การแปลงไทมด์ออโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึม

นายถิรวัตร สุตาลังกา

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2566

Transforming Probabilistic Timed Automata to PRISM Code

Mr. Thirawat Sutalungka

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 2023

หัวข้อวิทยานิพนธ์	การแปลงไทมด์ออโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึม
โดย	นายถิรวัตร สุตาลังกา
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

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

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(รองศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา)

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

..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.เนื่องวงศ์ ทวยเจริญ)

..... กรรมการภายนอกมหาวิทยาลัย
(ผู้ช่วยศาสตราจารย์ ดร.ชานนท์ เดชสุภา)

ถิรวัตร สุตาลังกา : การแปลงไทมด์ออโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึม. (Transforming Probabilistic Timed Automata to PRISM Code) อ.ที่ปรึกษาหลัก : รศ. ดร.วิวัฒน์ วัฒนาวุฒิ

วิทยานิพนธ์ฉบับนี้มุ่งศึกษากระบวนการแปลงไทมด์ออโตมาตาความน่าจะเป็นเป็นรหัสปริซึม ซึ่งมีความสำคัญในการวิเคราะห์ระบบเวลาจริงและระบบที่มีความซับซ้อนด้วยความน่าจะเป็น การวิจัยนี้เสนอกฎการแปลงสำหรับแปลงไทมด์ออโตมาตาจากเอกซ์เอ็มแอลเป็นรหัสปริซึม ,ช่วยในการวิเคราะห์และตรวจสอบคุณลักษณะที่ซับซ้อนของระบบ นอกจากนี้ วิทยานิพนธ์ยังได้พัฒนากระบวนการแปลงที่มีความสอดคล้องทางความหมาย ช่วยให้สามารถแปลงแบบจำลองที่ออกแบบด้วย UPPAAL ในรูปแบบเอกซ์เอ็มแอลไปยังรหัสปริซึมได้อย่างเหมาะสม

ผลลัพธ์จากการศึกษานี้มีความสำคัญในการประยุกต์ใช้ทฤษฎีในแบบจำลองจริงและนำไปสู่การประยุกต์ใช้งานที่มีประสิทธิผลมากขึ้นในหลากหลายด้าน วิทยานิพนธ์ยังนำเสนอการพัฒนาเครื่องมือสำหรับการแปลงรหัสที่มีประสิทธิภาพ ซึ่งใช้ภาษาจาวาและมีความยืดหยุ่นในการปรับใช้กับระบบต่างๆ การทดสอบเครื่องมือนี้พบว่าสามารถแปลงแบบจำลองได้อย่างถูกต้องและครบถ้วน รวมถึงการจัดการกับข้อผิดพลาดที่อาจเกิดขึ้น การทดสอบครอบคลุมแสดงให้เห็นถึงความสามารถของเครื่องมือในการวิเคราะห์และทดสอบแบบจำลองที่มีความซับซ้อน ลดเวลาและความพยายามในการวิเคราะห์แบบจำลองด้วยตนเอง และทำให้มั่นใจได้ว่าผลลัพธ์ที่ได้มีความถูกต้องและเชื่อถือได้ เครื่องมือนี้มีศักยภาพในการเป็นเครื่องมือมาตรฐานสำหรับการวิเคราะห์แบบจำลองไทมด์ออโตมาตาความน่าจะเป็นในอนาคต ช่วยจัดการกับความต้องการที่ซับซ้อนของระบบในโลกจริง

สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่อนิสิต .....

ปีการศึกษา 2566 ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

6272041021 : MAJOR SOFTWARE ENGINEERING

KEYWORD: Probabilistic Timed Automata, PRISM Code Transformation, Real-time Systems Analysis, Model Verification

Thirawat Sutalungka : Transforming Probabilistic Timed Automata to PRISM Code. Advisor: Assoc. Prof. WIWAT VATANAWOOD, Ph.D.

This dissertation focuses on studying the process of transforming timed automata probabilities into prism codes, which is important in analyzing real-time systems and complex systems with probabilities. The research proposes transformation rules for converting timed automata from XML to prism code, aiding in the analysis and verification of complex system characteristics. Additionally, the dissertation develops a semantically consistent transformation process, enabling the appropriate conversion of models designed with UPPAAL in XML format to prism code.

The outcomes of this study are significant for applying theory in real-world modeling and lead to more effective applications in various fields. The thesis also presents the development of an efficient code transformation tool using Java, offering flexibility for adaptation in different systems. Testing of this tool has shown that it can accurately and completely transform models, including managing potential errors. Comprehensive testing demonstrated the tool's ability to analyze and test complex models, reducing the time and effort in manual model analysis, and ensuring the accuracy and reliability of the results. This tool has the potential to become a standard tool for analyzing probabilistic timed automata models in the future, helping to manage the complex demands of real-world systems.

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

Academic Year: 2023 Advisor's Signature .....

กิตติกรรมประกาศ

ข้าพเจ้าขอแสดงความขอบคุณอย่างสูงสุดต่อ รศ.ดร.วิวัฒน์ วัฒนาวุฒิ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ได้ให้คำแนะนำที่ล้ำค่า การสนับสนุนอย่างไม่หยุดหย่อน และเป็นแรงบันดาลใจทางวิชาการอย่างยิ่ง ตลอดจนคอยดูและการทำวิทยานิพนธ์ของเข้าเจ้าจนลุล่วงสำเร็จได้ด้วยดี

ขอขอบคุณคณะกรรมการสอบ รศ.ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา ผศ.ดร.เนื่องวงศ์ ทวยเจริญ และผศ.ดร.ชานนท์ เดชสุภา ที่ได้ให้คำปรึกษาข้อเสนอแนะ และการวิพากษ์ที่เป็นประโยชน์อย่างมากต่อการพัฒนาวิทยานิพนธ์นี้

ขอบคุณครอบครัวของข้าพเจ้าที่เป็นหลักของชีวิตและแรงบันดาลใจที่ยิ่งใหญ่ สำหรับความรัก ความเข้าใจ การสนับสนุนที่ไม่มีวันสิ้นสุด และการอดทนต่อเวลาและความพยายามที่ข้าพเจ้าได้ลงทุนในการศึกษานี้

ขอขอบคุณเพื่อนๆ ที่มหาวิทยาลัยซึ่งเป็นเพื่อนร่วมทางในการศึกษาและเป็นแหล่งของความสุขและการสนับสนุนทางจิตใจในระหว่างการทำวิทยานิพนธ์นี้

ขอขอบคุณผู้ที่มีส่วนร่วมทุกท่านที่ไม่ได้กล่าวถึงโดยตรงในข้อความนี้ ทุกคำแนะนำ ความช่วยเหลือ และการสนับสนุนของคุณมีค่าอย่างยิ่งต่อความสำเร็จของงานวิจัยนี้.

ผลงานวิจัยนี้เป็นผลมาจากความร่วมมือและการสนับสนุนของทุกฝ่ายที่เกี่ยวข้อง ขอให้วิทยานิพนธ์นี้เป็นเครื่องหมายแห่งความขอบคุณที่ข้าพเจ้ามีต่อทุกท่าน

ถิรวัตร สุตาลังกา

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญรูปภาพ

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในยุคปัจจุบันที่เทคโนโลยีก้าวหน้า การพัฒนาระบบที่มีข้อจำกัดด้านเวลากลายเป็นสิ่งที่สำคัญและจำเป็น ซึ่งพิสูจน์ได้จากความต้องการระบบที่สามารถตอบสนองได้อย่างรวดเร็วและแม่นยำ เช่น ระบบควบคุมประตูรถไฟฟ้า หรือระบบควบคุมเครื่องจักรอุตสาหกรรม เป็นต้น[1] [2] การจำลองระบบเป็นขั้นตอนหลักในการออกแบบและทดสอบ ซึ่งช่วยในการอธิบายพฤติกรรมและการตรวจสอบหาข้อผิดพลาดของระบบ

อย่างไรก็ตาม ความท้าทายหนึ่งที่เกิดขึ้นคือการจำลองอาจไม่สามารถทำนายหรือแสดงพฤติกรรมที่แท้จริงของระบบได้ทั้งหมด ซึ่งมีความเกี่ยวข้องโดยเฉพาะในสถานการณ์ที่มีความน่าจะเป็นสูงในการต้องตัดสินใจภายใต้ข้อจำกัดของเวลา ตัวอย่างเช่น ที่อาจนำไปสู่การเปลี่ยนแปลงสถานะของระบบที่ไม่พึงประสงค์ [3] ดังนั้นในการวิจัยและพัฒนาระบบที่มีข้อจำกัดด้านเวลาจำเป็นต้องมีการพิจารณาอย่างรอบคอบและมีเครื่องมือที่เหมาะสมในการประเมินความน่าจะเป็นและความเสี่ยง โดยไทมด์ออโตมาตาความน่าจะเป็น (Probabilistic Timed Automata: PTA)[9] เป็นเครื่องมือที่มีความสำคัญอย่างยิ่งในการวิเคราะห์และจำลองระบบที่มีคุณสมบัติทั้งในด้านความน่าจะเป็นและเวลาที่แน่นอน สามารถจำลองสถานการณ์ที่ซับซ้อนที่ผลลัพธ์ขึ้นอยู่ไม่เพียงแค่กับลำดับของเหตุการณ์ แต่ยังรวมถึงความไม่แน่นอนและข้อจำกัดด้านเวลาภายในระบบ นี่ทำให้วิศวกรและนักวิจัยสามารถทำการประเมินความเสี่ยง วิเคราะห์ความน่าเชื่อถือ และตัดสินใจภายใต้สถานการณ์ที่ไม่แน่นอนได้อย่างแม่นยำยิ่งขึ้น

ในงานวิจัยนี้นำเอาเครื่องมือที่สำคัญมาใช้คือ UPPAAL ซอฟต์แวร์ที่สร้างแบบจำลองกราฟฟิคเพื่อวิเคราะห์ระบบเวลาจริง UPPAAL[4] มีความสามารถในการสร้างแผนภาพแบบจำลองซึ่งเป็นโครงสร้างพื้นฐานที่สื่อถึงสถานะและการเปลี่ยนสถานะของระบบตามเวลาที่กำหนด[2] การวิจัยนี้เน้นไปที่การสร้างแบบจำลองของพฤติกรรมที่เกิดจากความไม่แน่นอนและความน่าจะเป็น ซึ่งเป็นสิ่งจำเป็นในการแสดงการทำงานของระบบที่ซับซ้อนในโลกจริง

การนำความน่าจะเป็นมาใช้ในแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นสามารถจำลองสถานการณ์ที่มีการเปลี่ยนแปลงสถานะตามความน่าจะเป็นได้ และเพื่อการวิเคราะห์คุณสมบัติเชิงปริมาณที่เกี่ยวข้องกับความน่าจะเป็นของแบบจำลองที่สร้างด้วย UPPAAL การใช้แบบจำลองเดียวกันนี้ร่วมกับเครื่องมือที่มีความเชี่ยวชาญในการวิเคราะห์พฤติกรรมแบบสุ่มของระบบ จะช่วยให้

การทำความเข้าใจและประเมินพฤติกรรมที่ซับซ้อนของระบบได้ดีขึ้น และเปิดทางสู่การพัฒนาระบบที่ทั้งปลอดภัยและมีประสิทธิภาพสูง

การตรวจสอบและวิเคราะห์พฤติกรรมของระบบที่ซับซ้อนซึ่งแสดงความไม่แน่นอนและความน่าจะเป็นนั้นสำคัญมากในการพัฒนาทางวิศวกรรมซอฟต์แวร์ ในการศึกษานี้ผู้วิจัยเลือกใช้ PRISM Model Checker เครื่องมือที่ได้รับการยอมรับอย่างกว้างขวางสำหรับการตรวจสอบแบบจำลองความน่าจะเป็นปริซึม[5] ใช้ภาษาปริซึม (PRISM Language) เพื่ออธิบายแบบจำลองความน่าจะเป็นในรูปแบบของปฏิกิริยาแบบโมดูล[6] ซึ่งเป็นภาษาที่ช่วยให้สามารถอธิบายและวิเคราะห์พฤติกรรมที่เกี่ยวข้องกับเวลาจริงและความน่าจะเป็นได้อย่างชัดเจนและเชิงรูปนัย ตัวอย่างการใช้งานของปริซึมได้แก่ การคำนวณเวลาที่คาดหวังสูงสุดสำหรับการสิ้นสุดของโปรโตคอลการสื่อสาร หรือการประเมินปริมาณพลังงานสูงสุดที่ใช้ในการส่งข้อความผ่านช่องทางที่ไม่น่าเชื่อถือ การวิเคราะห์เหล่านี้มีคุณค่าอย่างยิ่งในการเพิ่มประสิทธิภาพจากข้อมูลวิเคราะห์เชิงปริมาณที่ได้

การสร้างแบบจำลองที่มีความสามารถในการตรวจสอบคุณสมบัติเชิงปริมาณภายในระบบที่มีความน่าจะเป็นเป็นงานที่ท้าทาย หนึ่งในอุปสรรคสำคัญคือความจำเป็นในการมีความเข้าใจที่ลึกซึ้งเกี่ยวกับโครงสร้างภาษาและการแปลแผนภาพเวลาไปยังรูปแบบที่สามารถใช้ในเครื่องมือ ปริซึม ข้อผิดพลาดในการตีความอาจเกิดขึ้นง่ายเมื่อมีการโอนย้ายรายละเอียดจากแบบจำลอง UPPAAL ไปยังปริซึม ซึ่งอาจนำไปสู่ผลลัพธ์ที่ไม่ถูกต้องหรือการวิเคราะห์ที่ไม่เพียงพอ[8]

เพื่อแก้ไขปัญหานี้และลดภาระการทำงานซ้ำซ้อนในการสร้างแบบจำลองที่แตกต่างกันใน UPPAAL และปริซึมงานวิจัยนี้ได้ตั้งเป้าหมายในการพัฒนาเครื่องมือที่สามารถทำการแปลงไทม์ออโตมาตาความน่าจะเป็นไปยังรหัสปริซึมได้อย่างราบรื่นและอัตโนมัติ โดยเครื่องมือนี้จะรองรับการวิเคราะห์คุณสมบัติเชิงปริมาณที่ซับซ้อน ทำให้ผู้วิจัยสามารถตรวจสอบคุณสมบัติต่างๆ ได้ดียิ่งขึ้น เช่น การคำนวณความน่าจะเป็นสูงสุดหรือต่ำสุดที่จะบรรลุเป้าหมาย รวมถึงการประเมินรางวัลที่คาดหวังในระบบ นวัตกรรมนี้จะช่วยขยายขอบเขตในการวิเคราะห์แบบจำลองของระบบที่มีความสลับซับซ้อน และเพิ่มความแม่นยำในการออกแบบและการตรวจสอบระบบให้สอดคล้องกับความต้องการของผู้ใช้และผู้มีส่วนได้ส่วนเสียในการพัฒนาระบบ

1.2 วัตถุประสงค์ของการวิจัย

1) ออกแบบกฎการแปลงแบบจำลองไทม์ออโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึมที่สนับสนุนการวิเคราะห์คุณสมบัติเชิงปริมาณ
2) พัฒนาเครื่องมือการแปลงแบบจำลองไทม์ออโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึมที่สนับสนุนการวิเคราะห์คุณสมบัติเชิงปริมาณ

1.3 ขอบเขตของการวิจัย

1) เอกสารเอกซ์เอ็มแอลไทมด์ออโตมาตาความน่าจะเป็นที่นำเข้าเป็นไปตามเครื่องมือ UPPAAL
2) สัญลักษณ์ของไทมด์ออโตมาตาความน่าจะเป็นรองรับตำแหน่ง, ค่ายืนยง, แทรนซิชัน, ตัวป้องกัน, การอัปเดต, จุดแยกร่วมกับน้ำหนัก
3) เอกสารผลลัพธ์ปริซึมของไทมด์ออโตมาตาความน่าจะเป็นสามารถนำมาใช้ผ่านตัวตรวจสอบแบบจำลองความน่าจะเป็นปริซึมได้
4) ใช้รหัสปริซึมครอบคลุมการประกาศโมดูล, การกำหนดตัวแปรตำแหน่ง, การกำหนดตัวแปรนาฬิกา, การกำหนดค่ายืนยง, การกำหนดสถานะเริ่มต้น, การกำหนดคำสั่งร่วมกับตัวป้องกัน , การกำหนดคำสั่งร่วมกับการอัปเดต, การกำหนดคำสั่งร่วมกับความน่าจะเป็น
5) พัฒนาเครื่องมือการแปลงแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึมที่สนับสนุนการตรวจสอบความน่าจะเป็น

1.4 ขั้นตอนในการวิจัย

1) ศึกษาโครงสร้างของแบบจำลองไทมด์ออโตมาตาความน่าจะเป็น เพื่อนำมาใช้กับการออกแบบกฎการแปลงในการทำวิทยานิพนธ์นี้
2) ศึกษาโครงสร้างของเอกสารเอกซ์เอ็มแอลที่ได้จากการวาดแบบจำลองในเครื่องมือ UPPAAL เพื่อใช้เป็นข้อมูลนำเข้าในการทำวิทยานิพนธ์นี้
3) ศึกษาการใช้เครื่องมือตรวจสอบแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นในปริซึมเพื่อทำความเข้าใจโครงสร้างของรหัสปริซึมทั้งหมด
4) สร้างกฎการแปลงภาษาปริซึมจากโครงสร้างของแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นในรูปแบบเอกสารเอกซ์เอ็มแอลจากเครื่องมือ UPPAAL
5) พัฒนาเครื่องมือแปลงแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นในรูปแบบเอกสารเอกซ์เอ็มแอลไปสู่รหัสปริซึม
6) ทดสอบความถูกต้องของเครื่องมือ ตั้งแต่เริ่มการทำงานไปจนถึงการส่งออกรหัสปริซึมที่แปลงจากแบบจำลองได้อย่างสมบูรณ์
7) สรุปผลของโครงงาน รวบรวมคำชี้แนะตลอดจนคำแนะนำต่างๆ เพื่อใช้เป็นข้อมูลในการต่อยอดและพัฒนาระบบ หรือประยุกต์ในโครงการอื่นต่อไป
8) จัดทำเอกสาร

บทที่ 2

ทฤษฎี และงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 ไทมด์ออโตมาตา (Timed Automata) [2]

ไทมด์ออโตมาตา (Timed Automata) เป็นทฤษฎีที่ใช้ในการสร้างแบบจำลองระบบที่มีความซับซ้อนซึ่งการทำงานขึ้นอยู่กับเวลา เป็นส่วนขยายของออโตมาตันที่มีสถานะและการเปลี่ยนแปลงสถานะที่ถูกควบคุมโดยนาฬิกาและเงื่อนไขด้านเวลา ไทมด์ออโตมาตาช่วยให้นักวิจัยและวิศวกรสามารถแสดงและตรวจสอบพฤติกรรมของระบบเวลาจริง (Real Time System) เช่น ระบบปฏิบัติการ ระบบควบคุมการผลิต และโปรโตคอลการสื่อสารที่ต้องการความแม่นยำด้านเวลาสูง

นอกจากนี้ ไทมด์ออโตมาตายังให้ความสามารถในการวิเคราะห์คุณสมบัติต่างๆ เช่น ความน่าจะเป็นของเหตุการณ์ต่างๆ ภายในระยะเวลาที่กำหนด การตรวจสอบความปลอดภัยที่เกี่ยวข้องกับเวลา และการวิเคราะห์ความสามารถในการทำงานภายใต้ข้อจำกัดเวลาที่เข้มงวด การใช้เครื่องมือและเทคนิคที่เกี่ยวข้องกับไทมด์ออโตมาตา เช่น UPPAAL หรือ TIMES ช่วยให้สามารถทดสอบและทำนายพฤติกรรมของระบบเวลาจริงได้อย่างแม่นยำก่อนที่จะนำไปใช้งานจริง

ในบริบทของการวิจัยการแปลงไทมด์ออโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึมการเข้าใจลึกซึ้งเกี่ยวกับไทมด์ออโตมาตามีความสำคัญอย่างยิ่ง การแปลงเช่นนี้ต้องรักษาความถูกต้องของพฤติกรรมเวลาและความน่าจะเป็นของระบบต้นฉบับเมื่อถูกแปลงไปยังรูปแบบใหม่ ความเข้าใจนี้จะช่วยให้การวิเคราะห์และการตรวจสอบโดยใช้ปริซึมนั้นเกิดความสมบูรณ์แบบและสามารถเชื่อถือได้ ซึ่งเป็นสิ่งจำเป็นสำหรับการออกแบบระบบที่มีประสิทธิภาพและปลอดภัย

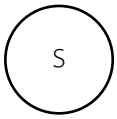
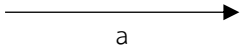
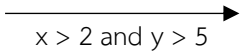
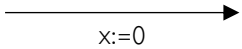
ไทมด์ออโตมาตาถูกพัฒนาขึ้นเพื่อตอบสนองความจำเป็นในการรูปแบบการทำงานของระบบที่มีเวลาเป็นปัจจัยสำคัญ แนวคิดนี้เริ่มต้นจากการทำงานของนักวิจัยในต้นทศวรรษ 1990 เมื่อพวกเขาตระหนักถึงข้อจำกัดของออโตมาตันดั้งเดิมที่ไม่สามารถจัดการกับความต้องการของระบบเวลาจริงได้ ไทมด์ออโตมาตาจึงถูกออกแบบมาเพื่อที่จะเพิ่มความสามารถในการแสดงและการคำนวณเงื่อนไขด้านเวลาเข้ากับระบบออโตมาตัน

การพัฒนาไทมด์ออโตมาตาได้นำไปสู่การวิจัยและนวัตกรรมใหม่ๆ มากมายในด้านการวิเคราะห์ระบบและการตรวจสอบแบบจำลอง ไทมด์ออโตมาตาได้รับการประยุกต์ใช้ในการวิเคราะห์ความปลอดภัยและความเชื่อถือได้ของระบบ เช่น ระบบควบคุมการบิน โปรโตคอลการสื่อสาร ระบบสุขภาพ และระบบอุตสาหกรรม การใช้งานไทมด์ออโตมาตาในการตรวจสอบแบบจำลองได้ช่วยให้สามารถคาดการณ์และหลีกเลี่ยงข้อผิดพลาดที่อาจเกิดขึ้นได้ในระบบเวลาจริง[4]

ตลอดระยะเวลาที่ผ่านมา เครื่องมือและเทคนิคสำหรับการทำงานกับไทมด์ออโตมาตาก็ได้รับการพัฒนาอย่างต่อเนื่อง ตัวอย่างเช่น UPPAAL และ TIMES ได้รับการออกแบบมาเพื่อให้สามารถสร้างแบบจำลอง จำลองการทำงาน และตรวจสอบคุณสมบัติต่างๆ ของระบบที่มีเงื่อนไขด้านเวลาเป็นส่วนสำคัญ นอกจากนี้การทำงานของไทมด์ออโตมาตายังถูกใช้เป็นพื้นฐานในการพัฒนาแบบจำลองความน่าจะเป็นที่มีเวลาเป็นส่วนสำคัญ (Probabilistic Timed Automata) ซึ่งเป็นการผสมผสานระหว่างความน่าจะเป็นและเวลาในการจำลองระบบ

ในไทมด์ออโตมาตา สถานะหรือตำแหน่งแทนสภาพต่างๆ ของระบบ และแทรนซิชันหรือการกระทำแสดงถึงการเปลี่ยนแปลงระหว่างสถานะเหล่านี้ ซึ่งถูกควบคุมโดยข้อจำกัดด้านเวลาที่เรียกว่าตัวป้องกัน การตั้งค่าใหม่อนุญาตให้ตั้งนาฬิกาเป็นศูนย์ สะท้อนเวลาใหม่สำหรับสถานะถัดไป โดยแบบจำลองไทมด์ออโตมาตามีองค์ประกอบของแบบจำลองตามตารางที่ 2.1 ดังนี้

ตาราง 2.1 แผนภาพองค์ประกอบของไทมด์ออโตมาตา [2]

สัญลักษณ์	ชื่อสัญลักษณ์	คำอธิบาย
	สถานะ (State) หรือ ตำแหน่ง (Location)	โหนด (Node) ที่บอกถึงสถานะของออโตมาตา โดยสามารถระบุเงื่อนไขเวลาในโหนด (Invariant) ที่เปลี่ยนสถานะได้ตามเงื่อนเวลาที่ระบุไว้
	แทรนซิชัน (Transition) หรือ การกระทำ (Action)	เส้นเชื่อมระหว่างโหนด ที่แสดงพฤติกรรมของระบบในการเปลี่ยนแปลงจากสถานะหนึ่งไปอีกสถานะหนึ่ง
	เงื่อนไขป้องกัน (Guards)	เงื่อนไขในการเปลี่ยนสถานะโดยระบุบนเส้นเชื่อม ทำหน้าที่ในการเปิดปิดกระแสของงานที่เข้าเงื่อนไขในขณะเกิดการเปลี่ยนแปลงสถานะของระบบ
	ตั้งค่าใหม่ (Reset)	การรีเซ็ตค่าของนาฬิกาใหม่ในขณะที่เกิดการเปลี่ยนแปลงสถานะของระบบ

การทำงานของไทมด์ออโตมาตานั้นเริ่มต้นด้วยการกำหนดค่านาฬิกาซึ่งถูกนับเป็นตัวเลขจำนวนเต็มและทำงานโดยเดินเครื่องอย่างต่อเนื่องโดยไม่มีการหยุดชะงัก ในขณะที่ออโตมาตันเริ่มทำงาน ค่าของนาฬิกาจะเพิ่มขึ้นอย่างเป็นประจำและอัตราที่ต่อเนื่องซึ่งส่งผลโดยตรงต่อการเปลี่ยนแปลงสถานะภายในระบบตามข้อจำกัดที่กำหนดด้วยเวลา

ในแต่ละสถานะของออโตมาตัน การเปลี่ยนแปลงสถานะจะเกิดขึ้นเมื่อเงื่อนไขด้านเวลาหรือ 'guards' ที่กำหนดไว้นั้นได้รับการปฏิบัติตาม ซึ่งอาจรวมถึงการรอจนกว่าเงื่อนไขเวลาจะถึงหรือเมื่อเหตุการณ์หนึ่งเกิดขึ้นในช่วงเวลาที่กำหนด เมื่อการเปลี่ยนแปลงสถานะเกิดขึ้น อาจมีการ 'reset'

นาฬิกาซึ่งหมายถึงการตั้งค่านาฬิกาให้เริ่มนับจากศูนย์อีกครั้ง เพื่อสะท้อนการเริ่มต้นเวลาใหม่สำหรับสถานะถัดไปที่ระบบจะเคลื่อนไป

การจำลองพฤติกรรมด้านเวลาด้วยไทมด์ออโตมาตาเป็นเครื่องมือที่มีความสำคัญในการออกแบบและการตรวจสอบระบบที่มีความต้องการความแม่นยำด้านเวลาสูง เช่น ระบบควบคุมอุตสาหกรรม ระบบนำทางอัตโนมัติ และระบบสื่อสารที่ต้องการการตอบสนองที่รวดเร็วและตรงเวลา นาฬิกาภายในไทมด์ออโตมาตาจึงไม่เพียงแต่ทำหน้าที่เป็นตัวนับเวลาแต่ยังเป็นกลไกสำคัญในการกำหนดพฤติกรรมและการตอบสนองของระบบตามเงื่อนไขเวลาที่ได้รับการกำหนดไว้อย่างละเอียด

ไทมด์ออโตมาตาเป็นรูปแบบทางคณิตศาสตร์ที่ใช้สำหรับการจำลองระบบที่พฤติกรรมของมันขึ้นอยู่กับเวลาไทมด์ออโตมาตาถูกนิยามเป็นทูเพิลซึ่งประกอบด้วยองค์ประกอบหลัก 6 ส่วน ได้แก่ $TA = \langle S, s_0, A, X, E, I \rangle$ โดยที่

S คือ เซตของสถานะทั้งหมดในออโตมาตา ซึ่งแทนด้วยจุดหรือโหนดที่ระบุสถานะหรือสภาพต่างๆ ของระบบที่จำลอง

s_0 คือ สถานะเริ่มต้นของออโตมาตา ซึ่งเป็นจุดเริ่มต้นที่ระบบจะเริ่มการทำงาน

A คือ เซตของเหตุการณ์หรือการกระทำ (actions) ที่สามารถเกิดขึ้นในระบบ แทนด้วยข้อความกำกับที่ติดกับเส้นเชื่อมระหว่างสถานะ

X คือ เซตของนาฬิกาซึ่งเป็นตัวแปรที่นับเวลาที่ผ่านไปในระบบและใช้ในการตรวจสอบเงื่อนไขเวลา

E คือ เซตของแทรนซิชัน (transitions) ซึ่งอธิบายการเปลี่ยนแปลงจากสถานะหนึ่งไปยังอีกสถานะหนึ่ง แทนด้วย {s, a, g, r, s'}ป ซึ่งหมายถึงการเปลี่ยนแปลงจากสถานะ s ไปยังสถานะ s' ผ่านเหตุการณ์ a โดยมี g เป็นเงื่อนไขป้องกัน (guards) ที่กำหนดเงื่อนไขด้านเวลา และ r เป็นชุดของนาฬิกาที่จะถูกรีเซ็ตเมื่อการเปลี่ยนแปลงเกิดขึ้น

I คือ เงื่อนไขเวลายืนยง (invariants) ที่กำหนดไว้สำหรับแต่ละสถานะ ซึ่งจะต้องเป็นจริงตลอดเวลาที่ระบบอยู่ในสถานะนั้นๆ

ไทมด์ออโตมาตาเป็นเครื่องมือที่มีความสำคัญในการวิเคราะห์ระบบเวลาจริงโดยให้ความสามารถในการระบุและตรวจสอบข้อจำกัดเกี่ยวกับเวลาในกระบวนการต่างๆ ของระบบ นอกจากนี้การใช้นาฬิกาและเงื่อนไขเวลายังช่วยให้สามารถกำหนดเงื่อนไขที่แม่นยำสำหรับการทำงานของระบบ เช่น การตั้งเวลาหมดเวลาสำหรับเหตุการณ์หรือการจำกัดเวลาตอบสนองสำหรับกระบวนการ ซึ่งทำให้ไทมด์ออโตมาตาเป็นเครื่องมือที่มีคุณค่าไม่เพียงแต่สำหรับการวิเคราะห์ทฤษฎี

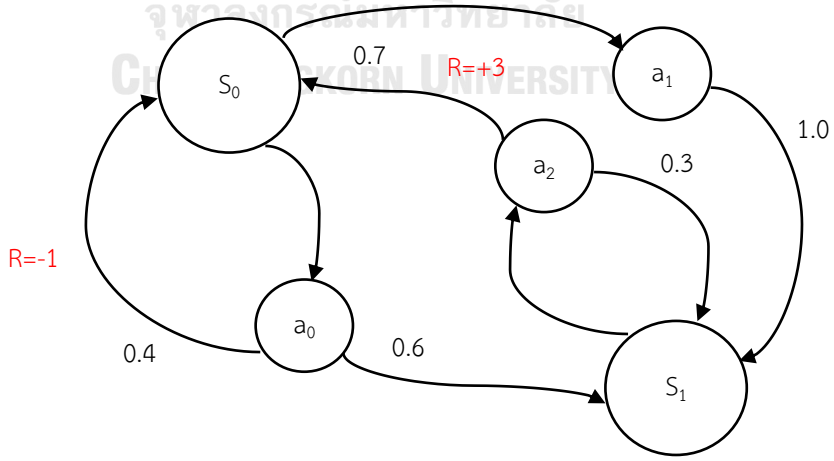
เท่านั้น แต่ยังรวมถึงการประยุกต์ใช้ในอุตสาหกรรมและการวิจัยเพื่อการออกแบบระบบที่ปลอดภัยและเชื่อถือได้

2.1.2 กระบวนการตัดสินใจของมาร์คอฟ (Markov decision process : MDP) [7]

กระบวนการตัดสินใจของมาร์คอฟ (MDP) คือแบบจำลองทางคณิตศาสตร์ที่ใช้สำหรับการอธิบายสถานการณ์ที่ผลลัพธ์สามารถถูกควบคุมได้ทั้งโดยความสุ่มและการตัดสินใจของผู้ปฏิบัติการ MDP ประกอบด้วยสถานะต่างๆ (states) การกระทำ (actions) และการเปลี่ยนแปลงสถานะ (state transitions) ที่เกิดจากการกระทำเหล่านั้น แต่ละการกระทำในสถานะที่กำหนดมีค่าตอบแทน (reward) ที่เกี่ยวข้อง ซึ่งอาจแตกต่างกันไปตามสถานะและการกระทำที่เลือก

MDP มีรากฐานมาจากงานของนักคณิตศาสตร์ชาวรัสเซีย Andrey Markov ที่สร้างห่วงโซ่มาร์คอฟ (Markov chains) ซึ่งเป็นกระบวนการสุ่มที่เวลาไม่ต่อเนื่องกันและทำให้เราสามารถวิเคราะห์สถานะต่อไปโดยพิจารณาเฉพาะสถานะปัจจุบัน MDP ถูกพัฒนาและนำมาใช้กว้างขวางในทศวรรษ 1950 โดยมีการนำไปใช้ในหลายด้าน เช่น คณิตศาสตร์ วิทยาศาสตร์คอมพิวเตอร์ วิศวกรรม เศรษฐศาสตร์ และวิทยาศาสตร์การจัดการ เป็นต้น

MDP มีประโยชน์อย่างมากในการศึกษาและการพัฒนาโซลูชั่นเพื่อปัญหาต่างๆ ที่ต้องการการตัดสินใจอย่างต่อเนื่อง รวมถึงการเพิ่มประสิทธิภาพในการแก้ไขปัญหาเหล่านั้น มันเป็นเครื่องมือหลักในการเขียนโปรแกรมแบบพลวัต (dynamic programming) และการเรียนรู้เชิงเสริม (reinforcement learning) ซึ่งเป็นสาขาของการเรียนรู้ของเครื่องจักร โดยแบบจำลองกระบวนการตัดสินใจของมาร์คอฟ สามารถแสดงตัวอย่างได้ดังรูปที่ 2.1



รูปที่ 2.1 ตัวอย่างแบบง่ายของกระบวนการตัดสินใจของมาร์คอฟ โดยมี 2 สถานะ 3 การกระทำ และ 2 รางวัล [7]

จากรปที่ 1 แสดงถึงกรณีที่ง่ายที่สุดของ MDP ที่มีเพียงสองสถานะและผู้ตัดสินใจสามารถทำการกระทำได้สามอย่าง ซึ่งนำไปสู่ผลลัพธ์ที่แตกต่างกันพร้อมกับรางวัลที่เกี่ยวข้อง ระบบมีสถานะ S_0

และ S_1 และการกระทำ a_0 , a_1 , a_2 เริ่มจาก S_0 ผู้ตัดสินใจสามารถเลือกการกระทำใดๆ ที่มีอยู่ หากเลือก a_0 ระบบอาจยังคงอยู่ที่ S_0 ด้วยความน่าจะเป็น 0.4 หรือเปลี่ยนไป S_1 ด้วยความน่าจะเป็น 0.6 การเปลี่ยนแปลงนี้สามารถนำผลตอบแทนมาสู่ผู้ตัดสินใจได้ ซึ่งในตัวอย่างนี้ถ้าเปลี่ยนไป S_0 จะได้รับผลตอบแทน -1 แต่ถ้าไป S_1 จะไม่ได้รับผลตอบแทนใดๆ

ฟังก์ชันดังนี้ $R_a(s, s')$ ระบุผลตอบแทนที่ได้รับจากการเปลี่ยนแปลงจากสถานะ s ไปยัง s' ด้วยการกระทำ a ฟังก์ชันการเปลี่ยนสถานะ $P_a(s, s')$ กำหนดความน่าจะเป็นในการเปลี่ยนสถานะเมื่อการกระทำ a เลือกในสถานะ s โดยมีผลต่อสถานะถัดไป S'และขึ้นอยู่กับสถานะปัจจุบัน S และการกระทำที่เลือกนี่ยืนยันคุณสมบัติของการไม่มีหน่วยความจำของมาร์คอฟ ซึ่งแสดงว่าสถานะถัดไปไม่ได้ขึ้นอยู่กับประวัติสถานะหรือการกระทำก่อนหน้านี้กล่าวอีกนัยคือ กระบวนการตัดสินใจของมาร์คอฟ เป็นไปตามคุณสมบัติของมาร์คอฟ

กระบวนการตัดสินใจของมาร์คอฟ (Markov Decision Process: MDP) ประกอบด้วยทูเพิลที่มีส่วนประกอบ 4 ส่วนหลัก คือ MDP = $\langle S, A, T, R \rangle$ โดยที่

S คือ (ปริภูมิสถานะ - State Space)

'S' แทนด้วยเซตของสถานะทั้งหมดในระบบ แต่ละสถานะใน 'S' คือการอธิบายสถานการณ์หรือสภาพแวดล้อมที่ระบบสามารถพบเจอได้

A คือ (ปริภูมิการกระทำ - Action Space)

'A' เป็นเซตของการกระทำที่เป็นไปได้ทั้งหมด ในแต่ละสถานะ s จากเซตของสถานะจะมีการกระทำ a จากเซตของการกระทำที่สามารถดำเนินการได้.

$T : S \times A \to \Pr(S)$ คือ (ฟังก์ชันการเปลี่ยนแปลงสถานะ - Transition Function)

'T' คือ ฟังก์ชันที่อธิบายความน่าจะเป็นของการเปลี่ยนจากสถานะหนึ่งไปยังอีกสถานะหนึ่งหลังจากการกระทำ สำหรับการกระทำ a ในสถานะ s ณ เวลา t ฟังก์ชันการเปลี่ยนแปลงสถานะจะกำหนดความน่าจะเป็นที่สถานะจะเปลี่ยนจาก s ไปยัง s' ในเวลา $t + 1$

$R : S \times A \to \mathbb{R}$ คือ (ฟังก์ชันผลตอบแทน - Reward Function)

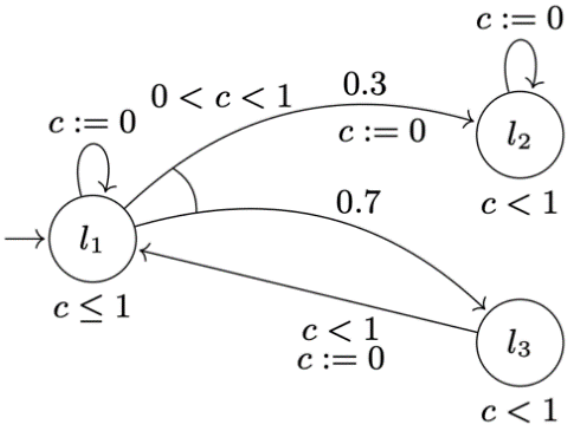
'R' คือ ฟังก์ชันที่กำหนดผลตอบแทนที่ได้รับเมื่อทำการกระทำ a ในสถานะ s ผลตอบแทนนี้สามารถเป็นค่าบวก ลบ หรือศูนย์ และมีความสำคัญในการกำหนดกลยุทธ์การตัดสินใจที่ดีที่สุด

2.1.3 ไทมด์ออโตมาตาความน่าจะเป็น (Probabilistic Timed Automata) [9]

ไทมด์ออโตมาตาความน่าจะเป็น (Probabilistic Timed Automata: PTAs) เป็นแนวคิดที่นำเสนอการผสมผสานระหว่างทฤษฎีไทมด์ออโตมาตา (Timed Automata: TA) และกระบวนการตัดสินใจของมาร์คอฟ (Markov Decision Processes: MDPs) ทฤษฎีไทมด์ออโตมาตาใช้เพื่อการวิเคราะห์และสร้างแบบจำลองพฤติกรรมของระบบที่มีการผูกข้อจำกัดเวลาเข้ากับเหตุการณ์ต่างๆ ในขณะที่กระบวนการตัดสินใจของมาร์คอฟช่วยเพิ่มความสามารถในการรองรับความไม่แน่นอนผ่านการใช้ความน่าจะเป็น

ในแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นการเปลี่ยนแปลงสถานะไม่ได้เกิดขึ้นอย่างแน่นอนทุกครั้ง เนื่องจากอาจถูกควบคุมโดยเงื่อนไขเวลาและอิทธิพลของความน่าจะเป็น ซึ่งทำให้ PTA เป็นเครื่องมือที่มีพลวัตและมีประสิทธิภาพสูงในการวิเคราะห์ระบบที่มีความซับซ้อนและความไม่แน่นอนสูง ซึ่งเป็นลักษณะที่พบในหลายๆ สถานการณ์จริง เช่น ระบบควบคุมการจราจรหรือระบบสื่อสาร การพิสูจน์ผลลัพธ์จากออโตมาตาที่จำกัดเวลาเป็นขั้นตอนสำคัญในการสร้างแบบจำลอง PTA เพราะสามารถตั้งค่าความน่าจะเป็นในการเข้าถึงสถานะต่างๆ โดยเฉพาะอย่างยิ่ง การคำนวณความน่าจะเป็นขั้นต่ำและสูงสุดในการเข้าถึงสถานะเหล่านั้นช่วยให้เราสามารถประเมินความเป็นไปได้และระดับความเสี่ยงที่เกี่ยวข้องกับการตัดสินใจต่างๆ

การวิเคราะห์และการตรวจสอบแบบจำลอง PTA จำเป็นต้องอาศัยกระบวนการตัดสินใจของมาร์คอฟเพื่อกำหนดความน่าจะเป็นและผลลัพธ์ที่เกี่ยวข้องกับการกระทำต่างๆ ในแบบจำลอง ด้วยวิธีนี้สามารถทำการวิเคราะห์ที่ลึกซึ้งและอธิบายพฤติกรรมของระบบได้อย่างครอบคลุม ตั้งแต่สถานะเริ่มต้นไปจนถึงสถานะสิ้นสุดที่เป็นไปได้ทั้งหมด นอกจากนี้ยังช่วยให้สามารถคาดการณ์การทำงานของระบบภายใต้เงื่อนไขและความไม่แน่นอนที่หลากหลายโดยตัวอย่างของไทมด์ออโตมาตาความน่าจะเป็น แสดงตามรูปที่ 2.2



รูปที่ 2.2 ตัวอย่างไทมด์ออโตมาตาความน่าจะเป็นที่ถูกควบคุมด้วยหนึ่งนาฬิกา [9]

ไทมด์ออโตมาตาความน่าจะเป็น นำเสนอการจำลองพฤติกรรมที่ขึ้นอยู่กับเวลาและความน่าจะเป็นอย่างมีประสิทธิภาพ โดยใช้กลไกของนาฬิกา ซึ่งเป็นตัวแปรเวลาที่เพิ่มขึ้นตามอัตราคงที่และไม่ลดลง เพื่อจำกัดช่วงเวลาที่การเปลี่ยนแปลงสถานะสามารถเกิดขึ้นได้ นาฬิกาเหล่านี้เป็นหัวใจสำคัญในการแสดงพฤติกรรมเชิงเวลาของระบบ เช่น การควบคุมเวลาในการทำงานของเครื่องจักรหรือการตอบสนองต่อเหตุการณ์ภายนอก

นอกเหนือจากนี้ ความน่าจะเป็นก็มีบทบาทสำคัญในการกำหนดการเปลี่ยนแปลงสถานะใน PTA ช่วยให้ระบบสามารถจัดการกับความไม่แน่นอนและเหตุการณ์สุ่มได้อย่างมีประสิทธิภาพ การตัดสินใจในแต่ละขั้นตอนของระบบไม่ได้ถูกกำหนดโดยกฎที่เข้มงวด แต่เป็นผลมาจากการกระจายความน่าจะเป็นที่ซึ่งสร้างจากการวิเคราะห์การเปลี่ยนแปลงสถานะในอดีตและการคาดการณ์เหตุการณ์ในอนาคต

การคำนวณความน่าจะเป็นบนเส้นเชื่อมแต่ละสายใน PTA ช่วยให้สามารถทำนายเส้นทางที่ระบบอาจจะตามไปตามความน่าจะเป็นที่ได้กำหนดไว้ ผลรวมของความน่าจะเป็นที่อยู่บนจุดแยกต้องเท่ากับ 1 ซึ่งเป็นหลักการพื้นฐานในทฤษฎีความน่าจะเป็นเพื่อให้แน่ใจว่าระบบมีความสมบูรณ์และสามารถทำงานได้ตามที่คาดหวัง การกำหนดผลตอบแทนใน PTA ยังเป็นไปตามหลักการของกระบวนการตัดสินใจของมาร์คอฟ[6] ซึ่งช่วยให้สามารถประเมินและเพิ่มประสิทธิภาพพฤติกรรมของระบบตามผลลัพธ์ที่ต้องการได้ ซึ่งทำให้ PTA เป็นเครื่องมือที่มีคุณค่าในการออกแบบและวิเคราะห์ระบบที่ต้องการความเชื่อถือได้สูงและการตอบสนองที่ปรับเปลี่ยนได้ตามสภาพแวดล้อมที่เปลี่ยนแปลงไป

ในบริบทของไทมด์ออโตมาตาความน่าจะเป็น โครงสร้างของผลตอบแทนมีบทบาทสำคัญในการประเมินประสิทธิภาพและความต้องการของสถานะและการเปลี่ยนแปลงต่างๆ คู่ของผลตอบแทน $r = (r_L, r_{Act})$ กำหนดส่วนประกอบหลักสองอย่าง โดยที่

r_L คือ ฟังก์ชันที่กำหนดอัตราการสะสมของผลตอบแทนให้แต่ละตำแหน่งในออโตมาตาตามเวลาที่ผ่านไป ซึ่งสะท้อนถึงความคิดที่ว่าสถานะบางอย่างภายในระบบอาจสะสมค่าหรือต้นทุนได้เองตามเวลาที่ผ่านไป เพื่อสนับสนุนหรือยับยั้งการอยู่ในสถานะเหล่านั้นนานเกินไป

r_{Act} คือ ฟังก์ชันนี้ระบุผลตอบแทนที่เกี่ยวข้องกับการกระทำทุกอย่างที่ดำเนินการในตำแหน่งที่กำหนด มันจับคู่ผลประโยชน์หรือโทษทันทีจากการดำเนินการใดๆ โดยมีวัตถุประสงค์ในการกำหนดกระบวนการตัดสินใจโดยเน้นผลลัพธ์ของการกระทำ

ทั้งสองฟังก์ชันผลตอบแทนนี้รวมกันเพิ่มความสามารถของไทมด์ออโตมาตาความน่าจะเป็นในการวิเคราะห์คุณสมบัติเชิงปริมาณโดยการนำมิติใหม่เข้าสู่แบบจำลอง การประเมินผลลัพธ์ตามผลตอบแทน *PTA* ขยายความสามารถของไทมด์ออโตมาตาแบบดั้งเดิมโดยการอนุญาตให้พิจารณาไม่เพียงแต่ข้อจำกัดเวลา แต่ยังรวมถึงธรรมชาติที่เป็นความน่าจะเป็นของสภาพแวดล้อมโลกแห่งความจริงที่การตัดสินใจมีผลลัพธ์และเส้นทางที่ดีที่สุดขึ้นอยู่กับทั้งเวลาและผลลัพธ์ทางความน่าจะเป็น ไทมด์ออโตมาตาความน่าจะเป็น เป็นทูเพิลที่ประกอบด้วย 8 ส่วนย่อยที่แต่ละองค์ประกอบคล้ายกับไทมด์ออโตมาตา [10] คือ $PTA = \langle L, \bar{l}, \mathcal{X}, Act, inv, enab, prob, \mathcal{L} \rangle$ โดยที่

L คือ เซตของสถานะที่ออโตมาตาสามารถมีอยู่ได้ แสดงถึงทุกสถานการณ์หรือการกำหนดที่ออโตมาตาสามารถพบเจอ

$\bar{l}$ คือ สถานะเริ่มต้นของออโตมาตา ซึ่งเป็นจุดที่ออโตมาตาเริ่มต้นการทำงาน

$\mathcal{X}$ คือ เซตของนาฬิกาหรือตัวแปรเวลาที่ใช้ในออโตมาตา

Act คือ เซตของการกระทำที่ออโตมาตาสามารถดำเนินการได้

inv คือ เงื่อนไขค่ายืนยง (invariant condition) เป็นเงื่อนไขที่กำหนดขอบเขตเวลาที่สถานะสามารถดำรงอยู่ได้ในสถานะหนึ่ง

$enab$ คือ เงื่อนไขการเปิดใช้ (enabling condition) เป็นเงื่อนไขที่ช่วยให้การเปลี่ยนแปลงระหว่างสถานะเกิดขึ้นได้ กล่าวคือ การเปลี่ยนจากสถานะแรกไปยังสถานะต่อไปสามารถกระทำได้เมื่อผ่านเงื่อนไข

$prob$ คือ ฟังก์ชันที่กำหนดความน่าจะเป็นของการดำเนินการบางอย่าง เพิ่มชั้นของการตัดสินใจแบบสุ่ม

$\mathcal{L}$ คือ เซตของป้ายกำกับหรือการแสดงความเป็นข้อเสนอแนะที่ให้บริบทเพิ่มเติมหรือข้อมูลเกี่ยวกับสถานะและการเปลี่ยนแปลง

2.1.4 ตัวตรวจสอบแบบจำลองความน่าจะเป็นปริซึม (Probabilistic Model Checker : PRISM) [5, 11]

ตัวตรวจสอบแบบจำลองความน่าจะเป็นปริซึม (Probabilistic Model Checker: PRISM) เป็นเครื่องมือที่ใช้ในการสร้างและวิเคราะห์ระบบที่มีพฤติกรรมสุ่มหรือพฤติกรรมที่มีความน่าจะเป็น โดยเครื่องมือนี้ถูกใช้ในการวิเคราะห์ระบบที่มีความหลากหลายของโดเมนการประยุกต์ใช้งาน รวมไปถึงระบบการสื่อสาร โปรโตคอลมัลติมีเดีย อัลกอริทึมการกระจายแบบสุ่ม โปรโตคอลความปลอดภัย ระบบชีวภาพ และอื่นๆ อีกมากมาย

PRISM สามารถสร้างและวิเคราะห์แบบจำลองความน่าจะเป็นได้หลายประเภทดังนี้

1) ห่วงโซ่มาร์คอฟแบบแยกเวลา (discrete-time Markov chains : DTMCs)
2) ห่วงโซ่มาร์คอฟแบบต่อเนื่อง (continuous-time Markov chains : CTMCs)
3) กระบวนการตัดสินใจของมาร์คอฟ (Markov decision processes : MDPs)
4) ออโตมาตาความน่าจะเป็น (probabilistic automata : PAs)
5) ไทมด์ออโตมาตาความน่าจะเป็น (probabilistic timed automata : PTAs)
6) กระบวนการตัดสินใจของมาร์คอฟที่สังเกตได้บางส่วน (partially observable Markov decision processes : POMDPs)
7) ไทมด์ออโตมาตาความน่าจะเป็นที่สังเกตได้บางส่วน (partially observable probabilistic timed automata : POPTAs)

นอกจากนี้ยังรวมถึงการขยายขอบเขตของแบบจำลองเพื่อการวิเคราะห์คุณสมบัติเชิงปริมาณในแง่ของต้นทุนและผลตอบแทนที่เฉพาะเจาะจง ซึ่งหมายถึงว่าสามารถวิเคราะห์พฤติกรรมของคุณสมบัติในรูปแบบต่างๆ ที่ไม่เพียงแค่เกี่ยวกับความน่าจะเป็นเท่านั้น แต่ยังรวมไปถึงการประเมินคุณสมบัติเหล่านั้นในเชิงปริมาณที่สามารถแสดงผลลัพธ์ตามพฤติกรรมของแบบจำลองที่

