect.pTime1)&&(effect.EventCounter<=effect.pTime2))
```

$\Box_{(t1,t2)}p$

```
#define MTL_Gpp(effect) [](!((effect.EventCounter>effect.pTime1)&&
 (effect.EventCounter<effect.pTime2))||((effect.Reaction
 &&(effect.EventCounter>effect.pTime1))&&(effect.EventCounter
 <effect.pTime2)))
```

$\Box_{[t1,t2)}p$

```
#define MTL_Gss(effect) [](!((effect.EventCounter>=effect.pTime1)&&
 (effect.EventCounter<=effect.pTime2))||((effect.Reaction&&
 (effect.EventCounter>=effect.pTime1))&&(effect.EventCounter
 <=effect.pTime2)))
```

รูปที่ 2.20 รูปแบบการแปลงเอ็มทีเอลเป็นแอลทีแอล [13]

ผลของการทวนสอบสำหรับระบบเรียลไทม์ ที่ประกอบด้วย 3 งาน ได้แก่ Alarm, AllClear, Shutdown โดยที่แบบจำลองทวนสอบจะแสดงสคีมาความสำเร็จที่ถูกต้อง ซึ่งสามารถสรุปได้ว่ารูปแบบของแอลทีแอลที่ถูกแปลงจากรูปแบบเอ็มทีแอลนั้นถูกต้อง

2.8.2. The Specification and Verification of Real-time System Based on the Temporal Logic of Action., Tang Zheng-yi, Peng Chang-gen, Li lun-tao, Li Xiang. 2010 [9]

งานวิจัยนี้นำเสนอทฤษฎีและคำนิยามของไทม์ออโตมาตา (Timed Automata) ซึ่งเป็นรูปแบบทฤษฎีสำหรับการวิเคราะห์และตรวจสอบระบบอัตโนมัติแบบเรียลไทม์แต่อย่างไรก็ตามการพัฒนารูปแบบนั้นยังพบข้อบกพร่องเกิดขึ้น ดังนั้นจึงนำตรรกศาสตร์ของการกระทำ (Temporal Logic of Action: TLA) โดยศึกษาวิธีการที่ใช้ตรรกศาสตร์ของการกระทำอธิบายระบบเรียลไทม์ ซึ่งด้วยในการแก้ปัญหาสเตจสเปซ (State-space) จะแสดงด้วยไทม์ออโตมาตาและการยืนยันกับกรณีศึกษาจริงเพื่อการันตีระบบว่ามีความน่าเชื่อถือ และดำเนินการอย่างถูกต้อง

ไทม์ออโตมาตาจะอธิบายสำหรับ อัลเลอ (Alur) และ ดิล (Dill) โดยสร้างขึ้นมาด้วยการเพิ่มกลไลของเวลาเข้าไป อย่างไรก็ตามไทม์ออโตมาตายังคงกำหนดจำนวนของตัวควบคุมตำแหน่งและเวลาของนาฬิกาจริง

คำนิยาม (Syntax) ของไทม์ออโตมาตา มีดังนี้

1) สร้างนาฬิกาและองค์ประกอบ (Clock and the set of clock constraints) โดยกำหนดให้

X แทน เซตจำกัดของออโตมาตา (the finite set of states)

C(X) แทน เซตขององค์ประกอบของนาฬิกา สามารถสร้างสัญลักษณ์ได้ดังนี้

$$\Phi ::= (x \sim c) \mid \Phi_1 \wedge \Phi_2 \mid \text{true, where } x \in X,\ \sim \in \{<, \leq, \geq, >\},\ c \in N^{+}。$$

รูปที่ 2.21 นิยามสำหรับการสร้างนาฬิกาและองค์ประกอบ [9]

2) การกำหนดค่าให้นาฬิกา (Assignment of Clock) โดยกำหนดให้

$$\mu:\ X \mid \longrightarrow R^{+}, \mu + t$$

รูปที่ 2.22 คำนิยามกำกำหนดค่าให้นาฬิกา [9]

3) ไทม์ออโตมาตา (Timed Automata) โดยกำหนดให้

Q แทน เซตจำกัดของออโตมาตา (The finite set of states)

I เป็นสมาชิกของ Q ที่เป็นเซตของสถานะเริ่มต้น (The set of initial states)

∑ แทน เซตจำกัดของการแปลง (The finite set of transfer label)

L แทน การกำหนดค่าเวลาให้กับแต่ละโหนด

E แทน เซตการแปลงค่า (The set of transfer)

$$X: TA = \langle Q, I, \Sigma, X, L, E \rangle$$

รูปที่ 2.23 คำนิยามสำหรับการกำหนดค่าให้นาฬิกา [9]

4) การแปลงความสัมพันธ์ (Transfer Relationship) แบ่งเป็น 2 แบบคือ การแปลงแบบหน่วง และการแปลงแบบไม่ต่อเนื่อง ดังรูปที่ 2.24

Delayed transfer: $(q, \mu) \xrightarrow{\delta} (q, \mu')$, if $\mu \models L(q)$ and $\mu' \models L(q)$, where $\mu' = \mu + \delta$, $\delta \in R^+$.

Discrete transfer: $(q, \mu) \xrightarrow{\sigma} (q', \mu')$, if$(q, \Phi, \sigma, Y, q') \in E$, $\mu \models \Phi$ and $\mu' \models L(q')$, where $\mu' = \mu[Y := 0]$.

รูปที่ 2.24 คำนิยามสำหรับการแปลงความสัมพันธ์ [9]

ความหมายของตรรกศาสตร์ของการกระทำ (The main semanticses TLA) มีดังนี้

1) การกำหนดตัวแปรเริ่มต้น (Primed Variable) ค่าตัวแปรที่ถูกกำหนดขึ้นมี V และ V′
2) การกระทำ (Action) เป็นขั้นตอนการสร้างตัวแปร การกำหนดตัวแปรเริ่มต้น และนิพจน์บูลีน
3) สตัสเตอริ่งสเต็ป (Stuttering Step) เป็นขั้นตอนที่เมื่อเกิดขึ้นแล้วระบบและตัวแปรใดๆ จะไม่มีการเปลี่ยนแปลง
4) ระบบการแปลงค่า (Label Transfer System)
5) รัน (Run) ขั้นตอนการรันทีแอลเอ
6) การเปิดใช้งาน (Enabled) เป็นการเปิดใช้งาน ณ สถานะใดๆ
7) วีคแฟร์เนส (Weak Fairness)
8) สตรองแฟร์เนส (Strong Fairness)

หลังจากการกำหนดคำนิยามและความหมายของตรรกศาสตร์การกระทำเรียบร้อยแล้ว เป็นขั้นตอนการตรวจสอบคุณสมบัติดังกล่าว ซึ่งตัวอย่างที่ถูกยกขึ้นมาเพื่อตรวจสอบคือสัญญาณไฟจราจร ซึ่งมีคุณสมบัติดังรูปที่ 2.25

