การแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาไปเป็นไทมด์ออโตมาตา

นายณรงค์กร วงศ์สิทธิไพฑูรย์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2565

Transforming YAWL Workflows with Time Interval Constraints to Timed Automata

Mr. Naronggorn Wongsitthiphaithun

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 2022

หัวข้อวิทยานิพนธ์	การแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาไปเป็นไทมด์ออโตมาตา
โดย	นายณรงค์กร วงศ์สิทธิไพฑูรย์
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
(ดร.เนื่องวงศ์ ทวยเจริญ)

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

ณรงค์กร วงศ์สิทธิไพฑูรย์ : การแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาไปเป็นไทมด์ออโตมาตา. (Transforming YAWL Workflows with Time Interval Constraints to Timed Automata) อ.ที่ปรึกษาหลัก : รศ. ดร.วิวัฒน์ วัฒนาวุฒิ

กระแสงานยอว์ลเป็นหนึ่งในกระแสงานทางธุรกิจที่ทันสมัย กระแสงานยอว์ลให้การแสดงเป็นภาพขั้นตอนกระแสงานของงานทางธุรกิจที่เข้าใจง่าย สามารถกำหนดเวลาการทำงานของงานภายในกระแสงานยอว์ลได้ วิทยานิพนธ์ฉบับนี้จะเน้นให้นำกระแสงานยอว์ลทั่วไปมาเพิ่มขีดความสามารถโดยการเพิ่ม ข้อจำกัดแบบช่วงเวลาให้กับแต่ละสัญลักษณ์งานในกระแสงานยอว์ล ผ่านค่าเฉลี่ยของข้อจำกัดของช่วงเวลาในรูปของขอบเขตล่าง และขอบเขตบนของเวลาที่สามารถเสร็จสิ้นการทำงานถูกกำหนดให้กับงานทางธุรกิจแต่ละงานในกระแสงาน เพื่อรับมือกับปัญหาที่ซับซ้อนมากขึ้นของประสิทธิภาพด้านเวลาในกระแสงานกระบวนการทางธุรกิจ ในการจำลองพฤติกรรมของกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาจะถูกแปลงเป็นออโตมาตาที่กำหนดเวลาไว้ที่สอดคล้องกัน และจำลองโดยใช้เครื่องมือ UPPAAL วิทยานิพนธ์ฉบับนี้มีการเสนอชุดของกฎการแปลงเพื่อเป็นแนวทางในการแปลงของสัญลักษณ์ยอว์ลข้อจำกัดของช่วงเวลาให้อยู่ในรูปแบบของไทมด์ออโตมาตา และเสนอเว็บแอปพลิเคชันในการแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาเป็นออโตมาตา โดยผลลัพธ์ที่เป็นไทมด์ออโตมาตาจะถูกแปลงอย่างถูกต้อง และจำลองโดยใช้เครื่องมือจำลอง UPPAAL

สาขาวิชา	วิศวกรรมซอฟต์แวร์	ลายมือชื่อนิสิต .....
ปีการศึกษา	2565	ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

6272032321 : MAJOR SOFTWARE ENGINEERING
KEYWORD:

Naronggorn Wongsitthiphaithun : Transforming YAWL Workflows with Time Interval Constraints to Timed Automata. Advisor: Assoc. Prof. WIWAT VATANAWOOD, Ph.D.

YAWL is one of the modern business process workflows. It provides the visualization of easy-to-understand the workflows of business tasks with the single fixed time delay. In this thesis, the ordinary YAWL would be extended by adding the time interval constraints attaching to each task symbols in the YAWL workflows. By mean of the time interval constraints, the labels of both lower bound and upper bound of the time delays are assigned to each business tasks in the workflow to cope with more complicated problems of time efficiency in business process workflows. As to simulate the behaviors of the proposed YAWL workflows with time interval constraints, it would be converted into the corresponding timed automata and simulated using the UPPAAL simulation tool. In this thesis, a set of transformation rules is proposed to guide the mapping template of YAWL workflows with time interval constraints into timed automata syntaxes and web application is proposed to convert YAWL workflows with time interval constraints to timed automata. The resulting timed automata are correctly converted and simulated using UPPAAL simulation tool.

Field of Study: Software Engineering Student's Signature .....
Academic Year: 2022 Advisor's Signature .....

กิตติกรรมประกาศ

ผู้วิจัยขอกราบขอบพระคุณอาจารย์ที่ปรึกษาวิทยานิพนธ์ รศ.ดร. วิวัฒน์ วัฒนาวุฒิ ที่ให้คำปรึกษาในการจัดทำวิทยานิพนธ์เล่มนี้เป็นอย่างดีโดยเสียสละเวลาให้คำแนะนำ คำปรึกษาและแก้ไขข้อบกพร่องต่าง ๆ ซึ่งช่วยทำให้วิทยานิพนธ์สำเร็จลุล่วงไปได้ด้วยดี ผู้วิจัยขอจดจำทุกทำสอน ที่ได้รับจากอาจารย์และถ่ายทอดให้กับบุคคลอื่น ๆ ในโอกาสต่าง ๆ

ขอกราบขอบพระคุณคณะกรรมการทุกท่าน และคณาจารย์ภาควิศวกรรมคอมพิวเตอร์ที่เสียสละเวลาอันมีค่า ให้คำปรึกษาด้านวิชาการ คำเสนอแนะในการปรับปรุงวิทยานิพนธ์เพื่อให้วิทยานิพนธ์เล่มนี้ถูกต้องและครบถ้วน

สุดท้ายนี้ขอขอบคุณทุกท่านที่ได้กล่าวถึงและผู้ที่ไม่ได้เอ่ยนามช่วยให้กำลังใจ คำชี้แนะมาโดยตลอด ผลักดันให้ข้าพเจ้าผ่านพ้นอุปสรรคต่าง ๆ จนสามารถจัดทำวิทยานิพนธ์เล่มนี้เสร็จสมบูรณ์

ณรงค์กร วงศ์สิทธิไพฑูรย์

สารบัญ

หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหา

ในการดำเนินธุรกิจมีการจัดการกระบวนการทางธุรกิจ (Business process management : BPM) เพื่อให้เป็นไปตามผลที่คาดหวัง ซึ่งในปัจจุบันกระบวนการการทำงานมีความซับซ้อนมาขึ้น เช่น สถานพยาบาล สายงานการผลิต การขนส่ง การบิน การสั่งซื้อสินค้า เป็นต้น การที่จะมองเห็นกระบวนการทำงานทั้งหมดนั้นเป็นไปได้ยาก จึงมีภาษาสำหรับกำหนดกระบวนการทางธุรกิจเพื่อสนับสนุนการกำหนด และวิเคราะห์กระบวนการทางธุรกิจให้ง่ายยิ่งขึ้น มีการอธิบายกระแสงานเป็นโมเดล และสัญลักษณ์ต่าง ๆ เช่น กระแสงานยอว์ล (YAWL workflows) และไทมด์ออโตมาตา (Timed automata) เป็นต้น ทำให้สามารถทำงานได้อย่างมีแบบแผนมีระเบียบเป็นขั้นเป็นตอนมากขึ้น ลดขั้นตอน หรือเวลา การทำงานที่ไม่จำเป็นออกไป