การอธิบายแบบจำลองโดยใช้ภาษาปริซึมนั้นเป็นวิธีการที่เรียบง่ายและเป็นระบบอิงตามสถานะ ซึ่งช่วยในการสนับสนุนการวิเคราะห์อัตโนมัติของคุณสมบัติเชิงปริมาณที่หลากหลายจากแบบจำลองที่ได้กล่าวไว้ ตัวอย่างของคำถามที่ปริซึมสามารถช่วยวิเคราะห์ได้ ได้แก่ "ความน่าจะเป็นของการที่ระบบจะล่มสลายภายใน 4 ชั่วโมงคือเท่าไร" "ความน่าจะเป็นสูงสุดที่โปรโตคอลจะสิ้นสุดด้วยความผิดพลาดเมื่อพิจารณาจากการตั้งค่าเริ่มต้นที่เป็นไปได้ทั้งหมดคือเท่าไร" "ขนาดขอคิวข้อความที่คาดว่าจะเกิดขึ้นหลังจากผ่านไป 30 นาทีคือเท่าไร" หรือ "เวลาที่คาดหวังในกรณีที่แย่ที่สุดที่อัลกอริทึมจะสิ้นสุดคือเท่าไร" คุณสมบัติเหล่านี้สามารถอธิบายได้โดยใช้ตรรกะชั่วคราว เช่น PCTL, CSL, LTL และ PCTL* รวมถึงการขยายคุณสมบัติเชิงปริมาณของผลตอบแทนในรูปแบบของต้นทุนหรือรางวัล

ตัวตรวจสอบแบบจำลองความน่าจะเป็นปริซึมนั้นรวมเอาโครงสร้างข้อมูลเชิงสัญลักษณ์และอัลกอริทึมที่ใช้ไดอะแกรมการตัดสินใจแบบไบนารี (Binary Decision Diagrams: BDD) และไดอะแกรมการตัดสินใจไบนารีหลายเทอร์มินัล (Multi-Terminal Binary Decision Diagrams: MTBDD) นอกจากนี้ยังมีเอ็นจิ้นการจำลองเหตุการณ์แบบไม่ต่อเนื่อง ซึ่งช่วยสนับสนุนการตรวจสอบแบบจำลองที่ประมาณได้และเชิงสถิติ รวมถึงการนำเทคนิคการวิเคราะห์ต่างๆ เช่น การปรับแต่งนามธรรมเชิงปริมาณและการลดลักษณะที่เหมือนกันออก เพื่อให้ได้ผลลัพธ์ที่มีประสิทธิภาพและแม่นยำยิ่งขึ้น

2.1.4.1 ภาษาปริซึม (The PRISM Language) [11]

ในการสร้างและวิเคราะห์แบบจำลองด้วยปริซึมจะต้องระบุด้วยภาษาปริซึมซึ่งเป็นภาษาที่เรียบง่ายและเป็นภาษาที่อ้างอิงตามสถานะ โดยยึดตามปฏิกิริยาแบบโมดูล [6] ของ Alur และ Henzinger ใช้สำหรับแบบจำลองทุกประเภทที่ปริซึมรองรับ

องค์ประกอบพื้นฐานของภาษาปริซึม คือ โมดูล และตัวแปร แบบจำลองประกอบด้วยโมดูลจำนวนหนึ่งที่สามารถโต้ตอบกันได้ โมดูลประกอบด้วยตัวแปรท้องถิ่นจำนวนหนึ่ง ค่าของตัวแปรเหล่านี้ในเวลาใดก็ตามถือเป็นสถานะของโมดูล สถานะโกลบอลของแบบจำลองทั้งหมดถูกกำหนดโดยสถานะท้องถิ่น (Local state) ของโมดูลทั้งหมด ลักษณะการทำงานของแต่ละโมดูลจะอธิบายโดยชุดคำสั่งตามใช้แบบฟอร์ม

[action] guard -> prob_1 : update_1 + ... + prob_n : update_n;

ในภาษาปริซึม [action] ใช้เพื่อกำหนดชื่อการกระทำ (action) สำหรับการเปลี่ยนแปลงสถานะในแบบจำลอง ในส่วนของคำสั่งตัวป้องกัน (Guard) มีผลกับตัวแปรทั้งหมดในแบบจำลอง (รวมถึงที่เป็นของโมดูลอื่นๆ) การอัปเดตแต่ละครั้งจะอธิบายแทรนซิชันที่โมดูลสามารถทำได้หากตัวป้องกันเป็นจริง แทรนซิชันจะถูกกำหนดโดยการให้ค่าใหม่ของตัวแปรในโมดูล ซึ่งอาจจะเป็นฟังก์ชันของตัวแปรอื่นๆ การอัปเดตแต่ละครั้งจะกำหนดความน่าจะเป็น (หรือในบางกรณีเป็นอัตราของคุณสมบัติ) ซึ่งจะกำหนดให้กับแทรนซิชันที่เกี่ยวข้อง คำสั่งนี้ยังรวมถึงการดำเนินการด้วย ไม่ว่าจะเป็นเพียงเพื่อใส่คำอธิบายประกอบหรือการซิงโครไนซ์

ตัวอย่างต่อไปนี้แสดงแนวคิดพื้นฐานของภาษาปริซึมที่ประกอบไปด้วยสองกระบวนการที่เหมือนกัน ซึ่งต้องดำเนินการภายใต้ข้อยกเว้นร่วมกัน แต่ละกระบวนการอยู่ในสถานะใดสถานะหนึ่งจาก 3 สถานะ {0,1,2} จากสถานะ 0 กระบวนการจะย้ายไปที่สถานะ 1 มีความน่าจะเป็น 0.2 และมีความน่าจะเป็น 0.8 ที่คงอยู่ในสถานะเดิม จากสถานะ 1 เพื่อไปยังส่วนวิกฤตในสถานะที่ 2 สามารถเกิดขึ้นได้ก็ต่อเมื่อกระบวนการอื่นไม่อยู่ในส่วนวิกฤต จากสถานะ 2 กระบวนการจะยังคงอยู่หรือย้ายกลับไปที่สถานะ 0 โดยมีความน่าจะเป็นเท่ากัน รหัสปริซึมเพื่ออธิบายรูปแบบของกระบวนการตัดสินใจของมาร์คอฟของระบบนี้สามารถดูได้ ดังรูปที่ 2.3

```
mdp

module M1
    x : [0..2] init 0;
    [] x=0 -> 0.8:(x'=0) + 0.2:(x'=1);
    [] x=1 & y!=2 -> (x'=2);
    [] x=2 -> 0.5:(x'=2) + 0.5:(x'=0);
endmodule

module M2
    y : [0..2] init 0;
    [] y=0 -> 0.8:(y'=0) + 0.2:(y'=1);
    [] y=1 & x!=2 -> (y'=2);
    [] y=2 -> 0.5:(y'=2) + 0.5:(y'=0);
endmodule
```

รูปที่ 2.3 กระบวนการตัดสินใจของมาร์คอฟสองกระบวนการที่ดำเนินการภายใต้ข้อยกเว้นร่วมกันในภาษาปริซึม[11]

ดังที่กล่าวไว้ข้างต้น ภาษาปริซึมสามารถใช้อธิบายแบบจำลองความน่าจะเป็นได้หลายประเภท เพื่อระบุว่ากำลังอธิบายประเภทใด แบบจำลองปริซึม มักจะมีคีย์เวิร์ดประเภทแบบจำลองดังนี้

1) **dtmc**: ห่วงโซ่มาร์คอฟแบบแยกเวลา
2) **ctmc**: ห่วงโซ่มาร์คอฟแบบต่อเนื่อง
3) **mdp**: กระบวนการตัดสินใจของมาร์คอฟ
4) **pta**: ไทมด์ออโตมาตาความน่าจะเป็น
5) **pomdp**: กระบวนการตัดสินใจของมาร์คอฟที่สังเกตได้บางส่วน
6) **popta**: ไทมด์ออโตมาตาความน่าจะเป็นที่สังเกตได้บางส่วน

โดยทั่วไปจะระบุไว้อยู่ที่จุดเริ่มต้นของแฟ้ม แต่จริง ๆ แล้วสามารถเกิดขึ้นได้ทุกที่ในแฟ้ม (ยกเว้นในโมดูลและการประกาศอื่น ๆ) หากไม่มีการประกาศประเภทแบบจำลองดังกล่าว โดยค่าเริ่มต้น แบบจำลองจะถือว่าเป็นกระบวนการตัดสินใจของมาร์คอฟ นอกจากนี้ปริซึมยังทำการตรวจหาประเภทแบบจำลองโดยอัตโนมัติ ตัวอย่างเช่น กระบวนการตัดสินใจของมาร์คอฟที่มีตัวแปรนาฬิกาจะถือว่าเป็นไทมด์ออโตมาตาความน่าจะเป็น และกระบวนการตัดสินใจของมาร์คอฟที่มีการสังเกต (observables) ถือว่าเป็น กระบวนการตัดสินใจของมาร์คอฟที่สังเกตได้บางส่วน

ในการกำหนดโมดูลและตัวแปรโดยใช้ภาษาปริซึมนั้นมีลักษณะคล้ายภาษาการเขียนโปรแกรมทั่วไป โดยโมดูลเป็นตัวแทนของแต่ละกระบวนการ สามารถระบุดังนี้

module name ... endmodule

คำจำกัดความของโมดูลประกอบไปด้วยสองส่วน คือ ตัวแปร (variables) และคำสั่ง (commands) ตัวแปรใช้อธิบายสถานะที่เป็นไปได้ที่โมดูลสามารถอยู่ได้ คำสั่งอธิบายพฤติกรรมของโมดูล ตัวอย่างเช่น วิธีการที่สถานะเปลี่ยนแปลงไปตามเวลา ปัจจุบันปริซึมรองรับตัวแปรอย่างง่ายเพียงไม่กี่ประเภท สามารถเป็นช่วงจำกัด จำนวนเต็มหรือบูลีน

ตัวอย่างเช่น โมดูลมีตัวแปรจำนวนเต็มหนึ่งตัว ที่มีช่วง [0..2] การประกาศตัวแปรจะใช้

x : [0..2] init 0;

โปรดสังเกตว่ามีการระบุค่าเริ่มต้นของตัวแปรด้วย และตัวแปรบูลีนถูกประกาศดังนี้:

b : bool init false;

นอกจากนี้สามารถละเว้นค่าเริ่มต้นได้ โดยกรณีนี้จะถือว่ามีค่าต่ำสุดในช่วง หรือเป็นเท็จกรณีตัวแปรสำหรับบูลีน ดังนั้นการประกาศตัวแปรที่แสดงด้านล่างจึงเทียบเท่ากับการประกาศข้างต้น ยังสามารถระบุสถานะเริ่มต้นหลายรายการสำหรับแบบจำลองได้

x : [0..2]; b : bool;

สำหรับการวิเคราะห์แบบจำลองบางประเภท เช่น การตรวจสอบแบบจำลองประมาณ ก็สามารถใช้ตัวแปรจำนวนเต็มที่มีช่วงที่ไม่จำกัด เช่น

x : int; y : int init 3;

ลักษณะการทำงานของแต่ละโมดูลจะอธิบายโดยคำสั่ง ซึ่งประกอบไปด้วยตัวป้องกันและการอัปเดตค่าอย่างน้อย 1 รายการ ตัวอย่างเช่น [] x=0 -> 0.8:(x'=0) + 0.2:(x'=1); อธิบายได้ว่าเมื่อตัวแปร x มีค่าเป็น 0 จะเกิดการอัปเดต (x’=0) ด้วยความน่าจะเป็น 0.8 และ (x’=1) ด้วยความน่าจะเป็น 0.2 ซึ่งความน่าจะเป็นต้องรวมกันเท่ากับหนึ่งเพื่อไม่ให้เกิดข้อผิดพลาดในการดำเนินการผ่านปริซึมโดยในกรณีที่ไม่สามารถระบุชื่อการกระทำได้ สามารถระบุ [] เป็นการเริ่มต้นคำสั่ง

ตัวอย่างที่สอง [] x=1 & y!=2 -> (x'=2); แสดงให้เห็นว่าตัวป้องกันสามารถมีข้อจำกัดในตัวแปรใดๆ ไม่ใช่แค่ตัวแปรในโมดูลนั้น เช่น พฤติกรรมของโมดูลหนึ่งสามารถขึ้นอยู่กับสถานะของอีกโมดูลหนึ่งได้ อย่างไรก็ตาม การอัปเดตสามารถระบุได้เฉพาะค่าตัวแปรที่เป็นของโมดูลเท่านั้น โดยทั่วไปแล้วโมดูลสามารถอ่านตัวแปรของโมดูลอื่น ๆ ได้ แต่เขียนเฉพาะในตัวเองเท่านั้น เมื่อคำสั่งประกอบด้วยการอัปเดตครั้งเดียวที่มีความน่าจะเป็น 1 สามารถละเว้น 1.0: ได้ดังที่ทำในตัวอย่างด้านบน

หากโมดูลมีมากกว่าหนึ่งตัวแปร การอัปเดตจะอธิบายค่าใหม่สำหรับแต่ละตัวแปร ตัวอย่างเช่น หากมีตัวแปร x1 และ x2 สองตัว คำสั่งที่เป็นไปได้คือ

[] x1=0& x2>0& x2<10 -> 0.5:(x1'=1)&(x2'=x2+1)

+0.5:(x1'=2)&(x2'=x2-1);

สังเกตว่าองค์ประกอบของการอัปเดตถูกเชื่อมด้วย & และแต่ละองค์ประกอบจะถูกวงเล็บแยกกัน หากการอัปเดตไม่ได้ให้ค่าใหม่สำหรับตัวแปรภายใน จะถือว่าไม่มีการเปลี่ยนแปลง ในกรณีนี้สามารถใช้คีย์เวิร์ด true สามารถใช้เพื่อแสดงการอัพเดทที่ไม่มีการเปลี่ยนแปลงค่าของตัวแปร กล่าวคือ ค่าดังรูปที่ 2.4 นี้เทียบเท่ากันทั้งหมด

```
[] x1>10 | x2>10 -> (x1'=x1)&(x2'=x2);
[] x1>10 | x2>10 -> (x1'=x1);
[] x1>10 | x2>10 -> true;
```

รูปที่ 2.4 กระบวนการที่ไม่มีการเปลี่ยนแปลงค่าตัวแปรในภาษาปริซึม[11]

สิ่งสำคัญคือต้องจำไว้ว่านิพจน์ทางด้านขวามือของการอัปเดตแต่ละครั้งอ้างอิงถึงสถานะของแบบจำลองก่อนการอัปเดตจะเกิดขึ้น ตัวอย่างเช่น คำสั่งต่อไปนี้การอัปเดตค่าตัวแปร x2 จะเท่ากับ 0 ไม่ใช่ 2

[] x1=0 & x2=1 -> (x1'=2)&(x2'=x1)

PRISM สนับสนุนการใช้ค่ายืนยง อาจเป็นจำนวนเต็ม จำนวนจุดลอยตัว หรือบูลีน สามารถกำหนดได้โดยใช้ค่าตามตัวอักษรหรือเป็นนิพจน์คงที่ โดยใช้คีย์เวิร์ด const ตัวอย่างในรูปที่ 2.5

```
const int radius = 12;
const double pi = 3.141592;
const double area = pi * radius * radius;
const bool yes = true;
```

รูปที่ 2.5 การกำหนดค่ายืนยงในภาษาปริซึม[11]

โดยค่ายืนยงอยู่ภายใต้กฎเดียวกันกับตัวแปร ค่ายืนยงสามารถใช้ได้ทุกที่ที่ต้องการดังตัวอย่างในรูปที่ 2.6

```
ctmc

const int N = 10;
const double mu = 1/10;
const double lambda = 1/2;
const double gamma = 1/3;

module queue
     q : [0..N];
     [] q<N -> mu:(q'=q+1);
     [] q=N -> mu:(q'=q);
     [serve] q>0 -> lambda:(q'=q-1);
endmodule

module server
     s : [0..1];
     [serve] s=0 -> 1:(s'=1);
     [] s=1 -> gamma:(s'=0);
endmodule
```

รูปที่ 2.6 CTMC แบบจำลองคิวงาน N-place และเซิร์ฟเวอร์ที่ลบงานออกจากคิว [11]

ตัวอย่างรูปที่ 2.6 CTMC ที่สร้างแบบจำลองคิวงาน N-place และเซิร์ฟเวอร์ที่ลบงานออกจากคิวและประมวลผล ค่ายืนยงสามารถใช้ได้ทุกที่ที่ต้องการค่ายืนยง เช่น ขอบเขตของตัวแปร ความน่าจะเป็น หรือที่ใดก็ได้ในตัวป้องกันและปรับปรุง เป็นต้น

PRISM **รองรับนิพจน์ที่มีค่าตามตัวอักษร** (12, 3.141592, true, false เป็นต้น) ตัวระบุค่า (ตัวแปร ค่ายืนยง ล) โดยดำเนินการผ่านตัวดำเนินการ ต่อไปนี้

1) - (ค่าติดลบ)
2) *, / (คูณ , หาร)
3) +, - (บวก, ลบ)
4) <, <=, >=, > (ตัวดำเนินการเชิงสัมพันธ์)
5) =, != (ตัวดำเนินการเท่าเทียมกัน)
6) ! (ปฏิเสธ)
7) & (และ)
8) | (หรือ)
9) <=> (ก็ต่อเมื่อ)
10) => (ความหมาย)
11) ? (การประเมินเงื่อนไข เช่น condition ? a : b หมายถึง "ถ้า condition เป็นจริงแล้วได้ a ถ้าไม่เช่นนั้นได้ b")

สำหรับนิพจน์ส่วนใหญ่จะเทียบเท่ากับ C/C++ หรือ Java ข้อยกเว้น คือตัวดำเนินการหาร / ดำเนินการผ่านทศนิยมเสมอ ไม่ใช่จำนวนเต็ม เช่น ผลลัพธ์ของ 22/7 คือ 3.142857... ไม่ใช่ 3 นิพจน์ทั้งหมดต้องประเมินอย่างถูกต้องในแง่ของประเภท (จำนวนเต็ม จำนวนจุดลอยตัว หรือ บูลีน)

แบบจำลองปริซึมสามารถระบุป้ายกำกับ (Label) เพื่อระบุชุดของตำแหน่งที่สนใจ โดยป้ายกำกับจะใช้ร่วมกับการระบุคุณสมบัติต้องเป็นประเภทบูลีน และใช้เครื่องหมายคำพูด(“ ”) แทนในการระบุชื่อของป้ายตัวอย่างเช่นรูปที่ 2.7

```
label "safe"= temp<=100 | alarm=true;
label "fail" = temp>100 & alarm=false;
```

รูปที่ 2.7 การระบุป้ายกำกับเพื่อตรวจสอบ [11]

PRISM รองรับข้อกำหนดและการวิเคราะห์คุณสมบัติตามต้นทุนและผลตอบแทน ซึ่งหมายความว่าสามารถใช้เพื่อให้เหตุผลได้ ไม่ใช่แค่เกี่ยวกับความน่าจะเป็นที่ตัวแบบจะมีพฤติกรรมในรูปแบบใดรูปแบบหนึ่ง แต่เกี่ยวกับการวัดเชิงปริมาณที่กว้างกว่าที่เกี่ยวข้องกับพฤติกรรมของแบบจำลอง ตัวอย่างเช่น สามารถใช้ปริซึมเพื่อคำนวณคุณสมบัติต่างๆ เช่น "เวลาที่คาดหวัง" "จำนวนการสูญหายของข้อความที่ถูกส่ง" หรือ "การใช้พลังงานที่คาดไว้"

แนวคิดพื้นฐานคือแบบจำลองความน่าจะเป็น (ทุกประเภท) ที่พัฒนาในปริซึมสามารถใช้การกำหนดต้นทุนหรือผลตอบแทน ค่าจริงที่เกี่ยวข้องกับสถานะบางอย่างหรือการเปลี่ยนแปลงของแบบจำลอง เนื่องจากไม่มีความแตกต่างในทางปฏิบัติระหว่างต้นทุนและผลตอบแทนปริซึมจึงสนับสนุนเฉพาะรางวัล (Rewards) เท่านั้น อย่างไรก็ตาม ผู้ใช้มีอิสระในการตีความค่าตามที่ผู้ใช้กำหนดเอง

ในภาษาปริซึมสามารถเพิ่มส่วนของรางวัลจากการเชื่อมโยงกับแบบจำลองโดยใช้ rewards ... endrewards ซึ่งสามารถปรากฏที่ใดก็ได้ในแฟ้มแบบจำลอง ยกเว้นภายในคำจำกัดความของโมดูล โครงสร้างเหล่านี้มีรายการรางวัลอย่างน้อยหนึ่งรายการ พิจารณาตัวอย่างง่ายๆ ต่อไปนี้

rewards true : 1; endrewards

กำหนดให้มอบรางวัล 1 ให้กับทุกสถานะของแบบจำลอง ประกอบด้วยไอเทมรางวัลชิ้นเดียว ด้านซ้ายเป็นตัวป้องกัน (เป็นจริง) และด้านขวาเป็นรางวัล (1) สถานะของแบบจำลองที่ตรงกับภาคแสดงในตัวป้องกันจะได้รับรางวัลที่สอดคล้องกัน โดยทั่วไปแล้ว รางวัลของแต่ละตำแหน่งสามารถระบุได้โดยใช้ระบุรางวัลหลายรายการ แต่ละแถวในรูปแบบ guard : reward โดยที่ตัวป้องกันจะวิวินิจฉัย (เหนือตัวแปรทั้งหมดของแบบจำลอง) และรางวัลคือนิพจน์ (ประกอบด้วยตัวแปร ค่ายืนยง ล จากแบบจำลอง) ตัวอย่างเช่นรูปที่ 2.8

```
rewards
    x=0 : 100;
    x>0 & x<10 : 2*x;
    x=10 : 100;
endrewards
```

รูปที่ 2.8 การกำหนดรางวัลตามเงื่อนไขของตัวแปร [11]

จากรูปที่ 2.8 กำหนดรางวัล 100 ให้กับสถานะที่กำหนด x=0 หรือ x=10 และรางวัล 2*x ให้กับสถานะที่กำหนด x>0 & x<10 รายการรางวัลเดียวสามารถกำหนดรางวัลที่แตกต่างกันให้กับสถานะต่างๆ ขึ้นอยู่กับค่าของตัวแปรแบบจำลองในแต่ละสถานะ ถ้าไม่เข้าเงื่อนไขของตัวป้องกันรายการรางวัลใด ๆ จะไม่มีการมอบรางวัลให้ สำหรับสถานะใดที่เข้าเงื่อนไขหลายตัวป้องกัน รางวัลที่มอบให้ในสถานะนั้นคือผลรวมของรางวัลที่เกี่ยวข้องทั้งหมด

สามารถกำหนดรางวัลให้กับแทรนซิชันในแบบจำลองได้ โดยระบุไว้ในลักษณะเดียวกันกับรางวัลของสถานะ ภายใต้โครงสร้าง **rewards ... endrewards** รายการผลตอบแทนที่อธิบายรางวัลในแต่ละการเปลี่ยนแปลง ใช้การระบุในรูปแบบ [action] guard : reward; ตีความคือการแทรนซิชันจากสถานะที่เข้าเงื่อนไขของตัวป้องกัน guard และติดป้ายการกระทำ action จะได้รับรางวัล reward ตัวอย่างรูปที่ 2.9

```
rewards
    [] true : 1;
    [a] true : x;
    [b] true : 2*x;
endrewards
```

รูปที่ 2.9 การกำหนดรางวัลตามป้ายกำกับของแทรนซิชัน [11]

กำหนดรางวัล 1 ให้กับการเปลี่ยนแปลงทั้งหมดในแบบจำลองที่ไม่มีป้ายกำกับการดำเนินการ และให้รางวัลเป็น x และ 2*x ให้กับการเปลี่ยนแปลงทั้งหมดที่ติดป้ายกำกับด้วยการกระทำ a และ b ตามลำดับ

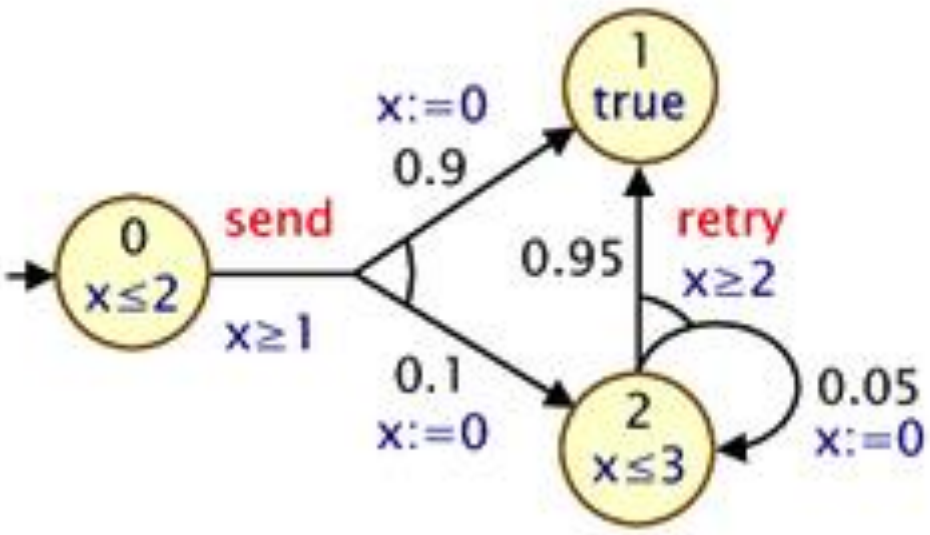
เช่นเดียวกับสถานะ รายการรางวัลหลายรายการสามารถระบุสำหรับแทรนซิชันครั้งเดียว ในกรณีนี้รางวัลที่ได้คือผลรวมของรางวัลแต่ละรายการ แบบจำลองสามารถระบุรางวัลสำหรับทั้งสถานะและการเปลี่ยน ทั้งหมดนี้รวมอยู่ในโครงสร้าง_rewards...endrewards แบบจำลองปริซึม สามารถมีโครงสร้างรางวัลได้หลายแบบ และกำหนดป้ายกำกับได้ดังรูปที่ 2.10

```
rewards "total_time"
    true : 1;
endrewards
rewards "num_failures"
    [fail] true : 1;
endrewards
```

รูปที่ 2.10 การกำกับป้ายรางวัลแบบเฉพาะเจาะจง [11]

PRISM ยังสนับสนุนแบบจำลองเรียลไทม์ โดยเฉพาะอย่างยิ่งไทมด์ออโตมาตาความน่าจะเป็น ซึ่งขยายกระบวนการตัดสินใจของมาร์คอฟ ด้วยความสามารถในการจำลองพฤติกรรมแบบเรียลไทม์ ซึ่งทำในรูปแบบของไทมด์ออโตมาตาโดยการเพิ่มนาฬิกาตัวแปรมูลค่าจริงที่เพิ่มขึ้นตามเวลาและสามารถรีเซ็ตได้

ก่อนอธิบายว่าคุณลักษณะไทมด์ออโตมาตาความน่าจะเป็นถูกรวมเข้ากับรหัสการสร้างแบบจำลองปริซึมอย่างไร ขอยกตัวอย่างไทมด์ออโตมาตาความน่าจะเป็น ตัวอย่างเช่นรูปที่ 2.11



รูปที่ 2.11 แบบจำลองไทมด์ออโตมาตาความน่าจะเป็นขนาดเล็กสำหรับส่งข้อความภายใต้เงื่อนไขเวลา [11]

จากรูปที่ 2.11 สามารถนำมาแปลงเป็นแบบจำลองปริซึมที่สอดคล้องกันได้ดังรูปที่ 2.12

```
pta
module M
    s : [0..2] init 0;
    x : clock;
    invariant
        (s=0 => x<=2) &
        (s=2 => x<=3)
    endinvariant
    [send] s=0 & x>=1 -> 0.9:(s'=1)&(x'=0)
+ 0.1:(s'=2)&(x'=0);
    [retry] s=2 & x>=2 -> 0.95:(s'=1)
+ 0.05:(s'=2)&(x'=0);
endmodule
```

รูปที่ 2.12 แบบจำลองปริซึมไทมด์ออโตมาตาความน่าจะเป็นสำหรับส่งข้อความภายใต้เงื่อนไขเวลา [11]

สำหรับการสร้างแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นในปริซึมมีประเภทข้อมูลใหม่คือ นาฬิกา (clock) สำหรับประกาศใช้ตัวแปรที่เป็นนาฬิกา จะถูกระบุในโมดูลโดยเฉพาะ ไม่สามารถใช้แบบโกลบอลเช่นเดียวกับตัวแปรปริซึมประเภทอื่นๆ สามารถกำหนดได้ตามปกติ ในตัวอย่างข้างต้นใช้เพียงตัวแปร s เป็นจำนวนเต็มตัวเดียว เพื่อแสดงตำแหน่งของไทมด์ออโตมาตาความน่าจะเป็น

ในไทมด์ออโตมาตาความน่าจะเป็นแทรนซิชันอยู่ในตัวป้องกัน ซึ่งจำกัดเวลาที่อาจเกิดขึ้นตามค่าปัจจุบันของนาฬิกา และรีเซ็ต ซึ่งระบุว่าควรตั้งค่าของนาฬิกาเป็นค่าใหม่ (จำนวนเต็ม) โดยถูกระบุในคำสั่งปริซึมด้วยวิธีปกติ: ดูตัวอย่างเช่นการรวม x>=1 ในตัวป้องกันสำหรับคำสั่งป้ายชื่อ send และการอัปเดตของแบบฟอร์ม (x'=0) ซึ่งรีเซ็ตนาฬิกา x เป็น 0

ใน PTA มีส่วนเพิ่มเติมเป็นโครงสร้างคงที่ (invariant) ซึ่งใช้เพื่อระบุนิพจน์ที่อธิบายค่ายืนยงของนาฬิกาสำหรับโมดูลปริซึมแต่ละโมดูล โดยกำหนดข้อจำกัดเกี่ยวกับค่าที่อนุญาตของตัวแปรนาฬิกาต่อค่าของตัวแปรอื่นๆที่ไม่ใช่นาฬิกา โครงสร้างคงที่ควรปรากฏขึ้นระหว่างการประกาศตัวแปรและคำสั่งของโมดูล บ่อยครั้งค่ายืนยงของนาฬิกาถูกอธิบายแยกกันในแต่ละสถานะของ PTA ในตัวอย่าง นาฬิกา x ต้องเป็นไปตาม x<=2 หรือ x<=3 เมื่อตัวแปร s เป็น 0 หรือ 2 ตามลำดับ ถ้า s เป็น 1 จะไม่มีข้อจำกัด (เนื่องจากค่ายืนยงเป็นจริงในกรณีนี้)

2.1.4.2 ข้อกำหนดคุณสมบัติ (Property Specification) [3]

ในการวิเคราะห์แบบจำลองความน่าจะเป็นที่สร้างในปริซึมจำเป็นต้องระบุคุณสมบัติของแบบจำลองอย่างน้อยหนึ่งรายการซึ่งสามารถประเมินได้โดยเครื่องมือ ข้อกำหนดคุณสมบัติของภาษาของปริซึมอยู่ภายใต้ตรรกะชั่วคราวที่เป็นที่รู้จักหลายตัว รวมทั้ง PCTL, CSL, ค่าความน่าจะเป็น LTL

ใช้สำหรับระบุคุณสมบัติของตัวแบบเวลาไม่ต่อเนื่อง เช่น DTMC เป็นต้น หรือแบบจำลองแบบเรียลไทม์ เช่น PTA เป็นต้น โดยแสดงตัวอย่างการเลือกคุณสมบัติโดยใช้ไวยากรณ์ปริซึมและการแปลภาษาธรรมชาติดังนี้

1) P>=1 [F "terminate"] อธิบายว่า ในที่สุดอัลกอริทึมก็สิ้นสุดลงด้วยความน่าจะเป็น 1
2) P<0.1 [F<=100 num_errors > 5] อธิบายว่า ความน่าจะเป็นที่จะเกิดข้อผิดพลาดมากกว่า 5 รายการภายใน 100 หน่วยเวลาแรกจะน้อยกว่า 0.1
3) S<0.01 [num_sensors < min_sensors] อธิบายว่า ในระยะยาว ความน่าจะเป็นที่เซ็นเซอร์ทำงานไม่เพียงพอจะมีค่าน้อยกว่า 0.01

จากตัวอย่างที่กล่าวข้างต้นเป็นการเลือกคุณสมบัติโดยใช้ไวยากรณ์ของปริซึมเพื่อยืนยันคุณสมบัติว่าเป็นไปตามที่คาดหวังหรือไม่ โดยผลลัพธ์จะตอบว่า “ใช่” หรือ “ไม่ใช่” เนื่องจากการอ้างอิงถึงความน่าจะเป็นเกี่ยวของกับอัตราของค่าในเชิงปริมาณ เพื่อใช้ในการตรวจสอบที่เกี่ยวข้องกับขอบเขตบนหรือล่างเพื่อทวนสอบว่าเป็นจริงหรือไม่ ในปริซึมสามารถระบุคุณสมบัติโดยตรงซึ่งผลลัพธ์ที่ประเมินจะเป็นตัวเลข โดยจะแสดงเป็นตัวอย่างดังนี้

1) P=? [!proc2_terminate U proc1_terminate] อธิบายว่า หาความน่าจะเป็นที่กระบวนการ 1 สิ้นสุดลงก่อนกระบวนการ 2
2) Pmax=? [F<=T messages_lost > 10] อธิบายว่า หาความน่าจะเป็นสูงสุดอยู่ที่ข้อความหายไปมากกว่า 10 ข้อความตามเวลา T
3) S=? [queue_size / max_size > 0.75] อธิบายว่า ในระยะยาว หาความน่าจะเป็นที่คิวเต็มมากกว่า 75%

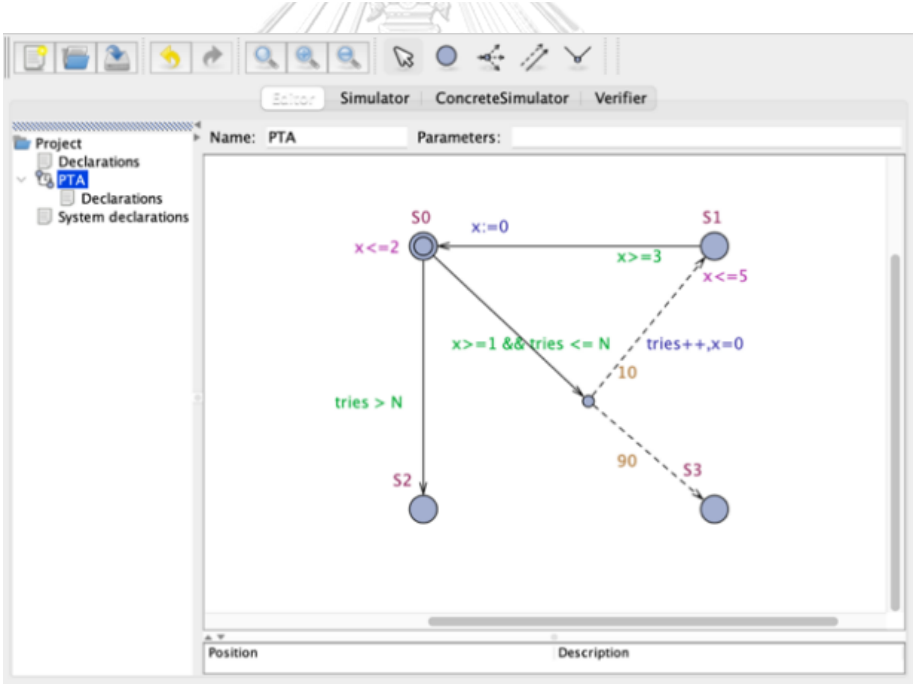
โดยปริซึมทำให้การรวมคุณสมบัติเหล่านี้เข้ากับนิพจน์ที่ซับซ้อนได้ง่าย สามารถคำนวณค่าสำหรับช่วงของตัวแปร และพล็อตกราฟผลลัพธ์จาการทดสอบได้ ซึ่งเป็นประโยชน์มากในการระบุรูปแบบหรือแนวโน้มที่น่าสนใจเกี่ยวกับพฤติกรรมของระบบ

2.1.5 เครื่องมือ UPPAAL [4]

มหาวิทยาลัยอุปซอลา (Uppsala University) ในประเทศสวีเดน และมหาวิทยาลัยอัลบอร์ (Aalborg University) ในประเทศเดนมาร์ก ได้ร่วมมือพัฒนาเครื่องมือทวนสอบ (Verification) ของระบบเวลาจริง โดยชื่อเครื่องมือมาจากอักษรข้างหน้าชื่อมหาวิทยาลัยทั้งสอง 3 ตัวอักษรรวมกันในชื่อ “UPPAAL” เปิดให้ใช้งานเวอร์ชันแรกในปี ค.ศ. 1988 เครื่องมือถูกออกแบบเพื่อทวนสอบความถูกต้องของแบบจำลองที่เป็นเครือข่ายไทมด์ออโตมาตา ปัจจุบันเป็นเวอร์ชัน 4.1.25 ปี ค.ศ. 2019 เครื่องมือ UPPAAL ประกอบไปด้วย 3 ส่วนหลัก

1) ส่วนภาษาคำอธิบาย (Description Language) เป็นคำสั่งของตัวป้องกันที่ไม่ได้ระบุไว้ล่วงหน้า และการอธิบายเพิ่มเติมในส่วนของตัวแปรพร้อมประเภทของข้อมูล เช่น จำนวนเต็มที่มีขอบเขต อาร์เรย์ เป็นต้น เพื่อทำหน้าในการอธิบายพฤติกรรมของระบบในเครือข่ายออโตมาตา
2) ส่วนจำลอง (Simulation) เป็นเครื่องมือตรวจสอบความถูกต้องซึ่งช่วยให้สามารถตรวจสอบการดำเนินการแบบพลวัต ที่เป็นไปได้ของระบบในส่วนของขั้นตอนการออกแบบ หรือการสร้างแบบจำลอง ในช่วงเริ่มต้น
3) ส่วนการทวนสอบแบบจำลอง (Model Checker) เป็นการทวนสอบค่ายืนยง วิเคราะห์ความสามารถในการเข้าถึงแบบจำลองของปริภูมิสถานะ (state space) ของระบบ รวมถึงตรวจสอบภาระติดตาย

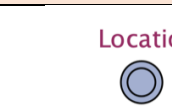
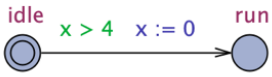
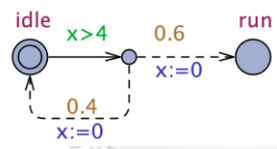
โดยองค์ประกอบของส่วนต่อประสานกราฟิกกับผู้ใช้งานเมื่อเปิดใช้งานจะเข้าสู่หน้าตัวแก้ไขเป็นหน้าแรกโดยจะให้ผู้ใช้งานสร้างแบบจำลองร่วมกับส่วนภาษาคำอธิบาย ดังรูปที่ 2.13 ด้านล่าง



รูปที่ 2.13 ส่วนต่อประสานกราฟิกกับผู้ใช้ของเครื่องมือ UPPAAL [4]

ในส่วนของไทมด์ออโตมาตาความน่าจะเป็นในเครื่องมือ UPPAAL มีลักษณะคล้ายกับไทมด์ออโตมาตาแต่จะเพิ่มในส่วนของการระบุความน่าจะเป็นผ่านจุดแยกไปยังตำแหน่งอื่นๆ โดยจะแสดงในรูปกราฟมีทิศทาง (Directed Graph) โดยมีองค์ประกอบดังนี้ ดังตารางที่ 2.2

ตารางที่ 2.1 ตัวอย่างองค์ประกอบของแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นในเครื่องมือ UPPAAL [4]

ชื่อ	สัญลักษณ์	คำอธิบาย
1. location		ตำแหน่ง ใช้เพื่อเป็นโหนดในแบบจำลองมีลักษณะเป็นวงกลม โดยกรณีที่ใช้วงกลมซ้อนจะใช้เพื่อระบุว่าเป็นโหนดเริ่มต้น
2. invariant		ค่ายืนยงจะเป็นค่าที่ระบุเงื่อนไขของตำแหน่ง เช่น x > T โดยที่ T เป็นจำนวนเต็ม และ x เป็นนาฬิกาเพื่ออนุมัติการกระทำ
3. enges		เส้นเชื่อมระหว่างสองตำแหน่ง โดยใช้อธิบายตัวป้องกันการอัปเดตค่า
3.1 guard		เงื่อนไขป้องกันจะระบุเงื่อนไขในการดำเนินการใดๆจากโหนดหนึ่งไปยังอีกโหนดหนึ่งโดยกำหนดตัวแปรและนาฬิกาที่เพื่อเปิดใช้งาน
3.2 update		การอัปเดตค่าเมื่อมีการเปลี่ยนสถานะของระบบ หรือเกิดแทรนซิชัน เพื่อปรับปรุงค่าตัวแปรที่ใช้ภายในระบบ
4. branch		จุดแยกเป็นโหนดที่เชื่อมระหว่างตำแหน่งหนึ่งแยกไปยังตำแหน่งอื่นๆ ตั้งแต่สองโหนด จะต้องระบุน้ำหนักหรือค่าของความน่าจะเป็นเพื่อแตกไปยังแต่ละโหนดที่จุดแยกชี้ไป

การอธิบายเพิ่มเติมในส่วนของตัวแปร เป็นการกำหนดตัวแปรที่จะใช้ร่วมกับส่วนจำลองโดยในเครื่องมือ UPPAAL โดยรองรับประเภทของตัวแปรที่แตกต่าง เช่น boolean, integer, double, clocks, scalar, arrays and structures โดยขอตัวอย่างการกำหนดตัวแปรได้ดังต่อไปนี้

1) ค่ายืนยงใช้การประกาศนำหน้าประเภทของตัวแปรเพื่อระบุเป็นค่ายืนยง เช่น const int N=50;
2) ตัวแปรข้อมูล (Data Variable) ใช้ประกาศตัวแปรที่จะใช้ในระบบ เช่น int[1,X] id_t;
3) นาฬิกาเป็นประเภทตัวแปลที่ใช้กำหนดเวลาในการทวนสอบ เช่น clock x;

การทวนสอบในเครื่องมือ UPPAAL จะใช้ภาษาในการสืบค้นเพื่อใช้ในการทวนสอบคุณสมบัติของแบบจำลองซึ่งเป็นซับเซตของลอจิกทรีคำนวณตามกำหนดเวลา (Timed Computation Tree Logic : TCTL) โดยมีการนิยามสัญลักษณ์ดังนี้

1) มีเส้นทาง (exists a path : E) ในเครื่องมือ UPPAAL จะใช้ “E”

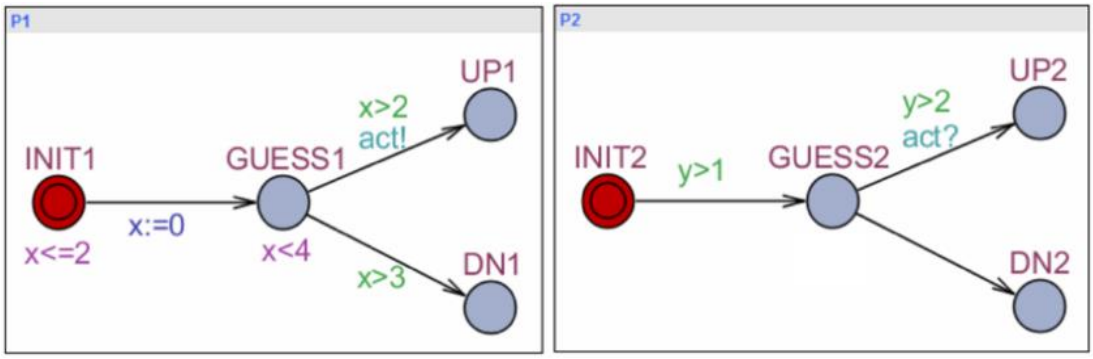
2) สำหรับทุกเส้นทาง (all paths : A) ในเครื่องมือ UPPAAL จะใช้ “A”
3) สำหรับทุกสถานะในเส้นทาง (all states in a path : G) ในเครื่องมือ UPPAAL จะใช้ “[]”
4) บางสถานะในเส้นทาง (some state in path : F) ในเครื่องมือ UPPAAL จะใช้ “<>”

2.1 งานวิจัยที่เกี่ยวข้อง

2.2.1 Stochastic Timed Automata Simulator [12] โดย V. Kaczmarczyk, M. Sír, Z. Bradác (2010)

งานวิจัยว่าด้วยเรื่องแนวคิดของตัวจำลองไทมด์ออโตมาตาในเครื่องมือจำลอง UPPAAL โดยตัวจำลองจะใช้การคำนวณเวลาที่เกิดการเปลี่ยนแปลงที่สอดคล้องกับตัวป้องกันและค่ายืนยงในแต่ละแทรนซิชันด้วยการดำเนินการแบบสุ่ม จากฟังก์ชันความหนาแน่นของความน่าจะเป็นซึ่งใช้ในการคำนวนจะถูกเพิ่มไปยังเส้นเชื่อมระหว่างตำแหน่ง

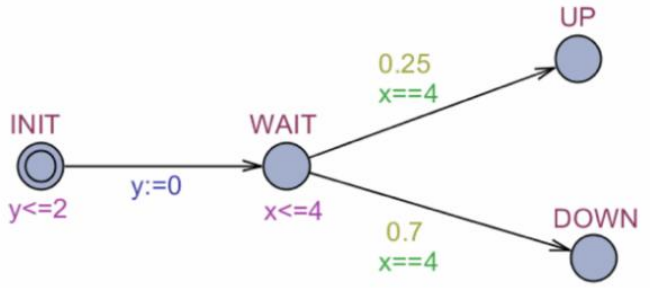
จากคำจำกัดความของระบบแทรนซิชันเป็นทูเพิล (S, s_0, Σ, A) เมื่อ S คือเซตของสถานะ, A คือ เซตข้อความกำกับ , $\Sigma \subseteq S \times A \times S$ คือ เซตของแทรนซิชัน และ s_0 คือ สถานะเริ่มต้น สำหรับ (q,a,q’) $\in \Sigma$ หมายถึง ระบบสามารถเปลี่ยนจาก q ไป q’ บนเหตุการณ์ a โดยเริ่มต้นสถานะที่ s_0 $\in$ S และเมื่อมีการนำเวลาจริงมาร่วมใช้กับระบบแทรนซิชันแล้วไทมด์ออโตมาตา คือทูเพิล (S, s_0, Σ, A, X, G, R, I) เมื่อ X คือ เซตของนาฬิกา, G คือ เซตของข้อจำกัดนาฬิกาบนแทรนซิชัน , I คือ เซตของข้อจำกัดนาฬิกาบนสถานะ และ R คือ เซตของนาฬิกาที่จะรีเซ็ต สำหรับ (s,a,g,r,s’) จะแสดงการเปลี่ยนแปลงจาก s ไปยัง s’ โดยเหตุการณ์ a มี g เป็นข้อจำกัดของนาฬิกาที่กำหนดเมื่อเปิดแทรนซิชัน และ r คือเซตของนาฬิกาที่ถูกรีเซ็ต สำหรับงานวิจัยได้กำหนดฟังก์ชันความหนาแน่นของความน่าจะเป็นในแต่ละแทรนซิชัน โดยเพิ่ม F คือ เซตของฟังก์ชันความหนาแน่นความน่าจะเป็น ดังนั้นไทมด์ออโตมาตาแบบขยายคือทูเพิล (S, s_0, Σ, F, A, X, G, R, I) โดยการเปลี่ยนแปลงแต่ละครั้งสามารถขยายโดยใช้ฟังก์ชันความหนาแน่นของความน่าจะเป็น กรณีศึกษาของงานวิจัยนี้แบ่งเป็นออโตมาตาปราศจากน้ำหนักและออโตมาตาร่วมกับน้ำหนัก โดยออโตมาตาปราศจากน้ำหนักกำหนดสองออโตมาตา P1 และ P2 ดังรูปที่ 2.14



รูปที่ 2.14 ระบบที่มีออโตมาตาสองสถานะ [12]

เมื่อพิจารณแล้วจะพบว่ามีความเป็นไปได้ที่สองแทรนซิชันสามารถเกิดขึ้นพร้อมกันหลังสถานะ GUESS ในครั้งเดียว ถึงแม้ความน่าจะเป็นไปได้ที่การเปลี่ยนแปลงสองครั้งเกิดขึ้นพร้อมกันเท่ากับศูนย์แต่ก็ควรคำนึงถึง ซึ่งงานวิจัยนี้ใช้ฟังก์ชันความหนาแน่นความน่าจะเป็นช่วยในการตัดสินใจเลือกโดยกำหนดความหนาแน่นความน่าจะเป็นในแต่ละแทรนซิชันที่เกิดขึ้น คำนวนค่าเพื่อตัดสินใจเลือกในแต่ละเส้นทาง

ออโตมาตาร่วมกับน้ำหนักสมมุติว่าแทรนซิชันเกิดขึ้นจากสถาะ WAIT ดังรูปที่ 2.15 ซึ่งทั้งสองแทรนซิชันเกิดขึ้นพร้อมกันในเวลาเท่ากับ 4 หน่วย และไม่มีเงื่อนไขพิเศษเพิ่มเติม การตัดสินใจง่ายๆ คือ เลือกเส้นทางใดทางหนึ่งด้วยความน่าจะเป็น 0.5 เพื่อกำหนดความน่าจะเป็นที่เฉพาะในรูปที่ 2.15 จะระบุนำหนักของแต่ละเส้นทางไว้เพื่อโดยจะมีโอกาส 0.05 ที่เกิดการชะงัก กล่าวคือน้ำหนักที่ระบุเป็นส่วนหนึ่งของความหนาแน่นความน่าจะเป็น



รูปที่ 2.15 ไทมด์ออโตมาตาแบบร่วมกับน้ำหนัก [12]

สิ่งที่นำมาใช้ในกับงานวิทยานิพนธ์ : ศึกษาตัวอย่างการสร้างไทมด์ออโตมาตาบนเครื่องมือ UPPAAL รวมถึง กรณีศึกษาที่เกี่ยวข้องกับความน่าจะเป็นที่อาจเกิดขึ้น เพื่อช่วยตัดสินใจเลือกเมื่อสามารถเกิดสองแทรนซิชันในเวลาเดียวกัน และเป็นแนวทางคิดในสร้างแบบจำลองไทมด์ออโตมาตาความน่าจะเป็น ซึ่งมีลักษณะคล้ายกับงานวิจัยนี้

สิ่งที่แตกต่างจากงานวิทยานิพนธ์ : งานวิจัยนี้ใช้เครื่องมือ UPPAAL ในการสร้างแบบจำลองไทมด์ออโตมาตาแบบขยายที่เพิ่มฟังก์ชันความหนาแน่นของความน่าจะเป็นในแต่ละแทรนซิชัน แต่ในวิทยานิพนธ์นี้ผู้วิจัยจะใช้เครื่องมือ UPPAAL ร่วมกับเครื่องมือปริซึมในการตรวจสอบแบบจำลองแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นของตัวแปรที่ส่งผลต่อคุณสมบัติโดยรวมของระบบ

2.2.2 Probabilistic Model Checking and Power-Aware Computing [13] โดย Marta Kwiatkowska, Gethin Norman, David Parker (2005)

งานวิจัยนี้ว่าด้วยเรื่องการเพิ่มประสิทธิภาพของระบบภายใต้ข้อจำกัดการใช้พลังงานและการกระจายพลังงานของแบตเตอรี่ให้เกิดการใช้พลังงานอย่างมีประสิทธิภาพ เนื่องจากการใช้อุปกรณ์

พกพา และอุปกรณ์อิเล็กทรอนิกส์แบบมือถือที่เพิ่มมากขึ้น โดยบทความนี้แสดงให้เห็นถึงการประยุกต์ใช้การตรวจสอบแบบจำลองความน่าจะเป็น เทคนิคการตรวจสอบอย่างเป็นทางการสำหรับการวิเคราะห์ระบบที่แสดง พฤติกรรมสุ่มในด้านการคำนวณการใช้พลังงาน โดยใช้เครื่องมือตรวจสอบแบบจำลองความน่าจะเป็นปริซึมในกรณีศึกษาการปรับขนาดแรงดันไฟฟ้าแบบพลวัต (Dynamic voltage scaling : DVS)