```
Init == ∧ t = 0          Red == ∧ gLight = 1     Green == ∧ rLight = 1
        ∧ rLight = 0            ∧ t >= 3                 ∧ t >= 3
        ∧ gLight = 1            ∧ rLight' = 1            ∧ gLight' = 1
                                ∧ gLight' = 0            ∧ rLight' = 0
                                ∧ t' = 0                 ∧ t' = 0
```

รูปที่ 2.25 สถานะเริ่มต้นของระบบสัญญาณไฟจราจร [9]

สำหรับการตรวจสอบจะแบ่งออกเป็น 2 ขั้นตอน ดังนี้

1) ข้อจำกัดพิเศษร่วม (Mutually Exclusive)

สถานะพิเศษที่ถูกกำหนดร่วมกันเป็นปัจจัยที่เป็นตัวกำหนดการทำงานของระบบสัญญาณไฟจราจร นั่นคือ ค่าของสถานะไฟใดๆ ที่สีเดียวกันจะต้องไม่เป็นจริงพร้อมกัน ณ เวลาใดๆ สามารถเขียนข้อกำหนดได้เป็นดังรูปที่ 2.26

$$\begin{aligned} \text{Mutex} == \vee \wedge\ & \text{rLight} = 0 \\ \wedge\ & \text{gLight} = 1 \\ \vee \wedge\ & \text{rLight} = 1 \\ \wedge\ & \text{gLight} = 0 \end{aligned}$$

รูปที่ 2.26 สถานะพิเศษของระบบสัญญาณไฟจราจร [9]

2) สถานะถาวร (State Persistence)

ระบบมีความต้องการให้ในทุกๆ สถานะไฟแต่ละสี จะต้องสามารถดำเนินการใน 3 หน่วยเวลา นั่นคือในกรณีที่สถานะของไฟสีแดงเป็นจริง ซึ่งจะทำให้นาฬิกาเพิ่มเวลาจนกระทั่งถึง 3 หน่วยเวลาในที่สุดดังรูปที่ 2.27

$$\begin{aligned} \text{RedLast} &== (\text{rLight} = 1) \Rightarrow (t = 3) \\ \text{GreenLast} &== (\text{gLight} = 1) \Rightarrow (t = 3) \end{aligned}$$

รูปที่ 2.27 การกำหนดสถานะของไฟจราจรแต่ละสีรวมกับเวลา [9]

ซึ่งจากผลการตรวจสอบและทวนสอบตัวอย่างระบบไฟจราจรสามารถสรุปได้ว่าแบบจำลองทีแอลซี ที่ถูกสร้างขึ้นนั้นสามารถตรวจสอบคุณสมบัติดังกล่าวได้อย่างถูกต้อง

2.8.3. Formal Verification of User-Level Real-Time Property Patterns., Ning Ge, Marc Pantel, Silvano Dal Zilio. 2017 [14]

งานวิจัยนี้นำเสนอการกำหนดเซตของรูปแบบของคุณลักษณะในระบบเรียลไทม์ โดยมีวัตถุประสงค์เพื่อเพิ่มประสิทธิภาพของข้อกำหนดความต้องการของระบบเรียลไทม์ (Real-time Requirement) คุณลักษณะของระบบเรียลไทม์ ประกอบด้วย 2 คุณลักษณะเด่นๆ คือ รูปแบบ (Patterns) และข้อจำกัด (Scope) ซึ่งศึกษาเพิ่มเติมจากงานวิจัยของ Dwyer ซึ่งได้นิยาม Dwyer's patterns มี 8 รูปแบบ ได้แก่ Absence, Existence, Bounded Existence, Precedence, Response, Chain Precedence, Chain Response พร้อมกับกำหนดอีก 5 ข้อจำกัด ได้แก่ Global, Before, After, Between, After-Until นอกจากนี้ยังศึกษาเพิ่มเติมจากงานวิจัยของ Konrad ซึ่งนิยาม Konrad's patterns ซึ่งได้ทำการปรับปรุงและนิยามจำนวนความต้องการขึ้นมาเพิ่มเติมอีก 5 ความต้องการได้แก่ Minimum Duration, Maximum Duration, Bounded Recurrence, Bounded Response, Bounded Invariance โดยงานวิจัยนี้ได้ทำการเพิ่มรูปแบบของคุณลักษณะในระบบเรียลไทม์ ดังนี้

1) เพิ่มข้อจำกัดที่มีชื่อว่า Periodically
2) กำหนดคำนำหน้าใหม่ ได้แก่ At least, At most, Within
3) กำหนดเซตของโครงสร้างพื้นฐาน

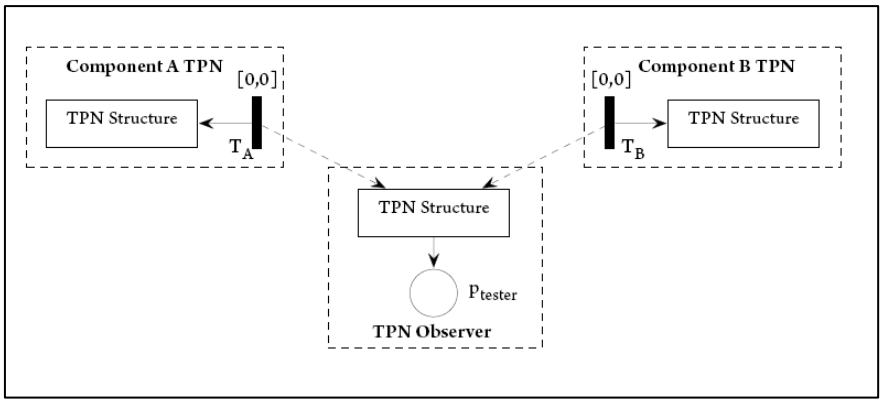
หลักการออกแบบไทม์เพททริเนทสำหรับออฟเซิร์ฟเวอร์ (Timed Petri Nets Observer) มีดังต่อไปนี้

1) Structure of Observer ซึ่งเป็น Sub-net ที่มีตัวตรวจจับพฤติกรรมของระบบ ในการประเมินคุณสมบัติตามสถานะการณ์ แสดงดังรูปที่ 2.28

2) Soundness of Observer สำหรับออฟเซิร์ฟเวอร์ที่ถูกขยายเพิ่มขึ้นมา ไม่ควรที่จะมีผลกระทบต่อระบบเดิมที่ทำให้ระบบเดิมไม่สามารถทำงานได้ หรือเกิดสภาวะติดตายเกิดขึ้น ซึ่งในบางครั้งอาจเกิดการเพิ่มโทเคนหรือลบโทเคนออกจากเพลสใดๆ ในระบบ ดังนั้นงานวิจัยนี้จึงได้กำหนดให้มีคุณลักษณะเพียงแค่อ่านอย่างเดียวเท่านั้น (Read-only)

3) Efficiency of Observer การเพิ่มประสิทธิภาพ งานวิจัยนี้ได้ใช้หลักปฏิบัติ 3 ข้อเพื่อทำให้ออฟเซิร์ฟเวอร์มีประสิทธิภาพสูงสุด ดังนี้

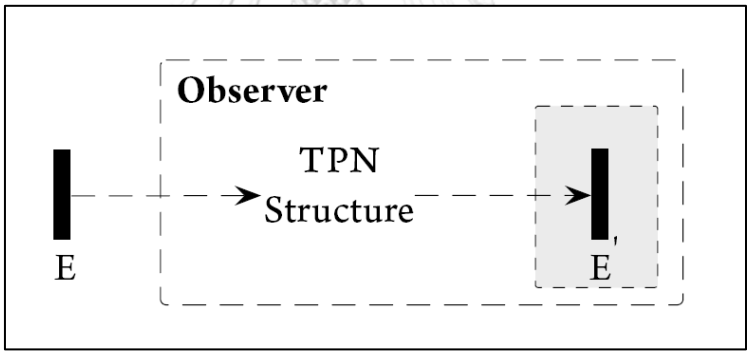
a. การสร้างกราฟที่เป็นแอฟสเตรกระดับสูง (High-level abstraction)
b. สร้างสถานะสเปซ (State Space) ที่มีขนาดเล็กที่สุดเท่าที่จะเป็นไปได้
c. การตรวจสอบคุณสมบัติของรูปแบบจะต้องเป็นเอกเทศไม่มีความเกี่ยวข้องหรือเชื่อมโยงกัน และสามารถทำงานขนานกันได้



รูปที่ 2.28 โครงสร้างของไทม์เพททริเนทออฟเซิร์ฟเวอร์ [14]

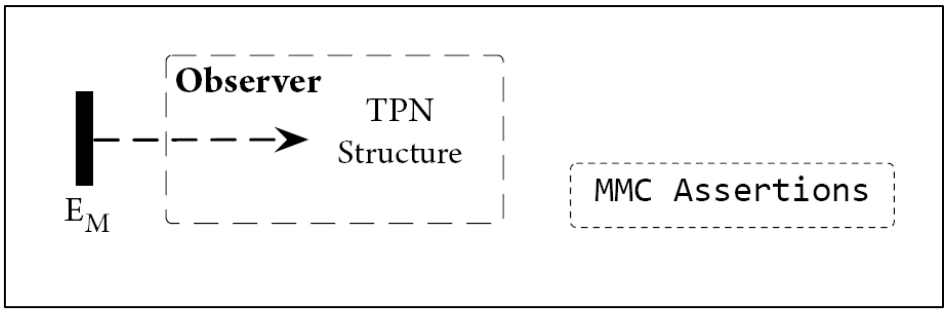
รูปแบบของคุณลักษณะเบื้องต้นของออฟเซิร์ฟเวอร์สำหรับการทวนสอบได้ถูกกำหนดขึ้นภายใต้หัวข้อดังต่อไปนี้

1) Basic Event Modifiers ซึ่งทำการเพิ่มเหตุการ์เบื้องต้นของออฟเซิร์ฟเวอร์ได้แก่ E^{i}, E^{-k}, $E^{/k}$, 1+t, E+1, S^{S} & S^{E} ซึ่งในหัวข้อนี้จะทำการเพิ่มหลักการของเวลาเริ่มและเวลาสิ้นสุด หรือการหน่วงเวลาให้กับเหตุการณ์เบื้องต้นของออฟเซิร์ฟเวอร์



รูปที่ 2.29 โครงสร้างของออฟเซิร์ฟเวอร์สำหรับการเหตุการณ์ต่อขยาย [14]

2) Basic Predicates รูปแบบของคุณสมบัติที่สร้างความน่าเชื่อถือให้เหตุการณ์หรือสถานะที่เกิดขึ้น โดยอ้างอิงจากรูปแบบของคุณสมบัติที่ถูกสร้างขึ้นจากโครงสร้างพื้นฐานของไทม์เพททริเนท จากรูปที่ 2.30 แสดงรูปแบบของเหตุการณ์ในออฟเซิร์ฟเวอร์ โดยอาศัยหลักการยืนยันด้วยเอ็มเอ็มซี (*mmc* Assertions)

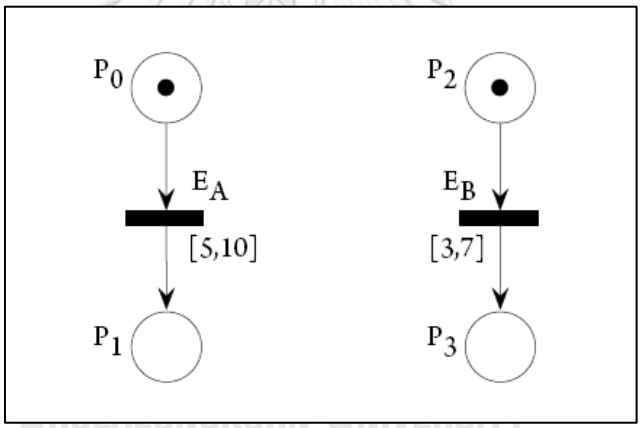


รูปที่ 2.30 รูปแบบพฤติกรรมของออฟเซิร์ฟเวอร์ [14]

3) Basic Scope Modifiers จะกำหนดข้อจำกัดทั้งหมดดังนี้ Global, Before E^i & After E^i, Between E_A and E_B, After E_A Until E_B

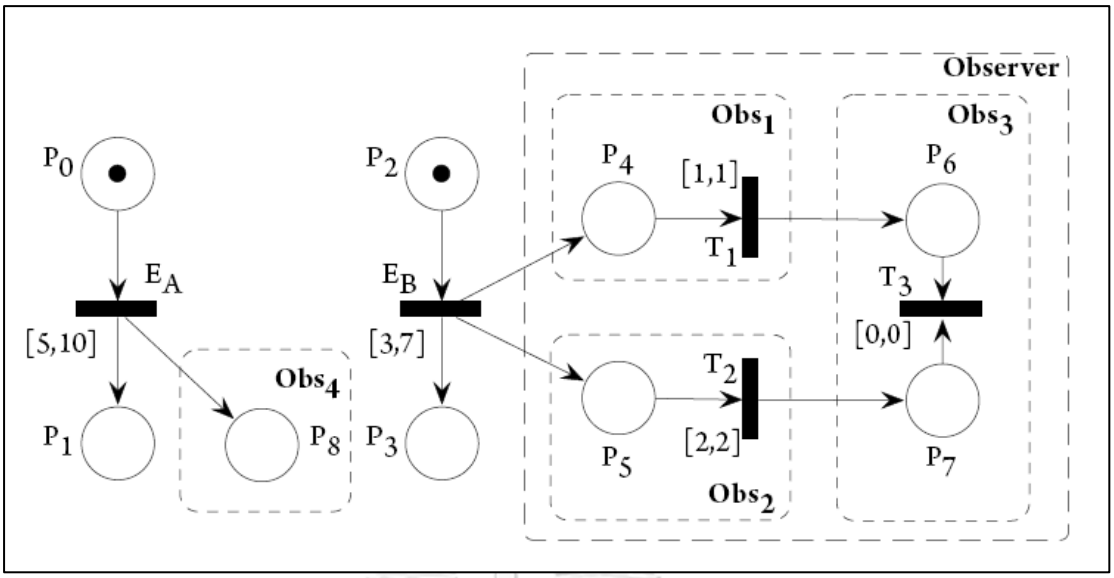
4) Occurrence Modifiers การปรับเปลี่ยนตัวดำเนินการของเหตุการณ์หรือสเตจ ซึ่งตัวดำเนินการที่ถูกปรับเปลี่ยนมีดังนี้ Exist, Absent, Always

หลังจากทำการปรับเปลี่ยนคุณสมบัติตามข้อกำหนดที่ได้กล่าวมาแล้วนั้น การทวนสอบคุณสมบัติของระบบเรียลไทม์ จะใช้แบบจำลองตามรูปที่ 2.31 เพื่อทวนสอบ

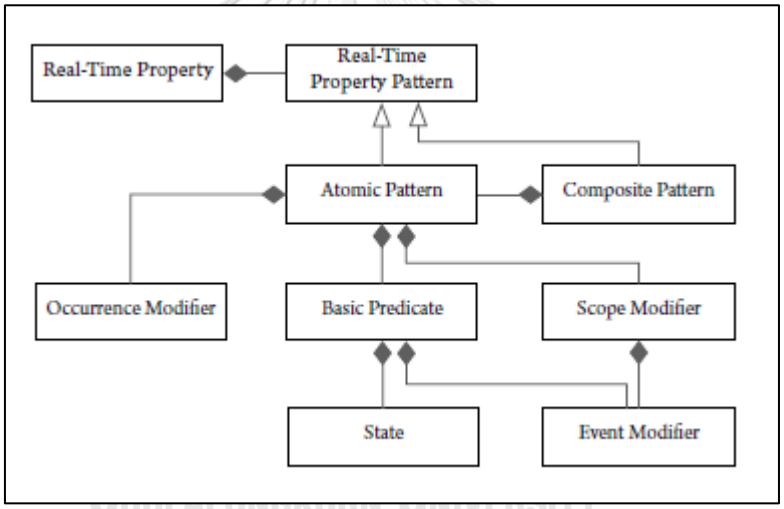


รูปที่ 2.31 ตัวอย่างแบบจำลองสำหรับการทวนสอบคุณสมบัติระบบเรียลไทม์ [14]

หลังจากนำแบบจำลองดังกล่าวประกอบเข้ากับคุณสมบัติของระบบเรียลไทม์ที่ถูกกำหนดเพิ่มขึ้น ผลลัพธ์ของงานวิจัยสามารถสร้างขึ้นได้ตามรูปที่ 2.32 เพื่อทวนสอบคุณสมบัติและการจัดกลุ่มเพื่อสร้างเซตของรูปแบบคุณลักษณะได้ดังรูปที่ 2.33



รูปที่ 2.32 ตัวอย่างไทม์เพททริเนทของออฟเซิร์ฟเวอร์ [14]



รูปที่ 2.33 เซตของรูปแบบของคุณลักษณะ [14]

บทที่ 3
งานวิจัยและกรณีศึกษาระบบแบบจำลองการบรรจุซองใบแจ้งยอดหนี้บัตรเครดิต

เนื้อหาในบทนี้จะกล่าวถึงภาพรวมของงานวิจัย ลำดับการทำงานของงานวิจัย และอธิบายการทำงานของแบบจำลองสำหรับกรณีศึกษาของการบรรจุซองใบแจ้งยอดหนี้บัตรเครดิต

3.1 หลักการและลำดับการทำงานของงานวิจัย

งานวิจัยสามารถแบ่งการทำงานออกเป็น 2 ฝั่งคือ สายงานที่ 1 ซึ่งเป็นการการวิเคราะห์ลำดับการทำงานของแบบจำลองซึ่งจะทำการวิเคราะห์ความต้องการของระบบและข้อจำกัดของระบบ และสายงานที่ 2 เป็นการกำหนดตัวดำเนินการที่จะทำการแปลง และกระบวนการแปลงคุณสมบัติจากรูปแบบเอ็มทีแอลให้อยู่ในรูปของแอลทีแอลเพื่อทำการทวนสอบแบบจำลองที่สร้างขึ้นในขั้นตอนสุดท้าย ซึ่งจะแสดงในรูปที่ 3.1

3.1.1 สายงานที่ 1 การวิเคราะห์และออกแบบโมเดลของแบบจำลองตามกรณีศึกษา

มีขั้นตอนดังนี้

1) กำหนดกรณีศึกษาของระบบการบรรจุซองใบแจ้งยอดหนี้บัตรเครดิต

2) วิเคราะห์และออกแบบข้อกำหนด ความต้องการของกรณีศึกษาที่สร้างขึ้น โดยกำหนดข้อจำกัดและลำดับการทำงานของกรณีศึกษาตั้งแต่ต้นจนจบ โดยการสร้างลำดับการทำงานให้อยู่ในแผนภาพลำดับการทำงานของระบบ

3) ออกแบบแผนภาพจากพฤติกรรมของแบบจำลองกรณีศึกษาที่เกิดขึ้นจากแผนภาพลำดับการทำงานที่ได้ออกแบบไว้ให้อยู่ในรูปแบบของไทม์พริทริเนท

4) พัฒนาแบบจำลองด้วยภาษาโพรเมลา บนโปรแกรมสปิน โดยอ้างอิงจากแผนภาพฤติกรรมของแบบจำลองในรูปแบบของไทม์แพททริเนท

3.1.2 สายงานที่ 2 การกำหนดและแปลงคุณลักษณะของตัวดำเนินการในรูปแบบเอ็มทีแอลไปเป็นรูปแบบแอลทีแอล

มีขั้นตอนดังนี้

1) ทำการกำหนดคุณลักษณะของตัวดำเนินการที่ต้องการแปลงจากรูปแบบเอ็มทีแอลไปเป็นรูปแบบแอลทีแอล มี 5 คุณลักษณะดังต่อไปนี้

(1) ตัวดำเนินการแบบตลอดไป ($\Box_{[t1, t2]}\varphi$)

(2) ตัวดำเนินการแบบในที่สุด ($\Diamond_{[t1, t2]}\varphi$)

(3) ตัวดำเนินการแบบถัดไป ($\bigcirc_{[t1, t2]}\varphi$)

(4) ตัวดำเนินการจนกระทั่งแบบเข้ม ($\varphi \, U_{[t1, t2]} \, \psi$)

(5) ตัวดำเนินการจนกระทั่งแบบอ่อน ($\varphi \, W_{[t1, t2]} \, \psi$)

2) สร้างเงื่อนไขการแปลงคุณลักษณะของพฤติกรรมที่ต้องการทวนสอบภายใต้ข้อจำกัดที่กำหนดขึ้น ให้อยู่ในรูปแบบของเอ็มทีแอล

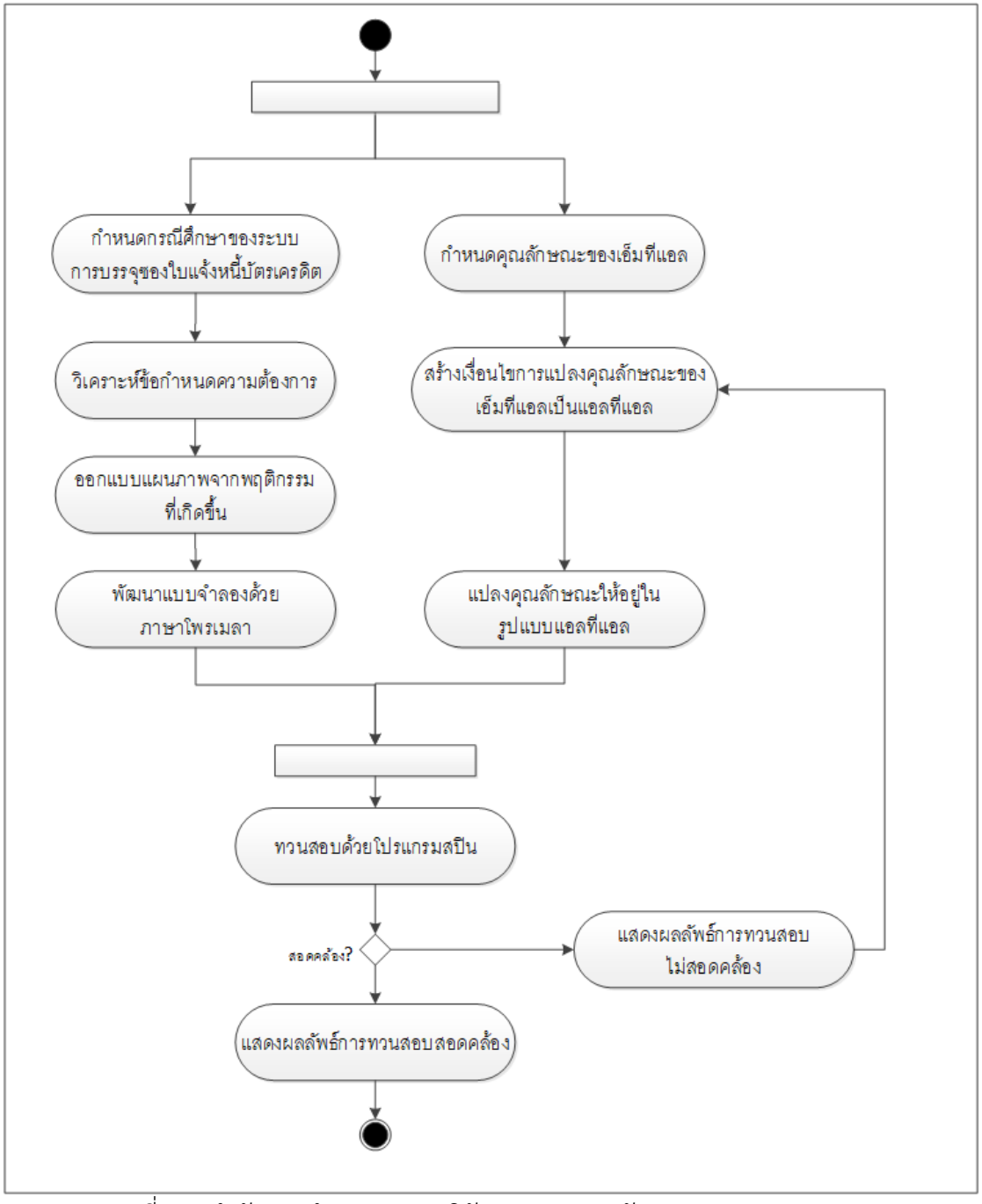
3) ทำการแปลงคุณลักษณะดังกล่าวให้อยู่ในรูปแบบของแอลทีแอลโดยภายใต้กรอบโมเดลของแอลทีแอล (LTL Template)

3.1.3 กระบวนการการทวนสอบ

1) นำแบบจำลองที่ถูกพัฒนาขึ้นด้วยภาษาโพรเมลาจากสายงานที่ 1 นำมาทวนสอบกับผลลัพธ์ของการแปลงคุณลักษณะจากรูปแบบเอ็มทีแอลเป็นรูปแบบแอลทีแอล ดังกล่าว

2) ทำการทวนสอบด้วยโปรแกรมสปินด้วยเงื่อนไขที่กำหนดขึ้น โดยที่เงื่อนไขที่จะนำมาทำการทวนสอบแบบจำลองกับรูปแบบแอลทีแอลนั้น จะครอบคลุมเงื่อนไขที่สอดคล้องและเงื่อนไขค้าน

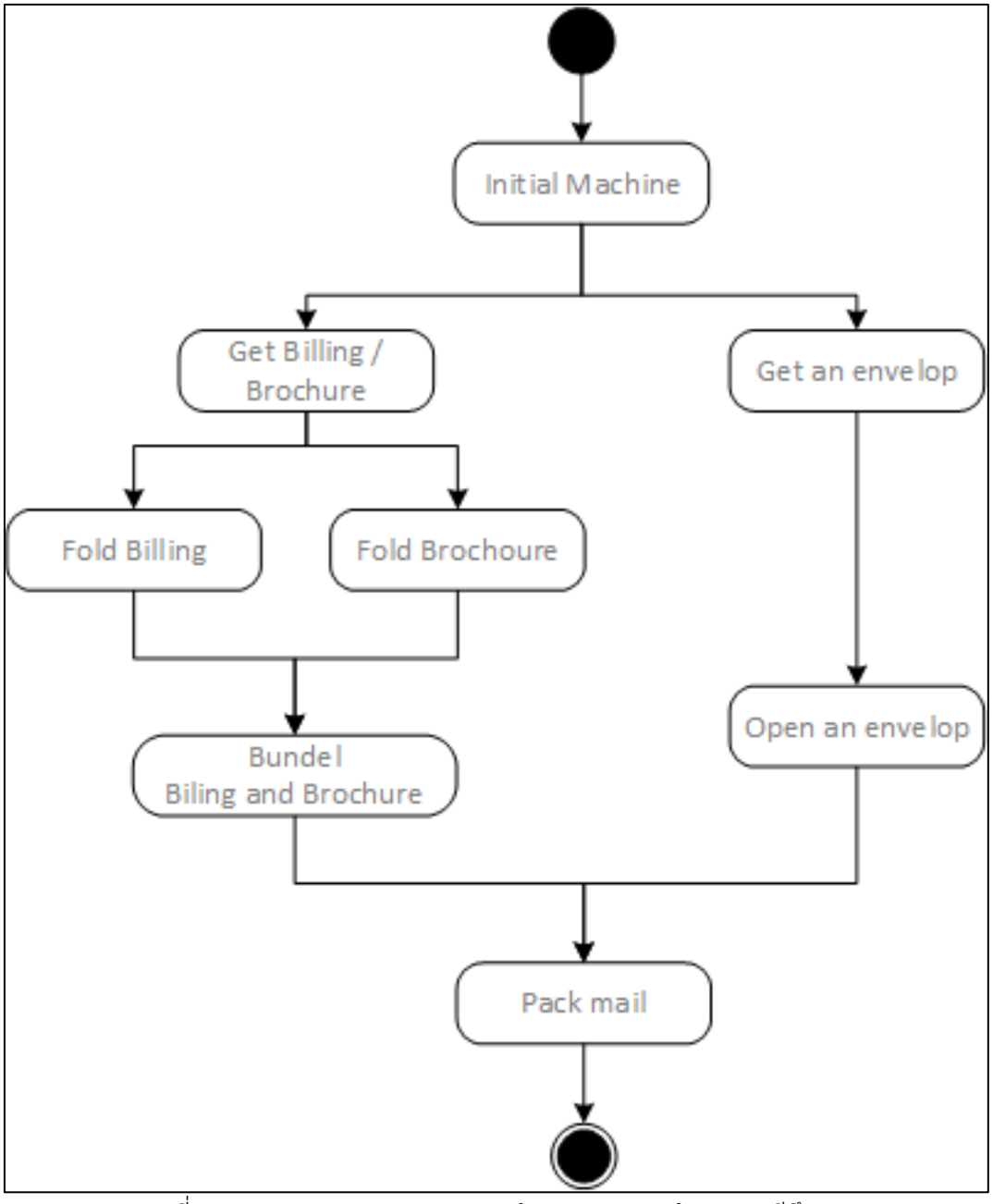
3) กรณีที่ผลลัพธ์จากการทวนสอบไม่สอดคล้องจะต้องกลับไปตรวจสอบความถูกต้องของเงื่อนไขการแปลงคุณลักษณะจากรูปแบบเอ็มทีแอลไปเป็นแอลทีแอลดังกล่าวว่าแปลงถูกต้องหรือไม่ ดังรูปที่ 3.1



รูปที่ 3.1 ลำดับการทำงานของงานวิจัยการแปลงคุณลักษณะและการทวนสอบ

3.2 กรณีศึกษาระบบแบบจำลองการบรรจุใบแจ้งยอดหนี้บัตรเครดิตและกระบวนการการทำงาน

จากการเลือกกรณีศึกษาด้านบน ในงานวิจัยนี้จะยกระบบแบบจำลองสำหรับกระบวนการบรรจุซองของใบแจ้งยอดหนี้บัตรเครดิตขึ้นมาเป็นกรณีศึกษา ซึ่งมีกระบวนการการทำงานดังรูปที่ 3.2



รูปที่ 3.2 แผนภาพกระบวนการการทำงานของแบบจำลองกรณีศึกษา

จากกระบวนการการทำงานดังกล่าวเริ่มจากจุดเริ่มต้นของกระบวนการซึ่งมี 2 สายพานในการทำงาน กระบวนการของสายพานด้านซ้ายของรูปที่ 3.2 เริ่มต้นจากการดึงใบแจ้งยอดหนี้และใบโปรโมชั่น เพื่อเตรียมพร้อมในกระบวนการ หลังจากนั้นจะแยกส่งใบแจ้งยอดหนี้และใบโปรโมชั่นดังกล่าว เข้าสายพานแยกเพื่อทำการพับใบแจ้งยอดหนี้และใบโปรโมชั่น เมื่อทำการพับทั้งสองสายพานเรียบร้อยแล้ว สายพานหลังจากการพับจะรวมกันอีกรอบเพื่อนำใบแจ้งยอดหนี้และใบโปรโมชั่นที่ผ่านการพับแล้วมารวมกัน เพื่อรอการบรรจุซอง ในขณะเดียวกันที่สายพานด้านซ้ายทำงาน สายพานด้านขวาก็จะทำการเตรียมซองและและทำการคลี่ซองเพื่อรอทำการบรรจุและปิดผนึก หลังจากที่ทำการรวมใบแจ้งยอดหนี้และใบโปรโมชั่นที่ทำการพับแล้ว เมื่อทั้งสองสายพานทำงานเรียบร้อยแล้ว ขั้นตอนต่อไปสายพานทั้งซ้ายและขวาจะรวมกันเพื่อเข้าสู่กระบวนการบรรจุเอกสารและปิดผนึกซองถือเป็นอันสิ้นสุดกระบวนการของการดำเนินงาน

บทที่ 4
กระบวนการออกแบบแบบจำลองเชิงรูปนัยสำหรับกรณีศึกษาและหลักการแปลงรูปแบบเอ็มทีแอลเป็นรูปแบบแอลทีแอล

จากภาพรวมของงานวิจัยและลำดับการทำงานของงานวิจัยที่ได้กล่าวไปในบทที่ 3 ในบทนนี้จะนำระบบที่ได้ออกแบบและวิเคราะห์แล้ว นำมาสร้างแบบจำลองเชิงรูปนัยที่แสดงลักษณะต่างๆ ของพฤติกรรมในกระบวนการทำงานโดยอาศัยหลักการของไทม์เพททริเนทโดยใช้ภาษาโพรเมลา และการทวนสอบการแปลงคุณลักษณะของตัวดำเนินการในรูปแบบเอ็มทีแอลเป็นรูปแบบแอลทีแอลดังกล่าวว่าสามารถทำการแปลงคุณลักษณะดังกล่าวได้ถูกต้องครบถ้วน

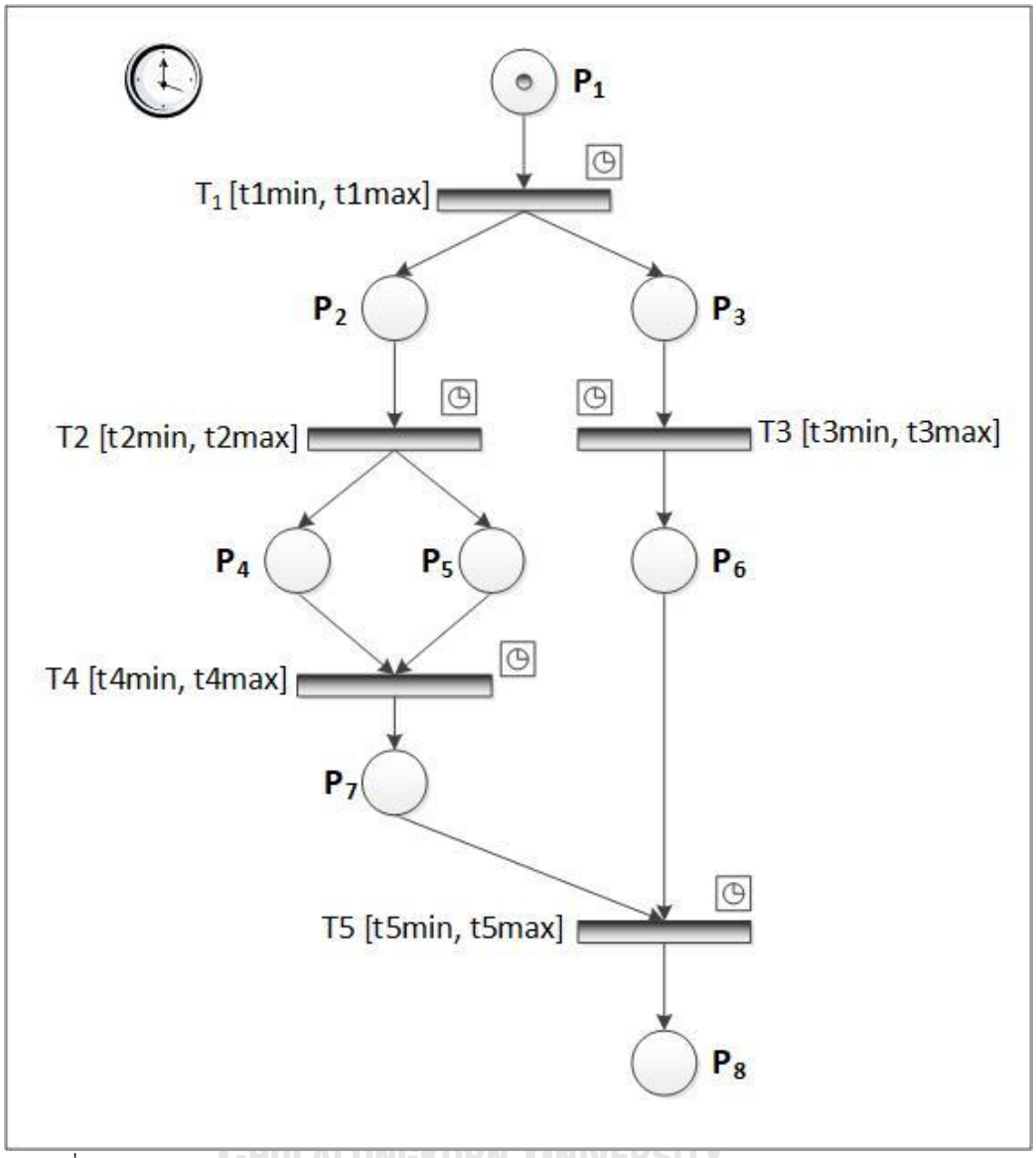
4.1. การออกแบบและวิเคราะห์กระบวนการการทำงานของแบบจำลองกรณีศึกษาในรูปแบบของไทม์เพททริเนท

4.1.1. แผนภาพกระบวนการทำงานแบบจำลองและข้อกำหนดความต้องการกรณีศึกษา

หลังจากที่วิเคราะห์และได้ลำดับการทำงานของแบบจำลองกรณีศึกษาจากแผนภาพดังรูปที่ 3.2 เรียบร้อยแล้วในขั้นตอนต่อไปเป็นกระบวนการการวิเคราะห์และออกแบบแบบจำลองเชิงรูปนัยสำหรับกรณีศึกษา โดยอาศัยหลักการของไทม์เพททริเนท เพื่อกำหนดความต้องการของระบบและใช้ในการพัฒนาแบบจำลองด้วยภาษาโพรเมลาต่อไป สามารถออกแบบโมเดลกระบวนการการทำงานแบบจำดังกล่าวได้

จากรูปที่ 4.1 กำหนดให้ P แทนแต่ละเพลส มีทั้งหมด 8 เพลส ตั้งแต่ P_1 – P_8 สามารถแทนด้วยเหตุการณ์ดังต่อไปนี้

1) P_1 แทนด้วยจุดเริ่มต้นของกระบวนการของกรณีศึกษาการบรรจุซอง
2) P_2 แทนด้วยเหตุการณ์ของการดึงใบแจ้งยอดหนี้และใบแนบโฆษณา
3) P_3 แทนด้วยเหตุการณ์การเลือกประเภทของซองใบแจ้งยอดหนี้บัตรเครดิต
4) P_4 แทนด้วยเหตุการณ์การพับใบแจ้งยอดหนี้บัตรเครดิต
5) P_5 แทนด้วยเหตุการณ์การพับใบแนบโฆษณา
6) P_6 แทนด้วยเหตุการณ์การเปิดซองอัตโนมัติเพื่อรองรับการบรรจุซอง
7) P_7 แทนด้วยเหตุการณ์การรวมใบแจ้งยอดบัตรเครดิตและใบแนบโฆษณาอยู่ในถาดพร้อมบรรจุซองในถาด
8) P8 แทนด้วยเหตุการณ์บรรจุใบแจ้งยอดบัตรเครดิตและใบแนบโฆษณาลงซองและทำการปิดผนึกซอง



รูปที่ 4.1 กระบวนการการทำงานของแบบจำลองกรณีศึกษาในรูปแบบของไทม์เพททริเนท

ซึ่งแบบจำลองดังกล่าวถูกควบคุมเวลาของทั้งกระบวนการด้วยนาฬิกาหลักของแบบจำลองขั้นตอนการบรรจุซองทั้งหมด และนาฬิกาย่อยๆ สำหรับแต่ละทรานซิชันที่แทนด้วย T1 – T5 ทรานซิชันละ 1 เรือน รวมทั้งสิ้น 5 เรือน เพื่อควบคุมเวลาของกระบวนการย่อยในแต่ละทรานซิชัน โดยแต่ละทรานซิชันจะถูกกำหนดเวลาเริ่มต้นและสิ้นสุดไว้ที่ [t(min), t(max)] ตามลำดับ และเป็นเวลาที่ถูกกำหนดค่าไว้แล้ว ซึ่งงานวิจัยนี้จะทำการพัฒนาตัวนับเวลา เพื่อนับเวลาของกระบวนการดังกล่าวด้วย

การทำงานของตัวนับเวลาในแบบจำลองการบรรจุซองใบแจ้งยอดหนี้บัตรเครดิตดังกล่าวจะเริ่มต้นการทำงานที่วินาทีที่เริ่มจาก 0 และเพิ่มขึ้นเรื่อยๆ โดยใช้นาฬิกาหลักของแบบจำลองเป็นตัวควบคุมเวลาของทั้งกระบวนการ และเมื่อกระบวนการแต่ละกระบวนการย่อยเริ่มต้น นาฬิกาของแต่ละทรานซิชัน จะเริ่มจับเวลาตั้งแต่มีการร้องขอการทำงานจนถึงสิ้นสุดการทำงาน ซึ่งมีเวลาการ

กำหนดเวลาเริ่มต้นกระบวนการและสิ้นสุดกระบวนการที่ชัดเจน จนสิ้นสุดกระบวนการทั้งหมด โดยไม่มีการใส่หน่วงเวลา

4.1.2. แผนภาพแบบจำลองสัญญาณนาฬิกาหลัก

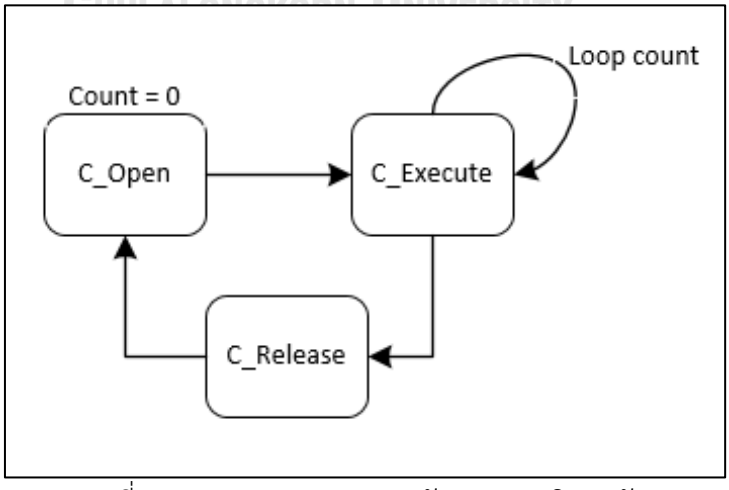
การออกแบบสัญญาณนาฬิกาหลักของกระบวนการกรณีศึกษาเพื่อควบคุมการทำงานทั้งหมดเป็นตัวนับเวลาของการทำงานตั้งแต่เริ่มกระบวนการไปจนสิ้นสุดกระบวนการการทำงาน สามารถกำหนดการนับเวลาได้เป็น 1 หน่วยเวลาในการเพิ่มการนับเวลาในแต่ละครั้ง โดยได้กำหนดค่าสูงสุดที่เป็นไปได้ในหน่วยเวลาเพื่อควบคุมนาฬิกาหลัก ซึ่งเมื่อนาฬิกานับได้ถึงค่าสูงสุดแล้วจะทำการกำหนดค่าเริ่มต้นใหม่ให้เท่ากับ 1 เพื่อเริ่มนับนาฬิการอบใหม่ และเพื่อแสดงสถานะของสัญญาณนาฬิการ ผู้วิจัยจะจำแนกสถานะของสัญญาณนาฬิกา โดยแบ่งออกเป็น 3 สถานะ ดังนี้

1) สถานะ C_Open คือสถานะเริ่มต้นของสัญญาณนาฬิกาและเป็นสถานะแสดงถึงว่านาฬิกาพร้อมเริ่มการนับเวลาเมื่อมีการยิงโทเคน หรือเริ่มต้นของกระบวนการทำงานของแบบจำลองกรณีศึกษา

2) สถานะ C_Excecute คือสถานะที่เริ่มนับเวลาของสัญญาณนาฬิกาซึ่ง เป็นการนับเวลาของกระบวนการการทำงานของแบบจำลองจากกรณีศึกษาที่ถูกสร้างขึ้น เริ่มตั้งแต่ต้นกระบวนการจนถึงสิ้นสุดกระบวนการของการทำงานของแบบจำลองกรณีศึกษา

3) สถานะ C_Release คือสถานะที่แสดงถึงสิ้นสุดกระบวนการ ซึ่งจะเป็นสถานะสุดท้ายของสัญญาณนาฬิกา และจะวนรอบสถานะเพื่อกำหนดค่าให้เป็นสถานะ C_Open และกำหนดค่าตั้งต้นของเวลาใหม่ให้เท่ากับ 0 เพื่อเตรียมตัวสำหรับการทำงานรอบถัดไปของกระบวนการทำงานของแบบจำลองกรณีศึกษา

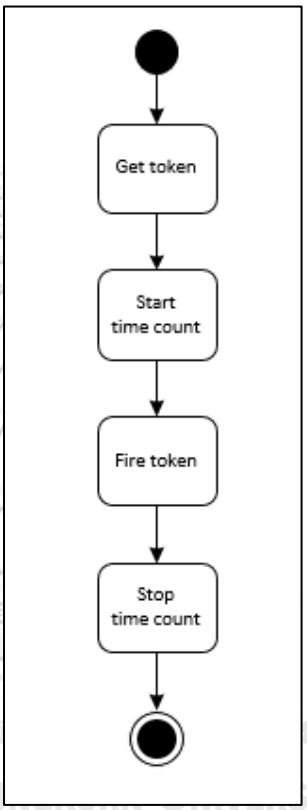
สามารถแสดงแผนภาพวงจรสถานะของแบบจำลองสัญญาณนาฬิกาหลักได้ดังรูปที่ 4.2



รูปที่ 4.2 แผนภาพสถานะของสัญญาณนาฬิกาหลัก

4.1.3. แผนภาพแบบจำลองสัญญาณนาฬิกาย่อย

วัตถุประสงค์ของการสร้างแบบจำลองสัญญาณนาฬิกาย่อย เพื่อเป็นตัวนับเวลาในแต่ละกระบวนการย่อย โดยจะทำการนับเวลาดังกล่าวเมื่อโทเคนเข้าสู่ทรานซิชันและหยุดนับเวลาเมื่อโทเคนออกจากทรานซิชัน ไปยังเพลส และในสัญญาณนาฬิกาย่อยในแต่ละทรานซิชันก็จะเป็นอิสระต่อกัน รูปแบบการทำงานของแบบจำลองสัญญาณนาฬิกาย่อยสามารถแสดงได้ตามแผนภาพกิจกรรมดังรูปที่ 4.3



รูปที่ 4.3 แผนภาพกิจกรรมการทำงานของสัญญาณนาฬิกาย่อย

4.2. การพัฒนาแบบจำลองจากกรณีศึกษาด้วยภาษาโพรเมลา

4.2.1. แบบจำลองสัญญาณนาฬิกาหลัก

แบบจำลองสัญญาณนาฬิกาหลักด้วยภาษาโพรเมลาสามารถแสดงดังรูปที่ 4.4

