การแปลงไทม์เพทริเน็ตเป็นโพรเมลา

น.ส.อรสุธี ชัยชมภู

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2561

TRANSFORMATION OF TIME PETRI NET INTO PROMELA

Miss Onsuthee Chaichompoo

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 2018

หัวข้อวิทยานิพนธ์	การแปลงไทม์เพทริเน็ตเป็นโพรเมลา
โดย	น.ส.อรสุธี ชัยชมภู
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ผู้ช่วยศาสตราจารย์ ดร.อาทิตย์ ทองทักษ์
อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม	รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

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

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(ศาสตราจารย์ ดร.บุญเสริม กิจศิริกุล)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(ผู้ช่วยศาสตราจารย์ ดร.อาทิตย์ ทองทักษ์)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม
(รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ)

..... กรรมการ
(รองศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์)

..... กรรมการภายนอกมหาวิทยาลัย
(ผู้ช่วยศาสตราจารย์ ดร.เด่นดวง ประดับสุวรรณ)

อรสุธี ชัยชมภู : การแปลงไทม์เพทริเน็ตเป็นโพรเมลา. (
TRANSFORMATION OF TIME PETRI NET INTO PROMELA) อ.ที่ปรึกษาหลัก : ผศ. ดร.อาทิตย์ ทองทักษ์, อ.ที่ปรึกษาร่วม : รศ. ดร.วิวัฒน์ วัฒนาวุฒิ

วิทยานิพนธ์นี้เสนอวิธีการแปลงไทม์เพทริเน็ตเป็นโพรเมลา เพื่อตรวจสอบคุณสมบัติพฤติกรรมของไทม์เพทริเนตที่เป็นระบบเวลา โดยคุณสมบัติที่ตรวจสอบมีคุณสมบัติเชิงคุณภาพและคุณสมบัติเชิงปริมาณ

ประการแรกการสร้างกฎการแปลงไทม์เพทริเน็ตเป็นโพรเมลา นำส่วนประกอบของไทม์เพทริเน็ต ตัวแปรเวลา พฤติกรรมของเวลา พฤติกรรมการเคลื่อนที่โทเค็น และโครงสร้างเนตมาสร้างกฎสี่กฎ ได้แก่ กฎกำหนดชื่อตัวแปรโพรเมลา กฎกำหนดการเปลี่ยนแปลงเวลา กฎการเคลื่อนที่โทเค็น และกฎโครงสร้างเน็ต ตามลำดับ เมื่อได้กฎการแปลงไทม์เพทริเน็ตเป็นโพรเมลาแล้ว จะได้โพราเมลา 4 ส่วน จากนั้นรวมโพรเมลาทั้งหมดจึงได้โพรเมลาของไทม์เพทริเน็ต ประการที่สองเครื่องมือสนับสนุนการแปลงไทม์เพทริเน็ตเป็นโพรเมลานั้น เนื่องจากไทม์เพทริเน็ตเป็นสัญลักษณ์เชิงรูปภาพ ดังนั้นเปลี่ยนสัญลักษณ์เชิงรูปภาพเป็นข้อมูลตัวอักษรที่เรียกว่าพีเอ็นเอ็มแอล แต่พีเอ็นเอ็มแอลมาตรฐานไม่รองกับไทม์เพทริเน็ต ฉะนั้นวิทยานิพนธ์นี้เสนอพีเอ็นเอ็มสำหรับไทม์เพทริเน็ตด้วย เพื่อทำให้พีเอ็นเอ็มแอลเป็นข้อมูลนำเข้าของเครื่องมือสนับสนุนการแปลงไทม์เพทริเน็ตเป็นโพรเมลา เมื่อนำข้อมูลนำเข้าพีเอ็นเอ็มแอลเข้าเครื่องมือสนับสนุนการแปลงแล้วจะได้โพรเมลาของไทม์เพทริเน็ตมา ประการที่สามคุณสมบัติที่ตรวจสอบมีคุณสมบัติเชิงคุณภาพและคุณสมบัติเชิงปริมาณ โดยการตรวจสอบนำโพรเมลาของไทม์เพทริเน็ตและแอลทีแอลไปตรวจสอบด้วยเครื่องมือสปิน ได้ผลลัพธ์การตรวจสอบที่ผลลัพธ์การตรวจสอบที่ผ่านและผลลัพธ์การตรวจสอบที่ไม่ผ่านอันเป็นไปตามผลลัพธ์ที่คาดหวัง

สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่อนิสิต .....
ปีการศึกษา 2561 ลายมือชื่อ อ.ที่ปรึกษาหลัก .....
ลายมือชื่อ อ.ที่ปรึกษาร่วม .....

5870985821 : MAJOR SOFTWARE ENGINEERING
KEYWORD: Formal Model, Time Petri Net, Promela

Onsuthee Chaichompoo :

TRANSFORMATION OF TIME PETRI NET INTO PROMELA. Advisor: Asst. Prof. Dr. Arthit Thongtak Co-advisor: Assoc. Prof. Dr. Wiwat Vatanawood

This thesis proposes a method transformation of Time Petri net into Promela. To verify Time Petri net behavioral which is time system. Property verified are qualitative and quantitative.

Firstly, Rule creation of transformation of Time Petri net into Promela. Take the component of Time Petri net, time behavior, token dynamic behavior, and net structure. Which are used create four rules respectively: rule of declare Promela variable, rule of time behavior, rule of token dynamic behavior, and rule of net structure. After rules creation, there are 4 Promela parts. Then assemble all parts which is Promela of Time Petri net. Secondly, the tool which support transformation of Time Petri net into Promela. Because Time Petri net is graphic symbol Therefore, changing the graphic symbol is character data called PNML. But the standard PMNL is not match to Time Petri net. So, this thesis proposes PNML for Time Petri net as well. For import PNML data into the tool which support transformation of Time Petri net into Promela, when importing PNML input data into the tool, then Promela of Time Petri net created. Thirdly, Property verified are qualitative and quantitative. Verification Promela of Time Petri net with LTL using SPIN, verification results are both passed result and failed result, which satisfied expected results.

Field of Study: Software Engineering Student's Signature .....
Academic Year: 2018 Advisor's Signature .....
Co-advisor's Signature .....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี เนื่องจากได้รับความกรุณาและความ ช่วยเหลือเป็นอย่างดียิ่งจาก ผู้ช่วยศาสตราจารย์ ดร.อาทิตย์ ทองทักษ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ และ รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิอาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม ซึ่งท่านทั้งสองได้ กรุณาให้ความรู้ คำแนะนำ คำปรึกษาและข้อคิดเห็นต่างๆ แก่ผู้วิจัยด้วยความตั้งใจ และเอาใจใส่ เป็นอย่างดีมาโดยตลอด จึงขอกราบขอบพระคุณเป็นอย่างสูง ณ ที่นี้ด้วย

ขอขอบพระคุณ ศาสตราจารย์ ดร. บุญเสริม กิจศิริกุลประธานกรรมการสอบ รองศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์ และ ผศ.ดร.เด่นดวง ประดับสุวรรณ คณะกรรมการสอบที่กรุณาสละเวลา ให้คำแนะนำและชี้ให้เห็น ถึงข้อบกพร่องต่างๆ ของงานวิจัยนี้

ขอขอบคุณบุคคลที่เกี่ยวข้องทุกท่านที่ได้ให้ความช่วยเหลือ และคอยอำนวยความ สะดวกในด้านต่างๆ ทำให้การทำงานวิจัยสำเร็จลุล่วงด้วยดี

ขอขอบคุณบิดา มารดา ที่คอยสนับสุน เป็นกำลังใจให้แก่ผู้วิจัยเป็นอย่างดีมาโดย ตลอดจนงานวิจัยสามารถสำเร็จลุล่วงไปได้ด้วยดี

อรสุธี ชัยชมภู

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญรูปภาพ

หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหา

โมเดลเช็คกิ้งเป็นวิธีการหนึ่งที่ตรวจสอบเชิงรูปนัย (Formal verification) โดยการสร้างแบบจำลองของระบบที่สนใจแล้วจากนั้นจึงนำมาตรวจสอบคุณสมบัติต่างๆว่าระบบที่พิจารณานั้นมีคุณสมบัติเป็นไปตามข้อกำหนดหรือไม่ คุณสมบัติในการตรวจสอบเชิงรูปนัยที่ใช้ในโมเดลเช็คกิ้งสามารถเขียนอธิบายได้ด้วยแอลทีแอล ซึ่งในปัจจุบันการตรวจสอบแบบโมเดลเช็คกิ้งนั้น ได้มีงานวิจัยเรื่องวิธีการสร้างแบบจำลองระบบคอนเคอเรนท์ (Concurrent system) ด้วยโพรเมลา และใช้เครื่องมือตรวจสอบโมเดลสปิน (SPIN) ทำการตรวจสอบคุณสมบัติด้านความปลอดภัย (Safety) และคุณสมบัติความคงอยู่ (Liveness) [1, 4, 5, 6] แต่ทั้งนี้ ยังไม่มีการศึกษาวิธีการสร้างแบบจำลองระบบคอนเคอเรนท์ที่มีเงื่อนไขเวลาหรือเรียกว่าระบบไทม์ (Time system) โดยใช้โพรเมลา

คณุตม์ บุญเรืองขาวและคณะ [1] เสนอวิธีการสร้างแบบจำลองเชิงรูปนัยสำหรับซิกแนลทรานซิชันกราฟ (Signal transition graph) โดยมีโทเค็นในแบบจำลอง ซึ่งแสดงถึงการเคลื่อนที่โทเค็นแบบพลวัตเพื่อใช้ตรวจสอบคุณสมบัติบางประการของวงจรที่แสดงด้วยซิกแนลทรานซิชันกราฟ เช่น คุณสมบัติความปลอดภัย และคุณสมบัติความทนทาน เป็นต้น แต่สำหรับวงจรที่ซับซ้อนมากขึ้น การเคลื่อนที่ของโทเค็นจะประกอบด้วยเงื่อนไขที่มากขึ้น ส่งผลให้การพิจารณาคุณสมบัติที่อธิบายด้วยซิกแนลทรานซิชันกราฟทำได้ยากยิ่งขึ้น อีกทั้งซิกแนลทรานซิชันกราฟมักจะใช้ในการอธิบายแบบจำลองฮาร์ดแวร์ แต่เพทริเน็ตสามารถอธิบายแบบจำลองหรือพฤติกรรมสำหรับระบบอื่นๆได้ ซึ่งแสดงเป็นแบบจำลองได้ทั้งซอฟต์แวร์และฮาร์ดแวร์ เช่น การสื่อสารโปรโตคอลเน็ตเวิร์ค ระบบเรียลไทม์ (Real-time system) [2, 4] เป็นต้น

ดังนั้นงานวิจัยนี้เสนอวิธีการสร้างแบบจำลองเชิงรูปนัยสำหรับไทม์เพทริเน็ต (Time Petri Nets) ด้วยโพรเมลา โดยเพิ่มการเคลื่อนที่ของโทเค็น (Token flow) แบบพลวัตเข้าไปในแบบจำลอง อีกทั้งเพิ่มเงื่อนไขในการเคลื่อนที่ของโทเค็นด้วย เช่น เงื่อนไขเวลา เงื่อนไขน้ำหนักโทเค็น เป็นต้น ซึ่งเงื่อนไขต่างๆส่งผลต่อการเคลื่อนที่ของโทเค็น โดยเฉพาะไทม์เพทริเน็ตที่มีความซับซ้อนดังเช่น เนตทางเลือก (Free-choice net) เพื่อใช้ตรวจสอบคุณสมบัติที่สนใจพิจารณา โดยการตรวจสอบโมเดลจะทำการตรวจสอบ 2 ประเภท ได้แก่ การตรวจสอบเชิงคุณภาพ (Qualitative checking) เป็นการตรวจสอบคุณสมบัติทั่วไปในเพทริเน็ต ซึ่งใช้วิธีการยืนยันในการตรวจสอบได้ เช่น ตรวจสอบ

คุณสมบัติความปลอดภัย และคุณสมบัติความคงอยู่ เป็นต้น อีกประเภทหนึ่งคือการตรวจสอบเชิงปริมาณ (Quantitative checking) เป็นการตรวจสอบระยะเวลาที่เป็นข้อกำหนดไว้ในข้อกำหนดเป็นการตรวจสอบคุณสมบัติของไทม์เพทริเน็ต

1.2 วัตถุประสงค์

1. เสนอวิธีการและกฎการแปลงไทม์เพทริเน็ตเป็นโพรเมลา
2. พัฒนาเครื่องมือการแปลงไทม์เพทริเน็ตเป็นโพรเมลา

1.3 ขอบเขตการดำเนินงาน

1. ไทม์เพทริเน็ตมีโครงสร้างสัญลักษณ์ ได้แก่ เพลส ทรานซิชัน อาร์ค น้ำหนักอินพุต น้ำหนักเอาท์พุต และไทม์ โดยที่ไทม์เป็นการแสดงเป็นช่วงเวลาน้อยที่สุดและเวลามากที่สุดในการเกิดทรานซิชัน
2. กฎการแปลงนำเสนอในรูปแบบตารางความสัมพันธ์ระหว่างไทม์เพทริเน็ตและโพรเมลา
3. เครื่องมือที่พัฒนาทำการแปลงไทม์เพทริเน็ตเป็นโพรเมลา โดยมีข้อมูลนำเข้าไทม์เพทริเน็ตพีเอ็นเอ็มแอล และมีข้อมูลนำออกเป็นโพรเมลาที่ใช้เครื่องมือสปินตรวจสอบคุณสมบัติความปลอดภัยและความคงอยู่เป็นอย่างน้อย
4. กรณีศึกษาเพื่อทดสอบความถูกต้องของเครื่องมือที่พัฒนาจำนวน 3 กรณีศึกษา เป็นอย่างน้อย

1.4 ขั้นตอนการดำเนินงาน

1. ศึกษาและทำความเข้าใจทฤษฎีเกี่ยวกับเพทริเน็ต
2. ศึกษาและทำความเข้าใจเกี่ยวกับข้อกำหนดของ ไทม์เพทริเน็ต
3. ศึกษางานวิจัยต่างๆที่เกี่ยวข้อง
4. ศึกษาโพรเมลา โปรแกรมสปิน และตรรกะตามระยะเวลาชั่วขณะ
5. นำข้อกำหนด ไทม์เพทริเน็ต มาวิเคราะห์ จากนั้นสร้างแบบจำลองโดยแปลงเป็นโพรเมลา
6. แปลงคุณสมบัติต่างๆที่ต้องการตรวจสอบด้วยวิธีการยืนยันเป็นโพรเมลา
7. จำลองการทำงานโดยใช้สปิน
8. ตรวจสอบหาวิธีพัฒนาเพิ่มเติม
9. สรุปผลการวิจัยและข้อเสนอแนะ
10. จัดทำบทความวิชาการ และนำเสนอผลงานวิจัย
11. จัดทำวิทยานิพนธ์

1.5.ประโยชน์ที่คาดว่าจะได้รับ

1. ได้วิธีการการสร้างโพรเมลาจากไทม์เพทริเน็ต
2. ได้แนวทางการตรวจสอบคุณสมบัติของไทม์เพทริเน็ตจากโพรเมลาด้วยสปิน

1.6 บทความที่ตีพิมพ์จากงานวิจัย

ส่วนหนึ่งของวิทยานิพนธ์ฉบับนี้ ได้รับการตีพิมพ์เป็นบทความทางวิชาการ เรื่อง “Transformation of Time Petri Net into Promela” โดย อรสุธี ชัยชมภู, วิวัฒน์ วัฒนาวุฒิ และ อาทิตย์ ทองทักษ์ ในงานประชุมวิชาการ The 11th International Conference on Telecommunication Systems, Services, and Applications (TSSA 2017) ที่จัดขึ้นโดยโรงเรียนวิศวกรรมไฟฟ้าและสารสนเทศของสถาบันเทคโนโลยีบันดุง (ITB) และรับรองโดย IEEE อินโดนีเซีย เมื่อวันที่ 26-27 ตุลาคม พ.ศ. 2560 ณ เมืองลอมบอก สาธารณรัฐอินโดนีเซีย

1.7 เนื้อหาของวิทยานิพนธ์

เนื้อหาของวิทยานิพนธ์ฉบับนี้จะแบ่งออกเป็น 6 บท โดยบทที่ 1 จะเป็นบทนำ จากนั้นในบทที่ 2 จะกล่าวถึงทฤษฎีต่างๆที่เกี่ยวข้องกับงานวิจัย ส่วนบทที่ 3 กฎการแปลงไทม์เพทริเน็ตเป็นโพรเมลา จะเป็นการอธิบายให้เห็นภาพรวมทั้งหมดของขั้นตอนการแปลงไทม์เพทริเน็ตเป็นโพรเมลาด้วยกฎการแปลงไทม์เพทริเน็ตเป็นโพรเมลา จนกระทั่งได้โพรเมลา ซึ่งในบทนี้แสดงตัวอย่างไทม์เพทริ-เน็ต พีเอ็นเอ็มแอล และโพรเมลาด้วย บทที่ 4 เครื่องมือที่พัฒนาเพื่อสนับสนุนการแปลงไทม์เพทริ-เน็ตเป็นโพรเมลา บทที่ 5 จะเป็นการตรวจสอบแบบจำลองตัวอย่าง สุดท้ายบทที่ 6 จะเป็นบทสรุปผลงานวิจัยรวมทั้งข้อเสนอแนะ

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ไทม์เพทริเน็ต

ไทม์เพทริเน็ตเป็นแบบจำลองเพทริเน็ตที่เพิ่มข้อกำหนดจากแบบจำลองเพทริเน็ตดั้งเดิม [2] โครงสร้างของเพทริเน็ตประกอบด้วย 4 ส่วน คือ N = {P ,T ,F ,W} ซึ่งมีความหมายสัญลักษณ์ต่างๆ ดังนี้ P แทนเซตของเพลส (Place), T แทนเซตของทรานซิชัน (Transition), F แทนเซตของอาร์ค (Arc) หรือทิศทางการไหล (Flow relation) และ W แทนฟังก์ชันน้ำหนัก (Weight function)

ไทม์เพทริเน็ต [3] ประกอบด้วย 5 ส่วน คือ Z = {P, T, F, V, I} ซึ่งมีความหมายสัญลักษณ์ต่างๆดังนี้ P แทนเซตของเพลส, T แทนเซตของทรานซิชัน, F แทนเซตของอาร์คหรือทิศทางการไหล, V แทนฟังก์ชันน้ำหนักที่อาร์ค โดยมี mo เป็นมาร์คกิ้งเริ่มต้น (Initial marking), I คือ ฟังก์ชันอันตร-ภาค (Interval function) ฟังก์ชันซึ่งแสดงถึงเวลาเคลื่อนที่ (Firing time) เช่น I(t) = (I1(t), I2(t)) โดย I1(t) ≤ I2(t)

ไทม์เพทริเน็ตเขียนแทนด้วยสัญลักษณ์ได้ดังนี้

S(Z) := (P, T, F, V,mo) โดยที่

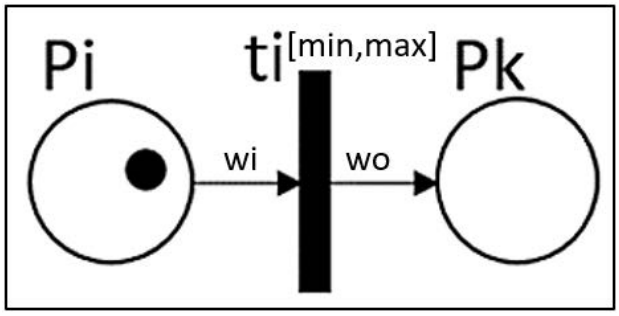
$P \cap T = \emptyset$, $P \cup T = \emptyset$, $F \subseteq (P \times T) \cup (T \times P)$,

$V : F \longrightarrow \mathbf{N+}$ แสดงน้ำหนักที่อาร์คเป็นจำนวนนับทางบวก, $mo : P \longrightarrow \mathbf{N}$ แสดง มาร์คกิ้งเริ่มต้นเป็นจำนวนนับ

$I : T \longrightarrow \mathbf{Q+0} \times (\mathbf{Q+0} \cup \{\infty\})$ แสดงเวลาเป็นจำนวนตรรกยะเศษส่วนตั้งแต่ศูนย์จนถึงอนันต์ และ I1(t) ≤ I2(t) ที่ $t \in T$, เมื่อ I(t) = (I1(t), I2(t)).

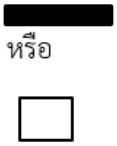
2.1.1 ส่วนประกอบเชิงรูปภาพไทม์เพทริเน็ต

ไทม์เพทริเน็ตสามารถแสดงเป็นรูปภาพโดยใช้ส่วนประกอบเชิงรูปภาพที่แสดงในตารางที่ 2-1 โดยประกอบกันเป็นแบบจำลองเชิงรูปภาพดังรูปที่ 2-1

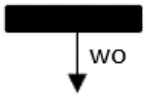


รูปที่ 2-1 ส่วนประกอบเชิงรูปภาพโดยรวม ไทม์เพทริเน็ต

ตารางที่ 2-1 ไทม์เพทริเน็ต สัญลักษณ์เชิงรูปภาพ ชื่อเรียกสัญลักษณ์ และคำอธิบาย

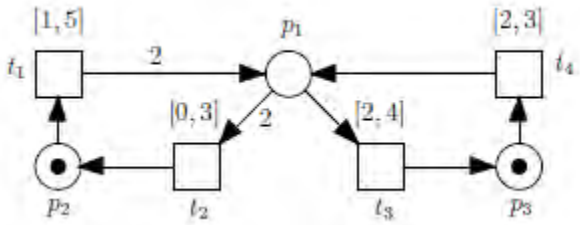
สัญลักษณ์	ชื่อเรียกสัญลักษณ์	คำอธิบาย
	เพลส	ตำแหน่งที่อยู่ของโทเค็น
	ทรานซิชัน	จุดเชื่อมต่อของการไหลโทเค็น บางกรณีมีการกำหนดเงื่อนไขการเคลื่อนที่โทเค็นด้วย เช่น เงื่อนไขเวลา เงื่อนไขน้ำหนักโทเค็น
	อาร์ค	เส้นความสัมพันธ์ ลูกศรแสดงทิศทางการไหลของโทเค็น
	น้ำหนักอินพุต	น้ำหนักของโทเค็นที่มีผลต่อการเคลื่อนที่โทเค็นและจำนวนโทเค็นก่อนเคลื่อนที่ สัญลักษณ์เขียนกำกับไว้ที่อาร์ค โดยที่น้ำหนักมีค่าตั้งแต่ 1 ขึ้นไป ตัวอย่างมีน้ำหนักเป็น wi

ตารางที่ 2-1 ไทม์เพทริเน็ต สัญลักษณ์เชิงรูปภาพ ชื่อเรียกสัญลักษณ์ และคำอธิบาย (ต่อ)

สัญลักษณ์	ชื่อเรียกสัญลักษณ์	คำอธิบาย
	น้ำหนักเอาท์พุต	น้ำหนักของโทเค็นที่มีผลต่อการเคลื่อนที่โทเค็นและจำนวนโทเค็นหลังเคลื่อนที่ สัญลักษณ์เขียนกำกับไว้ที่อาร์ค โดยที่น้ำหนักมีค่าตั้งแต่ 1 ขึ้นไป ตัวอย่าง มีน้ำหนักเป็น wo
[min,max]	ไทม์	จุดช่วงเวลาตั้งแต่เวลาหนึ่งจนถึงอีกเวลาหนึ่ง เป็นตัวกำหนดเงื่อนไขเวลาสำหรับการไหลโทเค็น สัญลักษณ์นี้อยู่คู่กับทรานซิชัน ตัวอย่าง คือ ตั้งแต่เวลาที่ min หน่วยจนถึงเวลาที่ max หน่วย

2.1.2 ทฤษฎีไทม์เพทริเน็ต [3]

สถานะ (State) ของไทม์เพทริเน็ตแทนด้วยสัญลักษณ์ z ประกอบด้วยเซตมาร์คกิ้ง (Marking) 2 ประเภท ได้แก่ พีมาร์คกิ้ง (P-marking) และทีมาร์คกิ้ง (T-marking) ซึ่งพีมาร์คกิ้ง คือ สถานะที่แสดงจำนวนโทเค็นที่อยู่แต่ละเพลสของไทม์เพทริเน็ต ณ ขณะเวลาหนึ่ง แทนด้วยสัญลักษณ์ m ส่วนทีมาร์คกิ้ง คือ เวลาที่อยู่แต่ละทรานซิชัน ณ ขณะเวลาหนึ่ง แทนด้วยสัญลักษณ์ h



รูปที่ 2-2 ตัวอย่างไทม์เพทริเน็ต [3]

เพทริเน็ตทั่วไปการเคลื่อนที่โทเค็นถูกกำหนดด้วยจำนวนโทเค็นของเพลสตั้งต้น ถ้ามีจำนวน 1 โทเค็นขึ้นไป จะทำให้โทเค็นเคลื่อนที่ไปเพลสปลายทางได้แล้ว และในกรณีที่เพลสตั้งต้นมีมากกว่า 1 ตำแหน่ง ต้องรอให้แต่ละเพลสตั้งต้นมีจำนวนโทเค็นมากกว่า 1 โทเค็นทั้งหมด จึงจะมีการเคลื่อนที่ไปเพลสปลายทางได้ แต่การเคลื่อนที่โทเค็นจากเพลสหนึ่งไปที่เพลสใดๆของไทม์เพทริเน็ตมีเงื่อนไข

กำหนดเพิ่มมากขึ้นจากเพทริเน็ตทั่วไป ซึ่งขึ้นอยู่กับเงื่อนไขเวลา เงื่อนไขน้ำหนักโทเค็น และเนตทางเลือก โดยงานวิจัยนี้ทำการกำหนดว่าเมื่อถึงเวลาที่กำหนดและน้ำหนักโทเค็นที่เพลสตั้งต้นครบตามเงื่อนไขทั้งหมดแล้ว โทเค็นจึงจะสามารถเคลื่อนที่ไปเพลสอื่นๆได้ โดยเวลากำหนดเป็นช่วงเวลาที่น้อยที่สุด (Earliest firing time) และเวลาที่มากที่สุด (Latest firing time) โดยการกำหนดเงื่อนไขน้ำหนักโทเค็นส่งผลให้เพลสตั้งต้นต้องรอจำนวนโทเค็นให้ครบตามเงื่อนไขก่อนซึ่งอาจต้องมีมากกว่า 1 โทเค็น จึงจะมีการเคลื่อนที่โทเค็นไปเพลสปลายทางเกิดขึ้นได้

จากตัวอย่างรูปที่ 2-2 จะได้สัญลักษณ์ของโครงสร้างส่วนประกอบไทม์เพทริเน็ตดังนี้

Z = (m,h)

m = (p1,p2,p3)

h = (t1,t2,t3,t4)

แสดงตัวอย่างสถานะของสถานะใดๆ ดังด้านล่าง ได้แก่ สถานะ Z0 และสถานะ Z1

สถานะตั้งต้น (Initial state) Z0 = (m0,h0) โดยที่

m0 = (0,1,1) แสดงจำนวนโทเค็นที่อยู่เพลส p2 มี 1 โทเค็นและเพลส p3 มี 1 โทเค็น

h0 = (0,#,#,0) แสดงทรานซิชัน t1 และ t4 มีเวลาเป็น 0 หน่วยเวลา

สถานะ Z1 (State Z1) = (m1,h1) โดยที่

m1 = (1,1,0) แสดงถึงมีโทเค็นอยู่ที่เพลส p1 1 โทเค็นเพลส p2 1 โทเค็นและเพลส p3 0 โทเค็น

h1 = (3,#,#,3) แสดงถึงทรานซิชัน t1 และ t4 มีเวลาเป็น 3 หน่วยเวลา

เขียนการเปลี่ยนสถานะเป็นสัญลักษณ์ได้ดังนี้

$Z0 \xrightarrow{3.0} Z1$ หรือ $((0,1,1),(0,\#,\#,0)) \xrightarrow{3.0} ((1,1,0),(3,\#,\#,3))$

2.1.3 พฤติกรรมของไทม์เพทริเน็ต

พฤติกรรมของไทม์เพทริเน็ตพิจารณาตามความพร้อมที่ทรานซิชัน มี 3 สถานะ ได้แก่ สถานะปิดใช้งาน (Disable), สถานะเปิดใช้งาน (Enable) และสถานะสแตนด์บาย (Standby)

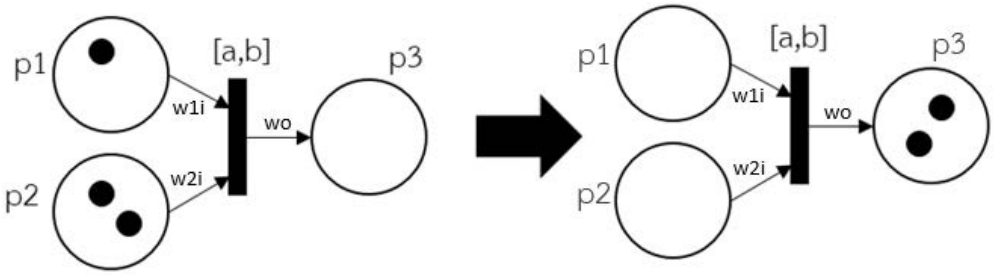
สถานะปิดใช้งาน คือ พฤติกรรมที่ทรานซิชันไม่พร้อมให้โทเค็นเคลื่อนที่ผ่านจากอินพุตเพลสไปเอาท์พุตเพลสทำให้โทเค็นไม่มีการเคลื่อนที่ และในขณะเดียวกันนั้นเวลาที่ทรานซิชันจะนับไปเรื่อยๆจนกระทั่งทรานซิชันพร้อมให้โทเค็นเคลื่อนที่ผ่าน จากนั้นเวลาที่ทรานซิชันจึงจะหยุดนับ โดยเงื่อนไขที่ทำให้เกิดสถานะปิดใช้งาน คือ จำนวนโทเค็นที่อินพุตเพลสมีค่าไม่ถึงน้ำหนักโทเค็นอินพุตหรือเวลาของระบบไม่ถึงเวลาที่น้อยที่สุดหรือมากที่สุดของทรานซิชัน