ในรูปแบบเดิมของระบบแบบเรียลไทม์ จะมีชุดของงาน T1,...Tn ที่ต้องดำเนินการเป็นระยะ แต่ละงาน Ti มีระยะเวลาที่เกี่ยวข้อง (Pi) และเวลาดำเนินการกรณีที่เลวร้ายที่สุด (Ci) งาน Ti ถูกปล่อยออกมาทุกหน่วยเวลาของ Pi และจำเป็นต้องดำเนินการให้เสร็จสิ้นก่อนถึงเส้นตายที่กำหนด ซึ่งโดยทั่วไปแล้วจะกำหนดให้เป็นจุดสิ้นสุดของช่วงเวลา กล่าวคือ งานจะต้องเสร็จสิ้นก่อนการปล่อยงานถัดไป ตัวจัดกำหนดการตามเวลาจริงต้องรับประกันว่างานจะตรงตามกำหนดเวลาเนื่องจากทั้งชุดงานสามารถจัดกำหนดการได้และไม่มีงานใดเกินกว่าเวลาที่คำนวณไว้

ตัวกำหนดตารางเวลาการปรับขนาดแรงดันไฟฟ้าแบบพลวัต ในแบบเรียลไทม์ซึ่งอิงตามตัวกำหนดตารางเวลาแบบเรียลไทม์มาตรฐานสองตัว ได้แก่

1) **ตัวกำหนดตารางเวลาเรทโมโนโทนิก (RM)** เป็นแบบคงที่และกำหนดลำดับความสำคัญของงานตามระยะเวลาของงาน โดยจะเลือกงานที่มีระยะเวลาสั้นที่สุดที่พร้อมทำงานเสมอ
2) **ตัวกำหนดตารางเวลาก่อนกำหนดส่งก่อน (EDF)** เป็นงานแบบพลวัตและจัดลำดับตามกำหนดเวลา โดยให้ลำดับความสำคัญสูงสุดกับงานที่เผยแพร่โดยด่วนที่สุด

ตัวกำหนดเวลาแบบแรกใช้เพื่อขยายตัวกำหนดตารางเวลาเรทโมโนโทนิก และตัวกำหนดตารางเวลาก่อนกำหนดส่งก่อน โดยการเลือกความถี่ในการทำงานที่ต่ำที่สุดเท่าที่จะเป็นไปได้ ซึ่งจะทำให้ตัวจัดกำหนดการสามารถบรรลุตามกำหนดกาลของเวลาทั้งหมดของชุดงาน โดยที่ความถี่ถูกกำหนดแบบคงที่ เรียกว่าคงที่ (static)

โดยทั่วไปแล้ว งานจะเสร็จสิ้นเร็วกว่าเส้นตาย ตัวจัดกำหนดการสามารถใช้ข้อเท็จจริงนี้เพื่อลดความถี่ในการดำเนินงาน เมื่อมีการปล่อยงานเพื่อดำเนินการจะไม่ทราบว่าต้องใช้เวลาในการคำนวณเท่าใด ดังนั้นอัลกอริทึมจึงตั้งสมมติฐานแบบอนุรักษ์นิยมว่างานจะต้องใช้เวลาถึงกำหนด อย่างไรก็ตามเมื่องานเสร็จสิ้นสามารถคำนวณจำนวนรอบที่งานไม่ได้ใช้เวลาถึงกำหนดได้ และด้วยเหตุนี้อัลกอริทึมจะคำนวณความถี่ใหม่โดยคำนึงถึงรอบที่ไม่ได้ใช้เวลาเยอะ เรียกว่าวงจรการอนุรักษ์ (cycle conserving : cc)

สำหรับกรณีศึกษานี้ จะพิจารณาโปรเซสเซอร์ที่มีความถี่ในการทำงานสามความถี่คือ 1, 0.75 และ 0.5 โดยมีแรงดันไฟฟ้าที่สอดคล้องกัน 5,4 และ 3 ชุดงานมีขนาดเท่ากับ 3 ดังค่าต่อตารางที่ 2.3

ตารางที่ 2.2 การทำงานของชุดทำงานโปรเซสเซอร์จำนวน 3 งาน [13]

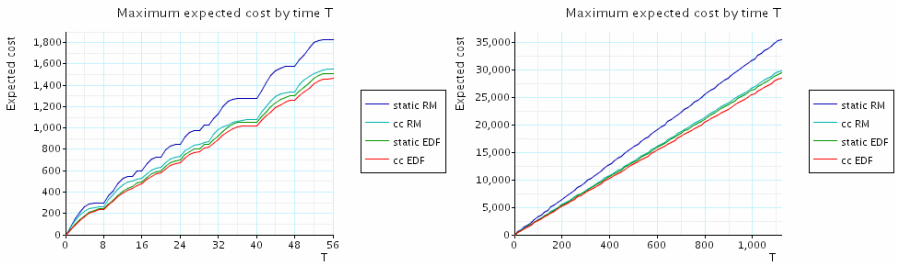
งาน		ระยะเวลาดำเนินงาน (period : Pi)	เวลาดำเนินการกรณีที่แย่ที่สุด (WCET : Ci)
T1		8	3
T2		10	3
T3		14	1

ในการสร้างแบบจำลองระบบในปริซึมจะใช้การแยกเวลาเพื่อลดความซับซ้อนของกระบวนการสร้างแบบจำลองโดยแบ่งหน่วยเวลาหนึ่งๆ นั้น สำหรับแต่ละความถี่ที่เป็นไปได้ จำนวนของขั้นตอนเวลาที่ไม่ต่อเนื่องที่งานต้องทำให้เสร็จใน WCET เป็นจำนวนเต็ม ตัวอย่างเช่น หากงานถูกรันที่ความถี่ 0.75 และงานนั้นใช้หน่วยเวลา WCET จะต้องใช้หน่วยเวลา WCET*4/3 เพื่อให้เสร็จสมบูรณ์ และการกระจายเวลาการเสร็จสิ้นของงานมีการกระจายอย่างสม่ำเสมอระหว่าง 1 ถึง WCET สำหรับรายละเอียดเพิ่มเติมสามารถดูได้ที่ [13]

แบบจำลองที่สร้างขึ้นคือกระบวนการตัดสินใจของมาร์คอฟเพื่อตรวจจับทั้งสองแบบของพฤติกรรมความน่าจะเป็น (เวลาดำเนินการของแต่ละงานเป็นแบบสุ่มเพราะทราบเพียงตัวเลขกรณีที่เลวร้ายที่สุดเท่านั้น) และความไม่แน่นอน (มีสถานการณ์ที่แผนงานไม่ได้ระบุงานที่จะกำหนดเวลา) เนื่องจากแบบจำลองเป็นกระบวนการตัดสินใจของมาร์คอฟจึงใช้ตรวจสอบพฤติกรรมกรณีที่เลวร้ายที่สุดของการใช้งานแต่ละอัลกอริทึม

ผลลัพธ์ที่ได้ จากการวิเคราะห์ตัวกำหนดตารางเวลาที่แตกต่างกันโดยเปรียบเทียบ (สูงสุด) ต้นทุนที่คาดหวังตามเวลา T เนื่องจาก T แตกต่างกันไป ฟังก์ชันต้นทุนที่พิจารณานั้นกำหนดโดยกำลังสองของแรงดันกระแสเนื่องจากค่านี้เป็นสัดส่วนกับการใช้พลังงาน

กราฟในรูปที่ 2.16 แสดงการเปรียบเทียบ “ค่าสูงสุดที่คาดหวังพลังงานที่ใช้โดยระยะเวลาที่กำหนด” สำหรับแผนการกำหนดเวลา 4 แบบที่ระบุไว้ข้างต้น ต้นทุนจริงที่วัดได้คือกำลังสองของแรงดันไฟฟ้าของระบบซึ่งเป็นสัดส่วนกับพลังงานที่ใช้



ก.ช่วงเวลาขนาดเล็ก ข.ช่วงเวลาขนาดใหญ่

รูปที่ 2.16 การใช้พลังงานที่คาดหวังสำหรับแผนการตั้งเวลา DVS ที่แตกต่างกันสี่แบบในช่วงเวลาหนึ่ง [13]

สิ่งที่นำมาใช้ในกับงานวิทยานิพนธ์ : ศึกษาตัวอย่างการสร้างแบบจำลอง จากกรณีศึกษาที่นำการปรับขนาดแรงดันไฟฟ้าแบบพลวัตมาสร้างเป็นรหัสปริซึมโดยผู้วิจัยจะประยุกต์ใช้ในการพัฒนากฎการแปลงเป็นรหัสปริซึมและศึกษากระบวนการตรวจสอบแบบจำลองในรูปแบบกระบวนการตัดสินใจของมาร์คอฟ ที่มีลักษณะคล้ายกับไทมด์ออโตมาตาความน่าจะเป็น

สิ่งที่แตกต่างจากงานวิทยานิพนธ์ : งานวิจัยนี้ใช้กระบวนการแปลงแบบจำลองไปสู่รหัสปริซึมในรูปแบบกระบวนการตัดสินใจของมาร์คอฟ สำหรับงานของผู้วิจัยจะใช้กระบวนการแปลงแบบจำลองไปสู่รหัสปริซึมในรูปแบบไทมด์ออโตมาตาความน่าจะเป็น

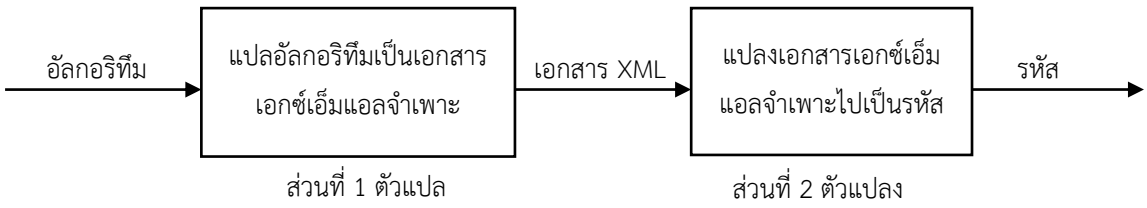
2.2.3 Algorithm To Code Converter [11] โดย Rahul Dubey, B. Edward, Reginald Lewis, Priya Karunakaran (2018)

งานวิจัยนี้ว่าด้วยการแปลงอัลกอริทึมเป็นโค้ด ช่วยให้ผู้เริ่มต้นการฝึกเขียนโค้ดมีสมาธิกับการสร้างตรรกะที่ถูกต้องในการแก้ปัญหา ที่กังวลเกี่ยวกับไวยากรณ์การเข้ารหัส และช่วยโปรแกรมเมอร์ที่มีความบกพร่องทางสายตา ด้วยตัวแปลงรหัส A2C เป็นกระบวนการแปลที่อนุญาตให้ผู้ใช้เขียนเฉพาะอัลกอริทึมของปัญหาด้วยภาษาอังกฤษ ที่ใช้ภาษากึ่งภาษาธรรมชาติซึ่งสามารถแปลงเพิ่มเติมเป็นภาษาคอมพิวเตอร์ Java หรือ C ได้

A2C ประกอบด้วยขั้นตอนกระบวนการแปลงที่มีหลายกระบวนการย่อยที่เกี่ยวข้อง เพื่อให้กระบวนการแปลงเป็นได้อย่างยอดเยี่ยม ระบบจึงไม่ได้เป็นเพียงการแทนที่ไวยากรณ์อย่างง่าย แต่จะจดจำโครงสร้างและตรรกะไว้ด้วย ตัวแปลง A2C จะรับอัลกอริทึมเป็นข้อมูลนำเข้าในแฟ้มข้อความปกติ ถูกอ่านทีละบรรทัดจนจบ แต่ละบรรทัดจะเป็นส่งผ่านไปยังโปรแกรม Naïve Bayes Classifier ซึ่งจะตีความประเภทของข้อความดังกล่าวแปลเป็นคำสั่งเริ่มต้น วนซ้ำหรือเงื่อนไข เมื่อระบุประเภทของแต่ละข้อความแล้ว ส่วนของ POS จะติดป้ายและตัวดำเนินการในประโยคนี้ ซึ่งข้อมูลที่จำเป็นจะถูกดึงออกมาสร้างเป็นเอกสารเอกซ์เอ็มแอลที่มีข้อมูลทั้งหมดที่จำเป็นในการสร้างรหัสให้ถูกต้อง และนำมาใช้ในกระบวนการแปลงรหัส เป็นภาษาโปรแกรมใดก็ได้ โดยการใช้โปรแกรมตัวจับคู่ (mapper) ระบบจะทำนายผลลัพธ์ที่เหมาะสมสำหรับข้อมูลนำเข้าที่ได้กำหนดไว้

ตัวแปลง A2C มีส่วนประกอบที่สำคัญ ได้แก่

1) อัลกอริทึม โมดูลนี้เป็นข้อมูลนำเข้าโดยผู้ใช้ปลายทาง ผ่านเว็บแอปพลิเคชัน โดยอัลกอริทึมที่เป็นที่ยอมรับในระบบจากการนำเข้าเอกสารผ่านการอัพโหลด โดยกระบวนการแปลงอัลกอริทึมเป็นภาษาโปรแกรมใด ๆ จะถูกแบ่งออกเป็นสองส่วนดังแสดงในรูปที่ 2.17



รูปที่ 2.17 กระบวนการแปลงอัลกอริทึมเป็นภาษาโปรแกรม [14]

2) ตัวแปล การแปลอัลกอริทึมข้อมูลนำเข้าเป็นแฟ้มข้อกำหนดเอกซ์เอ็มแอล โดยใช้การจัดหมวดหมู่โดย Naïve Bayes Classifier, POS Tagging และ Data Extraction ตัวอย่าง declare a = 5 ตัวแยกประเภท Naïve Bayes จะจัดประเภทคำสั่งนี้เป็นคำสั่งประกาศ เมื่อทราบแล้วว่าผู้ใช้พยายามประกาศตัวแปรบางตัว แล้วค้นหาข้อมูลทั้งหมดที่จำเป็น เช่น ประเภทข้อมูล ชื่อตัวแปร และค่าเริ่มต้น โดยการคำนวณความน่าจะเป็นว่าคำสั่งนั้นอยู่ในคลาสของคำสั่งแต่ละคลาสมากเพียงใด โดยคลาสที่มีความเป็นไปได้สูงสุดจะถูกเลือกและจัดข้อความให้อยู่ในประเภทนั้น จากนั้นจะดำเนินการโดย POS Tragging และดึงข้อมูลที่เกี่ยวข้องตามประเภทมาสร้างเป็นเอกสารเอกซ์เอ็มแอล

3) แฟ้มเอกซ์เอ็มแอลจำเพาะ เป็นเอกสารที่ใช้ในการแปลงเป็นรหัสโดยได้จากการแปลอัลกอริทึมที่นำเข้า ตัวอย่างเช่น

```
declare integer n
display “Enter a number”
read n from the user
check if n is divisible by 2
display “Number is even”
else display “Number is odd” endif
```

จะได้ผลลัพธ์จากการแปลเป็นเอกสารเอกซ์เอ็มแอลดังรูปที่ 2.18

```
<start>
      <declare>
           <datatype>int</datatype>
           <var>n</var>
           <val>0</val>
      </declare>
      <write>
           <value>Enter a number</value>
      </write>
      <input>
           <var>n</var>
      </input>
      <if>
       <condition> n%2==0</condition>
       <write>
           <value>Number is even</value>
    </write>
      </if>
      <else>
       <write>
           <value>Number is odd</value>
       </write>
      </else>
 </start>
```

รูปที่ 2.18 ผลลัพธ์การแปลข้อความเป็นเอกสารเอกซ์เอ็มแอล [14]

4) ตัวแปลง เกี่ยวข้องกับคำสั่งเงื่อนไขในการแยกวิเคราะห์เอกสารเอกซ์เอ็มแอลจะดำเนินการโดยคำสั่ง if else อย่างง่ายที่ใช้แปลงจากแท็กบนเอกสารเอกซ์เอ็มแอลไปเป็นรหัส แท็กเอกซ์เอ็มแอล ทั้งหมดที่อยู่ในเอกสารเอกซ์เอ็มแอลจะถูกจับคู่เข้ากับรหัสภาษา C ที่สอดคล้องกันโดยตัวแปลง และจะถูกแสดงผลในบนหน้าจอ
5) รหัสที่พึงประสงค์ เป็นผลลัพธ์ที่ได้จากตัวแปลง A2C จากอัลกอริทึมที่นำเข้า โดยตัวอย่างข้อมูลนำเข้าที่ระบุในข้อ 3 เมื่อถูกแปลงแล้วจะแสดงผลลัพธ์ภาษา C ตามดังรูปที่ 2.19

```
#include<stdio.h>
void main() {
      int n=0;
      printf("Enter a number");
      scanf("%d",&n);
      if(n%2==0)
     {
      print("Number is even");
     }
     else
      {
        print("Number is odd");
       }
 }
```

รูปที่ 2.19 ผลลัพธ์การแปลงเอกสารเอกซ์เอ็มแอลเป็นรหัสภาษา C [14]

สิ่งที่นำมาใช้ในกับงานวิทยานิพนธ์ : นำเอาหลักการออกแบบเอกสารเอกซ์เอ็มแอลโดยการออกแบบเอกซ์เอ็มแอล ที่คำนึงถึงข้อมูลนำเข้าก่อนที่จะถูกนำมาแปลเป็นเอกสารเอกซ์เอ็มแอลที่สอดคล้องกับรหัสผลลัพธ์ โดยใช้ตัวแปลงที่วิเคราะห์เอกเอกสารเอกซ์เอ็มแอลในแต่ละแท็กเพื่อจับคู่เข้ากับรหัสปริซึมที่สอดคล้องกัน

สิ่งที่แตกต่างจากงานวิทยานิพนธ์ : ผู้วิจัยเน้นการตีความจากแบบจำลองไทมด์ออโตมาตาความน่าจะเป็น เพื่อนำมาแปลเป็นเอกสารเอกซ์เอ็มแอลที่สื่อถึงแบบจำลองและนำเอกสารเอกซ์เอ็มแอลจำเพาะที่ได้ แปลงเป็นรหัสปริซึมโดยงานวิจัยนี้เน้นตีความจากเอกสารข้อความแปลไปเป็นเอกสารเอกซ์เอ็มแอลและแปลงเป็นภาษา C

2.2.4 Code Generation from UML Model: State of the Art and Practical Implications [15] โดย Andrejs Bajovs, Oksana Nikiforova, Janis Sejans. (2013)

งานวิจัยชิ้นนี้ว่าด้วยเรื่องปัญหาในการสร้างรหัสภายใต้การพัฒนาซอฟต์แวร์ขั้นสูง ที่ไม่สอดคล้องกับสิ่งที่คาดหวังตามหลักเกณฑ์ของการเขียนโปรแกรมเชิงวัตถุ โดยมีเป้าหมายเพื่อจัดระบบข้อมูลเกี่ยวกับวิธีการสร้างรหัส และเทคนิคที่ใช้ในการสร้างรหัสและทดสอบ เพื่อตอบคำถามว่าอะไรคือสาเหตุของการสร้างรหัสที่ทำให้กระบวนการทำงานไม่ถูกต้อง

การพัฒนาซอฟต์แวร์ให้ความสำคัญกับการลดเวลาและต้นทุนในการพัฒนาลง อย่างไรก็ตามยังใช้เวลามากเกินไป จากกระบวนการนำกลับมาใช้ใหม่เพราะเทคโนโลยีเปลี่ยนแปลงเร็วกว่าธุรกิจนำไปสู่สถานการณ์ที่ต้องเขียนรหัสซ้ำแล้วซ้ำเล่าทุกครั้งที่มีการเปลี่ยนแปลงใดๆ ในสถาปัตยกรรมเทคโนโลยี รวมถึงเครื่องมือที่ช่วยในการออกแบบแบบจำลองของสถาปัตยกรรมเหล่านี้เป็นเพียงเอกสารเท่านั้น ซึ่งทำให้เกิดช่องว่างระหว่างความหมายที่แบบจำลองอธิบายและภาษาของโปรแกรมที่ไม่ตรงกับการใช้งานจริง

เป้าหมายของงานวิจัยนี้คือค้นคว้าหลักการสำคัญและแนวทางในการแปลรหัสจากแบบจำลอง โดยอธิบายหลักการพัฒนาซอฟต์แวร์ที่ขับเคลื่อนด้วยแบบจำลอง และมาตรฐานที่เกี่ยวข้องกับการสร้างรหัส โดยได้กรอบการทำงานที่เรียกว่าการพัฒนาซอฟต์แวร์ที่ขับเคลื่อนด้วยแบบจำลอง (Model Driven Software Development : MDSD) เพื่อไม่ให้แบบจำลองเป็นเพียงเอกสารแต่เป็นส่วนหนึ่งของซอฟต์แวร์ โดยในปัจจุบันมักใช้คำว่าสถาปัตยกรรมขับเคลื่อนแบบจำลอง (Model Driven Architecture : MDA) ที่นำมาสู่การใช้ในแต่ละขั้นตอนของวงจรการพัฒนาซอฟต์แวร์ โดยกระบวนการของ MDA เสนอให้สร้างแบบจำลองพื้นฐานสี่แบบสำหรับการพัฒนาระบบ

1) แบบจำลองอิสระด้านการคำนวณ (Computation Independent Model : CIM) แบบจำลองนี้สะท้อนถึงด้านธุรกิจและกำหนดไว้ในระดับปัญหา โดยไม่ระบุถึงรายละเอียดด้านการคำนวณ
2) แบบจำลองอิสระด้านแพลตฟอร์ม (Platform Independent Model : PIM) แบบจำลองนี้แสดงการวิเคราะห์และการออกแบบของระบบซอฟต์แวร์และกำหนดไว้ในระดับโซลูชัน โดยไม่ขึ้นกับแพลตฟอร์มเฉพาะใด
3) แบบจำลองเฉพาะด้านแพลตฟอร์ม (Platform Specific Models : PSM) แบบจำลองเหล่านี้ให้รายละเอียดการออกแบบของระบบซอฟต์แวร์ที่กำลังก่อสร้าง โดยเน้นไปที่รายละเอียดของแพลตฟอร์มเฉพาะ และกำหนดไว้ในระดับซอฟต์แวร์
4) แบบจำลองเฉพาะด้านการดำเนินการ (Implementation Specific Models : ISM) แบบจำลองนี้ ไม่ว่าจะเป็นหนึ่งหรือหลายตัว สะท้อนถึงการดำเนินการและแบบจำลองรันไทม์ของระบบ กำหนดไว้ในระดับการนำไปใช้งาน และใช้ในระหว่างการปรับใช้จริงและการใช้งานซอฟต์แวร์

โดยงานวิจัยนี้เกี่ยวข้องกับการแปลง PSM ไปเป็น ISM หรือเรียกทั่วไปว่าการสร้างรหัส เพื่อแปลงทฤษฎีเป็นการปฏิบัติ ด้วยแนวทางในการสร้างรหัสดังต่อไปนี้

1) แนวทางการสร้างรหัส พบว่าการสร้างรหัสจากแบบจำลอง มีแนวโน้มวนซ้ำผ่านการแสดงข้อความของแบบจำลอง แนวทางที่ใช้คือการใช้เทมเพลตข้อความพิเศษ ที่เป็นชุดของกฎเกณฑ์ โดยระบุวิธีการสร้างรหัสเฉพาะ แต่ยังมีข้อเสียโดยตัวเทมเพลตจะมีความซับซ้อนอย่างรวดเร็ว สำหรับระบบที่ประมวลข้อมูลจำนวนมากเพื่อนำมาใช้งาน
2) ประเภทของตัวสร้างรหัส ได้แบ่งเป็น 5 ประเภทดังนี้
   (1) รหัสมังเกอร์ (Code munger) เป็นวิธีที่ง่ายที่สุด คือเปลี่ยนจากรูปหนึ่งไปเป็นอีกรูปหนึ่งจากการนำเข้าและคุณสมบัติที่สำคัญ (คีย์เวิร์ด แท็ก ล) ใช้เพื่อสร้างแฟ้มที่ส่งออก โดย

ตัวสร้างรหัสประเภทนี้ ใช้เพื่อสร้างเอกสาร เรียกค้นและรวบรวมข้อมูลเฉพาะบางส่วน หรือสร้างผลการวิเคราะห์ข้อมูลนำเข้าบางประเภท

(2) ตัวขยายรหัสแบบอินไลน์ (Inline-code expander) โดยการค้นหามาร์กอัปที่กำหนดไว้ล่วงหน้า เพื่อแทรกรหัสเข้าไป แต่ผลลัพธ์จะเหมือนกับแฟ้มที่นำเข้า ซึ่งอาจทำให้เกิดข้อผิดพลาดเพราะมาร์กอัปทำให้เกิดการรวมรหัสที่ถูกแปลง

(3) ตัวสร้างรหัสแบบผสม (Mixed-code generator) เป็นตัวสร้างรหัสที่ผสมระหว่างสองประเภทที่กล่าวข้างต้น ความแตกต่างคือสามารถใช้แฟ้มนำเข้านำมาสร้างใหม่ โดยแก้ไขแฟ้มนำเข้าใหม่คล้ายๆกับตัวขยายรหัสแบบอินไลน์ แต่แทนที่รหัสด้วยจากตัวสร้างรหัสแบบผสมแทนในส่วนของพื้นที่บริเวณมาร์กอัป

(4) ตัวสร้างบางส่วนของคลาส (Partial-class generator) เป็นการระบุแฟ้มจำกัดความและแฟ้มเทมเพลต โดยวิเคราะห์แฟ้มจำกัดความและเทมเพลตแล้วสร้างเป็นแฟ้มที่ส่งออก ผลลัพธ์จะขึ้นอยู่กับตัวแปรในแฟ้มจำกัดความ การสร้างจะเป็นรูปแบบการสร้างโครงสร้างหรือคลาสพื้นฐาน จากนั้นจะเป็นแบบแมนนวลเพื่อปรับปรุงแฟ้มด้วยฟังก์ชันในขั้นตอนสุดท้าย

(5) ตัวสร้างเทียร์หรือเลเยอร์ (Tier or layer generator) จะคล้ายกับตัวสร้างบางส่วนของคลาสแต่จะดำเนินการสร้างให้แล้วเสร็จสมบูรณ์ในระดับแอปพลิเคชัน หมายความว่าสร้างรหัสที่เพียงพอต่อการทำงานไม่ใช่เพียงคลาสพื้นฐาน โดยสามารถวางตำแหน่งของแม่แบบโดยใช้แบบจำลองยูเอ็มแอลเป็นตัวนำเข้าเพื่อสร้างผลลัพธ์

3) เทคนิคการสร้างรหัส มีเทคนิคในการสร้างรหัสดังนี้

(1) เทคนิคแม่แบบและการกรอง (Templates and filtering) จะใช้เทมเพลตข้อความที่แสดงถึงแต่ละชิ้นส่วนของรหัสต้นฉบับโดยระหว่างการสร้างรหัสตัวแปรของเทมเพลตจะเชื่อมโยงไปยังแบบจำลองของระบบ

(2) เทคนิคแม่แบบและเมตาโมเดล (Templates and meta-model) จะคล้ายกับเทคนิคก่อนหน้านี้แต่จะเพิ่มส่วนของการสร้างเมตาโมเดลก่อนจะถูกนำไปดำเนินการแปลงรหัสเพื่อให้เทมเพลตไม่ขึ้นอยู่กับรูปแบบไวยากรณ์ของภาษาการเขียนโปรแกรมใดๆ

(3) เทคนิคตัวประมวลผลแบบเฟรม (Frame processors) จะใช้การสร้างโดยระบุรหัสด้วยการใช้เฟรมแต่ละเฟรมซึ่งเป็นกลุ่มของคุณสมบัติเรียกว่าสล็อต เมื่อมีเฟรมใหม่สร้างขึ้นแต่ละสล็อตจะเชื่อมกับค่าเฉพาะ โดยแสดงในมุมมองต่างๆ และสามารถอ้างอิงไปเฟรมอื่นๆ ระหว่างสร้างโค๊ด เฟรมจะสร้างโครงสร้างแบบต้นไม้

(4) เทคนิคตัวสร้างตามแบบ API (API-based generator) จะใช้กับการแปลภาษาโปรแกรมเดียว โดยผู้ใช้จะมีเฟรมเวิร์กพิเศษ ที่ช่วยในการสร้างรหัสที่ง่ายขึ้น โดยจะใช้แม่แบบที่ระบุไว้

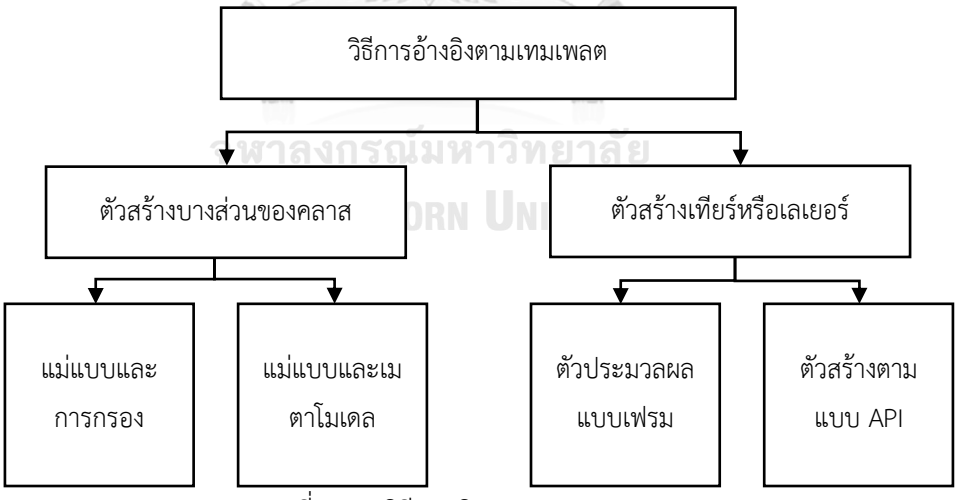
(5) เทคนิคการสร้างแบบอินไลน์ (In-line generation) กำหนดว่ารหัสต้นฉบับของโปรแกรมประกอบด้วยส่วนข้อความ ซึ่งในขณะที่สร้างรหัสจะขึ้นอยู่กับเงื่อนไขที่ใช้เพื่อนำมาขยายความหรือไม่นำมาใช้งานในโปรแกรม

(6) เทคนิครหัสแอตทริบิวต์ (Code attributes) จะกำหนดข้อความพิเศษ เช่นสามารถใส่ความคิดเห็นในโค้ดที่มีอยู่ โดยเป็นความคิดเห็นที่ตัวสร้างรหัสนั้นออกแบบไว้ เทคนิคนี้มักใช้ในการสร้างเอกสารอธิบายรายละเอียดของโปรแกรมโดยอัตโนมัติ

(7) เทคนิคการปะติดปะต่อรหัส (Code weaving) โดยบางส่วนของข้อความที่ไม่เกี่ยวข้องจะถูกแยกเขียนไว้ในเอกสารนำเข้า โดยแต่ละองค์ประกอบของข้อความย่อยเหล่านี้จะถูกรวบรวมเข้าไปในกระบวนการสร้างรหัส แล้วทำให้ข้อความทั้งหมดถูกประมวลผลผสมกันเพื่อสร้างรหัสต้นฉบับของโปรแกรม

4) อนุกรมวิธานตัวสร้างรหัส จากแนวทางการสร้างรหัส ประเภท และเทคนิคที่กล่าวไว้ข้างต้น สามารถนำมาบูรณาการเป็นตัวสร้างรหัส จากวิธีการสร้างรหัสดังนี้

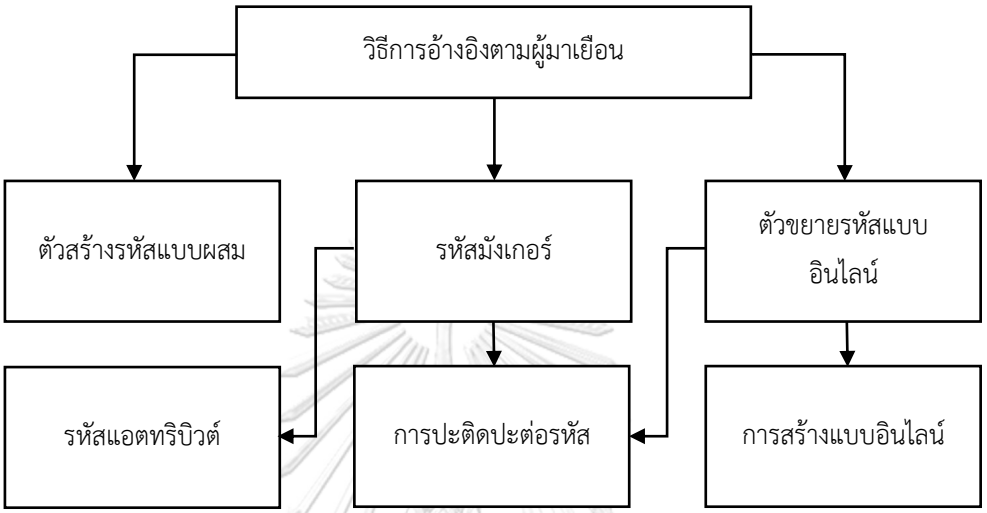
(1) วิธีการอ้างอิงตามแม่แบบ (template-based)



รูปที่ 2.20 วิธีการอิงตามเทมเพลต [15]

จากรูปที่ 2.20 แม่แบบและการกรอง รวมถึงแม่แบบและเมตาโมเดล เกี่ยวข้องกับตัวสร้างบางส่วนของคลาสเพราะ มีการทำงานร่วมกันแม่แบบ โดยเป็นวิธีที่ง่ายที่สุดเพราะอนุญาตให้สร้างเฉพาะบางส่วนของรหัสเท่านั้น สำหรับตัวสร้างเทียร์หรือเลเยอร์มีกลไกที่ซับซ้อนมากจึงใช้ตัวประมวลผลแบบเฟรมและตัวสร้างตามแบบ API

(2) วิธีการอ้างอิงตามผู้มาเยือน (visitor-based)



รูปที่ 2.21 วิธีการอ้างอิงตามผู้มาเยือน [15]

จากรูปที่ 2.21 เทคนิครหัสแอตทริบิวต์เกี่ยวข้องกับตัวสร้างรหัสมังเกอร์ เพราะความสามารถในการเอกสารของโปรแกรมแบบอัตโนมัติ เทคนิคการสร้างแบบอินไลน์ถูกนำมาใช้ในตัวขยายรหัสแบบอินไลน์เพราะไม่ได้เปลี่ยนรูปแบบการป้อนข้อมูลรหัสต้นฉบับ แต่เพิ่มชิ้นส่วนใหม่เข้าไปแทน เทคนิคการปะติดปะต่อรหัส นำมาใช้ได้ทั้งตัวสร้างรหัสมังเกอร์และตัวขยายรหัสแบบอินไลน์ โดยตัวสร้างรหัสข้างต้นสามารถแปลงข้อมูลนำเข้าหรือเพิ่มองค์ประกอบของรหัสใหม่เข้าใหม่ ส่วนตัวสร้างรหัสแบบผสมไม่ได้ระบุเทคนิคที่นำมาเพราะขึ้นอยู่กับวัตถุประสงค์และลักษณะของงาน

สิ่งที่นำมาใช้ในกับงานวิทยานิพนธ์ : ผู้วิจัยจะใช้แนวทางในการสร้างรหัสโดยศึกษาประเภทและเทคนิคการสร้างรหัส มาประยุกต์ใช้บางเทคนิค เพื่อแปลงไทม์ออโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึมด้วยกระบวนการแยกวิเคราะห์องค์ประกอบของเอกสารเอกซ์เอ็มแอล

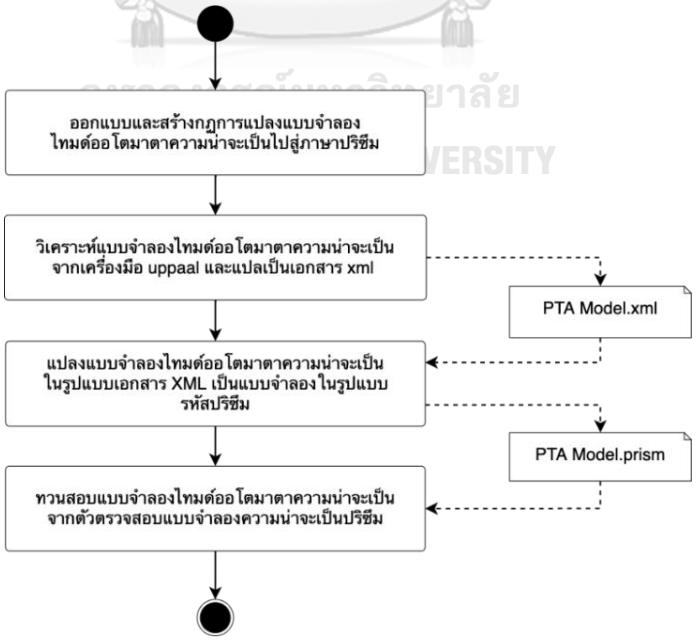
สิ่งที่แตกต่างจากงานวิทยานิพนธ์ : ในวิจัยชิ้นนี้ใช้วิธีการสร้างรหัสอัตโนมัติจากแม่แบบของแบบจำลองลองยูเอ็มแอลเพื่อแปลงเป็นรหัสจากตัวประมวลผลด้วยเทคนิคตัวประมวลผลแบบเฟรมเป็นหลัก แต่ในงานวิทยานิพนธ์นี้จะใช้ตัวสร้างรหัสแบบผสมในการแปลงแบบจำลองไทม์ออโตมาตาความน่าจะเป็นไปสู่รหัสปริซึม

บทที่ 3

การออกแบบขั้นตอนและกฎการแปลงแบบจำลองเป็นรหัสปริซึม

ในบทนี้จะอธิบายการออกแบบขั้นตอนและกฎการแปลงจากไทมด์ออโตมาตาความน่าจะเป็นไปเป็นรหัสสำหรับเครื่องมือตรวจสอบแบบจำลองความน่าจะเป็นในปริซึมนั้นมีความสำคัญเป็นอย่างยิ่ง หัวข้อนี้เกี่ยวข้องกับวิธีการนำเข้าโครงสร้างเอกสารเอ็กซ์เอ็มแอล ที่สร้างจากเครื่องมือ UPPAAL ซึ่งใช้ในการวาดแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นไว้แล้ว และเปลี่ยนเป็นรหัสภาษาปริซึมที่สามารถใช้ในการวิเคราะห์คุณสมบัติของแบบจำลองในเชิงปริมาณได้

กระบวนการแปลงนี้ครอบคลุมหลายขั้นตอนที่ต้องดำเนินการอย่างละเอียดและรอบคอบ เริ่มตั้งแต่การวิเคราะห์โครงสร้างของเอกสารเอ็กซ์เอ็มแอล ที่สร้างจาก UPPAAL ไปจนถึงการสร้างรหัสในภาษาปริซึมโดยทำการแปลงทั้งโครงสร้างและความน่าจะเป็นที่แสดงในไทมด์ออโตมาตาความน่าจะเป็นให้เข้ากับโครงสร้างและภาษาที่ปริซึมสามารถวิเคราะห์ได้ ซึ่งรวมถึงการแปลงกฎการเปลี่ยนแปลงสถานะ การคำนวณความน่าจะเป็น และการรับมือกับความไม่แน่นอนที่เกิดจากการทำงานของเหตุการณ์ต่างๆ ในระบบ การแปลงนี้สามารถแสดงภาพรวมได้ดังรูปที่ 3.1 ซึ่งจะแสดงให้เห็นว่าข้อมูลจากไทมด์ออโตมาตาความน่าจะเป็นในรูปแบบเอ็กซ์เอ็มแอล สามารถถูกนำเข้าสู่ปริซึมและแปลงเป็นแบบจำลองที่ปริซึมสามารถใช้ในการวิเคราะห์ได้ การแปลงนี้จำเป็นต้องมีความแม่นยำสูงเพื่อให้สามารถวิเคราะห์และตรวจสอบคุณสมบัติของระบบได้อย่างเชื่อถือได้และมีประสิทธิภาพ

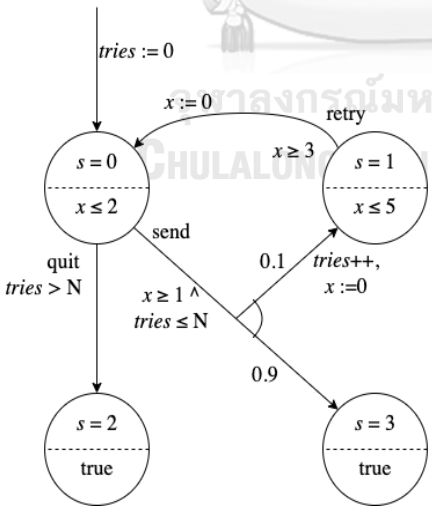


รูปที่ 3.1 ภาพรวมของการแปลงแบบจำลองการแปลงไทมด์ออโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึม

การศึกษานี้มุ่งเน้นที่การสร้างเครื่องมือที่เป็นสะพานเชื่อมระหว่าง UPPAAL และปริซึมซึ่งเป็นสองเครื่องมือที่มีความสำคัญในการวิเคราะห์ระบบที่มีความน่าจะเป็นและความซับซ้อนสูง ที่ช่วยเสริมความเข้า และช่วยในการสร้างเครื่องมือที่สามารถใช้ในการตรวจสอบและการวิเคราะห์ระบบอื่นๆ ที่มีลักษณะคล้ายคลึงกันได้อีกด้วย

3.1 ออกแบบและสร้างกฎการแปลงแบบจำลองไทม์ออโตมาตาความน่าจะเป็นไปสู่ภาษาปริซึม

การออกแบบและสร้างกฎการแปลงสำหรับแบบจำลองไทม์ออโตมาตาความน่าจะเป็นเป็นกระบวนการที่สำคัญซึ่งช่วยให้สามารถทำการวิเคราะห์ความถูกต้องและประสิทธิภาพของระบบที่มีความไม่แน่นอนและข้อจำกัดด้านเวลาได้ ผ่านการใช้ภาษาปริซึมซึ่งเป็นเครื่องมือที่มีประสิทธิภาพในการสร้างแบบจำลองเชิงคณิตศาสตร์ของระบบ และช่วยในการเข้าใจพฤติกรรมของระบบได้อย่างลึกซึ้ง โดยการออกแบบและสร้างกฎการแปลงเหล่านี้ ผู้วิจัยจะต้องเริ่มต้นด้วยการศึกษาและทำความเข้าใจกับแบบจำลองที่มีอยู่ เช่น การศึกษาจากบทความ PRISM 4.0: Verification of Probabilistic Real-time Systems [5] ซึ่งจะช่วยให้สามารถสร้างแบบจำลองที่แม่นยำและสามารถทำนายพฤติกรรมของระบบได้ถูกต้อง การทำความเข้าใจในโครงสร้างและลักษณะของแบบจำลองเป็นสิ่งจำเป็นที่จะนำไปสู่การพัฒนากฎการแปลงที่มีประสิทธิภาพและเที่ยงตรงสำหรับการแปลงไปยังภาษาปริซึมโดยจากศึกษาแบบจำลองจากบทความ PRISM 4.0: Verification of Probabilistic Real-time Systems ดังรูปที่ 3.2



ก.ไทม์ออโตมาตาความน่าจะเป็นในรูปแบบแผนภาพ

```
pta
const int N;
module transmitter
    // Local variables
    s: [0..3] init 0;
    tries: [0..N+1] init 0;
    x: clock;
    invariant
        (s=0=>x<=2) & (s=1=>x<=5)
    endinvariant
    // Guarded commands
    [send] s=0 &x>=1 &tries<=N -> 0.9 : (s'=3)
        + 0.1 : (s'=1)&(tries'=tries+1)&(x'=0);
    [retry] s=1 &x>=3 -> (s'=0)&(x'=0);
    [quit] s=0 &tries>N -> (s'=2);
endmodule
rewards "energy" (s=0) : 2.5; endrewards
```

ข.ไทม์ออโตมาตาความน่าจะเป็นในรูปแบบรหัสปริซึม

รูปที่ 3.2 การแปลงแบบจำลองการพยายามส่งข้อความผ่านช่องทางที่ไม่น่าเชื่อถือโดยมีเงื่อนไขด้านเวลา

จากตัวอย่างแบบจำลองดังรูปที่ 3.2 ที่จำลองการพยายามส่งข้อความผ่านช่องทางที่ไม่น่าเชื่อถือ โดยใช้ไทมด์ออโตมาตาความน่าจะเป็น ได้กำหนดตัวแปร N เป็นจำนวนเต็มบวก เพื่อใช้สำหรับการระบุจำนวนรอบของความพยายามในการส่งข้อความ โดยกำหนดตัวแปร tries เป็นช่วงของความพยายามในการส่งข้อมูลที่เริ่มต้นจาก 0 จนถึงจำนวน N เพื่อส่งข้อความที่มีความน่าจะเป็นอยู่ที่ 90 เปอร์เซ็นต์ที่สามารถส่งข้อความผ่าน และ 10 เปอร์เซ็นต์ที่ไม่สามารถส่งข้อความผ่านได้ภายใต้เงื่อนไขของเวลาที่จำกัด และในกรณีถ้าส่งไม่ผ่านจะเริ่มทำการส่งข้อความไปใหม่อีกครั้งตามจำนวน N รอบที่ได้รับการกำหนดไว้

เพื่อให้การแปลงแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นไปสู่รหัสปริซึมดังเช่นแบบจำลองรูปที่ 3.2 ก ที่อยู่ในรูปแบบจำลอง สามารถแปลงไปเป็นรหัสปริซึมดังเช่น แบบจำลองรูปที่ 3.2 ข จำเป็นต้องทำความเข้าใจโครงสร้างพื้นฐานของภาษาปริซึมโดยเมื่อพิจารณาแล้วโครงสร้างประกอบไปด้วย

1) module : เป็นโครงสร้างพื้นฐานที่ครอบคลุมความพฤติกรรมของส่วนหนึ่งของระบบ โมดูลสามารถสื่อสารและปฏิสัมพันธ์กับโมดูลอื่นๆ ผ่านการเปลี่ยนแปลงสถานะร่วมกันหรือซิงโครไนซ์เหตุการณ์

2) variables : ตัวแปรที่กำหนดในโมดูลสามารถรับค่าในช่วงที่กำหนด และสามารถนำมาใช้กำหนดเงื่อนไขหรือเป็นส่วนหนึ่งของคำสั่งได้ นอกจากนี้ ตัวแปรประเภทนาฬิกา (clock variables) สามารถใช้เพื่อกำหนดข้อจำกัดด้านเวลา

3) commands : แต่ละคำสั่งในโมดูลประกอบด้วยเงื่อนไข (guard) ที่เมื่อเป็นจริงจะทำให้เกิดการเปลี่ยนแปลง คำสั่งสามารถมีหลายเอาต์พุต (สถานะถัดไป) ที่อาจเกิดขึ้นโดยอิงตามความน่าจะเป็นที่กำหนด

4) labels: ป้ายกำกับเป็นเครื่องมือในการระบุกิจกรรมที่สำคัญในโมดูล ซึ่งสามารถใช้เพื่อเชื่อมโยงกับรางวัล (rewards) หรือเพื่อกำหนดเงื่อนไขในการซิงโครไนซ์กิจกรรมระหว่างโมดูล

5) probabilities: ในแบบจำลองความน่าจะเป็น คำสั่งสามารถมีการเปลี่ยนแปลงที่เกี่ยวข้องกับความน่าจะเป็น ซึ่งช่วยให้สามารถสร้างระบบที่มีความไม่แน่นอนและวิเคราะห์พฤติกรรมของระบบในสภาพแวดล้อมที่ซับซ้อนได้

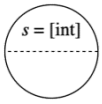
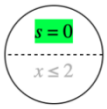
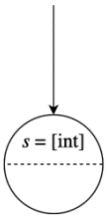
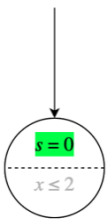
เพื่อเข้าใจอย่างลึกซึ้งว่าแบบจำลองไทมด์ออโตมาตาความน่าจะเป็น สามารถถูกแปลงเป็นรหัสภาษาปริซึมได้อย่างไร ความรู้พื้นฐานเกี่ยวกับโครงสร้างของ PTA จะต้องถูกนำมาวิเคราะห์เพื่อระบุว่าแต่ละองค์ประกอบของมันสอดคล้องกับโครงสร้างในภาษาปริซึมอย่างไร ซึ่งรวมถึงการเข้าใจวิธีการแสดงสถานะต่างๆ ในโมดูลของปริซึมการเปลี่ยนแปลงสถานะและเงื่อนไข (ที่ควบคุมการเปลี่ยนแปลง) และการใช้ความน่าจะเป็นในการกำหนดพฤติกรรมของระบบ การเปลี่ยนแปลงจาก

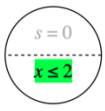
PTA ไปเป็นปริซึมจะต้องแสดงให้เห็นถึงการจัดการเงื่อนไขด้านเวลาและการเปลี่ยนแปลงที่มีความน่าจะเป็นที่แน่นอน ตารางที่ 3.1 ที่จัดเตรียมไว้จะช่วยในการแปลงค่าเหล่านี้เป็นรหัสปริซึมที่สามารถนำไปใช้ในการวิเคราะห์และการทดสอบระบบได้

ตารางที่ 3.1 ตัวอย่างการแปลงแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึม

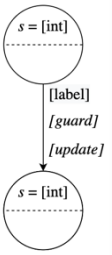
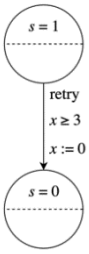
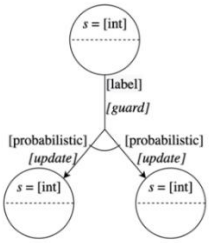
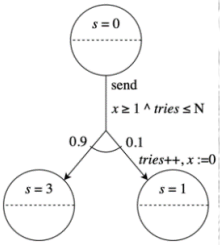
ส่วนประกอบของแบบจำลองของ PTA	ป้ายแสดงในแบบจำลองของ PTA	ตัวอย่างรหัสปริซึม (PRISM code)	คำอธิบาย

ตัวแปรบนแบบจำลอง	*tries, s, x, N*	`cont int N;`
		`s: [0..3] init 0;`
		`tries: [0..N+1] init 0;`
		`x: clock;`
			การกำหนดตัวแปร โดยกำหนดตัวแปร N เป็นจำนวนเต็ม รับค่าเมื่อเริ่มดำเนินการ ตัวแปร s และ tries เป็นจำนวนเต็ม และตัวแปร x เป็นนาฬิกา

		`s: [0..3]`	ตำแหน่ง (Loation) จะกำหนดเป็นตัวแปร s โดยมีขนาดเท่ากับจำนวนตำแหน่งในแบบจำลองทั้งหมดและกำหนดให้ค่าของ s แทนตำแหน่งของสถานะเริ่มต้นที่ 0
		`s:[0..3] init 0;`	จุดเริ่มต้น (Initial) เมื่อไม่ได้ระบุที่มาของลูกศรแล้วลูกศรชี้เข้าที่ตำแหน่งใดๆ จะกำหนดให้ตำแหน่งนั้นเป็นจุดเริ่มต้น โดยจากรูปตัวอย่างชี้เข้าที่ s=0 จึงกำหนด init 0;

		`invariant`
		`(s=0=>x<=2)`
		`& (s=1=>x<=5)`
		`endinvariant`
			ค่ายืนยง (Invariant) เป็นเงื่อนไขของตำแหน่ง การกำหนดค่าจะอยู่ในรูปแบบบูลีน (Boolean) หรือเงื่อนไขเพื่อให้ได้ค่าบูลีน โดยจากรูปตัวอย่าง เมื่ออยู่ในตำแหน่งที่ 0 แล้วเวลาน้อยกว่าหรือเท่ากับ 2 และเมื่ออยู่ในตำแหน่งที่ 1 แล้วเวลาน้อยกว่าหรือเท่ากับ 5