งานวิจัยชิ้นนี้เลือกใช้กระแสงานยอว์ลจากเครื่องมือยอว์ลที่เป็นโปรแกรมโอเพ่นซอร์สช่วยในการจัดการกระบวนการทางธุรกิจ ซึ่งมีการใช้งานอย่างแพร่หลาย รูปแบบกระแสงานสัญลักษณ์ที่ง่ายต่อการเข้าใจ มีความคล่องตัวในการแก้ไขกระแสงานเมื่อเวลาผ่านไปโดยไม่กระทบโปรแกรม ถึงแม้กระแสงานยอว์ลนั้นจะมีระบบจับเวลา (Timer system) และตัวช่วยตรวจสอบคุณสมบัติก่อนใช้งานแต่ก็ยังไม่สามารถวิเคราะห์ได้ครอบคลุมถึงกรณีที่มีเงื่อนไขด้านเวลาแบบเป็นช่วงเวลา เพราะระบบจับเวลาเป็นการบอกค่าเฉลี่ย หรือการถ่วงเวลาของกระบวนการนั้น ๆ ทำให้กระแสงานยอว์ลนั้นไม่สามารถใช้ในการวิเคราะห์เกี่ยวกับแนวโน้มของระยะเวลาที่ใช้ได้ เช่น ในสายงานการผลิตกรณีมีระบบจับเวลาเป็นค่าเฉลี่ยของแต่ละงานจะสามารถวิเคราะห์เวลาของกระบวนการ เป็นค่าเฉลี่ยได้เท่านั้นทำให้ไม่สามารถทราบระยะเวลาน้อยสุด มากสุดของกระบวนการที่ทำได้ เป็นต้น

ดังนั้นงานวิจัยนี้มุ่งหวังที่จะเสนอการแปลงกระแสงานยอว์ลไปเป็นไทมด์ออโตมาตาเพื่อเพิ่มความสามารถในการวิเคราะห์ด้านเวลาแบบมีข้อจำกัดช่วงเวลา เช่น กระแสงานที่ออกแบบใช้เวลาน้อยสุด มากสุดเท่าไร เป็นต้น ทำให้สามารถรู้ความสามารถด้านเวลาของกระแสงานนั้น ๆ โดยการเพิ่มตัวแปรขอบเขตบน (Upper bound) และขอบล่าง (Lower bound) ของช่วงเวลาเข้าไปในส่วนของอะตอมมิกทาสก์ (Atomic task) จากกระแสงานยอว์ลและพัฒนาซอฟต์แวร์เพื่อรองรับการแปลงกระแสงานยอว์ลไปเป็นไทมด์ออโตมาตาแบบมีข้อจำกัดช่วงเวลาที่สามารถนำเข้าเครื่องมือ UPPAAL และทวนสอบด้านเวลาภายในเครื่องมือได้

1.2 วัตถุประสงค์งานวิจัย

1) สร้างกฎการแปลงกระแสงานยอว์ลไปเป็นไทมด์ออโตมาตาแบบมีข้อจำกัดช่วงเวลา
2) เพื่อออกแบบและพัฒนาเครื่องมือที่ใช้สำหรับการแปลงกระแสงานยอว์ลไปเป็นไทมด์ออโตมาตา

1.3 ขอบเขตงานวิจัย

1) แฟ้มเอกซ์เอ็มแอลของกระแสงานยอว์ลที่นำเข้าให้เป็นไปตามเครื่องมือยอว์ล
2) รองรับสัญลักษณ์กระแสงานยอว์ลดังนี้ Input condition, Output condition, Atomic task, AND-split, AND-join, XOR-split, XOR-join, OR-split และ OR-join
3) แฟ้มข้อจำกัดช่วงเวลาถูกนำเข้ามาแยกจากเอกซ์เอ็มแอลในข้อ 1 โดยเป็นหน่วยเวลาจำนวนเต็ม เช่น Task A มี Lower bound =2 และ Upper bound =5 สามารถเขียนได้อยู่ในรูป [2, 5]
4) แฟ้มผลลัพธ์เอกซ์เอ็มแอลของไทมด์ออโตมาตาสามารถเปิดได้โดยเครื่องมือ UPPAAL
5) ใช้สัญลักษณ์ใน UPPAAL ดังนี้ Location name, Guard, Synchronisation, Update, Invariant, Initial, Urgent locations และ Committed locations
6) พัฒนาซอฟต์แวร์ที่ช่วยในการแปลงกระแสงานยอว์ลไปเป็นไทมด์ออโตมาตา ที่มีขีดความสามารถดังนี้
    - สามารถรองรับแฟ้มเอกซ์เอ็มแอลนำเข้าของเครื่องมือยอว์ล
    - เครื่องมือรองรับข้อมูลข้อจำกัดช่วงเวลาในแต่ละอะตอมมิกทาสก์
    - สามารถทำการแปลงสัญลักษณ์ในแฟ้มเอกซ์เอ็มเครื่องมือยอว์ลเป็นแฟ้มเอกซ์เอ็มแอลไทมด์ออโตมาตาที่เปิดใช้งานใน UPPAAL ได้
7) ใช้กรณีศึกษา 3 กรณีที่ครอบคลุมกระแสงานยอว์ลที่ได้ระบุไว้
8) แฟ้มเอกซ์เอ็มแอลของกระแสงานยอว์ลที่นำเข้าต้องอยู่ในรูปแบบที่ถูกต้องแล้ว

1.4 ขั้นตอนการดำเนินการ

1) ศึกษาระบบการจัดการกระบวนการทางธุรกิจ, ไทมด์ออโตมาตา เพื่อทำความเข้าใจ และนำมาใช้ในงานวิจัย
2) ศึกษาไทมด์ออโตมาตาใน UPPAAL และกระแสงานยอว์ลจากเครื่องมือยอว์ล เพื่อทำความเข้าใจแล้วนำมาใช้ในการสร้างกฎการแปลง
3) ออกแบบ และสร้างวิธีการแปลงกระแสงานยอว์ลไปเป็นไทมด์ออโตมาตา และตรวจสอบความถูกต้อง
4) เปรียบเทียบโครงสร้างกระแสงานยอว์ล และไทมด์ออโตมาตาเพื่อใช้ในการพัฒนาเครื่องมือแปลงกระแสงานยอว์ลไป เป็นไทมด์ออโตมาตา
5) พัฒนาเครื่องมือแปลงกระแสงานยอว์ลไปเป็นไทมด์ออโตมาตา
6) ทดสอบความถูกต้องของเครื่องมือ
7) สรุปผลงานวิจัย และจัดทำเอกสาร

1.5 บทความที่ตีพิมพ์จากงานวิจัย

ส่วนหนึ่งของวิทยานิพนธ์นี้ ได้รับการตีพิมพ์เป็นบทความวิชาการ ดังนี้

1) เรื่อง “Transforming YAWL Workflows with Time Interval Constraints to Timed Automata” โดย ณรงค์กร วงศ์สิทธิไพฑูรย์ และ วิวัฒน์ วัฒนาวุฒิ ในงานประชุมวิชาการ 2022 19th International Joint Conference on Computer Science and Software Engineering (JCSSE) เมื่อวันที่ 22-25 มิถุนายน พ.ศ. 2565 ณ มหาวิทยาลัยศิลปากร กรุงเทพ ประเทศไทย

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 การจัดการกระบวนการธุรกิจ [1]