สถานะเปิดใช้งาน คือ พฤติกรรมที่ทรานซิชันพร้อมให้โทเค็นเคลื่อนที่ผ่านจากอินพุตเพลสไปเอาท์พุตเพลส โดยเงื่อนไขที่ทำให้เกิดสถานะเปิดใช้งาน คือ จำนวนโทเค็นที่อินพุตเพลสมีค่าเท่ากับหรือมีค่ามากกว่าน้ำหนักอินพุตสำหรับทุกๆเพลสที่เป็นสมาชิกเซตเพลส และสำหรับไทม์เพทริเน็ตมีเงื่อนไขเวลาเพิ่มด้วย เงื่อนไขเวลา คือ เวลาของระบบมีค่าอยู่ในช่วงตั้งแต่เวลาที่น้อยที่สุดหรือมากที่สุดของทรานซิชันหรือไม่ เขียนเป็นสมการได้ดังรูปที่ 2-3

$$\forall p \in P, \quad M(p) \geq Pre(p, t_j)$$

รูปที่ 2-3 สมการเงื่อนไขที่ทำให้ทรานซิชันเกิดสถานะเปิดใช้งาน [8]

หลังจากโทเค็นเคลื่อนที่จากอินพุตเพลสไปยังเอาท์พุตเพลสแล้ว พฤติกรรมเวลาที่ทรานซิชันสถานะเปิดนั้นจะถูกรีเซต จำนวนโทเค็นที่อินพุตเพลสเท่ากับจำนวนโทเค็นเดิมลบด้วยค่าน้ำหนัก-อินพุตและจำนวนโทเค็นที่เอาท์พุตเพลสเท่ากับจำนวนเดิมบวกด้วยค่าน้ำหนักเอาท์พุต



รูปที่ 2-4 พฤติกรรมของไทม์เพทริเน็ตที่มีการเคลื่อนที่โทเค็นอินพุตเพลสไปยังเอาท์พุตเพลส

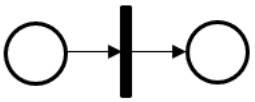
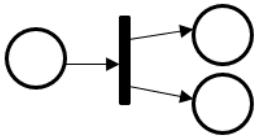
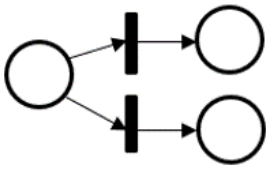
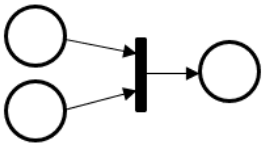
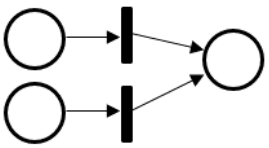
พฤติกรรมของไทม์เพทริเน็ตพิจารณาช่วงเวลาที่เป็นเงื่อนไขส่งผลต่อการเคลื่อนที่โทเค็น ซึ่งต่างจากเพทริเน็ตทั่วไป รูปที่ 2-4 ทรานซิชันมีช่วงเวลาเงื่อนไขเป็น [a, b] อินพุตเพลส p1 มีจำนวนโทเค็น p1token ซึ่งเท่ากับน้ำหนักอินพุตและอินพุตเพลส p2 มีจำนวนโทเค็น p2token ซึ่งเท่ากับน้ำหนักอินพุตแต่เวลาระบบยังไม่ถึงเวลาที่ a หน่วย ทำให้ทรานซิชันพฤติกรรมเป็นสถานะปิดใช้งาน จากนั้นเมื่อเวลาเพิ่มขึ้นเรื่อยๆจนกระทั่งถึงเวลาที่ a หน่วย ส่งผลให้ทรานซิชันเป็นสถานะเปิดใช้งาน เพราะเงื่อนไขครบพร้อมให้โทเค็นเคลื่อนที่ ต่อมาจึงเป็นสถานะหลังเคลื่อนย้ายโทเค็นทำให้โทเค็นเคลื่อนที่จากอินพุตเพลส p1 และ p2 ไปยังเอาท์พุตเพลส p3 จำนวนโทเค็นที่อินพุตเพลส p1 และp2 เป็นศูนย์ทั้งคู่ โดยที่จำนวนโทเค็นที่เอาท์พุตเพลส p3 มี p3token เพราะจำนวนเท่ากับค่าโทเค็นเดิมบวกกับค่าน้ำหนักเอาท์พุต ในขณะเดียวกันนั้นเวลาที่ทรานซิชันจะถูกรีเซต

สถานะสแตนด์บาย คือ ไทม์เพทริเน็ตที่ทุกทรานซิชันมีสถานะปิดใช้งาน ซึ่งโดยปกติแล้วไทม์เพทริเน็ตจะต้องมีทรานซิชันที่มีสถานะเปิดใช้งานอย่างน้อยหนึ่งทรานซิชันและทรานซิชันอื่นๆมีสถานะปิดใช้งาน

2.1.4 โครงสร้างเนตและพฤติกรรมของการเคลื่อนที่โทเค็น

โครงสร้างเนตแบ่งเป็น 5 แบบ ได้แก่ โครงสร้างอนุกรม โครงสร้างคอนเคอเรนท์ โครงสร้างเนตทางเลือก โครงสร้างและ โครงสร้างหรือ โดยโครงสร้างเนตนั้นประกอบด้วยส่วนประกอบของไทม์เพทริเน็ตเรียงตามลำดับดังนี้ อินพุตเพลส ทรานซิชัน และเอาท์พุตเพลส ซึ่งรายละเอียดดังตารางที่ 2-2

ตารางที่ 2-2 โครงสร้างเนตและพฤติกรรมของการเคลื่อนที่โทเค็น

โครงสร้างเนต	จำนวนส่วนประกอบไทม์เพทริเน็ต			พฤติกรรมของการเคลื่อนที่โทเค็น
	อินพุตเพลส	ทรานซิชัน	เอาท์พุตเพลส	
โครงสร้างอนุกรม (sequence) 	หนึ่ง	หนึ่ง	หนึ่ง	โทเค็นเคลื่อนที่จากอินพุตเพลสไปยังเอาท์พุตเพลส 1-1
โครงสร้างคอนเคอเรนท์ 	หนึ่ง	หนึ่ง	หลาย	โทเค็นเคลื่อนที่จากอินพุตเพลสไปยังเอาท์พุตเพลส 1-n
โครงสร้างเนตทางเลือกหรือนอนดีเทอร์มินิสติค 	หนึ่ง	หลาย	หลาย	โทเค็นเคลื่อนที่จากอินพุตเพลสไปยังเอาท์พุตเพลส 1-1 โดยที่เลือกเส้นทาง1 เส้นทางที่จะเคลื่อนที่ไปยังเอาท์พุตเพลส
โครงสร้างและ (AND) 	หลาย	หนึ่ง	หนึ่ง	โทเค็นเคลื่อนที่จากอินพุตเพลสไปยังเอาท์พุตเพลส n-1 โดยที่ขณะก่อนเคลื่อนที่นั้นทุกอินพุตเพลสจะต้องมีโทเค็นอยู่
โครงสร้างหรือ (OR) 	หลาย	หลาย	หนึ่ง	โทเค็นเคลื่อนที่จากอินพุตเพลสไปยังเอาท์พุตเพลส n-1 โดยที่ขณะก่อนเคลื่อนที่นั้นบางอินพุตเพลสจะต้องมีโทเค็นอยู่

2.2 โพรเมลา [10]

โพรเมลา (Promela : Protocol/Process Meta Language) เป็นภาษาที่เครื่องมือทวนสอบสปินรองรับ งานวิจัยนี้ใช้ทำแบบจำลองเพื่อนำไปตรวจสอบต่อไป เป็นภาษาโปรแกรมระดับสูงคล้ายภาษาธรรมชาติหรือภาษามนุษย์ รูปแบบภาษามีการประกาศตัวแปรและประเภทตัวแปร กระบวนการ โครงสร้างภาษาลูปหรือเงื่อนไขการตัดสินใจ ซึ่งเหมาะแก่การนำมาสร้างแบบจำลองระบบคอนเคอเรนท์

2.2.1 การประกาศตัวแปร

การประกาศตัวแปรมีรูปแบบคำสั่งดังนี้

ชนิดข้อมูล ชื่อตัวแปร 1, ชื่อตัวแปร 2, ชื่อตัวแปร 3, ..., ชื่อตัวแปร n;

โดยที่ ชนิดข้อมูลโพรเมลามีทั้งชนิดทั่วไปและชนิดที่ประกาศขึ้นมาพิเศษแบบโครงสร้าง (struct) ซึ่งแสดงรายละเอียดชนิดข้อมูลทั่วไปดังตารางที่ 2-3 และตัวอย่างการประกาศตัวแปรด้วยชนิดข้อมูลทั่วไปกับชนิดข้อมูลแบบโครงสร้างดังตารางที่ 2-4

ตารางที่ 2-3 โพรเมลาชนิดข้อมูลทั่วไป

ชื่อชนิด	บิต	ขนาด	ค่าของข้อมูล
bit	1	unsigned	0 และ 1
bool	1	unsigned	0 และ 1
byte	8	unsigned	0 ถึง 255
mtype	8	unsigned	0 ถึง 255
short	16	signed	-2^{15} ถึง $2^{15}-1$
int	32	signed	-2^{31} ถึง $2^{31}-1$

ตารางที่ 2-4 ตัวอย่างการประกาศตัวแปรโพรเมลา

ชนิดข้อมูล	ตัวอย่างการประกาศตัวแปรโพรเมลา	คำอธิบาย
ชนิดข้อมูลทั่วไป	int p1token = 0, p2token = 1;	ประกาศตัวแปร โดยมีชนิดข้อมูลเป็น int และตัวแปรชื่อ p1token กับ p2token

ชนิดข้อมูลแบบโครงสร้าง	typedef trans{
	int time_min;
	int time_max;
	int timer;
	};
	trans t1,t2;
		ประกาศตัวแปร โดยมีชนิดข้อมูลโครงสร้างชื่อ trans ซึ่งในชนิดนี้มีสมาชิก 3 ส่วน ได้แก่ time_min time_max และ timer เป็นอิสระต่อกัน อีกทั้งเป็นสมาชิกย่อยของตัวแปร t1 และ t2 รวมเป็นมีตัวแปรทั้งหมด 6 ตัวแปร

2.2.2 การประกาศกระบวนการหรือประกาศฟังก์ชัน

การประกาศกระบวนการโพรเมลาใช้รูปแบบคำสั่ง

init{ } หรือ proctype <ชื่อกระบวนการ>{ } ตัวอย่างดังนี้

```
init
{
        โค้ดแสดงหน้าที่ของกระบวนการ
}
หรือ
proctype A()
{
โค้ดแสดงหน้าที่ของกระบวนการ
}
```

2.2.3 คำสั่งลูป do และเงื่อนไข if

ลูปและเงื่อนไขเป็นตัวควบคุมการไหลและทิศทางของโค้ด ใช้คำสั่งภายในกระบวนการที่ได้ประกาศไว้แล้ว โดยเครื่องมือสปินที่รันคำสั่ง do จะทำให้โค้ดรันวนไปเรื่อยๆไม่จบลูป ส่วนคำสั่ง if ทำให้โค้ดมีทางแยกเงื่อนไขที่รันคู่ขนานเงื่อนไขหรือนอนดีเทอร์มินิสติค

แสดงตัวอย่างการใช้งานด้านล่าง

```
do::
if::เงื่อนไขที่ 1 -> การกระทำ 1;
  ::เงื่อนไขที่ 2 -> การกระทำ 2;
else -> การกระทำ 3;
fi;
od
```

2.2.4 คำสั่ง atomic

คำสั่ง atomic เป็นคำสั่งพิเศษมีหน้าที่กำหนดทิศทางลำดับขั้นตอนการไหลของโค้ด รูปแบบคำสั่ง atomic{ขั้นตอนที่ 1 -> ขั้นตอนที่ 2 -> ขั้นตอนที่ 3} ซึ่งเครื่องมือสปินทำการรันขั้นตอนที่ 1 เสร็จก่อนแล้วตามด้วยรันขั้นตอนที่ 2 กับขั้นตอนที่ 3 ตามลำดับ ไม่มีการสลับขั้นตอนในคำสั่ง atomic การใช้คำสั่ง atomic ตัวอย่างดังนี้

atomic{timeCntTran(t2time_min,t4timer,t1timer,t3timer)->
inp1(t2wi,p1token,t2timer,t2time_min) ; outp1(t2wo,p2token,t2timer)**}**

2.3 ตรรกะเวลาเชิงเส้น (แอลทีแอล) [10]

สปินสามารถตรวจสอบคุณสมบัติด้วยตรรกะเวลาเชิงเส้น (แอลทีแอล) (LTL : Linear Temporal Logic) ในการตรวจสอบคุณสมบัติของแบบจำลองภาษาโพรเมลาเพื่อตรวจสอบในเครื่องมือสปิน โดยเรียกใช้ในเครื่องมือสปิน รูปแบบดังนี้

ltl ชื่อข้อกำหนด { ตัวดำเนินการ (ข้อกำหนด) } โดยที่ ตัวดำเนินการ ได้แก่

- ตัวดำเนินการในเงื่อนไขตลอดไป (always) แทนด้วยสัญลักษณ์ []
- ตัวดำเนินการในเงื่อนไขในที่สุด (eventually) แทนด้วยสัญลักษณ์ <>
- ตัวดำเนินการในเงื่อนไขปฏิเสธ (negation) แทนด้วยสัญลักษณ์ !

การตรวจสอบเชิงคุณภาพมุมมองไทม์เพทริเน็ตจะตรวจสอบ 3 คุณสมบัติ ได้แก่ คุณสมบัติความปลอดภัย คุณสมบัติความคงอยู่ และคุณสมบัติความทนทาน

คุณสมบัติความปลอดภัย คือ สิ่งเลวร้ายจะต้องไม่เกิดขึ้น สำหรับไทม์เพทริเน็ตนั้น แต่ละเพลสต้องมีจำนวนโทเค็นไม่เกินกว่าค่าที่กำหนด ถ้ามีเพลสใดๆที่มีจำนวนโทเค็นเกินกว่าที่กำหนดแสดงว่าไทม์เพทริเน็ตไม่มีคุณสมบัติความปลอดภัย คุณสมบัตินี้แสดงให้เห็นด้วยการที่เพลสมีจำนวนโทเค็นมากขึ้นเรื่อยๆหรือมีมากกว่าค่าน้ำหนักที่กำหนด

ตัวอย่างแอลทีแอล

ltl q1 {[] (p1token < 2 && p2token < 5)} หมายถึง

ข้อกำหนด q1 เป็นจริงต่อเมื่อ มี p1token น้อยกว่า 2 และมี p2token น้อยกว่า 5 ตลอดไป

คุณสมบัติความคงอยู่ คือ สิ่งดีจะต้องเกิดขึ้น สำหรับไทม์เพทริเน็ต คุณสมบัตินี้แสดงให้เห็นด้วยการที่ในระบบไทม์เพทริเน็ตต้องมีจำนวนโทเค็นอยู่ในระบบเสมอ ซึ่งตรวจพบได้ที่เพลสใดๆจะต้องมีโทเค็นเคลื่อนที่ผ่าน

ตัวอย่างแอลทีแอล

ltl q2 {[] (p1token+p2token+p3token) > 0} หมายถึง

ข้อกำหนด q2 เป็นจริงต่อเมื่อ มีผลรวมของ p1token+p2token+p3token มากกว่า 0 ตลอดไป

คุณสมบัติความทนทาน คือ การไม่เกิดขึ้นของสถานะสัญญาณขัดแย้งกันในเวลาพร้อมกัน สำหรับไทม์เพทริเน็ต สามารถตรวจสอบคุณสมบัตินี้ได้ คือ ที่เพลสใดๆที่สนใจนั้นมีจำนวนโทเค็นอยู่หรือไม่ขณะในเวลาใดเวลาหนึ่ง โทเค็นที่เป็นตัวแปรตามต้องไม่เคลื่อนที่ไปยังเอาท์พุตเพลสที่ไม่พึงประสงค์ เพราะต้องรอจนกว่าโทเค็นตัวแปรต้นที่สัมพันธ์กันเคลื่อนที่ผ่านไปก่อน

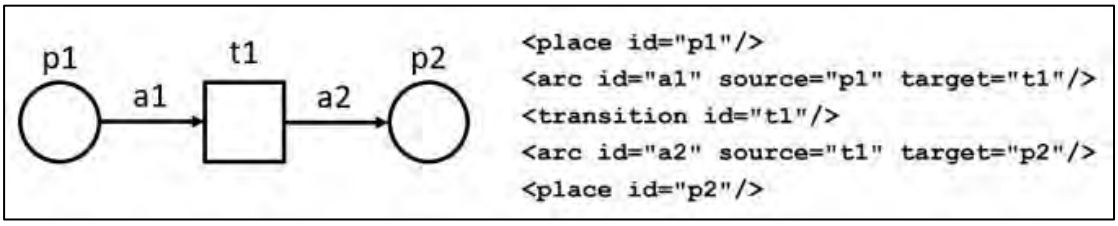
ตัวอย่างแอลทีแอล

ltl q3{[](p7token>0->p2token>0)} หมายถึง

ข้อกำหนด q3 เป็นจริงต่อเมื่อ ถ้ามี p7token มากกว่า 0 แล้วมี p2token มากกว่า 0 ตลอดไป

2.4 พีเอ็นเอ็มแอล

พีเอ็นเอ็มแอล (PNML : Petri Net Markup Language) ที่เสนอใน [9] คล้ายเอกซ์เอ็มแอล (XML : Extensible Markup Language) พีเอ็นเอ็มแอลเป็นรูปแบบที่ใช้แสดงส่วนประกอบของเพทริเน็ตในเชิงอักษร ซึ่งโดยทั่วไปเพทริเน็ตนิยมแสดงในเชิงรูปภาพ



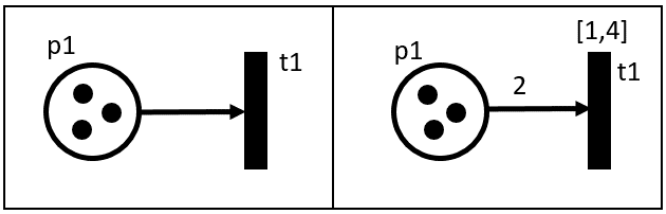
รูปที่ 2-5 ตัวอย่างเพทริเน็ตและพีเอ็นเอ็มแอล

ตัวอย่างสำหรับเพทริเน็ตแสดงดังรูปที่ 2-5 มีส่วนประกอบโดยเรียงลำดับตามบรรทัดดังนี้
บรรทัดที่ 1 คือ อินพุตเพลส p1
บรรทัดที่ 2 คือ อาร์ค a1 เชื่อมเส้นทางจากเพลส p1 ไปยังทรานซิชัน t1
บรรทัดที่ 3 คือ ทรานซิชัน t1
บรรทัดที่ 4 คือ อาร์ค a2 เชื่อมเส้นทางจากทรานซิชัน t1 ไปยังเอาท์พุตเพลส p2
บรรทัดที่ 5 คือ เอาท์พุตเพลส p2

เนื่องจากพีเอ็นเอ็มแอลแสดงโครงสร้างเพทริเน็ตทั่วไปไม่เพียงพอสำหรับโครงสร้างไทม์เพทริเน็ต ดังนั้นจึงได้ดัดแปลงพีเอ็นเอ็มแอลให้มีโครงสร้างที่เหมาะสมกับไทม์เพทริเน็ตด้วยการเพิ่มส่วนประกอบต่างๆ ได้แก่ เวลาน้อยที่สุดและมากที่สุดของทรานซิชัน น้ำหนักอินพุต น้ำหนักเอาท์พุต

รูปที่ 2-6 ทางซ้ายเป็นเพทริเน็ตทั่วไปที่มีโครงสร้าง 3 ส่วน ประกอบด้วยส่วนแรกเพลส p1 มีมาร์คกิ้งเริ่มต้นค่า 3 โทเค็น ส่วนที่สองอาร์ค a1 เชื่อมจากเพลส p1 ไปยังทรานซิชัน t1 และส่วนที่สามคือทรานซิชัน t1 ส่วนรูปทางด้านขวาเป็นไทม์เพทริเน็ตที่มีส่วนประกอบเพิ่มเติม ได้แก่

- ทรานซิชันเวลาน้อยที่สุด 1 และมากที่สุด 4 แทนด้วย [1,4]
- น้ำหนักอินพุต แทนด้วย 2 ที่เป็นตัวเลขกำกับอาร์คจากเพลส p1 ไปยังทรานซิชัน t1
- น้ำหนักเอาท์พุตที่ในรูปนี้ไม่มี



รูปที่ 2-6 เพทริเน็ตและไทม์เพทริเน็ต

ตารางที่ 2-6 ไทม์เพทริเน็ตมีส่วนเพิ่มเติม ได้แก่ ที่ทรานซิชัน t1 มีเวลาที่น้อยสุดและเวลาที่มากสุด แทนด้วยสัญลักษณ์โหนด <min> และ <max> ตามลำดับ ส่วนอาร์ค a1 มีน้ำหนักอินพุตแทนด้วยสัญลักษณ์โหนด <wi> และอาร์คที่เชื่อมจากทรานซิชันไปยังเอาท์พุตเพลสจะมีน้ำหนักเอาท์พุตเพิ่มเติมด้วย แทนด้วยสัญลักษณ์โหนด <wo>

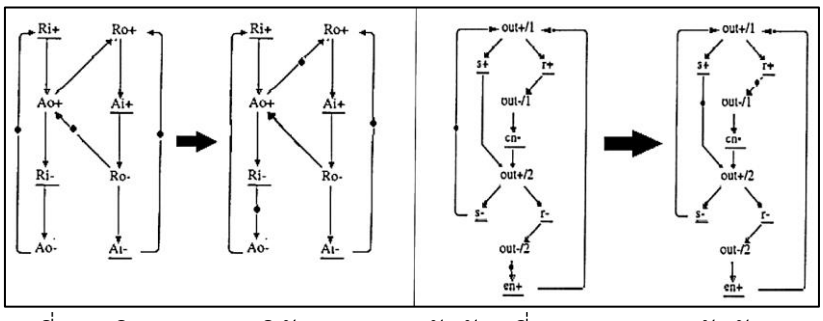
ตารางที่ 2-5 พีเอ็นเอ็มแอลและพีเอ็นเอ็มแอลที่ดัดแปลงสำหรับไทม์เพทริเน็ตจากรูปที่ 2-6

พีเอ็นเอ็มแอลดั้งเดิมสำหรับเพทริเน็ต	พีเอ็นเอ็มแอลที่ดัดแปลงสำหรับไทม์เพทริเน็ต
<place id="p1"> <initialMarking> 3 </initialMarking> </place> --- <transition id="t1"> </transition> --- <arc id="a1" source="p1" target="t1"> </arc>	<place id="p1"> <initialMarking> 3 </initialMarking> </place> --- <transition id="t1"> <min>1</min> <max>4</max> <wi>2</wi> <wo>0</wo> </transition> --- <net id="p1->t1"> <type="Sequence" source="inp1(p1,t1)" target="outp1(t1)"> </net>

2.5 งานวิจัยที่เกี่ยวข้อง

2.5.1 Formal modeling for Persistence checking of signal transition graph specification with Promela [1]

งานวิจัยเสนอวิธีการตรวจสอบคุณสมบัติวงจรฮาร์ดแวร์ด้วยวิธีแบบโมเดลเช็คกิ้ง โดยมีวิธีการแปลงแบบจำลองซิกแนลทรานซิชันกราฟเป็นแบบจำลองทางการโพรเมลาเพื่อตรวจสอบด้วยเครื่องมือสปิน ซึ่งแบบจำลองแสดงการเคลื่อนที่โทเค็นแบบไดนามิกด้วย งานวิจัยเสนอการแปลงแบบจำลองซิกแนลทรานซิชันกราฟทั้งแบบวัฏจักรเดี่ยวและแบบพหุวัฏจักร ซึ่งซิกแนลทรานซิชันกราฟแสดงดังรูปที่ 2-7 ตัวอย่างโพรเมลาแสดงรูปที่ 2-8



รูปที่ 2-7 ซิกแนลทรานซิชันกราฟแบบวัฏจักรเดี่ยวและแบบพหุวัฏจักร [1]

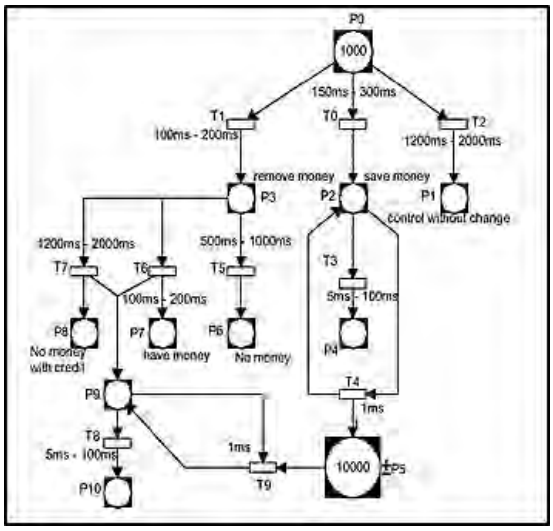
```
1  //define signal name
2  #define signal    byte
3  signal ripaop,aoprop,aoprim,rimaom,aomrip;
4  signal ropaip,aiprom,romaim,romaop,aimrop;
5
6  //define transition rule
7  #define inp1(x)      (x>0) -> x = x-1
8  #define inp2(x,y)    (x>0 && y>0) -> x = x-1; y=y-1
9  #define out1(x)      x = x+1
10 #define out2(x,y)    x = x+1; y = y+1
11
12
13 init {
14 //define initial marking
15 aomrip = 1; romaop = 1; aimrop = 1; /*initial marking*/
16
17 //define flow relation
18 do
19     :: atomic { inp1(aomrip)     -> out1(ripaop) }
20     :: atomic { inp2(ripaop,romaop) -> out2(aoprop,aoprim) }
21     :: atomic { inp1(aoprim)     -> out1(rimaom)}
22     :: atomic { inp1(rimaom)     -> out1(aomrip) }
23     :: atomic { inp2(aimrop,aoprop) -> out1(ropaip) }
24     :: atomic { inp1(ropaip)     -> out1(aiprom) }
25     :: atomic { inp1(aiprom)     -> out2(romaim,romaop)}
26     :: atomic { inp1(romaim)     -> out1(aimrop) }
27 od
28 }
```

รูปที่ 2-8 ตัวอย่างโพรเมลาจากการแปลงซิกแนลทรานซิชันกราฟ [1]

ขั้นตอนการแปลงประกอบด้วย 4 ขั้นตอน ได้แก่ ระบุชื่อสัญญาณ ระบุกฎทรานซิชัน ระบุมาร์คกิ้งตั้งต้น ระบุความสัมพันธ์เคลื่อนที่ รวมทั้งเสนอผลลัพธ์จากการใช้เครื่องมือตรวจสอบสปินด้วยแอลทีแอลเพื่อตรวจสอบคุณสมบัติความทนทาน (persistence) และคุณสมบัติความปลอดภัยของวงจร โดยระบุทรานซิชันความสัมพันธ์ของแต่ละสัญญาณสื่อถึงค่าความจริง แต่งานวิจัยนี้ยังไม่ได้เสนอแบบจำลองทางการสำหรับโครงสร้างแบบจำลองซับซ้อนมากขึ้นซึ่งเป็นระบบคอนเคอเรนท์ที่มีเงื่อนไขเวลาหรือเรียกว่า ระบบไทม์

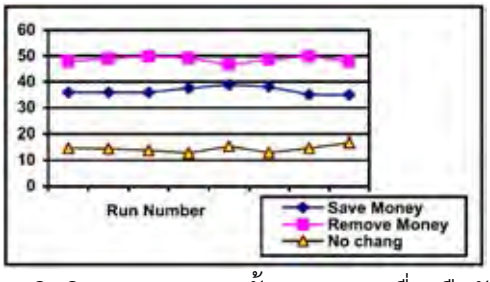
2.5.2 Modeling a Bank ATM with Two Directions Places Timed Petri Net (TPN) [4]

งานวิจัยเสนอเพทริเน็ตเป็นแบบจำลองสำหรับระบบเอทีเอ็มธนาคารที่มีความซับซ้อน มีการเพิ่มโมดูลเข้าไปในแบบจำลองระบบดั้งเดิมที่มีความซับซ้อนอยู่แล้ว คือ โมดูลเทลเลอร์ยูนิต (teller unit) ดังรูปที่ 2-9



รูปที่ 2-9 เทลเลอร์ยูนิต [4]