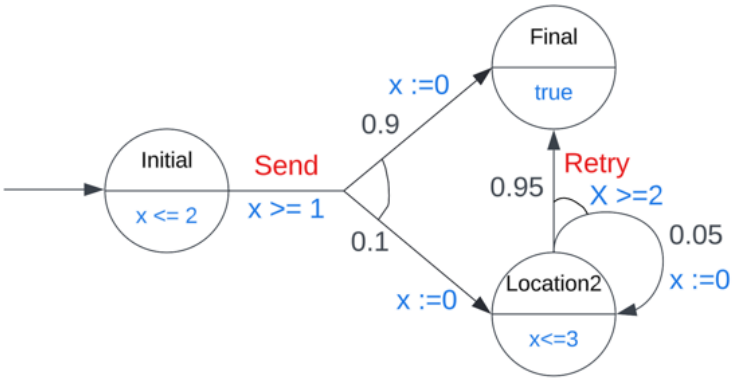
ตารางที่ 3.1 ตัวอย่างการแปลงแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึม(ต่อ)

ส่วนประกอบของแบบจำลองของ PTA	ป้ายแสดงในแบบจำลองของ PTA	ตัวอย่างรหัสปริซึม (PRISM code)	คำอธิบาย
		**[retry] s=1 & x>=3 -> (s'=0)&(x'=0);**	เส้นเชื่อม (Edgs) จะอยู่ระหว่างตำแหน่ง ซึ่งเส้นเชื่อมใช้อธิบาย ตัวป้องกันและการอัปเดตค่า เมื่ออยู่ในรหัสปริซึมจะเรียกว่าคำสั่ง (Command) โดยจากตัวอย่างจะระบุเงื่อนไขว่าถ้าอยู่ในตำแหน่ง 1 และเวลา มากกว่าหรือเท่ากับ 3 จากตัวป้องกันระบุแล้ว จะอัปเดตค่าเวลาเป็น 0 และตำแหน่งเป็น 0
		**[send] s=0 &x>=1 &tries<=N -> 0.9 : (s'=3) + 0.1 : (s'=1) & (tries'=tries+1) & (x'=0);**	เส้นเชื่อมแบบมีค่าความน่าจะเป็นใช้อธิบายความน่าจะเป็นที่จะเกิดการอัปเดตค่าอย่างน้อย 1 รายการ ตัวอย่างคำสั่งคือ เมื่ออยู่ในตำแหน่ง 0 และค่าเวลามากกว่าหรือเท่ากับ 1 และความพยายามมากกว่าค่า N ที่รับเข้ามา แล้วความน่าจะเป็นที่ 90% จะไปที่ตำแหน่ง 3 ความน่าจะเป็นที่ 10% จะอัปเดตความพยายามเพิ่มอีก 1 และอัปเดตค่าเวลาเป็น 0

ในการแปลงแบบจำลองไทมด์ออโตมาตาความน่าไปเป็นรหัสปริซึมเพื่อความชัดเจนและเป็นขั้นเป็นตอนจะขอจำแนกกระบวนการแปลงเป็นขั้นตอนตามลำดับ และอธิบายถึงกฎการแปลงในแต่ละลำดับดังนี้

ขั้นตอนที่ 1 กำหนดตัวแปรของระบบ

ขั้นตอนนี้เริ่มจากการแปลงสถานะและตัวแปรในแบบจำลองไทมด์ออโตมาตาความน่า ที่เกี่ยวข้องกับเวลาและการทดลองเป็นตัวแปรในปริซึมโดยกำหนดช่วงค่าและค่าเริ่มต้นตามที่เหมาะสม ดังตัวอย่างต่อไปนี้



รูปที่ 3.3 แบบจำลองไทมด์ออโตมาตาความน่าไปเป็นของการส่งข้อมูล

การสร้างตัวแปรที่เป็นตัวแทนของสถานะในแบบจำลองไทมด์ออโตมาตาความน่าไปเป็น คือกระบวนการที่สำคัญในการแปลงแบบจำลองเหล่านี้ไปสู่ภาษาปริซึมตัวแปรเหล่านี้จะช่วยกำหนดโครงสร้างและพฤติกรรมของระบบที่จะถูกวิเคราะห์ จากรูปที่ 3.3 และ 3.4 แสดงให้เห็นว่าระบบมีสามสถานะ Initial, Location2 และ Final สำหรับการจำลองในปริซึมตัวแปร s จะถูกใช้เพื่อระบุสถานะเหล่านี้ โดยค่าของ s จะเป็นตัวแทนของตำแหน่งสถานะใน PTA

- s=0 สำหรับสถานะเริ่มต้น (Initial)
- s=1 สำหรับสถานะตำแหน่ง 2 (Location 2)
- s=2 สำหรับสถานะสุดท้าย (Final)

การใช้ตัวแปร s ทำให้แบบจำลองมีความยืดหยุ่นและชัดเจนยิ่งขึ้น สามารถเข้าใจได้ง่ายว่าแต่ละค่าของ s หมายถึงสถานะใดในระบบ นอกจากนี้ยังช่วยในการนำแบบจำลองไปใช้กับเครื่องมือการตรวจสอบแบบจำลองปริซึม เพื่อทำการวิเคราะห์พฤติกรรมของระบบภายใต้สถานการณ์ต่างๆ และใช้สำหับกำหนดรางวัลในระบบ PTAs อีกด้วย

```
s: [0..2] init 0;
// Initial : 0
// Location1 : 1
// Final : 2
```

รูปที่ 3.4 การกำหนดตัวแปรจากตำแหน่งบนแบบจำลอง

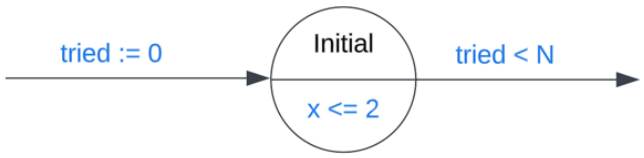
การสร้างตัวแปรในส่วนที่เหลือ เมื่อพิจารณาจากแบบจำลองแล้วจะพบว่ามีการระบุไว้อย่างชัดเจนสำหรับตัวแปรที่ใช้บนแบบจำลองนี้ โดยผู้สร้างแบบจำลองสามารถกำหนดได้ตามประเภทของตัวแปร เช่น นาฬิกา หรือจำนวนเต็ม เป็นต้น

เพื่อใช้ในการกำหนดเงื่อนไขสำหรับการเปลี่ยนแปลงเพื่อให้แบบจำลองในรูปแบบรหัสปริซึมมีความสมบูรณ์ จากรูปที่ 3.3 และ 3.4 ตัวแปร x ถูกกำหนดเพื่อเป็นตัวแทนของนาฬิกาในระบบ นาฬิกาในที่นี้คือตัวแปรที่ใช้เพื่อติดตามการผ่านไปของเวลาภายในระบบ และมักจะใช้ในการกำหนดเงื่อนไขเวลาสำหรับการเปลี่ยนแปลงสถานะ การกำหนดตัวแปร x เป็นนาฬิกาจะเขียนด้วยบรรทัด x : clock; ซึ่งแสดงให้เห็นว่า x จะใช้ในการวัดเวลาที่ผ่านไปและจะถูกตั้งค่าใหม่เมื่อเกิดการเปลี่ยนแปลงสถานะ

```
s: [0..2] init 0;
// Initial : 0
// Location1 : 1
// Final : 2
X : clock;
```

รูปที่ 3.5 การกำหนดตัวแปร x เพื่อใช้เป็นนาฬิกาของระบบ

ในกรณีที่ตัวแปรจำนวนเต็มถูกใช้งานก่อนที่แบบจำลองจะเริ่มทำงาน คือ ณ ตำแหน่งเริ่มต้น ตัวแปรดังกล่าวควรมีการกำหนดค่าเริ่มต้น เพื่อให้แบบจำลองทำงานได้อย่างถูกต้องตั้งแต่เริ่มแรก ตัวอย่างเช่น ถ้ามีตัวแปรที่ใช้ติดตามจำนวนครั้งที่พยายามส่งข้อมูล ตัวแปรนี้อาจถูกกำหนดค่าเริ่มต้นเป็น 0 ในรูปของภาษาปริซึมจากรูปที่ 3.6 นั่นคือ tries: int init 0; ซึ่งหมายความว่าในเวลาเริ่มต้นจำนวนครั้งที่พยายามคือศูนย์



รูปที่ 3.6 ส่วนหนึ่งของแบบจำลองที่แสดงการกำหนดตัวแปรที่ไม่ใช้นาฬิกา

อย่างไรก็ตาม ถ้าตัวแปรจำนวนเต็มถูกนำมาใช้หลังจากแบบจำลองเริ่มทำงานแล้ว ในทางปฏิบัติอาจไม่จำเป็นต้องกำหนดค่าเริ่มต้นทันที ตัวแปรดังกล่าวอาจถูกกำหนดค่าในช่วงต่อไปของการทำงานของแบบจำลอง ตัวอย่างเช่น รูปที่ 3.6 ตัวแปรที่ใช้ตรวจสอบจำนวนเหตุการณ์ที่เกิดขึ้นหลังจากสถานะเริ่มต้นอาจไม่มีการกำหนดค่าเริ่มต้น ซึ่งในปริซึมจะเขียนเป็น N: int; โดยที่ไม่มีการระบุค่าเริ่มต้น

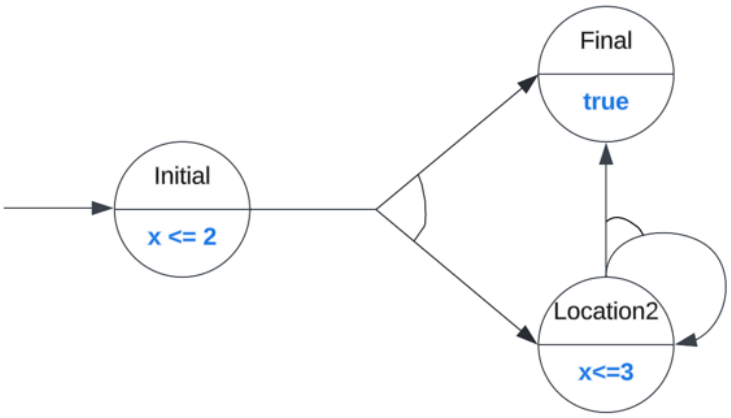
ดังนั้นให้ขั้นตอนที่ 1 การกำหนดตัวแปรสามารถสรุปกฎการแปลงได้ดั้งนี้

1. ตัวแปรสถานะ ใช้ระบุสถานะปัจจุบันของระบบ โดยนับจากจำนวนตำแหน่งบนแบบจำลองเพื่อกำหนดขนาดของตัวแปรนี้ และกำหนดสถานะเริ่มต้นของระบบจากลำดับตำแหน่งเริ่มต้นบนแบบจำลองที่ใช้ในการประกาศตัวแปรนี้
   ตัวอย่างเช่น **`s: [0..3] init 0;`** กำหนดว่ามีสถานะตั้งแต่ 0 ถึง 3 และเริ่มต้นที่สถานะ 0
2. ตัวแปรนาฬิกา ใช้เพื่อติดตามเวลาที่ผ่านไปในระบบ โดยมักสังเกตได้จากการกำหนดตัวแปรนาฬิกาที่จะถูกตั้งค่าใหม่เมื่อเกิดการเปลี่ยนแปลงสถานะ หากไม่ได้ระบุว่าใช้ตัวแปรใดเป็นนาฬิกา
   ตัวอย่างเช่น **`x: clock;`** กำหนดว่าตัวแปร x เป็นนาฬิกา
3. ตัวแปรจำนวนเต็ม ถูกนำมาใช้ในแบบจำลองเพื่อให้แบบจำลองมีความสมบูรณ์ การกำหนดตัวแปรนี้ต้องตรวจสอบในสถานะเริ่มต้นของระบบ ว่ามีการกำหนดค่าเริ่มต้นไว้หรือไม่
   ตัวอย่างเช่น **`tries: int init 0;`** กำหนดว่าตัวแปร tries เป็นจำนวนเต็มที่เริ่มต้นจากศูนย์

ขั้นตอนที่ 2 กำหนดเงื่อนไขค่ายืนยง

ในการแปลงแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึมเป็นกระบวนการที่นักวิจัยจะทำการแยกส่วนและเข้าใจกลไกการทำงานของแต่ละส่วนในแบบจำลองต้นฉบับ จากนั้นจะทำการสร้างโมดูลที่สอดคล้องในปริซึม สำหรับทุกโครงสร้างที่ระบุไว้ในแบบจำลองไทมด์ออโตมาตาความน่าจะเป็น หลักจากที่ได้กำหนดตัวแปรของระบบในขั้นตอนที่แรก ขั้นตอนที่สองคือการนำตัวแปรเหล่านั้นมาสร้างเป็นโมดูล

ในโมดูลต้องพิจารณาเงื่อนไขค่ายืนยง (Invariants) ที่กำหนดไว้สำหรับแต่ละสถานะของแบบจำลอง PTA เงื่อนไขค่ายืนยงเหล่านี้จะถูกนำมาใช้กำหนดสถานะที่สามารถอยู่ได้ในโมดูล และจะถูกกำหนดเป็นเงื่อนไขบูลีนที่ช่วยในการรักษาสถานะนั้นๆ ให้คงที่ การระบุเงื่อนไขค่ายืนยง ที่ชัดเจนในโมดูล เป็นส่วนสำคัญที่ช่วยให้สามารถตรวจสอบและวิเคราะห์พฤติกรรมของระบบในสถานการณ์ที่มีเวลาเป็นปัจจัยสำคัญได้อย่างเที่ยงตรง



รูปที่ 3.7 เงื่อนไขค่ายืนยง (Invariant) บนแบบจำลองไทมด์ออโตมาตาความน่าไปเป็น

จากแบบจำลองในรูปที่ 3.7 สามารถเห็นได้ว่ามีสถานะต่างๆ เช่น Initial (s=0) ที่มีเงื่อนไข x <= 2, หมายถึงนาฬิกา (ตัวแปร x) จะต้องน้อยกว่าหรือเท่ากับ 2 เพื่อที่จะอยู่ในสถานะนี้ สถานะ Location 2 (s=1) มีเงื่อนไข x <= 3 และสถานะ Final (s=2) มีค่าเป็น true ซึ่งหมายความว่าไม่มีเงื่อนไขเฉพาะใดๆ ที่จะต้องถูกปฏิบัติตามเพื่อรักษาสถานะนั้น โมดูลในภาษาปริซึมจะใช้ข้อมูลเงื่อนไขค่ายืนยงเหล่านี้เพื่อกำหนดพฤติกรรมของแต่ละสถานะในแบบจำลองของมันเอง

```
module PTA
  s: [0..2] init 0;
  // Initial : 0
  // Location1 : 1
  // Final : 2
  X : clock;

  invariant
    (s=0=>x <= 2) & (s=1=>x <= 3)
  endinvariant

   // command
endmodule
```

รูปที่ 3.8 เงื่อนไขค่ายืนยง (Invariant) บนแบบจำลองไทมด์ออโตมาตาความน่าไปเป็นในปริซึม

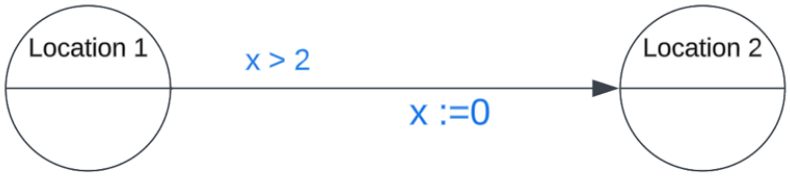
เงื่อนไขค่ายืนยง (Invariant) ในภาษปริซึมมีหน้าที่กำหนดเงื่อนไขที่ต้องเป็นจริงเพื่อให้ระบบสามารถอยู่ในสถานะนั้นๆ ได้ ในรูปที่ 3.8 Invariant ถูกใช้เพื่อกำหนดขอบเขตของตัวแปรนาฬิกา (x) ในแต่ละสถานะของแบบจำลอง โดยเขียนในรูปแบบ (s=0=>x <= 2) & (s=1=>x <= 3) ที่สื่อถึงเงื่อนไขของสถานะ Initial และ Location 2

ดังนั้นให้ขั้นตอนที่ 2 กำหนดเงื่อนไขค่ายืนยงสามารถสรุปกฎการแปลงได้ดั้งนี้

1. การแปลงเงื่อนไขค่ายืนยง (Invariant) ในภาษาปริซึมกำหนดจากเงื่อนไขที่ในแต่ละตำแหน่งบนแบบจำลองที่ได้กำหนดเงื่อนไขค่ายืนยงไว้เพื่อรอการพิสูจน์ว่าเป็นจริง โดยจะใช้ตัวแปรสถานะในการอ้างอิงตำแหน่งของเงื่อนไขค่ายืนยงนี้ควบคู่ไปด้วย
   ตัวอย่างเช่น **`invariant (s=0=>x <= 2) endinvariant`** ระบบจะอยู่ในตำแหน่ง s=0 เมื่อค่านาฬิกาน้อยกว่าหรือเท่ากับ 2. ภายใต้เงื่อนไขค่ายืนยงยังเป็นจริง
2. การแปลงเงื่อนไขค่ายืนยงสำหรับหลายเงื่อนไข หาในระบบมีการกำหนดเงื่อนไข ที่นอกเหนือจากค่าของนาฬิกาสามารถกำหนดควบคู่ไปกับเงื่อนไขอื่นในสถานะนั้นได้
   ตัวอย่างเช่น **`invariant (s=3 => x<20 & y<=5) endinvariant`** สถานะ s=3 อาจมีเงื่อนไขว่าต้องมีนาฬิกา x น้อยกว่า 20 และตัวแปรอื่น y ต้องไม่เกิน 5
3. การแปลงเงื่อนไขค่ายืนยงที่สถานะเชื่อมโยงกัน หากต้องการให้สถานะ s=4 และ s=5 มีเงื่อนไขเวลาที่เชื่อมโยงกัน, โดยสถานะ s=4 จำเป็นต้องมี x น้อยกว่าหรือเท่ากับ 8 และ s=5 ต้องมี x มากกว่าหรือเท่ากับ 9
   ตัวอย่างเช่น **`invariant (s=4 => x<=8) & (s=5 => x>=9) endinvariant`**

ขั้นตอนที่ 3 กำหนดคำสั่ง

คำสั่ง (Commands) ในภาษาปริซึมถูกใช้เพื่ออธิบายพฤติกรรมของแต่ละโมดูลและประกอบด้วยส่วนของตัวป้องกัน (guard) และการอัพเดทหนึ่งครั้งหรือมากกว่า คำสั่งเหล่านี้เป็นส่วนที่กำหนดว่าระบบจะตอบสนองอย่างไรเมื่อเข้าเงื่อนไข ส่วนตัวป้องกันหรือ "guard" ของคำสั่งคือเงื่อนไขที่ต้องเป็นจริงเพื่อให้การอัพเดทที่เกี่ยวข้องสามารถเกิดขึ้นได้ การอัพเดทแสดงถึงการเปลี่ยนแปลงค่าของตัวแปรหนึ่งหรือหลายตัวแปรและอาจรวมถึงการเปลี่ยนแปลงที่มีความน่าจะเป็นเฉพาะเจาะจง และยังอาจแสดงถึงการเลือกการเปลี่ยนแปลงที่มีความน่าจะเป็นหลายค่าเพื่อสะท้อนถึงความไม่แน่นอนในระบบ

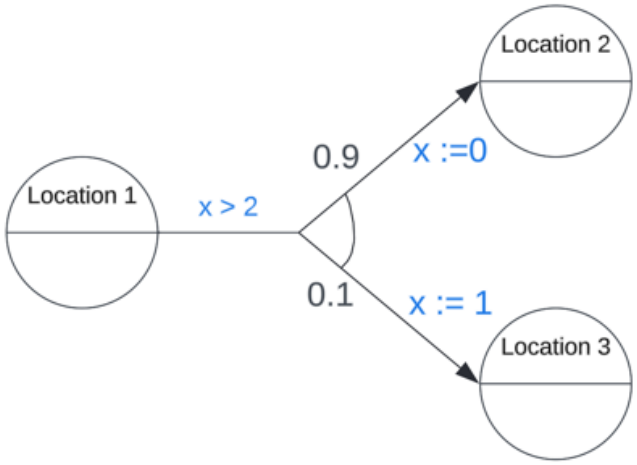


รูปที่ 3.9 การเปลี่ยนสถานะจาก Location 1 ไปยัง Location 2 โดยมีเงื่อนไขของเวลา

ในการเปลี่ยนแปลงสถานะของระบบ คำสั่งจะประกอบไปด้วยเงื่อนไขของตัวแปรสถานะและส่วนของตัวป้องกันเพื่อให้การเปลี่ยนแปลงสามารถในภาษาปริซึมเข้าใจว่าสถานะถูกเปลี่ยนไปจากเดิม ตัวอย่างเช่นรูปที่ 3.9 การเปลี่ยนสถานะจาก Location 1 ไปยัง Location 2 โดยมีเงื่อนไขของนาฬิกา สมมุติ s=0 คือ Location 1 และ s=1 คือ Location 2 โดยมีตัวป้องกันคือค่าของนาฬิกาต้องมากกว่า 2 เมื่อเข้าเงื่อนไข จะเปลี่ยนสถานะจาก Location 1 ไปยัง Location 2 และตั้งค่านาฬิกาใหม่เท่ากัน 0 ดั้งนั้นจะเขียนคำสั่งที่แสดงถึงการเปลี่ยนแปลงสถานะนี้ในภาษาปริซึมได้ คือ

```
[]s=0 & x > 2 -> s'=1 & x'=0;
```

ในการทำความเข้าใจการเปลี่ยนแปลงสถานะของระบบโดยมีเงื่อนไขของเวลาและความน่าจะเป็นเป็นส่วนสำคัญของการวิเคราะห์ระบบไทมด์ออโตมาตาความน่าจะเป็น โดยการใช้ภาษาปริซึมในการแบบจำลอง ซึ่งเป็นเครื่องมือที่สามารถทำการเขียนสคริปต์และการวิเคราะห์โดยอัตโนมัติได้



รูปที่ 3.10 การเปลี่ยนสถานะ โดยมีเงื่อนไขของเวลาและความน่าจะเป็น

จากรูปที่ 3.10 แสดงให้เห็นว่าจากสถานะ Location 1 (s=0) มีสองทางเลือกที่สามารถเกิดขึ้นได้โดยพิจารณาจากความน่าจะเป็นนั้นคือ การเปลี่ยนไปยัง Location 2 (s=1) ด้วยความน่าจะเป็น 90% และการเปลี่ยนไปยัง Location 3 (s=2) ด้วยความน่าจะเป็น 10% ซึ่งสะท้อนถึงความไม่แน่นอนในการตัดสินใจและการเปลี่ยนแปลงในระบบที่สัมพันธ์กับเวลาที่ผ่านไป

การกำหนดเงื่อนไขเวลา x > 2 หมายความว่าการเปลี่ยนสถานะจะไม่เกิดขึ้นจนกว่านาฬิกาในระบบจะมีค่ามากกว่า 2 หน่วยเวลา ซึ่งสามารถใช้เพื่อจำลองสถานการณ์เช่น timeout หรือการรอเงื่อนไขอื่นๆ ที่เกี่ยวข้องกับเวลา ในทางปฏิบัติ ความน่าจะเป็นที่แสดงในแบบจำลองปริซึ่มสามารถใช้เพื่อจำลองการตัดสินใจที่ขึ้นอยู่กับความน่าจะเป็น เช่น การส่งข้อความที่อาจล้มเหลวหรือการเกิด

ข้อผิดพลาดที่สุ่ม เป็นต้น การเขียนคำสั่งในภาษาปริซึมตามรูปที่ 3.10 นั้นเราสามารถทำได้ดังนี้

```
[]s=0 & x > 2 -> 0.9 : s'=1 & x'=0 + 0.1 : s'=1 & x'=1;
```

ในการเปลี่ยนสถานะของระบบ กรณีที่มีความน่าจะเป็นมาเกี่ยวข้องคำสั่งจะแบ่งการอัพเดทค่าใหม่ตามความน่าจะเป็นที่สถานะนั้นจะเปลี่ยนไปยังสถานะอื่น ตัวอย่างเช่นรูปที่ 3.10 การเปลี่ยนสถานะของ Location 1 (s=0) ภายใต้เงื่อนไขของเวลา มีความน่าจะเป็นที่จะอยู่ในสถานะ Location 2 (s=1) ที่ 90 เปอร์เซ็นต์ และอยู่ในสถานะ Location 3 (s=2) ที่ 10 เปอร์เซ็นต์ การเปลี่ยนแปลงสถานะนี้ในภาษาปริซึมจะเพิ่มส่วนของความน่าจะเป็นในส่วนของการอัพเดท สามารถเขียนได้ คือ

```
[]s=0 & x > 2 -> 0.9 : s'=1 & x'=0 + 0.1 : s'=1 & x'=1;
```

ดังนั้นให้ขั้นตอนที่ 3 การกำหนดคำสั่งสามารถสรุปกฎการแปลงได้ดั้งนี้

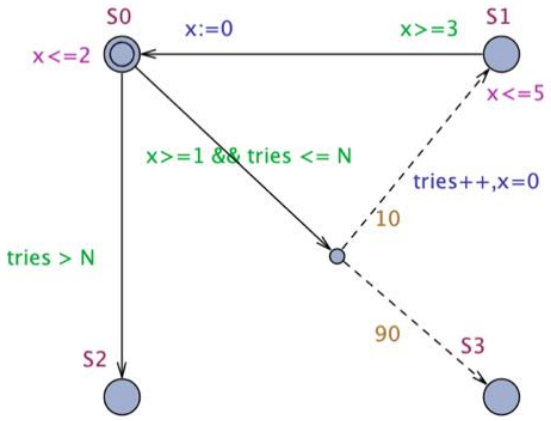
1. การแปลงเงื่อนไขสำหรับแทรนซิชัน (Transitions) สำหรับการเปลี่ยนแปลงจากหนึ่งสถานะไปยังอีกสถานะหนึ่งที่มีเงื่อนไขเวลาหรือเหตุการณ์เฉพาะ ตัวแปรสถานะในปริซึมจะถูกอัพเดทตามเงื่อนไขนั้น ตัวอย่างเช่น ถ้าโมดูลมีเงื่อนไขว่าสามารถเปลี่ยนจากสถานะ s=0 ไป s=1 เมื่อ x > 5 คำสั่งสำหรับการเปลี่ยนแปลงนี้ในภาษาปริซึม จะเขียนได้ดังนี้
   ```
   [] s=0 & x>5 -> (s'=1);
   ```
2. การกำหนดความน่าจะเป็นในการอัพเดทตัวแปร เมื่อต้องการกำหนดความน่าจะเป็นให้กับการเปลี่ยนแปลงตัวแปร คำสั่งในปริซึม จะมีการเพิ่มส่วนของความน่าจะเป็นในการอัพเดท โดยผลรวมของความน่าจะเป็นทุกแทรนซิชันจากจุดแยกต้องเท่ากับ 1
   ตัวอย่างเช่น ถ้าต้องการกำหนดว่าในสถานะ s=0 มีโอกาส 70% ที่จะเปลี่ยนไป s=1 และ 30% ที่จะเปลี่ยนไป s=2, คำสั่งจะเขียนได้ดังนี้: `[] s=0 -> 0.7:(s'=1) + 0.3:(s'=2);`

3.2 วิเคราะห์แบบจำลองไทมด์ออโตมาตาความน่าจะเป็นจากเครื่องมือ UPPAAL และแปลเป็นเอกสารเอ็กซ์เอ็มแอล

การแปลงแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นรหัสปริซึมนั้นเป็นกระบวนการที่มีความซับซ้อนซึ่งต้องการความเข้าใจลึกซึ้งเกี่ยวกับโครงสร้างและพฤติกรรมของระบบที่กำลังถูกจำลอง และการวิเคราะห์นั้นต้องพิจารณาจากหลายมุมมองเพื่อให้สามารถสร้างแบบจำลองที่สามารถทวนสอบและวิเคราะห์พฤติกรรมของระบบเวลาจริงได้อย่างถูกต้องและเที่ยงตรง

ในการสร้างแบบจำลองดังกล่าว ผู้วิจัยมักจะใช้เครื่องมือที่มีความสามารถในการจำลองระบบเวลาจริงอย่าง UPPAAL ซึ่งเป็นเครื่องมือที่ช่วยในการสร้างและการทดสอบแบบจำลองที่มีความซับซ้อนในรูปแบบที่สามารถเข้าใจได้ง่ายขึ้น และยังช่วยในการประยุกต์ใช้ความน่าจะเป็นในการจำลองพฤติกรรมของระบบ

จากรูปที่ 3.11 แบบจำลองที่สร้างจาก UPPAAL แสดงการพยายามส่งข้อความผ่านช่องทางที่ไม่น่าเชื่อถือ ซึ่งเป็นการจำลองสถานการณ์ที่มีความซับซ้อนในการสื่อสารที่อาจมีข้อผิดพลาดเกิดขึ้นได้ เครื่องมือนี้ช่วยให้สามารถทวนสอบและวิเคราะห์พฤติกรรมของแบบจำลองได้อย่างละเอียดและเชื่อถือได้



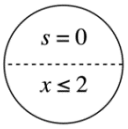
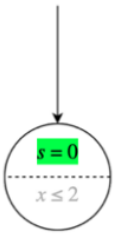
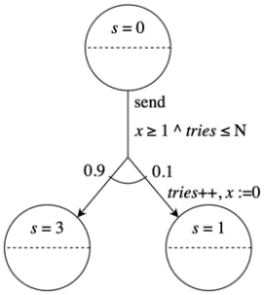
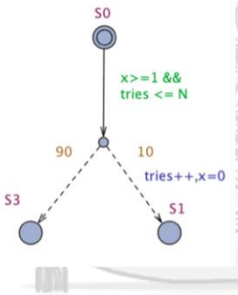
รูปที่ 3.11 แบบแสดงการพยายามส่งข้อความผ่านช่องทางที่ไม่น่าเชื่อถือจากเครื่องมือ UPPAAL

การวิเคราะห์โครงสร้างและพฤติกรรมของแบบจำลองจาก UPPAAL และการแปลงไปยังรหัสปริซึมต้องการความเข้าใจในการทำงานของแต่ละส่วนของแบบจำลอง ตัวอย่างเช่น ตำแหน่งต่างๆ ใน UPPAAL จะใช้วิธีการระบุชื่อที่ชัดเจนและเป็นไปตามรูปแบบข้อความ ซึ่งแตกต่างจากการเก็บค่าเป็นจำนวนเต็มในปริซึมซึ่งใช้ตัวแปร s เพื่อระบุสถานะ การกำหนดค่ายืนยงของตำแหน่งที่ใช้ในการจำลองสถานะต่างๆ ใน UPPAAL จะแสดงเป็นเงื่อนไขต่างๆ ที่ช่วยให้แบบจำลองสามารถประยุกต์ใช้กับเหตุการณ์ที่มีความน่าจะเป็นได้

การแปลงแบบจำลองจาก UPPAAL เป็นปริซึมจำเป็นต้องพิจารณาความแตกต่างเหล่านี้อย่างละเอียดรอบคอบ เพื่อให้แน่ใจว่าแบบจำลองในปริซึมสามารถสะท้อนพฤติกรรมและคุณสมบัติของระบบเดิมจาก UPPAAL ได้อย่างถูกต้อง ตารางที่ 3.2 ให้ตัวอย่างของจุดที่แตกต่างระหว่างแบบจำลองจาก UPPAAL และการจำลองในปริซึมซึ่งเป็นข้อมูลสำคัญที่ผู้วิจัยจะต้องใช้ในการประยุกต์และแปลงแบบจำลองไปยังรหัสปริซึมให้เกิดความถูกต้องและครอบคลุม

การทำความเข้าใจในความแตกต่างของสัญลักษณ์และการใช้งานของตำแหน่ง เงื่อนไขความน่าจะเป็น และเส้นเชื่อม ใน UPPAAL และปริซึมนี้เป็นสิ่งจำเป็นที่จะช่วยให้การแปลงแบบจำลองจากแบบจำลองหนึ่งไปยังอีกแบบจำลองหนึ่งเป็นไปอย่างมีประสิทธิภาพและแม่นยำ ซึ่งจะช่วยให้การวิเคราะห์พฤติกรรมของระบบเวลาจริงสามารถทำได้โดยไม่มีข้อผิดพลาดที่อาจเกิดขึ้นได้จากการไม่เข้าใจในรายละเอียดของระบบ

ตารางที่ 3.2 เปรียบเทียบโครงสร้างของแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นและแบบจำลองที่ได้จากเครื่องมือ UPPAAL

สัญลักษณ์ของแบบจำลอง	สัญลักษณ์ของแบบจำลองบน UPPAAL	คำอธิบาย
		ตำแหน่งใน UPPAAL จะใช้วิธีการระบุชื่อของตำแหน่งซึ่งเก็บใช้รูปแบบข้อความแทนค่าของตัวแปร s ที่เก็บเป็นจำนวนเต็ม ดังนั้นในการแปลงจะใช้วิธีการนับจำนวนตำแหน่งทั้งหมดเพื่อนำมาใช้เป็นขนาดของค่า s และแทนที่แต่ละตำแหน่งเป็นจำนวนนับเริ่มจาก 0
		จุดเริ่มต้น (Initial) จะใช้วงกลมสองชั้นเป็นเป็นจุดระบุจุดเริ่มต้นของออโตมาตา
		แทรนซิชันใน UPPAAL จะไม่ระบุชื่อการกระทำ และกรณีเพิ่มความน่าจะเป็นจะเพิ่มจุด ก่อนใช้เส้นปะหัวลูกศรชี้ออกไปยังตำแหน่งเป้าหมาย โดยที่ค่าของความน่าจะเป็นจะใช้จำนวนเต็มร้อยแทนจำนวนทศนิยม

การทำความเข้าใจโครงสร้างเอ็กซ์เอ็มแอล ของแบบจำลองไทมด์ออโตมาตาความน่าจะเป็น (PTAs) ที่ส่งออกจาก UPPAAL เป็นขั้นตอนแรกที่สำคัญในการเตรียมแปลงแบบจำลองเหล่านี้เป็นภาษาปริซึมแบบจำลองเอ็กซ์เอ็มแอล นี้ประกอบไปด้วยองค์ประกอบที่หลากหลายและมีโครงสร้างที่ซับซ้อน ซึ่งแต่ละส่วนทำหน้าที่อธิบายองค์ประกอบของ PTAs อย่างชัดเจน

ตามที่อธิบายไว้ในตารางที่ 3.3, โครงสร้างเอ็กซ์เอ็มแอล ประกอบไปด้วยส่วนเทมเพลต (Template) ที่มีองค์ประกอบสำคัญ เช่น ชื่อเทมเพลต (name) ประกาศตัวแปร (declaration) ตำแหน่งต่างๆ (location) จุดแยก (branchpoint) และการเปลี่ยนแปลงสถานะ (transition) องค์ประกอบเหล่านี้มีบทบาทในการกำหนดโครงสร้างของแบบจำลองที่จะถูกนำไปใช้ในปริซึม

ตารางที่ 3.3 ตัวอย่างโครงสร้างแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นในรูปแบบเอกสารเอกซ์เอ็มแอล

สัญลักษณ์ของไทมด์ออโตมาตาความน่าจะเป็น	ตัวอย่างรูปแบบเอกสารเอกซ์เอ็มแอล	คำอธิบาย
Template	<template> <name>PTA</name> <declaration> ... </declaration> <location> ... </location> <branchpoint> </branchpoint> <init/> <transition> ... </transition> </template>	เทมเพล็ตของแบบจำลอง ประกอบด้วยอิลิเมนต์ name : ชื่อเทมเพล็ต declaration : ประกาศ location : ตำแหน่ง branchpoint : จุดแยก transition : แทรนซิชัน

Declaration	<declaration>
	clock x;
	int tries = 0;
	const int N = 0;
	</declaration>
		ประกาศใช้เพื่อตัวแปรและเมธอด สำหรับดำเนินการในแบบจำลอง โดยประเภทตัวแปรที่น่าสนใจ ได้แก่ clock : นาฬิกา ใช้ในไทมด์ออโตมาตา

Location	<location id="id0"> <name>S0</name> <label kind="invariant"> x<=2 </label> </location>	ตำแหน่ง ใช้เพื่อกำหนดองค์ประกอบของตำแหน่งที่อยู่แต่ละจุดบนแบบจำลอง โดยมีแอตทริบิวต์ ที่สำคัญได้แก่ id : เอกลักษณ์ของตำแหน่งและ อิลิเมนต์ที่สำคัญได้แก่ name : ชื่อของตำแหน่ง label (Invariant) : ค่ายืนยง เพื่อเป็นเงื่อนไขของตำแหน่ง เช่น x<=2 ที่ใช้การอ้างอิงเอนทิตีตัวอักษรจาก ISO 8859-1 ให้ความหมาย x<=2
Branchpoint	<branchpoint id="id4"></branchpoint>	จุดแยก ใช้เพื่อกระจายแทรนซิชันจากตำแหน่งหนึ่งไปยังตำแหน่งอื่นๆ สองตำแหน่งขึ้นไปโดยมีแอตทริบิวต์ ที่สำคัญได้แก่ id : เอกลักษณ์ของจุดแยก * ต้องไม่ซ้ำกับเอกลักษณ์ของตำแหน่ง
Init	<init ref="id0"/>	จุดเริ่มต้น โดยมีแอตทริบิวต์ ที่สำคัญได้แก่ ref : อ้างอิงเอกลักษณ์ของตำแหน่งที่จะใช้เป็นจุดเริ่มต้น

ตารางที่ 3.3 ตัวอย่างโครงสร้างแบบจำลองไทม์ออโตมาตาความน่าจะเป็นในรูปแบบเอกสารเอกซ์เอ็มแอล (ต่อ)

สัญลักษณ์ของไทม์ออโตมาตาความน่าจะเป็น	ตัวอย่างรูปแบบเอกสารเอกซ์เอ็มแอล	คำอธิบาย
Transition	<transition> <source ref="id3"/> <target ref="id0"/> <label kind="guard"> x>=3 </label> <label kind="assignment"> x:=0 </label> </transition>	แทรนซิชัน ใช้ระบุเส้นทางจากตำแหน่งหนึ่งไปยังตำแหน่งถัดไป มีอิลิเมนต์ที่สำคัญได้แก่ source : แหล่งที่มา ใช้ ref ระบุเอกลักษณ์ของตำแหน่งหรือจุดแยก target : เป้าหมาย ใช้ ref ระบุเอกลักษณ์ของตำแหน่งหรือจุดแยก label (guard) : เงื่อนไขที่จะไปอีกตำแหน่งหนึ่งเช่น x<=3 label (assignment) : การมอบหมายงานเมื่อผ่านเงื่อนไข เช่น ปรับค่า x เป็น 0 จาก x:=0 * ref สามารถใช้เอกลักษณ์ของจุดแยกได้
Transition	<transition> <source ref="id4"/> <target ref="id2"/> <label kind="probability"> 90 </label> </transition>	แทรนซิชัน จากจุดแยก มีอิลิเมนต์ที่สำคัญได้แก่ Label (probability) : ความน่าจะเป็นที่จะไปยังอีกตำแหน่งหนึ่ง * ถ้าแหล่งที่มาเป็นจุดแยกแล้วเป้าหมายต้องเป็นตำแหน่งเท่านั้น

การแปลงเอกสารเอ็กซ์เอ็มแอล เป็นภาษาปริซึมนั้นต้องทำด้วยความระมัดระวังเพื่อให้สามารถรักษาความถูกต้องของโครงสร้างและพฤติกรรมของแบบจำลองดั้งเดิมไว้ได้ ซึ่งจำเป็นต้องใช้ความเข้าใจที่ลึกซึ้งและการตรวจสอบอย่างละเอียดในการแปลงข้อมูลเหล่านี้ให้เป็นโค้ดที่สามารถรันบนเครื่องมือปริซึมได้

3.3 แปลงแบบจำลองไทม์ออโตมาตาความน่าจะเป็นในรูปแบบเอกสารเอกซ์เอ็มแอลเป็นแบบจำลองในรูปแบบรหัสปริซึม

เมื่อพิจารณาถึงการแปลงแบบจำลองไทม์ออโตมาตาความน่าจะเป็น (PTAs) จากเครื่องมือ UPPAAL ไปสู่ภาษาปริซึมผู้วิจัยจำเป็นต้องมีความเข้าใจอย่างลึกซึ้งในโครงสร้างของเอกสารเอ็กซ์เอ็มแอล ที่ส่งออกจาก UPPAAL และมีความชำนาญในกฎการแปลงที่กำหนดไว้เพื่อการทำงานที่แม่นยำและมีประสิทธิภาพสูงสุดในการเปลี่ยนแปลงเป็นรหัสปริซึมการแปลงเอกสารเอ็กซ์เอ็มแอล สู่รหัสปริซึมเป็นกระบวนการที่เกี่ยวข้องกับการวิเคราะห์และการจับคู่โครงสร้างเอ็กซ์เอ็มแอล ที่ซับซ้อนด้วยความเที่ยงตรงต่อกฎการแปลงที่กำหนดไว้ ซึ่งหมายถึงการต้องทำความเข้าใจในทุกส่วนของ

โครงสร้างเอ็กซ์เอ็มแอล ตั้งแต่องค์ประกอบต่างๆ เช่น templates, declarations, locations, branchpoints, transitions ไปจนถึงค่าเริ่มต้นและค่ายืนยงที่กำหนดให้กับแต่ละส่วน

จากโครงสร้างองค์กอบของแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นในรูปแบบเอกสารเอกซ์เอ็มแอลที่ได้จากเครื่องมือ UPPAAL และกฎการแปลงแบบจำลองไปสู่รหัสปริซึมที่ได้กำหนดไว้ ผู้วิจัยจะดำเนินการสร้างเครื่องมือเพื่อใช้ในกระบวนการแปลงเอกสารเอกซ์เอ็มแอลเป็นรหัสปริซึมโดยตรวจสอบแต่ละองค์ประกอบของเอกสารเอกซ์เอ็มแอลเทียบกับกฎการแปลงสำหรับแปลงเป็นรหัสปริซึมได้ตามตัวอย่างตารางที่ 3.4

การพัฒนาเครื่องมือดังกล่าวจะต้องพิจารณาถึงการเชื่อมต่อกับเครื่องมือ UPPAAL และปริซึมอย่างเหมาะสมเพื่อให้สามารถใช้งานได้อย่างราบรื่นและทำการแปลงข้อมูลได้อย่างแม่นยำ ผู้วิจัยจะต้องทำการทดสอบและตรวจสอบเครื่องมือเหล่านี้เพื่อให้แน่ใจว่าสามารถแปลงแบบจำลอง PTAs ไปสู่รหัสปริซึมได้อย่างถูกต้องและเชื่อถือได้ กระบวนการนี้จะต้องมีการทดสอบอย่างเข้มข้นเพื่อตรวจจับข้อผิดพลาดและปรับปรุงการแปลงให้สอดคล้องกับมาตรฐานที่ต้องการ

ตารางที่ 3.4 ตัวอย่างการแปลงเอกสารเอกซ์เอ็มแอลไปเป็นรหัสปริซึมเมื่อใช้กฎการแปลงที่กำหนดไว้

ตัวอย่างรูปแบบเอกสารเอกซ์เอ็มแอล	ตัวอย่างรหัสปริซึม (PRISM code)	คำอธิบาย

`<declaration>`
`clock x;`
`int tries = 0;`
`const int N = 0;`
`</declaration>`
	`x: clock;`
	`tries: int init 0;`
	`cont int N;`
		ประกาศตัวแปร เป็นส่วนที่กำหนดค่าเริ่มต้นของตัวแปรที่จะนำมาใช้ในกระบวนการประมวลผลของปริซึม

`<location id="id0">`
`<name>S0</name>`
`<label kind="invariant">`
`x&lt;=2`
`</label>`
`</location>`
`<init ref="id0"/>`
	`s: [0..3] init 0;`
	`// 0 = S0`
	…
	`Invariant`
	`(s=0=>x<=2)`
	`endinvariant`
		กำหนดตัวแปรสำหรับตำแหน่งแทนที่ด้วย s โดยแทน id เป็นจำนวนนับตามตำแหน่งอิลิเมนต์ที่อยู่ในเอกสารเอกซ์เอ็มแอลเพื่อระบุขนาดของตัวแปร s และระบุค่าเริ่มต้นตามลำดับของตำแหน่ง ตามอิลิเมนต์ init กำหนด

`<transition>`
`<source ref="id3"/>`
`<target ref="id0"/>`
`<label kind="guard" x="178" y="-51">x&gt;=3</label>`
`<label kind="assignment" x="34" y="-51">x:=0</label>`
`</transition>`
	`[] s=1 &x>=3 ->`
	`(s'=0) & (x'=0);`
		แทรนซิชันจาก ตำแหน่งที่ระบุในอิลิเมนต์แหล่งที่มา (source) ไปยังตำแหน่งที่ระบุในอิลิเมนต์เป้าหมาย โดยมีตัวป้องกันและการอัปเดตค่าจากอิลิเมนต์ป้าย (label) ที่ระบุนำมาระบุในขึ้นต้นด้วยสัญลักษณ์ [ ] เพื่อประกาศแทรนซิชันกลุ่มรหัสที่อยู่ก่อนเครื่องหมาย -> จะระบุเงื่อนไขที่สามารถเริ่มต้นแทรนซิชันนี้ และหลัง -> จะระบุการกำหนดค่าใหม่

ตารางที่ 3.4 ตัวอย่างการแปลงเอกสารเอกซ์เอ็มแอลไปเป็นรหัสปริซึมเมื่อใช้กฎการแปลงที่กำหนดไว้ (ต่อ)

ตัวอย่างรูปแบบเอกสารเอกซ์เอ็มแอล	ตัวอย่างรหัสปริซึม (PRISM code)	คำอธิบาย
<transition> <source ref="id0"/> <target ref="id4"/> <label kind="guard" x="8" y="34">x>=1 && tries <= N</label> </transition> <transition> <source ref="id4"/> <target ref="id2"/> <label kind="probability" x="161" y="153">90</label> </transition> <transition> <source ref="id4"/> <target ref="id3"/> <label kind="assignment" x="187" y="51">tries++,x=0</label> <label kind="probability" x="161" y="76">10</label> </transition>	**[] s=0 &x>=1 &tries<=N -> 0.9 : (s'=3) + 0.1 : (s'=1) & (tries'=tries+1) &(x'=0);**	แทรนซิชันที่ระบุค่าความน่าจะเป็นโดยใช้วิธีระบุอิลิเมนต์เป้าหมายไปยังจุดแยกโดยมีตัวป้องกัน เพิ่มแทรนซิชันจากจุดแยกไปยังตำแหน่งเป้าหมาย ในปริซึม จะประกาศเริ่มต้นด้วย [ ] ตามด้วยเงื่อนไขของตัวป้องกันที่จะใช้ในการเริ่มต้นแทรนซิชัน และพื้นที่หลังเครื่องหมาย -> จะระบุกลุ่มค่าของความน่าจะเป็นต่อด้วยเครื่อง : เพื่อระบุการกำหนดค่าใหม่ และใช้เครื่องหมาย + สำหรับระบุค่าของความน่าจะเป็นในกลุ่มถัดไป

เมื่อได้โครงสร้างองค์ประกอบของแบบจำลองไทม์ออโตมาตาความน่าจะเป็นในรูปแบบเอกสารเอกซ์เอ็มแอล ผู้วิจัยจะดำเนินการสร้างเครื่องมือที่จะนำมาใช้สำหรับการแปลงแบบจำลองในรูปแบบเอกสารเอกซ์เอ็มแอลไปเป็น รหัสปริซึม เพื่อนำมาใช้ในการทวนสอบในเครื่องมือตรวจสอบแบบจำลองปริซึม โดยผลลัพธ์ที่คาดหวังจะได้รับจากการแปลงแสดงได้ ดังตัวอย่างดังรูปที่ 3.12

```
1  pta
2  const int N;
3  module transmitter
4      s: [0..3] init 0;
5      tries: [0..N+1] init 0;
6      x: clock;
7      invariant
8          (s=0=>x<=2) & (s=1=>x<=5)
9      endinvariant
10     [send] s=0 &x>=1 &tries<=N
11        -> 0.9 : (s'=3)
12         + 0.1 : (s'=1)&(tries'=tries+1)&(x'=0);
13     [retry] s=1 &x>=3 -> (s'=0)&(x'=0);
14     [quit] s=0 &tries>N -> (s'=2);
15 endmodule
```

รูปที่ 3.12 ตัวอย่างรหัสปริซึมที่คาดหวังจากการแปลงไทม์ออโตมาตาความน่าจะเป็นในรูปแบบเอกสารเอกซ์เอ็มแอล

3.4 การทวนสอบแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นจากตัวตรวจสอบแบบจำลองความน่าจะเป็นปริซึม

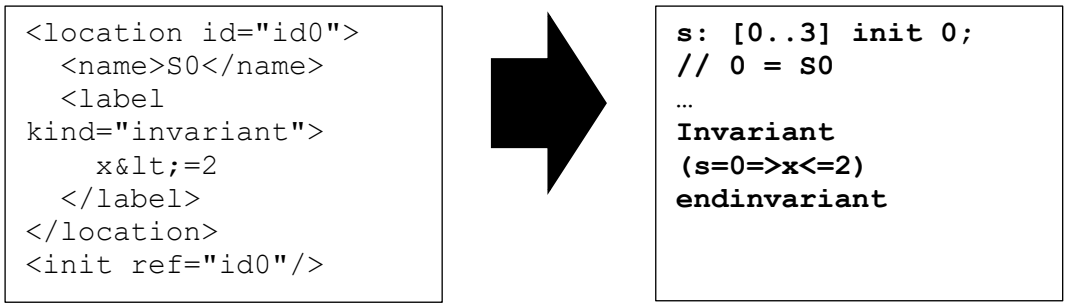
การตรวจสอบแบบจำลองไทมด์ออโตมาตาความน่าจะเป็น (PTAs) จากเอ็กซ์เอ็มแอล ไปเป็นภาษาปริซึมเป็นกระบวนการที่ซับซ้อนซึ่งต้องการความเข้าใจที่ลึกซึ้งเกี่ยวกับโครงสร้างและพฤติกรรมของระบบที่จะทดสอบ กระบวนการนี้ต้องใช้เครื่องมือที่มีความสามารถในการวิเคราะห์ทั้งความถูกต้องทางไวยกรณ์ และความถูกต้องทางอรรถศาสตร์(ความหมาย) ของแบบจำลองที่ถูกแปลง นอกจากนี้ยังต้องมีการเปรียบเทียบผลลัพธ์เชิงปริมาณกับข้อกำหนดหรือคุณสมบัติเฉพาะที่ต้องการตรวจสอบ เช่น การตอบสนองในเวลาจำกัด หรือความน่าจะเป็นในการบรรลุเงื่อนไขบางอย่าง

การทวนสอบแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นเพื่อตรวจสอบว่าการแปลงจากเอ็กซ์เอ็มแอล ไปเป็นภาษาปริซึมนั้นถูกต้อง สามารถทำได้โดยการตรวจสอบความสอดคล้องของโครงสร้างและพฤติกรรมระหว่างสองรูปแบบ โดยแบ่งหัวข้อการทวนสอบดังนี้

1) ตรวจสอบความถูกต้องของการประกาศตัวแปร (Declarations) ทุกตัวแปรในเอ็กซ์เอ็มแอลจะต้องถูกแปลงไปเป็นตัวแปรในปริซึมโดยมีชนิดข้อมูลและค่าเริ่มต้นที่ตรงกัน ตัวอย่างเช่นรูปที่ 3.13 ในเอกสารเอ็กซ์เอ็มแอล มีการประกาศตัวแปรไว้ 3 ตัว ตัวแปร x เป็นนาฬิกา ตัวแปร tries เป็นตัวเลขเริ่มต้นที่ 0 และตัวแปร N เป็นค่ายืนยงตัวเลขเริ่มต้นที่ 0 ดังนั้นตัวแปรในภาษาปริซึมจะต้องประกาศค่าตัวแปรเดียวกันที่ชนิดข้อมูลและค่าเริ่มต้นตรงกัน