```
1   mtype = {C_Open,  C_Execute, C_Release }
2   byte GlobalClockState = C_Open;
3   int mainCl = 1;
4   int mainClTimeout = 60;
5
6   active proctype mainClockState(){
7
8   printf("clockState - %e\n",GlobalClockState);
9   do
10
11          ::atomic{((GlobalClockState == C_Open) && !timeout) -> GlobalClockState=C_Execute;}
12          ::atomic{((GlobalClockState == C_Release) && !timeout) -> GlobalClockState=C_Open;}
13          ::atomic{((GlobalClockState == C_Execute) && !timeout) ->
14                  mainCl++ ->
15                  //mainClass();
16                  GlobalClockState=C_Release ;
17
18                  if:: (mainCl > mainClTimeout) -> mainCl = 0;
19                  :: else;
20                  fi;
21
22                  printf("fn - clockState - %e\n",GlobalClockState);
23          }
24  od;
25  }
```

รูปที่ 4.4 แบบจำลองสัญญาณนาฬิกาหลักด้วยภาษาโพรเมลา

จากรูปที่ 4.4 จากบรรทัดที่ 1 ซึ่งกำหนดค่าของแต่ละสถานะของนาฬิกาหลักไว้เป็น 3 สถานะ ดังนี้ C_Open, C_Execute, C_Release โดยที่สถานะ C_Open คือสถานะเริ่มต้นของนาฬิกาหลัก และเมื่อนาฬิกาหลักอยู่ในสถานนะเริ่มต้น และพร้อมใช้งานสถานะของนาฬิกาหลักจะเปลี่ยนเป็น C_Execute เมื่อจบการทำงานใน 1 รอบการทำงานของกระบวนการทำงาน ระบบนาฬิกาหลักก็จะทำการเปลี่ยนสถานะนาฬิกาเป็น C_Release เพื่อปล่อยให้สถานะของนาฬิกาหลักพร้อมที่จะกลับไปทำงานในค่าเริ่มต้นอีกครั้งในสถานะ C_Open นอกจากนี้ผู้วิจัยได้นำคำสั่ง timeout เข้ามาใช้เพื่อรองรับการทำงานของนาฬิกาหลักไม่ให้เปลี่ยนสถานะและเกิดการ timeouot ก่อนที่ การถือครองทรัพยากรกระบวนการการทำงานในระบบสิ้นสุดลง

4.2.2. แบบจำลองสัญญาณนาฬิกาย่อย

แบบจำลองสัญญาณนาฬิกาย่อยด้วยภาษาโพรเมลาสามารถแสดงดังรูปที่ 4.5 สัญญาณนาฬิกาย่อยที่รับพารามิเตอร์เป็นค่า T เพื่อตรวจสอบตำแหน่งของทรานซิชั่น tmin คือเวลาที่สามารถเริ่มยิงโทเคนได้ และ tmax คือ เวลามากที่สุดที่สามารถยิงโทเคนได้ สำหรับในงานวิจัยนี้ค่า tmin และค่าของ tmax จะมีค่าเท่ากันเสมอ หลักการการทำงานของนาฬิกาย่อย จะเริ่มนับเวลาด้วยตัวแปรที่ชื่อว่า subCl และนาฬิกาหลักที่ชื่อว่า mainCl ไปพร้อมๆ กัน ทำให้ในแบบจำลองนาฬิกาย่อยนี้จึงปรากฏการเรียกสัญญาณนาฬิกาหลักด้วย ตามบรรทัดที่ 5 แต่เนื่องจาก T3 ซึ่งเป็นทรานซิชันที่เป็น

การทำงานคู่ขนาน กับอีกด้านทำให้ไม่จำเป็นต้องนับสัญญาณนาฬิกาหลักในทรานซิชั่นดังกล่าว ซึ่งข้อจำกัดของการยิงโทเคนนี้ระบบจะต้องยิงโทเคนออกไป ณ เวลาที่ tmin เสมอ