เสนอแบบจำลองใหม่ คือ เพทริเน็ตสองขั้ว (bidirectional Petri nets) แสดงค่าโทเค็นที่เพลสเป็นค่าบวกและค่าลบ มีการกำกับเวลาด้วย จากนั้นใช้เครื่องมือรันด้วยการจำลอง (simulation) เพื่อผลลัพธ์ (รูปที่ 2-10) แสดงประสิทธิภาพของระบบเอทีเอ็มธนาคารทั้งหมดที่มีการรวมกับโมดูลเทลเลอร์ยูนิตที่เพิ่มขึ้น



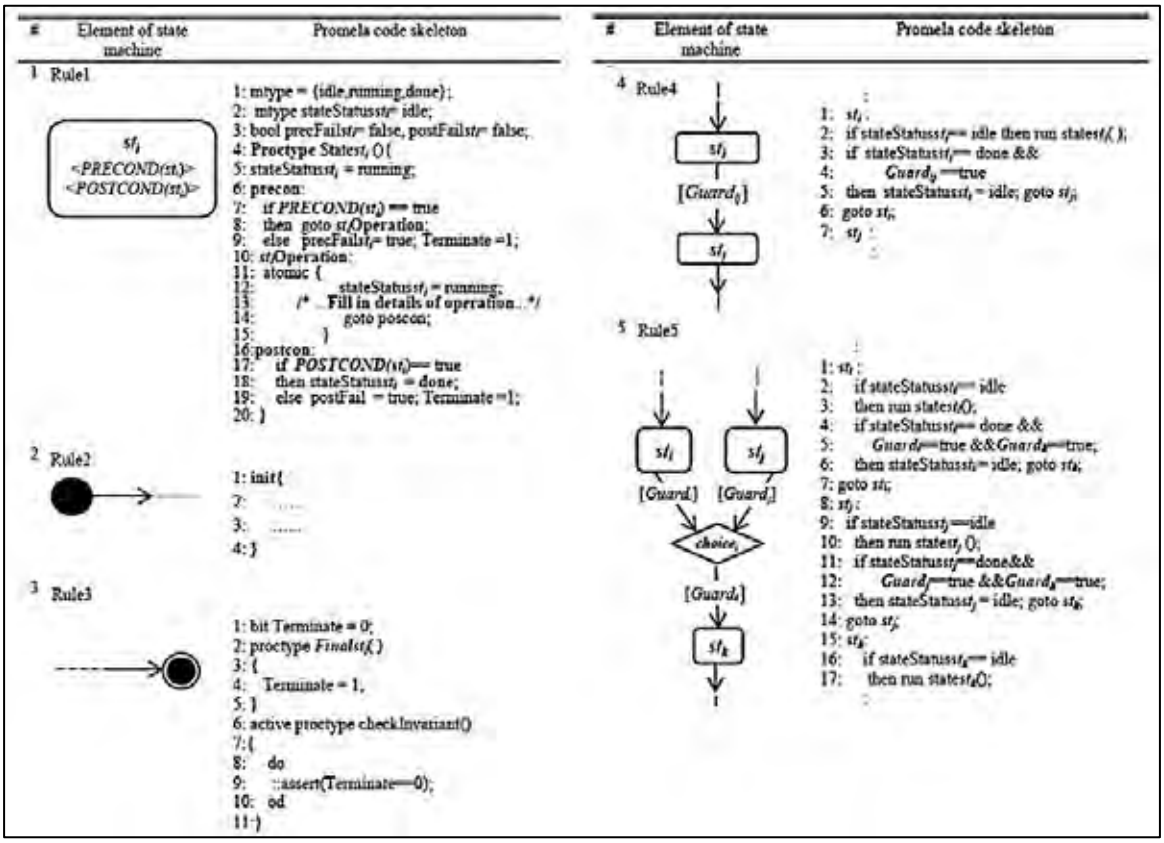
รูปที่ 2-10 ผลลัพธ์ประสิทธิภาพของระบบทั้งหมดจากเครื่องมือรันด้วยการจำลอง [4]

แต่งานวิจัยนี้ไม่มีการแสดงคุณสมบัติของระบบเพื่อการตรวจสอบโมเดลเชิงคุณภาพ (qualitative checking) เช่น คุณสมบัติความปลอดภัย คุณสมบัติความคงอยู่ ซึ่งคุณสมบัติเหล่านี้แบบจำลองของระบบที่มีความซับซ้อนควรมีการพิจารณาด้วย

2.5.3 Translating UML State Machine Diagram into Promela [5]

งานวิจัยนี้เสนอวิธีการแปลงสถานะยูเอ็มแอล (UML State Machine) เป็นโพรเมลา อีกทั้งสร้างเครื่องมือสนับสนุนการสร้างโค้ดแบบอัตโนมัติ เพื่อนำโพรเมลาไปใช้เป็นแบบจำลองสำหรับโมเดลเช็คกิงต่อไป

ก่อนทำการแปลงสถานะยูเอ็มแอลเป็นโพรเมลา ได้มีการสกัดสัญลักษณ์ต่างๆของยูเอ็มแอลเป็นรูปแบบเอกซ์เอ็มแอล จากนั้นจึงเลือกใช้คำสั่งโพรเมลาให้สอดคล้องกับโอซีแอล (OCL : Object Constraint Language) ที่ประกอบด้วยเงื่อนไขก่อน (Pre-conditions), เงื่อนไขหลัง (Post-conditions), และ ไม่แปรเปลี่ยน (Invariants) โดยเสนอขั้นตอนวิธีการแปลงจนเป็นโพรเมลา มีกฎการแปลง 5 กฎ ดังรูปที่ 2-11 และตัวอย่างของผลลัพธ์จากการแปลง ดังนั้นงานวิจัยนี้ทำให้พบแนวทางสำหรับการแปลงแผนภาพเป็นโพรเมลา



รูปที่ 2-11 กฎการแปลง 5 กฎที่แปลงสถานะยูเอ็มแอลเป็นโพรเมลา [5]

บทที่ 3

กฎการแปลงไทม์เพทริเน็ตเป็นโพรเมลา

ก่อนจะสร้างขั้นตอนการแปลงไทม์เพทริเน็ตเป็นโพรเมลานั้น จำเป็นต้องสร้างกฎการแปลงไทม์เพทริเน็ตเป็นโพรเมลาขึ้นมาก่อน อย่างไรก็ตามการจะได้มาซึ่งกฎในการแปลงนั้นจำเป็นต้องมีขั้นตอนเพื่อสร้างกฎการแปลงไทม์เพทริเน็ตเป็นโพรเมลา ซึ่งมีทั้งหมด 5 ขั้นตอนหลัก (รูปที่ 3-1) และรายละเอียดของแต่ละขั้นตอนจะกล่าวในหัวข้อถัดไป แต่ละขั้นตอนต้องปฏิบัติตามให้แล้วเสร็จเป็นลำดับขั้นไป ได้แก่

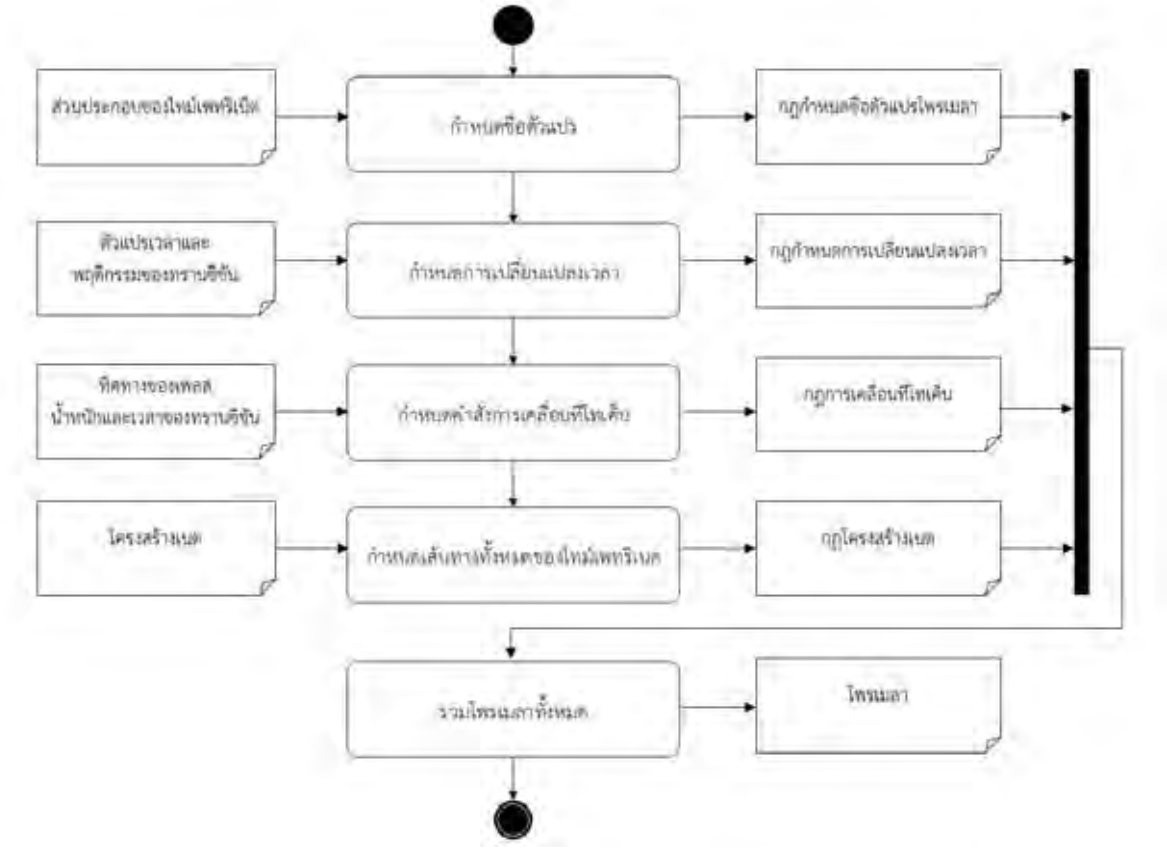
1) กำหนดชื่อตัวแปร

2) กำหนดการเปลี่ยนแปลงเวลา

3) กำหนดคำสั่งการเคลื่อนที่โทเค็น

4) กำหนดเส้นทางทั้งหมดของไทม์เพทริเน็ต

5) รวมโพรเมลาทั้งหมด

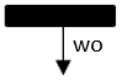


รูปที่ 3-1 แสดงขั้นตอนการแปลงไทม์เพทริเน็ตเป็นโพรเมลา

3.1 กำหนดชื่อตัวแปรโพรเมลา

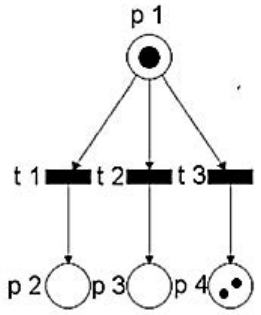
สร้างกฎกำหนดชื่อตัวแปรโพรเมลาดังแสดงในตารางที่ 3-1 โดยรูปแบบในการกำหนดตัวแปรโพรเมลาต้องเป็นจำนวนเต็มบวกเสมอ (“ชื่อตัวแปรโพรเมลา = จำนวนเต็มบวก”) ทั้งนี้ตัวอย่างการกำหนดชื่อตัวแปรโพรเมลาเป็นดังตารางที่ 3-2

ตารางที่ 3-1 กฎกำหนดชื่อตัวแปรโพรเมลา

สัญลักษณ์ของไทม์เพทริเน็ต	ส่วนประกอบของไทม์เพทริเน็ต	ชื่อตัวแปรโพรเมลา	คำอธิบายตัวแปรโพรเมลา
	เพลส	int ptoken;	จำนวนโทเค็นที่เพลสหรือมาร์คกิ้งเริ่มต้น
[min,max]	ไทม์	int timer;	เวลาที่ทรานซิชัน
[**min**,max]	ไทม์ (ขอบเวลาน้อยที่สุด)	int time_min;	ขอบเวลาน้อยที่สุดที่ทรานซิชัน
[min,**max**]	ไทม์ (ขอบเวลามากที่สุด)	int time_max;	ขอบเวลามากที่สุดที่ทรานซิชัน
	น้ำหนักอินพุต	int wi;	น้ำหนักอินพุต
	น้ำหนักเอาท์พุต	int wo;	น้ำหนักเอาท์พุต

ตารางที่ 3-2 ตัวอย่างการกำหนดชื่อตัวแปรโพรเมลา

ไทม์เพทริเน็ต	โพรเมลา

	**จำนวนโทเค็นที่เพลสแต่ละเพลส**
	int p1token = 1, p2token = 0, p3token = 0, p4token = 2;
	เวลาที่แต่ละทรานซิชัน
	int t1timer = 0, t2timer = 0, t3timer = 0;
	ขอบเวลาน้อยที่สุดและขอบเวลามากที่สุดของแต่ละทรานซิชัน
	int t1time_min=1, t1time_max=5, t2time_min =0,
	t2time_max=3, t3time_min=2, t3time_max=4;
	น้ำหนักอินพุตและน้ำหนักเอาท์พุตที่แต่ละทรานซิชัน
	int t1wi = 1, t1wo =1, t2wi = 1, t2wo =1, t3wi = 1, t3wo =1;

3.2 กำหนดการเปลี่ยนแปลงเวลา

กฎกำหนดการเปลี่ยนแปลงเวลาด้วยโพรเมลาประกอบด้วยเวลาของทั้งระบบและเวลาของแต่ละทรานซิชัน ดังนี้

<u>เวลาของทั้งระบบ</u>

#define timeCntSys(Tsys,Tlim) Tsys=Tsys+Tlim

<u>เวลาของแต่ละทรานซิชัน</u>

#define timeCntTran(Tlim,T1,T2,T3,...Tn) Tx1=Tx1+Tlim;Tx2=Tx2+Tlim;Tx3=Tx3+Tlim;... Txn=Txn+Tlim; โดยที่ n เป็นจำนวนนับ

คำอธิบายกฎ คือ

- โพรเมลา #define timeCntSys(Tsys,Tlim) แสดง เวลาของทั้งระบบตั้งแต่เริ่มต้นจนถึงปัจจุบัน โดยจะมีค่าเปลี่ยนแปลงไปตามขอบเวลาที่น้อยหรือมากที่สุดของทรานซิชันที่มีสถานะเปิดใช้งาน

- โพรเมลา #define timeCntTran(Tlim,T1,T2,T3,...Tn) แสดง เวลาของทุกทรานซิชันที่มีสถานะปิดใช้งานและยังไม่มีการเคลื่อนที่โทเค็น ซึ่งแต่ละทรานซิชันจะมีตัวแปร timer เป็นของตนเอง และจะนับเพิ่มไปตามเวลาของทรานซิชันหลังเคลื่อนย้ายโทเค็น ซึ่ง n คือจำนวนทรานซิชันทั้งหมดของไทม์เพทริเน็ต

กฎกำหนดการเปลี่ยนแปลงเวลาด้วยโพรเมลาจะมาจากพฤติกรรมเวลา (time behavior) ของทรานซิชันไทม์เพทริเน็ตซึ่งแบ่งเป็น 3 กรณี ได้แก่ ทรานซิชันที่มีสถานะปิดใช้งาน (ตารางที่ 3-3) ทรานซิชันที่มีสถานะเปิดใช้งาน (ตารางที่ 3-4) และ ทุกทรานซิชันมีสถานะสแตนด์บาย (standby transition) (ตารางที่ 3-5) โดยแต่ละสถานะของทรานซิชันแสดงพฤติกรรมของตัวแปรแตกต่างกัน

ตารางที่ 3-3 ตัวแปรและพฤติกรรมของทรานซิชันที่มีสถานะปิดใช้งาน

ตัวแปร	พฤติกรรม
Input place	ไม่มีการเคลื่อนที่โทเค็น
Output place	ไม่มีการเคลื่อนที่โทเค็น
Time transition	Time transition = Time transition + [a,b] โดยที่ a คือเวลาที่น้อยที่สุดและ b คือเวลาที่มากที่สุด
Time elapse	Time elapse = Time elapse + [a,b] โดยที่ a คือเวลาที่น้อยที่สุดและ b คือเวลาที่มากที่สุด

ตารางที่ 3-4 ตัวแปรและพฤติกรรมของทรานซิชันที่มีสถานะเปิดใช้งาน

ตัวแปร	พฤติกรรม
Input place	ถ้า ∀(TokenInputPlace >= wi) แล้ว ∀(TokenInputPlace - wi)
Output place	∀(TokenOutputPlace + wo)
Time transition	Time transition = 0
Firing time	Firing time = [a,b] โดยที่ a คือเวลาที่น้อยที่สุดและ b คือเวลาที่มากที่สุด

ตารางที่ 3-5 ตัวแปรและพฤติกรรมของทรานซิชันที่มีสถานะสแตนด์บาย

ตัวแปร	พฤติกรรม
Input place	ไม่มีการเคลื่อนที่โทเค็น
Output place	ไม่มีการเคลื่อนที่โทเค็น
Time transition	Time transition = Time transition + 1
Time elapse	Time elapse = Time elapse + 1

3.3 กำหนดคำสั่งการเคลื่อนที่โทเค็น

ในการศึกษานี้อ้างอิงกฎการกำหนดคำสั่งการเคลื่อนที่โทเค็นมาจากคณุตม์ และคณะ [1] แต่ได้มีการเพิ่มเติมตัวแปรที่ส่งผลต่อกฎ คือ น้ำหนักและเวลาของทรานซิชัน ทำให้การศึกษานี้มีตัวแปรและรายละเอียดของกฎกำหนดคำสั่งโพรเมลาแสดงการเคลื่อนที่โทเค็น ดังนี้คือ

x แทน จำนวนโทเค็นใด ๆ

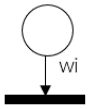
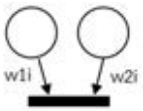
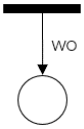
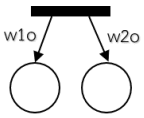
w แทน น้ำหนักซึ่งส่งผลให้โทเค็นเคลื่อนที่ และจะสัมพันธ์กับพฤติกรรมของทรานซิชันที่มีสถานะเปิดใช้งานเท่านั้น โดยแบ่งออกเป็น น้ำหนักอินพุต (wi) และน้ำหนักเอาท์พุต (wo)

T แทน เวลาของแต่ละทรานซิชัน (timer)

Tlim แทน ขอบเวลาที่น้อยที่สุดหรือขอบเวลาที่มากที่สุดของแต่ละทรานซิชัน

งานวิจัยนี้กำหนดกฎคำสั่งการเคลื่อนที่โทเค็น ซึ่งพิจารณาตามจำนวนและทิศทางของอินพุต-เพลสและเอาท์พุตเพลส ดังแสดงในตารางที่ 3-6

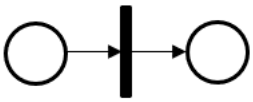
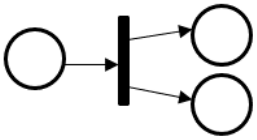
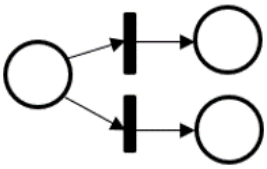
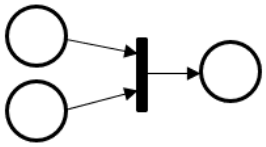
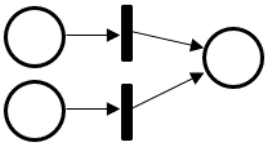
ตารางที่ 3-6 กฎการเคลื่อนที่โทเค็น

รูปแบบทิศทาง	โพรเมลา
อินพุตเพลส 1 ทิศทาง 	#define inp1(wi,x) (x > wi-1) -> x=x-wi; inp1 คือ มีอาร์ค 1 ทิศทาง
อินพุตเพลสหลายทิศทาง 	#define inpn(wni,x1,..,xn) (x1>w1i-1 &&...&& xn>wni-1) -> x1=x1-w;...;xn=xn-wni; inpn คือ มีอาร์ค n ทิศทาง
เอาท์พุตเพลส 1 ทิศทาง 	#define outp1(wo,x,T) x=x+wo; T= 0 outp1 คือ มีอาร์ค 1 ทิศทาง เวลาของทรานซิชันถูกรีเซต
เอาท์พุตเพลสหลายทิศทาง 	#define outpn(wn,x1,..,xn,T) x1=x1+w1o;..;xn=xn+wno; T=0 outpn คือ มีอาร์ค n ทิศทาง เวลาของทรานซิชันถูกรีเซต

3.4 กำหนดเส้นทางทั้งหมดของไทม์เพทริเน็ต

กำหนดกฎโครงสร้างเนตในการศึกษานี้ออกเป็น 5 โครงสร้าง แต่ละโครงสร้างเนตจะประกอบไปด้วยอินพุตเพลส ทรานซิชัน และเอาท์พุตเพลส และแสดงเป็นโพรเมลา ดังปรากฏในตารางที่ 3-7 โครงสร้างเนตเหล่านี้จะถูกนำมาประกอบรวมกันเพื่อเป็นเส้นทางทั้งหมดของไทม์เพทริ-เนตต่อไป ทั้งนี้ตัวอย่างเงื่อนไขของโครงสร้างเนตจะแสดงในหัวข้อแบบจำลองไทม์เพทริเน็ต

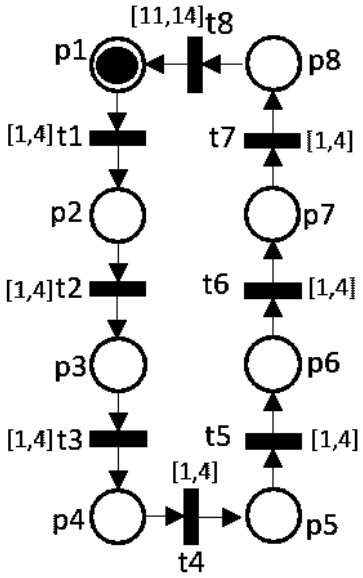
ตารางที่ 3-7 กฎโครงสร้างเนต

โครงสร้างเนต	โพรเมลา
โครงสร้างอนุกรม (sequence) 	if :: เงื่อนไข atomic{ inp1() -> outp1() } fi;
โครงสร้างคอนเคอเรนท์ 	if :: เงื่อนไข atomic{ inp1() -> outpn() } fi;
โครงสร้างเนตทางเลือกหรือ นอนดีเทอร์มินิสติค 	if :: เงื่อนไขของทางเลือกที่ 1 atomic{ inp1() -> outp1() } :: เงื่อนไขของทางเลือกที่ n atomic{ inp1() -> outp1() } fi;
โครงสร้างและ (AND) 	if :: ทุกทรานซิชันเปิดใช้งานและโทเค็นที่ทุกอินพุตเพลสมากกว่าน้ำหนักอินพุตอาร์คทุกเส้น atomic{ inpn() -> outp1() } fi;
โครงสร้างหรือ (OR) 	if :: เงื่อนไขที่ 1 atomic{ inp1() -> outp1() } fi; if :: เงื่อนไขที่ n atomic{ inp1() -> outp1() } fi;

3.5 ตัวอย่างแบบจำลองไทม์เพทริเน็ต

3.5.1 ตัวอย่างแบบจำลองไทม์เพทริเน็ตที่มีโครงสร้างอนุกรม

ตัวอย่างไทม์เพทริเน็ตที่แปลงเป็นโพรเมลา มีส่วนประกอบ 8 เพลส 8 ทรานซิชัน ขอบเวลาของแต่ละทรานซิชัน น้ำหนักอินพุต น้ำหนักเอาท์พุต โครงสร้างเนตแบบอนุกรม (รูปที่ 3-2)



รูปที่ 3-2 ตัวอย่างไทม์เพทริเน็ตที่มีโครงสร้างอนุกรม

ไทม์เพทริเน็ตในรูปที่ 3-2 จะถูกเขียนให้เป็นข้อมูลนำเข้าพีเอ็นเอ็มแอล ซึ่งประกอบด้วยข้อมูล 2 ส่วน ได้แก่ 1) การประกาศตัวแปรไทม์เพทริเน็ต และ 2) ส่วนโครงสร้างเนตที่เป็นโครงสร้างอนุกรม ดังแสดงข้างล่างนี้

1) การประกาศตัวแปรไทม์เพทริเน็ต

```
<place id="p1"><initialMarking>1</initialMarking></place>
<place id="p2"><initialMarking>0</initialMarking></place>
<place id="p3"><initialMarking>0</initialMarking></place>
<place id="p4"><initialMarking>0</initialMarking></place>
<place id="p5"><initialMarking>0</initialMarking></place>
<place id="p6"><initialMarking>0</initialMarking></place>
<place id="p7"><initialMarking>0</initialMarking></place>
<place id="p8"><initialMarking>0</initialMarking></place>
```

รูปที่ 3-3 พีเอ็นเอ็มแอลส่วนการประกาศตัวแปรของไทม์เพทริเน็ตในรูปที่ 3-2

```
<transitionid="t1">
        <min>1</min>
        <max>4</max>
        <wi>1</wi>
        <wo>1</wo>
</transition>
<transitionid="t2">
        <min>1</min>
        <max>4</max>
        <wi>1</wi>
        <wo>1</wo>
</transition>
<transitionid="t3">
        <min>1</min>
        <max>4</max>
        <wi>1</wi>
        <wo>1</wo>
</transition>
<transitionid="t4">
        <min>1</min>
        <max>4</max>
        <wi>1</wi>
        <wo>1</wo>
</transition>
<transitionid="t5">
        <min>1</min>
        <max>4</max>
        <wi>1</wi>
        <wo>1</wo>
</transition>
<transitionid="t6">
        <min>1</min>
        <max>4</max>
        <wi>1</wi>
        <wo>1</wo>
</transition>
<transitionid="t7">
        <min>1</min>
        <max>4</max>
        <wi>1</wi>
        <wo>1</wo>
</transition>
<transitionid="t8">
        <min>11</min>
        <max>14</max>
        <wi>1</wi>
        <wo>1</wo>
</transition>
```

รูปที่ 3-4 พีเอ็นเอ็มแอลส่วนการประกาศตัวแปรของไทม์เพทริเน็ตในรูปที่ 3-2 (ต่อ)

2) ส่วนโครงสร้างเนตที่เป็นโครงสร้างอนุกรม

```
<net id="p1->t1->p2" type="Series" source="inp1(p1,t1)" target="outp1(p2)">
</net>
<net id="p2->t2->p3" type="Series" source="inp1(p2,t2)" target="outp1(p3)">
</net>
<net id="p3->t3->p4" type="Series" source="inp1(p3,t1)" target="outp1(p4)">
</net>
<net id="p4->t4->p5" type="Series" source="inp1(p4,t4)" target="outp1(p5)">
</net>
<net id="p5->t5->p6" type="Series" source="inp1(p5,t5)" target="outp1(p6)">
</net>
<net id="p6->t6->p7" type="Series" source="inp1(p6,t6)" target="outp1(p7)">
</net>
<net id="p7->t7->p8" type="Series" source="inp1(p7,t7)" target="outp1(p8)">
</net>
<net id="p8->t8->p1" type="Series" source="inp1(p8,t8)" target="outp1(p1)">
</net>
```

รูปที่ 3-5 พีเอ็นเอ็มแอลส่วนโครงสร้างเนตของไทม์เพทริเน็ตในรูปที่ 3-2 (ต่อ)

ต่อมานำพีเอ็นเอ็มแอลมาแปลงเป็นโพรเมลา ทำการกำหนดชื่อตัวแปรโพรเมลาด้วยการประกาศตัวแปร กำหนดกฎการเปลี่ยนแปลงเวลาที่ประกอบด้วยเวลา จากนั้นกำหนดคำสั่งโพรเมลาแสดงการเคลื่อนที่โทเค็น และเส้นทางทั้งหมดของโครงสร้างไทม์เพทริเน็ตตามโครงสร้างเนต ซึ่งสุดท้ายจะได้โพรเมลาดังแสดงข้างล่างนี้

```
การประกาศตัวแปร
int p1token=1,p2token=0,p3token=0,p4token=0,p5token=0,p6token=0,p7token=0,p8token=0;
int t1time_min=1,t2time_min=1,t3time_min=1,t4time_min=1,t5time_min=1,t6time_min=1,t7time_min=1,t8time_min=11,
t1time_max=4,t2time_max=4,t3time_max=4,t4time_max=4,t5time_max=4,t6time_max=4,t7time_max=4,t8time_max=14;
int t1wi=1,t2wi=1,t3wi=1,t4wi=1,t5wi=1,t6wi=1,t7wi=1,t8wi=1;
int t1wo=1,t2wo=10,t3wo=1,t4wo=1,t5wo=1,t6wo=1,t7wo=1,t8wo=1;
int time_elapsed=1;
int t1timer = t1time_min,t2timer = t2time_min,t3timer =t3time_min,t4timer =t4time_min,t5timer = t5time_min,t6timer =
t6time_min,t7timer =t7time_min,t8timer =t8time_min;
กฎการเปลี่ยนแปลงเวลา
#define timeCntSys(Tsys,Tlim)   Tsys= Tsys+Tlim /*time_elapsed*/
#define timeCntTran(Tlim,Tx1,Tx2,Tx3,Tx4,Tx5,Tx6,Tx7)  Tx1=Tx1+Tlim;  Tx2=Tx2+Tlim;  Tx3=Tx3+Tlim;  Tx4=Tx4+Tlim;
Tx5=Tx5+Tlim;  Tx6=Tx6+Tlim;  Tx7=Tx7+Tlim;
```