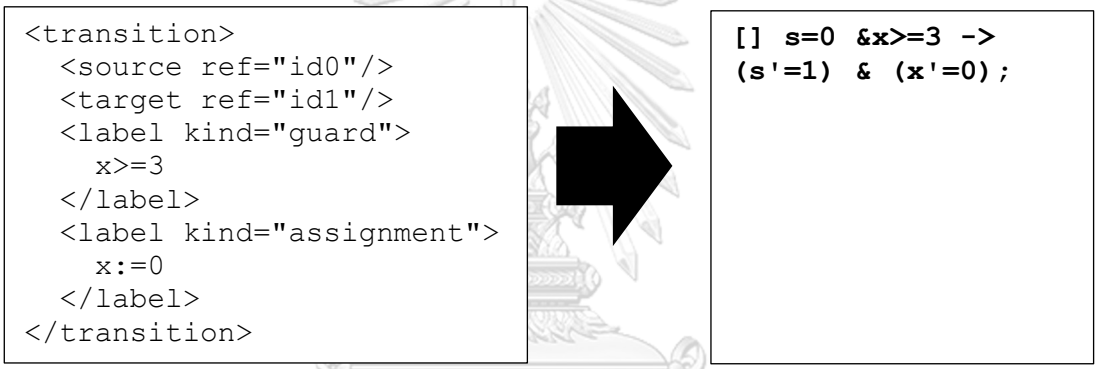
รูปที่ 3.13 ตัวอย่างการแปลงตัวแปรในเอ็กซ์เอ็มแอล ไปสู่ภาษาปริซึมที่ถูกต้อง

2) ตรวจสอบความสอดคล้องของตำแหน่ง (Locations) ตำแหน่งทั้งหมดในเอ็กซ์เอ็มแอล จะต้องมีการแทนที่โดย states ในปริซึมและมีการกำหนดค่ายืนยง (invariants) ที่สอดคล้องกัน ตัวอย่างเช่น รูปที่ 3.14 แสดงถึงการแปลงตำแหน่งของแบบจำลอง PTAs ที่ได้จาก UPPAAL ในรูปแบบเอ็กซ์เอ็มแอล ถูกแปลงเปนภาษาปริซึมได้อย่างถูกต้อง ทั้งส่วนของตำแหน่งแหน่งเริ่มต้นและเงื่อนค่ายืนยง



รูปที่ 3.14 ตัวอย่างการแปลงตำแหน่งและเงื่อนไขค่ายืนยงในเอ็กซ์เอ็มแอล ไปสู่ภาษาปริซึมที่ถูกต้อง

3) ตรวจสอบเงื่อนไขแทรนซิชัน (Transitions) แทรนซิชันทั้งหมดในเอ็กซ์เอ็มแอล ควรตรงกับแทรนซิชันในปริซึมพร้อมด้วยเงื่อนไข (guards) และการอัพเดท (updates) ตัวอย่างเช่น รูปที่ 3.15 แสดงถึงการแปลงแทรนซิชันของแบบจำลอง PTAs ที่ได้จาก UPPAAL ในรูปแบบเอ็กซ์เอ็มแอล โดยมีเงื่อนไขแทรนซิชัน และการอัพเดทถูกแปลงไปสู่ภาษาปริซึมที่ถูกต้อง



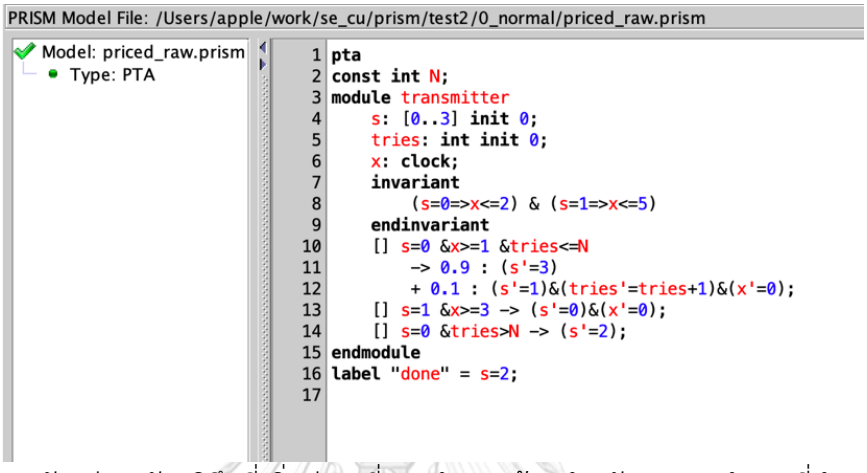
รูปที่ 3.15 ตัวอย่างการแปลงแทรนซิชัน (Transitions) ในเอ็กซ์เอ็มแอล ไปสู่ภาษาปริซึมที่ถูกต้อง

การทวนสอบแบบจำลองอาจรวมถึงการเปรียบเทียบค่าความน่าจะเป็น (probabilities) และเงื่อนไขเวลา (timers) ในแทรนซิชันสถานะ เพื่อให้แน่ใจว่าพฤติกรรมของระบบในรหัสปริซึมนั้นสอดคล้องกับแบบจำลอง PTAs ต้นฉบับอย่างเต็มรูปแบบและไม่มีความผิดพลาดในการแปลงความน่าจะเป็นหรือเงื่อนไขเวลา

เพื่อตรวจสอบว่าแปลงตัวป้องกันและการอัพเดทในแทรนซิชันจากเอ็กซ์เอ็มแอลจาก UPPAAL ได้อย่างครบถ้วนและถูกต้อง ผู้วิจัยจะกำหนดกรณีทดสอบโดยกำหนดแบบจำลองไทม์ออโตมาตาความน่าจะเป็นเป็นตัวอย่างทดสอบเพื่อเปรียบเทียบแทรนซิชันในแบบจำลองด้วยวิธีการแปลงรหัสปริซึมด้วยมือให้ได้ผลลัพธ์ที่คาดหวังและเปรียบเทียบกับผลลัพธ์ที่ได้จากเครื่องมือสนับสนุนการแปลงแบบจำลองไทม์ออโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึม โดยผลลัพธ์จริงที่เกิดขึ้นต้องตรงกับรหัสปริซึมที่แปลงด้วยมือ

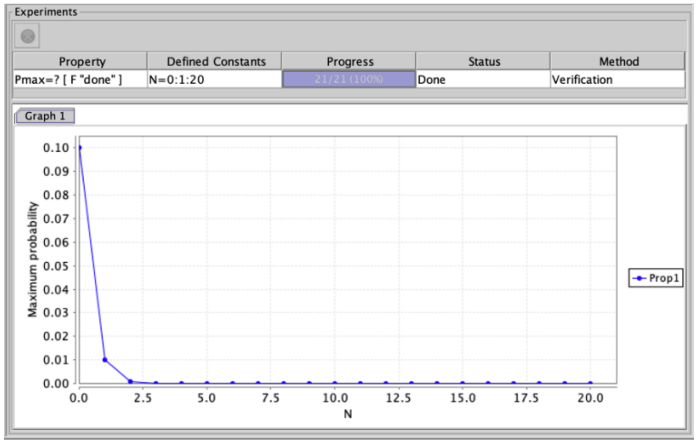
หลังจากได้นำแบบจำลองมาแปลงให้อยู่ในรูปแบบรหัสปริซึมแล้ว ผู้วิจัยจะนำรหัสปริซึมมาเพื่อใช้ในการทวนสอบความน่าจะเป็นในตัวตรวจสอบความน่าจะเป็นปริซึมตามรูปที่ 3.16 ในการ

ดำเนินการผ่านออโตมาตาความน่าจะเป็นที่ได้ โดยผู้วิจัยดำเนินการตั้งตัวแปรค่าคงที่ (constants) ไว้เพื่อใช้เป็นค่านำเข้าในแต่ละรอบของการตรวจสอบเพื่อหาแนวโน้มที่จะเพิ่ม หรือลดความน่าจะเป็นที่เกิดขึ้นในแบบจำลอง และกำหนดป้าย (label) สำหรับหรับกำหนดตำแหน่งที่ใช้วัดความน่าจะเป็นที่เกิดขึ้นเมื่อถึงตำแหน่งเป้าหมาย



รูปที่ 3.16 ตัวอย่างรหัสปริซึมที่เพิ่มค่าคงที่และกำหนดป้ายสำหรับจบการทำงานที่ตำแหน่งที่ 2

เมื่อเข้าสู่กระบวนการทดลองเพื่อตรวจสอบค่าความน่าจะเป็นสูงสุด (Maximum probability) ผู้วิจัยใช้วิธีการกำหนดกรอบของค่าคงที่ที่จะนำมาใช้ทดสอบและลำดับแทรนซิชันค่าคงที่ในแต่ละรอบ เพื่อตรวจสอบปัจจัยที่จะส่งผลให้ค่าความน่าจะเป็นอยู่ในขอบเขตที่ยอมรับได้ ตัวอย่างเช่น การทวนสอบหาความน่าจะเป็นสูงสุดในการพยายามส่งข้อความผ่านช่องทางที่ไม่น่าเชื่อถือตามรูปที่ 3.17 โดยในแต่ละรอบจะมีความน่าจะเป็นที่ 10% ในการส่งไม่สำเร็จ จะใช้วิธีการส่งใหม่อีกครั้งอย่างน้อยกี่รอบเพื่อลดความน่าจะเป็นในการส่งไม่สำเร็จให้น้อยที่สุด โดยผลลัพธ์ที่ได้คือ หากส่งไม่สำเร็จควรส่งใหม่อย่างน้อย 2 ครั้งที่จะทำให้การส่งมีความสำเร็จสูงที่สุด



รูปที่ 3.17 ตัวอย่างผลการทดลองการลดความน่าจะเป็นที่ส่งข้อความไม่สำเร็จ และมีการส่งใหม่โดยการหนดจำนวน N เริ่มจาก 0 ถึง 20

เพื่อตรวจสอบว่าแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นที่แปลงเป็นรหัสปริซึมมีองค์องค์ประกอบของตำแหน่งและแทรนซิชันที่ถูกต้องตามแบบจำลองที่ใช้เป็นข้อมูลนำเข้า ผู้วิจัยใช้วิธีการไล่ลำดับของตำแหน่งและแทรนซิชันที่เกิดขึ้นในระบบจากตัวตรวจสอบแบบจำลองความน่าจะเป็นปริซึม ในการตรวจสอบผลลัพธ์ที่ได้ว่ามีความสอดคล้องกับแผนภาพแบบจำลองนำเข้าจากลำดับพฤติกรรมของแบบจำลองและเส้นทางที่เกิดขึ้นตรงกัน รวมถึงการตรวจสอบจำนวนตำแหน่งและเส้นทางที่ระบุในแบบจำลองตรงกัน

บทที่ 4

การออกแบบเครื่องมือสนับสนุนการทำการแปลงแบบจำลองเป็นรหัสปริซึม

การออกแบบเครื่องมือที่สำคัญซึ่งสนับสนุนการแปลงภาษาทางคณิตศาสตร์ของแบบจำลองไทมด์ออโตมาตาความน่าจะเป็น (PTAs) ไปยังภาษาปริซึม (PRISM) ซึ่งเป็นภาษาสำหรับการจำลองและการตรวจสอบแบบจำลองที่มีความซับซ้อน การพัฒนาเครื่องมือนี้ไม่เพียงแต่จะเป็นการสร้างสะพานเชื่อมระหว่างทฤษฎีและปฏิบัติ แต่ยังจะแสดงให้เห็นถึงการประยุกต์ใช้กฎการแปลงที่ได้ถูกกำหนดไว้ในบทที่ 3 อีกด้วย การออกแบบเครื่องมือนี้จะต้องปฏิบัติตามหลักการและกระบวนการของวิศวกรรมซอฟต์แวร์ ซึ่งรวมถึงขั้นตอนต่างๆ ดังนี้

การออกแบบเครื่องมือสนับสนุนการทำการแปลงแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นเป็นรหัสปริซึมนั้นเริ่มต้นด้วยขั้นตอนการวิเคราะห์และกำหนดความต้องการ (Requirements Analysis and Specification) ซึ่งเป็นหัวใจสำคัญของกระบวนการออกแบบซอฟต์แวร์ตามหลักวิศวกรรมซอฟต์แวร์ ในส่วนนี้ ความต้องการของผู้ใช้งานและเงื่อนไขของระบบจะถูกวิเคราะห์อย่างละเอียดเพื่อกำหนดขอบเขตและเป้าหมายของเครื่องมือที่จะพัฒนา

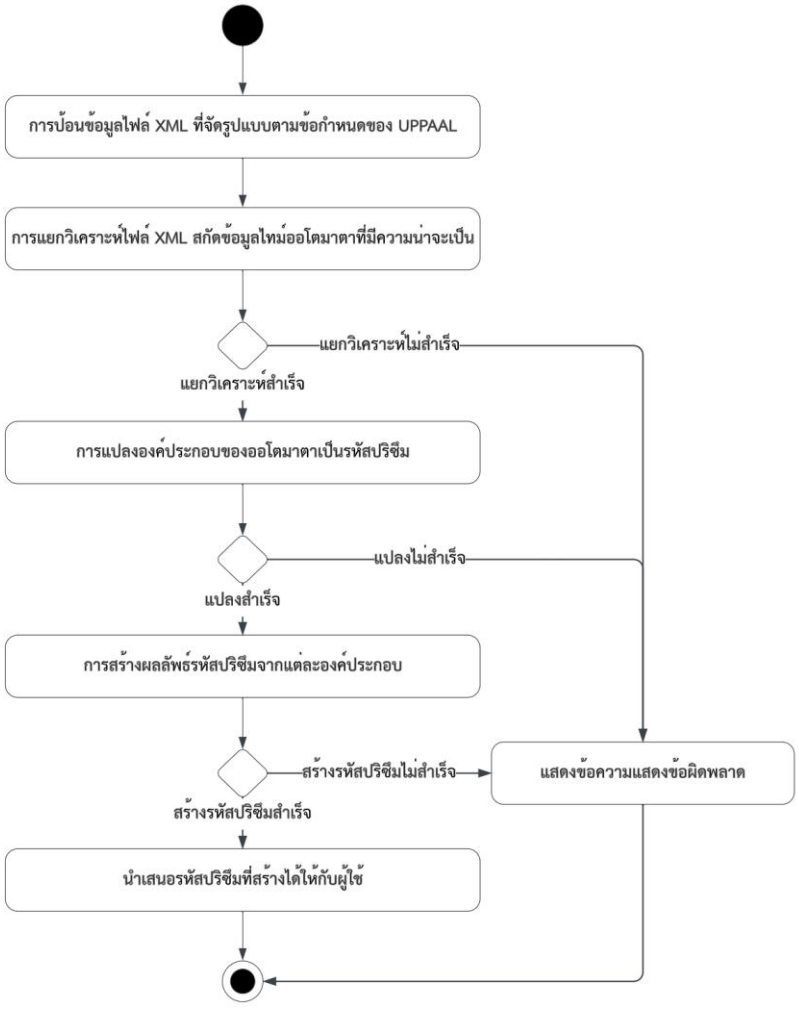
การกำหนดขอบเขตของเครื่องมือที่จะพัฒนานั้นครอบคลุมถึงการทำงานดังนี้

1) ระบบสามารถอ่านและแยกวิเคราะห์เอกสารเอ็กซ์เอ็มแอล ที่สอดคล้องกับรูปแบบ UPPAAL โดยแยกข้อมูลที่เกี่ยวข้องเกี่ยวกับออโตมาตาเวลาความน่าจะเป็น รวมถึงสถานะ การเปลี่ยนผ่าน ข้อจำกัดด้านเวลา และองค์ประกอบความน่าจะเป็น
2) ระบบสามารถแปลงไทมด์ออโตมาตาความน่าจะเป็นที่แยกวิเคราะห์เป็นรูปแบบภาษาปริซึม ซึ่งเกี่ยวข้องกับการแปลงสถานะ การเปลี่ยนภาพ ข้อมูลความน่าจะเป็นและเวลาให้เป็นไวยากรณ์ที่เข้ากันได้กับภาษาและสร้างแบบจำลองของปริซึมตามกฎการแปลงที่ระบุไว้ในบทที่ 3
3) ระบบสามารถตรวจสอบเอกสารเอ็กซ์เอ็มแอล อินพุตกับรูปแบบและโครงสร้างที่คาดหวังได้ นอกจากนี้ยังสามารถตรวจสอบข้อผิดพลาดเพื่อให้แน่ใจว่าออโตมาตะที่แสดงในเอ็กซ์เอ็มแอล นั้นมีความสอดคล้องกันทางตรรกะและสอดคล้องกับบรรทัดฐานของไทมด์ออโตมาตาความน่าจะเป็น
4) เครื่องมือนี้มีอินเตอร์เฟส (ไม่ว่าจะเป็นกราฟิกหรือคอมมานด์ไลน์) สำหรับผู้ใช้ในการป้อนเอกสารเอ็กซ์เอ็มแอล และระบุพารามิเตอร์หรือตัวเลือกที่จำเป็นสำหรับ

กระบวนการแปลง นอกจากนี้ยังแสดงผลตอบกลับที่เกี่ยวข้อง เช่น การยืนยันความสำเร็จหรือข้อความแสดงข้อผิดพลาด

5) หลังจากแปลงออโตมาตาแล้ว ระบบสามารถสร้างและส่งออกโค้ดปริซึมที่เกี่ยวข้องได้ ผลลัพธ์นี้สามารถนำเสนอในรูปแบบที่เป็นมิตรกับผู้ใช้ เช่น ไฟล์ข้อความหรือภายในอินเตอร์เฟส พร้อมสำหรับการใช้งานในการตรวจสอบแบบจำลองที่ใช้ปริซึมหรือการวิเคราะห์เพิ่มเติม

รูปที่ 4.1 แสดงถึงกระบวนการหลักของเครื่องมือที่ใช้ในการแปลงแบบจำลองไทมด์ออโตมาตาคที่มีความน่าจะเป็นสู่รูปแบบรหัสปริซึม สามารถทำความเข้าใจได้อย่างชัดเจนผ่านแผนภาพการไหลที่จัดทำขึ้น ซึ่งแผนภาพนี้แสดงขั้นตอนต่างๆ และการไหลของการทำงานตั้งแต่เริ่มต้นจนจบกระบวนการ ดังที่จะกล่าวต่อไปนี้



รูปที่ 4.1 แผนภาพการไหลของกระบวนการแปลงไทมด์ออโตมาตาคที่มีความน่าจะเป็นสู่รูปแบบรหัสปริซึม

4.1.1 ภาษาที่ใช้ในการพัฒนาเครื่องมือการแปลงแบบจำลองเป็นรหัสปริซึม

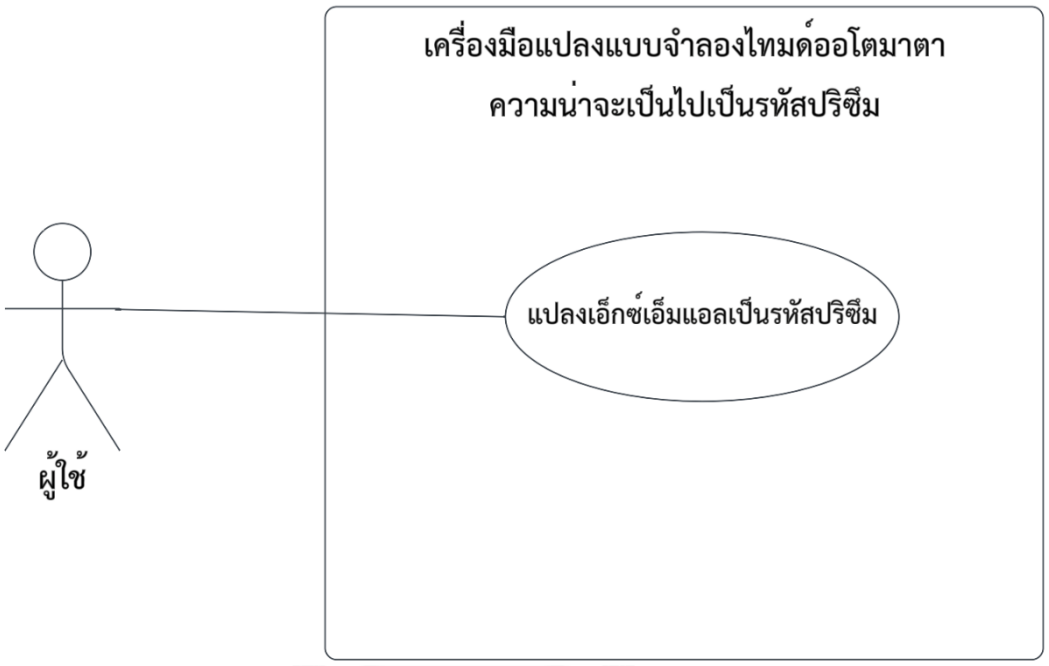
ในการพัฒนาเครื่องมือแปลงไทมด์ออโตมาตาคที่มีความน่าจะเป็นเป็นรหัสปริซึม ผู้วิจัยได้ตัดสินใจเลือกใช้ภาษาจาวา (JAVA) เพราะคุณสมบัติที่โดดเด่นและเหมาะสมกับงาน ดังต่อไปนี้

1) ความเป็นอิสระทางแพลตฟอร์ม : จาวาสามารถทำงานได้บนแพลตฟอร์มใดๆ ที่มี JVM (Java Virtual Machine) ซึ่งหมายความว่าโปรแกรมเดียวกันสามารถทำงานได้บนระบบปฏิบัติการใดก็ได้ นี่เป็นข้อดีสำคัญสำหรับเครื่องมือนี้ที่อาจถูกใช้งานบนแพลตฟอร์มต่างๆ
2) การสนับสนุนไลบรารี : จาวามีไลบรารีมาตรฐานที่หลากหลาย โดยเฉพาะสำหรับการจัดการกับเอ็กซ์เอ็มแอล ผ่านแพ็คเกจเช่น javax.xml ซึ่งช่วยให้การอ่านและแปลงข้อมูลเอ็กซ์เอ็มแอล ง่ายขึ้น นอกจากนี้ ไลบรารีมาตรฐานของจาวายังรองรับการทำงานกับเครือข่าย การพัฒนา GUI และการโต้ตอบกับระบบอื่นๆ อีกด้วย
3) การเขียนโปรแกรมแบบวัตถุ (OOP) : ความสามารถของจาวาใน OOP ช่วยให้สามารถสร้างโค้ดที่สามารถขยาย แก้ไข และบำรุงรักษาได้ง่าย นี่เป็นประโยชน์อย่างมากสำหรับเครื่องมือนี้ ซึ่งสามารถแบ่งแยกความสามารถต่างๆ เช่น การแยกวิเคราะห์เอ็กซ์เอ็มแอล ตรรกะการแปลง และการจัดการ UI เป็นโมดูลที่แยกจากกันได้
4) ประสิทธิภาพและการปรับขนาด : แม้ว่าจาวาอาจไม่ได้เร็วเท่ากับภาษาอื่นๆ เช่น C หรือ C++ ในบางงาน แต่ประสิทธิภาพของมันก็เพียงพอสำหรับการประมวลผลและการแปลงเอกสารเอ็กซ์เอ็มแอล กลไกการจัดการหน่วยความจำและการเก็บขยะของ Java ได้รับการปรับให้เหมาะสมมากสำหรับการจัดการชุดข้อมูลขนาดใหญ่ ซึ่งเป็นประโยชน์เมื่อต้องจัดการกับแบบจำลองออโตมาตาที่ซับซ้อน
5) การบูรณาการกับเครื่องมือและภาษาอื่นๆ: จาวาสามารถรวมกับภาษาโปรแกรมอื่นและเทคโนโลยีต่างๆ ได้อย่างง่ายดาย นี่เป็นข้อดีหากเครื่องมือต้องการใช้งานร่วมกับซอฟต์แวร์อื่นหรือไลบรารีที่เขียนด้วยภาษาอื่น
6) ความมั่นคงและเชื่อถือได้ : จาวาเป็นที่รู้จักในเรื่องของความมั่นคงและเชื่อถือได้ ซึ่งเป็นสิ่งสำคัญสำหรับเครื่องมือที่ผู้ใช้พึ่งพาในการแปลงแบบจำลองออโตมาตาอย่างถูกต้อง

โดยสรุปแล้ว ความสามารถของจาวา เช่น การทำงานแบบอิสระทางแพลตฟอร์ม การสนับสนุนไลบรารี ความมั่นคงและเชื่อถือได้ ประสิทธิภาพ การปรับขนาด และคุณสมบัติแบบวัตถุที่ทำให้ Java เป็นตัวเลือกที่ยอดเยี่ยมสำหรับการสร้างเครื่องมือที่ต้องการการประมวลผลเอ็กซ์เอ็มแอล ตรรกะการแปลง และความสามารถในการขยายตัวได้ ประสิทธิภาพ ความปลอดภัย และความสามารถในการรวมกันเพิ่มเติมทำให้มันเหมาะสมยิ่งขึ้นสำหรับนำมาใช้ในการพัฒนาเครื่องมือแปลงแบบจำลองนี้

4.1.2 กรณีใช้งานของเครื่องมือการแปลงแบบจำลองเป็นรหัสปริซึม

เพื่อให้เข้าใจได้ชัดเจนเกี่ยวกับขอบเขตและหน้าที่การทำงานของเครื่องมือที่ช่วยในการแปลงแบบจำลองไทมด์ออโตมาตาโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึม หัวข้อถัดไปจะนำเสนอแผนภาพยูสเคส (Use Case Diagram) ในรูปที่ 4.2 ที่อธิบายถึงกรณีการใช้งานต่างๆ ของเครื่องมือที่วางแผนจะพัฒนา เพื่อให้เห็นภาพรวมของการทำงานและสามารถจับต้องได้ถึงปฏิสัมพันธ์ระหว่างผู้ใช้และระบบ และสามารถอธิบายแผนภาพยูสเคสในตารางที่ 4.1



รูปที่ 4.2 แผนภาพยูสเคสของเครื่องมือแปลงแบบจำลองไทมด์ออโตมาตาโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึม

ตารางที่ 4.1 คำอธิบายแผนภาพยูสเคสของเครื่องมือแปลงแบบจำลองเป็นรหัสปริซึม

องค์ประกอบ	คำอธิบาย
ผู้ใช้	ผู้ที่ทำงานร่วมกับระบบเพื่อแปลงแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึม
เครื่องมือแปลง	ระบบที่ออกแบบมาเพื่อแปลงแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึม
แปลงเอ็กซ์เอ็มแอลเป็นรหัสปริซึม	กรณีการใช้งานหลักที่ผู้ใช้เริ่มต้นกระบวนการแปลงแบบจำลองรูปแบบเอ็กซ์เอ็มแอล ไปเป็นรหัสปริซึม

โดยกรณีใช้งานหลักของแผนภาพยูสเคสของเครื่องมือแปลงนี้ คือแปลงเอ็กซ์เอ็มแอลเป็นรหัสปริซึม สามารถอธิบายกรณีใช้งานได้ตามตารางที่ 4.2

ตารางที่ 4.2 กรณีการใช้งาน UC1: แปลงเอ็กซ์เอ็มแอลเป็นรหัสปริซึม

รหัสกรณีการใช้งาน	UC1
ชื่อกรณีการใช้งาน	แปลงเอ็กซ์เอ็มแอลเป็นรหัสปริซึม
ผู้แสดงบทบาท	ผู้ใช้
คำอธิบาย	กระบวนการแปลงไทมด์ออโตมาตาความน่าจะเป็นในรูปแบบเอกสารเอ็กซ์เอ็มแอลไปเป็นรหัสปริซึมเพื่อการตรวจสอบแบบจำลอง
เงื่อนไขก่อนการใช้งาน	ผู้ใช้ต้องมีเอกสารเอ็กซ์เอ็มแอลที่มีโครงสร้างของแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นที่เป็นไปตามเครื่องมือ UPPAAL
ขั้นตอนหลัก	1. ผู้ใช้งานป้อนเอกสารเอ็กซ์เอ็มแอลเข้าระบบ 2. ระบบจะทำการวิเคราะห์และสกัดข้อมูลจากเอกสารเอ็กซ์เอ็มแอลที่ผู้ใช้ป้อนเข้ามา โดยจะแยกส่วนสำคัญ เช่น สถานะ แทรนซิชัน และเงื่อนไขค่ายืนยง ตัวป้องกัน รวมถึงความน่าจะเป็นออกมา 3. จากข้อมูลที่ได้จากการวิเคราะห์เอกสารเอ็กซ์เอ็มแอลระบบจะทำการแปลงข้อมูลเหล่านั้นเป็นรหัสปริซึมตามโครงสร้างที่กำหนดไว้ 4. เมื่อการแปลงสำเร็จ ระบบจะนำเสนอรหัสปริซึมให้กับผู้ใช้ ซึ่งอาจจะแสดงผลบนหน้าจอ
ขั้นตอนทางเลือก	2. หากเอกสารเอ็กซ์เอ็มแอลมีรูปแบบที่ไม่ถูกต้องหรือไม่สามารถวิเคราะห์ได้ระบบจะแจ้งข้อผิดพลาด 3. หากข้อมูลที่วิเคราะห์เสร็จแล้วไม่สามารถแปลงเป็นรหัสปริซึมได้ระบบจะแจ้งข้อผิดพลาด
เงื่อนไขหลังการใช้งาน	ผู้ใช้จะได้รับรหัสปริซึมในรูปแบบที่สามารถใช้งานได้ในเครื่องมือตรวจสอบแบบจำลองปริซึม

จากตารางกรณีการใช้งานของเครื่องมือการแปลงไทมด์ออโตมาตาโตมาตาความน่าจะเป็นเป็นรหัสปริซึม สามารถเห็นว่าเครื่องมือนี้ถูกออกแบบมาเพื่อรองรับกระบวนการทำงานที่สมบูรณ์แบบและมีประสิทธิภาพ ตั้งแต่จุดเริ่มต้นด้วยการป้อนข้อมูลในรูปแบบเอกสารเอ็กซ์เอ็มแอล ไปจนถึงขั้นตอนสุดท้ายของการเสนอรหัสปริซึมที่ถูกต้องออกมาให้ผู้ใช้ กรณีการใช้งานไม่เพียงแค่เป็นส่วนสำคัญที่ทำให้เครื่องมือดำเนินงานได้สมบูรณ์ แต่ยังช่วยให้ผู้ใช้สามารถรับมือกับข้อผิดพลาดที่อาจเกิดขึ้นได้อย่างมีประสิทธิภาพ

4.2 การออกแบบระบบและสถาปัตยกรรมซอฟต์แวร์ (System and Software Architecture Design)

หัวข้อนี้จะสำรวจความสำคัญของการสร้างโครงสร้างที่เข้าใจง่าย พร้อมทั้งสามารถรองรับการเปลี่ยนแปลงและการขยายตัวในอนาคต จะกล่าวถึงการเลือกแนวทางและเทคนิคในการออกแบบที่มีประสิทธิภาพ ซึ่งรวมถึงการพิจารณาการทำงานร่วมกันของส่วนประกอบต่างๆ การจัดการข้อมูล การสื่อสารระหว่างระบบย่อย และการบูรณาการระบบภายนอก ดำเนินการประเมินข้อกำหนดทางเทคนิคและความต้องการของระบบเพื่อมั่นใจว่าสถาปัตยกรรมที่ออกแบบมานั้นสามารถตอบสนองต่อเป้าหมายที่ตั้งไว้ได้

4.2.1 ข้อกำหนดทางเทคนิคสำหรับเครื่องมือที่แปลงไทมด์ออโตมาตาความน่าจะเป็นจากเอ็กซ์เอ็มแอล (รูปแบบ UPPAAL) เป็นรหัสปริซึม

เพื่อให้ซอฟต์แวร์ที่พัฒนาขึ้นสามารถทำงานได้อย่างมีประสิทธิภาพและเป็นไปตามความต้องการของระบบ ผู้วิจัยได้ตั้งข้อกำหนดทางเทคนิคสำหรับเครื่องมือที่แปลงไทมด์ออโตมาตาแบบมีความน่าจะเป็นไปเป็นรหัสปริซึม ดังนี้

1) ข้อกำหนดของแพลตฟอร์ม : แพลตฟอร์มที่เครื่องมือสนับสนุนการแปลง จะทำงานโดยมีความต้องการขอระบบดังแสดงในตารางที่ 4.3 ดังนี้

ตารางที่ 4.3 ความต้องการของระบบสำหรับเครื่องสนับสนุนการแปลงแบบจำลองเป็นรหัสปริซึม

หัวข้อการพิจารณา	รายละเอียด	คำอธิบายเพิ่มเติม
ระบบปฏิบัติการ	Windows, Linux, และ macOS	การพัฒนาเครื่องมือควรให้ความสำคัญกับการทำงานได้หลายระบบปฏิบัติการเพื่อความสามารถในการเข้าถึงของผู้ใช้จำนวนมาก
CPU	ขั้นต่ำ 1 CORE	ข้อกำหนดขั้นต่ำแนะนำสำหรับการประมวลผลที่ราบรื่น

ตารางที่ 4.3 ความต้องการของระบบสำหรับเครื่องสนับสนุนการแปลงแบบจำลองเป็นรหัสปริซึม (ต่อ)

หัวข้อการพิจารณา	รายละเอียด	คำอธิบายเพิ่มเติม
RAM	ขั้นต่ำ 1 GB	ข้อกำหนดของปริมาณขั้นต่ำ RAM ที่เพียงพอสำหรับการจัดการไฟล์ขนาดใหญ่
Storage	ขั้นต่ำ 128 MB	ข้อกำหนดขั้นต่ำของพื้นที่เก็บข้อมูลเพียงพอสำหรับติดตั้งและจัดเก็บไฟล์

2) เกณฑ์ประสิทธิภาพ : กำหนดเมตริกประสิทธิภาพที่เครื่องมือควรตอบสนองได้ รวมถึงความเร็วในการประมวลผลเอกสารเอ็กซ์เอ็มแอล ขนาดใหญ่ ประสิทธิภาพการใช้หน่วยความจำ และความตอบสนองของเครื่องมือ แสดงได้ตามตารางที่ 4.4

ตารางที่ 4.4 เมตริกประสิทธิภาพที่เครื่องมือควรตอบสนองได้ตามหัวข้อที่กำหนด

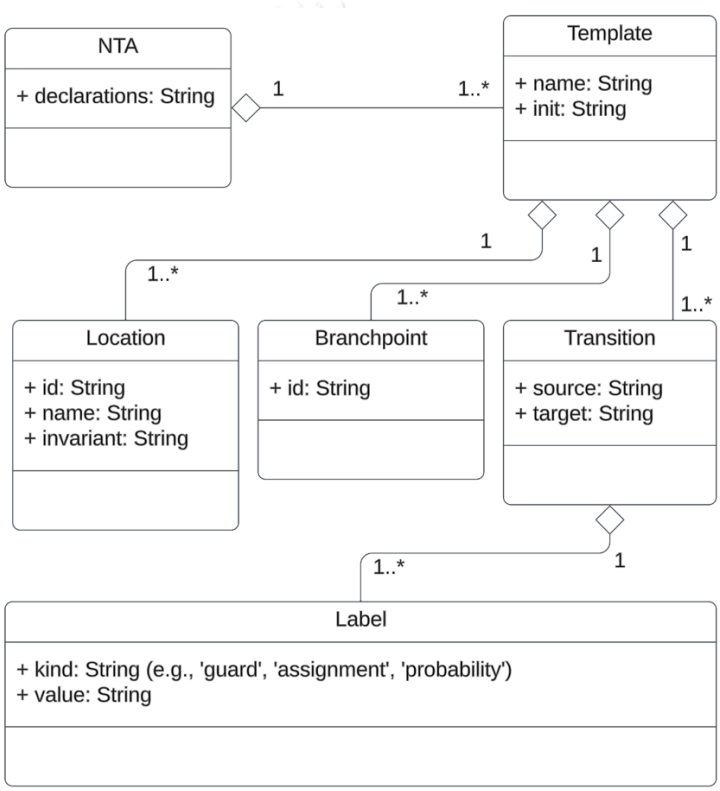
เมตริกประสิทธิภาพ	**ค่าเป้าหมาย**	**คำอธิบาย**	**ผลกระทบต่อผู้ใช้**
ความเร็วในการประมวลผลเอกสารเอ็กซ์เอ็มแอล	ประมวลผลไฟล์ขนาดใหญ่ไม่เกิน 10 mbภายใน 5 นาที	การประมวลผลเอกสารเอ็กซ์เอ็มแอล ขนาดใหญ่ให้เสร็จสิ้นอย่างรวดเร็ว เพื่อไม่ให้เกิดการหน่วงในกระบวนการทำงาน	ลดเวลารอและเพิ่มความพึงพอใจของผู้ใช้
ประสิทธิภาพการใช้หน่วยความจำ	ใช้หน่วยความจำไม่เกิน 75% ของ 1 GB	หลีกเลี่ยงการใช้หน่วยความจำเกินขีดจำกัด เพื่อไม่ให้เกิดปัญหาการช้าหรือค้าง	รักษาความเสถียรและประสิทธิภาพการทำงานของเครื่องมือ
ความตอบสนองของเครื่องมือ	ตอบสนองต่อคำสั่งภายใน 1-2 วินาที	ปฏิกิริยาต่อการโต้ตอบของผู้ใช้ทันที เพื่อความสะดวกและประสิทธิภาพ	เพิ่มความพึงพอใจและเชื่อมั่นในเครื่องมือ

4.2.2 ข้อมูลเชิงโครงสร้างของแบบจำลองในรูปเอ็กซ์เอ็มแอลที่ได้จาก UPPAAL

เมื่อพิจารณาถึงการวิเคราะห์และออกแบบซอฟต์แวร์ที่สามารถแปลงไทมด์ออโตมาตาแบบมีความน่าจะเป็นจากเอ็กซ์เอ็มแอล ในรูปแบบ UPPAAL เป็นรหัสปริซึม การเข้าใจโครงสร้างและองค์ประกอบของแบบจำลองเอ็กซ์เอ็มแอล นั้นมีความสำคัญยิ่ง ข้อมูลเอ็กซ์เอ็มแอล ที่ได้จาก

UPPAAL นำเสนอโครงสร้างที่ละเอียดและซับซ้อนซึ่งมีลักษณะเฉพาะของแบบจำลอง PTA เช่น สถานะ เงื่อนไข แทรนซิชันสถานะ และค่าความน่าจะเป็น

จากรูปที่ 4.3 จะอธิบายโครงสร้างของไดอะแกรมคลาส (Class Diagram) ที่สะท้อนถึงข้อมูลเหล่านี้ เพื่อให้เข้าใจถึงการแปลงข้อมูลเอ็กซ์เอ็มแอล เป็นรูปแบบที่สามารถจัดการได้ง่ายขึ้นในซอฟต์แวร์ การสร้างไดอะแกรมคลาสจะช่วยให้เห็นภาพรวมของโครงสร้างและการเชื่อมโยงของส่วนประกอบต่างๆ ภายในแบบจำลองเอ็กซ์เอ็มแอล นี้ ซึ่งรวมถึงการแสดงความสัมพันธ์ระหว่างส่วนประกอบต่างๆ เช่น สถานะ แทรนซิชัน และเงื่อนไขต่างๆ ที่เกี่ยวข้องกับการเวลาและความน่าจะเป็นในแบบจำลอง PTA



รูปที่ 4.3 โครงสร้างของเอ็กซ์เอ็มแอลสำหรับแบบจำลองในรูปแบบ UPPAAL

คลาสเหล่านี้สะท้อนถึงโครงสร้างพื้นฐานของไฟล์เอ็กซ์เอ็มแอลสำหรับไทมด์ออโตมาตาแบบมีความน่าใน UPPAAL โดยแต่ละคลาสแสดงถึงส่วนประกอบหลักที่พบในเอกสารเอ็กซ์เอ็มแอล ตัวอย่างเช่น คลาส 'Template' จะแทนที่โครงสร้าง <template> ในเอ็กซ์เอ็มแอล และมีความสัมพันธ์กับคลาส 'Location' และ 'Transition' เพื่อแสดงถึงตำแหน่งและแทรนซิชันภายในแต่ละแบบจำลองออโตมาตา

ตารางที่ 4.5 ถึงตารางที่ 4.10 จะอธิบายรายละเอียดของโครงสร้างของคลาสพื้นฐานของไฟล์เอ็กซ์เอ็มแอลสำหรับไทมด์ออโตมาตาแบบมีความน่าจะเป็น

ตารางที่ 4.5 รายละเอียดของคลาส “NTA”

ชื่อคลาส	NTA (Network of Timed Automata)
คำอธิบาย	คลาส NTA เป็นรากฐานของแบบจำลอง PTA ที่รวมข้อความประกาศต่างๆ เช่น การประกาศตัวแปรเวลา
แอทริบิวต์	
- declarations	ข้อความประกาศตัวแปรหรือนาฬิกาที่ใช้ในแบบจำลอง PTA ทั้งหมด

ตารางที่ 4.6 รายละเอียดของคลาส “Template”

ชื่อคลาส	Template
คำอธิบาย	คลาส Template แทนการเรียกการใช้งานของออโตมาตาซึ่งอาจมีหลาย instances ในระบบ
แอทริบิวต์	
- name	ชื่อของแม่แบบออโตมาตา เช่น PTA
- init	รหัสของสถานะเริ่มต้นในแบบจำลองนี้

ตารางที่ 4.7 รายละเอียดของคลาส “Location”

ชื่อคลาส	Location
คำอธิบาย	คลาส Location แทนสถานะหรือจุดที่อยู่ภายในแบบจำลองออโตมาตา
แอทริบิวต์	
- id	รหัสประจำตัวของสถานะภายในแม่แบบ
- name	ชื่อของสถานะ
- invariant	เงื่อนไขที่ต้องปฏิบัติตามในสถานะนี้

ตารางที่ 4.8 รายละเอียดของคลาส “Branchpoint”

ชื่อคลาส	Branchpoint
คำอธิบาย	คลาส Branchpoint แทนจุดแยกของแทรนซิชันสถานะที่อยู่ภายในแบบจำลองไทมด์ออโตมาตาความน่าจะเป็น ใช้ร่วมกับการกำหนดค่าความน่าจะเป็น
แอทริบิวต์	
- id	รหัสประจำตัวของจุดแยกภายในแม่แบบ

ตารางที่ 4.9 รายละเอียดของคลาส “Transition”

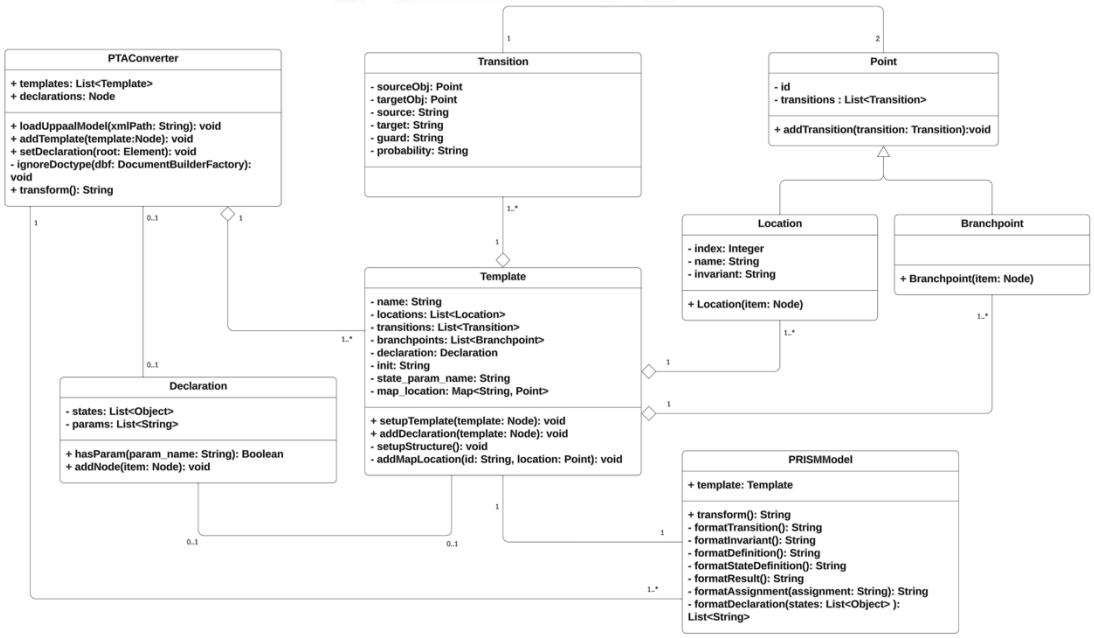
ชื่อคลาส	Transition
คำอธิบาย	คลาส Transition แทนแทรนซิชันจากสถานะหนึ่งไปยังสถานะอื่น
แอทริบิวต์	
- source	รหัสของสถานะเริ่มต้นหรือรหัสของจุดแยกของแทรนซิชัน
- target	รหัสของสถานะปลายทางหรือรหัสของจุดแยกของแทรนซิชัน

ตารางที่ 4.10 รายละเอียดของคลาส “Label”

ชื่อคลาส	Label
คำอธิบาย	คลาส Label ใช้สำหรับการแสดงข้อมูลเพิ่มเติมเกี่ยวกับ Transition เช่น เงื่อนไขที่จำเป็นสำหรับแทรนซิชัน
แอทริบิวต์	
- kind	ประเภทของป้ายกำกับ เช่น 'guard', 'assignment', 'probability'
- value	ค่าของป้ายกำกับ เช่น เงื่อนไขหรือค่าความน่าจะเป็น

4.2.3 ข้อมูลเชิงโครงสร้างของเครื่องมือสนับสนุนการแปลงแบบจำลองเป็นรหัสปริซึม

ในการพัฒนาเครื่องมือสนับสนุนการแปลงแบบจำลองไทม์ออโตมาตาโตมาตาความน่าจะเป็นเป็นรหัสปริซึม การออกแบบไดอะแกรมคลาสมีบทบาทสำคัญในการทำให้เห็นภาพรวมของโครงสร้างและส่วนประกอบต่างๆ ภายในระบบ หัวข้อนี้จะนำเสนอการวิเคราะห์และอธิบายถึงวิธีการทำงานของส่วนประกอบต่างๆ ภายในระบบ เพื่อให้มั่นใจว่าการพัฒนาเครื่องมือนั้นสามารถตอบสนองต่อความต้องการและการใช้งานจริงของผู้ใช้ได้อย่างเหมาะสมและมีประสิทธิภาพ



รูปที่ 4.4 ไดอะแกรมคลาสของเครื่องมือสนับสนุนการแปลง*แบบจำลองเป็นรหัสปริซึม*

รูปที่ 4.4 คือแผนภาพคลาสที่เป็นส่วนสำคัญของเครื่องมือสนับสนุนการแปลง โดยทำหน้าที่เป็นแผนที่แนะนำโครงสร้างและความสัมพันธ์ระหว่างส่วนประกอบต่างๆ ในการแปลงแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นให้เป็นรหัสปริซึม การออกแบบไดอะแกรมนี้ได้รับต้นแบบจากโครงสร้างเอกสารเอ็กซ์เอ็มแอล ของ UPPAAL ซึ่งเป็นมาตรฐานในการจัดการกับแบบจำลองไทมด์ออโตมาตา และยังรวมไปถึงการนำเอากฎการแปลงที่ได้กำหนดไว้มาใช้ร่วมกับการพัฒนา เพื่อให้เครื่องมือสามารถประมวลผลได้อย่างแม่นยำและเชื่อถือได้ ซึ่งจะช่วยในการพัฒนาและการทดสอบระบบที่มีความซับซ้อนสูงในด้านความน่าจะเป็นและเวลาที่เกี่ยวข้อง

ตารางที่ 4.11 ถึงตารางที่ 4.18 จะอธิบายรายละเอียดของโครงสร้างของคลาสพื้นฐานของเครื่องมือสนับสนุนการแปลงแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึม

ตารางที่ 4.11 รายละเอียดของคลาส “PTAConverter” ของเครื่องมือ

ชื่อคลาส	PTAConverter
คำอธิบาย	แปลงเอกสารเอ็กซ์เอ็มแอล ของ UPPAAL เป็นโค้ดปริซึม
แอทริบิวต์	
- templates	รายการของ Template ที่นำมาใช้ในการแปลง
- declarations	โหนดสำหรับการประกาศตัวแปรของแบบจำลอง
เมธอด	
- loadUppaalModel	โหลดแบบจำลอง UPPAAL จากเอกสารเอ็กซ์เอ็มแอล
- addTemplate	รหัสของสถานะปลายทางหรือรหัสของจุดแยกของแทรนซิชัน
- setDeclaration	ปรับเปลี่ยนหรือตั้งค่าการประกาศในแบบจำลอง
- transform	แปลงแบบจำลอง UPPAAL ที่โหลดเข้ามาเป็นโค้ดปริซึม
- ignoreDocType	แก้ไขข้อผิดพลาดหลังจากแยกวิเคราะห์เอกสารเอ็กซ์เอ็มแอล

ตารางที่ 4.12 รายละเอียดของคลาส “PRISMModel” ของเครื่องมือ

ชื่อคลาส	PRISMModel
คำอธิบาย	จัดการโครงสร้างและข้อมูลของแบบจำลองปริซึม
แอทริบิวต์	
- template	รายการของ Template ที่นำมาใช้ในการแปลง
เมธอด	
- transform	ใช้เพื่อการเปลี่ยนแปลงหรือแปลงโครงสร้างเป็นรหัสปริซึม
- formatTransition	ใช้เพื่อการจัดรูปแบบแทรนซิชันภายในแบบจำลองเป็นรหัสปริซึม

ตารางที่ 4.12 รายละเอียดของคลาส “PRISMModel” ของเครื่องมือ (ต่อ)

- formatInvariant	ใช้เพื่อการจัดรูปแบบเงื่อนไข invariant ในแบบจำลองเป็นรหัสปริซึม
- formatDefinition	ใช้ในการจัดรูปแบบการนิยามของแบบจำลอง อาทิเช่นการนิยามของตัวแปร
- formatStateDefinition	ใช้เพื่อการจัดรูปแบบนิยามของสถานะต่างๆ ในแบบจำลอง
- formatResult	ใช้เพื่อการจัดรูปแบบผลลัพธ์ที่ได้จากการแปลง
- formatAssignment	ใช้เพื่อการจัดรูปแบบการกำหนดค่าหรือการจัดการกับการกำหนดค่า
- formatDeclaration	ใช้เพื่อการจัดรูปแบบการประกาศสถานะต่างๆ ในแบบจำลอง

ตารางที่ 4.13 รายละเอียดของคลาส “Template” ของเครื่องมือ

ชื่อคลาส	Template
คำอธิบาย	เป็นโครงสร้างพื้นฐานของแบบจำลองปริซึม ที่กำหนดสถานะและแทรนซิชันที่เกิดขึ้นภายในแบบจำลองนั้นๆ
แอทริบิวต์	
- name	ชื่อของเทมเพลต
- locations	ลิสต์ของสถานที่ต่างๆ ในเทมเพลต
- transitions	ลิสต์ของแทรนซิชันสถานะที่เกิดขึ้นในเทมเพลต
- branchpoints	ลิสต์ของจุดแยกที่อาจเกิดขึ้นในกระบวนแทรนซิชันสถานะ
- declaration	ออบเจ็คที่จัดการกับการประกาศต่างๆ ภายในเทมเพลต
- init	ค่าเริ่มต้นของเทมเพลต
- state_param_name	ชื่อของพารามิเตอร์สถานะในเทมเพลต
- map_location	แมปที่เชื่อมชื่อสถานที่กับออบเจ็ค Point ที่แทนสถานะหรือจุดภายในเทมเพลต
เมธอด	
- setupTemplate	ใช้เพื่อตั้งค่าเทมเพลต หรือเตรียมเทมเพลตให้พร้อมใช้งาน
- addDeclaration	ใช้เพื่อเพิ่มการประกาศใหม่เข้าไปในเทมเพลต
- setupStructure	ใช้สำหรับการตั้งค่าโครงสร้างภายในของเทมเพลต
- addMapLocation	ใช้เพื่อเพิ่มการแมประหว่างชื่อสถานที่กับออบเจ็ค Point ที่แทนสถานะหรือจุดภายในเทมเพลต

ตารางที่ 4.14 รายละเอียดของคลาส “Point” ของเครื่องมือ

ชื่อคลาส	Point
คำอธิบาย	แสดงถึงจุดหรือสถานะในแบบจำลอง
แอทริบิวต์	
- id	รหัสประจำของจุด
- transitions	ลิสต์ของแทรนซิชันสถานะที่ออกจากจุดนี้
เมธอด	
- addTransition	เพิ่มแทรนซิชันสถานะใหม่เข้าไปในลิสต์ของแทรนซิชัน

ตารางที่ 4.15 รายละเอียดของคลาส “Location” ของเครื่องมือ

ชื่อคลาส	Location (สืบทอดมาจาก Point)
คำอธิบาย	แสดงถึงตำแหน่งในแบบจำลอง
แอทริบิวต์	
- index	ดัชนีของสถานที่
- name	ชื่อของสถานที่
- invariant	เงื่อนไขที่คงที่สำหรับสถานที่นี้
เมธอด	
- Location	สร้างอินสแตนซ์ของคลาส Location โดยมีพารามิเตอร์เป็น Node จากเอ็กซ์เอ็มแอล

ตารางที่ 4.16 รายละเอียดของคลาส “Branchpoint” ของเครื่องมือ

ชื่อคลาส	Branchpoint (สืบทอดมาจาก Point)
คำอธิบาย	เป็นโครงสร้างพื้นฐานของแบบจำลองปริซึม ที่กำหนดสถานะและแทรนซิชันที่เกิดขึ้นภายในแบบจำลองนั้นๆ
เมธอด	
- Branchpoint	สร้างอินสแตนซ์ของคลาส Branchpoint โดยมีพารามิเตอร์เป็น Node จากเอ็กซ์เอ็มแอล

ตารางที่ 4.17 รายละเอียดของคลาส “Transition” ของเครื่องมือ

ชื่อคลาส	Transition
คำอธิบาย	แสดงถึงการเปลี่ยนแปลงสถานะภายในแบบจำลอง โดยมีจุดเริ่มต้นและจุดหมายปลายทางที่ระบุด้วยออบเจ็ค Point นอกจากนี้ยังมีลักษณะเพิ่มเติมอย่างเงื่อนไข guard และความน่าจะเป็นที่จะช่วยกำหนดเงื่อนไขการเปลี่ยนจากหนึ่งสถานะไปยังอีกสถานะหนึ่ง
แอทริบิวต์	
- sourceObj	ออบเจ็ค Point ที่แทนจุดเริ่มต้นของแทรนซิชัน
- targetObj	ออบเจ็ค Point ที่แทนจุดหมายปลายทางของแทรนซิชัน
- source	ชื่อของจุดเริ่มต้นในแทรนซิชัน
- target	ชื่อของจุดหมายปลายทางในแทรนซิชัน
- guard	เงื่อนไขที่จะต้องเป็นจริงก่อนที่จะเกิดแทรนซิชัน
- probability	ความน่าจะเป็นที่แทรนซิชันนี้จะเกิดขึ้น

ตารางที่ 4.18 รายละเอียดของคลาส “Declaration” ของเครื่องมือ

ชื่อคลาส	Declaration
คำอธิบาย	ใช้สำหรับจัดการการประกาศในแบบจำลองปริซึม โดยมีลิสต์ของสถานะและพารามิเตอร์ที่เกี่ยวข้อง ซึ่งเมท็อดต่างๆ ให้ความสามารถในการตรวจสอบและเพิ่มส่วนประกอบใหม่ๆ ลงในการประกาศนั้นๆ
แอทริบิวต์	
- states	ลิสต์ของสถานะต่างๆ ภายในการประกาศ
เมธอด	
- hasParam	ตรวจสอบว่ามีพารามิเตอร์ที่มีชื่อตรงกับในการประกาศหรือไม่ ส่งกลับค่า Boolean.
- addNode	เพิ่มโหนดใหม่เข้าไปในการประกาศ.