```
1       inline subTimer(T, tmin, tmax) {
2               int i;
3               subCl = 0;
4               for( i : 1 ..  (tmin)) {
5                       if ::(T != T3) -> mainTimer();
6                         ::else
7                       fi;
8
9                       if::(subCl > tmax) -> skip;
10                       ::else ->subCl++;
11                      fi;
12              }
13              printf("Fire at : %u:%d \n",mainCl, subCl);
14              subCl = 0;
15      }
```

รูปที่ 4.5 แบบจำลองสัญญาณนาฬิกาย่อยด้วยภาษาโพรเมลา

4.2.3. แบบจำลองการทำงานของกรณีศึกษาจากภาษาโพรเมลา

จากรูปที่ 4.6 แสดงการกำหนดค่าตัวแปรที่ใช้ในการสร้างแบบจำลองการทำงานของกรณีศึกษา ซึ่งได้ทำการกำหนดค่าตัวแปรสำหรับสถานะของนาฬิกา และค่าเริ่มต้นสำหรับสถานะของนาฬิกา โดยค่าที่เป็นไปได้คือ C_Open, C_Execute และ C_Release ตามบรรทัดที่ 1 และ 10 ลำดับต่อมา เป็นการกำหนดค่าตัวแปรสำหรับควบคุมลำดับการทำงานภายใต้ตัวแปรที่ชื่อว่า react1 – react5 โดยที่ react1 เป็นตัวแปรในการควบคุมการยิงโทเคนออกจากจุดเริ่มต้นของสายพาน ไปยังการดึงใบแจ้งยอดหนี้และเลือกประเภทของซองแจ้งยอดหนี้ ส่วนตัวแปร react2 เป็นการขั้นตอนการพับกระดาษเพื่อเตรียมรอทำการบรรจุซอง หลังจากที่ทำการพับเรียบร้อยแล้ว ตัวแปรที่ทำการรวมสายพานย่อยดังกล่าวจะถูกกำหนดขึ้นด้วยตัวแปร react3 เพื่อรอกระบวนการบรรจุซอง หลังจากที่เลือกประเภทซองจดหมายเพื่อเตรียมบรรจุแล้ว ตัวแปร react4 จะแทนด้วยการเปิดซองเพื่อเตรียมนำใบแจ้งยอดบรรจุ และปิดผนึกซองด้วยตัวแปร react5 ตามตัวแปรบรรทัดที่ 2 จากตัวแปรตั้งแต่ บรรทัดที่ 13 – บรรทัดที่ 20 แสดงจำนวนของโทเคนที่แสดงในแต่ละเพลส โดยโทเคนเริ่มต้นจะอยู่ที่จุดเริ่มต้นของกระบวนการ (P1) เมื่อโทเคนถูกยิงผ่านทรานซิชั่นใดๆ ซึ่งค่าตัวแปรเวลาที่ควบคุมในแต่ละทรานซิชั่นจะถูกกำหนดขึ้นด้วยค่าปิดที่เป็นค่าเวลาเริ่มต้นและเวลาที่สิ้นสุดด้วยตัวแปร t[x]min และ t[x]max ตามบรรทัดที่ 22 – บรรทัดที่ 35 ซึ่งเป็นค่าคงที่ที่ถูกกำหนดขึ้นมาแล้ว โดยที่ x แทนค่าตัวเลขตามเลขทรานซิชั่นนั้นๆ

```
1       mtype = {C_Open,  C_Execute, C_Release };
2       mtype = {react1, react2, react3, react4, react5} ;
3       //-react1 -> p1 > p2,p3
4       //-react2 -> p2 >p4,p5
5       //-react3 - > p4,p5 > p7
6       //-react4 -> p3 > p6
7       //-react5 -> p6, p7 > p8
8
9       byte GlobalClockState = C_Open;
10
11      /*define var*/
12      int p1Token = 1;
13      int p2Token = 0;
14      int p3Token = 0;
15      int p4Token = 0;
16      int p5Token = 0;
17      int p6Token = 0;
18      int p7Token = 0;
19      int p8Token = 0;
20
21      int T1 = 0;
22      int T2 = 0;
23      int T3 = 0;
24      int T4 = 0;
25      int T5 = 0;
26
27      int t1min = 1;
28      int t1max = 1;
29
30      int t2min = 2;
31      int t2max = 2;
32
33      int t3min = 2;
34      int t3max = 2;
35
36      int t4min = 3;
37      int t4max = 3;
38
39      int t5min = 4;
40      int t5max = 4;
41
42      int mainCl = 0;
43      int total = 0;
44      int outTime = 0;
45      int mainClTimeout = 60;
```

รูปที่ 4.6 ตัวแปรที่ถูกกำหนดขึ้นเพื่อสร้างแบบจำลองในกรณีศึกษา

จากรูปที่ 4.7 แสดงแบบจำลองของกรณีศึกษาที่ถูกพัฒนาขึ้นด้วยภาษาโพรเมลา โปรแกรมจะเริ่มต้นการทำงานจากพร็อกไทป์ที่ชื่อว่า mainClass(); พร็อกไทป์นี้ ทำการควบคุมการทำงานของสายพานโดยการตรวจสอบเงื่อนไขกับโทเคนที่เพลสนั้นๆ ตามบรรทัดที่ 39 46 53 60 67 และ 73 เมื่อตรวจสอบแล้วพบว่า ระบบสามารถทำงานได้ที่เพลสใดๆ ก็จะทำการนับเวลาด้วยการเรียกฟังก์ชั่น subTimer(T[x], t[x]min, t[x]max) ที่สร้างขึ้น โดยจะทำการตรวจสอบและทำงานพร้อมกันกับสัญญาณนาฬิกาหลักและข้อกำหนดด้านเวลาที่ถูกกำหนดขึ้นในแต่ละเหตุการณ์ ถูกประกาศด้วยเวลาปิดที่จุดเริ่มต้น และจุดสิ้นสุด แทนด้วย t[x]min และ t[x]max เพื่อตรวจสอบสถานะความพร้อมยิงโทเคน เมื่อโทเคนอยู่ในสถานะที่พร้อมยิง โปรแกรมจะทำการ เรียกพร็อกไทป์ ที่ชื่อว่า controlFlow(react[x]) เพื่อส่งค่าโทเคนให้แต่ละเหตุการณ์ในกระบวนการทำงานนั้นๆ ได้เริ่มการทำงาน ตั้งแต่จุดเริ่มต้นการทำงานไปจนถึงจุดสิ้นสุดการทำงาน