การจัดการกระบวนการทางธุรกิจเป็นแนวทางที่มีระเบียบวินัยในการระบุ ออกแบบ ดำเนินการ จัดทำเอกสาร วัดผล ตรวจสอบ และควบคุมกระบวนการทางธุรกิจทั้งแบบอัตโนมัติ และแบบไม่อัตโนมัติ เพื่อให้บรรลุเป้าหมายที่สอดคล้องกัน และสอดคล้องกับเป้าหมายเชิงกลยุทธ์ขององค์กร การจัดการกระบวนการธุรกิจเกี่ยวข้องกับ คำจำกัดความการปรับปรุงนวัตกรรม และการจัดการกระบวนการทางธุรกิจแบบครบวงจรที่รอบคอบ ร่วมมือกัน และใช้เทคโนโลยีช่วยมากขึ้นเรื่อย ๆ ที่ขับเคลื่อนผลลัพธ์ทางธุรกิจ สร้างมูลค่า และทำให้องค์กรสามารถบรรลุวัตถุประสงค์ทางธุรกิจด้วยความคล่องตัวมากขึ้น การจัดการกระบวนการทางธุรกิจช่วยให้องค์กรสามารถจัดกระบวนการทางธุรกิจให้สอดคล้องกับกลยุทธ์ทางธุรกิจซึ่งนำไปสู่ผลการปฏิบัติงานของบริษัทโดยรวมที่มีประสิทธิผล ผ่านการปรับปรุงกิจกรรมการทำงานเฉพาะทั้งภายในแผนกเฉพาะ ทั่วทั้งองค์กรหรือระหว่างองค์กร

2.1.1.1 ประเภทของการจัดการกระบวนการธุรกิจ [2]

ในองค์กรทุกองค์กรจะประกอบด้วยการจัดการกระบวนการทางธุรกิจซึ่งมีทั้งรูปแบบกระบวนการที่สามารถดำเนินการด้วยระบบอัตโนมัติทั้งหมด กระบวนการที่ระบบเข้ามาสนับสนุนมนุษย์ให้ทำงานได้อย่างมีประสิทธิภาพมากขึ้น และรูปแบบขั้นตอนที่ต้องอาศัยมนุษย์เป็นศูนย์กลางของกระบวนการโดยการจัดการกระบวนการทางธุรกิจทั้ง 3 รูปแบบ มีรายละเอียดดังนี้

1) เอกสารเป็นศูนย์กลางการจัดการกระบวนการทางธุรกิจในกระบวนการที่เน้นเอกสารเป็นศูนย์กลาง เป้าหมายของกระบวนการนี้คือการส่งเอกสารไปยังผู้อนุมัติหลายคนในกระแสงาน และให้แต่ละคนอนุมัติเอกสารนั้น การจัดการกระบวนการทางธุรกิจที่เน้นเอกสารเป็นศูนย์กลาง จะช่วยลดการส่งเอกสารทางอีเมลไปมาตัวอย่างเช่น กระบวนการอนุมัติงบประมาณการจัดการกระบวนการทางธุรกิจที่เน้นเอกสารเป็นศูนย์กลางผู้ริเริ่มโครงการกรอกแบบฟอร์มที่มีรายละเอียดทั้งหมดเกี่ยวกับคำของบประมาณ และส่งไปยังบุคคลที่เกี่ยวข้องตามกระแสงาน เป็นต้น

2) มนุษย์เป็นศูนย์กลางการจัดการกระบวนการทางธุรกิจ การจัดการกระบวนการ ทางธุรกิจที่ใช้มนุษย์เป็นศูนย์กลาง ขั้นตอนการทำงานจะถูกพิจารณาโดยมนุษย์เป็นหลัก และอาศัยการสนับสนุนจากฟังก์ชันอัตโนมัติต่าง ๆ กระบวนการเหล่านี้ต้องอาศัยมนุษย์ โดยที่ไม่สามารถใช้ระบบอัตโนมัติมาแทนที่ได้อย่างง่ายดายตัวอย่างเช่น การให้บริการลูกค้า และการจัดการข้อร้องเรียน การจัดการข้อร้องเรียนหรือให้บริการลูกค้าต้องอาศัยความเป็นมนุษย์ในการคิด และตัดสินใจเพื่อเชื่อมโยงเรื่องความรู้สึกเป็นกระบวนการที่ยากที่จะใช้ระบบอัตโนมัติเข้ามาทดแทน เป็นต้น

3) ระบบเป็นศูนย์กลางการจัดการกระบวนการทางธุรกิจ การจัดการกระบวนการ ทางธุรกิจที่ใช้ระบบเป็นศูนย์กลาง จะนิยมสำหรับการจัดการธุรกรรมของลูกค้า ข้อมูล หรือขั้นตอนการดำเนินการบางอย่างที่ธุรกิจคาดการณ์ได้ และใช้การเชื่อมโยงข้อมูล ผ่านการเข้าถึงของแต่ละซอฟต์แวร์เข้ามาผสานที่ระบบศูนย์กลาง ตัวอย่าง

ของระบบที่ธุรกิจมีอยู่ เช่น ซอฟต์แวร์บริหารทรัพยากรบุคคล (Human resource management system : HRMS) , ระบบการจัดการลูกค้าสัมพันธ์ (Customer relationship management : CRM) หรือการบริหารความเสี่ยงองค์กร (Enterprise risk management : ERM) เป็นต้น

2.1.1.2 วงจรชีวิตของการจัดการกระบวนการทางธุรกิจ [2]

1) ออกแบบ : ขั้นตอนแรกของการจัดการกระบวนการทางธุรกิจ คือ การออกแบบกระบวนการทั้งหมด ในขั้นตอนนี้ผู้ออกแบบควรเข้ามาศึกษากฎหรือรูปแบบของธุรกิจในปัจจุบัน สัมภาษณ์ผู้มีส่วนได้เสียต่าง ๆ พร้อมทั้งหารือเกี่ยวกับผลลัพธ์ที่ฝ่ายบริหารต้องการ

2) สร้างแบบจำลอง : การสร้างแบบจำลองกระบวนการทางธุรกิจ ในขั้นตอนนี้จะมีรายละเอียด ขั้นตอนการดำเนินการในธุรกิจ ณ ปัจจุบัน นำมาศึกษาว่าขั้นตอนใดจำเป็น ขั้นตอนใดไม่จำเป็น ขั้นตอนใดสามารถ เพิ่มประสิทธิภาพได้ เพื่อนำกระบวนการเหล่านี้ไปสร้าง แบบจำลองการจัดการกระบวนการทางธุรกิจในอนาคต

3) การดำเนินการของกระบวนการ : ในขั้นตอนที่ 3 ของการติดตั้งการจัดการกระบวนการทางธุรกิจ คือ การนำแบบจำลองที่สร้างไปทดลองใช้งานกับธุรกิจในขั้นทดลองควรทดสอบ กระบวนการทางธุรกิจให้ครบทั้ง 3 รูปแบบ

4) การตรวจสอบ : ระหว่างที่กระบวนการทางธุรกิจดำเนินการคุณควรสร้างดัชนีชี้วัดผลงาน หรือความสำเร็จของงาน (Key performance indicators : KPIs) สำหรับการตรวจสอบ และติดตามการวัดผลเพื่อประเมินว่างานที่สำคัญ ดำเนินไปอย่างไรในช่วงของการทดสอบ

5) การเพิ่มประสิทธิภาพ : ผู้ออกแบบระบบจะพยายามปรับปรุงฟอร์มกระแสงาน และแก้ไขปัญหาคอขวดของธุรกิจ โดยอาศัยข้อมูลเชิงลึกจากขั้นตอนการตรวจสอบ เพื่อให้กระบวนการมีประสิทธิภาพมากยิ่งขึ้น

2.1.2 ไทมด์ออโตมาตา [3] [4]