รูปที่ 3-6 โพรเมลาของไทม์เพทริเน็ตในรูปที่ 3-2

```
กฎการเคลื่อนที่โทเคิน
#define inp1(wi,x,T,Tlim)                    (x > wi-1) -> x=x-wi
#define inp2(wi1,wi2,x1,x2,T,Tlim)           (x1>wi1-1 && x2>wi2-1) -> x1=x1-wi1
#define outp1(wo,x,T)                        x=x+wo;T=0;
#define outp2(wo1,wo2,x1,x2,T)               x=x1+wo1;x2=x2+wo2;T=0;
เส้นทางทั้งหมดของโครงสร้างไทม์เพทริเน็ต
init{
do
::if
::time_elapsed>t1time_min-1&&p1token>0->
atomic{timeCntSys(time_elapsed,t1time_min);timeCntTran(t1time_min,t2timer,t3timer,t4timer,t5timer,t6timer,t7timer,t8ti
mer);inp1(t1wi,p1token,t1timer,t1time_min);outp1(t1wo,p2token,t1timer)}
::time_elapsed<t1time_max+1&&p1token>0->
atomic{timeCntSys(time_elapsed,t1time_max);timeCntTran(t1time_max,t2timer,t3timer,t4timer,t5timer,t6timer,t7timer,t8ti
mer);inp1(t1wi,p1token,t1timer,t1time_min);outp1(t1wo,p2token,t1timer)}
fi;
::if
::time_elapsed>t2time_min-1&&p2token>0->
atomic{timeCntSys(time_elapsed,t2time_min);timeCntTran(t2time_min,t1timer,t3timer,t4timer,t5timer,t6timer,t7timer,t8ti
mer);inp1(t2wi,p2token,t2timer,t2time_min);outp1(t2wo,p3token,t2timer)}
::time_elapsed<t2time_max+1&&p2token>0->
atomic{timeCntSys(time_elapsed,t2time_max);timeCntTran(t2time_max,t1timer,t3timer,t4timer,t5timer,t6timer,t7timer,t8ti
mer);inp1(t2wi,p2token,t2timer,t2time_min);outp1(t2wo,p3token,t2timer)}
fi;
::if
::time_elapsed>t3time_min-1&&p3token>0->
atomic{timeCntSys(time_elapsed,t3time_min);timeCntTran(t3time_min,t1timer,t2timer,t4timer,t5timer,t6timer,t7timer,t8ti
mer);inp1(t3wi,p3token,t3timer,t3time_min);outp1(t3wo,p4token,t3timer)}
::time_elapsed<t3time_max+1&&p3token>0->
atomic{timeCntSys(time_elapsed,t3time_max);timeCntTran(t3time_max,t1timer,t2timer,t4timer,t5timer,t6timer,t7timer,t8ti
mer);inp1(t3wi,p3token,t3timer,t3time_min);outp1(t3wo,p4token,t3timer)}
fi;
::if
::time_elapsed>t4time_min-1&&p4token>0->
atomic{timeCntSys(time_elapsed,t4time_min);timeCntTran(t4time_min,t1timer,t2timer,t3timer,t5timer,t6timer,t7timer,t8ti
mer);inp1(t4wi,p4token,t4timer,t4time_min);outp1(t4wo,p5token,t4timer)}
::time_elapsed<t4time_max+1&&p4token>0->
atomic{timeCntSys(time_elapsed,t4time_max);timeCntTran(t4time_max,t1timer,t2timer,t3timer,t5timer,t6timer,t7timer,t8ti
mer);inp1(t4wi,p4token,t4timer,t4time_min);outp1(t4wo,p5token,t4timer)}
fi;
```

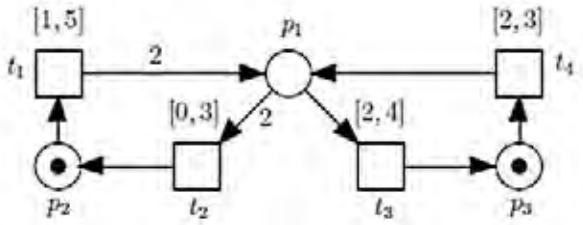
รูปที่ 3-7 โพรเมลาของไทม์เพทริเน็ตในรูปที่ 3-2 (ต่อ)

```
::if
::time_elapsed>t5time_min-1&&p5token>0->
atomic{timeCntSys(time_elapsed,t5time_min);timeCntTran(t5time_min,t1timer,t2timer,t3timer,t4timer,t6timer,t7timer,t8ti
mer);inp1(t5wi,p5token,t5timer,t5time_min);outp1(t5wo,p6token,t5timer)}
::time_elapsed<t5time_max+1&&p5token>0->
atomic{timeCntSys(time_elapsed,t5time_max);timeCntTran(t5time_max,t1timer,t2timer,t3timer,t4timer,t6timer,t7timer,t8ti
mer);inp1(t5wi,p5token,t5timer,t5time_min);outp1(t5wo,p6token,t5timer)}
fi;
::if
::time_elapsed>t6time_min-1&&p6token>0->
atomic{timeCntSys(time_elapsed,t6time_min);timeCntTran(t6time_min,t1timer,t2timer,t3timer,t4timer,t5timer,t7timer,t8ti
mer);inp1(t6wi,p6token,t6timer,t6time_min);outp1(t6wo,p7token,t6timer)}
::time_elapsed<t6time_max+1&&p6token>0->
atomic{timeCntSys(time_elapsed,t6time_max);timeCntTran(t6time_max,t1timer,t2timer,t3timer,t4timer,t5timer,t7timer,t8ti
mer);inp1(t6wi,p6token,t6timer,t6time_min);outp1(t6wo,p7token,t6timer)}
fi;
::if
::time_elapsed>t7time_min-1&&p7token>0->
atomic{timeCntSys(time_elapsed,t7time_min);timeCntTran(t7time_min,t1timer,t2timer,t3timer,t4timer,t5timer,t6timer,t8ti
mer);inp1(t7wi,p7token,t7timer,t7time_min);outp1(t7wo,p8token,t7timer)}
::time_elapsed<t7time_max+1&&p7token>0->
atomic{timeCntSys(time_elapsed,t7time_max);timeCntTran(t7time_max,t1timer,t2timer,t3timer,t4timer,t5timer,t6timer,t8ti
mer);inp1(t7wi,p7token,t7timer,t7time_min);outp1(t7wo,p8token,t7timer)}
fi;
::if
::time_elapsed>t8time_min-1&&p8token>0->
atomic{timeCntSys(time_elapsed,t8time_min);timeCntTran(t8time_min,t1timer,t2timer,t3timer,t4timer,t5timer,t6timer,t7ti
mer);inp1(t8wi,p8token,t8timer,t8time_min);outp1(t8wo,p1token,t8timer)}
::time_elapsed<t8time_max+1&&p8token>0->
atomic{timeCntSys(time_elapsed,t8time_max);timeCntTran(t8time_max,t1timer,t2timer,t3timer,t4timer,t5timer,t6timer,t7ti
mer);inp1(t8wi,p8token,t8timer,t8time_min);outp1(t8wo,p1token,t8timer)}
fi;
::else ; time_elapsed = time_elapsed+1;
od
}
```

รูปที่ 3-8 โพรเมลาของไทม์เพทริเน็ตในรูปที่ 3-2 (ต่อ)

3.5.2 ตัวอย่างแบบจำลองไทม์เพทริเน็ตที่มีโครงสร้างเนตทางเลือกหรือนอนดีเทอร์มินิสติค

ตัวอย่างไทม์เพทริเน็ตที่แปลงเป็นโพรเมลารูปที่ 3-9 มีส่วนประกอบ 3 เพลส 4 ทรานซิชัน และขอบเวลาของแต่ละทรานซิชัน น้ำหนักอินพุต น้ำหนักเอาท์พุต โดยโครงสร้างเนตไทม์เพทริเน็ตนี้ที่เพลส p1 มีโครงสร้างนอนดีเทอร์มินิสติค



รูปที่ 3-9 ตัวอย่างไทม์เพทริเน็ตที่มีโครงสร้างเนตทางเลือกหรือนอนดีเทอร์มินิสติค [3]

ข้อมูลนำเข้าอยู่ในพีเอ็นเอ็มแอลจากไทม์เพทริเน็ต (รูปที่ 3-9) ซึ่งประกอบด้วยข้อมูล 2 ส่วน ได้แก่ 1) การประกาศตัวแปรไทม์เพทริเน็ต และ 2) ส่วนโครงสร้างเนตที่เป็นโครงสร้างเนตทางเลือกหรือนอนดีเทอร์มินิสติค ดังแสดงข้างล่างนี้

1) การประกาศตัวแปรไทม์เพทริเน็ต

<place id="p1"> <initialMarking>0</initialMarking> </place> <place id="p2"> <initialMarking>1</initialMarking> </place> <place id="p3"> <initialMarking>1</initialMarking> </place>	<transition id="t1"> <min>1</min> <max>5</max> <wi>1</wi> <wo>2</wo> </transition>	<transition id="t2"> <min>0</min> <max>3</max> <wi>2</wi> <wo>1</wo> </transition>	<transition id="t3"> <min>2</min> <max>4</max> <wi>1</wi> <wo>1</wo> </transition>	<transition id="t4"> <min>2</min> <max>3</max> <wi>1</wi> <wo>1</wo> </transition>

รูปที่ 3-10 พีเอ็นเอ็มแอลส่วนการประกาศตัวแปรของไทม์เพทริเน็ตในรูปที่ 3-9

2) ส่วนโครงสร้างเนตที่มีโครงสร้างเนตทางเลือกหรือนอนดีเทอร์มินิสติค

```
<net id="p1->t3->p3" type="Nondeterministic" source="inp1(p1,t3)" target="outp1(p3)">
</net>
<net id="p1->t2->p2" type="Nondeterministic" source="inp1(p1,t2)" target="outp1(p2)">
</net>
<net id="p2->t1->p1" type="Series" source="inp1(p2,t1)" target="outp1(p1)">
</net>
<net id="p3->t4->p1" type="Series" source="inp1(p3,t4)" target="outp1(p1)">
</net>
```

รูปที่ 3-11 พีเอ็นเอ็มแอลส่วนการประกาศตัวแปรของไทม์เพทริเน็ตในรูปที่ 3-9 (ต่อ)

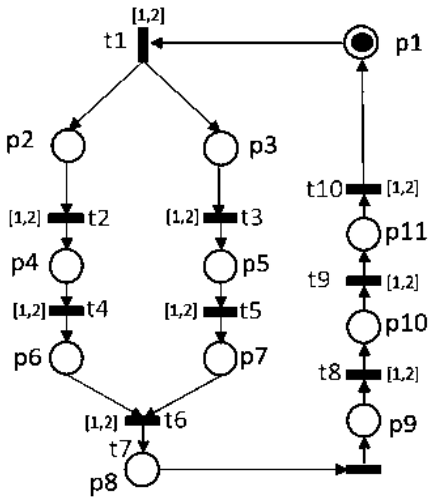
ต่อมาแปลงข้อมูลนำเข้าพีเอ็นเอ็มแอลเป็นโพรเมลา ได้โพรเมลาที่มีการประกาศตัวแปร กฎการเปลี่ยนแปลงเวลาที่ประกอบด้วยเวลาของทั้งระบบและเวลาของแต่ละทรานซิชัน กฎการเคลื่อนที่โทเค็น และสุดท้ายเส้นทางทั้งหมดของโครงสร้างไทม์เพทริเน็ตตามโครงสร้างเนต ดังนี้

```
การประกาศตัวแปร
int time_elapsed=1;
int p1token = 0,p2token = 1,p3token = 1;
int t1timer = 1,t2timer = 1,t3timer =1,t4timer =1;
int t1wi = 1,t1wo = 2,t2wi = 2,t2wo = 1,t3wi = 1,t3wo = 1,t4wi = 1,t4wo = 1;
int t1time_min = 1,t1time_max = 5,t2time_min = 0,t2time_max = 3,t3time_min = 2,t3time_max = 4,t4time_min
= 2,t4time_max = 3;
กฎการเปลี่ยนแปลงเวลา
#define timeCntSys(Tsys,Tlim)          Tsys= Tsys+Tlim
#define timeCntTran(Tlim,Tx1,Tx2,Tx3)  Tx1=Tx1+Tlim;  Tx2=Tx2+Tlim;  Tx3=Tx3+Tlim;
กฎการเคลื่อนที่ไทเส็น
#define inp1(wi,x)                     (x > wi-1) -> x=x-wi
#define inp2(wi1,wi2,x1,x2)            (x1>wi1-1 && x2>wi2-1) -> x1=x1-wi1
#define outp1(wo,x,T)                  x=x+wo; T=0
#define outp2(wo1,wo2,x1,x2,T)         x1=x1+wo1;x2=x2+wo2; T=0
เส้นทางทั้งหมดของโครงสร้างไทม์เพทริเน็ต
init{
do::
::if /*--p2 -> t1 -> p1--*/
::time_elapsed>t1time_min-1&&p2token>0->
atomic{timeCntSys(time_elapsed,t1time_min)->timeCntTran(t1time_min,t4timer,t2timer,t3timer)
->inp1(t1wi,p2token,t1timer,t1time_min)->outp1(t1wo,p1token,t1timer)}
::time_elapsed<t1time_max+1&&p2token>0
->atomic{timeCntSys(time_elapsed,t1time_max)->timeCntTran(t1time_max,t4timer,t2timer,t3timer)
->inp1(t1wi,p2token,t1timer,t1time_max)->outp1(t1wo,p1token,t1timer)}
fi;
::if /*--p3 -> t4 -> p1--*/
::time_elapsed>t4time_min-1&&p3token>0->
atomic{timeCntSys(time_elapsed,t4time_min)->timeCntTran(t4time_min,t1timer,t2timer,t3timer)
->inp1(t4wi,p3token,t4timer,t4time_min)->outp1(t4wo,p1token,t4timer)}
::time_elapsed<t4time_max+1&&p3token>0->
atomic{timeCntSys(time_elapsed,t4time_max)->timeCntTran(t4time_max,t1timer,t2timer,t3timer)
->inp1(t4wi,p3token,t4timer,t4time_max)->outp1(t4wo,p1token,t4timer)}
fi;
:: if/*--p1 -> t2 -> p2 โครงสร้างเนตทางเลือกหรือนอนดีเทอร์มินิสติค --*/
::time_elapsed>t2time_min-1&&p1token>0->
atomic{timeCntSys(time_elapsed,t2time_min)->timeCntTran(t2time_min,t4timer,t1timer,t3timer)
->inp1(t2wi,p1token,t2timer,t2time_min)->outp1(t2wo,p2token,t2timer)}
::time_elapsed<t2time_max+1&&p1token>0->
atomic{timeCntSys(time_elapsed,t2time_max)->timeCntTran(t2time_max,t4timer,t1timer,t3timer)
->inp1(t2wi,p1token,t2timer,t2time_max)->outp1(t2wo,p2token,t2timer)}
/*--p1 -> t3 -> p3 โครงสร้างเนตทางเลือกหรือนอนดีเทอร์มินิสติค --*/
::time_elapsed>t3time_min-1&&p1token>0->
atomic{timeCntSys(time_elapsed,t3time_min)->timeCntTran(t3time_min,t4timer,t1timer,t2timer)
->inp1(t3wi,p1token,t3timer,t3time_min)->outp1(t3wo,p3token,t3timer)}
::time_elapsed<t3time_max+1&&p1token>0->
atomic{timeCntSys(time_elapsed,t3time_max)->
timeCntTran(t3time_max,t4timer,t1timer,t2timer)->inp1(t3wi,p1token,t3timer,t3time_max)
->outp1(t3wo,p3token,t3timer)}
fi;
::else -> time_elapsed = time_elapsed+1;
od
}
```

รูปที่ 3-12 โพรเมลาของไทม์เพทริเน็ตในรูปที่ 3-9

3.5.3 ตัวอย่างแบบจำลองไทม์เพทริเน็ตที่มีโครงสร้างคอนเคอเรนท์กับโครงสร้างและ

ตัวอย่างไทม์เพทริเน็ต (รูปที่ 3-4) ประกอบด้วย 10 เพลส 10 ทรานซิชันที่มีขอบเวลา น้ำหนักอินพุต น้ำหนักเอาท์พุต โครงสร้างเนตมีโครงสร้างคอนเคอเรนท์ที่เพลส p1 p2 p3 กับโครงสร้างที่และเพลส p6 p7 p8 โดยที่มาร์คกิ้งเริ่มเริ่มต้นอยู่ที่เพลส p1 มีค่า 1 โทเค็น



รูปที่ 3-13 ตัวอย่างไทม์เพทริเน็ตโครงสร้างคอนเคอเรนท์ที่เพลส p1 p2 p3 กับโครงสร้างและที่เพลส p6 p7 p8

ข้อมูลนำเข้าอยู่ในพีเอ็นเอ็มแอลจากไทม์เพทริเน็ต (รูปที่ 3-4) ซึ่งประกอบด้วยข้อมูล 2 ส่วน ได้แก่ 1) การประกาศตัวแปรไทม์เพทริเน็ต และ 2) ส่วนโครงสร้างเนตที่เป็นโครงสร้างคอนเคอเรนท์ที่เพลส p1 p2 p3 กับโครงสร้างที่และเพลส p6 p7 p8 ดังแสดงข้างล่างนี้

1) การประกาศตัวแปรไทม์เพทริเน็ต

```
<place id="p1"><initialMarking>1</initialMarking></place>
<place id="p2"><initialMarking>0</initialMarking></place>
<place id="p3"><initialMarking>0</initialMarking></place>
<place id="p4"><initialMarking>0</initialMarking></place>
<place id="p5"><initialMarking>0</initialMarking></place>
<place id="p6"><initialMarking>0</initialMarking></place>
<place id="p7"><initialMarking>0</initialMarking></place>
<place id="p8"><initialMarking>0</initialMarking></place>
<place id="p9"><initialMarking>0</initialMarking></place>
<place id="p10"><initialMarking>0</initialMarking></place>
<place id="p11"><initialMarking>0</initialMarking></place>
```

รูปที่ 3-14 พีเอ็นเอ็มแอลส่วนการประกาศตัวแปรของไทม์เพทริเน็ตในรูปที่ 3-13

```
<transition id="t1">
<min>1</min><max>2</max>
<wi>1</wi><wo>1</wo>
</transition>
<transition id="t2">
<min>1</min><max>2</max>
<wi>1</wi><wo>1</wo>
</transition>
<transition id="t3">
<min>1</min><max>2</max>
<wi>1</wi><wo>1</wo>
</transition>
<transition id="t4">
<min>1</min><max>2</max>
<wi>1</wi><wo>1</wo>
</transition>
<transition id="t5">
<min>1</min><max>2</max>
<wi>1</wi><wo>1</wo>
</transition>
```

```
<transition id="t6">
<min>1</min><max>2</max>
<wi>1</wi><wo>1</wo>
</transition>
<transition id="t7">
<min>1</min><max>2</max>
<wi>1</wi><wo>1</wo>
</transition>
<transition id="t8">
<min>1</min><max>2</max>
<wi>1</wi><wo>1</wo>
</transition>
<transition id="t9">
<min>10</min><max>15</max>
<wi>1</wi><wo>1</wo>
</transition>
<transition id="t10">
<min>1</min><max>2</max>
<wi>1</wi><wo>1</wo>
</transition>
```

รูปที่ 3-15 พีเอ็นเอ็มแอลส่วนการประกาศตัวแปรของไทม์เพทริเน็ตในรูปที่ 3-13 (ต่อ)

2) ส่วนโครงสร้างเนตที่มีโครงสร้างคอนเคอเรนทกับโครงสร้างที่และ

```
<net id="p1->t1->p2,p3" type="Concurrent" source="inp1(p1,t1)" target="outp2(p2,p3)">
</net>
<net id="p2->t2->p4" type="Series" source="inp1(p2,t2)" target="outp1(p4)">
</net>
<net id="p4->t4->p6" type="Series" source="inp1(p4,t4)" target="outp1(p6)">
</net>
<net id="p3->t3->p5" type="Series" source="inp1(p3,t3)" target="outp1(p5)">
</net>
<net id="p5->t5->p7" type="Series" source="inp1(p5,t5)" target="outp1(p7)">
</net>
<net id="p6,p7->t6->p8" type="And" source="inp2(p6,p7,t6)" target="outp1(p8)">
</net>
<net id="p8->t7->p9" type="Series" source="inp1(p8,t7)" target="outp1(p9)">
</net>
<net id="p9->t8->p10" type="Series" source="inp1(p9,t8)" target="outp1(p10)">
</net>
<net id="p10->t9->p11" type="Series" source="inp1(p10,t9)" target="outp1(p11)">
</net>
<net id="p11->t10->p1" type="Series" source="inp1(p11,t10)" target="outp1(p1)">
</net>
```

รูปที่ 3-16 พีเอ็นเอ็มแอลส่วนโครงสร้างเนตของไทม์เพทริเน็ตในรูปที่ 3-13 (ต่อ)

จากนั้นแปลงพีเอ็นเอ็มแอลเป็นโพรเมลา ซึ่งโพรเมลามีการประกาศตัวแปร กฎการเปลี่ยนแปลงเวลา กฎการเคลื่อนที่โทเค็น และสุดท้ายเส้นทางทั้งหมดของโครงสร้างไทม์เพทริเน็ตตามโครงสร้างเนตสร้างเนตที่มีโครงสร้างคอนเคอเรนทกับโครงสร้างที่และ ดังนี้

การประกาศตัวแปร

```
int p1token=1, p2token=0, p3token=0, p4token=0, p5token=0, p6token=0, p7token=0, p8token=0, p9token=0, p10token=0, p11token=0;
int t1time_min=1, t2time_min=1, t3time_min=1, t4time_min=1, t5time_min=1, t6time_min=1, t7time_min=1, t8time_min=1, t9time_min=1,
t10time_min=1;
int t1time_max=2, t2time_max=2, t3time_max=2, t4time_max=2, t5time_max=2, t6time_max=2, t7time_max=2, t8time_max=2,
t9time_max=2, t10time_max=2;
int t1wi=1,t2wi=1,t3wi=1,t4wi=1,t5wi=1,t6wi1=1,t6wi2=1,t7wi=1,t8wi=1,t9wi=1,t10wi=1;
int t1wo1=1,t1wo2=1,t2wo=1,t3wo=1,t4wo=1,t5wo=1,t6wo=1,t7wo=1,t8wo=1,t9wo=1,t10wo=1;
int time_elapsed=1;
int t1timer = t1time_min,t2timer = t2time_min,t3timer =t3time_min,t4timer =t4time_min,t5timer = t5time_min,t6timer = t6time_min,
t7timer =t7time_min,t8timer =t8time_min,t9timer =t9time_min,t10timer =t10time_min;
```

กฎการเปลี่ยนแปลงเวลา

```
#define    timeCntSys(Tsys,Tlim)                                    Tsys= Tsys+Tlim /*time_elapsed*/
#define    timeCntTran(Tlim,Tx1,Tx2,Tx3,Tx4,Tx5,Tx6,Tx7,Tx8,Tx9)    Tx1=Tx1+Tlim;    Tx2=Tx2+Tlim;    Tx3=Tx3+Tlim;
Tx4=Tx4+Tlim;  Tx5=Tx5+Tlim;  Tx6=Tx6+Tlim;  Tx7=Tx7+Tlim;  Tx8=Tx8+Tlim;  Tx9=Tx9+Tlim;
```

กฎการเคลื่อนที่โทเค็น

```
#define inp1(wi,x,T,Tlim)                (x > wi-1) -> x=x-wi
#define inp2(wi1,wi2,x1,x2,T,Tlim)       (x1>wi1-1 && x2>wi2-1) -> x1=x1-wi1;x2=x2-wi2
#define outp1(wo,x,T)                    x=x+wo;T=0;
#define outp2(wo1,wo2,x1,x2,T)           x1=x1+wo1;x2=x2+wo2;T=0;
```

เส้นทางทั้งหมดของโครงสร้างไทม์เพทริเน็ต

```
init{
do
::if
::time_elapsed>t1time_min-1&&p1token>0->
atomic{timeCntSys(time_elapsed,t1time_min);timeCntTran(t1time_min,t2timer,t3timer,t4timer,t5timer,t6timer,t7timer,t8ti
mer,t9timer,t10timer);inp1(t1wi,p1token,t1timer,t1time_min);outp2(t1wo1,t1wo2,p3token,p2token,t1timer)}
::time_elapsed<t1time_max+1&&p1token>0->
atomic{timeCntSys(time_elapsed,t1time_max);timeCntTran(t1time_max,t2timer,t3timer,t4timer,t5timer,t6timer,t7timer,t8ti
mer,t9timer,t10timer);inp1(t1wi,p1token,t1timer,t1time_min);outp2(t1wo1,t1wo2,p3token,p2token,t1timer)}
fi;
::if
::time_elapsed>t3time_min-1&&p3token>0->
atomic{timeCntSys(time_elapsed,t3time_min);timeCntTran(t3time_min,t1timer,t2timer,t4timer,t5timer,t6timer,t7timer,t8ti
mer,t9timer,t10timer);inp1(t3wi,p3token,t3timer,t3time_min);outp1(t3wo,p5token,t3timer)}
::time_elapsed<t3time_max+1&&p3token>0->
atomic{timeCntSys(time_elapsed,t3time_max);timeCntTran(t3time_max,t1timer,t2timer,t4timer,t5timer,t6timer,t7timer,t8ti
mer,t9timer,t10timer);inp1(t3wi,p3token,t3timer,t3time_min);outp1(t3wo,p5token,t3timer)}
fi;
::if
::time_elapsed>t5time_min-1&&p5token>0->
atomic{timeCntSys(time_elapsed,t5time_min);timeCntTran(t5time_min,t1timer,t2timer,t3timer,t4timer,t6timer,t7timer,t8ti
mer,t9timer,t10timer);inp1(t5wi,p5token,t5timer,t5time_min);outp1(t5wo,p7token,t5timer)}
::time_elapsed<t5time_max+1&&p5token>0->
```

รูปที่ 3-17 โพรเมลาของไทม์เพทริเน็ตในรูปที่ 3-13

```
atomic{timeCntSys(time_elapsed,t5time_max);timeCntTran(t5time_max,t1timer,t2timer,t3timer,t4timer,t6timer,t7timer,t8ti
mer,t9timer,t10timer);inp1(t5wi,p5token,t5timer,t5time_min);outp1(t5wo,p7token,t5timer)}
fi;
::if
::time elapsed>t2time min-1&&p2token>0->
atomic{timeCntSys(time elapsed,t2time min);timeCntTran(t2time min,t1timer,t3timer,t4timer,t5timer,t6timer,t7timer,t8ti
mer,t9timer,t10timer);inp1(t2wi,p2token,t2timer,t2time_min);outp1(t2wo,p4token,t2timer)}
::time elapsed<t2time max+1&&p2token>0->
atomic{timeCntSys(time elapsed,t2time max);timeCntTran(t2time max,t1timer,t3timer,t4timer,t5timer,t6timer,t7timer,t8ti
mer,t9timer,t10timer);inp1(t2wi,p2token,t2timer,t2time_min);outp1(t2wo,p4token,t2timer)}
fi;
::if
::time elapsed>t4time min-1&&p4token>0->
atomic{timeCntSys(time elapsed,t4time min);timeCntTran(t4time min,t1timer,t2timer,t3timer,t5timer,t6timer,t7timer,t8ti
mer,t9timer,t10timer);inp1(t4wi,p4token,t4timer,t4time_min);outp1(t4wo,p6token,t4timer)}
::time elapsed<t4time max+1&&p4token>0->
atomic{timeCntSys(time elapsed,t4time max);timeCntTran(t4time max,t1timer,t2timer,t3timer,t5timer,t6timer,t7timer,t8ti
mer,t9timer,t10timer);inp1(t4wi,p4token,t4timer,t4time_min);outp1(t4wo,p6token,t4timer)}
fi;
::if
::time elapsed>t6time min-1&&p6token>0&&p7token>0->
atomic{timeCntSys(time elapsed,t6time min);timeCntTran(t6time min,t1timer,t2timer,t3timer,t4timer,t5timer,t7timer,t8ti
mer,t9timer,t10timer);inp2(t6wi1,t6wi2,p6token,p7token,t6timer,t6time min);outp1(t6wo,p8token,t6timer)}
::time elapsed<t6time max+1&&p6token>0&&p7token>0->
atomic{timeCntSys(time elapsed,t6time max);timeCntTran(t6time max,t1timer,t2timer,t3timer,t4timer,t5timer,t7timer,t8ti
mer,t9timer,t10timer);inp2(t6wi1,t6wi2,p6token,p7token,t6timer,t6time min);outp1(t6wo,p8token,t6timer)}
fi;
::if
::time elapsed>t7time min-1&&p8token>0->
atomic{timeCntSys(time elapsed,t7time min);timeCntTran(t7time min,t1timer,t2timer,t3timer,t4timer,t5timer,t6timer,t8ti
mer,t9timer,t10timer);inp1(t7wi,p8token,t7timer,t7time_min);outp1(t7wo,p9token,t7timer)}
::time elapsed<t7time max+1&&p8token>0->
atomic{timeCntSys(time elapsed,t7time max);timeCntTran(t7time max,t1timer,t2timer,t3timer,t4timer,t5timer,t6timer,t8ti
mer,t9timer,t10timer);inp1(t7wi,p8token,t7timer,t7time_min);outp1(t7wo,p9token,t7timer)}
fi;
::if
::time elapsed>t8time min-1&&p9token>0->
atomic{timeCntSys(time elapsed,t8time min);timeCntTran(t8time min,t1timer,t2timer,t3timer,t4timer,t5timer,t6timer,t7ti
mer,t9timer,t10timer);inp1(t8wi,p9token,t8timer,t8time_min);outp1(t8wo,p10token,t8timer)}
::time elapsed<t8time max+1&&p9token>0->
atomic{timeCntSys(time elapsed,t8time max);timeCntTran(t8time max,t1timer,t2timer,t3timer,t4timer,t5timer,t6timer,t7ti
mer,t9timer,t10timer);inp1(t8wi,p9token,t8timer,t8time_min);outp1(t8wo,p10token,t8timer)}
fi;
::if
::time elapsed>t9time min-1&&p10token>0->
atomic{timeCntSys(time elapsed,t9time min);timeCntTran(t9time min,t1timer,t2timer,t3timer,t4timer,t5timer,t6timer,t7ti
mer,t8timer,t10timer);inp1(t9wi,p10token,t9timer,t9time min);outp1(t9wo,p11token,t9timer)}
::time elapsed<t9time max+1&&p10token>0->
atomic{timeCntSys(time elapsed,t9time max);timeCntTran(t9time max,t1timer,t2timer,t3timer,t4timer,t5timer,t6timer,t7ti
mer,t8timer,t10timer);inp1(t9wi,p10token,t9timer,t9time min);outp1(t9wo,p11token,t9timer)}
fi;
::if
::time elapsed>t10time min-1&&p11token>0->
atomic{timeCntSys(time elapsed,t10time min);timeCntTran(t10time min,t1timer,t2timer,t3timer,t4timer,t5timer,t6timer,t7
timer,t8timer,t9timer);inp1(t10wi,p11token,t10timer,t10time_min);outp1(t10wo,p1token,t10timer)}
::time elapsed<t10time max+1&&p11token>0->
atomic{timeCntSys(time elapsed,t10time max);timeCntTran(t10time max,t1timer,t2timer,t3timer,t4timer,t5timer,t6timer,t
7timer,t8timer,t9timer);inp1(t10wi,p11token,t10timer,t10time_min);outp1(t10wo,p1token,t10timer)}
fi;

::else -> time elapsed = time elapsed+1;
od
}
```