```
inline controlFlow(token){
	printf("Token : %u  - %e \n\n", token, token);
	do::
		if::(token == react1 && p1Token > 0) ->    //-react1 -> p1 > p2,p3
			p2Token = 1;
			p3Token = 1;
			p1Token = 0;
			break;
		::(token == react2 && p2Token > 0) ->  //-react2 -> p2 >p4,p5
			p4Token = 1;
			p5Token = 1;
			p2Token = 0;
			break;
		::(token == react4 && p4Token > 0 && p5Token > 0) ->   //-react4 -> p4,p5 > p7
			p7Token = 1;
			p4Token = 0;
			p5Token = 0;
			break;
		::(token == react3 && p3Token > 0) ->  //-react3 -> p3 > p6
			p6Token = 1;
			p3Token = 0;
			break;
		::(token == react5  && p6Token > 0 && p7Token >0) ->   //-react5 -> p6, p7 > p8
			p8Token = 1;
			p6Token = 0;
			p7Token = 0;
			break;
		::else->
			//p1Token = 1;
			//p8Token = 0;
			break;
		fi;
	od;
}
/*Model flow*/
proctype mainClass(){
	do
		/*#: p1>T1>p2 && p1>T1>p3*/
		::atomic{( p1Token >0) ->
			subTimer(T1, t1min) ->
			controlFlow(react1)  ->
			printf("T1mainCl : %u : %u : %u\n\n", mainCl, (t1min+total), (t1max+total));
		}
		/*#: p2>T2>p4 && p2>T2>p5 */
		::atomic{(p2Token > 0 )  ->
			subTimer(T2, t2min) ->
			controlFlow(react2) ->
			printf("T2mainCl : %u : %u : %u\n\n", mainCl, (t2min+total), (t2max+total));
		}
		/*#: p4>T4>p7 && p5>T4>p7*/
		::atomic{( p4Token > 0 && p5Token  >0) ->
			subTimer(T4, t4min) ->
			controlFlow(react4) ->
			printf("T4mainCl : %u : %u : %u\n\n", mainCl, (t4min+total), (t4max+total));
		}
		/*#:  p3>T3>p6*/
		::atomic{(p3Token > 0) ->
			subTimer(T3, t3min) ->
			controlFlow(react3)->
			printf("T3mainCl : %u : %u : %u\n\n", mainCl, (t3min+total), (t3max+total));
		}
		/*#: p6>T5>p8 && p7>T5>p8*/
		::atomic{(p6Token > 0 && p7Token  >0) ->
			subTimer(T5, t5min) ->
			controlFlow(react5)->
			printf("T5mainCl : %u : %u : %u\n\n", mainCl, (t5min+total), (t5max+total));
		}
		::(p8Token > 0) ->
			//p8Token = 0;
			//p1Token = 1;
			printf("--End--\n");
			break;
	od;
}
```

รูปที่ 4.7 แบบจำลองของกรณีศึกษาในภาษาโพรเมลา

พร๊อกไทป์ mainClass() ระบบจะทำการตรวจสอบความพร้อมของจุดเริ่มต้นของการะบวนการของกรณีศึกษาในการบรรจุซองว่ามีสถานะที่พร้อมที่จะเริ่มต้นการทำงาน ด้วย p1Token ว่าสามารถทำงานได้หรือไม่ เมื่อระบบอยู่ในสถานะที่พร้อมทำงาน ระบบนาฬิกาย่อย T1 ก็จะเริ่มต้นนับเวลาพร้อมกับนาฬิกาหลัก และตรวจสอบว่าโทเคนสามารถยิงออกไปได้หรือไม่ จากค่าตัวแปรเวลา [t1min, t1max] ซึ่งมีค่าเป็น [1, 2] ตามลำดับ เมื่อถึงเวลาที่สามารถให้โทเคนยิงออกไปได้ สัญญาณนาฬิกาย่อยก็จะยิงโทเคนออกไป แต่ถ้ายังไม่สามารถยิงออกไปได้สัญญาณนาฬิกาย่อยก็จะการวนลูปนับไปเรื่อยๆ จนกว่าจะสามารถยิงโทเคนออกไปได้ เมื่อระบบตรวจสอบแล้วว่าสามารถยิงโทเคนได้ จากนั้นจะทำการพร๊อกไทป์ controlFlow(react1) ซึ่งจะทำการยิงโทเคนให้กับเหตุการณ์ของการดึงซองของใบแจ้งยอดหนี้และใบแนบโฆษณา (P2) และต่อไปนี้จะเรียกการบวนการการทำงานในด้านนี้ว่า สายพานด้านซ้าย ก็เริ่มทำงานไปพร้อมกันกับเหตุการณ์การเลือกประเภทของซองบรรจุ (P3) และเรียกกระบวนการการทำงานในด้านนี้ว่า สายพานด้านขวา ซึ่งเป็นสายพานการทำงานที่คู่ขนานกัน และในจุดเริ่มต้นของสายพานก็จะอยู่ในสถานะที่ถูกพักไว้ เพื่อให้กระบวนการลำดับถัดไปทำงานได้ สายพานการทำงานในด้านซ้าย เมื่อดึงใบแจ้งยอดหนี้และใบแนบโฆษณาจากถาดได้แล้วนั้น ก็จะทำการตรวจสอบสถานะความพร้อมของการทำงานเช่นเดิม โดยที่จะทำการเปรียบเทียบกับเงื่อนไขของตัวแปรเวลา [t2min, t2max] ด้วยค่า [2, 3] ตามลำดับ จากนั้นระบบจะทำการพับใบแจ้งยอดหนี้ บัตรเครดิตและใบแนบโฆษณา ด้วยการเรียกพร๊อกไทป์ controlFlow(react2) ซึ่งในสายพานการทำงานทั้งสองเหตุการณ์จะต้องอยู่ภายใต้เงื่อนไขที่จะต้องให้เหตุการณ์ทั้งคู่ เสร็จสมบูรณ์เรียบร้อยจึงจะสามารถยิงโทเคนไปยังเหตุการณ์ที่ใบแจ้งยอดและใบแนบโฆษณาที่ถูกพับเรียบร้อยแล้ว ด้วย controlFlow(react3) มาอยู่ในถาดพร้อมกันทั้งคู่ ซึ่งในระหว่านี้นั้นสายพานด้านขวา สำหรับเหตุการณ์ที่ทำการเลือกประเภทซองบรรจุก็จะเริ่มต้นกระบวนการทำงานไปพร้อมกันกับสายพานด้านซ้าย ซึ่งตรวจสอบด้วยเงื่อนไขตัวแปรของเวลาที่ถูกกำหนดขึ้นด้วย [t3min, t3max] ด้วยค่า [2, 7] ดำเนินการด้วย controlFlow(react4) และเมื่อกระบวนการการเลือกประเภทของซองเรียบร้อยแล้ว จะทำให้เหตุการณ์การเปิดซองบรรจุเพื่อเตรียมนำใบแจ้งยอดและใบแนบโฆษณา เริ่มทำงาน ซึ่งในจังหวะนี้ เหตุการณ์การบรรจุใบแจ้งยอดหนี้และใบแนบโฆษณา จะเริ่มต้นการทำงานได้นั้นจะต้องรอให้เหตุการณ์รวมใบแจ้งยอดและใบแนบโฆษณาหลังจากที่พับแล้วมาอยู่ในถาด และเหตุการณ์ที่เปิดซองบรรจุนั้นจะต้องสิ้นสุดแล้วทั้งสองเหตุการณ์ จึงจะสามารถบรรจุใบแจ้งยอดหนี้และใบแนบโฆษณาลงในซองและทำการปิดผนึกได้ ผ่าน controlFlow(react5) เป็นอันเสร็จสิ้นกระบวนการการทำงานของกรณีศึกษาใน 1 รอบการทำงานและจะต้องตั้งค่าความพร้อมการทำงานให้กับการทำงานรอบใหม่ สำหรับการทำงานในรอบต่อๆ ไป

4.3. หลักการแปลงคุณลักษณะของรูปแบบเอ็มทีแอลแปลงไปเป็นรูปแบบแอลทีแอล

1) กำหนดรูปแบบของสมการของเอ็มทีแอลที่จะใช้

$$\Box_{(tmin,\ tmax)}\ \varphi$$

2) จากข้อ 1) ทำการตรวจสอบว่าตัวดำเนินการที่อยู่ในสมการเป็นตัวดำเนินการใด

$$\Box_{(tmin,\ tmax)}\ \varphi \Rrightarrow \Box$$

3) ทำการอ่านค่าของเวลาที่มีอยู่ในสมการ เพื่อเก็บใส่ตัวแปรของเวลาทั้งเวลาเริ่มต้นและเวลาสิ้นสุด

$$(time >= tmin) \wedge (time <= tmax)$$

4) ทำการแปลงเป็นรูปแบบแอลทีแอล โดยเชื่อมแต่ละค่าของช่วงเวลาและเหตุการณ์ด้วย $\wedge$

$$\Box(\ \varphi \wedge (time >= tmin) \wedge (time <= tmax))$$

4.4. การแปลงคุณลักษณะของรูปแบบเอ็มทีแอลแปลงไปเป็นรูปแบบแอลทีแอล

4.4.1. การแปลงเอ็มทีแอลเป็นแอลทีแอลในการสร้างตัวดำเนินการแบบตลอดไป (Always)

จากสมการตัวดำเนินการแบบตลอดไปในรูปแบบของเอ็มทีแอล คุณลักษณะ $\Box_{[t1,\ t2]}\ \varphi$ โดยที่ φ แทน เหตุการณ์ใดๆ และ t1, t2 แทนช่วงเวลาปิดของจุดเริ่มต้นและสิ้นสุดของเหตุการณ์ใดๆ ตามลำดับ ที่สามารถเกิดขึ้นได้ ซึ่งสำหรับตัวดำเนินการแบบตลอดไป การแปลงคุณลักษณะสามารถแปลงให้อยู่ในรูปของรูปแบบแอลทีแอล ได้ดังนี้

$\Box$[t1, t2] φ แปลเป็น $\Box(\varphi \wedge$ (time>=t1 $\wedge$ time<=t2) สมการที่ 4. 1

4.4.2. การแปลงเอ็มทีแอลเป็นแอลทีแอลในการสร้างตัวดำเนินการแบบในที่สุด (Eventually)

จากสมการตัวดำเนินการแบบในที่สุดในรูปแบบของเอ็มทีแอล คุณลักษณะ $\Diamond_{[t1,\ t2]}\ \varphi$ โดยที่ φ แทน เหตุการณ์ใดๆ และ t1, t2 แทนช่วงเวลาปิดของจุดเริ่มต้นและสิ้นสุดของเหตุการณ์ใดๆ ตามลำดับ ที่สามารถเกิดขึ้นได้ ซึ่งสำหรับตัวดำเนินการแบบในที่สุด การแปลงคุณลักษณะสามารถแปลงให้อยู่ในรูปของรูปแบบแอลทีแอล ได้ดังนี้

$\Diamond_{[t1,\ t2]}\ \varphi$ แปลเป็น $\Diamond(\varphi \wedge$ time>=t1 $\wedge$ time<=t2) สมการที่ 4. 2

4.4.3. การแปลงเอ็มทีแอลเป็นแอลทีแอลในการสร้างตัวดำเนินการแบบถัดไป (Next)

จากสมการตัวดำเนินการแบบถัดไปในรูปแบบของเอ็มทีแอล คุณลักษณะ $\bigcirc_{[t1, t2]} \varphi$ โดยที่ φ แทน เหตุการณ์ใดๆ และ t1, t2 แทนช่วงเวลาปิดของจุดเริ่มต้นและสิ้นสุดของเหตุการณ์ใดๆ ตามลำดับ ที่สามารถเกิดขึ้นได้ ซึ่งสำหรับตัวดำเนินการแบบถัดไป งานวิจัยนี้สนใจที่สถานะถัดไป (Next state) แบบระบบอะซิงโครไนซ์ (Asynchronous) ซึ่งนั่นคือจะไม่สนใจเวลาที่เป็นเวลาต่อเนื่องหรือว่าเวลาต่อจากเวลาเดิม การแปลงคุณลักษณะสามารถแปลงให้อยู่ในรูปของรูปแบบแอลทีแอล ได้ดังนี้

$\bigcirc_{[t1, t2]} \varphi$ แปลเป็น $\bigcirc (\varphi \wedge \text{time} >= t1 \wedge \text{time} <= t2)$ สมการที่ 4. 3

4.4.4. การแปลงเอ็มทีแอลเป็นแอลทีแอลในการสร้างตัวดำเนินการจนกระทั่งแบบเข้ม (Strong Until / Until)

จากสมการตัวดำเนินการจนกระทั่งแบบเข้ม ในรูปแบบของเอ็มทีแอล คุณลักษณะ $\varphi \cup_{[t1, t2]} \psi$ โดยที่ φ แทน เหตุการณ์ลำดับที่ 1 ψ แทนเหตุการณ์ลำดับที่ 2 และ t1, t2 แทนช่วงเวลาปิดของจุดเริ่มต้นและสิ้นสุดของเหตุการณ์ของทั้งสองเหตุการณ์ ตามลำดับ ที่สามารถเกิดขึ้นได้ ซึ่งสำหรับตัวดำเนินการจนกระทั่งแบบเข้ม การแปลงคุณลักษณะสามารถแปลงให้อยู่ในรูปของรูปแบบแอลทีแอล ได้ดังนี้

$\varphi \cup_{[t1, t2]} \psi$ แปลเป็น

$((\varphi \wedge (\text{time} >= t1 \wedge \text{time} <= t2)) \cup (\psi \wedge (\text{time} >= t1 \wedge \text{time} <= t2)))$ สมการที่ 4. 4

4.4.5. การแปลงเอ็มทีแอลเป็นแอลทีแอลในการสร้างตัวดำเนินการจนกระทั่งแบบอ่อน (Weak Until/ Unless)

จากสมการตัวดำเนินการจนกระทั่งแบบอ่อน ในรูปแบบของเอ็มทีแอล คุณลักษณะ $\varphi \omega_{[t1, t2]} \psi$ โดยที่ φ แทน เหตุการณ์ลำดับที่ 1 ψ แทนเหตุการณ์ลำดับที่ 2 และ t1, t2 แทนช่วงเวลาปิดของจุดเริ่มต้นและสิ้นสุดของเหตุการณ์ของทั้งสองเหตุการณ์ ตามลำดับ ที่สามารถเกิดขึ้นได้ ซึ่งสำหรับตัวดำเนินการจนกระทั่งแบบอ่อน การแปลงคุณลักษณะสามารถแปลงให้อยู่ในรูปของรูปแบบแอลทีแอล ได้ดังนี้

$\varphi \omega_{[t1, t2]} \psi$ แปลเป็น

$((\varphi \wedge (\text{time} >= t1 \wedge \text{time} <= t2)) \omega (\psi \wedge (\text{time} >= t1 \wedge \text{time} <= t2)))$ สมการที่ 4. 5

4.5. เงื่อนไขการทวนสอบสำหรับแบบจำลองกรณีศึกษา

จากคุณลักษณะเอ็มทีแอลสำหรับเหตุการณ์ที่ทำการทวนสอบแบบจำลองในกรณีศึกษาทั้งหมด 6 เหตุการณ์ เพื่อให้สอดคล้องกันกับการแปลงแอลทีแอลให้ให้อยู่ในรูปแบบของแอลทีแอลที่สามารถใช้ในการทวนสอบสำหรับแบบจำลองได้ ซึ่งประกอบไปด้วย

4.5.1. เงื่อนไขการทวนสอบที่ 1

การทวนสอบด้วยเงื่อนไขที่ว่า ระบบไม่สามารถบรรจุใบแจ้งยอดหนี้บัตรเครดิตและใบแนบโฆษณาและปิดผนึกซองได้ตั้งแต่เวลา 1 ถึง 9 เสมอ

4.5.2. เงื่อนไขการทวนสอบที่ 2

การทวนสอบด้วยเงื่อนไขที่ว่า ใน 1 รอบของการทำงานระบบจะดึงใบแจ้งยอดหนี้บัตรเครดิตและใบแนบโฆษณา 1 ครั้งในช่วงเวลาที่ 1 – 2

4.5.3. เงื่อนไขการทวนสอบที่ 3

การทวนสอบด้วยเงื่อนไขที่ว่า เมื่อมีการเปิดซองใบแจ้งยอดหนี้เพื่อรอการบรรจุและใบแจ้งยอดหนี้พร้อมกับใบแนบโฆษณาอยู่ในถาดเรียบร้อยแล้ว ขั้นตอนต่อไประบบสามารถที่จะทำการบรรจุและปิดผนึกซองได้

4.5.4. เงื่อนไขการทวนสอบที่ 4

การทวนสอบด้วยเงื่อนไขที่ว่า ระบบสามารถเลือกประเภทของซองบรรจุใบแจ้งยอดหนี้ พร้อมกันกับการดึงใบแจ้งยอดหนี้บัตรเครดิตและใบแนบโฆษณาได้

4.5.5. เงื่อนไขการทวนสอบที่ 5

การทวนสอบด้วยเงื่อนไขที่ว่า จะไม่เกิดเหตุการณ์การดึงใบแจ้งยอดหนี้บัตรเครดิตและใบแนบโฆษณา หลังจากระบบเริ่มการพับใบแจ้งยอดหนี้และใบแนบโฆษณาได้อีก

4.5.6. เงื่อนไขการทวนสอบที่ 6

การทวนสอบด้วยเงื่อนไขขัดแย้งไม่เป็นไปตามกระบวนการของแบบจำลอง ด้วยเหตุการณ์การพับใบแจ้งยอดหนี้บัตรเครดิตกำลังทำงานอยู่ และเหตุการณ์รวมใบแจ้งยอดบัตรเครดติดและใบแนบโฆษณาจะเริ่มต้นการทำงานในเงื่อนไขเวลาเดียวกันในเวลาเริ่มต้นเท่ากับ 3 และเวลาสิ้นสุดเท่ากับ 4

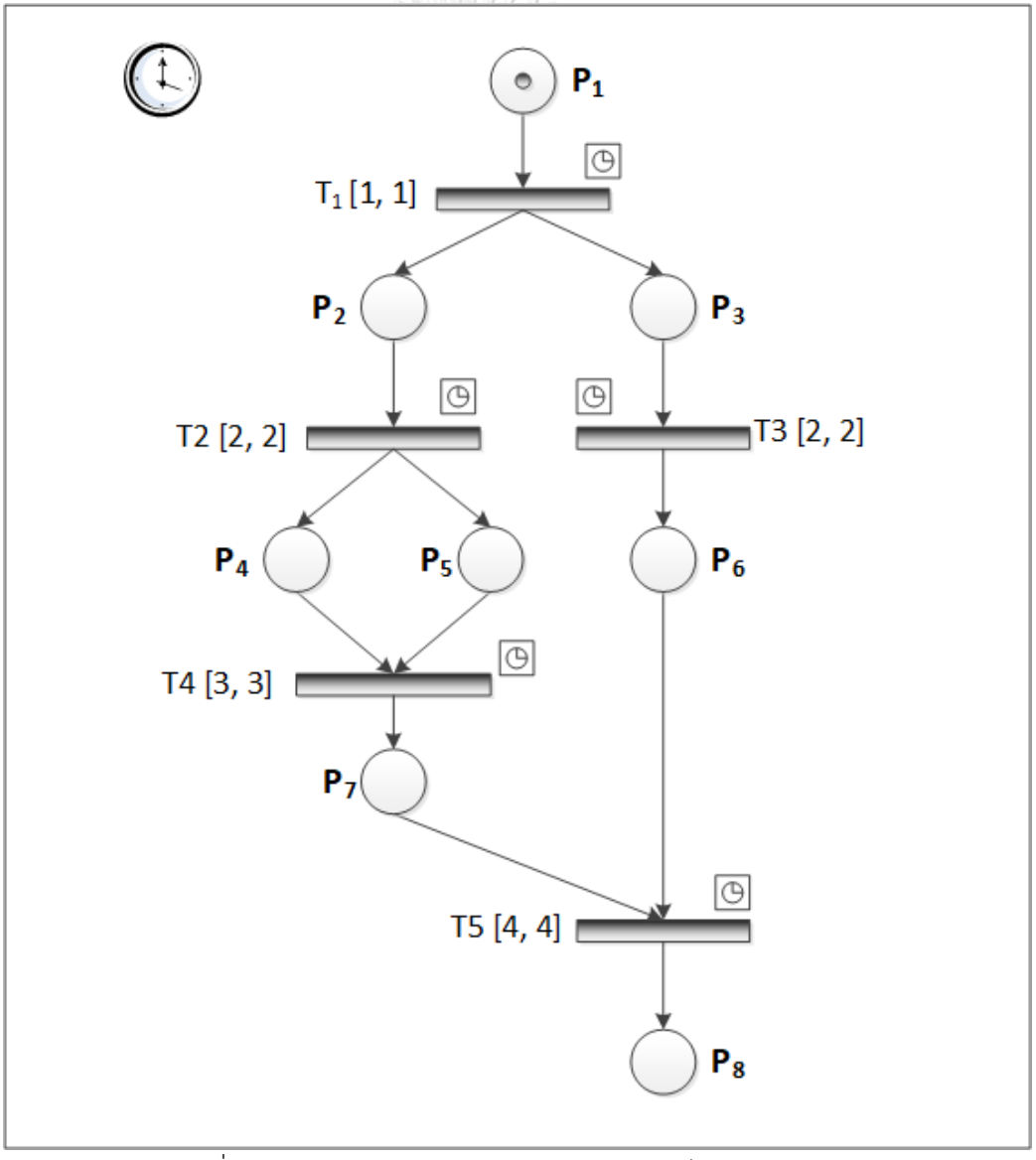
4.5.7. เงื่อนไขการทวนสอบที่ 7

การทวนสอบด้วยเงื่อนไขขัดแย้งของกรณีศึกษาที่ 1 คือระบบสามารถบรรจุใบแจ้งยอดหนี้บัตรเครดิตและใบแนบโฆษณาและปิดผนึกซองได้ตั้งแต่เวลา 1 ถึง 9

บทที่ 5
การทวนสอบแบบจำลองกรณีศึกษา

ในบทนี้จะกล่าวถึงการทวนสอบแบบจำลองกรณีศึกษาเพื่อพิสูจน์หลักการแปลงที่ถูกสร้างขึ้นว่าสามารถได้ทำการแปลงค่าจากเอ็มทีแอลเป็นแอลทีแอลนั้นสามารถทวนสอบแบบจำลองกรณีศึกษาได้ถูกต้อง และนอกจากนี้เพื่อแสดงให้เห็นว่าระบบจะทำงานได้ในกรอบเวลาที่กำหนด จึงแสดงการทวนสอบแบบจำลองกรณีศึกษาด้วยเงื่อนไขขัดแย้งกับแบบจำลองกรณีศึกษาด้วย

แบบจำลองของกรณีศึกษาในรูปแบบของไทม์เพททริเนทที่อ้างอิงจากรูปที่ 4.1

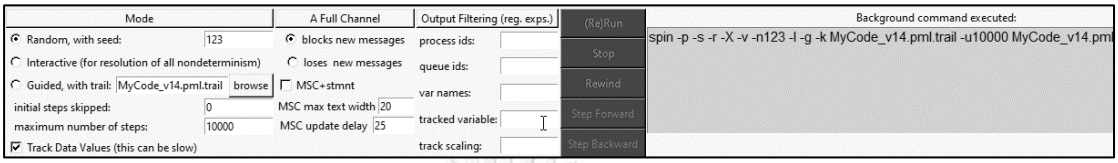


รูปที่ 5.1 แบบจำลองกรณีศึกษาในรูปแบบของไทม์เพททริเนท

จากรูปที่ 5.1 แสดงแบบจำลองกรณีศึกษาในรูปแบบของไทม์เพทริเน็ทซึ่งได้ทำการกำหนดตัวแปรเวลาไว้เรียบร้อยแล้ว สำหรับการทวนสอบแบบจำลองจะอ้างอิงจากรูปดังกล่าว

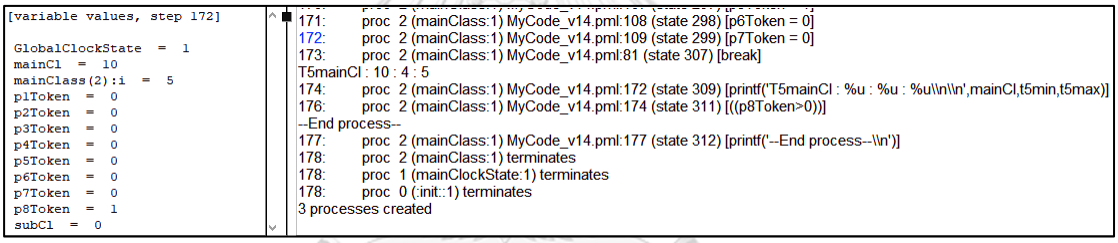
5.1. การทดสอบแบบจำลองกรณีศึกษา

การทดสอบแบบจำลอง ตั้งค่าที่ระดับ 10,000 ระดับ และ Random Seed ที่ 123 ดังรูปที่ 5.2



รูปที่ 5.2 หน้าจอการตั้งค่าทดสอบแบบจำลองที่ระดับ 10000 ระดับ

และพบว่าแบบจำลองสามารถทำงานได้จนครบ 1 รอบ โดยใช้ระดับทั้งหมด 172 ระดับ และโทเคนสุดท้ายจะอยู่ที่เพลสสุดท้ายนั่นคือ P8 ตามผลที่แสดง p8Token = 1 ดังรูปที่ 5.3

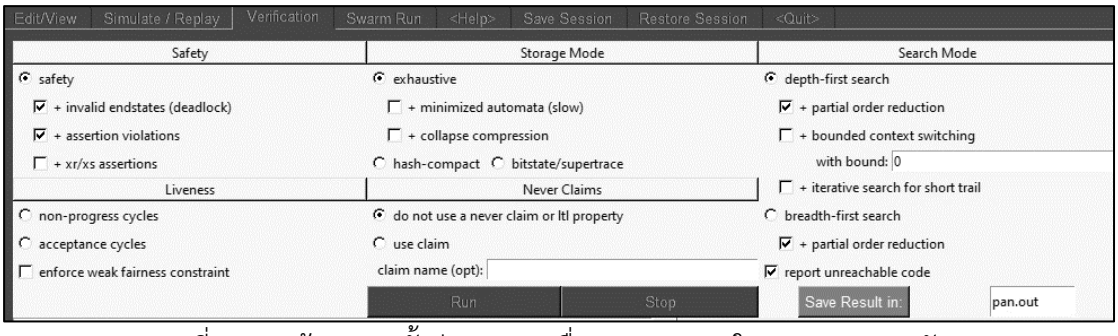


รูปที่ 5.3 หน้าจอผลการทดสอบที่ระดับ 10000

5.2. การทวนสอบแบบจำลองกรณีศึกษาในรูปแบบปลอดภัย (Safety)

1) วัตถุประสงค์การทวนสอบในรูปแบบปลอดภัย

การทวนสอบแบบจำลองกรณีศึกษาในรูปแบบปลอดดภัยในสถานะของการทำงานใน 1 รอบการทำงาน สามารถแสดงสถานะที่เข้าถึงแต่ละฟังก์ชั่นและทำงานได้ในทุกสถานะ โดยทำการตั้งค่าเพื่อทวนสอบความสามารถของแบบจำลองว่าจะสามารถทำงานได้ตั้งแต่ต้นจนจบโดยไม่มีการหยุดการทำงานระหว่างทาง ดังรูปที่ 5.4



รูปที่ 5.4 หน้าจอการตั้งค่าทดสอบเพื่อการทวนสอบในรูปแบบปลอดภัย

2) ผลที่คาดว่าจะได้รับ

แบบจำลองสามารถเข้าถึงการทำงานของแต่ละฟังก์ชั่นได้ และสามารถทำงานได้ในทุกๆ สถานะ โดยไม่เกิดการหยุดระหว่างการทำงาน

3) ผลการทวนสอบ

```
spin -a  MyCode_v14.pml
C:/cygwin64/bin/gcc.exe -DMEMLIM=1024 -O2 -DXUSAFE -DSAFETY -DNOCLAIM -w -o pan pan.c
./pan -m10000  -c1
Pid: 12144

(Spin Version 6.4.3 -- 16 December 2014)
        + Partial Order Reduction

Full statespace search for:
        never claim         - (not selected)
        assertion violations +
        cycle checks        - (disabled by -DSAFETY)
        invalid end states +