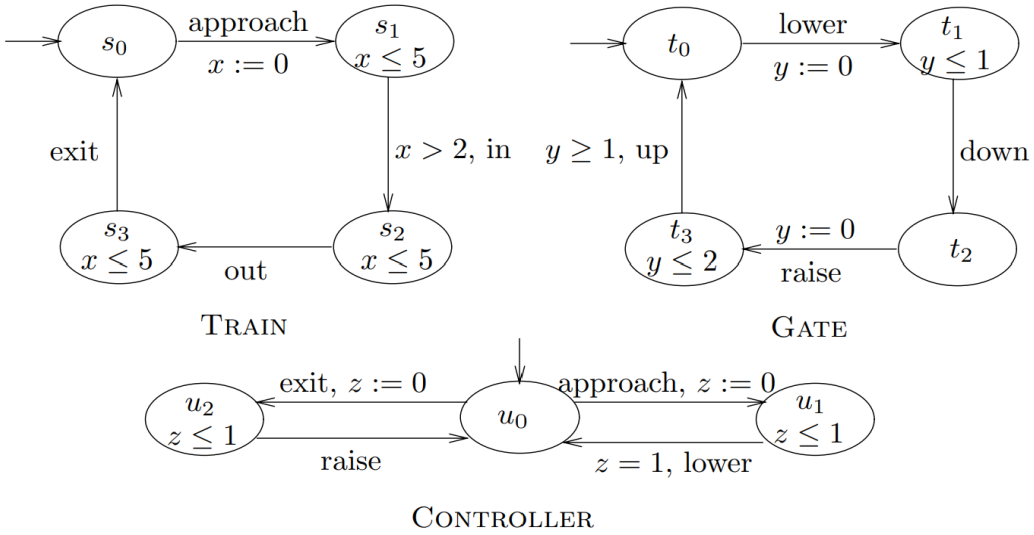
จากทฤษฎีออโตมาตา, ไทมด์ออโตมาตาเป็นออโตมาตาจำกัด (Finite automata) โดยเพิ่ม ในส่วนของนาฬิกาจริงเข้ามาระหว่างการทำงานของไทมด์ออโตมาตา ค่านาฬิกาจะเพิ่มขึ้นที่ความเร็วคงที่ตลอดช่วงเวลาการเปลี่ยนแปลงของออโตมาตาค่านาฬิกาเปรียบเทียบได้กับจำนวนเต็ม การเปรียบเทียบแบบนี้เป็นการป้องกันการให้สิทธิ์หรือไม่ให้สิทธิ์การเปลี่ยนแปลง และโดยการทำเช่นนี้จะเป็นการจำกัดพฤติกรรมของออโตมาตา นอกจากนี้ยังสามารถรีเซ็ตนาฬิกาได้ โดยไทมด์ออโตมาตาเป็นแบบหนึ่ง ของออโตมาตาแบบไฮบริด (Hybrid automata)

ไทมด์ออโตมาตาสามารถใช้ในการสร้างแบบจำลอง และวิเคราะห์พฤติกรรมของระบบคอมพิวเตอร์ เช่น ระบบหรือเครือข่ายเวลาจริง โดยมีเครื่องมือมากมายสำหรับการนำเข้า และวิเคราะห์ไทมด์ออโตมาตา รวมไปถึงอุปกรณ์ตรวจสอบโมเดล UPPAAL เครื่องมือนี้มีวุฒิภาวะมากขึ้นเรื่อย ๆ และไทมด์ออโตมาตา [5] เป็นทูเพิลที่มีองค์ประกอบดังนี้ $A=(\ L\ ,L_0\ ,\Sigma,X,I\ ,E\ )$ โดยสามารถอธิบายองค์ประกอบได้ดังตารางที่ 2.1

ตารางที่ 2.1 อธิบายองค์ประกอบของไทมด์ออโตมาตา [5]

ส่วนประกอบ	คำอธิบาย
L	เซตจำกัดของตำแหน่ง (Location)
L_0	ซับเซตของ L คือเซตของตำแหน่งเริ่มต้น
Σ	เซตจำกัดของป้ายกำกับ (Labels)
X	เซตจำกัดของนาฬิกา
I	ฟังก์ชันเงื่อนไขการอยู่ในสถานะโดยติดป้ายกำกับแต่ละตำแหน่งด้วยข้อจำกัดด้านนาฬิกาใน $\Phi(X)$ $I: L \longrightarrow \Phi(X)$
E	ซับเซตของ $L \times \Sigma \times \Phi(X) \times 2^X \times L$ คือเซตของการเปลี่ยนสถานะหรือสวิตช์ (Switches)

งานวิจัยนี้ขอเสนอตัวอย่างการเขียนไทมด์ออโตมาตาของการควบคุมไม้กั้นทางรถไฟที่เปิดปิดเพื่อข้ามทางรถไฟอัตโนมัติ [5] ระบบประกอบด้วยสามองค์ประกอบ : รถไฟ (Train) ไม้กั้นทางรถไฟ (Gate) และตัวควบคุมไม้กั้นทาง (Controller) ดังรูปที่ 2.1 ข้อกำหนดด้านความปลอดภัยที่ถูกต้องสำหรับระบบคือ เมื่อใดก็ตามที่รถไฟอยู่ภายใน ไม้กั้นทางควรปิด สิ่งนี้สอดคล้องกับการระบุว่าในทุก ๆ สถานะที่เข้าถึงได้ หากตำแหน่งของรถไฟคือ *s2* ตำแหน่งของไม้กั้นทางควรเป็น *t2* สังเกตว่าตำแหน่งดังกล่าวสามารถเข้าถึงได้ในกราฟผลิตภัณฑ์ (Product graph) ตัวอย่างเช่น มีเส้นทางจากตำแหน่งเริ่มต้น *(s0, t0, u0)* ถึง *(s1, t0, u1)* และจาก *(s1, t0, u1)* ถึง *(s2, t0, u1)* ซึ่งสอดคล้องกับสถานการณ์โดยที่เหตุการณ์ *approach* ที่เข้ามาจะถูกตามด้วยเหตุการณ์ *in* ทันที เนื่องจากผลิตภัณฑ์เป็นเพียงการดำเนินการแบบหลักไวยากรณ์ที่อธิบายตำแหน่งผลิตภัณฑ์ด้วยคำสันธานของค่าคงที่ และเส้นทางผลิตภัณฑ์ที่มีคำสันธานของเงื่อนไขที่เปิดใช้งานโดยไม่มีการวิเคราะห์ใด ๆ หากพิจารณาข้อมูลเวลาแล้วสามารถระบุได้ว่าเหตุการณ์ *approach* ไม่สามารถตามทันเหตุการณ์ *in* : ในตำแหน่ง *(s1, t0, u1)* ทั้งนาฬิกา *x* และ *z* มีค่าเท่ากัน ดังนั้นเหตุการณ์ *lower* ด้วยเงื่อนไข *z=1* รับประกันว่าเกิดก่อนเหตุการณ์รถไฟเข้าที่เงื่อนไข *x>2* ปัญหาการคำนวณในการตรวจสอบเวลาคือการทำการหักล้างดังกล่าวโดยการวิเคราะห์ข้อจำกัดด้านเวลา



รูปที่ 2.1 การทำงานของการควบคุมไม้กั้นทางรถไฟที่เปิดปิดเพื่อข้ามทางรถไฟอัตโนมัติ [5]

2.2. งานวิจัยที่เกี่ยวข้อง

ในงานวิจัยที่เกี่ยวข้องนั้นจะพูดถึง 4 งานวิจัยได้แก่ Timed Automata for Workflow Modeling and Analysis, Transforming YAWL Workflows with Time Constraints to Timed Automata, เครื่องมือ UPPAAL และกระแสงานยอว์ล ซึ่งถูกอธิบายดังต่อไปนี้