โครงสร้างคลาสจากรูปที่ 4.4 แสดงให้เห็นว่าระบบนี้ประกอบด้วยคลาสต่างๆ ที่มีความสัมพันธ์กันเพื่อแสดงถึงแบบจำลองของปริซึม สำหรับการแปลงเอกสารเอ็กซ์เอ็มแอล ของ UPPAAL คลาส PRISM Model เป็นตัวแทนของแบบจำลองปริซึม ที่มีเทมเพลตและการประกาศต่างๆ คลาส Template จัดการโครงสร้างและประกอบด้วยสถานที่ (Locations) และจุดแยก (Branchpoints) พร้อมกับการเปลี่ยนแปลงสถานะ (Transitions) Location และ Branchpoint

นั้นสืบทอดมาจากคลาส Point ซึ่งเป็นพื้นฐานของสถานที่และจุดในแบบจำลอง คลาส Transition แสดงถึงการเปลี่ยนแปลงสถานะในแบบจำลองที่มีเงื่อนไขเฉพาะ และคลาส Declaration จัดการกับการประกาศต่างๆ ในแบบจำลอง โครงสร้างนี้สะท้อนถึงการทำงานร่วมกันของส่วนประกอบต่างๆ ในการสร้างและจัดการแบบจำลองที่ซับซ้อน

บทที่ 5

การพัฒนาเครื่องมือสนับสนุนการแปลงแบบจำลองเป็นรหัสปริซึม

ในบทนี้จะอธิบายเกี่ยวกับการพัฒนาเครื่องมือสนับสนุนการแปลงแบบจำลองอัตโนมัติที่จะต่อยอดจากบทที่ 4 ที่ได้ทำการออกแบบไว้แล้ว โดยได้อธิบายถึงกรณีใช้งาน (Use Case) และแผนภาพแสดงคลาส (Class Diagram) ซึ่งในบทที่ 5 จะเน้นไปที่การนำแผนการออกแบบที่ได้จากบทที่ 4 มาสู่การดำเนินการพัฒนาเครื่องมือสนับสนุนการแปลงแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึม โดยจะมีขั้นตอนต่างๆ ดังนี้

5.1 เครื่องมือที่ใช้ในการพัฒนา

การพัฒนาเครื่องมือสนับสนุนการแปลงแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นให้เป็นรหัสปริซึมผู้วิจัยได้ดำเนินการโดยใช้เครื่องมือหลากหลายดังนี้:

1) UPPAAL รุ่น 4.1.24 สำหรับสร้างแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นในรูปแบบของแผนภาพแบบจำลองของระบบ และนำออกมาเป็นเอกสารเอ็กซ์เอ็มแอล ซึ่งเป็นข้อมูลสำคัญสำหรับกระบวนการแปลง
2) PRISM รุ่น 4.7 เป็นเครื่องมือวิเคราะห์แบบจำลองอัตโนมัติที่จะนำมาใช้สำหรับการตรวจสอบและยืนยันความถูกต้องของผลลัพธ์ที่ได้จากการแปลงแบบจำลอง
3) openjdk รุ่น 21.0.1 ใช้เป็นรันไทม์สำหรับเครื่องมือสนับสนุนการแปลง ที่พัฒนาด้วยภาษาจาวา ซึ่งช่วยให้รองรับการทำงานหลายแพลตฟอร์ม
4) php-fpm รุ่น 7.4.33 ทำหน้าที่เป็นรันไทม์ของระบบติดต่อประสานงานโปรแกรมประยุกต์ (API) สำหรับสร้างส่วนต่อประสานระหว่างเครื่องมือและผู้ใช้งานที่เรียกจากเว็บบราวเซอร์
5) angular รุ่น 16.2.12 สำหรับสร้างส่วนต่อประสานกับผู้ใช้งานรองรับการทำงานแบบไดนามิกของเครื่องมือสนับสนุนการแปลงแบบจำลองเป็นรหัสปริซึม
6) nginx รุ่น 1.23.1 เป็นเว็บเซิร์ฟเวอร์สำหรับติดตั้งเครื่องมือสนับสนุนการแปลงแบบจำลองเป็นรหัสปริซึม และกำหนดเครือข่ายบนสำหรับเรียกใช้งานส่วนต่อประสานบนเว็บบราวเซอร์
7) VS Code รุ่น 1.84.2 เป็นสภาพแวดล้อมการพัฒนาที่สนับสนุนการเขียนโปรแกรมภาษาจาวา ทำให้การพัฒนาเป็นไปอย่างราบรื่นและมีประสิทธิภาพ

5.2 พัฒนาเครื่องมือสนับสนุนการแปลงแบบจำลองเป็นรหัสปริซึม

เพื่อให้เครื่องมือสนับสนุนการแปลงสามารถถูกสร้างและทำงานได้อย่างถูกต้องและเหมาะสมกับความต้องการของผู้ใช้งาน ผู้วิจัยได้พัฒนาเครื่องมือโดยปฏิบัติตามการออกแบบที่ระบุไว้ในบทที่ 4 ที่กล่าวถึงองค์ประกอบของพัฒนาระบบสนับสนุนการแปลงที่มีความชัดเจน ดังนั้นในหัวข้อนี้จะอธิบายถึงขั้นตอนการพัฒนาเครื่องมือ ส่วนต่อประสานกับผู้ใช้ และการนำมาใช้บนอินเทอร์เน็ตเพื่อให้ผู้ใช้งานสามารถเข้าถึงและใช้งานจากที่ไหนก็ได้ ผ่านเว็บบราวเซอร์ในรูปแบบของเว็บแอปพลิเคชัน

5.2.1 สร้างเครื่องมือสนับสนุนการแปลงตามที่ออกแบบ

จากแผนภาพแบบจำลองของระบบที่ได้ออกแบบไว้ในบทที่ 4 ในส่วนของแผนภาพกรณีใช้งาน (Use Case) ที่ได้อธิบายถึงบทบาทของผู้ใช้งานที่เกี่ยวข้องกับระบบ ผู้วิจัยได้นำมาวิเคราะห์เพื่อกำหนดลักษณะการทำงานของแต่ละโมดูลในเครื่องมือ แต่ละกรณีใช้งานได้ถูกแปลงเป็นส่วนประกอบของระบบที่สามารถพัฒนาและทดสอบได้อย่างอิสระพร้อมทั้งนำมาผสานรวมกันให้เป็นระบบงานที่สมบูรณ์

ผู้วิจัยได้ตรวจสอบและวิเคราะห์การไหลของข้อมูลทั้งหมดภายในระบบ เพื่อระบุจุดที่ข้อมูลถูกสร้าง แปรรูป และถูกจัดเก็บ ขั้นตอนนี้ช่วยในการกระบวนการจัดการข้อมูลที่มีประสิทธิภาพ สำหรับไดอะแกรมคลาสโครงสร้างและความสัมพันธ์ของคลาสต่างๆ ได้ถูกแปลงเป็นรหัสภาษาจาวาที่ใช้ในการพัฒนาเครื่องมือ ผู้วิจัยได้สร้างคลาสต่างๆ ที่มีความสอดคล้องกับไดอะแกรมคลาสและมีการทดสอบความถูกต้องของคลาสเหล่านั้นอย่างละเอียด

เพื่อเพิ่มความเร็วในกระบวนการประมวลผล ผู้วิจัยได้เลือกเก็บข้อมูลของแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นที่วิเคราะห์แล้วไว้ในหน่วยความจำชั่วคราว (RAM) ข้อมูลนี้ถูกจัดเก็บในรูปแบบของออบเจกต์ (Object) ซึ่งสอดคล้องกับโครงสร้างของเอกสารเอ็กซ์เอ็มแอล ที่นำเข้ามาจาก UPPAAL การใช้งานออบเจกต์ช่วยให้การจัดการข้อมูลและการเรียกใช้ฟังก์ชันการทำงาน (เมธอด) ในขณะที่โปรแกรมกำลังทำงานเป็นไปอย่างรวดเร็วและมีประสิทธิภาพ ในส่วนของคลาส PRISMModel มีการออกแบบเพื่อนำเอาออบเจกต์ของแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นมาแปลงเป็นรหัสปริซึมชั่วคราว ซึ่งหลังจากนั้นจะนำผลลัพธ์ไปเก็บไว้ในหน่วยความจำถาวร (Harddisk) เพื่อให้สามารถนำไปแสดงผลลัพธ์จากการแปลงแบบจำลองเป็นรหัสปริซึมได้อย่างถูกต้องและรวดเร็ว

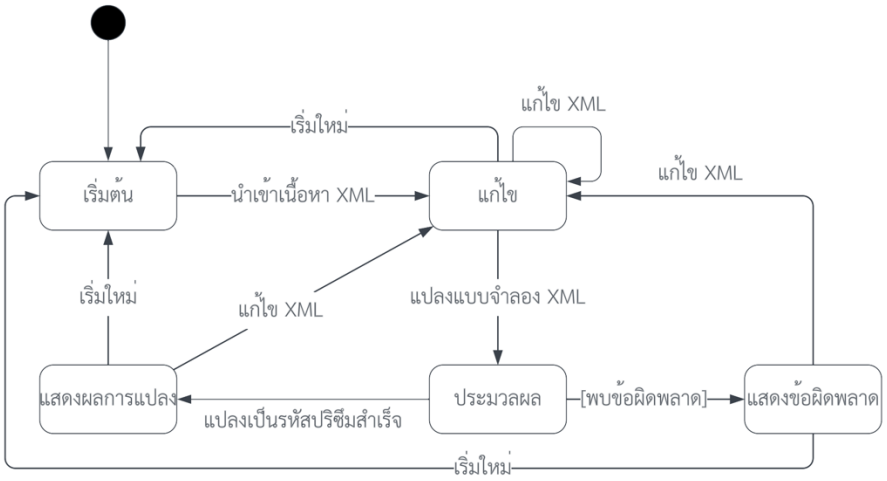
การใช้ RAM เป็นการเพิ่มประสิทธิภาพในการเข้าถึงและประมวลผลข้อมูลได้เร็วขึ้นเมื่อเทียบกับการเก็บข้อมูลในหน่วยความจำถาวร นอกจากนี้การออกแบบนี้ยังช่วยลดเวลาที่ใช้ในการโหลดและประมวลผลข้อมูล ทำให้เครื่องมือสนับสนุนการแปลงสามารถตอบสนองต่อคำขอของผู้ใช้ได้อย่างรวดเร็วและมีประสิทธิภาพ

การพัฒนาเครื่องมือของผู้วิจัยได้ดำเนินการตามแนวทางการเขียนโปรแกรมเชิงวัตถุ (OOP) โดยอิงตามแบบและกระบวนการที่ได้กำหนดไว้ในบทที่ 4 และตามกฎการแปลงที่ระบุไว้ในบทที่ 3 ของการวิจัย โปรแกรมที่พัฒนาขึ้นได้ถูกเขียนด้วยภาษาจาวา เพื่อให้สามารถทำงานได้อย่างหลากหลายบนแพลตฟอร์มต่างๆ ที่รองรับภาษาจาวา ความสามารถของภาษาจาวาในการทำงานได้หลายแพลตฟอร์มทำให้มันเป็นทางเลือกที่เหมาะสมในการพัฒนาเครื่องมือนี้

ผู้วิจัยได้พัฒนาเครื่องมือและจัดเก็บเป็นไฟล์นามสกุล .jar ซึ่งเป็นไฟล์ที่สามารถนำไปเรียกใช้งานได้ง่ายบนอุปกรณ์ต่างๆ ที่มี Java Runtime Environment (JRE) ไฟล์ .jar นี้มีข้อดีคือสามารถถูกนำไปใช้งานบนระบบปฏิบัติการต่างๆ ได้อย่างไม่มีข้อจำกัด ไม่ว่าจะเป็น Windows, macOS, Linux หรือแม้กระทั่งบนอุปกรณ์พกพาที่รองรับ Java การใช้งานไฟล์ .jar ช่วยให้กระบวนการการแปลงแบบจำลองสามารถดำเนินการได้อย่างราบรื่นและมีความยืดหยุ่นในการปรับตามความต้องการใช้งานที่หลากหลาย

5.2.2 การสร้างส่วนต่อประสานผู้ใช้งาน

ในส่วนของการพัฒนาส่วนต่อประสานผู้ใช้งาน ผู้วิจัยได้ให้ความสำคัญกับการออกแบบที่เข้าใจง่ายและใช้งานสะดวก เพื่อให้ผู้ใช้สามารถทำงานกับเครื่องมือได้อย่างราบรื่น ส่วนต่อประสานถูกออกแบบให้สามารถใช้งานได้ผ่านเว็บบราวเซอร์ในรูปแบบของเว็บแอปพลิเคชัน เพื่อให้เครื่องมือมีความยืดหยุ่นในการใช้งาน ในรูปที่ 5.1 จะอธิบายถึงสถานะของการแปลงแบบจำลองไปเป็นรหัสปริซึม



รูปที่ 5.1 ไฟไนต์สเตตแมชชีนของเครื่องมือการแปลงแบบจำลองไปเป็นรหัสปริซึม

จากรูปที่ 5.1 คือแบบจำลองไฟไนต์สเตตแมชชีน(Finite State Machine ,FSM) ที่อธิบายถึงสถานะของเครื่องมือการแปลงโดยสามารถอธิบายได้ดังนี้

1) สถานะเริ่มต้น : เมื่อเริ่มต้นระบบจะเริ่มที่สถานะนี้ หากมีการนำเข้าเนื้อหาของแบบจำลองในรูปแบบเอ็กซ์เอ็มแอล ของ UPPAAL เข้าสู่ระบบจะเปลี่ยนไปยังสถานะแก้ไข

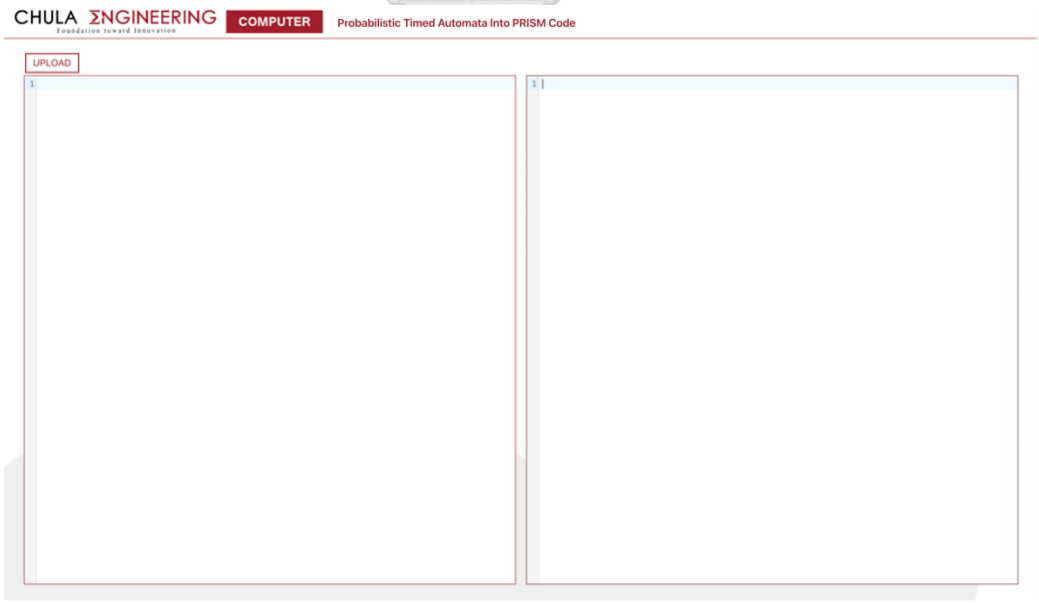
2) สถานะแก้ไข : ผู้ใช้งานสามารถแก้ไขเนื้อหาของแบบจำลองเอ็กซ์เอ็มแอล ได้ในสถานะนี้โดยเมื่อพร้อมแล้วสามารถแปลงแบบจำลองเอ็กซ์เอ็มแอล จะเปลี่ยนไปยังสถานะประมวลผล แต่ถ้ายังไม่พร้อมสามารถเริ่มใหม่กลับไปสู่สถานะเริ่มต้น

3) สถานะประมวลผล : เครื่องมือแปลงแบบจำลองจะเริ่มต้นประมวลผลแบบจำลองเอ็กซ์เอ็มแอล เพื่อสร้างเป็นรหัสปริซึม โดยถ้าแปลงเป็นรหัสปริซึมสำเร็จจะไปยังสถานะผลการแปลง แต่ถ้าพบข้อผิดพลาดจะไปยังสถานะข้อผิดพลาด

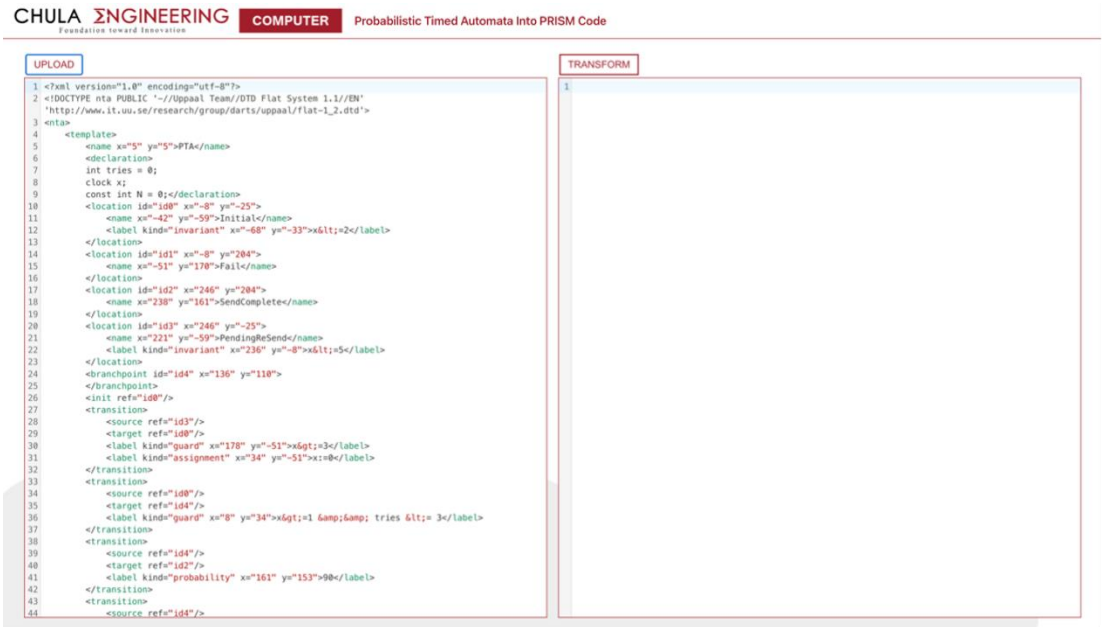
4) สถานะแสดงผลการแปลง : ระบบจะแสดงรหัสปริซึมจากการแปลงโดยผู้ใช้สามารถเริ่มใหม่จะกลับไปยังสถานะเริ่มต้น หรือแก้ไขเอ็กซ์เอ็มแอล จะไปยังสถานะแก้ไข

5) สถานะแสดงข้อผิดพลาด : ระบบจะแสดงข้อผิดพลาดที่เกิดขึ้นจากการแปลงหากมีการแก้ไขเอ็กซ์เอ็มแอล ไปยังสถานะแก้ไข หรือผู้ใช้จะเริ่มใหม่เพื่อกลับไปยังสถานะเริ่มต้น

หลังจากที่ได้กำหนดสถานะของระบบในการดำเนินประบวนการแปลงแล้ว ผู้วิจัยจะสร้างส่วนต่อประสานที่สอดคล้องกับสถานะของระบบที่ได้วางแผนไว้โดยในรูปที่ 5.2 ถึง 5.6 จะแสดงส่วนต่อประสานที่สอดคล้องกัน

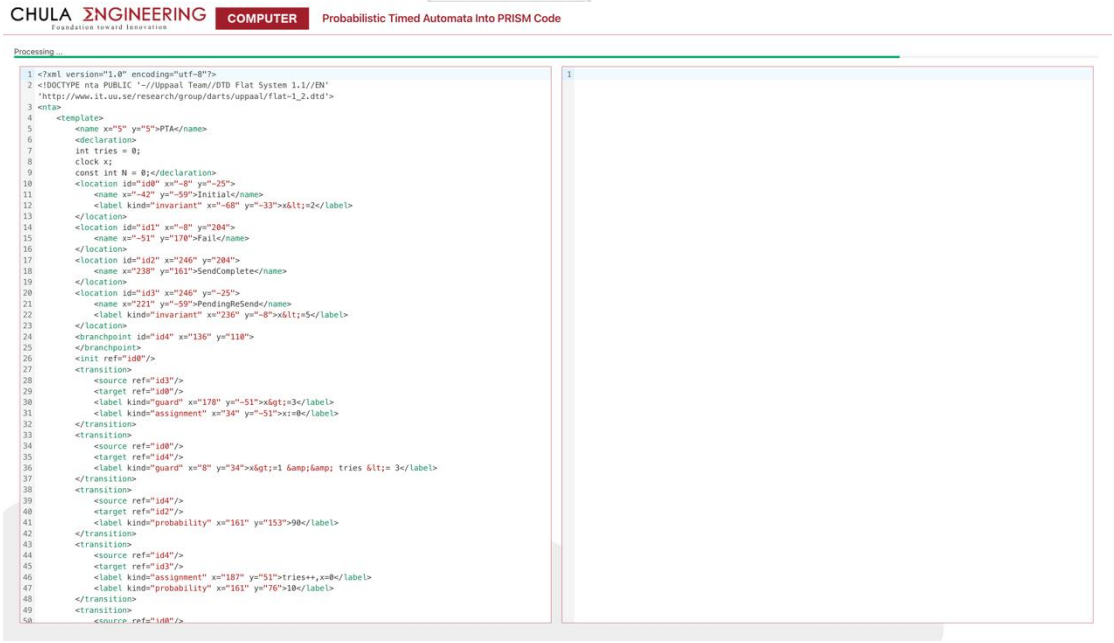


รูปที่ 5.2 ตัวอย่างหน้าจอในสถานะเริ่มต้นของเครื่องมือการแปลงแบบจำลองเป็นรหัสปริซึม



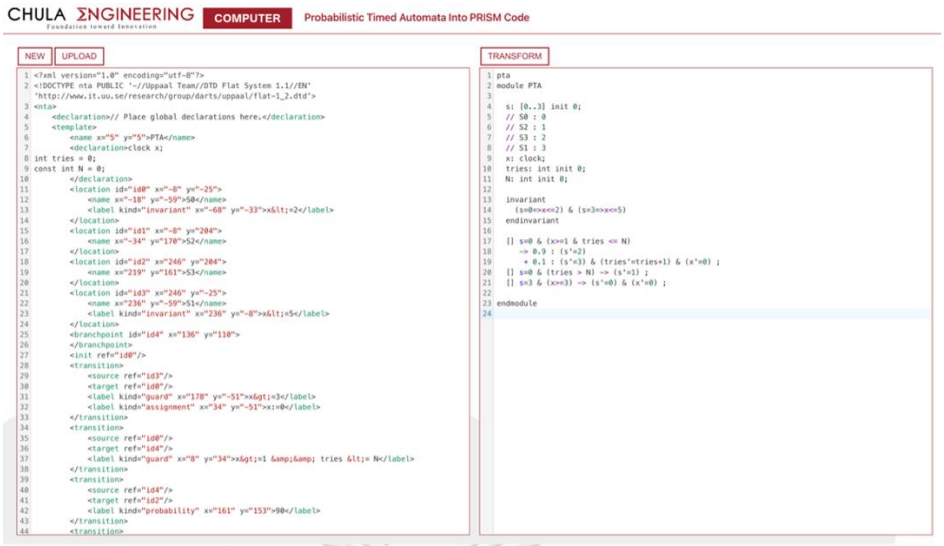
รูปที่ 5.3 ตัวอย่างหน้าจอในสถานะแก้ไขของเครื่องมือการแปลงแบบจำลองเป็นรหัสปริซึม

รูปที่ 5.3 แสดงให้เห็นถึงส่วนของตัวแก้ไขรหัสเอ็กซ์เอ็มแอล (XML Editor) ที่แสดงเนื้อหาของแบบจำลอง PTAs ในรูปแบบเอ็กซ์เอ็มแอล โดยผู้ใช้สามารถปรับแก้ไขได้และจะปรากฏปุ่ม TRANSFORM เพื่อเริ่มต้นกระบวนการแปลงแบบจำลองไปสู่รหัสปริซึม



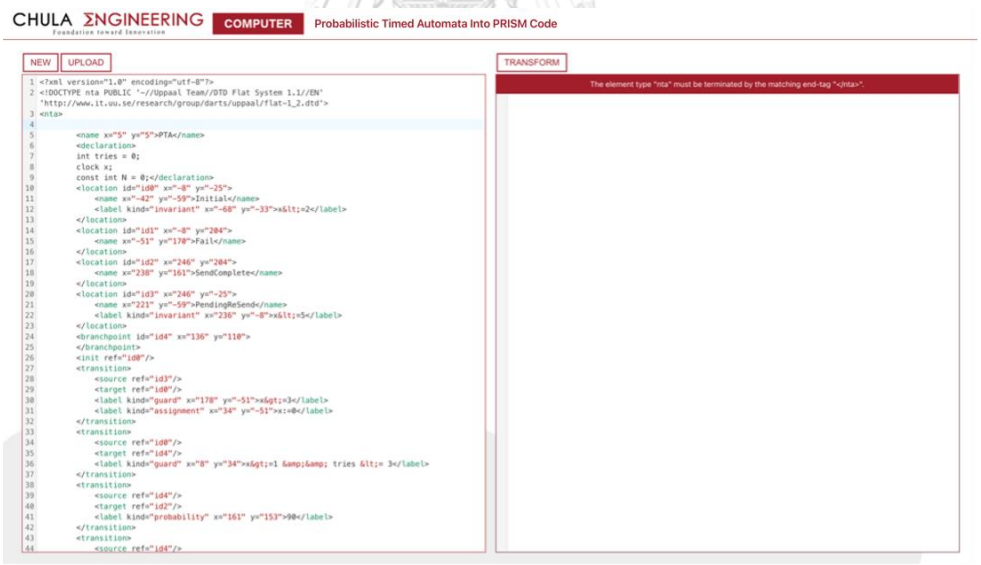
รูปที่ 5.4 ตัวอย่างหน้าจอในสถานะประมวลผลของเครื่องมือการแปลงแบบจำลองเป็นรหัสปริซึม

รูปที่ 5.4 แสดงให้เห็นถึงส่วนของแถบโหลดสีเขียวที่ไหลจากซ้ายไปขวาเมื่อเครื่องมือนำเนื้อหาของเอ็กซ์เอ็มแอล ไปประมวลผลผ่านส่วนต่อประสานงานนี้กับระบบหลังบ้าน (Backend)



รูปที่ 5.5 ตัวอย่างหน้าจอสถานะแสดงผลการแปลงของเครื่องมือการแปลงแบบจำลองเป็นรหัสปริซึม

รูปที่ 5.5 แสดงให้เห็นถึงผลลัพธ์ของการแปลงแบบจำลอง PTAs จากรูปแบบเอ็กซ์เอ็มแอล ไปสู่รหัสปริซึมที่สอดคล้องกัน โดยในสถานะนี้ระบบจะแสดงปุ่ม New เพื่อกลับไปเริ่มใหม่อีกครั้ง หรือผู้ใช้จะแก้ไขเอ็กซ์เอ็มแอล เพื่อแปลงแบบจำลองซ้ำอีกครั้งก็ได้เช่นกัน

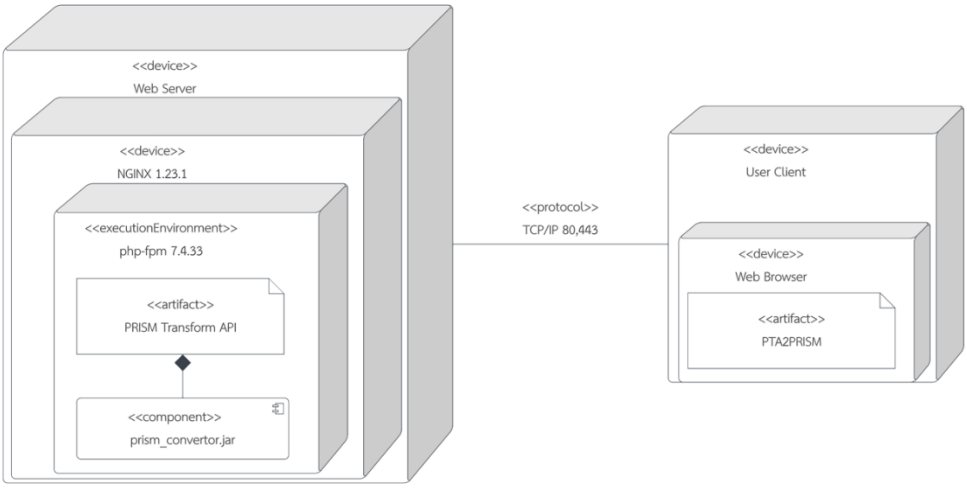


รูปที่ 5.6 ตัวอย่างหน้าจอสถานะแสดงข้อผิดพลาดของเครื่องมือการแปลงแบบจำลองเป็นรหัสปริซึม

รูปที่ 5.6 แสดงให้เห็นถึงข้อผิดพลาดของการแปลงแบบจำลอง PTAs ที่เกิดจากการประมวลผลแล้วพบข้อผิดพลาดเกิดขึ้นในขั้นตอนของกระบวนการแปลง ที่อาจเกิดจากประบวนการแปลงแบบจำลองไม่ถูกต้องตามกฎการแปลงที่ได้กำหนดไว้ โดยจะแสดงในแถบสีแดงที่แจ้งผู้ใช้งานให้เห็นข้อผิดพลาด

5.2.3 การติดตั้งและนำมาใช้

ผู้วิจัยได้นำเครื่องมือที่พัฒนาจัดทำในรูปแบบของเว็บแอปพลิเคชัน ให้สามารถติดตั้งและเข้าถึงได้ผ่านเว็บบราวเซอร์ นี้ทำให้เครื่องมือมีความสะดวกในการเข้าถึงและใช้งานได้จากที่ใดก็ตามที่มีการเชื่อมต่ออินเทอร์เน็ต จากรูปที่ 5.7 แผนภาพการปรับใช้แสดงให้เห็นถึงวิธีการติดตั้งเครื่องมือสนับสนุนการแปลงแบบจำลองเป็นรหัสปริซึม และการใช้เรียกใช้เครื่องมาผ่านอุปกรณ์ของผู้ใช้งาน



รูปที่ 5.7 ไดอะแกรมการติดตั้ง (Deployment Diagram) ของเครื่องมือสนับสนุนการแปลงแบบจำลองเป็นรหัสปริซึม

ไดอะแกรมการติดตั้งในรูปที่ 5.7 แสดงถึงโครงสร้างของเครื่องมือสนับสนุนการแปลงแบบจำลอง PTAs ไปเป็นรหัสปริซึม ในตารางที่ 5.1 จะอธิบายถึงองค์ประกอบต่างๆ ที่ปรากฏในแบบจำลอง

ตารางที่ 5.1 องค์ประกอบและคำอธิบายไดอะแกรมการติดตั้ง

องค์ประกอบ	คำอธิบาย
Web Server	อุปกรณ์ที่ทำหน้าที่เป็นเซิร์ฟเวอร์เว็บ มีหน้าที่ให้บริการเนื้อหาในเว็บไซต์และรับคำร้องขอจากไคลเอ็นต์
NGINX 1.23.1	ซอฟต์แวร์เซิร์ฟเวอร์ที่ทำงานบนเว็บเซิร์ฟเวอร์ มีหน้าที่จัดการการเชื่อมต่อและนำเสนอเนื้อหา
php-fpm 7.4.33	สภาพแวดล้อมการทำงานของ PHP ที่ใช้สำหรับการประมวลผลสคริปต์ PHP บนเซิร์ฟเวอร์
PRISM Transform API	อินเตอร์เฟส API ที่พัฒนาขึ้นเพื่อทำการแปลงแบบจำลอง PTAs ไปเป็นรหัสปริซึม

ตารางที่ 5.1 องค์ประกอบและคำอธิบายไดอะแกรมการติดตั้ง (ต่อ)

องค์ประกอบ	คำอธิบาย
prism_converter.jar	ไฟล์ JAR ที่ประกอบด้วยโปรแกรมหรือส่วนประกอบของโปรแกรมที่ทำการแปลงข้อมูล
User Client	อุปกรณ์ของผู้ใช้ที่ใช้เพื่อเข้าถึงเครื่องมือสนับสนุนการแปลง
Web Browser	โปรแกรมที่ใช้เพื่อเรียกดูเว็บเพจ ทำงานบนอุปกรณ์ของผู้ใช้
PTA2PRISM	เว็บแอปพลิเคชันที่ใช้เพื่อการแปลงแบบจำลอง PTAs
TCP/IP 80,443	โปรโตคอลสำหรับการสื่อสารผ่านเครือข่าย โดยทั่วไปหมายถึง HTTP และ HTTPS ซึ่งใช้สำหรับการถ่ายโอนข้อมูลเว็บ

ไดอะแกรมการติดตั้งนี้ช่วยให้เราเห็นภาพรวมของการจัดวางและการทำงานร่วมกันของส่วนต่างๆ ในระบบสนับสนุนการแปลง PTAs เป็นรหัสปริซึม แต่ละองค์ประกอบทำหน้าที่เฉพาะและทำงานร่วมกันเพื่อให้บริการการแปลงแบบจำลองได้อย่างมีประสิทธิภาพ

5.3 การปรับปรุงข้อจำกัดในแบบจำลองเพื่ออำนวยความสะดวกในการแปลง

การพัฒนาเครื่องมือสนับสนุนการแปลงไม่ได้ปราศจากความท้าทายและข้อจำกัดที่อาจเกิดขึ้นในระหว่างกระบวนการ ในหัวข้อนี้จะระบุและปรับปรุงข้อจำกัดที่พบในแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นเพื่อให้กระบวนการแปลงเป็นไปอย่างราบรื่นและมีประสิทธิภาพ โดยพบข้อจำกัดจากการแปลงดังนี้

1) ข้อจำกัดของการแยกวิเคราะห์(parse) โครงสร้างเอ็กซ์เอ็มแอล ที่ได้จาก UPPAAL โดยใช้จาวา พบว่าโครงสร้างในส่วนของการการประกาศมาร์กอัปที่มีอยู่หรือชี้ไปที่การประกาศประเภทเอกสาร มีรูปแบบที่ไม่ถูกต้อง อันเนื่องมาจากกระบวนการ parse ของจาวาได้กำหนดคุณลักษณะการตรวจสอบคำจำกัดความประเภทเอกสารภายนอก (external DTD) ให้เปิดใช้งานเป็นค่าเริ่มต้นเพื่อป้องกันไม่ให้การปรับแก้เอกสารเอ็กซ์เอ็มแอล ไม่เป็นไปตามข้อกำหนดของ UPPAAL

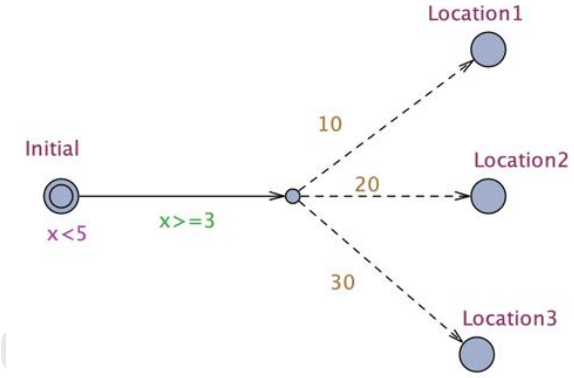
```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE nta PUBLIC '-//Uppaal Team//DTD Flat System 1.1//EN'
'http://www.it.uu.se/research/group/darts/uppaal/flat-1_2.dtd'>
<nta>
    <template>
        <name x="5" y="5">PTA</name>
        ...
        <branchpoint id="id4" x="136" y="110">
        </branchpoint>
        ...
</nta>
```

รูปที่ 5.8 ตัวอย่างของเอ็กซ์เอ็มแอลจาก UPPAAL ที่ใช้คำจำกัดความประเภทเอกสารจากภายนอก

รูปที่ 5.8 แสดงให้เห็นถึงตัวอย่างของโครงสร้างเอกสารเอ็กซ์เอ็มแอล ที่ได้จากเครื่องมือ UPPAAL พบว่ามีการกำหนดคำจำกัดความประเภทเอกสาร หรือ Document Type Definition(DTD) จากแหล่งอ้างอิงของ DTD นี้

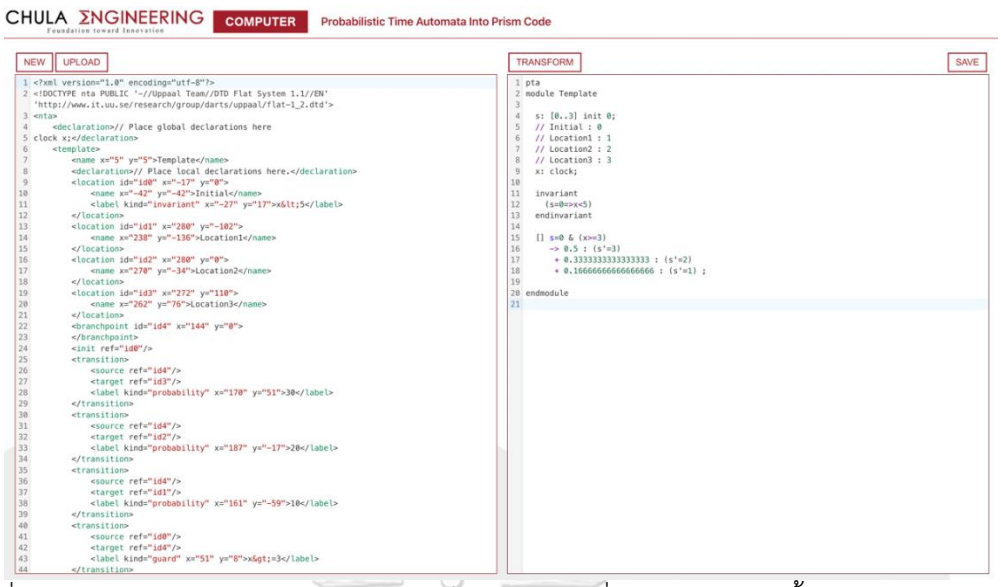
ผู้วิจัยพบว่าแม้เอกสารเอ็กซ์เอ็มแอล ดังกล่าวจะถูกนำออกจะเครื่องมือ UPPAAL แต่ด้วยรูปแบบมาร์กอัปประเภทจุดแยก (branchpoint) ไม่ได้กำหนดในคำจำกัดความประเภทเอกสารของ UPPAAL ทำให้กระบวนการแยกวิเคราะห์เอกสารเอ็กซ์เอ็มแอล จาก UPPAAL ไม่สามารถดำเนินการต่อได้จากคุณสมบัตินี้ ผู้วิจัยจึงเลือกปิดคุณลักษณะการตรวจสอบนี้เพื่อให้กระบวนการแยกวิเคราะห์สามารถดำเนินการต่อไปได้ในการเข้าถึงองค์ประกอบบของแบบจำลอง PTAs ในรูปแบบเอ็กซ์เอ็มแอล ต่อไป

2) ข้อจำกัดในการกำหนดค่าความน่าจะเป็นบนจุดแยกของเครื่องมือ UPPAAL นั้นที่ไม่มีรูปแบบเฉพาะสามารถกำหนดโดยไม่จำเป็นต้องอยู่ในรูปของเปอร์เซ็นต์ แต่เป็นค่าน้ำหนักที่กำหนดไว้บนเส้นทางย่อยดังแสดงในรูปที่ 5.9 ที่สถานะเริ่มต้นมีการเงื่อนไขของนาฬิกา เมื่อเข้าเงื่อนไขของตัวป้องกันจะเจอกับจุดแยกที่ให้ความน่าจะเป็นเป็นน้ำหนักในแต่ละเส้นทาง โดยไปที่ Location 1 ที่น้ำหนักเท่ากับ 10 ไปที่ Location 2 ที่น้ำหนักเท่ากับ 20 ไปที่ Location 3 ที่น้ำหนักเท่ากับ 30



รูปที่ 5.9 แบบจำลอง PTAs จาก UPPAAL ที่เส้นแยกกำหนดน้ำหนักของความน่าจะเป็น

แต่ในเครื่องมือแปลงแบบจำลองปริซึมนั้นจำเป็นต้องกำหนดค่าความน่าจะเป็นในรูปแบบของเปอร์เซ็นต์ โดยผลรวมของค่าความน่าจะเป็นจากจุดแยกไปยังสถานะต้องครบ 100% หรือค่า 1 ในกรณีนี้พบว่าการแปลงค่าความน่าจะเป็นจาก UPPAAL ไปเป็นความน่าจะเป็นของปริซึมจะต้องคำนวนใหม่เพื่อให้ได้ค่าความน่าจะเป็นในรูปแบบของเปอร์เซ็นต์ให้ครบ 100% หรือค่า 1 ดังนั้นผู้วิจัยจึงได้ปรับปรุงกระบวนการแปลงในส่วนของความน่าจะเป็นให้สมบูรณ์ยิ่งขึ้น โดยแปลงผลรวมของน้ำหนักเทียบบัญญัติไตรยางศ์ให้ได้ผลลัพธ์เท่ากับ 100%



รูปที่ 5.10 ผลการแปลงแบบจำลอง PTAs เป็นรหัสปริซึมที่เส้นแยกกำหนดน้ำหนักของความน่าจะเป็น

ตัวอย่างจากรูปที่ 5.9 พบว่าเส้นทางย่อยจากจุดแยกมีการกำหนดน้ำหนักไว้ดังนี้ คือ 10, 20 และ 30 เมื่อนำมาปรับให้ค่าใหม่ให้ผลรวมเท่ากับ 1 จะใช้การหาร้อยละในแต่ค่านำหนัก พบว่าน้ำหนักเท่ากับ 10 จะแปลงค่าได้ 0.16666666666666666 และน้ำหนักเท่ากับ 20 จะแปลงค่าได้ 0.3333333333333333 เนื่องจากการหารไม่ลงตัวจะได้ในภาษาจาวาจะกำหนดทศนิยมไว้ที่ 17 ตำแหน่งจากหางของทศนิยมที่มากกว่าหรือเท่ากับ 5 และทศนิยมไว้ที่ 16 ตำแหน่งจากหางของทศนิยมที่น้อยกว่า และสุดท้ายน้ำหนักเท่ากับ 30 จะแปลงค่าได้ 0.5 ดังแสดงผลลัทธ์ในรูปที่ 5.10 ที่การแปลงแทรนซิชันแบบมีความน่าจะเป็นแบบระบุน้ำหนักถูกแปลงเป็นคำสั่งในรูปแบบของรหัสปริซึม

3) การประกาศตัวแปรใน UPPAAL เป็นข้อจำกัดที่ผู้วิจัยพบว่ามีความสำคัญในการแปลงแบบจำลองไปยังรหัสปริซึม การเน้นย้ำถึงความจำเป็นของการมีรูปแบบการประกาศที่มีโครงสร้างเพื่อให้การแปลงสามารถเกิดขึ้นได้อย่างถูกต้องและมีประสิทธิภาพ ในรูปที่ 5.11 แสดงตัวอย่างของรูปแบบการประกาศตัวแปรที่หลากหลายใน UPPAAL เช่น การประกาศค่าคงที่ การประกาศอาร์เรย์ การประกาศโครงสร้างข้อมูล และการประกาศช่องทางการสื่อสาร เป็นต้น

```
// Place global declarations here
clock x;
const int a = 1;
bool b[8], c[4];
int[0,100] d=5;
int e[2][3] = { { 1, 2, 3 }, { 4, 5, 6} };
clock y, z;
chan f;
urgent chan g;
struct { int h; bool j; } s1 = { 2, true };
```

รูปที่ 5.11 ตัวอย่างของรูปแบบการประกาศตัวแปรที่หลากหลายใน UPPAAL

เพื่อปรับปรุงกระบวนการแปลงแบบจำลองอย่างมีประสิทธิภาพ ผู้วิจัยได้ปรับปรุงกลไกการแปลงโดยจำกัดรูปแบบการประกาศตัวแปรให้เฉพาะประเภท clock, int, และ bool เท่านั้น การประกาศแต่ละตัวแปรจะต้องปฏิบัติตามรูปแบบที่ชัดเจน มีการประกาศหนึ่งครั้งต่อหนึ่งตัวแปร และควรประกอบด้วยขั้นตอนต่อไปนี้

3.1) การกำหนดว่าเป็นค่าคงที่หรือไม่ (optional)

3.2) ระบุประเภทของตัวแปร (clock, int, หรือ bool)

3.3) ตั้งชื่อตัวแปร

3.4) สามารถกำหนดค่าเริ่มต้นให้กับตัวแปรได้ (optional)

ตัวอย่างของการประกาศตัวแปรที่ถูกต้องตามรูปแบบที่กำหนดไว้สำหรับใช้ในกระบวนการแปลงแบบจำลอง PTAs จาก UPPAAL ไปเป็นรหัสปริซึม คือ **`int count = 1;`** สำหรับการประกาศตัวแปรปกติที่มีการกำหนดค่าเริ่มต้น **`const int N = 1;`** สำหรับการประกาศค่าคงที่ และ **`clock x;`** สำหรับตัวแปรประเภทนาฬิกาที่ไม่มีการกำหนดค่าเริ่มต้น

การปรับปรุงนี้มีจุดมุ่งหมายเพื่อปรับให้การแปลงตัวแปรสอดคล้องกับระบบปริซึมเพื่อให้กระบวนการแปลงตัวแปรเหล่านี้เป็นรหัสปริซึมนั้นง่ายและปราศจากข้อผิดพลาด นอกจากนี้ยังช่วยให้สามารถตรวจสอบความถูกต้องของข้อมูลได้ง่ายขึ้น และป้องกันข้อผิดพลาดที่อาจเกิดจากการใช้รูปแบบการประกาศที่ไม่มีโครงสร้างหรือไม่สอดคล้องกัน

4) ข้อจำกัดในการกำหนดชื่อของการกระทำ โดยผู้วิจัยพบว่าแบบจำลองที่สร้างจากเครื่องมือ UPPAAL ที่ไม่สามารถกำหนดชื่อของการกระทำบนเส้นแทรนซิชันหลังจากวาดแบบจำลองและนำออกมาในรูปแบบของเอกสารเอ็กซ์เอ็มแอล ได้ ทำให้การกำหนดชื่อของการกระทำในการกำหนดคำสั่งของภาษาปริซึมไม่สามารถทำได้เช่นกัน ตัวอย่างเช่น การกระทำ send ที่มีตัวป้องกัน **`x < 2`** เพื่อเปลี่ยนสถานะจาก **`s=1`** เป็น **`s =2`** ในภาษาปริซึม สามารถเขียนได้คือ **`[send] s=1 & x <2 -> s′=2`** แต่เครื่องมือ UPPAAL ไม่สามารถกำหนดคำว่า send บนเส้นของแทรนซิชันได้ ดังนั้นผู้วิจัยจึงต้องแก้ไขการกำหนดชื่อของการกระทำเหลือไว้แค่สัญลักษณ์ [] เพื่อระบุคำสั่งของการกระทำ ดังนั้นการกำหนดคำสั่งดังกล่าวจะเหลือเพียง **`[] s=1 & x <2 -> s′=2`** ที่กำหนดการแปลี่ยนแปลงของตำแหน่ง