State-vector 136 byte, depth reached 100, errors: 0
     17 states, stored
      2 states, matched
     19 transitions (= stored+matched)
    150 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
   0.003   equivalent memory usage for states (stored*(State-vector + overhead))
   0.252   actual memory usage for states
 128.000 memory used for hash table (-w24)
   0.534   memory used for DFS stack (-m10000)
 128.730 total actual memory usage


unreached in proctype mainClass
        MyCode_v14.pml:66, state 14, "(1)"
        MyCode_v14.pml:91, state 36, "p5Token = 1"
        MyCode_v14.pml:92, state 37, "p2Token = 0"
        MyCode_v14.pml:97, state 41, "p4Token = 0"
        MyCode_v14.pml:98, state 42, "p5Token = 0"
        MyCode_v14.pml:103, state 46, "p3Token = 0"
        MyCode_v14.pml:108, state 50, "p6Token = 0"
        MyCode_v14.pml:109, state 51, "p7Token = 0"
        .....
        (42 of 317 states)
unreached in proctype mainClockState
        (0 of 21 states)
unreached in init
        (0 of 3 states)

pan: elapsed time 0.003 seconds
No errors found -- did you verify all claims?
```

รูปที่ 5.5 หน้าจอแสดงผลการทวนสอบในรูปแบบปลอดภัย

จากรูปที่ 5.5 แสดงผลการทวนสอบในรูปแบบที่ปลอดภัยซึ่งสามารถทำงานได้ทั้งหมดในสถานะที่ปลอดภัยทั้งหมด 78 โดยไม่มีข้อผิดพลาด สำหรับคำแจ้งเตือนจากผลการทวนสอบแบบจำลองที่ปรากฏเป็น unreached in proctype mainClass เกิดจากวนลูปโดยใช้ do... while และมีเงื่อนไข if...else และการ break สำหรับควบคุมโทนเคน ซึ่งในการวนลูป do...while 1 รอบภายใต้การใช้ if...else ทำให้ spin ไม่สามารถเข้าถึง state ในบาง state ได้ จึงเกิดการแจ้งเตือนดังกล่าวขึ้น สำหรับเวลาในการทำงาน 1 รอบทั้งหมด 0.003 วินาที ซึ่งไม่มีข้อผิดพลาดตามข้อมูล No errors found

4) สรุปผล

จากการตั้งค่าเพื่อทวนสอบนรูปแบบปลอดภัยดังรูปที่ 5.4 และจากผลทวนสอบแบบจำลองในรูปแบบปลอดภัยสามารถทำงานได้ทุกสถานะโดยไม่มีการหยุดการทำงาน และใน 1 รอบการทำงานสามารถทำงานอย่างปลอดภัยได้ 100 สถานะดังรูปที่ 5.5

5.3. การทวนสอบแบบจำลองในรูปแบบแอลทีแอล สำหรับเงื่อนไขการทวนสอบที่ 1

1) เงื่อนไขการทวนสอบ

การทวนสอบด้วยเงื่อนไขที่ว่า ระบบไม่สามารถบรรจุใบแจ้งยอดหนี้บัตรเครดิตและใบแนบโฆษณาและปิดผนึกซองได้ตั้งแต่เวลา 1 ถึง 9 เสมอ

ด้วยเงื่อนไขการทวนสอบและจากสมการที่ 4. 1 สามารถเขียนเป็นสมการในรูปของเอ็มทีแอลได้ดังนี้

$$\square_{[1,\,9]}\,(\neg P)$$

โดยที่ P แทนด้วย เหตุการณ์การบรรจุใบแจ้งยอดหนี้และปิดผนึกซอง

นำไปแปลงให้อยู่ในรูปแบบของแอลทีแอลที่ใช้ในการทวนสอบได้ดังรูปที่ 5.6 ชื่อ pe0

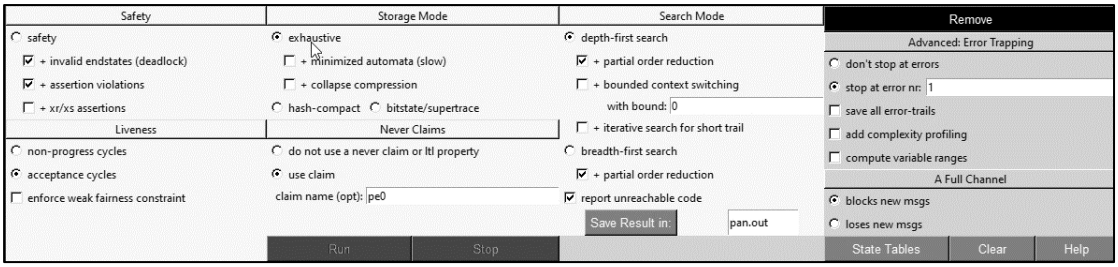
```
1   #define MTL_always ((!(mainCl >= 1&&mainCl <= 9)) || (p8Token == 0 && (mainCl >= 1&&mainCl <= 9)))
2   ltl pe0 { [] MTL_always}
```

รูปที่ 5.6 เงื่อนไขการทวนสอบแบบจำลองในรูปแบบแอลทีแอลในกรณีศึกษาที่ 1

2) ผลที่คาดว่าจะได้รับ

เหตุการณ์ของการบรรจุใบแจ้งยอดหนี้บัตรเครดิตและใบแนบโฆษณาและปิดผนึกซองไม่สามารถปิดผนึกซองได้ในช่วงเวลาที่ 1 – 11

3) ผลการทวนสอบ



รูปที่ 5.7 หน้าจอการตั้งค่าทวนสอบแบบจำลองโดยใช้รูปแบบแอลทีแอล pe0

จากรูปที่ 5.7 แสดงหน้าจอการทวนสอบแบบจำลองโดยใช้รูปแบบแอลทีแอลที่ชื่อว่า pe0 ที่แถบ Never Claims เลือกที่เมนู use claim แล้วใส่ชื่อรูปแบบแอลทีแอลที่ต้องการใช้ในการทวนสอบเป็นค่า pe0

```
spin -a  MyCode_v14.pml
ltl pe0: [] ((! (((mainCl>=1)) && ((mainCl<=9)))) || (((p8Token==0)) && (((mainCl>=1)) && ((mainCl<=9)))))
C:/cygwin64/bin/gcc.exe -DMEMLIM=1024 -O2 -DXUSAFE -w -o pan pan.c
./pan -m10000  -a -c1 -N pe0
Pid: 1368
warning: only one claim defined, -N ignored

(Spin Version 6.4.3 -- 16 December 2014)
        + Partial Order Reduction

Full statespace search for:
        never claim         + (pe0)
        assertion violations + (if within scope of claim)
        acceptance   cycles  + (fairness disabled)
        invalid end states - (disabled by never claim)

State-vector 152 byte, depth reached 115, errors: 0
     17 states, stored
      3 states, matched
     20 transitions (= stored+matched)
    150 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
   0.003   equivalent memory usage for states (stored*(State-vector + overhead))
   0.251   actual memory usage for states
 128.000 memory used for hash table (-w24)
   0.534   memory used for DFS stack (-m10000)
 128.730 total actual memory usage


unreached in proctype mainClass
        MyCode_v14.pml:66, state 14, "(1)"
        MyCode_v14.pml:91, state 36, "p5Token = 1"
        MyCode_v14.pml:92, state 37, "p2Token = 0"
        MyCode_v14.pml:97, state 41, "p4Token = 0"
        MyCode_v14.pml:98, state 42, "p5Token = 0"
        MyCode_v14.pml:103, state 46, "p3Token = 0"
        MyCode_v14.pml:108, state 50, "p6Token = 0"
        MyCode_v14.pml:109, state 51, "p7Token = 0"
        .....
        (42 of 317 states)
unreached in proctype mainClockState
        (0 of 21 states)
unreached in init
        (0 of 3 states)
unreached in claim pe0
        _spin_nvr.tmp:8, state 10, "-end-"
        (1 of 10 states)

pan: elapsed time 0.003 seconds
No errors found -- did you verify all claims?
```

รูปที่ 5.8 ผลการทวนสอบแบบจำลองโดยใช้รูปแบบแอลทีแอล pe0

จากผลการทวนสอบแสดงผลที่ได้คือแบบจำลองใช้รูปแบบแอลทีแอล pe0 ในการทวนสอบสามารถทำงานได้ใน 115 สถานะโดยไม่มีข้อผิดพลาด

4) สรุปผล

ผลการทวนสอบแบบจำลองในกรณีที่ใชแอลที่แอล pe0 สามารถสรุปได้ว่า ณ ช่วงเวลาที่ 1 – 9 จะไม่สามารถบรรจุซองและปิดผนึกได้ ซึ่งแสดงผลดังรูปที่ 5.8

5.4. การทวนสอบแบบจำลองในรูปแบบแอลทีแอล สำหรับเงื่อนไขการทวนสอบที่ 2

1) เงื่อนไขในการทวนสอบ

การทวนสอบด้วยเงื่อนไขที่ว่า ใน 1 รอบของการทำงานระบบจะดึงใบแจ้งยอดหนี้บัตรเครดิตและใบแนบโฆษณา 1 ครั้งในช่วงเวลาที่ 1 – 2

ด้วยเงื่อนไขการทวนสอบและจากสมการที่ 4. 2 สามารถเขียนเป็นสมการในรูปของเอ็มทีแอลได้ดังนี้

$$\Diamond_{[1,\,2]}(P)$$

โดยที่ P แทนด้วย เหตุการณ์ดึงใบแจ้งยอดหนี้บัตรเครดิตและใบแนบโฆษณา

นำไปแปลงให้อยู่ในรูปแบบของแอลทีแอลที่ใช้ในการทวนสอบได้ดังรูปที่ 5.9

```
1       #define MTL_eventually (p2Token == 1 && ((mainCl >=  1)&&(mainCl <= 2)))
2       ltl pe1 { <> MTL_eventually}
```

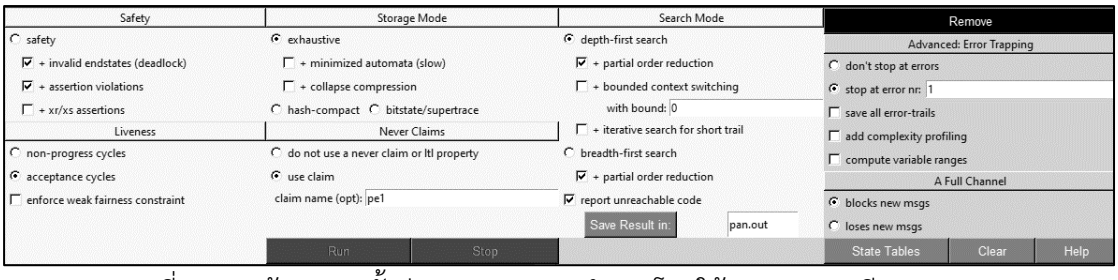
รูปที่ 5.9 เงื่อนไขการทวนสอบแบบจำลองในรูปแบบแอลทีแอลในกรณีศึกษาที่ 2

2) ผลที่คาดว่าจะได้รับ

เหตุการณ์การดึงใบแจ้งยอดหนี้บัตรเครดิตและใบแนบโฆษณาจะทำงานได้ในเวลาที่ 1 - 2

3) ผลการทวนสอบ

จากผลการทวนสอบด้วยรูปแบบแอลทีแอล pe1 ดังรูปที่ 5.9



รูปที่ 5.10หน้าจอการตั้งค่าทวนสอบแบบจำลองโดยใช้รูปแบบแอลทีแอล pe1

และผลการทวนสอบแบบจำลองและการแปลงคุณลักษณะดังกล่าวแสดงว่าเหตุการณ์ดึงใบแจ้งยอดหนี้บัตรเครดิตและใบแนบโฆษณาทำงานได้ในเวลาที่ 1 - 2 ดังรูปที่ 5.11

```
spin -a  MyCode_v14.pml
ltl pe1: <> (((p2Token==1)) && (((mainCI>=1)) && ((mainCI<=2))))
C:/cygwin64/bin/gcc.exe -DMEMLIM=1024 -O2 -DXUSAFE -w -o pan pan.c
./pan -m10000  -a -c1 -N pe1
Pid: 8096
warning: only one claim defined, -N ignored

(Spin Version 6.4.3 -- 16 December 2014)
        + Partial Order Reduction

Full statespace search for:
        never claim         + (pe1)
        assertion violations + (if within scope of claim)
        acceptance   cycles  + (fairness disabled)
        invalid end states - (disabled by never claim)

State-vector 152 byte, depth reached 24, errors: 0
       6 states, stored (12 visited)
       4 states, matched
      16 transitions (= visited+matched)
      28 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
   0.001   equivalent memory usage for states (stored*(State-vector + overhead))
   0.252   actual memory usage for states
 128.000 memory used for hash table (-w24)
   0.534   memory used for DFS stack (-m10000)
 128.730 total actual memory usage


unreached in proctype mainClass
        MyCode_v14.pml:66, state 14, "(1)"
        MyCode_v14.pml:87, state 36, "p5Token = 1"
        MyCode_v14.pml:88, state 37, "p2Token = 0"
        MyCode_v14.pml:93, state 41, "p4Token = 0"
        MyCode_v14.pml:94, state 42, "p5Token = 0"
        MyCode_v14.pml:99, state 46, "p3Token = 0"
        MyCode_v14.pml:104, state 50, "p6Token = 0"
        MyCode_v14.pml:105, state 51, "p7Token = 0"
        .....
        (86 of 317 states)
unreached in proctype mainClockState
        (0 of 21 states)
unreached in init
        (0 of 3 states)
unreached in claim pe1
        _spin_nvr.tmp:6, state 6, "-end-"
        (1 of 6 states)

pan: elapsed time 0.002 seconds
No errors found -- did you verify all claims?
```

รูปที่ 5.11 ผลการทวนสอบแบบจำลองโดยใช้รูปแบบแอลทีแอล pe1

4) สรุปผล

เหตุการณ์ที่การดึงใบแจ้งยอดหนี้บัตรเครดิตและใบแนบโฆษณาพร้อมทำงานในเวลาที่ 1 - 2 จำนวน 1 ครั้งใน 1 รอบการทำงานของสายพานการบรรจุซองซึ่งสามารถทำงานได้อย่างถูกต้อง

5.5. การทวนสอบแบบจำลองในรูปแบบแอลทีแอล สำหรับเงื่อนไขการทวนสอบที่ 3

1) เงื่อนไขในการทวนสอบ

การทวนสอบด้วยเงื่อนไขที่ว่า เมื่อมีการเปิดซองใบแจ้งยอดหนี้เพื่อรอการบรรจุและใบแจ้งยอดหนี้พร้อมกับใบแนบโฆษณาอยู่ในถาดเรียบร้อยแล้ว ขั้นตอนต่อไประบบสามารถที่จะทำการบรรจุและปิดผนึกซองได้

ด้วยเงื่อนไขการทวนสอบและจากสมการที่ 4. 3 เขียนเป็นสมการในรูปของ เอ็มทีแอลได้ดังนี้

$$\square_{[6,\,7]}((P1 \wedge P2) \rightarrow \bigcirc_{[10,\,11]} (P3))$$

โดยที่ P1 แทนด้วย เหตุการณ์การเปิดซองอัตโนมัติเพื่อรองรับการบรรจุ

P2 แทนด้วย เหตุการณ์การรวมใบแจ้งยอดหนี้บัตรเครดิตและใบแนบโฆษณาในถาด

P3 แทนด้วย เหตุการณ์การบรรจุใบแจ้งยอดหนี้และปิดผนึกซอง

สามารถอธิบายให้อยู่ในรูปแบบของแอลทีแอลที่ใช้ในการทวนสอบ ดังรูปที่ 5.12

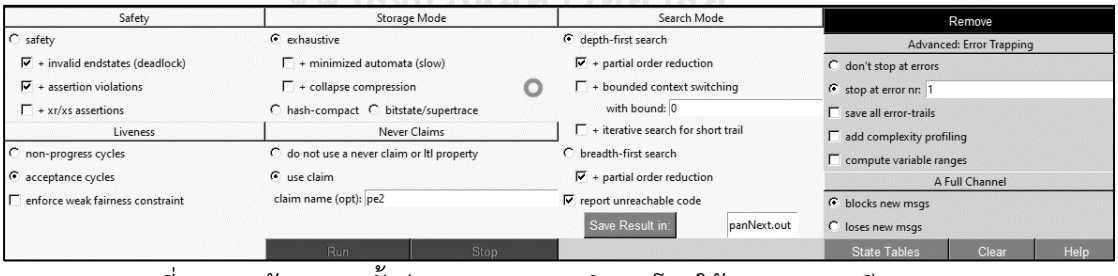
```
1       #define MTL_nextP1 (p6Token == 1 && (mainCI >= 6&&mainCI<=7)) && (p7Token == 1 && (mainCI >= 6&&mainCI<=7))
2       #define MTL_nextP2 X(p8Token == 1 && (mainCI >= 10&&mainCI<11))
3       ltl pe2 { [] (MTL_nextP1 -> MTL_nextP2)}
```

รูปที่ 5.12เงื่อนไขการทวนสอบแบบจำลองในรูปแบบแอลทีแอลในกรณีศึกษาที่ 3

2) ผลที่คาดว่าจะได้รับ

ระบบสามารถทำการบรรจุและปิดผนึกซองได้ ก็ต่อเมื่อมีการเปิดซองเพื่อรอการบรรจุพร้อมใบแจ้งยอดหนี้และใบแนบโฆษณารวมกันอยู่ในถาด

3) ผลการทวนสอบ



รูปที่ 5.13หน้าจอการตั้งค่าทวนสอบแบบจำลองโดยใช้รูปแบบแอลทีแอล pe2

จากรูปที่ 5.14 แสดงผลการทวนสอบภายใต้เงื่อนไรที่ว่าระบบสามารถบรรจุและปิดผนึกได้หลังจากที่ใบแจ้งยอดหนี้บัตรเครดิตและใบแนบโฆษณาถูกรวมกันในถาดและทำการเปิดซองรอบรรจุ

```
spin -a  MyCode_v14.pml
ltl pe2: [] ((! ((((p6Token==1)) && (((mainCI>=6)) && ((mainCI<=7)))) && (((p7Token==1)) && (((mainCI>=6)) &&
((mainCI<=7)))))) || (X (((p8Token==1)) && (((mainCI>=10)) && ((mainCI<11))))))
C:/cygwin64/bin/gcc.exe -DMEMLIM=1024 -O2 -DXUSAFE -w -o pan pan.c
./pan -m10000  -a -c1 -N pe2
Pid: 3800
warning: only one claim defined, -N ignored

(Spin Version 6.4.3 -- 16 December 2014)
        + Partial Order Reduction

Full statespace search for:
        never claim         + (pe2)
        assertion violations + (if within scope of claim)
        acceptance   cycles  + (fairness disabled)
        invalid end states - (disabled by never claim)

State-vector 152 byte, depth reached 115, errors: 0
      18 states, stored
       3 states, matched
      21 transitions (= stored+matched)
     173 atomic steps
hash conflicts:        0 (resolved)