รูปที่ 3-18 โพรเมลาของไทม์เพทริเน็ตในรูปที่ 3-13 (ต่อ)

บทที่ 4
เครื่องมือที่พัฒนา

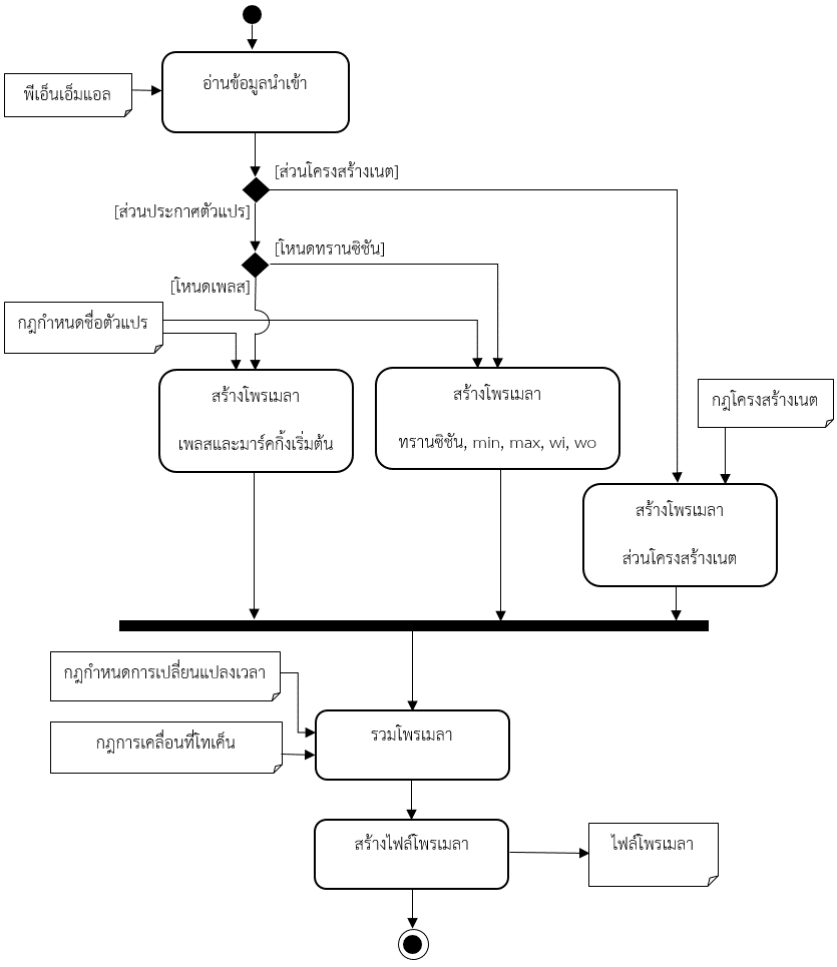
4.1 ภาพรวมของเครื่องมือสนับสนุนการแปลงไทม์เพทริเน็ตเป็นโพรเมลา

เครื่องมืออ่านข้อมูลนำเข้าพีเอ็นเอ็มแอลสำหรับไทม์แพทริเนต จากนั้นเครื่องมือทำการสร้างโพรเมลาของแต่ละส่วนประกอบจากพีเอ็นเอ็มแอล ได้แก่ ส่วนประกาศตัวแปร (ตารางที่ 4-1) และส่วนโครงสร้างเนต (ตารางที่ 4-2)

พีเอ็นเอ็มแอลส่วนประกาศตัวแปรในโหนดเพลสมีส่วนเพลสและมาร์คกิ้งเริ่มต้น ส่วนในโหนดทรานซิชันประกอบด้วย ทรานซิชัน เวลา น้ำหนักอินพุต น้ำหนักเอาท์พุต

พีเอ็นเอ็มแอลส่วนโครงสร้างเนตจะมีโครงสร้างเนตที่ประกอบด้วยเส้นทางของอินพุทเพลส ทรานซิชัน และเอาท์พุตเพลส

ต่อมาเครื่องมือรับกฎกำหนดการเปลี่ยนแปลงเวลาและกฎการเคลื่อนที่โทเค็นแล้วนำมาประกอบรวมโพรเมลาทั้งหมด และสุดท้ายสร้างโพรเมลาขึ้นมา แผนภาพดังรูปที่ 4-1



รูปที่ 4-1 แผนภาพเครื่องมือสนับสนุนการแปลงไทม์เพทริเน็ตเป็นโพรเมลา

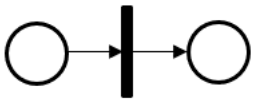
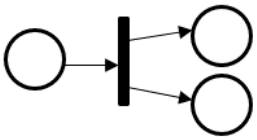
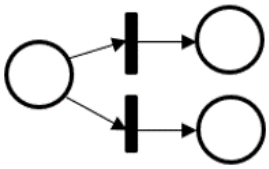
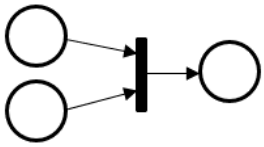
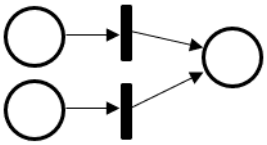
4.2 ข้อมูลนำเข้าสำหรับเครื่องมือที่พัฒนา

พีเอ็นเอ็มแอลในงานวิจัยนี้แบ่งเป็น 2 ส่วน ได้แก่ ส่วนการประกาศตัวแปรและส่วนโครงสร้างเนต โดยส่วนประกาศตัวแปรจะนำข้อมูลนำเข้าไปสร้างโพรเมลาตามรายละเอียดถัดไป ซึ่งพีเอ็นเอ็มแอลส่วนการประกาศตัวแปรเป็นดังตารางที่ 4-2 และพีเอ็นเอ็มแอลส่วนโครงสร้างเนตตัวอย่างดังตารางที่ 4-3

ตารางที่ 4-1 การประกาศตัวแปรไทม์เพทริเน็ตและพีเอ็นเอ็มแอลส่วนการประกาศตัวแปร

การประกาศตัวแปร	โหนดในพีเอ็นเอ็มแอล	พีเอ็นเอ็มแอล
เพลสและมาร์คกิ้งเริ่มต้น	โหนดเพลส	<place id="p1"> <initialMarking>0</initialMarking> </place>
ทรานซิชัน ขอบเวลาน้อยที่สุดที่ทรานซิชัน ขอบเวลามากที่สุดที่ทรานซิชัน น้ำหนักอินพุต น้ำหนักเอาท์พุต	โหนดทรานซิชัน	<transition id="t1"> <min>1</min> <max>1</max> <wi>2</wi> <wo>1</wo> </transition>

ตารางที่ 4-2 โครงสร้างเนตและพีเอ็นเอ็มแอลส่วนโครงสร้างเนต

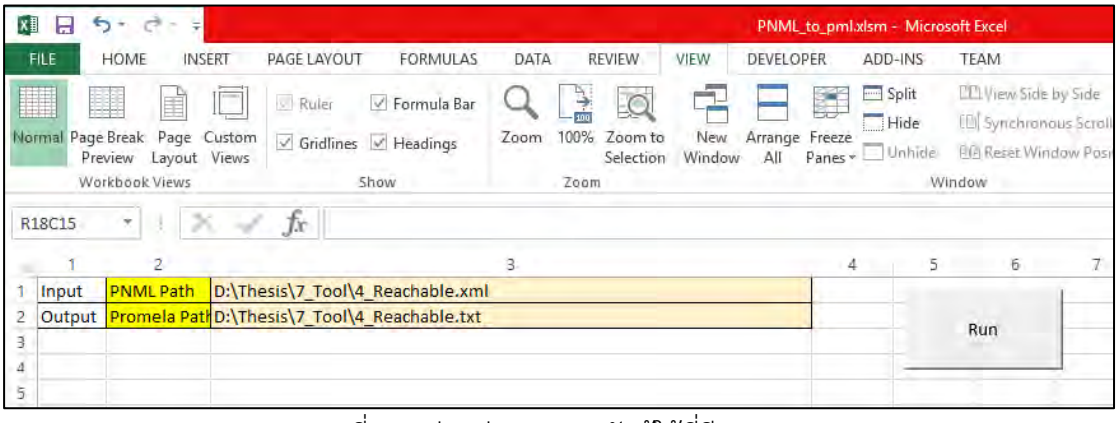
โครงสร้างเนต	พีเอ็นเอ็มแอล
โครงสร้างอนุกรม (sequence) 	<net id="px->t1->py" type="Sequence" source="inp1(px,t1)" target="outp1(py)"> </net>
โครงสร้างคอนเคอเรนท์ 	<net id="px->t1->py,pz" type="Concurrent" source="inp1(px,t1)" target="outpn(py,pz)"> </net>
โครงสร้างเนตทางเลือกหรือ นอนดีเทอร์มินิสติค 	<net id="px->t1->py" เงื่อนไขของทางเลือกที่ 1 type="Nondeterministic" source="inp1(px,t1)" target="outp1(py)"> </net> <net id="px->t2->pz"> เงื่อนไขของทางเลือกที่ n type="Nondeterministic" source="inp1(px,t2)" target="outp1(pz)"> </net>
โครงสร้างและ (AND) 	ทุกทรานซิชันเปิดใช้งานและโทเค็นที่ทุกอินพุตเพลสมากกว่าน้ำหนักอินพุตที่อาร์คทุกเส้น <net id="px,py->t1->pz"> type="And" source="inpn(px,py,t1)" target="outp1(pz)"> </net>
โครงสร้างหรือ (OR) 	โพรเมลาเหมือนโครงสร้างอนุกรมที่มี 2 เส้นทาง <net id="px->t1->py"/> เงื่อนไขของทางเลือกที่ 1 type="Or" source="inp1(px,t1)" target="outp1(py)"> </net> <net id="pz->t2->py"/> เงื่อนไขของทางเลือกที่ n type="Or" source="inp1(pz,t2)" target="outp1(py)"> </net>

4.3 ส่วนต่อประสานกับผู้ใช้

ส่วนต่อประสานกับผู้ใช้ในรูปแบบของเอ็กซ์เซล (Excel) ประกอบด้วย 3 ชีท ได้แก่ ชีท Run ชีท Read_PNML และ ชีท Promela (รูปที่ 4-2, รูปที่ 4-3, รูปที่ 4-5 ตามลำดับ) โดยที่ผู้ใช้ต้องใส่ข้อมูลอินพุตไฟล์พีเอ็นเอ็มแอลและเอาท์พุตไฟล์โพรเมลา จากนั้นจึงกดปุ่ม Run เพื่อให้เครื่องมือสร้างไฟล์ภาษาโพรเมลาออกมา

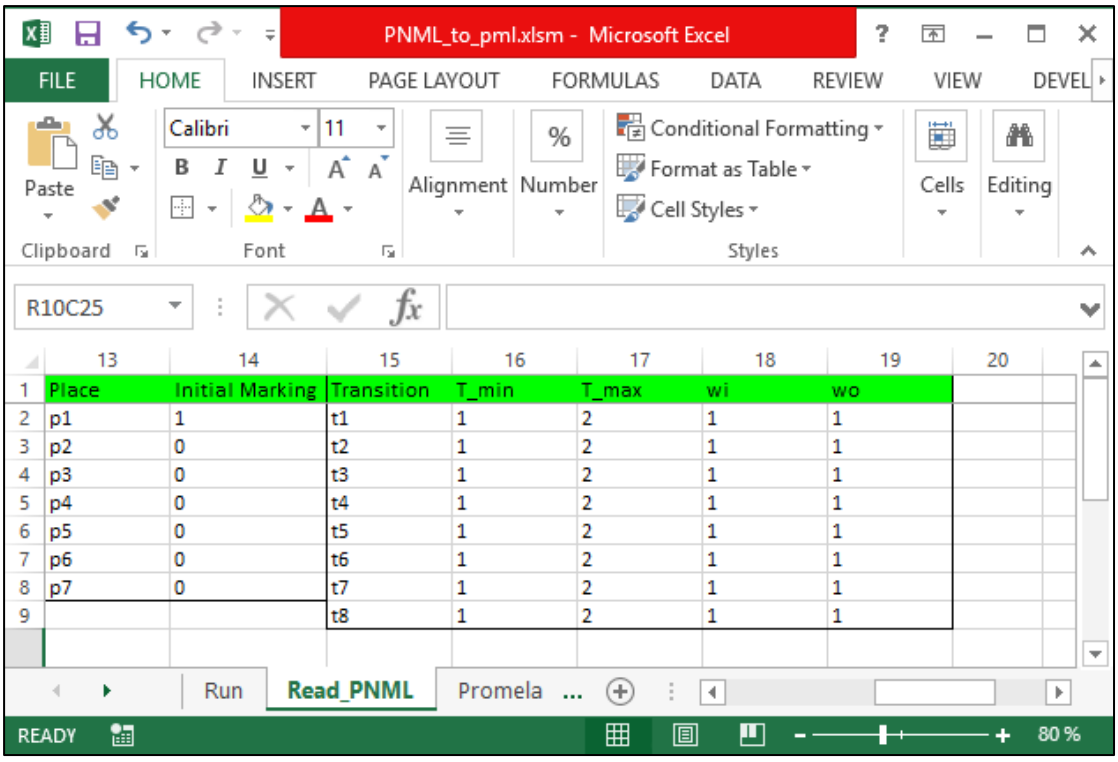
ส่วนต่อประสานกับผู้ใช้ที่ชีท Run นั้นผู้ใช้ต้องกรอกข้อมูลอินพุตไฟล์พีเอ็นเอ็มแอลและที่อยู่ของไฟล์ จากนั้นกรอกข้อมูลเอาท์พุตไฟล์โพรเมลาและที่อยู่ของไฟล์ ดังตัวอย่างดังด้านล่าง

- ข้อมูลอินพุตไฟล์พีเอ็นเอ็มแอลและที่อยู่ D:\Thesis\7_Tool\4_Reachable.xml
- ข้อมูลเอาท์พุตไฟล์โพรเมลาและที่อยู่ D:\Thesis\7_Tool\4_Reachable.txt



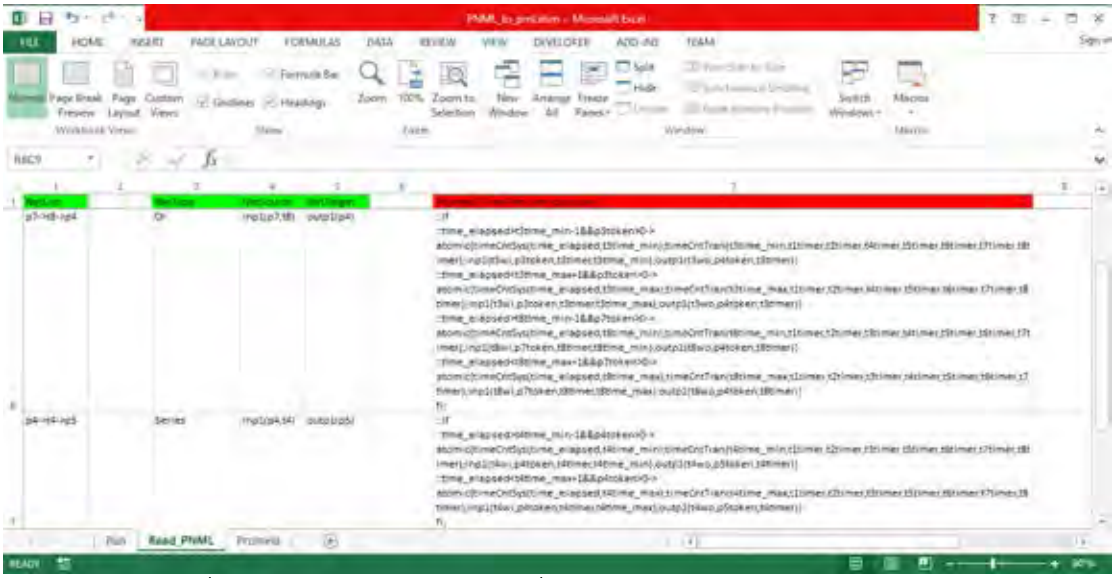
รูปที่ 4-2 ส่วนต่อประสานกับผู้ใช้ที่ชีท Run

ส่วนต่อประสานกับผู้ใช้ที่ชีท Read_PNML ประกอบด้วย 2 ส่วน ได้แก่ ส่วนประกาศตัวแปรและส่วนโครงสร้างเนต ซึ่งเครื่องมือจะแสดงข้อมูลโพลเมลาให้ผู้ใช้ได้เห็น โดยส่วนประกาศตัวแปร (รูปที่ 4-3) แสดงเพลสและทรานซิชันที่ตัวแปรเป็นไปตามกฎกำหนดชื่อตัวแปรโพรเมลา และส่วนโครงสร้างเนต (รูปที่ 4-4) แสดงโพรเมลาที่เป็นไปตามกฎโครงสร้างเนต ฟังก์ชันหลักของโปรแกรมคือ อ่านพีเอ็นเอ็มแอลส่วนการประกาศตัวแปรและส่วนโครงสร้างเนต จากนั้นนำข้อมูลมาแสดงผลที่ชีท Read_PNML ตามที่กล่าวไปแล้วข้างต้น



	13	14	15	16	17	18	19	20
1	Place	Initial Marking	Transition	T_min	T_max	wi	wo	
2	p1	1	t1	1	2	1	1	
3	p2	0	t2	1	2	1	1	
4	p3	0	t3	1	2	1	1	
5	p4	0	t4	1	2	1	1	
6	p5	0	t5	1	2	1	1	
7	p6	0	t6	1	2	1	1	
8	p7	0	t7	1	2	1	1	
9			t8	1	2	1	1	

รูปที่ 4-3 ส่วนต่อประสานกับผู้ใช้ที่ชีท Read_PNML ส่วนประกาศตัวแปร



รูปที่ 4-4 ส่วนต่อประสานกับผู้ใช้ที่ชีท Read_PNML ส่วนโครงสร้างเนต

ส่วนต่อประสานกับผู้ใช้ที่ชีท Promela นี้ (รูปที่ 4-5) ผู้ใช้ต้องกรอกแอลทีแอล กฎกำหนดการเปลี่ยนแปลงเวลา และกฎการเคลื่อนที่โทเค็นในเซลล์ เพื่อให้เครื่องมืออ่านแล้วนำไปสร้างโพรเมลาต่อไป ดังตัวอย่างดังด้านล่าง

แอลทีแอล

#define safe (p1token < 3 && p2token < 2 && p3token < 2)

ltl safety {[] safe}

กฎกำหนดการเปลี่ยนแปลงเวลา และกฎการเคลื่อนที่โทเค็น

#define timeCntSys(Tsys,Tlim) Tsys= Tsys+Tlim

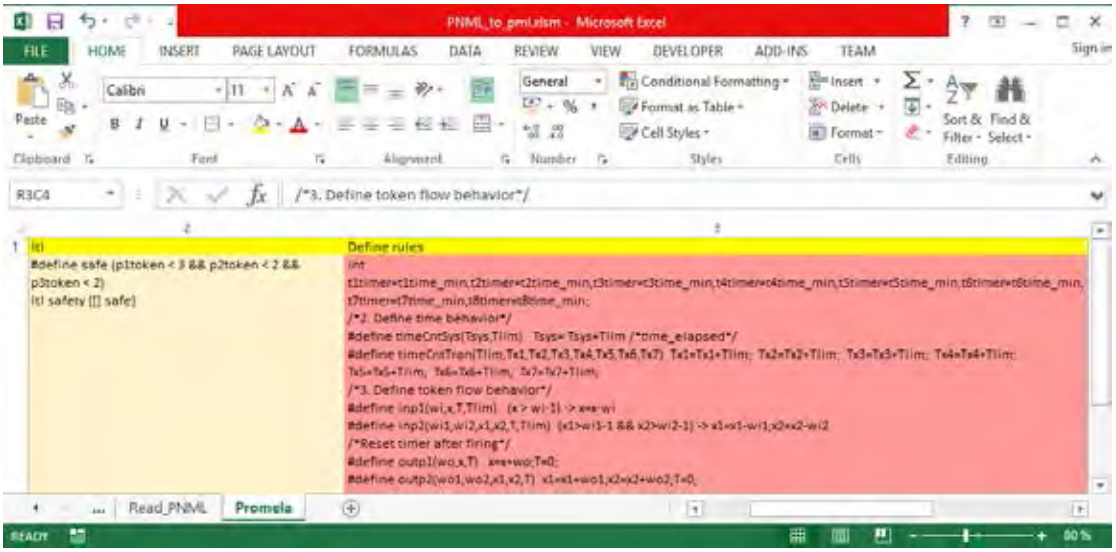
#define timeCntTran(Tlim,Tx1,Tx2,Tx3,Tx4,Tx5,Tx6,Tx7) Tx1=Tx1+Tlim; Tx2=Tx2+Tlim; Tx3=Tx3+Tlim; Tx4=Tx4+Tlim; Tx5=Tx5+Tlim; Tx6=Tx6+Tlim; Tx7=Tx7+Tlim;

#define inp1(wi,x,T,Tlim) (x > wi-1) -> x=x-wi

#define inp2(wi1,wi2,x1,x2,T,Tlim) (x1>wi1-1 && x2>wi2-1) -> x1=x1-wi1;x2=x2-wi2

#define outp1(wo,x,T) x=x+wo;T=0;

#define outp2(wo1,wo2,x1,x2,T) x1=x1+wo1;x2=x2+wo2;T=0;



รูปที่ 4-5 ส่วนต่อประสานกับผู้ใช้ที่ชีท Promela

บทที่ 5

การตรวจสอบแบบจำลอง

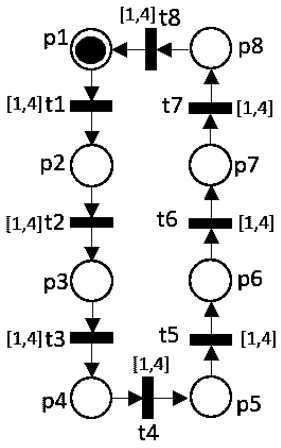
การตรวจสอบคุณสมบัติของไทม์เพทริเน็ตที่สนใจพิจารณา มีตรวจสอบโมเดล 2 ประเภท ได้แก่ การตรวจสอบเชิงคุณภาพและการตรวจสอบเชิงปริมาณ โดยการตวรจสอบจะตรวจสอบด้วยเครื่องมือสปินกับแบบจำลองรูปแบบโพรเมลา

การตรวจสอบเชิงคุณภาพเป็นการตรวจสอบคุณสมบัติทั่วไป ซึ่งใช้วิธีการแอลทีแอลในการตรวจสอบ ประกอบด้วยการตรวจสอบคุณสมบัติความปลอดภัย ความคงอยู่ และความทนทาน

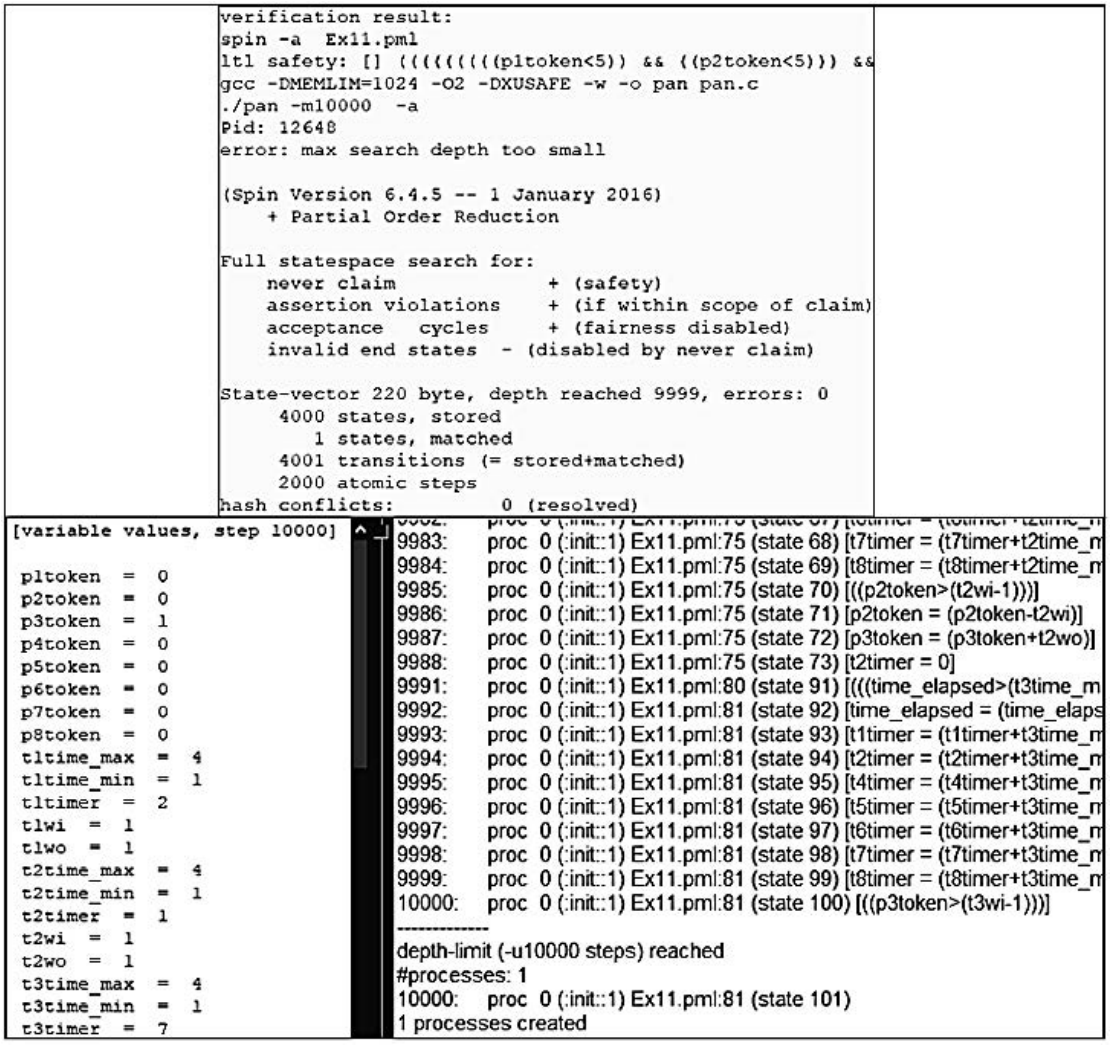
การตรวจสอบเชิงปริมาณเป็นการตรวจสอบปริมาณระยะเวลาที่เป็นข้อกำหนดคุณสมบัติของไทม์เพทริเน็ต มีเวลาที่แต่ละทรานซิชัน เวลาของทั้งระบบ อีกทั้งสามารถตรวจสอบปริมาณโทเค็นที่แต่ละเพลสได้อีกด้วย

5.1 ผลลัพธ์การตรวจสอบคุณสมบัติความปลอดภัยที่ตรวจสอบได้ผลว่าผ่านของแบบจำลองโครงสร้างอนุกรม

ข้อกำหนดแอลทีแอลตรวจสอบความปลอดภัยว่า ทุกเพลสต้องมีจำนวนโทเค็นน้อยกว่า 5 เสมอ ซึ่งเขียนเป็นแอลทีแอลเป็น ltl safety {[] (p1token < 5 && p2token < 5 && p3token < 5 && p4token < 5 && p5token < 5 && p6token < 5 && p7token < 5 && p8token < 5 &&)} ทำการตรวจสอบกับแบบจำลองรูปที่ 5-1 พบว่าโทเค็นมีการเคลื่อนที่และข้อกำหนดแอลทีแอลมีผลตรวจสอบผ่าน แสดง “errors: 0” และรันครบ 10000 สเต็ป ตามรูปที่ 5-2