บทที่ 6

การทดสอบเครื่องมือสนับสนุนการแปลงแบบจำลองเป็นรหัสปริซึม

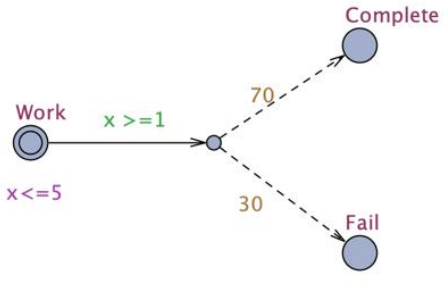
ในบทนี้จะเป็นการนำเสนอกระบวนการทดสอบที่ผู้วิจัยได้ดำเนินการเพื่อตรวจสอบความเที่ยงตรงและประสิทธิภาพของเครื่องมือสนับสนุนการแปลงแบบจำลอง PTAs ไปเป็นรหัสปริซึม การทดสอบมุ่งเน้นไปที่การตรวจสอบความถูกต้องของโค้ดที่สร้างขึ้น การตอบสนองของระบบต่อข้อมูลนำเข้าที่หลากหลาย และความสามารถในการจัดการกับเงื่อนไขด้านเวลาที่ซับซ้อน โดยขั้นตอนการทดสอบดังนี้

6.1 กำหนดกรอบการทดสอบ

ผู้วิจัยจะนำเสนอกรอบการทดสอบที่ออกแบบมาเพื่อตรวจสอบและยืนยันคุณภาพของเครื่องมือ กรอบการทดสอบนี้จะประกอบไปด้วยการนิยามเกณฑ์สำคัญที่เครื่องมือต้องผ่านเพื่อถือว่าเป็นการทดสอบที่สำเร็จ รวมถึงการกำหนดเป้าหมายการทดสอบที่ชัดเจน เพื่อให้แน่ใจว่าเครื่องมือทำงานได้ตามที่คาดหมายในทุกสถานการณ์ที่เป็นไปได้ โดยได้กำหนดกรณีทดสอบไว้ดังนี้

1) กรณีทดสอบความถูกต้องของการแปลง โดยการทดสอบนี้จะตรวจสอบว่าการแปลงแบบจำลองจาก PTAs ไปเป็นรหัสปริซึมเกิดขึ้นอย่างถูกต้องเป็นไปตามกฎการแปลง โดยได้กำหนดกรณีทดสอบความถูกต้องของการแปลงดังที่แสดงในตารางที่ 6.1 ถึง 6.3

ตารางที่ 6.1 กรณีทดสอบความถูกต้องของการแปลงจากแบบจำลองการทำงานของเครื่องจักร

ชื่อแบบจำลอง	แบบจำลองการทำงานของเครื่องจักร	
อธิบาย	กระบวนทำงานของเครื่องจักรที่มีความน่าจะเป็นในการสิ้นสุด	
ประกาศ (declaration)	`clock x;`	
แบบจำลอง		
ลำดับที่	**คำอธิบายกรณีทดสอบ**	**ผลลัพธ์ที่คาดหวัง**

1	แปลงตัวแปรสถานะนับจากจำนวนตำแหน่งบนแบบจำลอง และกำหนดสถานะเริ่มต้นของระบบจากลำดับตำแหน่งเริ่มต้น	`s: [0..2] init 0;`
		`// Work : 0`
		`// Fail : 1`
		`// Complete : 2`

ตารางที่ 6.1 กรณีทดสอบความถูกต้องของการแปลงจากแบบจำลองการทำงานของเครื่องจักร (ต่อ)

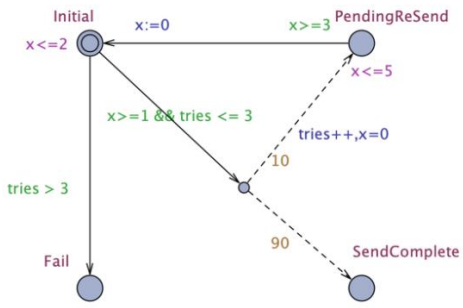
2	แปลงตัวแปรประเภท clock, int, และ bool ที่มีการประกาศหนึ่งครั้งต่อหนึ่งตัวแปร	`x: clock;`
3	การแปลงเงื่อนไขค่ายืนยงในแต่ละตำแหน่งบนแบบจำลอง	`Invariant (s=0=>x<=5) endinvariant`
4	แปลงเงื่อนไขสำหรับแทรนซิชันระหว่างสถานะที่มีค่าความน่าจะเป็น	`[] s=0 & (x >=1) -> 0.7 : (s'=1) + 0.3 : (s'=2);`

จากตารางที่ 6.1 กรณีทดสอบนี้จะทดสอบว่าระบบสามารถแปลงแบบจำลองการทำงานของเครื่องจักรมีการทำงานได้อย่างถูกต้องหรือไม่ โดยจะตรวจสอบสามส่วนหลัก: (1) การแปลงสถานะตามตำแหน่งในแบบจำลองและกำหนดสถานะเริ่มต้น (2) การแปลงตัวแปรประเภทต่างๆ ตามการประกาศที่กำหนด และ (3) การแปลงเงื่อนไขค่ายืนยงและเงื่อนไขความน่าจะเป็นสำหรับการเปลี่ยนสถานะ

ตารางที่ 6.2 กรณีทดสอบความถูกต้องของการแปลงจากแบบจำลองการส่งข้อความผ่านช่องทางที่ไม่น่าเชื่อถือ

ชื่อแบบจำลอง	แบบจำลองการส่งข้อความผ่านช่องทางที่ไม่น่าเชื่อถือ	
อธิบาย	กระบวนการส่งข้อความผ่านช่องทางที่ไม่น่าเชื่อถือ หรือที่เรียกว่า "Retransmission Protocol" ในระบบการสื่อสาร	

ประกาศ (declaration)	`int tries = 0;`
	`clock x;`
	`const int N = 0;`	

แบบจำลอง		
ลำดับที่	**คำอธิบายกรณีทดสอบ**	**ผลลัพธ์ที่คาดหวัง**

1	แปลงตัวแปรสถานะนับจากจำนวนตำแหน่งบนแบบจำลอง และกำหนดสถานะเริ่มต้นของระบบจากลำดับตำแหน่งเริ่มต้น	`s: [0..3] init 0;`
		`// Initial : 0`
		`// Fail : 1`
		`// SendComplete : 2`
		`// PendingReSend : 3`

ตารางที่ 6.2 กรณีทดสอบความถูกต้องของการแปลงจากแบบจำลองการส่งข้อความผ่านช่องทางที่ไม่น่าเชื่อถือ (ต่อ)

2	แปลงตัวแปรประเภท clock, int, และ bool ที่มีการประกาศหนึ่งครั้งต่อหนึ่งตัวแปร	`tries: int init 0;`
		`x: clock;`
		`N: int init 0;`

3	การแปลงเงื่อนไขค่ายืนยงในแต่ละตำแหน่งบนแบบจำลอง	`invariant`
		`(s=0=>x<=2) & (s=3=>x<=5)`
		`endinvariant`

4	แปลงเงื่อนไขสำหรับแทรนซิชันระหว่างสถานะที่ไม่มีค่าความน่าจะเป็น	`[] s=0 & (tries > 3) -> (s'=1);`
		`[] s=3 & (x>=3) -> (s'=0) &`
		`(x'=0);`

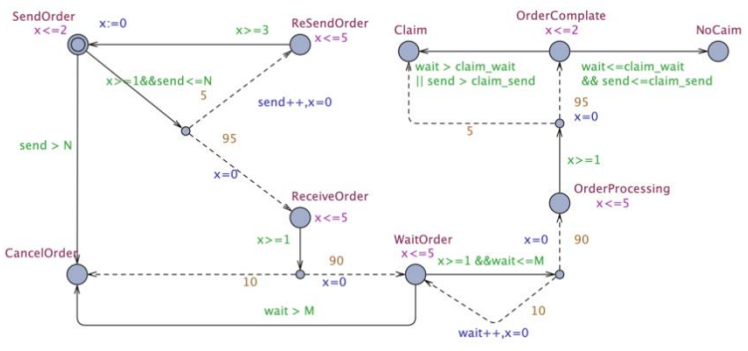
5	แปลงเงื่อนไขสำหรับแทรนซิชันระหว่างสถานะที่มีค่าความน่าจะเป็น	`[] s=0 & (x>=1 & tries <= 3)`
		`-> 0.9 : (s'=2)`
		`+ 0.1 : (s'=3) &`
		`(tries'=tries+1) & (x'=0);`

จากตารางที่ 6.2 กรณีทดสอบนี้จะทดสอบวิธีการแปลงแบบจำลองการสื่อสารผ่านช่องทางที่มีความไม่แน่นอน การทดสอบรวมถึงการแปลงสถานะและตัวแปรต่างๆ การจัดการกับเงื่อนไขค่ายืนยง และการแปลงเงื่อนไขที่มีความน่าจะเป็น การทดสอบมุ่งเน้นไปที่ความถูกต้องของการแปลงสถานะและตัวแปรต่างๆ ในแบบจำลอง รวมถึงการแปลงกฎการเปลี่ยนสถานะที่ซับซ้อนพร้อมกับความน่าจะเป็นที่กำหนดไว้

ตารางที่ 6.3 กรณีทดสอบความถูกต้องของการแปลงจากแบบจำลองระบบสั่งซื้ออาหาร

ชื่อแบบจำลอง	แบบจำลองระบบสั่งซื้ออาหาร
อธิบาย	กระบวนสั่งอาหารที่จำลองการสั่งบริการขนส่งอาหารที่มีการกำหนดความน่าจะเป็นของการร้องเรียนที่อาจเกิดขึ้น

ประกาศ (declaration)	`clock x;`
	`int send = 0;`
	`const int N = 0;`
	`int wait = 0;`
	`const int M = 0;`
	`int claim_wait = 1;`
	`int claim_send = 1;`

แบบจำลอง	

ตารางที่ 6.3 กรณีทดสอบความถูกต้องของการแปลงจากแบบจำลองระบบสั่งซื้ออาหาร (ต่อ)

ลำดับที่	คำอธิบายกรณีทดสอบ	ผลลัพธ์ที่คาดหวัง

1	แปลงตัวแปรสถานะนับจากจำนวนตำแหน่งบนแบบจำลอง และกำหนดสถานะเริ่มต้นของระบบจากลำดับตำแหน่งเริ่มต้น	`s: [0..8] init 0;`
		`// SendOrder : 0`
		`// ReceiveOrder : 1`
		`// ReSendOrder : 2`
		`// CancelOrder : 3`
		`// OrderProcessing : 4`
		`// Claim : 5`
		`// OrderComplate : 6`
		`// WaitOrder : 7`
		`// NoCaim : 8`

2	แปลงตัวแปรประเภท clock, int, และ bool ที่มีการประกาศหนึ่งครั้งต่อหนึ่งตัวแปร	`x: clock;`
		`send: int init 0;`
		`N: int init 0;`
		`wait: int init 0;`
		`M: int init 0;`
		`claim_wait: int init 1;`
		`claim_send: int init 1;`

3	การแปลงเงื่อนไขค่ายืนยงในแต่ละตำแหน่งบนแบบจำลอง	`invariant`
		`(s=0=>x<=2) & (s=1=>x<=5) &`
		`(s=2=>x<=5) & (s=4=>x<=5) &`
		`(s=6=>x<=2) & (s=7=>x<=5)`
		`endinvariant`

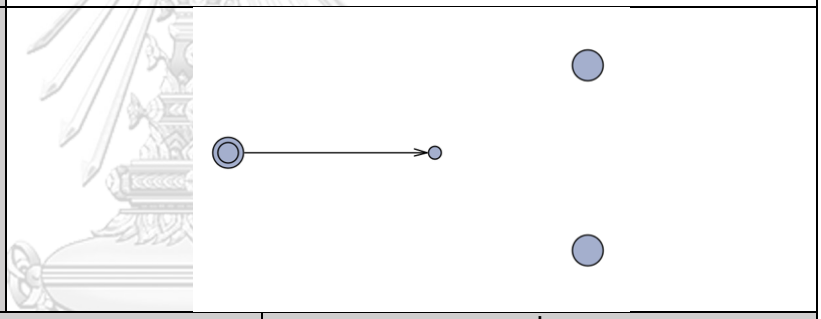
4	แปลงเงื่อนไขสำหรับแทรนซิชันระหว่างสถานะที่ไม่มีค่าความน่าจะเป็น	`[] s=0 & (send > N) -> (s'=3) ;`
		`[] s=2 & (x>=3) -> (s'=0) &`
		`(x'=0) ;`
		`[] s=6 & (wait<=claim_wait &`
		`send<=claim_send) -> (s'=8) ;`
		`[] s=6 & (wait > claim_wait |`
		`send > claim_send) -> (s'=5) ;`
		`[] s=7 & (wait > M) -> (s'=3) ;`

5	แปลงเงื่อนไขสำหรับแทรนซิชันระหว่างสถานะที่มีค่าความน่าจะเป็น	`[] s=0 & (x>=1&send<=N)`
		`-> 0.05 : (s'=2) &`
		`(send'=send+1) & (x'=0)`
		`+ 0.95 : (s'=1) & (x'=0) ;`
		`[] s=4 & (x>=1)`
		`-> 0.95 : (s'=6) & (x'=0)`
		`+ 0.05 : (s'=5) ;`
		`[] s=1 & (x>=1)`
		`-> 0.1 : (s'=3)`
		`+ 0.9 : (s'=7) & (x'=0) ;`
		`[] s=7 & (x>=1 &wait<=M)`
		`-> 0.1 : (s'=7) &`
		`(wait'=wait+1) & (x'=0)`
		`+ 0.9 : (s'=4) ;`

จากตารางที่ 6.3 กรณีทดสอบนี้จะทดสอบการแปลงกระบวนการสั่งซื้ออาหารในระบบที่มีความไม่แน่นอนและความน่าจะเป็นของเหตุการณ์ต่างๆ การทดสอบจะตรวจสอบว่าการแปลงสถานะตามแบบจำลองที่กำหนดไว้ในการประกาศค่าตัวแปรทั้งหมดถูกต้องหรือไม่ รวมถึงการแปลงเงื่อนไขและเงื่อนไขความน่าจะเป็นที่เกี่ยวข้องกับแทรนซิชันสถานะในกระบวนการสั่งอาหาร

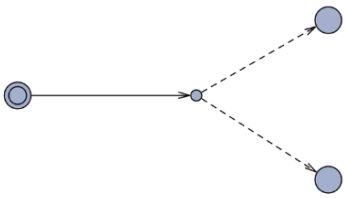
2) ทดสอบการจัดการข้อผิดพลาด ในการทดสอบนี้ จะมีการจำลองสถานการณ์ที่เกิดข้อผิดพลาดในข้อมูลนำเข้า เพื่อดูว่าระบบจัดการข้อผิดพลาดได้อย่างไร และจะมีการทดสอบว่าระบบสามารถแสดงข้อความแจ้งเตือนหรือนำไปสู่กระบวนการแก้ไขอย่างไร ดังที่แสดงในตารางที่ 6.4 ถึง 6.6

ตารางที่ 6.4 กรณีทดสอบความผิดพลาดแบบจำลองไม่ได้กำหนดเส้นแยกบนจุดแยก

ชื่อแบบจำลอง	แบบไม่จำลองไม่ได้กำหนดเส้นแยกบนจุดแยก	
อธิบาย	ผู้ใช้งานสร้างแบบจำลองแต่ไม่ได้กำหนดเส้นแยกบนจุดแยก	
ประกาศ (declaration)	-	
แบบจำลอง		
ลำดับที่	**คำอธิบายกรณีทดสอบ**	**ผลลัพธ์ที่คาดหวัง**
1	กรณีที่ไม่มีเส้นแยกบนจุดแยก จะแสดงข้อผิดพลาด	แสดงข้อผิดพลาด ไม่มีแทรนซิชันจากจุดแยก

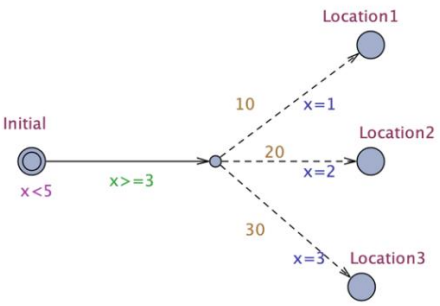
จากตารางที่ 6.4 กรณีทดสอบนี้จะมุ่งทดสอบการจัดการข้อผิดพลาดในแบบจำลองที่ไม่มีการกำหนดเส้นทางแยกที่จุดแยก จุดประสงค์คือเพื่อตรวจสอบว่าระบบสามารถตรวจจับและแสดงข้อผิดพลาดในกรณีที่มีการพยายามทำการเปลี่ยนแปลงสถานะจากจุดแยกโดยไม่มีการกำหนดทิศทางที่ชัดเจน คาดหวังว่าระบบจะไม่ดำเนินการใดๆ และจะแจ้งให้ผู้ใช้ทราบถึงข้อผิดพลาดที่เกิดขึ้น

ตารางที่ 6.5 กรณีทดสอบความผิดพลาดแบบจำลองไม่ได้กำหนดค่าความน่าจะเป็น

ชื่อแบบจำลอง	แบบจำลองไม่ได้กำหนดค่าความน่าจะเป็น	
อธิบาย	แบบจำลองไม่ได้กำหนดความน่าจะเป็นในแบบจำลอง	
ประกาศ (declaration)	-	
แบบจำลอง		
ลำดับที่	**คำอธิบายกรณีทดสอบ**	**ผลลัพธ์ที่คาดหวัง**
1	กรณีที่ไม่มีความน่าจะเป็นกำหนดไปตรงตำแหน่งของเส้นแยก จะแสดงข้อผิดพลาด	แสดงข้อผิดพลาด ไม่มีความน่าจะเป็นบนแทรนซิชันของจุดแยก

จากตารางที่ 6.5 กรณีทดสอบนี้จะทดสอบสถานการณ์ที่แบบจำลองไม่ได้กำหนดค่าความน่าจะเป็นในการเปลี่ยนสถานะ คาดหวังว่าระบบจะตรวจจับและรายงานข้อผิดพลาดเมื่อพบว่ามีการเปลี่ยนแปลงสถานะที่ไม่มีค่าความน่าจะเป็นที่กำหนดไว้ ซึ่งเป็นสิ่งจำเป็นสำหรับการทำงานของแบบจำลองที่มีเงื่อนไขทางเลือก

ตารางที่ 6.6 กรณีทดสอบความผิดพลาดแบบจำลองกำหนดตัวแปรซ้ำ

ชื่อแบบจำลอง	แบบจำลองกำหนดตัวแปรซ้ำ	
อธิบาย	แบบจำลองกำหนดตัวแปร x ซ้ำโดยใช้ทั้งรูปแบบนาฬิกาและจำนวนเต็ม	
ประกาศ (declaration)	`clock x;` `int x;`	
แบบจำลอง		
ลำดับที่	**คำอธิบายกรณีทดสอบ**	**ผลลัพธ์ที่คาดหวัง**
1	กรณีที่กำหนดตัวแปรซ้ำ จะแสดงข้อผิดพลาดในชื่อตัวแปรนั้น	แสดงข้อผิดพลาด ตัวแปรชื่อ ‘x’ ใช้ซ้ำ

จากตารางที่ 6.6 กรณีทดสอบนี้จะทดสอบสถานการณ์ที่ตัวแปรถูกกำหนดซ้ำในแบบจำลอง ซึ่งไม่ควรเกิดขึ้นในระบบการเขียนโปรแกรมที่ถูกต้อง ในกรณีนี้ ตัวแปร x ถูกกำหนดทั้งเป็นนาฬิกา (clock) และเป็นจำนวนเต็ม (int) สร้างความขัดแย้งในประเภทข้อมูล คาดหวังว่าระบบจะรายงานข้อผิดพลาดเนื่องจากการประกาศตัวแปรซ้ำนี้เป็นการละเมิดกฎของการประกาศตัวแปร

6.2 ทดสอบความถูกต้องของการแปลงรหัสปริซึม

หลังจากที่ผู้วิจัยได้กำหนดกรอบการทดสอบแล้ว ผู้วิจัยจะทำการทดสอบเพื่อยืนยันว่ารหัสปริซึมที่ได้จากการแปลงแบบจำลองต่างๆ สามารถสื่อสารได้อย่างถูกต้องและแม่นยำตามที่ได้กำหนดไว้ในสเปค จากกรณีทดสอบที่ได้ระบุไว้ โดยได้ผลของการทดสอบดังนี้

1) กรณีทดสอบความถูกต้องของการแปลงจากแบบจำลองการทำงานของเครื่องจักร

จากกรณีทดสอบความถูกต้องของการแปลงในตารางที่ 6.1 เมื่อนำออกแบบจำลองจากเครื่องมือ UPPAAL แล้วได้เอ็กซ์เอ็มแอล นำเข้าเครื่องมือแปลงแบบจำลอง ทำให้ได้ผลการทดสอบในตารางที่ 6.7

ตารางที่ 6.7 ผลการทดสอบความถูกต้องของการแปลงจากแบบจำลองการทำงานของเครื่องจักร

ลำดับที่	คำอธิบายกรณีทดสอบ	ผลลัพธ์ที่ได้จริง	สถานะการทดสอบ

1	แปลงตัวแปรสถานะนับจากจำนวนตำแหน่งบนแบบจำลอง และกำหนดสถานะเริ่มต้นของระบบจากลำดับตำแหน่งเริ่มต้น	`s: [0..2] init 0;`
		`// Work : 0`
		`// Fail : 1`
		`// Complete : 2`
			ผ่าน

2	แปลงตัวแปรประเภท clock, int, และ bool ที่มีการประกาศหนึ่งครั้งต่อหนึ่งตัวแปร	`x: clock;`	ผ่าน

3	การแปลงเงื่อนไขค่ายืนยงในแต่ละตำแหน่งบนแบบจำลอง	`Invariant`
		`(s=0=>x<=5)`
		`endinvariant`
			ผ่าน

4	แปลงเงื่อนไขสำหรับแทรนซิชันระหว่างสถานะที่มีค่าความน่าจะเป็น	`[] s=0 & (x >=1)`
		`-> 0.7 : (s'=1)`
		`+ 0.3 : (s'=2);`
			ผ่าน

2) กรณีทดสอบความถูกต้องของการแปลงจากแบบจำลองการส่งข้อความผ่านช่องทางที่ไม่น่าเชื่อถือ

จากกรณีทดสอบความถูกต้องของการแปลงในตารางที่ 6.2 เมื่อนำออกแบบจำลองจากเครื่องมือ UPPAAL แล้วได้เอ็กซ์เอ็มแอล เมื่อนำเข้าเครื่องมือแปลงแบบจำลอง ทำให้ได้ผลการทดสอบในตารางที่ 6.8

ตารางที่ 6.8 ผลการทดสอบความถูกต้องของการแปลงจากแบบจำลองการส่งข้อความผ่านช่องทางที่ไม่น่าเชื่อถือ

ลำดับที่	คำอธิบายกรณีทดสอบ	ผลลัพธ์ที่ได้จริง	สถานะการทดสอบ

1	แปลงตัวแปรสถานะนับจากจำนวนตำแหน่งบนแบบจำลอง และกำหนดสถานะเริ่มต้นของระบบจากลำดับตำแหน่งเริ่มต้น	`s: [0..3] init 0;`
		`// Initial : 0`
		`// Fail : 1`
		`// SendComplete : 2`
		`// PendingReSend : 3`
			ผ่าน

2	แปลงตัวแปรประเภท clock, int, และ bool ที่มีการประกาศหนึ่งครั้งต่อหนึ่งตัวแปร	`tries: int init 0;`
		`x: clock;`
		`N: int init 0;`
			ผ่าน

3	การแปลงเงื่อนไขค่ายืนยงในแต่ละตำแหน่งบนแบบจำลอง	`invariant`
		`(s=0=>x<=2) &`
		`(s=3=>x<=5)`
		`Endinvariant`
			ผ่าน

4	แปลงเงื่อนไขสำหรับแทรนซิชันระหว่างสถานะที่ไม่มีค่าความน่าจะเป็น	`[] s=0 & (tries > 3) ->`
		`(s'=1);`
		`[] s=3 & (x>=3) ->`
		`(s'=0) & (x'=0);`
			ผ่าน

5	แปลงเงื่อนไขสำหรับแทรนซิชันระหว่างสถานะที่มีค่าความน่าจะเป็น	`[] s=0 & (x>=1 & tries <= 3)`
		`-> 0.9 : (s'=2)`
		`+ 0.1 : (s'=3) &`
		`(tries'=tries+1) &`
		`(x'=0);`
			ผ่าน

3) กรณีทดสอบความถูกต้องของการแปลงจากแบบจำลองระบบสั่งซื้ออาหาร

จากกรณีทดสอบความถูกต้องของการแปลงในตารางที่ 6.3 เมื่อนำออกแบบจำลองจากเครื่องมือ UPPAAL แล้วได้เอ็กซ์เอ็มแอล และนำเข้าเครื่องมือแปลงแบบจำลอง ทำให้ได้ผลการทดสอบในตารางที่ 6.9

ตารางที่ 6.9 ผลการทดสอบความถูกต้องของการแปลงจากแบบจำลองระบบสั่งซื้ออาหาร

ลำดับที่	คำอธิบายกรณีทดสอบ	ผลลัพธ์ที่ได้จริง	สถานะการทดสอบ

1	แปลงตัวแปรสถานะนับจากจำนวนตำแหน่งบนแบบจำลอง และกำหนดสถานะเริ่มต้นของระบบจากลำดับตำแหน่งเริ่มต้น	`s: [0..8] init 0;`
		`// SendOrder : 0`
		`// ReceiveOrder : 1`
		`// ReSendOrder : 2`
		`// CancelOrder : 3`
		`// OrderProcessing : 4`
		`// Claim : 5`
		`// OrderComplate : 6`
		`// WaitOrder : 7`
		`// NoCaim : 8`
			ผ่าน

2	แปลงตัวแปรประเภท clock, int, และ bool ที่มีการประกาศหนึ่งครั้งต่อหนึ่งตัวแปร	`x: clock;`
		`send: int init 0;`
		`N: int init 0;`
		`wait: int init 0;`
		`M: int init 0;`
		`claim_wait: int init 1;`
		`claim_send: int init 1;`
			ผ่าน

3	การแปลงเงื่อนไขค่ายืนยงในแต่ละตำแหน่งบนแบบจำลอง	`invariant`
		`(s=0=>x<=2) & (s=1=>x<=5) & (s=2=>x<=5) & (s=4=>x<=5) & (s=6=>x<=2) & (s=7=>x<=5)`
		`endinvariant`
			ผ่าน

4	แปลงเงื่อนไขสำหรับแทรนซิชันระหว่างสถานะที่ไม่มีค่าความน่าจะเป็น	`[] s=0 & (send > N) -> (s'=3) ;`
		`[] s=2 & (x>=3) -> (s'=0) & (x'=0) ;`
		`[] s=6 & (wait<=claim_wait & send<=claim_send) -> (s'=8) ;`
		`[] s=6 & (wait > claim_wait | send > claim_send) -> (s'=5) ;`
		`[] s=7 & (wait > M) -> (s'=3) ;`
			ผ่าน

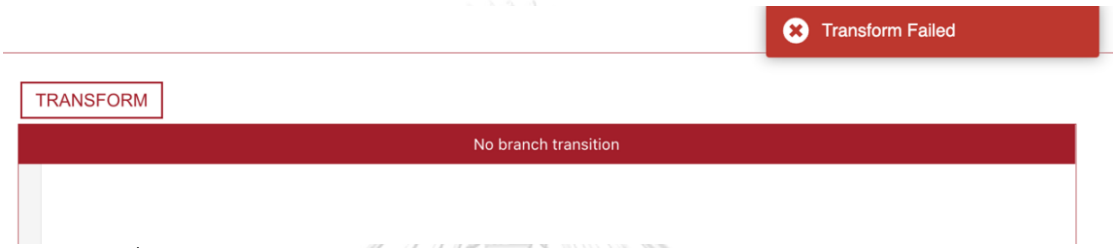
5	แปลงเงื่อนไขสำหรับแทรนซิชันระหว่างสถานะที่มีค่าความน่าจะเป็น	`[] s=0 & (x>=1&send<=N)`
		`-> 0.05 : (s'=2) & (send'=send+1) & (x'=0)`
		`+ 0.95 : (s'=1) & (x'=0)`
		`;`
		`[] s=4 & (x>=1)`
		`-> 0.95 : (s'=6) & (x'=0)`
		`+ 0.05 : (s'=5) ;`
		`[] s=1 & (x>=1)`
		`-> 0.1 : (s'=3)`
		`+ 0.9 : (s'=7) & (x'=0)`
		`;`
		`[] s=7 & (x>=1 &wait<=M)`
		`-> 0.1 : (s'=7) & (wait'=wait+1) & (x'=0)`
		`+ 0.9 : (s'=4) ;`
			ผ่าน

6.3 ทดสอบการจัดการข้อผิดพลาด

ผู้วิจัยได้มุ่งเน้นที่การตรวจจับและจัดการกับข้อผิดพลาดที่อาจเกิดขึ้นจากการแปลง เพื่อให้แน่ใจว่าเครื่องมือสามารถปฏิบัติงานได้อย่างเสถียรและน่าเชื่อถือ จากกรณีทดสอบที่ได้กำหนดกรอบการทดสอบเป็นส่วนสำคัญที่ช่วยยกระดับคุณภาพและความเชื่อมั่นในเครื่องมือ

1) กรณีทดสอบความผิดพลาดแบบจำลองไม่ได้กำหนดเส้นแยกบนจุดแยก

จากกรณีทดสอบการจัดการข้อผิดพลาดในตารางที่ 6.4 เมื่อนำการออกแบบจำลองจากเครื่องมือ UPPAAL แล้วได้เอ็กซ์เอ็มแอล เมื่อนำเข้าเครื่องมือแปลงแบบจำลองจะแสดงผลลัพธ์ดังรูปที่ 6.1 และแสดงผลการทดสอบในตารางที่ 6.10



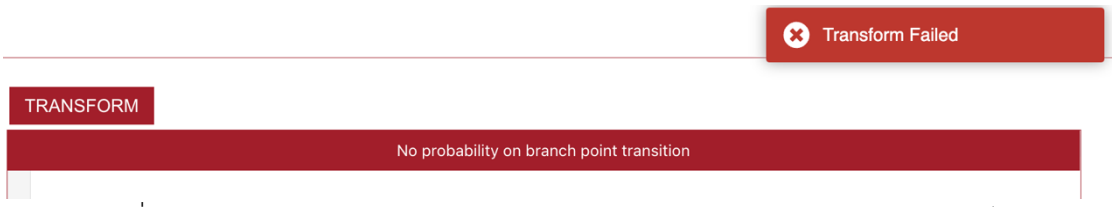
รูปที่ 6.1 การแจ้งเตือนความผิดพลาดของแบบจำลองไม่ได้กำหนดเส้นแยกบนจุดแยก

ตารางที่ 6.10 ผลการทดสอบความผิดพลาดการแปลงแบบจำลองไม่ได้กำหนดเส้นแยกบนจุดแยก

ลำดับที่	คำอธิบายกรณีทดสอบ	ผลลัพธ์ที่ได้จริง	สถานะการทดสอบ
1	กรณีที่ไม่มีเส้นแยกบนจุดแยก จะแสดงข้อผิดพลาด	แสดงข้อผิดพลาด ไม่ระบุแทรนซิชันจากจุดแยก	ผ่าน

2) กรณีทดสอบความผิดพลาดแบบจำลองไม่ได้กำหนดค่าความน่าจะเป็น

จากกรณีทดสอบการจัดการข้อผิดพลาดในตารางที่ 6.5 เมื่อนำการออกแบบจำลองจากเครื่องมือ UPPAAL แล้วได้เอ็กซ์เอ็มแอล เมื่อนำเข้าเครื่องมือแปลงแบบจำลองจะแสดงผลลัพธ์ดังรูปที่ 6.2 และแสดงผลการทดสอบในตารางที่ 6.11



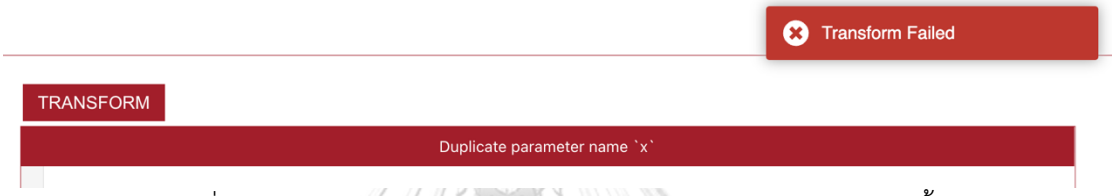
รูปที่ 6.2 การแจ้งเตือนความผิดพลาดของแบบจำลองไม่ได้กำหนดค่าความน่าจะเป็น

ตารางที่ 6.11 ผลการทดสอบความผิดพลาดการแปลงแบบจำลองไม่ได้กำหนดค่าความน่าจะเป็น

ลำดับที่	คำอธิบายกรณีทดสอบ	ผลลัพธ์ที่ได้จริง	สถานะการทดสอบ
1	กรณีที่ไม่มีความน่าจะเป็นกำหนดไปตรงตำแหน่งของเส้นแยก จะแสดงข้อผิดพลาด	แสดงข้อผิดพลาด ไม่มีความน่าจะเป็นบนแทรนซิชันของจุดแยก	ผ่าน

3) กรณีทดสอบความผิดพลาดแบบจำลองกำหนดตัวแปรซ้ำ

จากกรณีทดสอบการจัดการข้อผิดพลาดในตารางที่ 6.6 เมื่อนำการออกแบบจำลองจากเครื่องมือ UPPAAL แล้วได้เอ็กซ์เอ็มแอล เมื่อนำเข้าเครื่องมือแปลงแบบจำลองจะแสดงผลลัพธ์ดังรูปที่ 6.3 และแสดงผลการทดสอบในตารางที่ 6.12



รูปที่ 6.3 การแจ้งเตือนความผิดพลาดของแบบจำลองกำหนดตัวแปรซ้ำ

ตารางที่ 6.12 ผลการทดสอบความผิดพลาดการแปลงแบบจำลองกำหนดตัวแปรซ้ำ

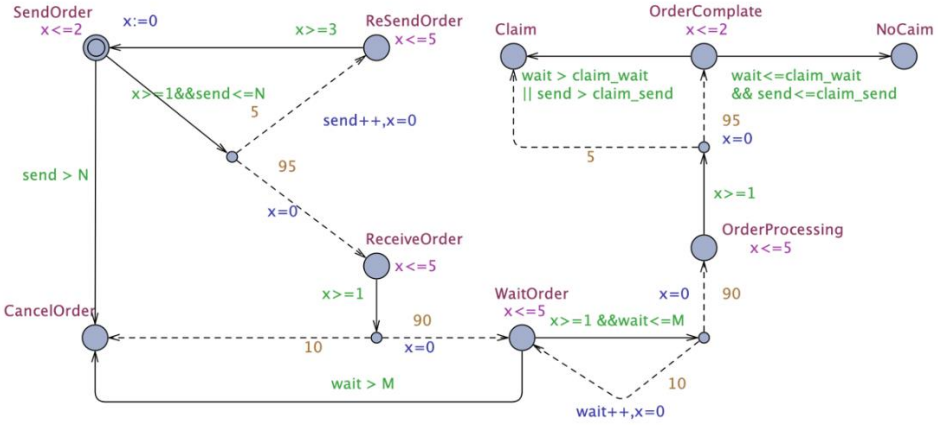
ลำดับที่	คำอธิบายกรณีทดสอบ	ผลลัพธ์ที่ได้จริง	สถานะการทดสอบ
1	กรณีที่กำหนดตัวแปรซ้ำ จะแสดงข้อผิดพลาดในชื่อตัวแปรนั้น	แสดงข้อผิดพลาด ตัวแปรชื่อ ‘x’ ใช้ซ้ำ	ผ่าน

6.4 ทดสอบจากกรณีศึกษา

ในหัวข้อนี้จะทดสอบเครื่องมือสนับสนุนการแปลงแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึมและนำผลลัพธ์ที่ได้มาวิเคราะห์ในเครื่องมือตรวจสอบแบบจำลองปริซึมกับกรณีทดสอบ 2 กรณี เพื่อตรวจสอบความถูกต้องและครบถ้วนตามที่ได้ระบุไว้ในขอบเขตของงานวิจัยนี้

1) กรณีศึกษาระบบสั่งซื้ออาหาร

จากรูปที่ 6.4 ผู้วิจัยได้จำลองระบบสั่งอาหารออนไลน์ที่เต็มไปด้วยการตัดสินใจและเงื่อนไขต่างๆ แสดงให้เห็นถึงการทำงานของกระบวนการนี้ผ่านแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นจากจุดเริ่มต้นที่ลูกค้าส่งคำสั่งซื้อ ระบบจะติดตามเวลาที่ใช้ในแต่ละขั้นตอน ตั้งแต่การรับคำสั่ง การประมวลผล การรอคำสั่ง จนถึงการจัดส่ง รวมถึงการจัดการกับเหตุการณ์ที่ไม่คาดคิดเช่นการยกเลิกหรือการร้องเรียน อธิบายแต่ละสถานะจะแสดงในตารางที่ 6.13



รูปที่ 6.4 แบบจำลองไทมด์ออโตมาตาความน่าจะเป็นของระบบสั่งอาหาร

ตารางที่ 6.13 คำอธิบายสถานะในแต่ละตำแหน่งของแบบจำลองระบบสั่งอาหาร

ตำแหน่ง	คำอธิบาย
SendOrder (ส่งคำสั่งซื้อ)	สถานะเริ่มต้นเมื่อลูกค้าส่งคำสั่ง มีเงื่อนไขที่จำกัดเวลาในสถานะนี้ไม่เกิน 2 หน่วยเวลา โดยมีความน่าจะเป็น 95% ที่คำสั่งซื้อจะถูกรับ นาฬิกา x จะถูกตั้งค่าใหม่เป็น 0 และ 5% ที่คำสั่งซื้อจะส่งซ้ำไปใหม่ ถ้าจำนวนครั้งที่พยายามส่ง (send) เกิน N ครั้ง ระบบจะเปลี่ยนไปยังสถานะ CancelOrder (ยกเลิกคำสั่งซื้อ)
ReceiveOrder (รับคำสั่งซื้อ)	สถานะนี้จะเกิดขึ้นหลังจากที่คำสั่งซื้อได้รับการยืนยัน จะมีเงื่อนไขว่าต้องทำการเปลี่ยนสถานะก่อนที่เวลาจะเกิน 5 หน่วยเวลา โดยมีความน่าจะเป็น 10% ที่คำสั่งซื้อจะถูกยกเลิกจากผู้รับหรือผู้ซื้อก่อน และมีความน่าจะเป็น 90% ที่จะรอคำสั่งซื้อดำเนินการ
ReSendOrder (ส่งคำสั่งซื้อซ้ำ)	ถ้าคำสั่งซื้อไม่ได้รับการยืนยันจะมีการพยายามส่งคำสั่งซื้อซ้ำ นาฬิกา x จะถูกตั้งค่าเป็น 0 ในทุกครั้งที่เข้าสู่สถานะนี้และรอเวลาไม่เกิน 5 หน่วยเวลาเพื่อส่งใหม่
CancelOrder (ยกเลิกคำสั่งซื้อ)	ถ้าคำสั่งซื้อไม่สามารถดำเนินการได้สำเร็จ สถานะนี้จะถูกเรียกใช้เพื่อยกเลิกคำสั่ง
WaitOrder (รอคำสั่งซื้อ)	สถานะนี้จะเกิดขึ้นระหว่างการรอคำสั่งซื้อดำเนินการ มีการกำหนดเงื่อนไขเวลาสำหรับระยะเวลารอไม่เกิน 5 หน่วย มีโอกาส 10% ที่คำสั่งซื้อยังไม่ดำเนินการและต้องรอต่อไป และมีโอกาส 90% ที่คำสั่งซื้อจะดำเนินการ และถ้าความพยายามในการรอเกิน M ครั้ง ระบบจะเปลี่ยนไปยังสถานะ CancelOrder (ยกเลิกคำสั่งซื้อ)

ตารางที่ 6.13 คำอธิบายสถานะในแต่ละตำแหน่งของแบบจำลองระบบสั่งอาหาร (ต่อ)

ตำแหน่ง	คำอธิบาย
OrderProcessing (ดำเนินการตามคำสั่งซื้อ)	สถานะที่คำสั่งซื้อกำลังดำเนินการมีการกำหนดเงื่อนไขเวลาสำหรับระยะเวลารอไม่เกิน 5 หน่วย โดยเมื่อดำเนินการเสร็จแล้วมีโอกาส 5% ที่จะได้ข้อร้องเรียนจากผู้ซื้อ และคำสั่งซื้อไม่สำเร็จ
OrderComplete (คำสั่งซื้อเสร็จสิ้น)	สถานะที่คำสั่งซื้อกำลังดำเนินการเสร็จแล้ว โดยจะจำลองวิเคราะห์จากจำนวนการรอหรือการส่งคำสั่งซื้อซ้ำเพื่อประเมินการร้องเรียนของผู้ใช้
NoClaim (ไม่มีการเรียกร้อง)	สถานะสุดท้ายหลังจากที่คำสั่งซื้อได้รับการดำเนินการเสร็จสิ้นและไม่มีการเรียกร้องใดๆ เกิดขึ้น.
Claim (การร้องเรียน)	สถานะที่กำหนดถึงการร้องเรียนของผู้ใช้

เพื่อวิเคราะห์หาโอกาสที่จะเกิดข้อร้องเรียนจากผู้ใช้ โดยประเมินจากการรอหรือการส่งคำสั่งซื้อซ้ำ ผู้วิจัยได้กำหนดตัวแปรสำหรับการตั้งค่าไว้ 4 ตัวแปรดังนี้

1.1) N คือ จำนวนครั้งที่ยอมให้มีการส่งคำสั่งซื้อใหม่ โดยถ้าส่งเกินกว่าจำนวนนี้จะเปลี่ยนไปสถานะยกเลิกคำสั่งซื้อ

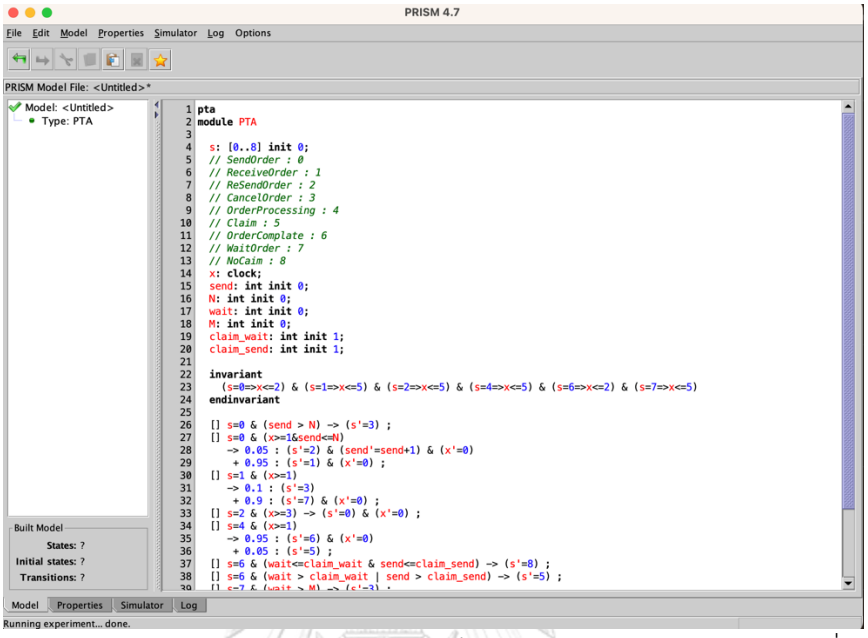
1.2) M คือ จำนวนครั้งที่ยอมให้มีการรอการดำเนินการต่อคำสั่งซื้อ โดยถ้ารอเกินกว่าจำนวนนี้จะเปลี่ยนไปสถานะยกเลิกคำสั่งซื้อ

1.3) claim_send คือ จำนวนของความอดทนที่มีต่อการส่งคำสั่งซื้อใหม่ ที่อาจเกิดจากคำสั่งซื้อถูกส่งไปแล้วแต่ไม่มีผู้รับคำสั่งซื้อนี้ ตัวแปรนี้จะกำหนดให้ทุกๆการส่งใหม่มีผลต่อการแจ้งข้อร้องเรียน

1.4) claim_wait คือ จำนวนของความอดทนที่มีต่อการรอคำสั่งซื้อดำเนินการ ที่อาจเกิดขึ้นจากการที่คำสั่งอาหารถึงร้านค้าแล้วแต่ไม่ได้ดำเนินการใด เพราะลูกค้าเยอะ หรือด้วยเหตุผลอื่นๆ ตัวแปรนี้จะกำหนดความอดทนของลูกค้าต่อการรอที่ส่งผลต่อการแจ้งข้อร้องเรียน

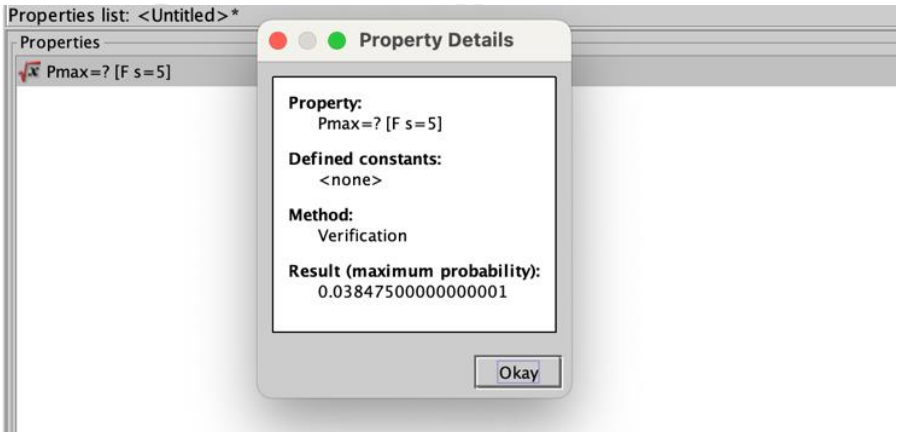
เมื่อต้องการวิเคราะห์หาความน่าจะเป็นบนแบบจำลองนี้ผู้วิจัยได้ส่งออกแบบจากเครื่องมือ UPPAAL และนำเข้าเครื่องมือแปลงแบบจำลอง PTAs ไปเป็นรหัสปริซึมโดยตัวอย่างของรหัสปริซึมการแปลงแบบจำลองระบบการสั่งซื้ออาหาร เพื่อนำมาวิเคราะห์รหัสปริซึมที่สร้างขึ้น ผู้วิจัยสามารถทำนายและเตรียมการสำหรับการร้องเรียนที่อาจเกิดขึ้นได้ ช่วยให้สามารถปรับปรุงคุณภาพการบริการและลดโอกาสของปัญหาที่ลูกค้าอาจพบเจอ

ในเครื่องมือตรวจสอบแบบจำลองปริซึมผู้วิจัยได้นำรหัสปริซึมที่ได้จากแบบจำลองเข้าสู่ระบบเพื่อใช้วิเคราะห์หาความน่าจะเป็นของการร้องเรียนที่เกิดขึ้นแสดงได้ในรูปที่ 6.5 ที่แสดงตัวอย่างของรหัสปริซึมที่ได้บนเครื่องมือวิเคราะห์แบบจำลอง



รูปที่ 6.5 แบบจำลองไทม์ออโตมาตาความน่าจะเป็นในรูปแบบรหัสปริซึมของระบบสั่งอาหารบน

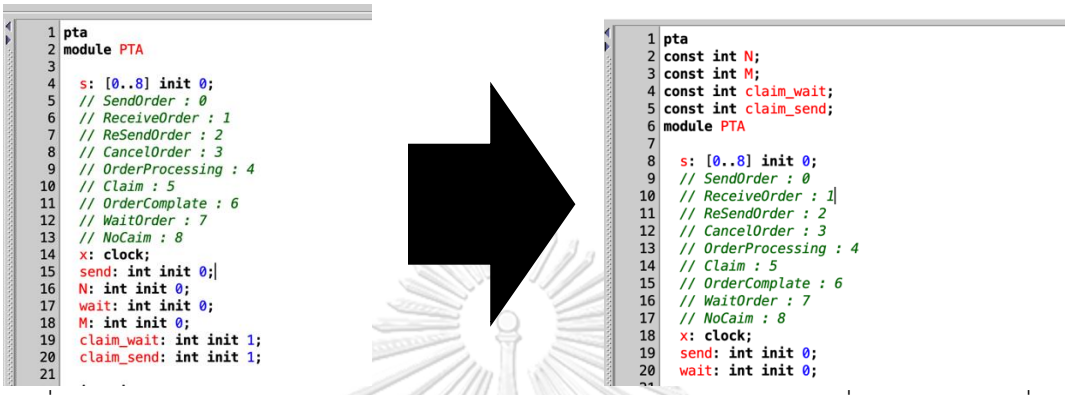
โดยรหัสปริซึมที่ได้นี้สามารถนำมาใช้ในการวิเคราะห์แบบจำลองได้ในรูปแบบการตรวจสอบความน่าจะเป็นสูงสุด ดังแสดงตัวอย่างของผลของการตรวจสอบในรูปที่ 6.6 ที่แสดงความน่าจะเป็นสูงที่สุดที่ผู้ใช้จะร้องเรียนถ้าไม่มีการส่งใหม่หรือการรอเกิดขึ้น แต่อาจเกิดจากการร้องเรียนหลังจากคำสั่งซื้อถูกดำเนินการ จะอยู่ที่ 3.84%



รูปที่ 6.6 ผลการตรวจสอบความน่าจะเป็นของแบบจำลองระบบสั่งอาหารบนที่อาจเกิดการร้องเรียน

เมื่อนำมาใช้กับเครื่องมือตรวจสอบแบบจำลองปริซึมพบว่าตัวแปรที่ได้กำหนดไว้ใน UPPAAL เมื่อแปลงมาเป็นรหัสปริซึมแล้วค่าของตัวแปรจะเท่ากับค่าที่กำหนดไว้ในเครื่องมือ UPPAAL โดย

ในทางเทคนิคแล้วเครื่องมือตรวจสอบแบบจำลองปริซึมสามารถสร้างผลลัพธ์ของการทดลองได้ แต่ต้องเปลี่ยนรูปแบบการกำหนดตัวแปรเป็นรูปแบบใหม่เพื่อตอบสนองต่อความต้องการของปริซึมในรูปที่ 6.7 จะเป็นการปรับปรุงการกำหนดตัวแปรใหม่โดยใช้การกำหนดตัวแปรค่าคงที่แบบไม่ระบุค่าเริ่มต้น



รูปที่ 6.7 การปรับปรุงการกำหนดตัวแปรใหม่โดยใช้การกำหนดตัวแปรค่าคงที่แบบไม่ระบุค่าเริ่มต้น

หลังจากที่ปรับปรุงการกำหนดตัวแปรแบบใหม่ สามารถนำตัวแปรที่กำหนดไว้มาใช้ในกระบวนการทดลองของเครื่องมือตรวจสอบแบบจำลองปริซึมโดยสรุปผลตรวจสอบแบบจำลองได้ในตารางที่ 6.14

ตารางที่ 6.14 ตารางแสดงผลการตรวจสอบแบบจำลองระบบสั่งอาหารโดยตรวจสอบความน่าจะเป็นสูงสุด

คุณสมบัติ	คำอธิบาย	claim_wait	claim_send	ความน่าจะเป็นสูงสุด
Pmax=? [F s=5]	หาความน่าจะเป็นสูงสุดที่จะเกิดการร้องเรียนโดยกำหนดให้ N = 0 M = 0	0	0	3.84%
		0	1	3.84%
		2	2	3.84%
		1	0	3.84%
		3	3	3.84%
Pmax=? [F s=5]	หาความน่าจะเป็นสูงสุดที่จะเกิดการร้องเรียนโดยกำหนดให้ N = 2 M = 2	0	0	16.79%
		1	0	9.48%
		0	1	13.14%
		1	1	5.46%
		3	3	4.49%