Stats on memory usage (in Megabytes):
   0.003   equivalent memory usage for states (stored*(State-vector + overhead))
   0.250   actual memory usage for states
 128.000 memory used for hash table (-w24)
   0.534   memory used for DFS stack (-m10000)
 128.730 total actual memory usage


unreached in proctype mainClass
        MyCode_v14.pml:66, state 14, "(1)"
        MyCode_v14.pml:87, state 36, "p5Token = 1"
        MyCode_v14.pml:88, state 37, "p2Token = 0"
        MyCode_v14.pml:93, state 41, "p4Token = 0"
        .....
        (42 of 317 states)
unreached in proctype mainClockState
        (0 of 21 states)
unreached in init
        (0 of 3 states)
unreached in claim pe2
        _spin_nvr.tmp:12, state 15, "-end-"
        (1 of 15 states)

pan: elapsed time 0.002 seconds
No errors found -- did you verify all claims?
```

รูปที่ 5.14ผลการทวนสอบแบบจำลองโดยใช้รูปแบบแอลทีแอล pe2

4) สรุปผล

ระบบสามารถบรรจุและปิดผนึกซองได้หลังจากที่ นำใบแจ้งยอดหนี้บัตรเครดิตและใบแนบโฆษณา พร้อมกันกับซองที่ทำการเปิดเพื่อรอการบรรจุได้อย่างถูกต้องดังรูปที่ 5.14

5.6. การทวนสอบแบบจำลองในรูปแบบแอลทีแอล สำหรับเงื่อนไขการทวนสอบที่ 4

1) เงื่อนไขในการทวนสอบ

การทวนสอบด้วยเงื่อนไขที่ว่า ระบบสามารถเลือกประเภทของซองบรรจุใบแจ้งยอดหนี้ พร้อมกันกับการดึงใบแจ้งยอดหนี้บัตรเครดิตและใบแนบโฆษณาได้

ด้วยเงื่อนไขการทวนสอบและจากสมการที่ 4. 4 เขียนเป็นสมการในรูปของ เอ็มทีแอลได้ดังนี้

$$P1 \ U_{[1,\ 2]} \ P2$$

โดยที่ P1 แทนด้วย เหตุการณ์เลือกประเภทของซองบรรจุใบแจ้งยอดหนี้

P2 แทนด้วย เหตุการณ์ดึงใบแจ้งยอดหนี้และใบแนบโฆษณา

สามารถอธิบายให้อยู่ในรูปแบบของแอลทีแอลที่ใช้ในการทวนสอบ ดังนี้ เมื่อมีการเลือกประเภทของซองบรรจุใบแจ้งยอดหนี้ที่เวลาที่ 1 – 2 และในขณะเดียวกันระบบก็จะดึงใบแจ้งยอดหนี้บัตรเครดิตและใบแนบโฆษณาในเวลาเดียวกัน ดังรูปที่ 5.15

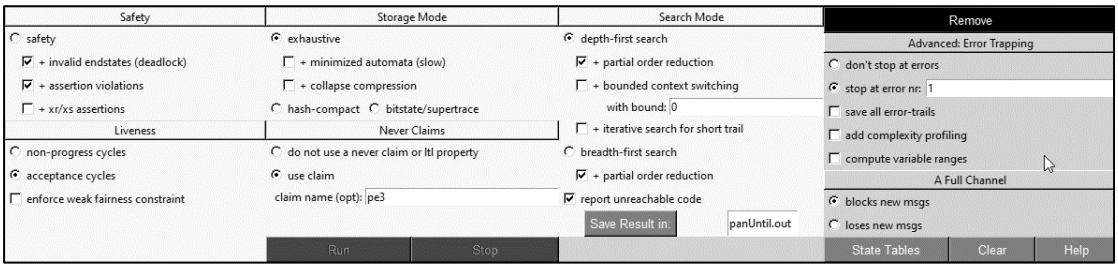
```
1    #define MTL_untilP1 (p2Token == 1 && (mainCl >= 1 && mainCl <=2))
2    #define MTL_untilP2 (p3Token == 1 && (mainCl >= 1 && mainCl <= 7))
3    ltl pe3 {[]<>MTL_untilP1 -> []<>MTL_untilP2}
```

รูปที่ 5.15เงื่อนไขการทวนสอบแบบจำลองในรูปแบบแอลทีแอลในกรณีศึกษาที่ 4

2) ผลที่คาดว่าจะได้รับ

เหตุการณ์ที่การเลือกซองของใบแจ้งยอดหนี้บัตรเครดิตสามารถทำงานไปพร้อมๆกับเหตุการณ์ที่ระบบทำการดึงใบแจ้งยอดหนี้และใบแนบโฆษณาได้

3) ผลการทวนสอบ



รูปที่ 5.16หน้าจอการตั้งค่าทวนสอบแบบจำลองโดยใช้รูปแบบแอลทีแอล pe3

จากผลการทวนสอบดังรูปที่ 5.17 เหตุการณ์ดึงใบแจ้งยอดหนี้บัตรเครดิตและใบแนบโฆษณานั้นสามารถทำงานได้พร้อมกันกับเวลาที่เหตุการณ์เลือกประเภทของซองบรรจุที่เวลา 1 – 2

```
spin -a  MyCode_v14.pml
ltl pe3: (! ([] (<> (((p2Token==1)) && (((mainCl>=1)) && ((mainCl<=2)))))) || ([] (<> (((p3Token==1)) && (((main
Cl>=1)) && ((mainCl<=7))))))
C:/cygwin64/bin/gcc.exe -DMEMLIM=1024 -O2 -DXUSAFE -w -o pan pan.c
./pan -m10000  -a -c1 -N pe3
Pid: 9704
warning: only one claim defined, -N ignored

(Spin Version 6.4.3 -- 16 December 2014)
        + Partial Order Reduction

Full statespace search for:
        never claim         + (pe3)
        assertion violations + (if within scope of claim)
        acceptance   cycles  + (fairness disabled)
        invalid end states - (disabled by never claim)

State-vector 152 byte, depth reached 115, errors: 0
     35 states, stored (44 visited)
     28 states, matched
     72 transitions (= visited+matched)
    568 atomic steps
hash conflicts:        0 (resolved)

Stats on memory usage (in Megabytes):
   0.006   equivalent memory usage for states (stored*(State-vector + overhead))
   0.246   actual memory usage for states
 128.000 memory used for hash table (-w24)
   0.534   memory used for DFS stack (-m10000)
 128.730 total actual memory usage


unreached in proctype mainClass
        MyCode_v14.pml:66, state 14, "(1)"
        MyCode_v14.pml:87, state 36, "p5Token = 1"
        MyCode_v14.pml:88, state 37, "p2Token = 0"
        MyCode_v14.pml:93, state 41, "p4Token = 0"
        MyCode_v14.pml:94, state 42, "p5Token = 0"
        MyCode_v14.pml:99, state 46, "p3Token = 0"
        .....
        (42 of 317 states)
unreached in proctype mainClockState
        (0 of 21 states)
unreached in init
        (0 of 3 states)
unreached in claim pe3
        _spin_nvr.tmp:34, state 49, "-end-"
        (1 of 49 states)

pan: elapsed time 0.002 seconds
No errors found -- did you verify all claims?
```

รูปที่ 5.17ผลการทวนสอบแบบจำลองโดยใช้รูปแบบแอลทีแอล pe3

4) สรุปผล

จากผลการทวนสอบจากรูปที่ 5.17 เมื่อเหตุการณ์การดึงใบแจ้งยอดหนี้และใบแนบโฆษณาสามารถทำงานได้พร้อมกันกับการเลือกประเภทของซองบรรจุ

5.7. การทวนสอบแบบจำลองในรูปแบบแอลทีแอล สำหรับเงื่อนไขการทวนสอบที่ 5

1) เงื่อนไขในการทวนสอบ

การทวนสอบด้วยเงื่อนไขที่ว่า จะไม่เกิดเหตุการณ์การดึงใบแจ้งยอดหนี้บัตรเครดิตและใบแนบโฆษณา หลังจากระบบเริ่มการพับใบแจ้งยอดหนี้และใบแนบโฆษณาได้อีก

ด้วยเงื่อนไขการทวนสอบและจากสมการที่ 4. 5 เขียนเป็นสมการในรูปของ เอ็มทีแอลได้ดังนี้

$$P1\ \omega_{[3,4]}\ P2$$

โดยที่ P1 แทนด้วย เหตุการณ์การดึงใบแจ้งยอดหนี้บัตรเครดิตและใบแนบโฆษณา

P2 แทนด้วย เหตุการณ์การพับใบแจ้งยอดหนี้และพับใบแนบโฆษณา

สามารถอธิบายให้อยู่ในรูปแบบของแอลทีแอลที่ใช้ในการทวนสอบ ดังรูปที่ 5.18

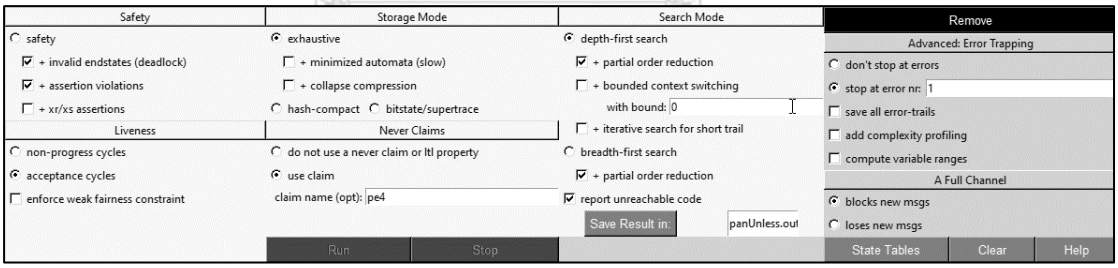
```
1       #define MTL_weakP1 (p2Token == 0 && (mainCI >= 5 && mainCI <=6))
2       #define MTL_weakP2 ((p4Token == 1 && (mainCI >=3 && mainCI <=4))  && (p5Token == 1 && (mainCI >=3 && mainCI <=4)))
3       ltl pe4 {<>[]MTL_weakP1 ->  []<>MTL_weakP2}
```

รูปที่ 5.18เงื่อนไขการทวนสอบแบบจำลองในรูปแบบแอลทีแอลในกรณีศึกษาที่ 5

2) ผลที่คาดว่าจะได้รับ

เหตุการณ์การเลือกประเภทของซองใบแจ้งยอดหนี้สามารถทำงานขนานกันกับเหตุการณ์การพับใบแจ้งยอดหนี้บัตรเครดิตและการพับใบแนบโฆษณา ภายใต้เงื่อนไขช่วงเวลาเดียวกันได้

3) ผลการทวนสอบ



รูปที่ 5.19หน้าจอการตั้งค่าทวนสอบแบบจำลองโดยใช้รูปแบบแอลทีแอล pe4

```
spin -a  MyCode_v14.pml
ltl pe4: (! (<> ([] (((p2Token==0)) && (((mainCI>=5)) && ((mainCI<=6)))))) || ([] (<> ((((p4Token==1)) && (((mai
nCI>=3)) && ((mainCI<=4)))) && (((p5Token==1)) && (((mainCI>=3)) && ((mainCI<=4))))))
C:/cygwin64/bin/gcc.exe -DMEMLIM=1024 -O2 -DXUSAFE -w -o pan pan.c
./pan -m10000  -a -c1 -N pe4
Pid: 5448
warning: only one claim defined, -N ignored

(Spin Version 6.4.3 -- 16 December 2014)
        + Partial Order Reduction

Full statespace search for:
        never claim         + (pe4)
        assertion violations + (if within scope of claim)
        acceptance   cycles  + (fairness disabled)
        invalid end states - (disabled by never claim)

State-vector 152 byte, depth reached 115, errors: 0
     36 states, stored (38 visited)
     23 states, matched
     61 transitions (= visited+matched)
    520 atomic steps
hash conflicts:        0 (resolved)

Stats on memory usage (in Megabytes):
   0.006  equivalent memory usage for states (stored*(State-vector + overhead))
   0.248  actual memory usage for states
 128.000 memory used for hash table (-w24)
   0.534  memory used for DFS stack (-m10000)
 128.730 total actual memory usage


unreached in proctype mainClass
        MyCode_v14.pml:66, state 14, "(1)"
        MyCode_v14.pml:87, state 36, "p5Token = 1"
        MyCode_v14.pml:88, state 37, "p2Token = 0"
        MyCode_v14.pml:93, state 41, "p4Token = 0"
        MyCode_v14.pml:94, state 42, "p5Token = 0"
        .....
        (42 of 317 states)
unreached in proctype mainClockState
        (0 of 21 states)
unreached in init
        (0 of 3 states)
unreached in claim pe4
        _spin_nvr.tmp:22, state 31, "-end-"
        (1 of 31 states)

pan: elapsed time 0.002 seconds
No errors found -- did you verify all claims?
```

รูปที่ 5.20ผลการทวนสอบแบบจำลองโดยใช้รูปแบบแอลทีแอล pe4

4) สรุปผล

เหตุการณ์การเลือกประเภทของซองใบแจ้งยอดหนี้สามารถทำงานขนานกันกับเหตุการณ์การพับใบแจ้งยอดหนี้บัตรเครดิตและการพับใบแนบโฆษณา ภายใต้เงื่อนไขช่วงเวลาเดียวกันได้โดยที่ไม่มีข้อผิดพลาดดังรูปที่ 5.20

5.8. การทวนสอบแบบจำลองในรูปแบบแอลทีแอลด้วยเงื่อนไขค้าน สำหรับเงื่อนไขการทวนสอบที่ 6

1) เงื่อนไขในการทวนสอบ

การทวนสอบด้วยเงื่อนไขขัดแย้งไม่เป็นไปตามกระบวนการของแบบจำลอง ด้วยเหตุการณ์การพับใบแจ้งยอดหนี้บัตรเครดิตกำลังทำงานอยู่ และเหตุการณ์รวมใบแจ้งยอดบัตรเครดติดและใบแนบโฆษณาจะเริ่มต้นการทำงานในเงื่อนไขเวลาเดียวกันในเวลาเริ่มต้นเท่ากับ 3 และเวลาสิ้นสุดเท่ากับ 4

ด้วยเงื่อนไขการทวนสอบและจากสมการที่ 4. 4 ที่ถูกเขียนเป็นสมการรูปแบบแอลทีแอลได้ ดังนี้

$$P1 \ U_{[3,\ 4]} \ P2$$

โดยที่ P1 แทนด้วย เหตุการณ์พับใบแจ้งยอดหนี้บัตรเครดิต

P2 แทนด้วย เหตุการณ์นำใบแจ้งยอดหนี้และใบแนบโฆษณาใส่ถาดเพื่อรอการบรรจุ

ในกรณีการทวนสอบสำหรับเงื่อนไขค้านดังกล่าว จะต้องทำการแก้ไขรูปแบบแอลทีแอล ให้เป็นไปตามบรรทัดที่ 3 ดังรูปที่ 5.21

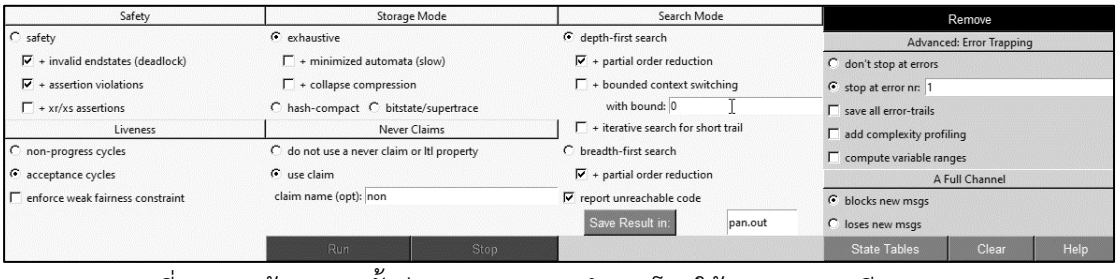
```
1    #define MTL_nonP1 (p4Token == 1 && (mainCl >= 3 && mainCl <= 4))
2    #define MTL_nonP2 (p7Token == 1 && (mainCl >= 3 && mainCl <= 4))
3    ltl pe5 {[]<>MTL_nonP1 && []<>MTL_nonP2}
```

รูปที่ 5.21เงื่อนไขการทวนสอบแบบจำลองในรูปแบบแอลทีแอลในกรณีศึกษาที่ 6

2) ผลที่คาดว่าจะได้รับ

การทวนสอบเหตุการณ์รวมใบแจ้งยอดหนี้บัตรเครดิตจะไม่สามารถทำงานได้ พร้อมกันกับเหตุการณ์พับใบแจ้งยอดหนี้บัตรเครดิตที่เวลา 3 - 4

3) ผลการทวนสอบ



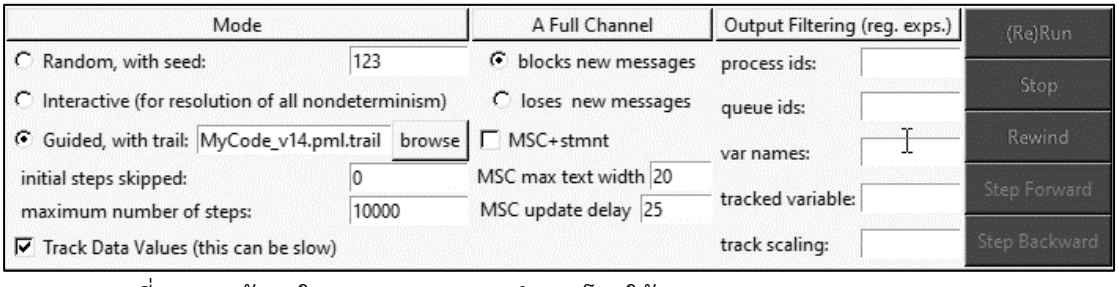
รูปที่ 5.22หน้าจอการตั้งค่าทวนสอบแบบจำลองโดยใช้รูปแบบแอลทีแอล non

```
spin -a  MyCode_v14.pml
ltl non: ([] (<> (((p4Token==1)) && (((mainCl>=3)) && ((mainCl<=4)))))) && ([] (<> (((p7Token==1)) && (((mainCl>=
3)) && ((mainCl<=4))))))
C:/cygwin64/bin/gcc.exe -DMEMLIM=1024 -O2 -DXUSAFE -w -o pan pan.c
./pan -m10000  -a -c1 -N non
Pid: 7568
warning: only one claim defined, -N ignored
pan:1: acceptance cycle (at depth 114)
pan: wrote MyCode_v14.pml.trail

(Spin Version 6.4.3 -- 16 December 2014)
Warning: Search not completed
        + Partial Order Reduction