รูปที่ 5-1 ตัวอย่างไทม์เพทริเน็ตที่มีโครงสร้างอนุกรมมีคุณสมบัติความปลอดภัย



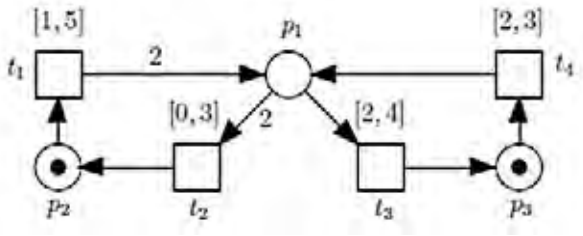
รูปที่ 5-2 ผลลัพธ์การตรวจสอบความปลอดภัยด้วยแอลทีแอลที่ตรวจสอบได้ผลว่าผ่าน

5.2 ผลลัพธ์การตรวจสอบคุณสมบัติความปลอดภัยที่ตรวจสอบได้ผลว่าผ่านของแบบจำลองโครงสร้างเนตทางเลือก

การตรวจสอบคุณสมบัติความปลอดภัยตรวจด้วยวิธีการแอลทีแอล โดยพิจารณาข้อกำหนดว่าแต่ละเพลสต้องมีจำนวนโทเค็นน้อยกว่าค่าที่กำหนดตลอดเวลา เขียนเป็นแอลทีแอลได้ดังนี้

ltl p: [] ((((p1token<4)) && ((p2token<4))) && ((p3token<4))) หมายถึง เพลส p1 มีโทเค็นน้อยกว่า 4 และเพลส p2 มีโทเค็นน้อยกว่า 4 และเพลส p3 มีโทเค็นน้อยกว่า 4 เสมอ

โดยที่ทำการตรวจสอบกับแบบจำลองที่มีเพลส p1 เป็นโครงสร้างเนตทางเลือกตามรูปที่ 5-3 ซึ่งผลลัพธ์จากสปินเป็นดังรูปที่ 5-4 ผลการตรวจสอบผ่านเพราะไม่มีเพลสใดมีจำนวนโทเค็นมากกว่าค่าที่กำหนดตามข้อกำหนดแอลทีแอล



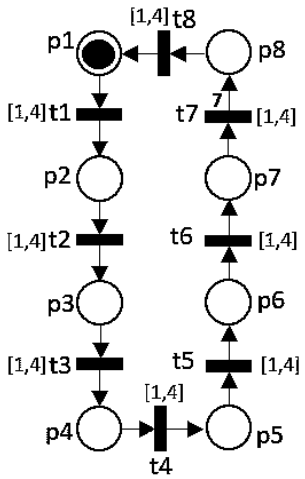
รูปที่ 5-3 ตัวอย่างไทม์เพทริเน็ตที่มีโครงสร้างเนตทางเลือกมีคุณสมบัติความปลอดภัย

```
verification result:
spin -a  Louchka3Modi3_SeperateMaxMin_safeYes.pml
ltl p: [] ((((p1token<4)) && ((p2token<4))) && ((p3token<4)))
gcc -DMEMLIM=1024 -O2 -DXUSAFE -w -o pan pan.c
./pan -m10000  -a
Pid: 10160
error: max search depth too small
Depth=    9999 States=    1e+06 Transitions= 1.36e+06 Memory=   250.897 t=    0.919 R=   1e+06
Depth=    9999 States=    2e+06 Transitions= 2.73e+06 Memory=   372.968 t=     1.77 R=   1e+06
Depth=    9999 States=    3e+06 Transitions= 4.09e+06 Memory=   495.038 t=     2.63 R=   1e+06
Depth=    9999 States=    4e+06 Transitions= 5.46e+06 Memory=   617.108 t=     3.52 R=   1e+06
Depth=    9999 States=    5e+06 Transitions= 6.82e+06 Memory=   739.179 t=     4.41 R=   1e+06
Depth=    9999 States=    6e+06 Transitions= 8.19e+06 Memory=   861.249 t=     5.31 R=   1e+06
Depth=    9999 States=    7e+06 Transitions= 9.55e+06 Memory=   983.319 t=     6.19 R=   1e+06
pan: reached -DMEMLIM bound
    1.07368e+09 bytes used
    102400 bytes more needed
    1.07374e+09 bytes limit
hint: to reduce memory, recompile with
  -DCOLLAPSE # good, fast compression, or
  -DMA=116   # better/slower compression, or
  -DHC # hash-compaction, approximation
  -DBITSTATE # supertrace, approximation

(Spin Version 6.4.5 -- 1 January 2016)
Warning: Search not completed
    + Partial Order Reduction

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

State-vector 116 byte, depth reached 9999, errors: 0
  7333398 states, stored
  2670616 states, matched
 10004014 transitions (= stored+matched)
  4720494 atomic steps
hash conflicts:    525949 (resolved)
```

รูปที่ 5-4 ผลลัพธ์การตรวจสอบที่ตรวจสอบได้ผลว่าผ่านของแบบจำลองโครงสร้างเนตทางเลือก

การตรวจสอบคุณสมบัติความปลอดภัยรูปที่ 5-4 สปินแสดงผลลัพธ์ “errors: 0” พบว่าตรวจสอบผ่านตามข้อกำหนดแอลทีแอล ดังนั้นไทม์เพทริเน็ตนี้มีคุณสมบัติความปลอดภัยเนื่องจากเพลสมีจำนวนโทเค็นเกินกว่าค่าที่กำหนด

5.3 ผลลัพธ์การตรวจสอบคุณสมบัติความปลอดภัยที่ตรวจสอบได้ผลว่าไม่ผ่านของแบบจำลองโครงสร้างอนุกรม



รูปที่ 5-5 ตัวอย่างไทม์เพทริเน็ตที่มีโครงสร้างอนุกรมไม่มีคุณสมบัติความปลอดภัย

การทดลองด้วยแบบจำลองรูปที่ 5-5 กำหนดเปลี่ยนแปลงค่าน้ำหนักที่ตัวแปรน้ำหนักเอาท์พุตของทรานซิชัน 7 ให้มีค่าเป็น 7 ส่วนน้ำหนักอินพุตและน้ำหนักเอาท์พุตอื่นๆมีค่า 1 ซึ่งส่วนน้ำหนักอินพุตและน้ำหนักเอาท์พุตโพรเมลาดังนี้

int t1wi = 1,t1wo = 1,t2wi = 2,t2wo = 1,t3wi = 1,t3wo = 1,t4wi = 1,t4wo = 1;

int t5wi = 1,t5wo = 1,t6wi = 2,t6wo = 1,t7wi = 1,**t7wo = 7**,t8wi = 1,t8wo = 1;

โดยที่ข้อกำหนดแอลทีแอลตรวจสอบความปลอดภัยว่า เพลส p8 ต้องมีจำนวนโทเค็นน้อยกว่า 5 เสมอ ซึ่งเขียนเป็นแอลทีแอลเป็น ltl safety {[] (p8token < 5)}

เมื่อรันสปินตรวจสอบความปลอดภัย พบว่าสปินพบ “errors: 1” แสดงว่าแบบจำลองนี้ไม่มีคุณสมบัติความปลอดภัย และสปินหยุดทำงานในสเต็ปที่ 36 เนื่องจากพบคุณสมบัติดังกล่าวแล้ว ผลลัพธ์ดังรูปที่ 5-6 จากนั้นต้องการทราบว่าแต่ละเพลสมีจำนวนโทเค็นอยู่เท่าใดที่ทำให้แบบจำลอง

ไม่มีคุณสมบัติความปลอดภัย จึงได้ทำการรันสปินในโหมด simulation ต่อไปเพื่อหาว่าแต่ละเพลสมีค่าโทเค็นเท่าใดตอนที่แบบจำลองไม่มีคุณสมบัติความปลอดภัย

```
verification result:
spin -a  Ex1.pml
ltl safety: [] ((p8token<5))
gcc -DMEMLIM=1024 -O2 -DXUSAFE -w -o pan pan.c
./pan -m10000  -a
Pid: 6080
pan:1: assertion violated  !( !((p8token<5))) (at depth 35)
pan: wrote Ex1.pml.trail

(Spin Version 6.4.5 -- 1 January 2016)
Warning: Search not completed
    + Partial Order Reduction

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

State-vector 220 byte, depth reached 35, errors: 1
       15 states, stored
        0 states, matched
       15 transitions (= stored+matched)
        7 atomic steps
hash conflicts:         0 (resolved)
```

รูปที่ 5-6 ผลลัพธ์การตรวจสอบความปลอดภัยด้วยแอลทีแอลที่ตรวจสอบได้ผลว่าไม่ผ่าน

```
[variable values, step  36]

p1token  =  0
p2token  =  0
p3token  =  0
p4token  =  0
p5token  =  0
p6token  =  0
p7token  =  0
p8token  =  7
t1time_max  =  4
t1time_min  =  1
```

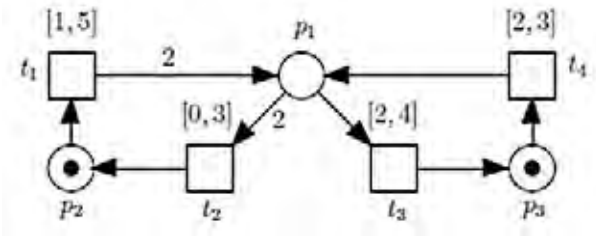
```
 35:    proc  0 (:init::1) Ex1.pml:100 (state 191) [p7token = (p7token-t7wi)]
 35:    proc  0 (:init::1) Ex1.pml:100 (state 192) [p8token = (p8token+t7wo)]
 35:    proc  0 (:init::1) Ex1.pml:100 (state 193) [t7timer = 0]
MSC: ~G line 3
 36:    proc  - (safety:1) _spin_nvr.tmp:3 (state 1) [(!((p8token<5)))]
spin: _spin_nvr.tmp:3, Error: assertion violated
spin: text of failed assertion: assert(!(!((p8token<5))))
#processes: 1
 36:    proc  0 (:init::1) Ex1.pml:60 (state 243)
 36:    proc  - (safety:1) _spin_nvr.tmp:3 (state 2)
1 processes created
```

รูปที่ 5-7 ผลลัพธ์การตรวจสอบแสดงจำนวนโทเค็นที่แต่ละเพลส ณ สเต็ปที่ 36

ผลลัพธ์การรันโหมดจำลอง (simulation) ทำให้ทราบว่าแต่ละเพลสมีจำนวนทเค็นอยู่เท่าใด ดังรูปที่ 5-7 ซึ่งจากแบบจำลองโครงสร้างอนุกรมจะเห็นว่าโทเค็นมีการเคลื่อนที่จากเพลส p1 ไปยังเพลส p2, p3, p4, p5, p6, p7 และ p8 ตามลำดับ จนกระทั่ง ณ สเต็ปที่ 36 พบว่าเพลส p8 มีจำนวนโทเค็นอยู่ 7 แสดงที่ตัวแปร p8token สปินหยุดรันเพราะพบความผิดพลาดตามแอลทีแอลที่กำหนด ดังนั้นจึงสรุปได้ว่าแบบจำลองไม่มีคุณสมบัติความปลอดภัยเพราะมีจำนวนโทเค็นเกินกว่าที่กำหนดในแอลทีแอล

5.4 ผลลัพธ์การตรวจสอบคุณสมบัติความปลอดภัยที่ตรวจสอบได้ผลว่าไม่ผ่านของแบบจำลองโครงสร้างเนตทางเลือก

การตรวจสอบคุณสมบัติความปลอดภัยตรวจด้วยวิธีการแอลทีแอล โดยไทม์เพทริเน็ต พิจารณาข้อกำหนดว่าแต่ละเพลสต้องมีจำนวนโทเค็นไม่เกินกว่าค่าที่กำหนดตลอดเวลา



รูปที่ 5-8 ตัวอย่างไทม์เพทริเน็ตที่มีโครงสร้างเนตทางเลือกไม่มีคุณสมบัติความปลอดภัย

จากตัวอย่างไทม์เพทริเน็ตรูปที่ 5-8 พิจารณาข้อกำหนด p โดยกำหนดให้

- เพลส p1 มีโทเค็นน้อยกว่า 3 โทเค็น
- เพลส p2 มีโทเค็นน้อยกว่า 2 โทเค็น
- เพลส p3 มีโทเค็นน้อยกว่า 2 โทเค็น

ซึ่งทำการกำหนดคุณสมบัติเป็นโพรเมลาได้ดังนี้

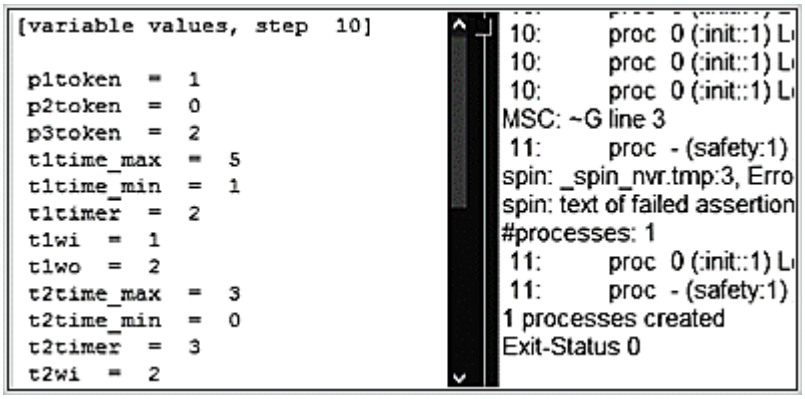
#define safety (p1token < 3 && p2token < 2 && p3token < 2)

จากนั้นกำหนดแอลทีแอลเพื่อตรวจสอบคุณสมบัติดังนี้ ltl p{[] safe} แล้วจึงไปรันกับเครื่องมือสปินเพื่อหาผลลัพธ์ต่อไป

```
verification result:
spin -a  Louchka3Modi3_SeperateMaxMin_safe.pml
ltl p: [] ((((p1token<3)) && ((p2token<2))) && ((p3token<2)))
gcc -DMEMLIM=1024 -O2 -DXUSAFE -w -o pan pan.c
./pan -m10000  -a
Pid: 3428
pan:1: assertion violated  !( !(((((p1token<3)&&(p2token<2))&&(p3token<2)))) (at depth 10)
pan: wrote Louchka3Modi3_SeperateMaxMin_safe.pml.trail

(Spin Version 6.4.5 -- 1 January 2016)
Warning: Search not completed
    + Partial Order Reduction

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

State-vector 116 byte, depth reached 10, errors: 1
        5 states, stored
        0 states, matched
        5 transitions (= stored+matched)
        2 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.001   equivalent memory usage for states (stored*(State-vector + overhead))
    0.281   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
```

รูปที่ 5-9 ผลลัพธ์สปินจากการตรวจสอบคุณสมบัติความปลอดภัยที่ตรวจสอบได้ผลว่าไม่ผ่าน

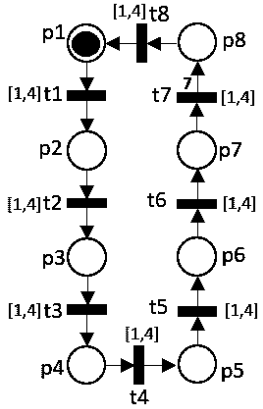
การตรวจสอบคุณสมบัติความปลอดภัยด้วยโหมดการตรวจสอบรูปที่ 5-9 สปินแสดงผลลัพธ์ “errors: 1” หมายถึงไทม์เพทริเน็ตนี้ไม่มีคุณสมบัติความปลอดภัยเนื่องจากเพลสมีจำนวนโทเค็นเกินกว่าค่าที่กำหนด จากนั้นทำการตรวจสอบหาความไม่ปลอดภัยอีกครั้งด้วยโหมดจำลอง ผลลัพธ์จากสปินดังรูปที่ 5-10 พบว่าที่เพลส p3 มีจำนวนโทเค็นมากกว่าที่กำหนดจากข้อกำหนด p เนื่องจากตัวแปร p3token = 2 จึงเกิดคุณสมบัติไม่ปลอดภัย



รูปที่ 5-10 ผลลัพธ์สปินุ่ตรวจสอบคุณสมบัติความปลอดภัยและจำนวนโทเค็นที่แต่ละเพลส

5.5 ผลลัพธ์การตรวจสอบคุณสมบัติความคงอยู่ที่ตรวจสอบได้ผลว่าผ่านของแบบจำลองโครงสร้างอนุกรม

คุณสมบัติความคงอยู่สำหรับไทม์เพทริเน็ต คือ ในระบบไทม์เพทริเน็ตจะต้องมีโทเค็นอยู่อย่างน้อย 1 โทเค็น ทำการทดลองกับแบบจำลองที่มีการกำหนดค่าเดียวกันกับรูปที่ 5-11



รูปที่ 5-11 ตัวอย่างไทม์เพทริเน็ตที่มีโครงสร้างอนุกรมมีคุณสมบัติความคงอยู่

พิจารณาคุณสมบัติความคงอยู่โดยกำหนดแอลทีแอลดังนี้

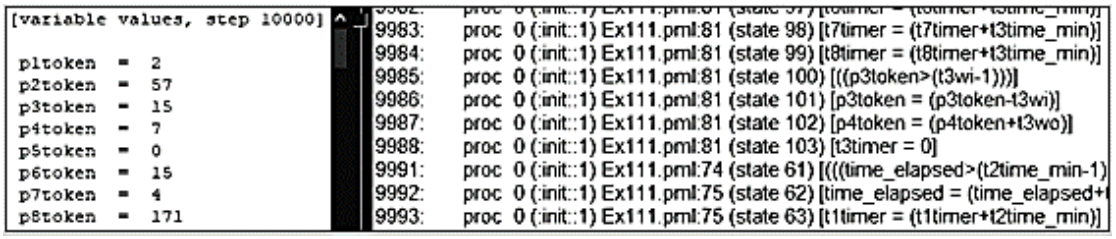
ltl q{[](p1token+p2token+p3token+p4token+p5token+p6token+p7token+p8token)>0}

```
verification result:
spin -a  Ex111.pml
ltl live: ('M'>0)
gcc -DMEMLIM=1024 -O2 -DXUSAFE -w -o pan pan.c
./pan -m10000  -a
Pid: 4604

(Spin Version 6.4.5 -- 1 January 2016)
    + Partial Order Reduction

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

State-vector 220 byte, depth reached 0, errors: 0
        1 states, stored
        0 states, matched
        1 transitions (= stored+matched)
        0 atomic steps
hash conflicts:         0 (resolved)
```

รูปที่ 5-12 ผลลัพธ์การตรวจสอบความคงอยู่ด้วยแอลทีแอลที่ตรวจสอบได้ผลว่าผ่าน

ผลลัพธ์จากสปินรูปที่ 5-12 พบว่า แบบจำลองนี้มีคุณสมบัติความคงอยู่ ดูได้จากสปินแสดง “errors: 0” จากการตรวจสอบด้วยแอลทีแอลข้างต้น จากนั้นดูผลลัพธ์ต่อไปที่สเต็ป 10000 เห็นค่า

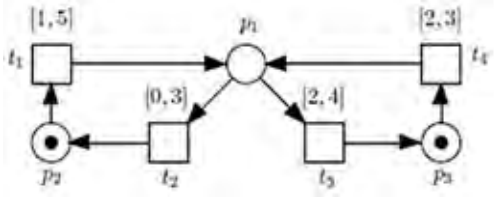
โทเค็นที่อยู่แต่ละเพลส พบว่าทั้งระบบมีโทเค็นอย่างน้อย 1 โทเค็น ทำให้โทเค็นมีการเคลื่อนที่เสมอ ใน 10000 สเต็ป ดังผลลัพธ์รูปที่ 5-13



รูปที่ 5-13 ผลลัพธ์การตรวจสอบแสดงจำนวนโทเค็นที่แต่ละเพลส ณ สเต็ปที่ 10000

5.6 ผลลัพธ์การตรวจสอบคุณสมบัติความคงอยู่ที่ตรวจสอบได้ผลว่าผ่านของแบบจำลองโครงสร้างเนตทางเลือก

ตรวจสอบคุณสมบัติความคงอยู่จากตัวอย่างแบบจำลองโครงสร้างเนตทางเลือก เพื่อตรวจสอบว่าแบบจำลองสามารถมีคุณสมบัติความคงอยู่ได้ โดยนำแบบจำลองเดิมมากำหนดค่าน้ำหนักอินพุตใหม่เป็น t1wi = 1, t2wi = 1, t3wi = 1 ,t4wi = 1 และกำหนดค่าน้ำหนักเอาท์พุตใหม่เป็น t1wo = 1, t2wo = 1, t3wo = 1 ,t4wo = 1 ดังแบบจำลองรูปที่ 5-14



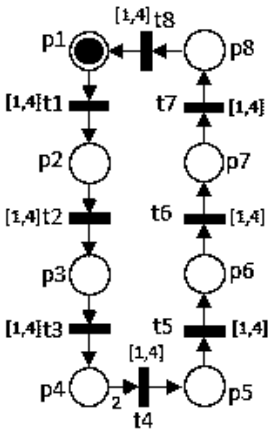
รูปที่ 5-14 ตัวอย่างไทม์เพทริเน็ตที่มีโครงสร้างเนตทางเลือกที่มีคุณสมบัติความคงอยู่

รูปที่ 5-15 ผลลัพธ์สปินตรวจสอบคุณสมบัติความคงอยู่ที่ตรวจสอบได้ผลว่าผ่าน

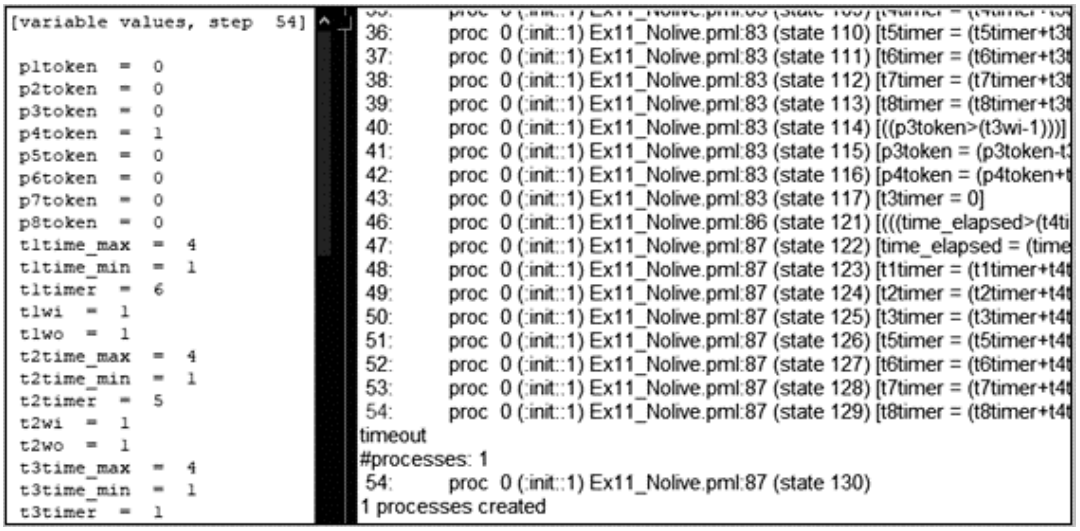
ผลลัพธ์ดังรูปที่ 5-15 พบว่าสปินรันได้ถึงสเต็ป 10000 และผลตรวจสอบแอลทีแอลผ่านจากผลลัพธ์ “errors: 0” แสดงว่ามีโทเค็นเคลื่อนในระบบตลอดเวลา อีกทั้งสเต็ปที่ 10000 จำนวนโทเค็นเป็นดพลส p1 มี 1 โทเค็น เพลส p2 ไม่มีโทเค็น และเพลส p3 มี 1 โทเค็น ดังนั้นแบบจำลองมีคุณสมบัติความคงอยู่

5.7 ผลลัพธ์การตรวจสอบคุณสมบัติความคงอยู่ที่ตรวจสอบได้ผลว่าไม่ผ่านของแบบจำลองโครงสร้างอนุกรม

แบบจำลองโครงสร้างอนุกรมนี้ รูปที่ 5-16 มีการกำหนดน้ำหนักอินพุตเพิ่มเติมเป็น int t4wi=2; และส่วนทรานซิชันอื่นๆน้ำหนักอินพุตมีค่าเป็น 1 เพื่อคาดหวังผลลัพธ์ให้โทเค็นหยุดการเคลื่อนที่ ไม่มีคุณสมบัติความคงอยู่



รูปที่ 5-16 ตัวอย่างไทม์เพทริเน็ตที่มีโครงสร้างอนุกรมไม่มีคุณสมบัติความคงอยู่

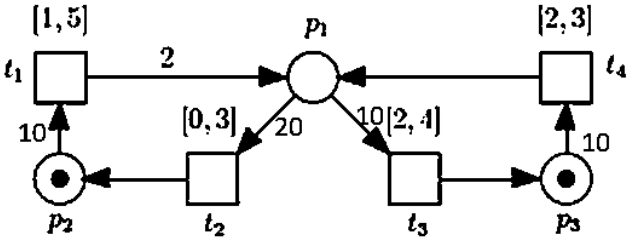


รูปที่ 5-17 ผลลัพธ์สปินของไทม์เพทริเน็ตที่มีโครงสร้างอนุกรมไม่มีคุณสมบัติความคงอยู่ ณ สเต็ป 54

ผลลัพธ์สปินรูปที่ 5-17 ทำการทดลองตั้งค่าสปินที่ 10000 สเต็ปแต่สปินรันได้ 54 สเต็ปแล้วหยุด จะเห็นว่าโทเค็นเคลื่อนที่จากเพลส p1 ไปยังเพลส p2 p3 และ p4 ตามลำดับ แต่โทเค็นไม่เคลื่อนที่ไปยังเพลส p5 เพราะจำนวนโทเค็นที่เพลส p4 ไม่เพียงพอต่อน้ำหนักอินพุตที่ทรานซิชัน t4 ดังนั้นเพลสหยุดการเคลื่อนที่และไม่มีคุณสมบัติความคงอยู่

5.8 ผลลัพธ์การตรวจสอบคุณสมบัติความคงอยู่ที่ตรวจสอบได้ผลว่าไม่ผ่านของแบบจำลองโครงสร้างเนตทางเลือก

ตรวจสอบคุณสมบัติความคงอยู่จากตัวอย่างโดยนำแบบจำลองเดิมมากำหนดค่าน้ำหนักอินพุตใหม่เป็น t1wi = 10, t2wi = 20, t3wi = 10 ,t4wi = 10 แบบจำลองรูปที่ 5-18



รูปที่ 5-18 ตัวอย่างไทม์เพทริเน็ตที่มีโครงสร้างเนตทางเลือกไม่มีคุณสมบัติความคงอยู่

เพื่อให้แน่ใจว่าแบบจำลองไม่มีคุณสมบัติความคงอยู่ เนื่องจากที่อินพุตเพลสมีจำนวนโทเค็นน้อยกว่าน้ำหนักอินพุต ทำให้โทเค็นไม่เคลื่อนที่ การทดลองกำหนดคุณสมบัติการตรวจสอบโพรเมลาด้วยแอลทีแอลแล้วจึงไปรันกับเครื่องมือสปิน โดยที่แอลทีแอลดังนี้

ltl {[] (p1token+p2token+p3token) > 0}

ผลลัพธ์ดังรูปที่ 5-19 พบว่าสปินมีส่วนของโครงสร้างเนตที่เข้าถึงไม่ได้และหยุดรันที่สเต็ป 5 แสดงว่าโทเค็นไม่เคลื่อนที่ เพลส p2 และ p3 มีจำนวนโทเค็นเท่ากับมาร์คกิ้งเริ่มต้น แบบจำลองนี้จึงไม่มีคุณสมบัติความคงอยู่

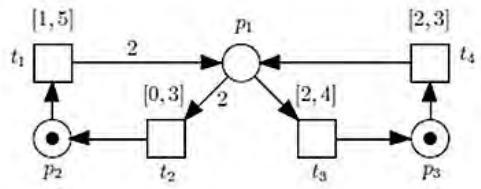
```
verification result:
spin -a  Louchka_live2.pml
ltl ltl_0: ('M'>0)
gcc -DMEMLIM=1024 -O2 -DXUSAFE -w -o pan pan.c
./pan -m10000  -a
Pid: 8284

(Spin Version 6.4.5 -- 1 January 2016)
    + Partial Order Reduction

Full statespace search for:
    never claim             + (ltl_0)
    assertion violations    + (if within scope of claim)
    acceptance   cycles     + (fairness disabled)
    invalid end states  - (disabled by never claim)

State-vector 116 byte, depth reached 0, errors: 0
        1 states, stored
        0 states, matched
        1 transitions (= stored+matched)
        0 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.000   equivalent memory usage for states (stored*(State-vector + overhead))
    0.282   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 init
    Louchka_live2.pml:27, state 4, "t1timer = (t1timer+t3time_min)"
    Louchka_live2.pml:27, state 5, "t2timer = (t2timer+t3time_min)"
    Louchka_live2.pml:27, state 6, "((p1token>(t3wi-1)))"
    Louchka_live2.pml:27, state 8, "p3token = (p3token+t3wo)"
```

```
(variable values, step   5)

 p1token  =  0
 p2token  =  1
 p3token  =  1
 t1time_max  =  5
 t1time_min  =  1
 t1timer  =  1
 t1wi  =  10
 t1wo  =  2
 t2time_max  =  3
 t2time_min  =  0
 t2timer  =  5
 t2wi  =  20
 t2wo  =  1
 t3time_max  =  4
 t3time_min  =  2
 t3timer  =  7
 t3wi  =  10
 t3wo  =  1
 t4time_max  =  3
 t4time_min  =  2
 t4timer  =  7
 t4wi  =  10
 t4wo  =  1
 time_elapsed  =  6
```

```
  0:    proc  - (:root:) creates proc  0 (:init:)
ltl ltl_0: ('M'>0)
  1:    proc  0 (:init::1) Louchka_live2.pml:40 (state 53) [((((time_elapsed<(t1time_max+1
  2:    proc  0 (:init::1) Louchka_live2.pml:41 (state 54) [time_elapsed = (time_elapsed+t
  3:    proc  0 (:init::1) Louchka_live2.pml:41 (state 55) [t4timer = (t4timer+t1time_max)]
  4:    proc  0 (:init::1) Louchka_live2.pml:41 (state 56) [t2timer = (t2timer+t1time_max)]
  5:    proc  0 (:init::1) Louchka_live2.pml:41 (state 57) [t3timer = (t3timer+t1time_max)]
timeout
#processes: 1
  5:    proc  0 (:init::1) Louchka_live2.pml:41 (state 58)
1 processes created
```