2.2.1 Timed Automata for Workflow Modeling and Analysis [6]

โดย Afsoon Soltani, 2014

เป็นงานวิจัยว่าด้วยการนำเสนอการเสริมสร้างกระแสงานของระบบการจัดการกระแสงาน (A Workflow Management System : WFMS) โดยการสร้างแบบจำลอง และการตรวจสอบแนวทางนี้มุ่งเน้นไปที่ประเด็นความถูกต้องโดยใช้ไทมด์ออโตมาตา โดยการกำหนดเส้นทางอธิบายขั้นตอนการควบคุมของแบบจำลองกระแสงานเป็นมุมมองที่เป็นทางการซึ่งแสดงเป็นชุดกิจกรรมกระบวนการที่เชื่อมโยงกัน เพื่อให้บรรลุเป้าหมายของกระแสงานในกรณีการกำหนดเส้นทางตามลำดับ (Sequential Routing) การกำหนดเส้นทางแบบขนาน (Parallel Routing) การกำหนดเส้นทางเลือกเส้นทาง (Selective Routing) แล้วนำมาทำการวิเคราะห์ผ่าน UPPAAL สิ่งที่นำมาใช้ในงานวิจัยนี้คือหลักการ และกฎเกณฑ์เบื้องต้นในการแปลงกระแสงานไปเป็นไทมด์ออโตมาตา ความแตกต่างจากงานของผู้วิจัยนี้คือ งานวิจัยนี้ทำการแปลงกระแสงานไปเป็นไทมด์ออโตมาตาซึ่งเป็นภาพรวมของระบบการจัดการกระแสงาน แต่งานวิจัยชิ้นนี้ ทำการนำเอากระแสงานยอว์ลซึ่งเป็นระบบการควบคุมกระบวนการทางธุรกิจที่มีความต่างกัน ในด้านของงาน การกำหนดเส้นทางที่มีการทำงานในตัวเอง และรูปแบบการกำหนดเส้นทางแบบผสม ทำการแปลงกระแสงานไปเป็นไทมด์ออโตมาตา

2.2.2 Transforming YAWL Workflows with Time Constraints to Timed Automata [7]

โดย Maruth-Ravibanjurdkul, 2021

เป็นงานวิจัยว่าด้วยการนำเสนอการขยายกระแสงานยอว์ลให้สามารถระบุเงื่อนไขด้านเวลาได้โดยผู้ทำวิจัยจะทำการเพิ่มค่าเฉลี่ยของเวลา (Mean Time) หรือ μ (Mu) เข้าไปในสัญลักษณ์กระแสงานยอว์ลมาตรฐานในส่วนงานย่อย จากนั้นจะนำเสนอวิธีการแปลงกระแสงานยอว์ลให้ไทมด์ออโตมาตา โดยขั้นตอนการทำวิจัยชิ้นนี้ผู้ทำวิจัยได้แบ่งออกเป็น 5 ขั้นตอน ดังนี้

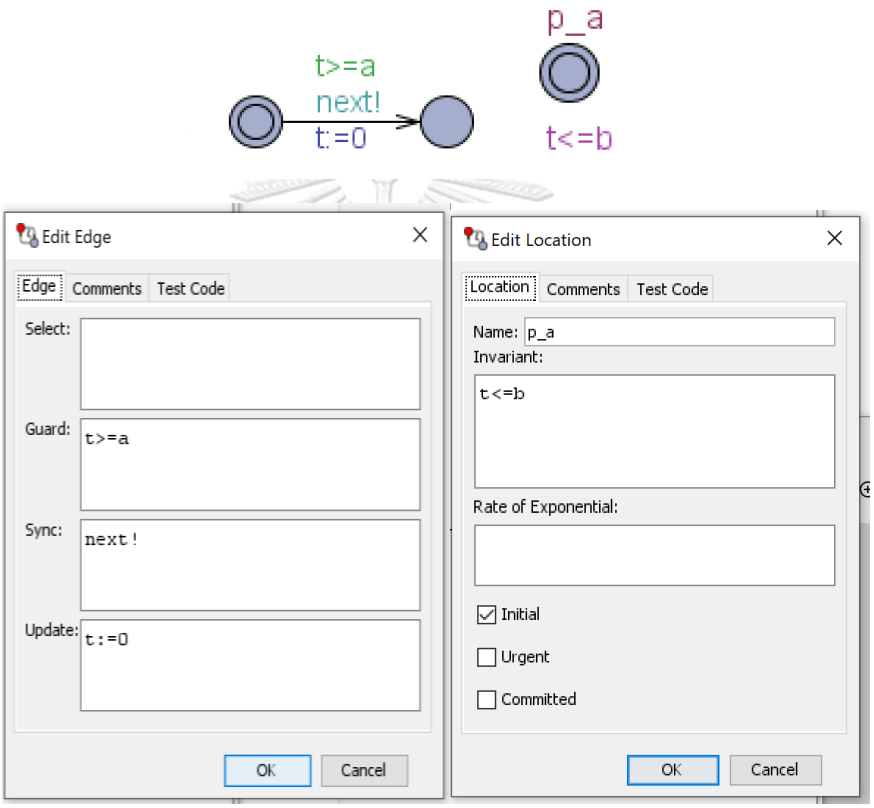
1) สร้าง และวิเคราะห์โครงสร้างแฟ้มกระแสงานยอว์ลด้วยเครื่องมือยอว์ล
2) ผู้ใช้งานระบุข้อจำกัดด้านเวลาในรูปแบบของค่าเฉลี่ยลงในส่วนของอะตอมมิกทาสก์
3) กำหนดกฎการแปลงจากกระแสงานยอว์ลไปเป็นไทมด์ออโตมาตา
4) สร้างเครื่องมือสำหรับแปลงกระแสงานยอว์ลไปเป็นไทมด์ออโตมาตาโดยอัตโนมัติ
5) ทวนสอบผลลัพธ์ที่ได้จากเครื่องมือที่สร้างขึ้นด้วยเครื่องมือ UPPAAL

โดยสิ่งที่นำมาใช้ในงานวิจัยคือหลักการ และกฎเกณฑ์การแยกองค์ประกอบย่อยของแฟ้มไทมด์ออโตมาตาและการปรับแต่งแฟ้มก่อนเข้าสู่เครื่องมือ UPPAAL โดยความแตกต่างจากงานของผู้วิจัยคือ งานวิจัยของคุณมารุตนี้ทำการแปลงกระแสงานยอว์ลไปเป็นไทมด์ออโตมาตา โดยนำเอาหลักการใส่ข้อจำกัดด้านเวลาโดยใช้วิธีการให้ผู้ใช้งานใส่ค่าเฉลี่ยของเวลา (Mean Time) แต่ยังเกิดความคลาดเคลื่อนของเวลาภายในแต่ละงานของกระแสงานอยู่ สำหรับงานวิจัยชิ้นนี้ทำการนำเอาหลักการใส่ข้อจำกัดด้านเวลา โดยใช้วิธีการให้ผู้ใช้งานใส่ค่าเวลาของขอบเขตบนสุด (Upper bound) และขอบเขตล่างสุด (Lower bound) ซึ่งแนวคิดโครงสร้างกฎการแปลง และข้อมูลนำเข้านั้นแตกต่างกัน

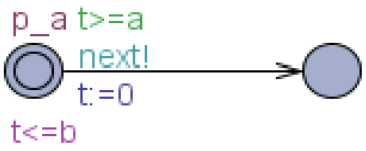
2.2.3 เครื่องมือ UPPAAL [4], [8]