Full statespace search for:
        never claim         + (non)
        assertion violations + (if within scope of claim)
        acceptance   cycles  + (fairness disabled)
        invalid end states - (disabled by never claim)

State-vector 152 byte, depth reached 115, errors: 1
      15 states, stored
       0 states, matched
      15 transitions (= stored+matched)
      86 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
   0.003   equivalent memory usage for states (stored*(State-vector + overhead))
   0.250   actual memory usage for states
 128.000 memory used for hash table (-w24)
   0.534   memory used for DFS stack (-m10000)
 128.730 total actual memory usage



pan: elapsed time 0.004 seconds
To replay the error-trail, goto Simulate/Replay and select "Run"
```

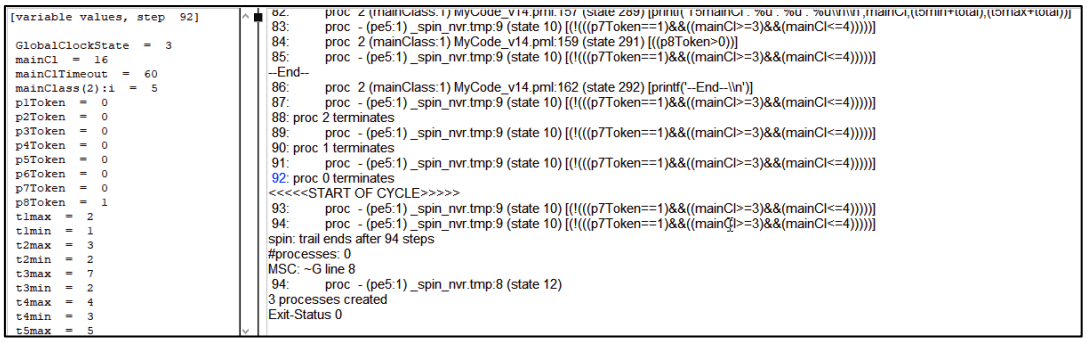
รูปที่ 5.23ผลการทวนสอบแบบจำลองโดยใช้รูปแบบแอลทีแอล non

จากผลการทวนสอบเหตุการณ์รวมใบแจ้งยอดหนี้บัตรเครดิต และการพับใบแจ้งยอดหนี้ ภายใต้เงื่อนไขเวลาที่ 3 - 4 ไม่สามารถทำงานได้ ซึ่งแสดงข้อผิดพลาดของการทวนสอบดังรูปที่ 5.23

หลังจากที่ทวนสอบแบบจำลองในเหตุการณ์ที่ขัดแย้งทำให้ผลการทวนสอบที่ออกมามีข้อผิดพลาดดังที่กล่าวไว้แล้ว ดังนั้นเพื่อทดสอบแบบจำลองดังกล่าวว่าไม่สามารถทำงานได้ จึงได้ทำการทดสอบแบบจำลองโดยการใช้ Guided, with trail ดังรูปที่ 5.24 ซึ่งผลการทดสอบดังกล่าวแสดงให้เห็นในรูปที่ 5.25 ว่าเหตุการณ์ที่ทำการทวนสอบนี้ไม่สามารถทำงานได้และระบบถูกทำลาย



รูปที่ 5.24หน้าจอในการทดสอบแบบจำลองโดยใช้ Guilded, with trail ของ non



รูปที่ 5.25ผลการทดสอบแบบจำลองโดยใช้ Guilded, with trail ของ non

4) สรุปผล

จากผลการทวนสอบเหตุการณ์รวมใบแจ้งยอดหนี้บัตรเครดิตและการพับใบแจ้งยอดหนี้ ซึ่งถูกกำหนดขึ้นให้ขัดแย้งกับการทำงานกับการทำงานแบบปกติ ภายใต้เงื่อนไขเวลาเดียวกัน ไม่สามารถทำงานได้

5.9. การทวนสอบแบบจำลองในรูปแบบแอลทีแอลด้วยเงื่อนไขค้าน สำหรับเงื่อนไขการทวนสอบที่ 7

1) เงื่อนไขในการทวนสอบ

การทวนสอบด้วยเงื่อนไขขัดแย้งของเงื่อนไขการทวนสอบที่ 1 คือระบบสามารถบรรจุใบแจ้งยอดหนี้บัตรเครดิตและใบแนบโฆษณาและปิดผนึกซองได้ตั้งแต่เวลา 1 ถึง 9

ด้วยเงื่อนไขค้านของกรณีศึกษา สามารถเขียนเป็นสมการในรูปของ เอ็มทีแอลได้ดังนี้

$$\square_{[1,\,9]}\,(P)$$

โดยที่ P แทนด้วย เหตุการณ์การบรรจุใบแจ้งยอดหนี้และปิดผนึกซอง

นำไปแปลงให้อยู่ในรูปแบบของแอลทีแอลที่ใช้ในการทวนสอบได้ดังรูปที่ 5.26 ชื่อ nonPe0

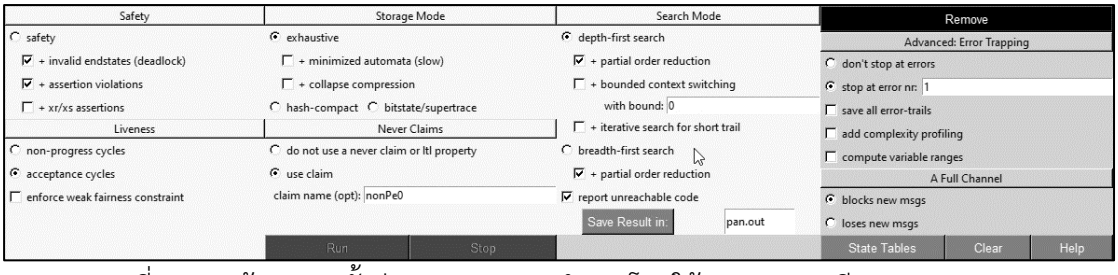
```
1   #define MTL_alwaysOver ((!(mainCl >= 1&&mainCl <= 9)) || (p8Token == 1 && (mainCl >= 1&&mainCl <= 9)))
2   ltl nonPe0 { [] MTL_alwaysOver}
```

รูปที่ 5.26เงื่อนไขการทวนสอบแบบจำลองในรูปแบบแอลทีแอลในกรณีศึกษาที่ 7

2) ผลที่คาดว่าจะได้รับ

ระบบจะไม่สามารถปิดซองได้ในเวลาที่ 1 – 9

3) ผลการทวนสอบ

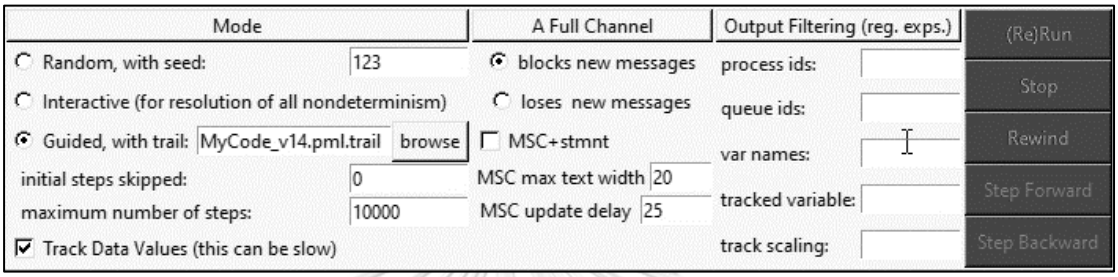


รูปที่ 5.27หน้าจอการตั้งค่าทวนสอบแบบจำลองโดยใช้รูปแบบแอลทีแอล nonPe0

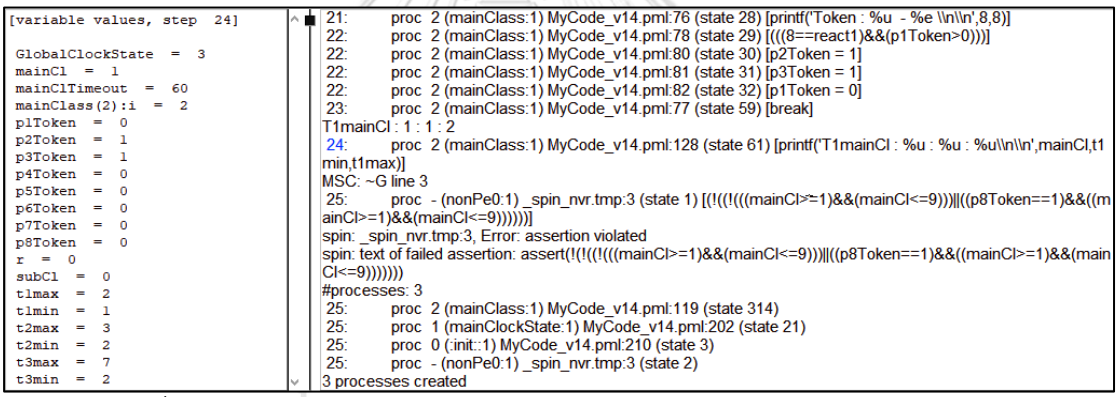
```
spin -a  MyCode_v14.pml
ltl nonPe0: [] ((! (((mainCI>=1)) && ((mainCI<=9)))) || (((p8Token==1)) && (((mainCI>=1)) && ((mainCI<=9)))))
C:/cygwin64/bin/gcc.exe -DMEMLIM=1024 -O2 -DXUSAFE -w -o pan pan.c
./pan -m10000  -a -c1 -N nonPe0
Pid: 10344
warning: only one claim defined, -N ignored
pan:1: assertion violated  !( !(( !(((mainCI>=1)&&(mainCI<=9)))||((p8Token==1)&&((mainCI>=1)&&(mainCI<=9))))))
(at depth 24)
pan: wrote MyCode_v14.pml.trail

(Spin Version 6.4.3 -- 16 December 2014)
Warning: Search not completed
        + Partial Order Reduction

Full statespace search for:
        never claim         + (nonPe0)
        assertion violations + (if within scope of claim)
        acceptance   cycles  + (fairness disabled)
        invalid end states - (disabled by never claim)

State-vector 152 byte, depth reached 24, errors: 1
       6 states, stored
       0 states, matched
       6 transitions (= stored+matched)
      14 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
   0.001   equivalent memory usage for states (stored*(State-vector + overhead))
   0.251   actual memory usage for states
 128.000 memory used for hash table (-w24)
   0.534   memory used for DFS stack (-m10000)
 128.730 total actual memory usage



pan: elapsed time 0.004 seconds
To replay the error-trail, goto Simulate/Replay and select "Run"
```

รูปที่ 5.28ผลการทวนสอบแบบจำลองโดยใช้รูปแบบแอลทีแอล nonPe0

จากผลการทวนสอบเหตุการณ์ที่บรรจุใบแจ้งยอดหนี้และปิดผนึกซองที่เวลา 1 – 9 ไม่สามารถทำงานได้ ซึ่งแสดงข้อผิดพลาดของการทวนสอบดังรูปที่ 5.28 หลังจากที่ทวนสอบแบบจำลองในเหตุการณ์ที่ขัดแย้งทำให้ผลการทวนสอบที่ออกมามีข้อผิดพลาดดังที่กล่าวไว้แล้ว ดังนั้นเพื่อทดสอบแบบจำลองดังกล่าวว่าไม่สามารถทำงานได้ จึงได้ทำการทดสอบแบบจำลองโดยการใช้ Guided, with trail ดังรูปที่ 5.29 ซึ่งผลการทดสอบดังกล่าวแสดงให้เห็นในรูปที่ 5.30 ว่าเหตุการณ์ที่ทำการทวนสอบนี้ไม่สามารถทำงานได้จนจบกระบวนการ



รูปที่ 5.29หน้าจอในการทดสอบแบบจำลองโดยใช้ Guilded, with trail ของ nonPe0



รูปที่ 5.30ผลการทดสอบแบบจำลองโดยใช้ Guilded, with trail ของ nonPe0

4) สรุปผล

จากผลการทวนสอบเหตุการณ์การบรรจุใบแจ้งยอดหนี้ลงซองและปิดผนึกไม่สามารถทำงานได้

บทที่ 6
สรุปผลงานวิจัยและข้อเสนอแนะ

6.1 สรุปผลงานวิจัย

การออกแบบแบบจำลองกรณีศึกษาของสายพานการบรรจุซองใบแจ้งยอดหนี้บัตรเครดิต ภายใต้ทฤษฎีของไทม์เพททริเนท เป็นระบบที่ถูกพัฒนาโดยใช้ภาษาโพรเมลา ด้วยโปรแกรมสปิน เพื่อจำลองการทำงานของระบบในกรณีศึกษา การทำงานของแบบจำลองจะสอดคล้องกับสัญญาณนาฬิกาหลักและสัญญาณนาฬิกาย่อย เพื่อทำการทวนสอบแบบจำลองที่ถูกพัฒนาขึ้นว่า สามารถทำงานได้หรือไม่ จึงได้นำหลักการการทวนสอบในรูปแบบแอลทีแอลที่ทำการแปลงมาจากคุณลักษณะของเอ็มทีแอล ทั้ง 5 คุณลักษณะได้แก่ ตัวดำเนินการแบบตลอดไป ตัวดำเนินการแบบในที่สุด ตัวดำเนินการแบบถัดไป ตัวดำเนินการจนกระทั่งแบบเข้ม และตัวดำเนินการจนกระทั่งแบบอ่อน มาใช้ในการทวนสอบแบบจำลอง นอกจากนี้เพื่อพิสูจน์ให้เห็นได้ชัดเจนขึ้นว่าแบบจำลอง และการแปลงคุณลักษณะที่ได้กล่าวไว้นั้นถูกต้อง ดังนั้นงานวิจัยนี้ได้ทวนสอบด้วยเงื่อนไขสอดคล้องกับแบบจำลอง เพื่อแสดงว่าระบบสามารถทำงานได้ถูกต้อง และการทวนสอบด้วยเงื่อนไขค้านการทำงานของแบบจำลอง เพื่อแสดงว่าระบบไม่สามารถทำงานได้หรือทำงานได้ไม่สมบูรณ์เมื่อมีเหตุการณ์ที่ขัดแย้งกับแบบจำลอง ซึ่งผลที่ออกมาสามารถสรุปได้ว่าการพัฒนาแบบจำลองที่สร้างขึ้นมาเพื่อเป็นกรณีศึกษานั้นสามารถทำงานได้และ การแปลงคุณลักษณะทั้ง 5 คุณลักษณะนั้น ทวนสอบได้อย่างถูกต้อง

6.2 ข้อจำกัดงานวิจัย

สำหรับงานวิจัยนี้จะทำการแปลงสัญลักษณ์ของเวลาที่เป็นเวลา [tmin, tmax] ซึ่งค่า tmin จะต้องมีค่าเท่ากันกับ tmax เท่านั้น

6.3 ข้อเสนอแนะ

สำหรับในงานวิจัยนี้ได้นำเครื่องมือสปินมาประยุกต์เพื่อช่วยในการทวนสอบแบบจำลองในกรณีศึกษาที่หลังจากได้ทำการทวนสอบแบบจำลองกรณีศึกษาและหลักการแปลงรูปแบบเอ็มทีแอลเป็นรูปแบบแอลทีแอลในการสร้างตัวดำเนินการต่างๆ เพื่อทวนสอบแบบจำลอง ซึ่งสามารถสรุปได้ว่าสามารถทำงานได้ตามข้อสันนิษฐานที่วางไว้ได้อย่างถูกต้อง แต่อย่างไรก็ตามหากมีระบบใดๆ ก็ตามที่สนใจการทวนสอบแบบระบุเช่วงวลาที่ ค่าเวลา tmin และ tmax เป็นมีค่าไม่เท่ากัน สามารถนำไปพัฒนาเพิ่มเติมได้ด้วยการเพิ่มการทำงานแบบการสุ่มค่าตัวเลข เพื่อหาค่าเวลาสุ่มที่สามารถยิงโทเคนระหว่างเวลาที่อยู่ในช่วง tmin และ tmax ได้

บรรณานุกรม

1. Pnueli, A., *THE TEMPORAL LOGIC OF PROGRAMS*. 1977, IEEE Computer Society Washington, DC, USA. p. 46-67.
2. Galton, A., *Temporal Logic*, in *Stanford Encyclopedia of Philosophy (SEP)*. 2010.
3. Thati, P. and G. Ro¸su, *Monitoring Algorithms for Metric Temporal Logic Specifications*. 2005, Electronic Notes in Theoretical Computer Science 113. p. 145–162.
4. Ouaknine, J. and J. Worrell, *Some Recent Results in Metric Temporal Logic*. 2008, Springer-Verlag Berlin Heidelberg 2008. p. 1-13.
5. Holzmann, G.J., *The Model Checker SPIN*, in *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*. May 1997.
6. *SPIN*. Available from: http://spinroot.com/spin/whatispin.html.
7. Bosnacki, D. and D. Dams, *Integrating Real Time into Spin : A Prototype Implementation*, in *FORTE/PSTV XVIII Conference*. 1998: Kluwer. p. 423-439.
8. Ray, S., *Scalable Techniques for Formal Verification :Springer Science + Business Media*. 2010.
9. Zheng-yi, T., et al., *The Specification and Verification of Real-time System Based on the Temporal Logic of Action*, in *201O International Conference on Computer Application and System Modeling (ICCASM 2010)*. 2010. p. 514-517.
10. Konur, S., *Real-time and Probabilistic Temporal Logics: An Overview.* Frontiers of Computer Science 7(3), 2013: p. 370-403.
11. Popova-Zeugmann, L., *Time Petri Nets*. 2013, Springer-Verlag Berlin Heidelberg.
12. Wang, J., *Petri Nets for Dynamic Event-Driven System Modeling*. 2007, CRC Press LLC.
13. Sukvanich, P., A. Thongtak, and W. Vatanawood, *Translating Basic Metric Temporal Logic Formulas into Promela*, in *Information Science and Applications (ICISA) 2016*. 2016. p. 1035-1043.

14.Ge, N., M. Pantel, and S.D. Zilio, *Formal Verification of User-Level Real-Time Property Patterns*. 2017, European Union.

ประวัติผู้เขียน

ชื่อ-สกุล	จุฑามาศ กะวิเศษ
วัน เดือน ปี เกิด	27 ธันวาคม 2528
สถานที่เกิด	เพชรบูรณ์
วุฒิการศึกษา	วิทยาศาสตรบัณฑิต (เทคโนโลยีสารสนเทศ)
ที่อยู่ปัจจุบัน	128 หมู่ที่ 5 ต.บ้านหวาย อ.หล่มสัก จ.เพชรบูรณ์ 67110