รูปที่ 5-19 ผลลัพธ์สปินตรวจสอบคุณสมบัติความคงอยู่ที่ตรวจสอบได้ผลว่าไม่ผ่าน

รูปที่ 5-20 เป็นแบบจำลองที่นำมาตรวจสอบหาคุณสมบัติความคงอยู่ มีเพลส p1 เป็นโครงสร้างเนตทางเลือก ที่ทรานซิชันมีน้ำหนักอินพุตและน้ำหนักเอาท์พุต ส่งผลต่อจำนวนโทเค็นในระบบ ทำให้อาจเกิดการไม่มีคุณสมบัติความคงอยู่



รูปที่ 5-20 ตัวอย่างไทม์เพทริเน็ตที่มีโครงสร้างเนตทางเลือกไม่มีคุณสมบัติความคงอยู่ขณะหนึ่ง

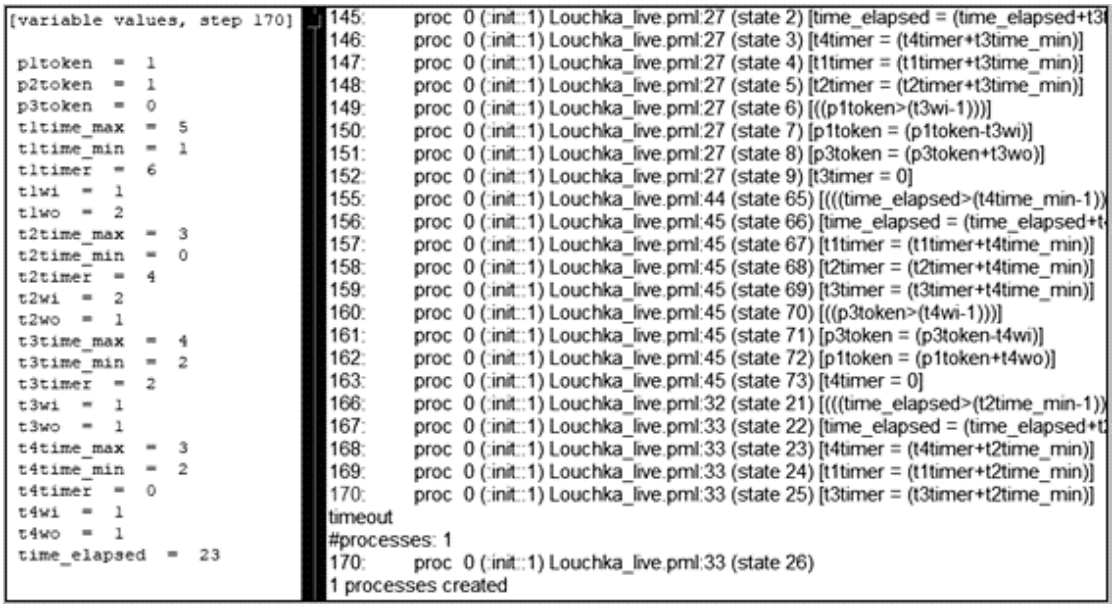
การตรวจสอบคุณสมบัติความคงอยู่ตรวจด้วยวิธีการแอลทีแอล โดยไทม์เพทริเน็ตพิจารณาข้อกำหนดว่าต้องมีโทเค็นอยู่ในระบบเสมอหรือเพลสใดเพลสหนึ่งในระบบของไทม์เพทริเน็ตต้องมีโทเค็นอย่างน้อย 1 โทเค็น ซึ่งทำการกำหนดคุณสมบัติเป็นโพรเมลาด้วยแอลทีแอลแล้วจึงไปรันกับเครื่องมือสปินเพื่อหาผลลัพธ์ โดยที่แอลทีแอลดังนี้ ltl {[] (p1token+p2token+p3token) > 0}

```
verification result:
spin -a  Louchka_live.pml
ltl ltl_0: ('M'>0)
gcc -DMEMLIM=1024 -O2 -DXUSAFE -w -o pan pan.c
./pan -m10000  -a
Pid: 11668

(Spin Version 6.4.5 -- 1 January 2016)
    + Partial Order Reduction

Full statespace search for:                                    Capture
    never claim             + (ltl_0)
    assertion violations    + (if within scope of claim)
    acceptance   cycles     + (fairness disabled)
    invalid end states  - (disabled by never claim)

State-vector 116 byte, depth reached 0, errors: 0
        1 states, stored
        0 states, matched
        1 transitions (= stored+matched)
        0 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.000   equivalent memory usage for states (stored*(State-vector + overhead))
    0.282   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 init
    Louchka_live.pml:27, state 4, "t1timer = (t1timer+t3time_min)"
    Louchka_live.pml:27, state 5, "t2timer = (t2timer+t3time_min)"
    Louchka_live.pml:27, state 6, "((p1token>(t3wi-1)))"
    Louchka_live.pml:27, state 8, "p3token = (p3token+t3wo)"
```

รูปที่ 5-21 ผลลัพธ์สปินจากการตรวจสอบคุณสมบัติความคงอยู่ที่ตรวจสอบได้ผลว่าผ่าน

การตรวจสอบคุณสมบัติความคงอยู่โหมดการตรวจสอบ รูปที่ 5-21 สปินแสดงผลลัพธ์ “errors: 0” หมายถึงไทม์เพทริเน็ตนี้มีคุณสมบัติความคงอยู่ แต่สปินพบว่ามีส่วนของโครงสร้างเนตที่เข้าถึงไม่ได้ (unreached) ดังนั้นจึงต้องไปรันสปินโหมดจำลองเพื่อหาว่าแบบจำลองไม่มีคุณสมบัติความคงอยู่เมื่อใด

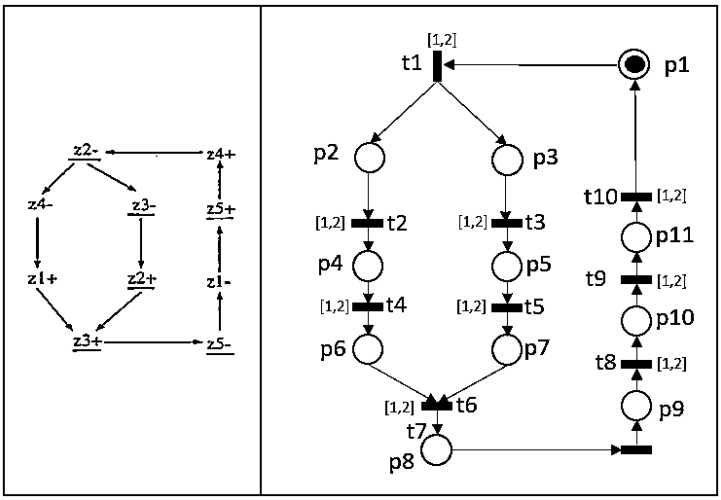
ผลลัพธ์สปินดังรูปที่ 5-22 พบว่าสเต็ปที่ 170 สปินหยุดรัน ไม่มีการเคลื่อนที่ของโทเค็นแต่ก่อนหน้านั้นมีการเคลื่อนที่โทเค็น แสดงว่าแบบจำลองมีคุณสมบัติความคงอยู่จนกระทั่งสเต็ปที่ 170 แบบจำลองไม่มีคุณสมบัติความคงอยู่อีกต่อไป



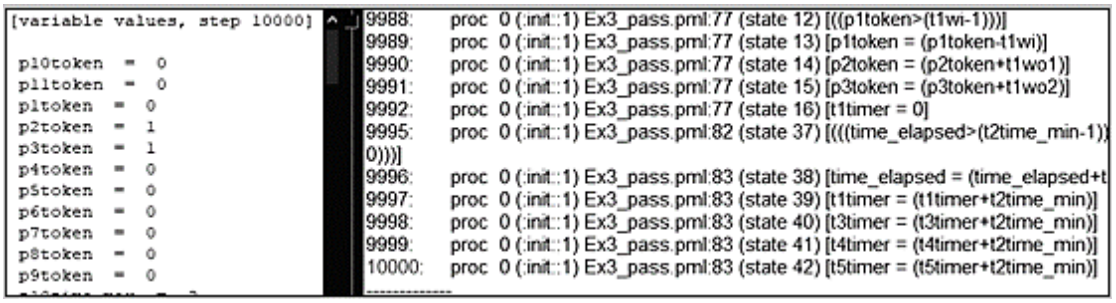
รูปที่ 5-22 ผลลัพธ์สปินตรวจสอบคุณสมบัติความคงอยู่ ณ สเต็ปที่ 170

5.9 ผลลัพธ์การตรวจสอบคุณสมบัติความทนทานที่ตรวจสอบได้ผลว่าผ่าน

รูปที่ 5-23 รูปด้านขวาไทม์เพทริเน็ตที่แปลงมาจากเอสทีจีรูปด้านซ้าย เอสทีจีมีสัญญาณ z1 z2 z3 z4 ซึ่งแต่ละสัญญาณมีสถานะบวกกับลบ โดยการตรวจสอบคุณสมบัติความทนทาน พิจารณาว่า z2- และ z2+ ต้องไม่เกิดขึ้นพร้อมกัน ไทม์เพทริเน็ตแทน z2- ด้วยทรานซิชัน t1 ส่วน z2+ แทนด้วยทรานซิชัน t5 การตรวจสอบพิจารณาข้อกำหนดว่าโทเค็นจะต้องไม่เคลื่อนที่ไปยังเพลส p2 และเพลส p7 ในขณะเดียวกัน



รูปที่ 5-23 ตัวอย่างไทม์เพทริเน็ตที่ตรวจสอบผ่านคุณสมบัติความทนทาน



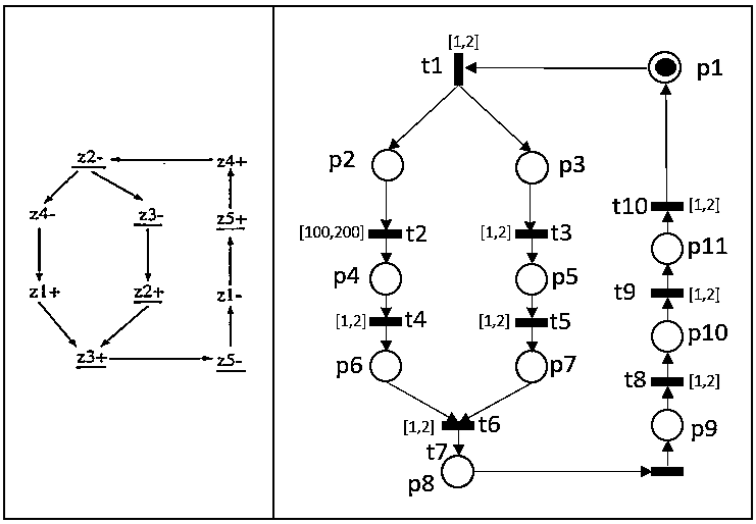
รูปที่ 5-24 ผลลัพธ์สปินตรวจสอบของคุณสมบัติความทนทานที่ตรวจสอบได้ผลว่าผ่าน

ผลลัพธ์สปินรูปที่ 5-24 รันได้ครบ 10000 สเต็ปและตรวจสอบผ่าน ดังนั้นไทม์เพทริเน็ตนี้ตรวจสอบผ่านคุณสมบัติความทนทาน ไทม์เพทริเน็ตจึงไม่มีคุณสมบัติความทนทาน

5.10 ผลลัพธ์การตรวจสอบคุณสมบัติความทนทานที่ตรวจสอบได้ผลว่าไม่ผ่าน

รูปที่ 5-25 มีการกำหนดค่าเวลาที่ทรานซิชัน t2 เป็น [100,200] เพื่อทำการทดลองให้โทเค็นไม่เคลื่อนที่ออกจากเพลส p2 และรอจนเพลส p7 มีโทเค็นเคลื่อนที่เข้ามา เพื่อจำลองว่าไทม์เพทริเน็ตนี้มีคุณสมบัติความทนทานเกิดขึ้น ซึ่งการตรวจสอบพิจารณาข้อกำหนดว่าโทเค็นจะต้องไม่เคลื่อนที่ไปยังเพลส p2 และเพลส p7 ในขณะเดียวกัน ดังนั้นแอลทีแอลเขียนเป็น

ltl q{[](p7token>0->p2token>0)}

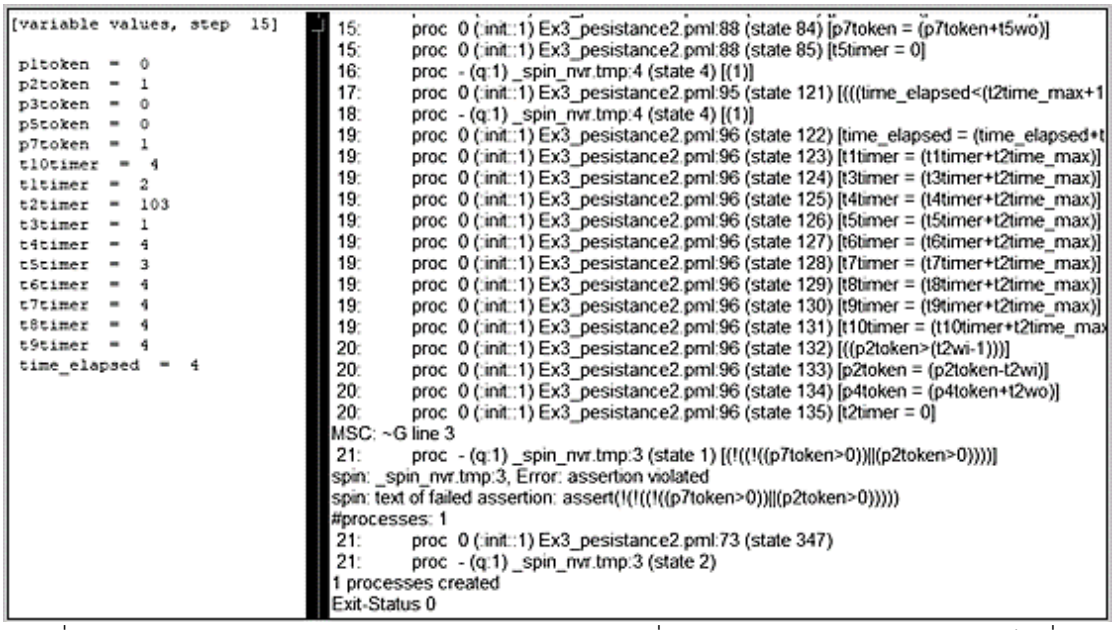


รูปที่ 5-25 ตัวอย่างไทม์เพทริเน็ตที่ตรวจสอบไม่ผ่านคุณสมบัติความทนทาน

รูปที่ 5-26 ผลลัพธ์สปินพบว่า “errors: 1” ดังนั้นแอลทีแอลเป็นจริง จากนั้นไปตรวจสอบเพิ่มเติมด้วยการดูค่าโทเค็นที่เพลส p2 และ p7 ผลลัพธ์ดังรูป 5-27 พบว่ามีโทเค็นอยู่ที่เพลส p2 และ p7 ในเวลาเดียวกันพร้อมกัน ดังนั้นไทม์เพทริเน็ตนี้มีคุณสมบัติความทนทาน

```
verification result:
spin -a  Ex3_pesistance2.pml
ltl q: [] ((! ((p7token>0))) || ((p2token>0)))
gcc -DMEMLIM=1024 -O2 -DXUSAFE -w -o pan pan.c
./pan -m10000  -a
Pid: 12804
pan:1: assertion violated  !( !(( !((p7token>0))||(p2token>0)))) (at depth 20)
pan: wrote Ex3_pesistance2.pml.trail

(Spin Version 6.4.5 -- 1 January 2016)
Warning: Search not completed
    + Partial Order Reduction

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

State-vector 276 byte, depth reached 20, errors: 1
        9 states, stored
        0 states, matched
        9 transitions (= stored+matched)
        4 atomic steps
hash conflicts:         0 (resolved)
```

รูปที่ 5-26 ผลลัพธ์สปินตรวจสอบของคุณสมบัติความทนทานที่ตรวจสอบได้ผลว่าไม่ผ่าน



รูปที่ 5-27 ผลลัพธ์สปินตรวจสอบคุณสมบัติความทนทานที่ตรวจสอบได้ผลว่าไม่ผ่าน ณ สเต็ปที่ 15

5.11 การตรวจสอบเชิงปริมาณ

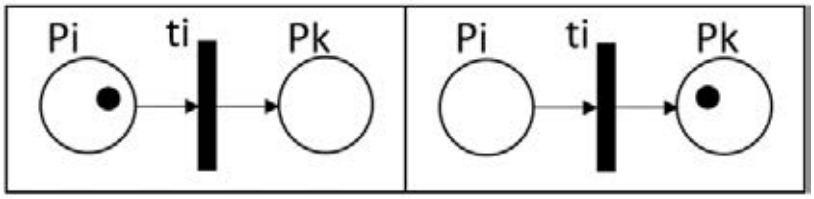
การตรวจสอบเชิงปริมาณมุมมองไทม์เพทริเน็ตในงานวิจัยนี้ตรวจสอบ 2 ชนิด ได้แก่ ปริมาณโทเค็นที่แต่ละเพลสและปริมาณเวลาที่แต่ละทรานซิชัน ซึ่งตัวแปรโพรเมลาที่แสดงปริมาณเป็นดังนี้

pxtoken โดยที่ x แทนด้วยจำนวนเพลส

ตัวแปร pxtoken แทนปริมาณโทเค็นในแต่ละเพลส ณ เวลาใดๆ

tytimer โดยที่ y แทนด้วยจำนวนทรานซิชัน

ตัวแปร tytimer แทนปริมาณเวลาที่แต่ละทรานซิชัน



รูปที่ 5-28 โทเค็นเคลื่อนที่จากเพลส Pi ไปยังเพลส Pk

วิธีการหาเวลาจาก Pi ไปยัง Pk จากรูปที่ 5-28

1.หาสเต็ปที่ทำให้โทเค็นเคลื่อนที่จาก Pi ไป Pk โดยที่

2.สเต็ปตั้งแต่ Pi ขณะที่ทรานซิชันมีสถานะเปิดใช้งานและมีโทเค็นอยู่ที่ Pi >= wi

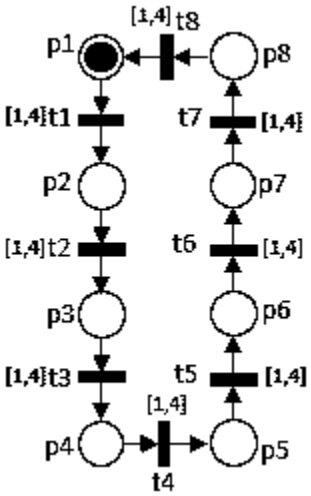
3.สเต็ปขณะเคลื่อนที่ผ่าน Pi = Pi – wi และ Pk = Pk + wo

4.จนถึงสเต็ปที่เวลาทรานซิชัน ti = 0

5.ดังนั้นเวลาจาก Pi ไปยัง Pk คือ ti_timer ที่ค่าสุดท้ายของสเต็ปก่อนหน้าที่ ti = 0

ในกรณีที่ต้องการหาเวลาจากหลายเพลสให้นำเวลาตัวแปร timer มารวมกันด้วยการบวกกัน

ไทม์เพทริเน็ตรูปที่ 5-29 ได้รับการตรวจสอบคุณสมบัติความคงอยู่มาแล้วและผลลัพธ์ผ่าน ดังนั้นเลือกแบบจำลองนี้มาเป็นตัวอย่างตรวจสอบเชิงปริมาณ แบบจำลองมีเพลส p1 p2 p3 p4 p5 p6 p7 p8 ซึ่งแต่ละเพลสมีจำนวนโทเค็นแทนด้วยตัวแปร p1token, p2token, p3token, p4token, p5token, p6token, p7token, p8token ตามลำดับ และแบบจำลองมีทรานซิชัน t1 t2 t3 t4 t5 t6 t7 t8 ซึ่งแต่ละทรานซิชันมีไทม์เมอร์แต่ละตัวเป็นตัวแปร t1timer t2timer t3timer t4timer t5timer t6timer t7timer t8timer ตามลำดับ



รูปที่ 5-29 ไทม์เพทริเน็ตที่ถูกตรวจสอบเชิงปริมาณ

วิธีการหาเวลาจาก p2 ไปยัง p3 จากรูปที่ 5-29

1.หาสเต็ปที่ทำให้โทเค็นเคลื่อนที่จาก p2 ไป p3 โดยที่ p2token =1

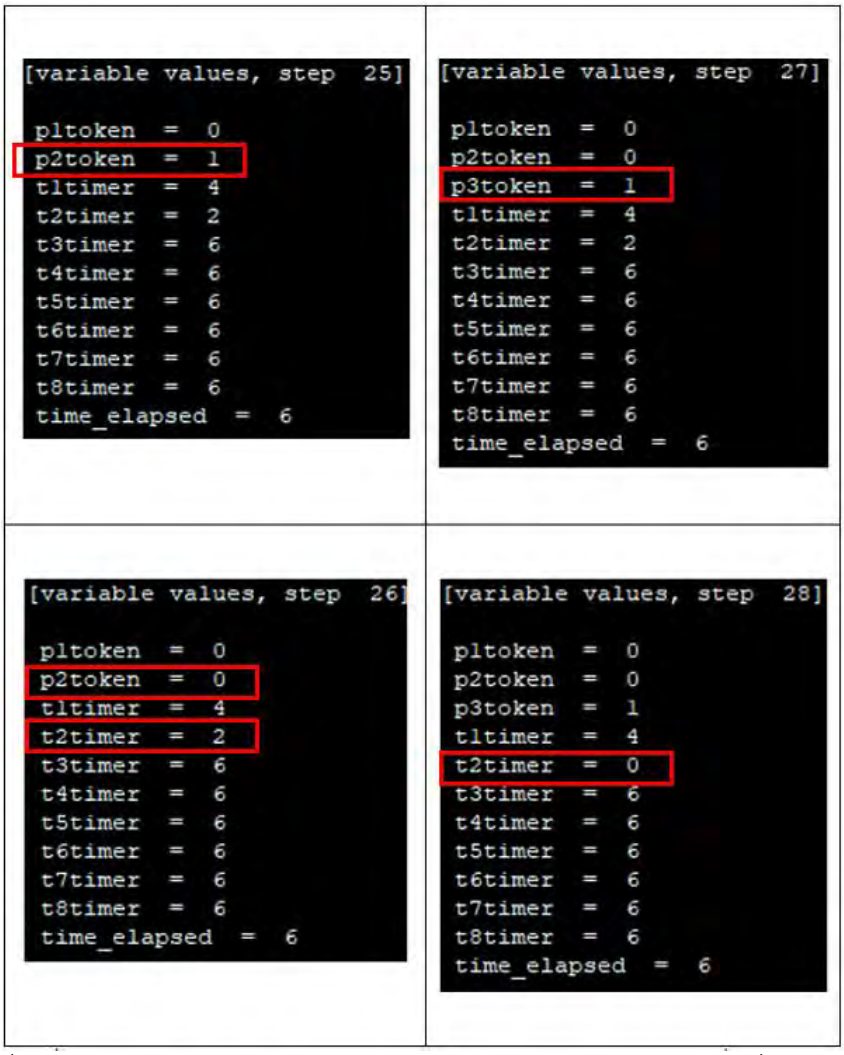
2.สเต็ปตั้งแต่ p2 ขณะที่ทรานซิชันมีสถานะเปิดใช้งานและมีโทเค็นอยู่ที่ p2token >= t2wi

3.สเต็ปขณะเคลื่อนที่ผ่าน p2token = p2token = 0 กับ p3token = 1

4.จนถึงสเต็ปที่เวลาทรานซิชัน t2 คือ t2timer = 0

5.ดังนั้นเวลาจาก p2 ไปยัง p3 คือ t2timer = 2 เป็นค่าสุดท้ายของสเต็ปก่อนหน้าที่ t2timer = 0

ผลลัพธ์การจำลองด้วยสปินจะเห็นว่ามีการแสดงค่าตัวแปรต่างๆ ซึ่งผลลัพธ์ของแบบจำลองไทม์เพทริเน็ตที่แปลงเป็นโพรเมลาแล้ว มีการเคลื่อนที่ของโทเค็นแสดงที่ตัวแปร p1token, p2token, p3token, p4token, p5token, p6token, p7token, p8token ซึ่งสปินมีการแสดงค่าจำนวนโทเค็น ณ ขณะใดๆ ส่วนตัวแปร timer แสดงถึงเวลาที่แต่ละทรานซิชัน ณ ขณะใดๆ โดยที่หลังจากทรานซิชันนั้นมีสถานะหลังเคลื่อนย้ายโทเค็นไปแล้วจะมีการรีเซตเวลาที่ทรานซิชันด้วย ผลลัพธ์ปริมาณโทเค็นที่แต่ละเพลสและปริมาณเวลาที่แต่ละทรานซิชันแสดงดังรูปที่ 5-30 การหาเวลาจากเพลส p2 ไป p3 หรือปริมาณเวลาให้พิจารณาตามหลังจาก firing แล้วจะเห็นว่าโทเค็นเคลื่อนที่ไปยังเอาท์พุตเพลสและมีการรีเซตเวลาที่ทรานซิชัน หลังจากที่โทเค็นเคลื่อนที่ไปถึงเอาท์พุตเพลสปลายทางสุดท้ายที่สนใจแล้ว การดูปริมาณเวลานั้นดูค่า timer ย้อนหลังก่อนที่จะมีการรีเซต 1 สเต็ปของทรานซิชันที่อยู่ก่อนหน้าเอาท์พุตเพลสปลายทางสุดท้าย เวลาที่ทรานซิชันแทนด้วยตัวแปร timer



รูปที่ 5-30 ผลลัพธ์การจำลองการทำงาน (simulation) ด้วยสปิน สเต็ปที่ 25 ถึง 28

จากรูป 5-30 พบว่า สเต็ปที่ 25 p2token = 1 มีโทเค็นอยู่ที่ p2 จากนั้นสเต็ปที่ 26 p2token = 0 ไม่มีโทเค็นอยู่ที่ p2 และกำลังเคลื่อนที่ไปที่ p3 ต่อมาสเต็ปที่ 27 p3token = 1 โทเค็นเคลื่อนที่เข้า p3 และ t2timer = 2 สุดท้ายมาสเต็ปที่ 28 t2timer = 0 ดังนั้นเวลาจาก p2 ไปยัง p3 คือค่าสุดท้ายของสเต็ปก่อนหน้าที่ t2timer = 0 มีค่าเท่ากับ t2timer = 2 และในกรณีที่ต้องการหาปริมาณเวลาการเคลื่อนที่โทเค็นจาก p2 ไปยัง p4 ให้นำเวลาจากตัวแปร timer มาบวกกัน ได้แก่ตัวแปร t2timer บวกกับ t3timer

บทที่ 6

สรุปผลงานวิจัยและข้อเสนอแนะ

6.1 สรุปผลงานวิจัย

งานวิจัยนี้นำเสนอการสร้างแบบจำลองโดยการแปลงไทม์เพทริเน็ตเป็นโพรเมลา เพื่อนำมาตรวจสอบคุณสมบัติด้วยเครื่องมือสปิน แบบจำลองระบบเวลาคอนเคอร์เรนท์มีพฤติกรรมของเวลาแสดงด้วยไทม์เพทริเน็ตซึ่งเป็นพีเอ็นเอ็มแอล ขั้นตอนการสร้างแบบจำลองมี 4 ขั้นตอน ได้แก่ กำหนดชื่อตัวแปรโพรเมลา กำหนดการเปลี่ยนแปลงเวลาด้วยโพรเมลา กำหนดคำสั่งโพรเมลาแสดงการเคลื่อนที่โทเค็น กำหนดเส้นทางทั้งหมดของไทม์เพทริเน็ตตามกฎโครงสร้างเนต หลังจากที่สร้างแบบจำลองเสร็จแล้ว ได้มีการตรวจสอบแบบจำลองโดยการตรวจสอบคุณสมบัติต่างๆด้วยเครื่องมือสปินกับวิธีแอลทีแอล เบื้องต้นตรวจสอบคุณสมบัติความปลอดภัยและคุณสมบัติความคงอยู่

การตรวจสอบแบบจำลองสามารถตรวจสอบได้ทั้งเชิงคุณภาพและเชิงปริมาณ จากผลการตรวจสอบพบว่า ปัจจัยที่มีผลต่อคุณสมบัติของไทม์เพทริเน็ต คือ เวลา จำนวนโทเค็น น้ำหนักอินพุต น้ำหนักเอาท์พุต โดยที่เป็นตัวแปรที่เพิ่มขึ้นมาจากเพทริเน็ตทั่วไป ส่งผลให้พีเอ็นเอ็มแอลต้องมีการดัดแปลงสำหรับตัวแปรไทม์เพทริเน็ตด้วยซึ่งเพิ่มเติมจากพีเอ็นเอ็มแอลสำหรับเพทริเน็ตทั่วไปเช่นกัน

6.2 ข้อจำกัดและข้อเสนอแนะ

การตรวจสอบเชิงปริมาณยังไม่เป็นอัตโนมัติ เนื่องจากการเก็บข้อมูลตัวแปรเชิงปริมาณยังต้องให้ผู้ใช้เก็บข้อมูลด้วยตนเอง ตัวแปรเชิงปริมาณ เช่น ตัวแปรเวลาจากเพลสต้นทางไปยังเพลสปลายทาง จำนวนโทเค็นที่แต่ละสเต็ปการตรวจสอบ ดังนั้นควรมีเครื่องมือในการรวบรวมและแสดงข้อมูลของตัวแปรต่างๆแบบอัตโนมัติ

พฤติกรรมของไทม์เพทริเน็ตในงานวิจัยนี้ในส่วนของการเคลื่อนที่โทเค็นเป็นลำดับขั้นตอน คือ โทเค็นเข้ามายังอินพุตเพลสก่อนแล้วจึงเคลื่อนที่ไปยังเอาท์พุตเพลสโดยจะต้องจบขั้นตอนตามลำดับนี้ก่อน จึงจะรับโทเค็นเข้ามาจากอินพุตเพลสใหม่ได้อีกรอบ ซึ่งยังไม่รองรับการเคลื่อนที่ในขณะที่โทเค็นยังไม่เคลื่อนที่ไปยังเอาท์พุตเพลสก่อน ดังนั้นในกรณีนี้จึงต้องมีช่องว่างของช่วงเวลาให้โทเค็นทำการเคลื่อนที่ไปยังเอาท์พุตเพลสให้แล้วเสร็จก่อน สำหรับในอนาคตอาจมีอัลกอริทึมที่รองรับการเคลื่อนที่โทเค็นเมื่อเคลื่อนที่ไปยังเอาท์พุตเพลสและรับโทเค็นในเวลาพร้อมกัน

บรรณานุกรม

[1] K. Boonroeangkaow, A. Thongtak, and W. Vatanawood, “Formal modeling for Persistence checking of signal transition graph specification with Promela,” IMECS2017, pp161-165, 2017.