UPPAAL เป็นกล่องเครื่องมือสำหรับตรวจสอบระบบเรียลไทม์ที่พัฒนาโดยมหาวิทยาลัยอุปซอลา และมหาวิทยาลัยอัลบอร์ก ได้ถูกนำไปใช้ในกรณีศึกษาตั้งแต่โพรโทคอลการสื่อสารไปจนถึงแอปพลิเคชันมัลติมีเดีย เครื่องมือได้รับการออกแบบมาเพื่อตรวจสอบระบบที่สามารถจำลองเป็นเครือข่ายของออโตมาตาที่กำหนดเวลาไว้ ซึ่งขยายด้วยตัวแปรจำนวนเต็ม ชนิดข้อมูลที่มีโครงสร้าง ฟังก์ชันที่ผู้ใช้กำหนด และการซิงโครไนซ์ช่องสัญญาณ UPPAAL เวอร์ชันแรกเปิดตัวในปี 1995 นับแต่นั้นมา มีการพัฒนาอย่างต่อเนื่อง โดยสัญลักษณ์ และองค์ประกอบที่นำมาใช้ในงานวิจัยนี้ คือเส้นเชื่อม (Edge) ซึ่งมีองค์ประกอบคือ (Guard, Sync, Update) ดังรูปที่ 2.2 ด้านซ้ายมือ ซึ่งมีข้อมูล Guard คือ *t>=a,* Sync คือ *next!*, Update คือ *t:=0* และตำแหน่ง (Location) ซึ่งมีองค์ประกอบคือ (Name, Invariant, Initial, Urgent, Committed) ดังรูปที่ 2.2 ด้านขวามือซึ่งมีข้อมูล Name คือ *p_a*, Invariant

คือ *t<=b* และกำหนดให้เป็นสถานะเริ่มต้น (Initial state) โดยรูปที่ 2.3 เป็นการรวมข้อมูลจากรูปที่ 2.2 ทั้ง เส้นเชื่อม และตำแหน่งเข้าด้วยกันให้ไทมด์ออโตมาตามีความสมบูรณ์มากขึ้น และมีการอธิบายพื้นฐานของไวยากรณ์ขององค์ประกอบดังตารางที่ 2.2 โดยตารางจะอธิบาย Guard, Sync, Update, Name, Invariant, Initial, Urgent และ Committed



รูปที่ 2.2 แสดงองค์ประกอบของเส้นเชื่อม และตำแหน่งของไทมด์ออโตมาตา UPPAAL



รูปที่ 2.3 แสดงองค์ประกอบพื้นฐานของไทมด์ออโตมาตาใน UPPAAL

ตารางที่ 2.2 คำจำกัดความพื้นฐานของไวยากรณ์ และความหมายสำหรับไทมด์ออโตมาตา [7], [4]

ชื่อ	สัญลักษณ์	คำอธิบาย
Guard	t>=a	เงื่อนไขที่ต้องตรวจสอบก่อนการเปลี่ยนแปลงสถานะ (State) ของ ออโตมาตา เงื่อนไขที่กำหนดนี้จะต้องเป็นจริงเท่านั้น การผ่านหรือการเปลี่ยนสถานะ (Transition) จึงจะเกิดขึ้น, Guard จะแสดงอยู่บนเส้นเชื่อมเสมอ
Synchronisation	next!	อยู่ในรูปแบบ ! หรือ ? เป็นช่องสัญญาไว้ทำการส่งและรับสัญญาณเพื่อให้เกิดการเปลี่ยนสถานะ
Update	t:=0	รายการนิพจน์ที่คั่นด้วยจุลภาคที่มีผลข้างเคียงนิพจน์ต้องอ้างอิงถึงนาฬิกา ตัวแปรจำนวนเต็ม และค่าคงที่เท่านั้น และกำหนดค่าจำนวนเต็มให้กับนาฬิกาเท่านั้น
Invariant	t<=b	เงื่อนไขที่ตรงตามนิพจน์ของ Invariant เท่านั้นถึงจะสามารถอยู่ในสถานะนี้ได้ ถ้าไม่ตรงจะไม่สามารถเปลี่ยนแปลงเข้ามายังสถานะนี้ได้ หรือถ้าอยู่ในสถานะนี้อยู่แล้วจะถูกบังคับเปลี่ยนแปลงสถานะ
Initial		สถานะเริ่มต้นของไทมด์ออโตมาตา
Urgent locations		มีความหมายเทียบเท่ากับการเพิ่มนาฬิกาพิเศษ x นั้น ถูกรีเซ็ตบนขอบขาเข้าทั้งหมด และมี Invariant x<=0 บนสถานะ จึงไม่ปล่อยให้เวลาผ่านไปเมื่อระบบอยู่ในสถานะเร่งด่วนและจะถูกบังคับเปลี่ยนแปลงสถานะทันที
Committed locations		เป็นการจำกัดการดำเนินการมากกว่า Urgent locations สถานะมีความมุ่งมั่นหากมีตำแหน่งใด ๆ ในสถานะที่กระทำตำแหน่งที่มุ่งมั่นจะล่าช้าไม่ได้ และการเปลี่ยนสถานะครั้งต่อไปต้องมีส่วนร่วมขาออกของตำแหน่งที่กำหนด อย่างน้อยหนึ่งแห่ง

โดยเครื่องมือ UPPAAL มีองค์ประกอบ 3 ส่วนหลักคือ Description language, Simulation, Model Checker ที่ทำงานร่วมกันในชื่อของ UPPAAL Engine ซึ่งอธิบายได้ดังตารางที่ 2.3 โดยสามารถออกแบบไทมด์ออโตมาตาผ่าน Description language จำลองการทำงานผ่าน Simulation และสามารถทวนสอบผ่าน Model checker ได้

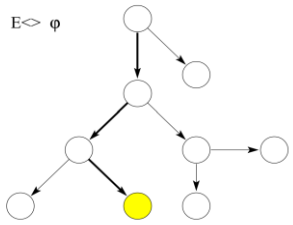
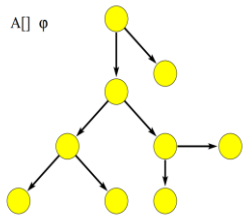
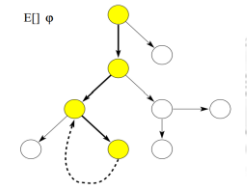
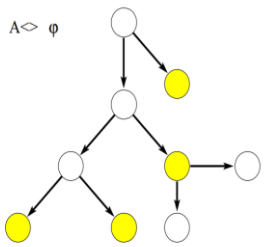
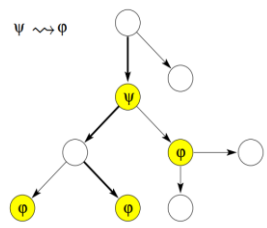
ตารางที่ 2.3 แสดงส่วนประกอบหลักของ UPPAAL [7], [4], [9]

ชื่อองค์ประกอบ	คำอธิบาย
Description language	ส่วนที่ทำหน้าที่อธิบายความหมายของตัวแปรต่าง ๆ ทำงานโดยการ เป็นแบบจำลองหรือภาษาที่ถูกออกแบบไว้ เพื่ออธิบายพฤติกรรมของระบบเครือข่ายออโตมาตา
Simulation	ส่วนที่ทำหน้าที่เป็นเครื่องมือสำหรับทดสอบรูปแบบการประมวลผลแบบไดนามิกในระหว่างการออกแบบ หรือในระหว่างการสร้างแบบจำลองเพื่อลดโอกาสผิดพลาดลงก่อนที่จะเข้าสู่ส่วนของ Model Checker
Model checker	ส่วนที่ทำหน้าที่ตรวจสอบเงื่อนไขช่วงเวลา (Invariant), คุณสมบัติการเข้าถึง (Reachability) ในรูปแบบของแบบจำลองปริภูมิสถานะ (State space)