ตารางที่ 6.14 ตารางแสดงผลการตรวจสอบแบบจำลองระบบสั่งอาหารโดยตรวจสอบความน่าจะเป็นสูงสุด (ต่อ)

คุณสมบัติ	คำอธิบาย	claim_wait	claim_send	ความน่าจะเป็นสูงสุด
Pmax=? [F s=3]	หาความน่าจะเป็นสูงสุดที่จะเกิดการยกเลิกคำสั่งซื้อโดยกำหนดให้ N = 0 M = 0	0	0	23.04%
		0	1	23.04%
		2	2	23.04%
		1	0	23.04%
		3	3	23.04%
Pmax=? [F s=3]	หาความน่าจะเป็นสูงสุดที่จะเกิดการยกเลิกคำสั่งซื้อโดยกำหนดให้ N = 2 M = 2	0	0	10.10%
		1	1	10.10%
		2	2	10.10%
		2	3	10.10%
		3	3	10.10%

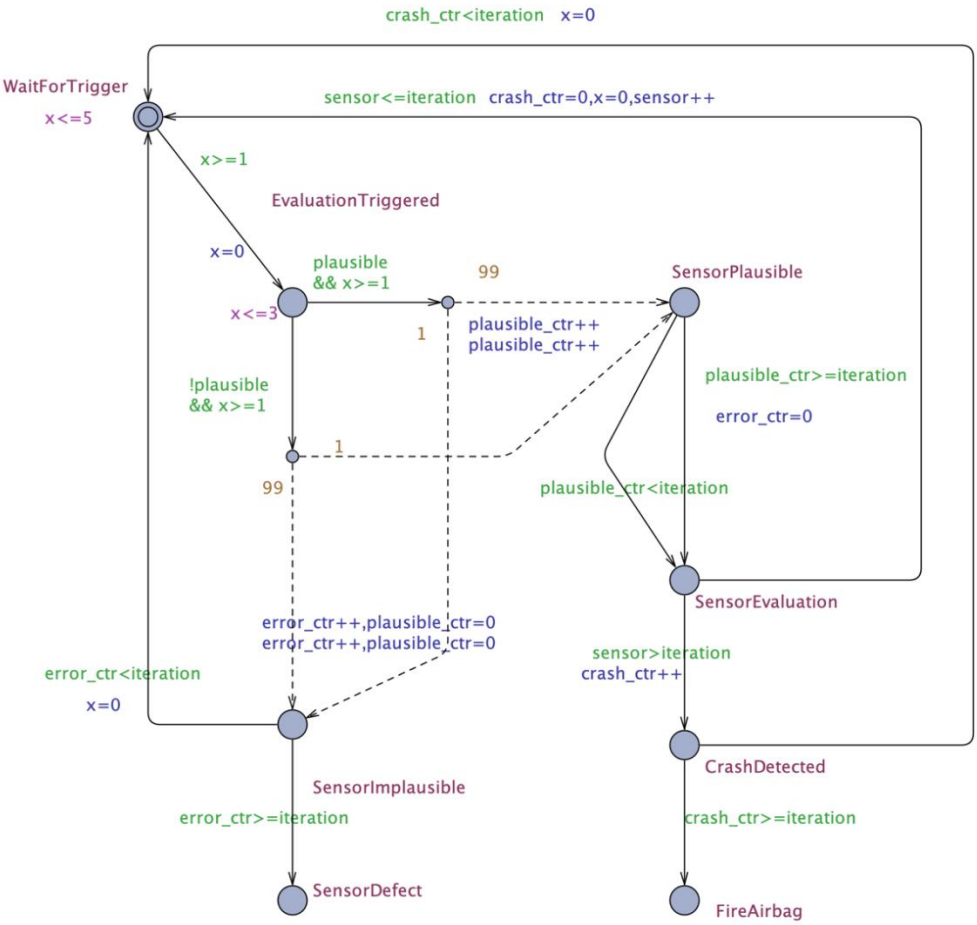
การวิเคราะห์แบบจำลองของระบบสั่งอาหารพบว่า จำนวนครั้งที่อนุญาตให้มีการส่งคำสั่งซื้อใหม่และรอการดำเนินการมีผลต่อโอกาสในการยกเลิกคำสั่งซื้อและการร้องเรียน โดยเฉพาะอย่างยิ่งพบว่าหากไม่อนุญาตให้มีการส่งคำสั่งซื้อซ้ำหรือการรอการดำเนินการ จะมีโอกาสยกเลิกคำสั่งซื้อสูงถึง 23.04% ในทางตรงกันข้าม หากมีนโยบายที่ยืดหยุ่นมากขึ้นที่อนุญาตให้ส่งคำสั่งซื้อใหม่และรอการดำเนินการ โอกาสในการยกเลิกคำสั่งซื้อจะลดลงเหลือเพียง 10.10% ผลการศึกษานี้สอดคล้องกับแบบจำลองที่ออกแบบไว้ซึ่งรวมถึงเงื่อนไขเวลา แสดงให้เห็นว่าความพึงพอใจและการรักษาลูกค้าสามารถปรับปรุงได้อย่างมากโดยปรับนโยบายการดำเนินงานเพื่ออนุญาตให้มีความยืดหยุ่นในเวลาดำเนินการ ความยืดหยุ่นนี้ดูเหมือนจะลดความไม่พอใจของลูกค้าและลดการยกเลิกคำสั่งซื้อในสภาพแวดล้อมบริการที่ถูกจำลองไว้

จากการวิเคราะห์พฤติกรรมของระบบสั่งอาหารปัจจัย claim_send และ claim_wait ซึ่งเป็นตัวแปรที่กำหนดความอดทนของลูกค้าต่อการส่งและการรอคำสั่งซื้อมีอิทธิพลสำคัญต่อการยื่นข้อร้องเรียน การทดลองพบว่าหากไม่มีนโยบายในการรอหรือส่งคำสั่งซื้อซ้ำเลย โอกาสการเกิดข้อร้องเรียนจะอยู่ที่ 3.84% อย่างไรก็ตาม เมื่อเพิ่มช่วงเวลาที่ยอมรับให้ลูกค้ารอและส่งคำสั่งซื้อซ้ำจาก 0 เป็น 2 หน่วยเวลา โอกาสในการยื่นข้อร้องเรียนเพิ่มขึ้นอย่างมีนัยสำคัญเป็น 16.79% การศึกษานี้

ชี้ให้เห็นว่านโยบายการจัดการกับความล่าช้าในการส่งและการรอคำสั่งซื้อสามารถส่งผลต่อความพึงพอใจของลูกค้าและการรักษาฐานลูกค้าได้อย่างมาก

2) กรณีศึกษาระบบการทำงานของถุงลมนิรภัย

จากรูปที่ 6.8 ผู้วิจัยได้จำลองระบบควบคุมถุงลมนิรภัยภายในรถยนต์โดยใช้แบบจำลองไทมด์ออโตมาตาความน่าจะเป็น เพื่อจำลองสถานการณ์ต่างๆ ที่อาจเกิดขึ้นและการตอบสนองของระบบนิรภัย ตั้งแต่การรอคำสั่งจากเหตุการณ์ชนหรือกระแทก การประเมินค่าที่ได้จากเซนเซอร์ การตรวจจับข้อบกพร่องของเซนเซอร์ ไปจนถึงการเปิดถุงลมนิรภัย ทุกสถานะในระบบนี้ถูกออกแบบมาเพื่อให้การป้องกันและความปลอดภัยกับผู้ใช้รถเป็นสำคัญ โดยระบบจะต้องตรวจสอบและประเมินสัญญาณจากเซนเซอร์อย่างต่อเนื่อง และเมื่อเกิดเหตุการณ์ชนหรือกระแทกที่รุนแรง ระบบจะทำการเปิดถุงลมนิรภัยโดยอัตโนมัติ เพื่อลดอันตรายที่อาจเกิดขึ้น การอธิบายแต่ละสถานะและการทำงานได้รับการอธิบายโดยละเอียดในตารางที่ 6.15



รูปที่ 6.8 แบบจำลองไทมด์ออโตมาตาความน่าจะเป็นของระบบการทำงานของถุงลมนิรภัย[16]

ตารางที่ 6.15 คำอธิบายสถานะในแต่ละตำแหน่งของแบบจำลองระบบการทำงานของถุงลมนิรภัย

ตำแหน่ง	คำอธิบาย
WaitForTrigger	สถานะเริ่มต้นของระบบ ที่ระบบจะรอคำสั่งหรือเหตุการณ์ที่จะกระตุ้นให้เกิดการประเมิน (ตัวอย่างเช่น การชน) ตัวแปร x ใช้เพื่อติดตามเวลาที่ระบบอยู่ในสถานะนี้ โดยจะต้องไม่เกิน 5 หน่วยเวลา
EvaluationTriggered	เมื่อได้รับการกระตุ้น ระบบจะเปลี่ยนไปยังสถานะนี้ และตัวแปร x จะถูกรีเซ็ตเป็น 0 เพื่อเริ่มการนับเวลาใหม่ ระบบจะประเมินค่าจากเซนเซอร์เพื่อตัดสินใจว่าจะเข้าสู่สถานะถัดไปหรือไม่
SensorPlausible	ถ้าค่าจากเซนเซอร์ดูเป็นไปได้ (plausible) ระบบจะเข้าสู่สถานะนี้ ตัวนับ plausible_ctr จะเพิ่มขึ้นทุกครั้งที่เข้าสู่สถานะนี้
SensorImplausible	ถ้าค่าจากเซนเซอร์ไม่เป็นไปได้ (!plausible) ระบบจะเข้าสู่สถานะนี้ และตัวนับ error_ctr จะเพิ่มขึ้น
SensorDefect	ถ้าตัวนับ error_ctr ถึง iteration หรือมากกว่า แสดงว่าเซนเซอร์มีข้อบกพร่อง และระบบจะเข้าสู่สถานะนี้
SensorEvaluation	ระบบจะประเมินเซนเซอร์อีกครั้งในสถานะนี้ ถ้าตัวนับ sensor เกิน iteration ระบบจะถือว่ามีการชนและ crash_ctr จะเพิ่มขึ้น
CrashDetected	ระบบตรวจจับว่ามีการชนจริงเกิดขึ้นแล้ว
FireAirbag	ถ้าตัวนับ crash_ctr ถึง iteration หรือมากกว่า ระบบจะเปิดถุงลมนิรภัย

จากรูปที่ 6.8 ผู้วิจัยได้จำลองการทำงานที่ผิดพลาดของเซ็นเซอร์โดยได้กำหนดว่าถ้าค่าจากเซนเซอร์ดูเป็นไปได้ที่จะเกิดการชน จะมีโอกาส 1% ที่จะทำงานผิดพลาด และถ้าเซ็นเซอร์ดูเป็นไปไม่ได้ว่าเกิดชนจะมีโอกาส 1% ที่ทำงานผิดพลาด และความรุนแรงของการชนจะถูกกำหนดโดยตัวแปร sensor โดยจำลองว่าทุกครั้งที่ประเมินเซนเซอร์ใหม่จะเพิ่มความรุนแรงขึ้น + 1

โดยสมมุติว่าในแต่ละรอบของการประเมินจะใช้ระยะเวลาในส่วนของการตรวจสอบการชนไม่เกิน 5 หน่วย และวิเคราะห์เซ็นเซอร์ไม่เกิน 3 หน่วยเวลา เมื่อนำแบบจำลองนี้มาแปลงเป็นรหัสปริซึมตามเครื่องมือที่ได้พัฒนา จะปรากฏผลลัพธ์ดังรูปที่ 6.9

```
pta
module Template

 s: [0..7] init 0;
 // WaitForTrigger : 0
 // EvaluationTriggered : 1
 // SensorPlausible : 2
 // SensorImplausible : 3
 // SensorEvaluation : 4
 // SensorDefect : 5
 // CrashDetected : 6
 // FireAirbag : 7
 x: clock;
 plausible: bool init true;
 plausible_ctr: int init 0;
 error_ctr: int init 0;
 sensor: int init 1;
 crash_ctr: int init 0;
 iteration: int init 3;



 invariant
  (s=0=>x<=5) & (s=1=>x<=3)
 endinvariant

 [] s=0 & (x>=1) -> (s'=1) & (x'=0) ;
 [] s=1 & (plausible& x>=1)
   -> 0.01 : (s'=3) & (error_ctr'=error_ctr+1) & (plausible_ctr'=0)
    + 0.99 : (s'=2) & (plausible_ctr'=plausible_ctr+1) ;
 [] s=1 & (!plausible& x>=1)
   -> 0.01 : (s'=2) & (plausible_ctr'=plausible_ctr+1)
    + 0.99 : (s'=3) & (error_ctr'=error_ctr+1) & (plausible_ctr'=0) ;
 [] s=2 & (plausible_ctr<iteration) -> (s'=4) ;
 [] s=2 & (plausible_ctr>=iteration) -> (s'=4) & (error_ctr'=0) ;
 [] s=3 & (error_ctr>=iteration) -> (s'=5) ;
 [] s=3 & (error_ctr<iteration) -> (s'=0) & (x'=0) ;
 [] s=4 & (sensor>iteration) -> (s'=6) & (crash_ctr'=crash_ctr+1) ;
 [] s=4 & (sensor<=iteration) -> (s'=0) & (crash_ctr'=0) & (x'=0) & (sensor'=sensor+1) ;
 [] s=6 & (crash_ctr>=iteration) -> (s'=7) ;
 [] s=6 & (crash_ctr<iteration) -> (s'=0) & (x'=0) ;

endmodule
```

รูปที่ 6.9 รหัสปริซึมจากการแปลงของแบบจำลองระบบการทำงานของถุงลมนิรภัย

ในแบบจำลองนี้ได้กำหนดตัวแปรสำหรับใช้ในการปรับแต่งแบบจำลอง เพื่อวิเคราะห์หาโอกาสที่ถุงลมนิรภัยจะเปิดหรือโอกาสที่เซนเซอร์จะทำงานผิดปกติ ผู้วิจัยได้กำหนดตัวแปรสำหรับการตั้งค่าไว้ 2 ตัวแปรดังนี้

2.1) plausible คือ ผลการประเมินที่ระบบทำกับข้อมูลที่ได้รับจากเซนเซอร์เพื่อดูว่ามีความสมเหตุสมผลหรือไม่ โดยกำหนดให้เป็นจริง (true) ถ้าสมเหตุสมผล และเป็นเท็จ (false) ถ้าไม่สมเหตุสมผล

2.2) iteration คือ จำนวนรอบของการประเมินโดยตัวแปรนี้จะส่งผลกับเวลาและความน่าจะเป็น

เพื่อวิเคราะห์ความน่าจะเป็นที่เกิดขึ้นในระบบ ผู้วิจัยจะนำเข้าแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นในรูปแบบภาษาปริซึมเข้าสู่เครื่องมือเพื่อทำการวิเคราะห์ โดยได้ผลลัพธ์ของการวิเคราะห์แสดงในตารางที่ 6.16

ตารางที่ 6.16 *ตารางแสดงผลการตรวจสอบแบบจำลองระบบการทำงานของถุงลมนิรภัย*

คุณสมบัติ	คำอธิบาย	iteration	ความน่าจะเป็นสูงสุด
Pmax=? [F s=7]	หาความน่าจะเป็นที่ถุงลมนิรภัยจะเปิด โดยกำหนดให้ plausible = true	0	0.9801%
		1	0.97029%
		2	0.999111%
		3	0.9999563%
		4	0.99999727%
Pmax=? [F<=15 s=7]	หาความน่าจะเป็นที่ถุงลมนิรภัยจะเปิดในระยะเวลา 15 หน่วย โดยกำหนดให้ plausible = true	0	0.9801%
		1	0.97029%
		2	0.999111%
		3	0.93206534%
		4	0%

จากการวิเคราะห์ความน่าจะเป็นของระบบถุงลมนิรภัยเพื่อหาโอกาสที่ถุงลมนิรภัยจะเปิด พบว่าการเพิ่มจำนวนรอบในการประเมินผลของเซนเซอร์ตรวจจับการชนและวิเคราะห์ความรุนแรงของการชน จะทำให้ความน่าจะเป็นที่ถุงลมนิรภัยจะเปิดสูงขึ้นโดยดูจากความละเอียดของการคำนวณความน่าจะเป็นที่ได้จากเครื่องมือตรวจสอบแบบจำลองปริซึมถ้าไม่เพิ่มรอบในการตรวจสอบเซนเซอร์เลยพบว่าความน่าจะเป็นอยู่ที่ 0.9801% แต่ถ้าเพิ่มรอบการตรวจสอบเป็น 4 รอบความน่าจะเป็น

สูงสุดอยู่ที่ 0.99999727% ซึ่งเพิ่มโอกาสในการเปิดถุงลมนิรภัยเกือบ 2% ซึ่งแน่นอนว่าสามารถลดอันตรายที่อาจเกิดขึ้น

แต่ถ้าหากมีเงื่อนไขทางด้านเวลามาเกี่ยวข้องโดยจำลองว่าระยะเวลา 15 หน่วยคือช่วงเวลาเดียวที่ถุงลมนิรภัยสามารถช่วยลดอันตรายได้จะพบว่าการเพิ่มรอบของการตรวจสอบมีผลต่อเวลาที่เกิดขึ้นในระบบซึ่งถ้าเพิ่มรอบการตรวจสอบเซนเซอร์เป็น 3 รอบมีโอกาสที่ถึงลมนิรภัยจะเปิดลดลงเหลือ 0.93206534% และถ้าเพิ่มรอบเป็น 4 รอบ มีโอกาสที่ถึงลมนิรภัยจะเปิดลดลงเหลือ 0% ซึ่งเป็นไปได้ว่าไม่สามารถเปิดถุงลมนิรภัยได้ทันตามเวลาเงื่อนไขของเวลา

6.5 สรุปผลการทดสอบ

การทดสอบครอบคลุมและการประเมินเครื่องมือการแปลงไทมด์ออโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึม ได้สำเร็จด้วยผลลัพธ์ที่น่าพึงพอใจอย่างมาก ทั้งนี้เครื่องมือที่พัฒนามาได้ทำงานตามที่ได้ออกแบบไว้และสามารถประยุกต์ใช้กฎการแปลงที่กำหนดขึ้นได้อย่างแม่นยำ เมื่อทำการทดสอบการแปลงข้อมูลจาก UPPAAL ในรูปแบบเอ็กซ์เอ็มแอล ไปเป็นรหัสปริซึมผลลัพธ์ที่ได้สอดคล้องกับเงื่อนไขและขอบเขตของวิทยานิพนธ์นี้อย่างเต็มรูปแบบ โดยมีความถูกต้อง 100% จากทุกกรณีทดสอบที่ได้ดำเนินการไป

นอกจากนี้ยังมีการทดสอบความสามารถในการตรวจจับข้อผิดพลาดของแบบจำลอง ซึ่งในกรณีที่มีข้อผิดพลาดในแบบจำลองเอ็กซ์เอ็มแอล ระบบสามารถรายงานข้อผิดพลาดได้อย่างแม่นยำ 100% จากกรณีทดสอบทั้งหมดที่ได้ทำไป สุดท้ายการทดสอบโดยใช้กรณีศึกษายังแสดงให้เห็นว่ารหัสปริซึมที่ได้จากเครื่องมือสามารถนำไปวิเคราะห์ต่อในเครื่องมือตรวจสอบแบบจำลองปริซึมและได้ผลลัพธ์ที่สอดคล้องกับแบบจำลองที่กำหนดไว้

บทที่ 7

สรุปผลการวิจัย

7.1 สรุปผลการวิจัย

ในวิทยานิพนธ์ฉบับนี้ได้พัฒนากระบวนการและกฎการแปลงเพื่อการแปลงแบบจำลอง ไทมด์ออโตมาตาความน่าจะเป็นรหัสปริซึมที่มีความสอดคล้องทางความหมาย กระบวนการและกฎที่ออกแบบมานี้ช่วยให้สามารถแปลงแบบจำลองที่ถูกออกแบบไว้โดยเครื่องมือ UPPAAL ในรูปแบบเอ็กซ์เอ็มแอล ได้อย่างมีเหมาะสม การแปลงช่วยให้สามารถนำแบบจำลองที่ได้ไปวิเคราะห์พฤติกรรมที่มีความน่าจะเป็นผ่านเครื่องมือตรวจสอบแบบจำลองปริซึมได้อย่างง่ายดาย ผลลัพธ์จากการศึกษานี้ทำให้สามารถประยุกต์ใช้ทฤษฎีได้ในแบบจำลองจริง และนำไปสู่การใช้งานที่ประสิทธิผลมากขึ้นในด้านต่างๆ เช่น การวิเคราะห์ระบบความปลอดภัย การทดสอบโปรโตคอลการสื่อสาร และการจำลองสถานการณ์ที่มีความซับซ้อนสูง ฯลฯ

นอกจากนี้วิทยานิพนธ์ฉบับนี้ยังได้นำเสนอกระบวนการพัฒนาเครื่องมือที่สามารถแปลงแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึมได้อย่างมีประสิทธิภาพ ความสำเร็จของการแปลงนี้ไม่เพียงแต่ช่วยให้สามารถวิเคราะห์แบบจำลองได้ง่ายขึ้น แต่ยังช่วยให้สามารถนำเทคนิคที่ได้มาใช้กับกรณีศึกษาจริง ๆ ได้อีกด้วย การใช้ภาษาจาวาจากการสร้างโปรแกรมเชิงวัตถุที่มีความสอดคล้องสูงกับรูปแบบเอ็กซ์เอ็มแอล ที่ได้จาก UPPAAL ในการพัฒนาเครื่องมือทำให้มันมีความยืดหยุ่นในการปรับใช้กับระบบต่าง ๆ และสามารถปรับปรุงหรือพัฒนาต่อได้อย่างง่ายดาย การทดสอบเครื่องมือที่พัฒนาขึ้นพบว่าสามารถแปลงแบบจำลองได้อย่างถูกต้องและครบถ้วนตามที่ได้ระบุไว้ในสเปค และแม้แต่การจัดการกับข้อผิดพลาดที่อาจเกิดขึ้นระหว่างกระบวนการแปลงก็ทำได้ด้วยความแม่นยำสูง

การประยุกต์ใช้เครื่องมือนี้กับกรณีศึกษาที่หลากหลายได้แสดงให้เห็นถึงความสามารถของเครื่องมือในการวิเคราะห์และทดสอบแบบจำลองที่มีความซับซ้อน เครื่องมือนี้ไม่เพียงแต่ช่วยลดเวลาและความพยายามที่จำเป็นในการวิเคราะห์แบบจำลองด้วยตนเอง แต่ยังช่วยให้มั่นใจได้ว่าผลลัพธ์ที่ได้มีความถูกต้องและเชื่อถือได้ ด้วยการทดสอบที่ครอบคลุมและการจัดการข้อผิดพลาดที่แม่นยำ เครื่องมือนี้พิสูจน์ได้ถึงคุณค่าในการประยุกต์ใช้ในสภาพแวดล้อมที่หลากหลายและมีความต้องการทางเทคนิคที่สูง

เครื่องมือที่พัฒนานี้ยังมีศักยภาพในการเป็นเครื่องมือมาตรฐานสำหรับการวิเคราะห์แบบจำลองไทมด์ออโตมาตาความน่าจะเป็นในอนาคต ด้วยการใช้แนวทางที่มีระเบียบวิธีและเป็น

ระบบ วิทยานิพนธ์นี้ได้ส่งเสริมให้เกิดการพัฒนาเครื่องมือที่สามารถนำไปใช้ในการวิเคราะห์ระบบที่มีความซับซ้อนและมีความผันผวนสูงในโลกของการพัฒนาซอฟต์แวร์สมัยใหม่ ผลลัพธ์จากการทดสอบที่มีความถูกต้องและครบถ้วนนี้เป็นหลักฐานที่สนับสนุนความสำคัญของการพัฒนาเครื่องมือเชิงระบบที่สามารถจัดการกับความต้องการที่ซับซ้อนของระบบในโลกจริง

7.2 ประโยชน์ที่ได้รับ

1) ได้ทราบกฎการแปลงแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึมที่สนับสนุนการวิเคราะห์คุณสมบัติเชิงปริมาณ
2) ได้เครื่องมือการแปลงแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึมที่สนับสนุนการวิเคราะห์คุณสมบัติเชิงปริมาณ

7.3 ปัญหาและอุปสรรค

ในการออกแบบกฎการแปลงและพัฒนาเครื่องมือการแปลงแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นเป็นรหัสปริซึม ผู้วิจัยได้เผชิญกับอุปสรรคและความท้าทายหลายประการ

ประการแรกคือการทำความเข้าใจรูปแบบและโครงสร้างของแบบจำลองไทมด์ออโตมาความน่าจะเป็นที่ถูกสร้างบนเครื่องมือ UPPAAL ที่สามารถสร้างได้ในรูปแบบที่หลากหลายมีความซับซ้อนและกระบวนการที่ต้องใช้ในการทำให้แบบจำลองมีความสอดคล้องทางความหมายกับรหัสปริซึม นอกจากนี้ยังมีการจัดการกับข้อกำหนดทางเทคนิคของระบบแบบจำลองเช่น การจำกัดเวลาและเงื่อนไขความน่าจะเป็นที่ต้องถูกแปลงอย่างถูกต้องเพื่อให้สามารถวิเคราะห์ได้โดยเครื่องมือปริซึม

ประการที่สองคือข้อจำกัดด้านเทคนิคของการแปลงข้อมูลและความแม่นยำที่ต้องมีเมื่อทำการแปลงจาก UPPAAL ไปยังรหัสปริซึมทำให้ต้องพัฒนาอัลกอริทึมและวิธีการที่รัดกุมเพื่อจัดการกับความไม่ตรงกันของโครงสร้างข้อมูลและความแม่นยำที่ต้องการ นอกจากนี้ ผู้วิจัยยังต้องปรับปรุงอย่างต่อเนื่องเพื่อให้สามารถรองรับลักษณะเฉพาะตัวอย่างเช่น รูปแบบการกำหนดตัวแปรที่หลากหลายของ UPPAAL รวมไปถึงการกำหนดความน่าจะเป็นในรูปแบบนำหนักและเปอร์เซ็นที่ไม่สอดคล้องกัน

สุดท้ายนี้ การรักษาความเข้ากันได้ของแบบจำลองระหว่างเครื่องมือ UPPAAL และปริซึมการทำงานที่สอดคล้องกัน เนื่องจากในบางกรณีการแปลงแบบจำลองจากเครื่องมือ UPPAAL เมื่อนำมาตรวจสอบแบบจำลองบนเครื่องมือปริซึมพบว่าต้องใช้ทรัพยากรการคำนวณอย่างมาก อันเกิดจากการออกแบบออโตมาตาที่ทำให้เกิดกระบวนการประมวลผลไม่สิ้นสุดบนเครื่องมือปริซึมที่ปัจจุบันยังไม่สามารถแก้ปัญหานี้ได้

7.4 แนวทางในการประยุกต์ร่วมกับงานวิจัยอื่น

แนวทางในการประยุกต์ร่วมกับงานวิจัยอื่นสำหรับเครื่องมือการแปลงแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึมนั้นมีหลายด้านที่น่าสนใจ และสามารถส่งเสริมการวิจัยในหลากหลายสาขาได้อย่างมีประสิทธิภาพในด้านต่างๆ ที่เป็นไปได้ได้แก่

1) การวิจัยในด้านระบบความปลอดภัย: เครื่องมือการแปลงสามารถใช้ในการวิเคราะห์ระบบความปลอดภัยที่มีความซับซ้อน เช่น ระบบการจราจรอากาศหรือระบบควบคุมในโรงงานอุตสาหกรรม เพื่อทดสอบและวิเคราะห์ความเป็นไปได้ของสถานการณ์ต่างๆ ที่อาจเกิดขึ้น
2) การพัฒนาและทดสอบโปรโตคอลการสื่อสาร: โปรโตคอลการสื่อสารที่มีความซับซ้อนสามารถวิเคราะห์ได้ดีขึ้นด้วยการใช้เครื่องมือนี้ โดยสามารถจำลองสถานการณ์ที่แตกต่างกันได้หลายอย่าง เพื่อทดสอบและยืนยันความถูกต้องของโปรโตคอล
3) การวิจัยด้านระบบสุขภาพ: ในด้านการแพทย์และสุขภาพ การใช้เครื่องมือนี้อาจช่วยในการวิเคราะห์ระบบต่างๆ เช่น ระบบการจัดการยา หรือระบบภายในโรงพยาบาล เพื่อประเมินความเสี่ยงและปรับปรุงกระบวนการทำงาน
4) การทดสอบและประเมินผลระบบการเงินและการธนาคาร: ระบบการเงินและธนาคารที่มีความซับซ้อนสามารถใช้เครื่องมือนี้ในการจำลองสถานการณ์ต่างๆ เพื่อทดสอบและประเมินผลกระทบของการตัดสินใจทางการเงินต่างๆ
5) การวิจัยด้านระบบขนส่งและโลจิสติกส์: เครื่องมือนี้สามารถช่วยในการวิเคราะห์และทดสอบระบบขนส่งและโลจิสติกส์ เช่น การจำลองกระบวนการจัดส่งสินค้าและการเคลื่อนย้ายสินค้า

โดยรวมแล้ววิทยานิพนธ์นี้นำเสนอเครื่องมือมีศักยภาพในการประยุกต์ใช้ในหลายสาขาวิชาและสามารถนำไปใช้ร่วมกับงานวิจัยอื่นๆ เพื่อพัฒนาวิธีการและกระบวนการที่มีประสิทธิภาพและแม่นยำมากขึ้นในการจัดการกับปัญหาที่มีความซับซ้อนและหลากหลาย

7.5 แนวทางในวิจัยต่อไป

แนวทางในการวิจัยต่อไปสำหรับการพัฒนากฎการแปลงและเครื่องมือการแปลงแบบจำลองไทมด์ออโตมาตาความน่าจะเป็นไปเป็นรหัสปริซึมให้มีความลึกซึ้งและกว้างขวางขึ้น ดังนี้

1) การพัฒนากฎการแปลงให้สนับสนุน “urgent” ที่บ่งบอกถึงเหตุการณ์หรือสถานะที่ต้องเกิดขึ้นทันทีโดยไม่มีการหน่วงเวลา และ “synchronize”การประสานการทำงานระหว่างสองออโตมาตาหรือมากกว่า

2) การกำหนดกฎการแปลงที่รองรับแทรนซิชันความน่าจะเป็นในรูปแบบของตัวแปรที่ช่วยให้สามารถสร้างระบบที่มีความซับซ้อนสูง
3) การออกพัฒนา UX/UI ที่รองรับการวาดแบบจำลองที่ให้ผลลัพธ์สอดคล้องกับเครื่องมือ UPPAAL
4) การผสานร่วมกันระหว่างเครื่องมือแปลงแบบจำลอง และเครื่องมือตรวจสอบแบบจำลองปริซึม

บรรณานุกรม

1. Gethin Norman, D.P., Jeremy Sproston, *Model Checking for Probabilistic Timed Automata.* Formal Methods in System Design, 2013: p. 164-190.
2. Alur, R.a.D.L.D., *A theory of timed automata.* Theoretical Computer Science 126, 1994: p. 183-235.
3. Arnd Hartmanns, H.H., *A Modest Approach to Checking Probabilistic Timed Automata*, in *2009 Sixth International Conference on Quantitative Evaluation of Systems*. 2009. p. 187-196.
4. Gerd Behrmann, A.D., Kim G. Larsen, *"A Tutorial on Uppaal 4.0" Updated November 28.* 2006: p. 1-19.
5. Marta Kwiatkowska, G.N., David Parker, *PRISM 4.0: Verification of Probabilistic Real-Time Systems*, in *Computer Aided Verification*. 2011. p. 585-591.
6. Rajeev Alur, T.A.H., *Reactive Modules.* Formal Methods in System Design 15, 1999: p. 7-48.
7. Bellman, R., *A Markovian Decision Process.* Indiana Univ. Math. J. 6 No. 4, 1957: p. 679–684.
8. Aaamir Naeem, F.A., Anam Amjad, Muhammad Waseem Anwar *Comparison of Model Checking Tools Using Timed Automata - PRISM and UPPAAL* in *IEEE International Conference on Computer and Communication Engineering Technology (CCET)* 2018. p. 248-253.
9. Simon Wimmer, J.H., *MDP + TA = PTA:Probabilistic Timed Automata, Formalized.*, in *8th International Conference on Interactive Theorem Proving* 2018. p. 597-603.
10. Gethin Norman, D.P., Jeremy Sproston, *Model Checking for Probabilistic Timed Automata.* 2013: p. 164-190.
11. *PRISM Manual version 4.7*. 2021,19 March, Available from: http://www.prismmodelchecker.org/manual.
12. V. Kaczmarczyk, M.S., Z. Bradác, *Stochastic timed automata simulator*, in *Proceedings of the 4th conference on European computing conference*. 2010. p.

52-57.

13. Marta Kwiatkowska, G.N., David Parker, *Probabilistic Model Checking and Power-Aware Computing*, in *7th International Workshop on Performability Modeling of Computer and Communication Systems (PMCCS)*. 2005. p. 6-9.
14. Rahul Dubey, B.E., Reginald Lewis, Priya Karunakaran, *Algorithm To Code Converter*, in *2nd International Conference on Trends in Electronics and Informatics (ICOEI)*. 2018. p. 657-661.
15. Andrejs Bajovs, O.N., Janis Sejans, *Code Generation from UML Model: State of the Art and Practical Implications.* Applied Computer Systems, 2013: p. 9-18.
16. Felix Hübner, W.-l.H., Jan Peleska, *Experimental evaluation of a novel equivalence class partition testing strategy.* Software and Systems Modeling, 2019. **18**: p. 1-21.

ประวัติผู้เขียน

ชื่อ-สกุล	นายถิรวัตร สุตาลังกา
วัน เดือน ปี เกิด	3 พฤศจิกายน 2535
สถานที่เกิด	ลำปาง
วุฒิการศึกษา	มหาวิทยาลัยเชียงใหม่ ศิลปศาสตรบัณฑิต สาขาการจัดการสมัยใหม่และเทคโนโลยีสารสนเทศ
ที่อยู่ปัจจุบัน	179 หมู่ 1 ต.หนองหล่ม อ.ห้างฉัตร จ.ลำปาง 52190

ภาคผนวก

ภาคผนวก ก
การทดสอบความถูกต้องของการแปลงแบบจำลอง

1. กรณีทดสอบความถูกต้องของการแปลงจากแบบจำลองการทำงานของเครื่องจักร

เอกสารเอ็กซ์เอ็มแอล จากเครื่องมือ UPPAAL ของแบบจำลองการทำงานของเครื่องจักร

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE nta PUBLIC '-//Uppaal Team//DTD Flat System 1.1//EN'
'http://www.it.uu.se/research/group/darts/uppaal/flat-1_2.dtd'>
<nta>
   <declaration>
   // Declare the clock variable
   clock x;
 </declaration>
   <template>
      <name>P</name>
      <location id="id0" x="110" y="93">
         <name x="100" y="63">Work</name>
         <label kind="invariant" x="93" y="119">x&amp;lt;=5</label>
      </location>
      <location id="id1" x="340" y="25">
         <name x="330" y="-5">Complete</name>
      </location>
      <location id="id2" x="340" y="170">
         <name x="330" y="140">Fail</name>
      </location>
      <branchpoint id="id3" x="238" y="93">
      </branchpoint>
      <init ref="id0"/>
      <transition>
         <source ref="id0"/>
         <target ref="id3"/>
         <label kind="guard" x="161" y="68">x &amp;gt;=1</label>
      </transition>
      <transition>
         <source ref="id3"/>
         <target ref="id1"/>
         <label kind="probability" x="263" y="51">70</label>
      </transition>
      <transition>

         <source ref="id3"/>
         <target ref="id2"/>
         <label kind="probability" x="255" y="127">30</label>
      </transition>
   </template>
   <system>pta1 = P();
system pta1;</system>
   <queries>
      <query>
         <formula></formula>
         <comment></comment>
      </query>
   </queries>
</nta>
```

รหัสปริซึมจากการแปลงของแบบจำลองการทำงานของเครื่องจักร

```
pta
module P

  s: [0..2] init 0;
  // Work : 0
  // Complete : 1
  // Fail : 2
  x: clock;

  invariant
    (s=0=>x<=5)
  endinvariant

  [] s=0 & (x >=1)
     -> 0.7 : (s'=1)
      + 0.3 : (s'=2) ;

endmodule
```

2. กรณีทดสอบความถูกต้องของการแปลงจากแบบจำลองการส่งข้อความผ่านช่องทางที่ไม่น่าเชื่อถือ

เอกสารเอ็กซ์เอ็มแอล จากเครื่องมือ UPPAAL ของแบบจำลองการส่งข้อความผ่านช่องทางที่ไม่น่าเชื่อถือ

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE nta PUBLIC '-//Uppaal Team//DTD Flat System 1.1//EN'
'http://www.it.uu.se/research/group/darts/uppaal/flat-1_2.dtd'>
<nta>
	<template>
		<name x="5" y="5">PTA</name>
		<declaration>
		int tries = 0;
		clock x;
		const int N = 0;</declaration>
		<location id="id0" x="-8" y="-25">
			<name x="-42" y="-59">Initial</name>
			<label kind="invariant" x="-68" y="-33">x&amp;lt;=2</label>
		</location>
		<location id="id1" x="-8" y="204">
			<name x="-51" y="170">Fail</name>
		</location>
		<location id="id2" x="246" y="204">
			<name x="238" y="161">SendComplete</name>
		</location>
		<location id="id3" x="246" y="-25">
			<name x="221" y="-59">PendingReSend</name>
			<label kind="invariant" x="236" y="-8">x&amp;lt;=5</label>
		</location>
		<branchpoint id="id4" x="136" y="110">
		</branchpoint>
		<init ref="id0"/>
		<transition>
			<source ref="id3"/>
			<target ref="id0"/>
			<label kind="guard" x="178" y="-51">x&amp;gt;=3</label>
			<label kind="assignment" x="34" y="-51">x:=0</label>
		</transition>
		<transition>
			<source ref="id0"/>
			<target ref="id4"/>
			<label kind="guard" x="8" y="34">x&amp;gt;=1 &amp;amp;&amp;amp; tries &amp;lt;=
3</label>
		</transition>
		<transition>
			<source ref="id4"/>
			<target ref="id2"/>
			<label kind="probability" x="161" y="153">90</label>
		</transition>
		<transition>
```

```
                        <source ref="id4"/>
                        <target ref="id3"/>
                        <label kind="assignment" x="187" y="51">tries++,x=0</label>
                        <label kind="probability" x="161" y="76">10</label>
                </transition>
                <transition>
                        <source ref="id0"/>
                        <target ref="id1"/>
                        <label kind="guard" x="-85" y="102">tries &gt; 3</label>
                </transition>
        </template>
        <system>pta1 = PTA();
system pta1;</system>
        <queries>
                <query>
                        <formula>E&lt;&gt; pta1.S2 </formula>
                        <comment></comment>
                </query>
                <query>
                        <formula>Pr[&lt;=100](&lt;&gt; pta1.S2 )</formula>
                        <comment></comment>
                </query>
        </queries>
</nta>
```

รหัสปริซึมจากการแปลงของแบบจำลองการส่งข้อความผ่านช่องทางที่ไม่น่าเชื่อถือ

```
pta
module PTA

 s: [0..3] init 0;
 // Initial : 0
 // Fail : 1
 // SendComplete : 2
 // PendingReSend : 3
 tries: int init 0;
 x: clock;
 N: int init 0;

 invariant
  (s=0=>x<=2) & (s=3=>x<=5)
 endinvariant

 [] s=0 & (x>=1 & tries <= 3)
   -> 0.9 : (s'=2)
    + 0.1 : (s'=3) & (tries'=tries+1) & (x'=0) ;
 [] s=0 & (tries > 3) -> (s'=1) ;
 [] s=3 & (x>=3) -> (s'=0) & (x'=0) ;

endmodule
```

3. กรณีทดสอบความถูกต้องของการแปลงจากแบบจำลองระบบสั่งซื้ออาหาร

เอกสารเอ็กซ์เอ็มแอล จากเครื่องมือ UPPAAL ของแบบจำลองระบบสั่งซื้ออาหาร

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE nta PUBLIC '-//Uppaal Team//DTD Flat System 1.1//EN'
'http://www.it.uu.se/research/group/darts/uppaal/flat-1_2.dtd'>
<nta>
	<declaration>// Place global declarations here.
clock x;
int send = 0;
const int N = 0;
int wait = 0;
const int M = 0;
int claim_wait = 1;
int claim_send = 1;</declaration>
	<template>
		<name x="5" y="5">PTA</name>
		<declaration>// Place local declarations here.</declaration>
		<location id="id0" x="-755" y="-229">
			<name x="-832" y="-271">SendOrder</name>
			<label kind="invariant" x="-807" y="-255">x&amp;lt;=2</label>
		</location>
		<location id="id1" x="-493" y="-25">
			<name x="-503" y="-59">ReceiveOrder</name>
			<label kind="invariant" x="-476" y="-34">x&amp;lt;=5</label>
		</location>
		<location id="id2" x="-493" y="-229">
			<name x="-503" y="-263">ReSendOrder</name>
			<label kind="invariant" x="-476" y="-246">x&amp;lt;=5</label>
		</location>
		<location id="id3" x="-756" y="42">
			<name x="-841" y="8">CancelOrder</name>
		</location>
		<location id="id4" x="-187" y="-42">
			<name x="-170" y="-68">OrderProcessing</name>
			<label kind="invariant" x="-144" y="-51">x&amp;lt;=5</label>
		</location>
		<location id="id5" x="-365" y="-221">
			<name x="-382" y="-255">Claim</name>
		</location>
		<location id="id6" x="-187" y="-221">
			<name x="-238" y="-272">OrderComplate</name>
			<label kind="invariant" x="-204" y="-255">x&amp;lt;=2</label>
		</location>
		<location id="id7" x="-357" y="42">
			<name x="-382" y="-8">WaitOrder</name>
			<label kind="invariant" x="-374" y="8">x&amp;lt;=5</label>
		</location>
```

```
                    <location id="id8" x="0" y="-221">
                        <name x="-26" y="-255">NoCaim</name>
                    </location>
                    <branchpoint id="id9" x="-187" y="42">
                    </branchpoint>
                    <branchpoint id="id10" x="-493" y="42">
                    </branchpoint>
                    <branchpoint id="id11" x="-187" y="-136">
                    </branchpoint>
                    <branchpoint id="id12" x="-628" y="-127">
                    </branchpoint>
                    <init ref="id0"/>
                    <transition>
                        <source ref="id6"/>
                        <target ref="id8"/>
                        <label kind="guard" x="-161" y="-212">wait&lt;=claim_wait
&amp;&amp; send&lt;=claim_send</label>
                    </transition>
                    <transition>
                        <source ref="id7"/>
                        <target ref="id3"/>
                        <label kind="guard" x="-535" y="76">wait &gt; M</label>
                        <nail x="-357" y="102"/>
                        <nail x="-756" y="102"/>
                    </transition>
                    <transition>
                        <source ref="id6"/>
                        <target ref="id5"/>
                        <label kind="guard" x="-357" y="-212">wait &gt; claim_wait
|| send &gt; claim_send</label>
                    </transition>
                    <transition>
                        <source ref="id10"/>
                        <target ref="id3"/>
                        <label kind="probability" x="-561" y="42">10</label>
                    </transition>
                    <transition>
                        <source ref="id10"/>
                        <target ref="id7"/>
                        <label kind="assignment" x="-467" y="42">x=0</label>
                        <label kind="probability" x="-459" y="17">90</label>
                    </transition>
                    <transition>
                        <source ref="id1"/>
                        <target ref="id10"/>
                        <label kind="guard" x="-544" y="-8">x&gt;=1</label>
                    </transition>
                    <transition>
                        <source ref="id9"/>
```

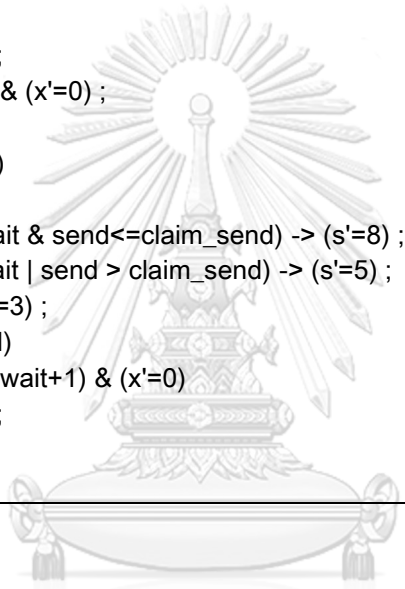
```
                        <target ref="id7"/>
                        <label kind="assignment" x="-306" y="102">wait++,x=0</label>
                        <label kind="probability" x="-221" y="76">10</label>
                        <nail x="-263" y="102"/>
                </transition>
                <transition>
                        <source ref="id9"/>
                        <target ref="id4"/>
                        <label kind="assignment" x="-229" y="-8">x=0</label>
                        <label kind="probability" x="-170" y="-8">90</label>
                </transition>
                <transition>
                        <source ref="id7"/>
                        <target ref="id9"/>
                        <label kind="guard" x="-331" y="17">x&amp;gt;=1
&amp;amp;&amp;amp;wait&amp;lt;=M</label>
                </transition>
                <transition>
                        <source ref="id11"/>
                        <target ref="id6"/>
                        <label kind="assignment" x="-170" y="-153">x=0</label>
                        <label kind="probability" x="-170" y="-170">95</label>
                </transition>
                <transition>
                        <source ref="id11"/>
                        <target ref="id5"/>
                        <label kind="probability" x="-297" y="-136">5</label>
                        <nail x="-365" y="-136"/>
                </transition>
                <transition>
                        <source ref="id4"/>
                        <target ref="id11"/>
                        <label kind="guard" x="-178" y="-102">x&amp;gt;=1</label>
                </transition>
                <transition>
                        <source ref="id0"/>
                        <target ref="id3"/>
                        <label kind="guard" x="-824" y="-119">send &amp;gt; N</label>
                </transition>
                <transition>
                        <source ref="id2"/>
                        <target ref="id0"/>
                        <label kind="guard" x="-569" y="-255">x&amp;gt;=3</label>
                        <label kind="assignment" x="-730" y="-263">x:=0</label>
                </transition>
                <transition>
                        <source ref="id12"/>
                        <target ref="id2"/>
                        <label kind="assignment" x="-543" y="-170">send++,x=0</label>
```

```
                        <label kind="probability" x="-611" y="-178">5</label>
                </transition>
                <transition>
                        <source ref="id12"/>
                        <target ref="id1"/>
                        <label kind="assignment" x="-595" y="-85">x=0</label>
                        <label kind="probability" x="-585" y="-127">95</label>
                </transition>
                <transition>
                        <source ref="id0"/>
                        <target ref="id12"/>
                        <label kind="guard" x="-722" y="-
195">x&amp;gt;=1&amp;amp;&amp;amp;send&amp;lt;=N</label>
                </transition>
        </template>
        <system>// Place template instantiations here.
Process = PTA();
// List one or more processes to be composed into a system.
system Process;
   </system>
        <queries>
                <query>
                        <formula></formula>
                        <comment></comment>
                </query>
        </queries>
</nta>
```

รหัสปริซึมจากการแปลงของแบบจำลองระบบสั่งซื้ออาหาร


```
pta
module PTA

  s: [0..8] init 0;
  // SendOrder : 0
  // ReceiveOrder : 1
  // ReSendOrder : 2
  // CancelOrder : 3
  // OrderProcessing : 4
  // Claim : 5
  // OrderComplate : 6
  // WaitOrder : 7
  // NoCaim : 8
  x: clock;
  send: int init 0;
  N: int init 0;
  wait: int init 0;
  M: int init 0;
```

```
  claim_wait: int init 1;
  claim_send: int init 1;

  invariant
    (s=0=>x<=2) & (s=1=>x<=5) & (s=2=>x<=5) & (s=4=>x<=5) & (s=6=>x<=2) &
(s=7=>x<=5)
  endinvariant

  [] s=0 & (send > N) -> (s'=3) ;
  [] s=0 & (x>=1&send<=N)
     -> 0.05 : (s'=2) & (send'=send+1) & (x'=0)
      + 0.95 : (s'=1) & (x'=0) ;
  [] s=1 & (x>=1)
     -> 0.1 : (s'=3)
      + 0.9 : (s'=7) & (x'=0) ;
  [] s=2 & (x>=3) -> (s'=0) & (x'=0) ;
  [] s=4 & (x>=1)
     -> 0.95 : (s'=6) & (x'=0)
      + 0.05 : (s'=5) ;
  [] s=6 & (wait<=claim_wait & send<=claim_send) -> (s'=8) ;
  [] s=6 & (wait > claim_wait | send > claim_send) -> (s'=5) ;
  [] s=7 & (wait > M) -> (s'=3) ;
  [] s=7 & (x>=1 &wait<=M)
     -> 0.1 : (s'=7) & (wait'=wait+1) & (x'=0)
      + 0.9 : (s'=4) & (x'=0) ;

endmodule
```



ภาคผนวก ข
การทดสอบการจัดการข้อผิดพลาด

1. กรณีทดสอบความผิดพลาดแบบจำลองไม่ได้กำหนดเส้นแยกบนจุดแยก

เอกสารเอ็กซ์เอ็มแอล จากเครื่องมือ UPPAAL ของแบบจำลองไม่ได้กำหนดเส้นแยกบนจุดแยก

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE nta PUBLIC '-//Uppaal Team//DTD Flat System 1.1//EN'
'http://www.it.uu.se/research/group/darts/uppaal/flat-1_2.dtd'>
<nta>
        <declaration>// Place global declarations here.</declaration>
        <template>
                <name x="5" y="5">Template</name>
                <declaration>// Place local declarations here.</declaration>
                <location id="id0" x="-17" y="0">
                </location>
                <location id="id1" x="263" y="-68">
                </location>
                <location id="id2" x="263" y="76">
                </location>
                <branchpoint id="id3" x="144" y="0">
                </branchpoint>
                <init ref="id0"/>
                <transition>
                        <source ref="id0"/>
                        <target ref="id3"/>
                </transition>
        </template>
        <system>// Place template instantiations here.
Process = Template();
// List one or more processes to be composed into a system.
system Process;
   </system>
        <queries>
                <query>
                        <formula></formula>
                        <comment></comment>
                </query>
        </queries>
</nta>
```

2. กรณีทดสอบความผิดพลาดแบบจำลองไม่ได้กำหนดค่าความน่าจะเป็น

เอกสารเอ็กซ์เอ็มแอล จากเครื่องมือ UPPAAL ของแบบจำลองไม่ได้กำหนดค่าความน่าจะเป็น

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE nta PUBLIC '-//Uppaal Team//DTD Flat System 1.1//EN'
'http://www.it.uu.se/research/group/darts/uppaal/flat-1_2.dtd'>
<nta>
        <declaration>// Place global declarations here.</declaration>
        <template>
                <name x="5" y="5">Template</name>
                <declaration>// Place local declarations here.</declaration>
                <location id="id0" x="-17" y="0">
                </location>
                <location id="id1" x="263" y="-68">
                </location>
                <location id="id2" x="263" y="76">
                </location>
                <branchpoint id="id3" x="144" y="0">
                </branchpoint>
                <init ref="id0"/>
                <transition>
                        <source ref="id3"/>
                        <target ref="id2"/>
                </transition>
                <transition>
                        <source ref="id3"/>
                        <target ref="id1"/>
                </transition>
                <transition>
                        <source ref="id0"/>
                        <target ref="id3"/>
                </transition>
        </template>
        <system>// Place template instantiations here.
Process = Template();
// List one or more processes to be composed into a system.
system Process;
   </system>
        <queries>
                <query>
                        <formula></formula>
                        <comment></comment>
                </query>
        </queries>
</nta>
```