[2] T. Murata, "Petri nets: properties analysis and applications", Proceedings of the IEEE, vol. 77, no. 4, pp. 541-580, 1989.

[3] Popova-Zeugmann, Louchka, and Monika Heiner. "Quantitative Evaluation of Time Petri Nets and Applications to Technical and Biochemical Networks." Online], Pages: 1-12, Jul. 20, 2016.

[4] Fatholahzadeh, Nasrin. "Modeling a Bank ATM with Two Directions Places Timed Petri Net (TPN)." Journal of Artificial Intelligence in Electrical Engineering 2.6 (2013): 9-16.

[5] P. Damjan, W. Vatanawood, “Translating UML State Machine Diagram into Promela”, IMECS 2017.

[6] W. Lawsunnee, A. Thongtak, and W. Vatanawood, “Signal persistence checking of asynchronous system implementation using SPIN,” Lect. Notes Eng. Comput. Sci., vol. 2, pp. 604–609, 2015.

[7] Ville R. Koskinen and Juha Plosila, “Applications for the SPIN Model Checker – A Survey,” TUCS Technical Report No. 782, September, 2006.

[8] Silva J.R., del Foyo P.M.G. “Timed Petri Nets”. Pawel Pawlewski (Ed.), Petri Nets: Manufacturing and Computer Science, InTech (2012), pp. 359-378.

[9] Bastide, R., Billington, J., Kindler, E., Kordon, F., Mortensen, K.H., eds.: Meeting on XML/SGML based Interchange Formats for Petri Nets, University of Aarhus, Dept. of Computer Science (2000)

[10] M. Ben-Ari, Principles of the SPIN Model Checker. 2008.

บรรณานุกรม

ภาคผนวก

1. การพัฒนาเครื่องมือสนับสนุนการแปลงไทม์เพทริเน็ตเป็นโพรเมลา

งานวิจัยนี้สร้างเครื่องมือสนับสนุนการแปลงไทม์เพทริเน็ตเป็นโพรเมลา พัฒนาเครื่องมือด้วยภาษาวิชวลเบสิก (VBA : Visual Basic) โดยข้อมูลนำเข้าเป็นพีเอ็นเอ็มแอล และผลลัพธ์ได้เป็นไฟล์โพรเมลา เพื่อนำไปใช้ตรวจสอบคุณสมบัติเชิงคุณภาพและปริมาณด้วยเครื่องมือสปินต่อไป

1.1 โค้ดภาษาวิชวลเบสิก

พัฒนาเครื่องมือสนับสนุนการแปลงไทม์เพทริเน็ตเป็นโพรเมลาด้วยภาษาวิชวลเบสิกโดยเครื่องมือของเอ็กซ์เซล ซึ่งสร้างฟังก์ชันหลายโมดูล รูปแบบดังด้านล่าง

Sub ชื่อฟังก์ชัน ()

Dim ตัวแปร **As** ชนิดตัวแปร

กระบวนการฟังก์ชัน

End Sub

1.1.1 Sub main()

ฟังก์ชันหลักชื่อ main ซึ่งเรียกฟังก์ชันย่อย Declare_variable, Define_pml_rules, Net_structure, Print_txt_pml โค้ดดังนี้

```
Sub main()
    Dim PNML_dir, pml_dir As String
    PNML_dir = ThisWorkbook.Sheets("Run").Cells(1, 3).Value
    pml_dir = ThisWorkbook.Sheets("Run").Cells(2, 3).Value
    Sheet_Run
    Sheet_Read_PNML
If PNML_dir <> "" And pml_dir <> "" Then
    Declare_variable
    Define_pml_rules
    Net_structure
    Print_txt_pml
ElseIf PNML_dir = "" Then
    MsgBox ("Plese add PNML path at cell C1")
Else
    MsgBox ("Plese add Promela path at cell C2")
End If
End Sub
```

1.1.2 Sub Read_PNML()

```
Sub Read_PNML()
    Dim PNML_path As String
    Dim RunWorkBook As Workbook
    Set RunWorkBook = ActiveWorkbook

    PNML_path = RunWorkBook.Sheets("Run").Range("C" & 1).Value

    If PNML_path = "" Then
    MsgBox ("Please add PNML path")
    End If
    'Sheet_Run
    Set oXMLFile = CreateObject("Microsoft.XMLDOM")
        XmlFileName = PNML_path
        oXMLFile.Load (XmlFileName)

'----------------------------------------
    '---Read_PNML_Declare_varable---
    Dim mainWorkBook As Workbook
    Set mainWorkBook = ActiveWorkbook

    '//Declare variable section: Node Place Get initialMarking token value and Place
name
    'Reading the Attributes // Get Place name
    Set PlaceNodes_Attribute = oXMLFile.SelectNodes("/pnml/place")
    Set initialMarkingNodes = oXMLFile.SelectNodes("/pnml/place/initialMarking/text()")
    For i = 0 To (PlaceNodes_Attribute.Length - 1)
        PlaceAttributes_pml(i) = PlaceNodes_Attribute(i).getAttribute("id")
        mainWorkBook.Sheets("Read_PNML").Range("M" & i + 2).Value =
PlaceAttributes_pml(i)
    Next

    For i = 0 To (PlaceNodes_Attribute.Length - 1)
        initialMarking_pml(i) = initialMarkingNodes(i).NodeValue
        mainWorkBook.Sheets("Read_PNML").Range("N" & i + 2).Value =
initialMarking_pml(i)
    Next
    '-------------------------------------------------------------------------------

    '//Declare variable section: Node Transition Get transition time min/max value
    'Reading the Attributes // Get transition name
    Set Nodes_Attributetransition = oXMLFile.SelectNodes("/pnml/transition")
    For i = 0 To (Nodes_Attributetransition.Length - 1)
```

```
        TransitionAttributes_pml(i) = Nodes_Attributetransition(i).getAttribute("id")
        mainWorkBook.Sheets("Read_PNML").Range("O" & i + 2).Value =
TransitionAttributes_pml(i)
    Next

    Set transition_minNodes = oXMLFile.SelectNodes("/pnml/transition/min/text()")
    Set transition_maxNodes = oXMLFile.SelectNodes("/pnml/transition/max/text()")
    For i = 0 To (transition_minNodes.Length - 1)
        transition_min_pml(i) = transition_minNodes(i).NodeValue
        transition_max_pml(i) = transition_maxNodes(i).NodeValue
        mainWorkBook.Sheets("Read_PNML").Range("P" & i + 2).Value =
transition_min_pml(i)
        mainWorkBook.Sheets("Read_PNML").Range("Q" & i + 2).Value =
transition_max_pml(i)
    Next


    '//Declare variable section: Get transition weight in/out at Promela
    Set transition_minNodes = oXMLFile.SelectNodes("/pnml/transition/wi/text()")
    Set transition_maxNodes = oXMLFile.SelectNodes("/pnml/transition/wo/text()")

    '//Get transition weight in/out value
    For i = 0 To (transition_minNodes.Length - 1)
        transition_wi_pml(i) = transition_minNodes(i).NodeValue
        transition_wo_pml(i) = transition_maxNodes(i).NodeValue
        mainWorkBook.Sheets("Read_PNML").Range("R" & i + 2).Value =
transition_wi_pml(i)
        mainWorkBook.Sheets("Read_PNML").Range("S" & i + 2).Value =
transition_wo_pml(i)
    Next

    '---Read_PNML_Net_structure---

    '-------------------------------------------------------------------------------
    '//Net Structure section: Node Net Get source and target
    'Reading the Attributes // Get Netlist name
    Set NetNodes_Attribute = oXMLFile.SelectNodes("/pnml/net")


    For i = 0 To (NetNodes_Attribute.Length - 1)
        NetAttributes_pml(i) = NetNodes_Attribute(i).getAttribute("id")
        NetTypeAttributes_pml(i) = NetNodes_Attribute(i).getAttribute("type")
        NetSourceAttributes_pml(i) = NetNodes_Attribute(i).getAttribute("source")
        NetTargetAttributes_pml(i) = NetNodes_Attribute(i).getAttribute("target")
```

```
        mainWorkBook.Sheets("Read_PNML").Range("A" & i + 2).Value =
NetAttributes_pml(i)
        mainWorkBook.Sheets("Read_PNML").Range("C" & i + 2).Value =
NetTypeAttributes_pml(i)
        mainWorkBook.Sheets("Read_PNML").Range("D" & i + 2).Value =
NetSourceAttributes_pml(i)
        mainWorkBook.Sheets("Read_PNML").Range("E" & i + 2).Value =
NetTargetAttributes_pml(i)
    Next

    '-------------------------------------------------------------------------------------
    Number_place = PlaceNodes_Attribute.Length
    Number_transition = Nodes_Attributetransition.Length
    Number_Net = NetNodes_Attribute.Length
    '-------------------------------------------------------------------------------------
End Sub
```

1.1.3 Sub Declare_variable()

```
Sub Declare_variable()
    Read_PNML 'Call other sub for Read_PNML_Declare_varable
    Dim Palce_pml() As Variant 'pml Declare_varable section
    Dim Tmin_pml() As Variant 'pml Declare_varable section
    Dim Tmax_pml() As Variant 'pml Declare_varable section
    Dim Twi_pml() As Variant 'pml Declare_varable section
    Dim Two_pml() As Variant 'pml Declare_varable section
    ReDim Palce_pml(Number_place)
    ReDim Tmin_pml(Number_transition)
    ReDim Tmax_pml(Number_transition)
    ReDim Twi_pml(Number_transition)
    ReDim Two_pml(Number_transition)
    Dim Pn, Pn0 As String

    For i = 0 To (Number_place - 1)
        Palce_pml(i) = "int " + CStr(PlaceAttributes_pml(i)) + "token=" +
CStr(initialMarking_pml(i)) + ";" '//Write txt - Promela declare place and initial marking
    Next

    For i = 0 To (Number_transition - 1)
        Tmin_pml(i) = "int " + CStr(TransitionAttributes_pml(i)) + "time_min=" +
CStr(transition_min_pml(i)) + ";" '//Write txt - Promela declare time min of transition
        Tmax_pml(i) = "int " + CStr(TransitionAttributes_pml(i)) + "time_max=" +
CStr(transition_max_pml(i)) + ";" '//Write txt - Promela declare time max of transition
```

```
        Twi_pml(i) = "int " + CStr(TransitionAttributes_pml(i)) + "wi=" +
CStr(transition_wi_pml(i)) + ";"  '//Write txt - Promela declare transition_weight_input
        Two_pml(i) = "int " + CStr(TransitionAttributes_pml(i)) + "wo=" +
CStr(transition_wo_pml(i)) + ";"  '//Write txt - Promela declare
transition_weight_output
    Next

    ThisWorkbook.Sheets("Promela").Cells(1, 1).Value = "Declare variable"
    ThisWorkbook.Sheets("Promela").Cells(1, 1).Interior.ColorIndex = 3
        ThisWorkbook.Sheets("Promela").Cells(2, 1).Value = Join(Palce_pml, vbLf)
        ThisWorkbook.Sheets("Promela").Cells(3, 1).Value = Join(Tmin_pml, vbLf)
        ThisWorkbook.Sheets("Promela").Cells(4, 1).Value = Join(Tmax_pml, vbLf)
        ThisWorkbook.Sheets("Promela").Cells(5, 1).Value = Join(Twi_pml, vbLf)
        ThisWorkbook.Sheets("Promela").Cells(6, 1).Value = Join(Two_pml, vbLf)
End Sub

Sub Define_pml_rules()

    Dim Define_time_behavior, Define_token_flow_behavior As String
    Dim Number_transition_list As Integer
    Number_transition_list = ThisWorkbook.Sheets("Read_PNML").Cells(Rows.Count,
"O").End(xlUp).Row
    Dim Tx() As Variant 'pml Define rule: varable for update timer each transition
    ReDim Tx(Number_transition_list - 1)
    Dim Tx_out() As Variant
    ReDim Tx_out(Number_transition_list - 1)
    Dim Timer, Timer_cnt, Timer_cnt_out As String
    '--Define time behavior---------------------------------

    For i = 1 To Number_transition - 1
        Tx(i) = "Tx" + CStr(i)
        Tx_out(i) = Tx(i) + "=" + Tx(i) + "+Tlim; "
    Next

    Timer = Join(Tx, ",")
    Timer = Left(Timer, Len(Timer) - 1)

    Timer_cnt = Join(Tx, "+Tlim; ")
    Timer_cnt = Right(Timer_cnt, Len(Timer_cnt) - 7)
    'ThisWorkbook.Sheets("Promela").Cells(4, 4).Value = Timer_cnt
    Timer_cnt_out = Join(Tx_out, " ")
```

'ThisWorkbook.Sheets("Promela").Cells(3, 4).Value = "#define timeCntTran(Tlim" + Timer + ") " + Timer_cnt_out '#define timeCntTran(Tlim,Tx1,Tx2,Tx3) Tx1=Tx1+Tlim; Tx2=Tx2+Tlim; Tx3=Tx3+Tlim;
'ThisWorkbook.Sheets("Promela").Cells(5, 4).Value = Timer_cnt_out

ThisWorkbook.Sheets("Promela").Cells(2, 4).Value = "/*2. Define time behavior*/" + vbCrLf + "#define timeCntSys(Tsys,Tlim) Tsys= Tsys+Tlim /*time_elapsed*/" + vbCrLf + "#define timeCntTran(Tlim" + Timer + ") " + Timer_cnt_out
Define_time_behavior = ThisWorkbook.Sheets("Promela").Cells(2, 4).Value
Define_token_flow_behavior = ThisWorkbook.Sheets("Promela").Cells(3, 4).Value
ThisWorkbook.Sheets("Promela").Cells(2, 3).Value = Define_time_behavior + vbCrLf + Define_token_flow_behavior
'---
End Sub

1.1.4 Sub Net_structure()

```
Sub Net_structure()
Dim NetList, Net_type, Net_pml, NetSource, NetTarget As String 'pml Net structure section
Dim Net_pml_Nondeterministic, Net_pml_Nondeterministic_previous, NetType_Nondeterministic_previous As String
Dim Net_pml_Or, Net_pml_Or_previous As String
Dim Net_pml_tmp As String
Dim Src, Tran, Tar As String
Dim TransitionList(10) As String
Dim Tran_other As String 'T_timer
Dim timer1, timer2, timer3 As String 'T_timer
Dim NetList_Var() As String 'Split_String p1->t2_>p3
Dim Src_Var() As String 'Split_String
Dim Tar_Var() As String 'Split_String
Dim Src_sub_Var() As String 'Split_String
Dim Tar_sub_Var() As String 'Split_String
Dim Tran_other_Var() As String 'Split_String
Dim pml_time_elapsed_min, pml_time_elapsed_max As String
Dim pml_timeCntSys_min, pml_timeCntSys_max As String
Dim pml_timeCntTran_min, pml_timeCntTran_max As String
Dim pml_Src, pml_Tar As String
Dim pml_Net_min, pml_Net_max As String

Read_PNML
'Read_PNML_Net_structure
'---GetTransition all name-----------------------------------------------
```

```
ThisWorkbook.Sheets("Read_PNML").Cells(1, 7).Value = "Promela Time Petri net
structure"
ThisWorkbook.Sheets("Read_PNML").Cells(1, 7).Interior.ColorIndex = 3
'----Number_transition = reads from PNML--------
For j = 2 To (Number_transition + 1)
TransitionList(j - 2) = ThisWorkbook.Sheets("Read_PNML").Cells(j, 15).Value
'ThisWorkbook.Sheets("Read_PNML").Cells(j, 11).Value = TransitionList(j - 2)
Next
'-------------------------------------------------------------------------
'Net Type selected
'For i = 2 To (Number_net + 1)
'Number_Net = 7
For i = 2 To (Number_Net + 1)

NetList = ThisWorkbook.Sheets("Read_PNML").Cells(i, 1).Value
NetType = ThisWorkbook.Sheets("Read_PNML").Cells(i, 3).Value
NetSource = ThisWorkbook.Sheets("Read_PNML").Cells(i, 4).Value
NetTarget = ThisWorkbook.Sheets("Read_PNML").Cells(i, 5).Value

If NetList <> "" Then
NetList_Var = Split(NetList, "->")
Src_Var = Split(NetSource, "(")
Tar_Var = Split(NetTarget, "(")

Src = NetList_Var(0) + "token"
Tran = NetList_Var(1)
Tar = NetList_Var(2) + "token"
End If

If NetType = "Concurrent" Then
    Tar_sub_Var = Split(Tar, ",")
    'ThisWorkbook.Sheets("Read_PNML").Cells(i, 6).Value = Tar_sub_Var(1)
    Tar = Tar_sub_Var(0) + "token," + Tar_sub_Var(1)
ElseIf NetType = "And" Then
    'Src_sub_Var = Split(Src, ",")
    Src_sub_Var = Split(NetSource, ",")
    Src = Src_sub_Var(0) + "token," + Src_sub_Var(1)
End If

'ThisWorkbook.Sheets("Read_PNML").Cells(i, 10).Value = Src
'ThisWorkbook.Sheets("Read_PNML").Cells(i, 11).Value = Tran
'ThisWorkbook.Sheets("Read_PNML").Cells(i, 12).Value = Tar

'---Find list of Other Transition--------------------------------
```

```
If Tran <> TransitionList(0) Then
Tran_other = TransitionList(0)
Else
Tran_other = ""
End If
If Tran <> TransitionList(1) Then
Tran_other = Tran_other + TransitionList(1)
End If
If Tran <> TransitionList(2) Then
Tran_other = Tran_other + TransitionList(2)
End If
If Tran <> TransitionList(3) Then
Tran_other = Tran_other + TransitionList(3)
End If
'--Print transition other---------------------------------------
'ThisWorkbook.Sheets("Read_PNML").Cells(i, 8).Value = Tran_other

Tran_other_Var = Split(Tran_other, "t")
timer1 = "t" + Tran_other_Var(1) + "timer,"
timer2 = "t" + Tran_other_Var(2) + "timer,"
timer3 = "t" + Tran_other_Var(3) + "timer"
'ThisWorkbook.Sheets("Read_PNML").Cells(i, 10).Value = timer1
'ThisWorkbook.Sheets("Read_PNML").Cells(i, 11).Value = timer2
'ThisWorkbook.Sheets("Read_PNML").Cells(i, 12).Value = timer3
'-------------------------------------------------------------
'pml_time_elapsed_min = "::time_elapsed>" + Tran + "time_min-1&&" + Tar +
"token>0->"
'pml_time_elapsed_max = "::time_elapsed<" + Tran + "time_max+1&&" + Tar +
"token>0->"
pml_time_elapsed_min = "::time_elapsed>" + Tran + "time_min-1&&" + Src + ">0->"
pml_time_elapsed_max = "::time_elapsed<" + Tran + "time_max+1&&" + Src + ">0->"
pml_timeCntSys_min = "timeCntSys(time_elapsed," + Tran + "time_min)->"
pml_timeCntSys_max = "timeCntSys(time_elapsed," + Tran + "time_max)->"
pml_timeCntTran_min = "timeCntTran(" + Tran + "time_min," + timer1 + timer2 +
timer3 + ")->"
pml_timeCntTran_max = "timeCntTran(" + Tran + "time_max," + timer1 + timer2 +
timer3 + ")->"
'pml_Src = Src_Var(0) + "(" + Tran + "wi," + Src + "token," + Tran + "timer," + Tran +
"time_min)->"
'pml_Tar = Tar_Var(0) + "(" + Tran + "wo," + Tar + "token," + Tran + "timer)}"
pml_Src = Src_Var(0) + "(" + Tran + "wi," + Src + "," + Tran + "timer," + Tran +
"time_min)->"
pml_Tar = Tar_Var(0) + "(" + Tran + "wo," + Tar + "," + Tran + "timer)}"
```

```
pml_Net_min = pml_time_elapsed_min + vbCrLf + "atomic{" + pml_timeCntSys_min
+ pml_timeCntTran_min + pml_Src + pml_Tar
pml_Net_max = pml_time_elapsed_max + vbCrLf + "atomic{" +
pml_timeCntSys_max + pml_timeCntTran_max + pml_Src + pml_Tar

    If NetType = "Series" Then
        Net_pml = "::if" + vbCrLf + pml_Net_min + vbCrLf + pml_Net_max + vbCrLf +
"fi;"
    ElseIf NetType = "Concurrent" Then
        Net_pml = "::if" + vbCrLf + pml_Net_min + vbCrLf + pml_Net_max + vbCrLf +
"fi;"
    ElseIf NetType = "Nondeterministic" Then
        'Rows(1).Insert shift:=xlShiftDown
        Net_pml = ""
        Net_pml_Nondeterministic = pml_Net_min + vbCrLf + pml_Net_max
        Net_pml = Net_pml_Nondeterministic
        Net_pml_Nondeterministic_previous =
ThisWorkbook.Sheets("Read_PNML").Cells(i - 1, 7).Value
        NetType_Nondeterministic_previous =
ThisWorkbook.Sheets("Read_PNML").Cells(i - 1, 3).Value
            If NetType_Nondeterministic_previous = "Nondeterministic" Then
                Net_pml = "::if" + vbCrLf + Net_pml_Nondeterministic_previous + vbCrLf +
Net_pml_Nondeterministic + vbCrLf + "fi;"
                ThisWorkbook.Sheets("Read_PNML").Cells(i - 1, 7).Value = ""
            End If
        'ThisWorkbook.Sheets("Promela").Cells(i, 7).Value = "::if" + Net_pml + vbCrLf +
"fi;"
    ElseIf NetType = "And" Then
        Net_pml = "::if" + vbCrLf + pml_Net_min + vbCrLf + pml_Net_max + vbCrLf +
"fi;"
    ElseIf NetType = "Or" Then
        Net_pml = ""
        Net_pml_Or = pml_Net_min + vbCrLf + pml_Net_max
        Net_pml = Net_pml_Or
        Net_pml_Or_previous = ThisWorkbook.Sheets("Read_PNML").Cells(i - 1, 7).Value
        NetType_Or_previous = ThisWorkbook.Sheets("Read_PNML").Cells(i - 1, 3).Value
            If NetType_Or_previous = "Or" Then
                Net_pml = "::if" + vbCrLf + Net_pml_Or_previous + vbCrLf + Net_pml_Or
+ vbCrLf + "fi;"
                ThisWorkbook.Sheets("Read_PNML").Cells(i - 1, 7).Value = ""
            End If
    Else
        Net_pml = ""
```

```
    End If
ThisWorkbook.Sheets("Read_PNML").Cells(i, 7).Value = Net_pml
'print_txt_Net
Next
End Sub
```

1.2.5 Sub Print_txt_pml()

```
Sub Print_txt_pml()
    Dim Pml_path As String
    Dim RunWorkBook As Workbook
    Set RunWorkBook = ActiveWorkbook

    Pml_path = RunWorkBook.Sheets("Run").Range("C" & 2).Value
    If Pml_path = "" Then
    MsgBox ("Please add Promela path")
    End If



'--------------------------------
'Dim Number_Net As Integer
Dim pml_declare_variable_P, pml_declare_variable_Tmin,
pml_declare_variable_Tmax, pml_declare_variable_wi, pml_declare_variable_wo As
String
Dim pml_syntax, pml_define_rules, pml_ltl As String
Dim pml_time_elapsed, pml_Timers As String

    ThisWorkbook.Sheets("Promela").Cells(1, 2).Value = "ltl"
    ThisWorkbook.Sheets("Promela").Cells(1, 2).Interior.ColorIndex = 6

    ThisWorkbook.Sheets("Promela").Cells(1, 3).Value = "Define rules"
    ThisWorkbook.Sheets("Promela").Cells(1, 3).Interior.ColorIndex = 6

'Open "D:\VM_Share\zFILM\7_Tool3_Hello\out.txt" For Output As #1
Open Pml_path For Output As #1

'---Declare_variable ------------------------------------------------------------
pml_declare_variable_P = ThisWorkbook.Sheets("Promela").Cells(2, 1).Value
pml_declare_variable_Tmin = ThisWorkbook.Sheets("Promela").Cells(3, 1).Value
pml_declare_variable_Tmax = ThisWorkbook.Sheets("Promela").Cells(4, 1).Value
pml_declare_variable_wi = ThisWorkbook.Sheets("Promela").Cells(5, 1).Value
pml_declare_variable_wo = ThisWorkbook.Sheets("Promela").Cells(6, 1).Value
Print #1, pml_declare_variable_P
Print #1, pml_declare_variable_Tmin
```

```
Print #1, pml_declare_variable_Tmax
Print #1, pml_declare_variable_wi
Print #1, pml_declare_variable_wo

pml_time_elapsed = "int time_elapsed=1;"
Print #1, pml_time_elapsed
pml_Timers = "int t1timer = t1time_min,t2timer = t2time_min,t3timer
=t3time_min,t4timer =t4time_min;"
Print #1, pml_Timers
'---Define ltl pml (if any)----------------------------------------------------
pml_ltl = ThisWorkbook.Sheets("Promela").Cells(2, 2).Value
Print #1, pml_ltl
'---Define rules pml------------------------------------------------------------
pml_define_rules = ThisWorkbook.Sheets("Promela").Cells(2, 3).Value
Print #1, pml_define_rules
'------------------------------------------------------------------------------
pml_syntax = "init{" + vbCrLf + "do" + vbCrLf
Print #1, pml_syntax

'--Net structure----------------------------------------------------------------
'Number_Net = 7
For i = 3 To (Number_Net + 2)
    Print #1, ThisWorkbook.Sheets("Read_PNML").Cells(i - 1, 7).Value
Next
'------------------------------------------------------------------------------
pml_syntax = "::else -> time_elapsed = time_elapsed+1;" + vbCrLf + "od" + vbCrLf +
"}"

Print #1, pml_syntax
    Close #1
    MsgBox "Promela DONE"
End Sub
```

ประวัติผู้เขียน

ชื่อ-สกุล	อรสุธี ชัยชมภู
วัน เดือน ปี เกิด	10 ตุลาคม 2532
สถานที่เกิด	จังหวัดลำปาง ประเทศไทย
วุฒิการศึกษา	สำเร็จการศึกษาปริญญาตรีวิศวกรรมศาสตร์ ภาควิชาวิศวกรรมอุตสาหการและเมคคาทรอนิกส์ สาขาวิศวกรรมเมคคาทรอนิกส์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี ปีการศึกษา 2554 และเข้าศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต ภาควิชาวิศวกรรมคอมพิวเตอร์ สาขาวิศวกรรมซอฟต์แวร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อ พ.ศ. 2558
ที่อยู่ปัจจุบัน	34/3 ถนนวังขวา ตำบลสบตุ๋ย อำเภอเมือง จังหวัดลำปาง 52100
ผลงานตีพิมพ์	“Transformation of Time Petri Net into Promela” ในงานประชุมวิชาการ The 11th International Conference on Telecommunication Systems, Services, and Applications (TSSA 2017)
รางวัลที่ได้รับ	-