การทวนสอบคุณสมบัติของ UPPAAL ผ่าน Model checker จะกระทำในรูปแบบของแบบสอบถาม (Query) ของตรรกศาสตร์ต้นไม้การคำนวณเวลา (Timed computation tree logic : TCTL) ผ่านตัวตรวจสอบโมเดล (Model checker) โดยตัวดำเนินการของตรรกศาสตร์ต้นไม้การคำนวณเวลาประกอบไปด้วยคือ ประพจน์ (Proposition) เช่น ϕ, ψ, φ ตัวดำเนินการลอจิก (Logical operators) [10] ตัวดำเนินการเชิงตรรกะคือตัวดำเนินการปกติ : ¬ นิเสธ (Not), ∨ หรือ (Or), ∧ และ (And), ⇒ ถ้าแล้ว (If..then) และ ⇔ ก็ต่อเมื่อ (If and only if) และตัวดำเนินการเชิงเวลา (Temporal operators) โดยประกอบด้วย Aϕ – ทั้งหมด (All) : ϕ ต้องยึดทุกเส้นทางที่เริ่มต้นจากสถานะปัจจุบัน Eϕ – มีอยู่ (Exists) : มีอย่างน้อยหนึ่งเส้นทางที่เริ่มต้นจากสถานะปัจจุบันที่ ϕ อยู่ และตัวระบุเฉพาะเส้นทาง Xϕ – ถัดไป (Next) : ϕ ต้องอยู่ในสถานะถัดไป (บางครั้งตัวดำเนินการนี้จะถูกระบุ N แทนที่จะเป็น X) Gϕ – ทั่วโลก (Globally) : ϕ ต้องยึดเส้นทางที่ตามมาทั้งหมด Fϕ – ในที่สุด (Finally) : ในที่สุด ϕ ก็ต้องอยู่ในสถานะ (ที่ใดสักแห่งบนเส้นทางที่ตามมา) ϕ U ψ – จนถึง (Until) : ϕ ต้องถือไว้อย่างน้อยก็จนกว่าจะถึงตำแหน่งที่ ψ ถืออยู่ ซึ่งหมายความว่า ψ จะได้รับการยืนยันในอนาคต ϕ W ψ – จนถึงแบบไม่แข็งแรง (Weak until) : ϕ ต้องถือไว้จนกว่า ψ จะคงอยู่ ความแตกต่างกับ U คือไม่มีการรับประกันว่า ψ จะได้รับการยืนยัน ตัว

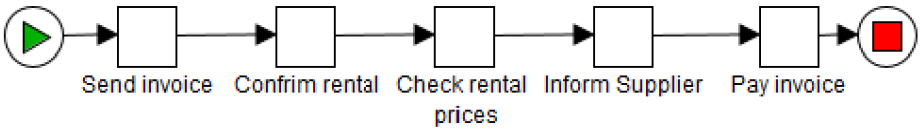
ดำเนินการ W บางครั้งเรียกว่า "เว้นแต่" (unless) โดยมีสัญลักษณ์ และอธิบายเพิ่มเติมได้ดังตารางที่ 2.4 โดยในตารางนี้จะอธิบายตรรกศาสตร์ต้นไม้การคำนวณเวลาที่เครื่องมือ UPPAAL รองรับ

ตารางที่ 2.4 อธิบายคุณสมบัติเบื้องต้นของตรรกศาสตร์ต้นไม้การคำนวณเวลาเพื่อการทวนสอบแบบจำลอง [7], [4]

คุณสมบัติ	คำอธิบาย
 คุณสมบัติการเข้าถึง (Reachability properties)	เพื่อยืนยันว่าสถานะที่กำหนดนั้นสามารถเป็นไปได้หรือไม่ ทำได้โดยการตรวจสอบว่ามีอย่างน้อยหนึ่งเส้นทาง (Path) ที่ไปถึงตำแหน่งที่ต้องการ สามารถเขียนในรูปของตรรกศาสตร์ต้นไม้การคำนวณเวลาได้ดังนี้ E<>**φ**
  คุณสมบัติความปลอดภัย (Safety properties)	เป็นการทวนสอบโดยยืนยันว่าในทุก ๆ เส้นทางที่มีตำแหน่ง ตรงตามต้องการ สามารถเขียนในรูปของตรรกศาสตร์ต้นไม้การคำนวณเวลาได้ดังนี้ A[]**φ** และ E[]**φ**
  คุณสมบัติการคงอยู่ (Liveness properties)	ในทุก ๆ เส้นทางจะนำไปสู่ตำแหน่ง **φ** ที่ต้องการทวนสอบ สามารถเขียนในรูปของตรรกศาสตร์ต้นไม้การคำนวณเวลาได้ดังนี้ A<>**φ**

2.2.4 กระแสงานยอว์ล [11] [12]

กระแสงานยอว์ลเป็นหนึ่งในกระแสงานที่สามารถอธิบายการกระบวนการทางธุรกิจได้ซึ่งเครื่องมือยอว์ลประกอบด้วยตัวแก้ไขสำหรับคำจำกัดความของข้อกำหนดเฉพาะของกระบวนการ และเอ็นจิ้น (Engine) สำหรับการดำเนินการข้อกำหนดเฉพาะของกระบวนการประกอบด้วยตัวแทนของกระบวนการจริงทั้งหมดที่จำเป็นสำหรับกระบวนการอัตโนมัติของกระบวนการขั้นตอนการควบคุม ทรัพยากร และข้อมูล โดยกระบวนการทางธุรกิจถูกอธิบายเป็นลำดับของงานในรูปแบบกราฟิก ดังแสดงในรูปที่ 2.4 ผ่านเครื่องมือยอว์ลซึ่งเป็นกระแสงานยอว์ลเกี่ยวกับการเรียกเก็บเงินของผู้ผลิต เริ่มต้นด้วยผู้ผลิตทำการส่งใบแจ้งหนี้ วิศวกรสามารถแก้ไขระยะเวลาการเช่าในงาน *“Confirm rental”* หลังจากขั้นตอนนั้น เจ้าหน้าที่คลังสินค้าจะตรวจสอบใบแจ้งหนี้ด้วย และสามารถแก้ไขราคาได้ ถัดไปผู้ผลิตจะได้รับแจ้งเกี่ยวกับใบแจ้งหนี้ที่แก้ไขแล้ว และฝ่ายการเงินจะชำระเงินตามใบแจ้งหนี้

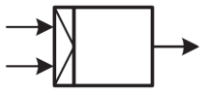
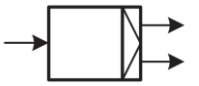
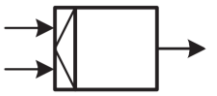
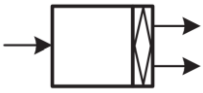


รูปที่ 2.4 ตัวอย่างกระบวนการจากเครื่องมือยอว์ล

2.2.4.1 สัญลักษณ์กระแสงานยอว์ล [7], [13]

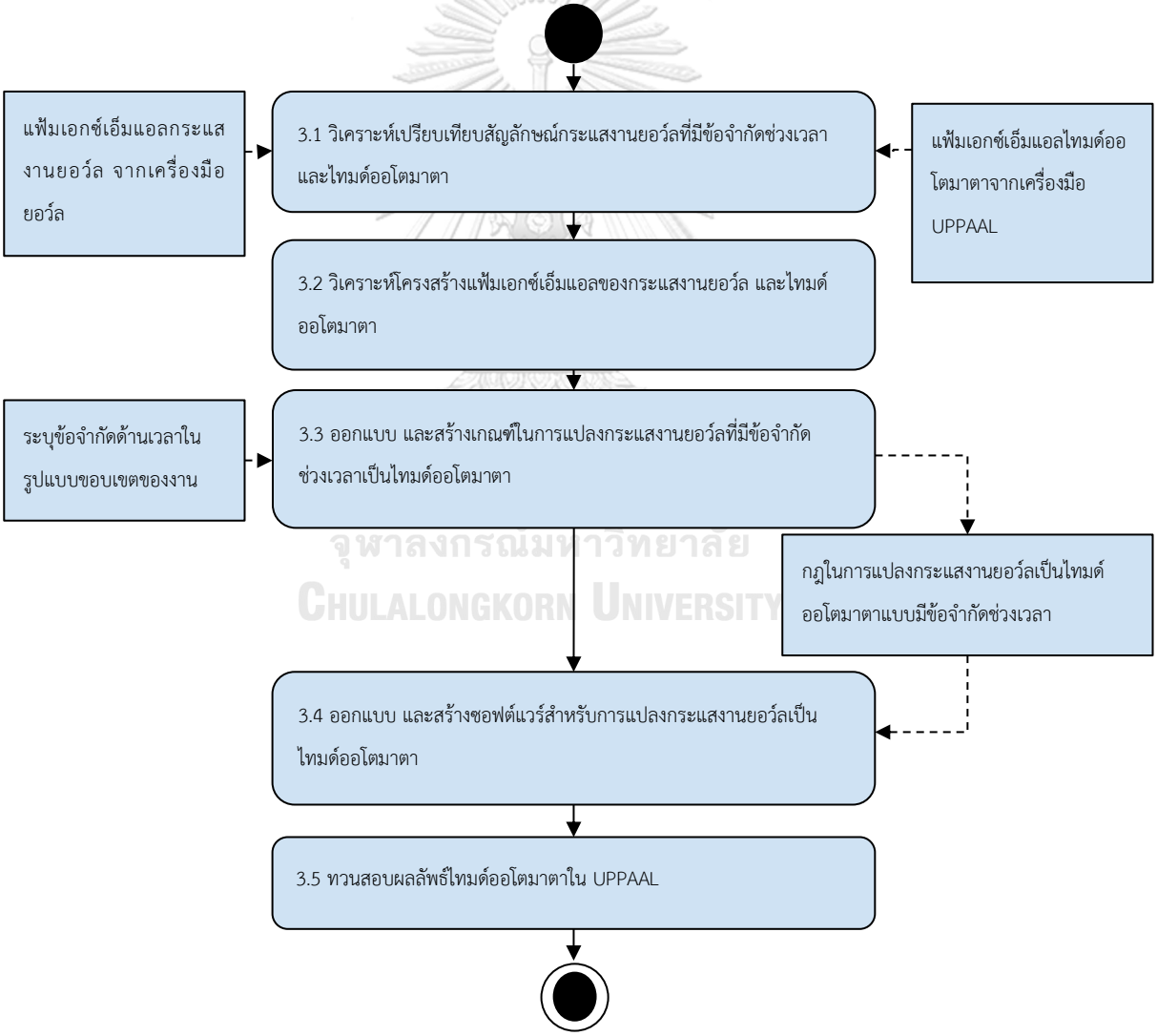
ในโปรแกรมแก้ไขกระแสงานยอว์ล ตัวแก้ไขกระแสงานยอว์ลมีสัญลักษณ์ที่ใช้ในงานวิจัยดังนี้ Input condition, Output condition, Atomic task, AND-split task, AND-join task, XOR-split task, XOR-join task, OR-split task และ OR-join task โดยถูกอธิบายดังตารางที่ 2.5 อธิบายตรรกะของสัญลักษณ์กระแสงานยอว์ล

ตารางที่ 2.5 สัญลักษณ์ที่ใช้อธิบายกระบวนการทำงานของกระแสงานยอว์ลมีดังต่อไปนี้ [7], [14]

ชื่อ	สัญลักษณ์	คำอธิบาย
Input condition		จุดเริ่มต้นกระแสงานยอว์ล
Output condition		จุดสิ้นสุดกระแสงานยอว์ล
Atomic task		ระบบงานที่ดำเนินงาน 1 งาน
AND-split task		ส่งออกกระแสงานพร้อมกันทุกกระแสงานหลังจบกระแสงาน
AND-join task		เริ่มกระแสงานต่อเมื่อมีกระแสงานเข้ามาครบทุกกระแสงาน
XOR-split task		เลือกส่งออกกระแสงาน 1 กระแสงานหลังจบกระแสงาน
XOR-join task		เริ่มกระแสงานทันทีที่มีกระแสงานเข้ามา 1 กระแสงาน
OR-split task		ส่งออกกระแสงานทั้งหมดหรือบางกระแสงานหลังจบกระแสงานปัจจุบัน
OR-join task		เริ่มกระแสงานเมื่อมีกระแสงานเข้ามาบางกระแสงานหรือเข้ามาทั้งหมด

บทที่ 3
กระบวนการแปลงกระแสงานยอว์ลไปเป็นไทมด์ออโตมาตา

บทนี้จะนำเสนอกระบวนการแปลงกระแสงานยอว์ลไปเป็นไทมด์ออโตมาตาอธิบายภาพรวมของการทำวิจัยการแปลงจากกระแสงานยอว์ลไปเป็นไทมด์ออโตมาตา โดยมีการเพิ่มข้อจำกัดช่วงเวลา ซึ่งแรกสุดจะนำเสนอภาพรวมของงานวิจัยดังรูปที่ 3.1 โดยมีขั้นตอนดังนี้ วิเคราะห์เปรียบเทียบสัญลักษณ์กระแสงานยอว์ล และไทมด์ออโตมาตา หลังจากนั้นวิเคราะห์โครงสร้างแฟ้มเอกซ์เอ็มแอลของกระแสงานยอว์ล และไทมด์ออโตมาตาเพื่อนำมาออกแบบเกณฑ์ในการแปลง และแอปพลิเคชันต่อไป ในท้ายที่สุดนั้นจะนำเอาผลลัพธ์ที่ได้จากการแปลงเข้าไปทวนสอบในเครื่องมือ UPPAAL โดยอธิบายรายละเอียดของแต่ละกิจกรรมต่อไป



รูป 3.1 ภาพรวมของกระบวนการวิจัย

3.1 การวิเคราะห์เปรียบเทียบสัญลักษณ์กระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลา และไทมด์ออโตมาตา

ในกระบวนการนี้ ขั้นแรกสุดนิยามกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาเพื่อให้กระแสงานยอว์ลเดิมสามารถระบุข้อจำกัดช่วงเวลาได้ และนำมาวิเคราะห์ว่าจะแปลงไปเป็นส่วนของแผนภาพไทมด์ออโตมาตาได้รูปแบบใด หลังจากนั้นผู้วิจัยได้ลองออกแบบ และสร้างกระแสงานยอว์ลแล้วนำไปลองออกแบบไทมด์ออโตมาตาว่าควรมีลักษณ์เป็นอย่างไรให้ได้การทำงานมีสอดคล้องกัน โดยใช้แนวคิดให้แต่ละงานในกระแสงานยอว์ลเป็น 1 ไทมด์ออโตมาตาที่มีพฤติกรรมเหมือนกัน เพื่อศึกษาโครงสร้างในการแปลงต่อไป

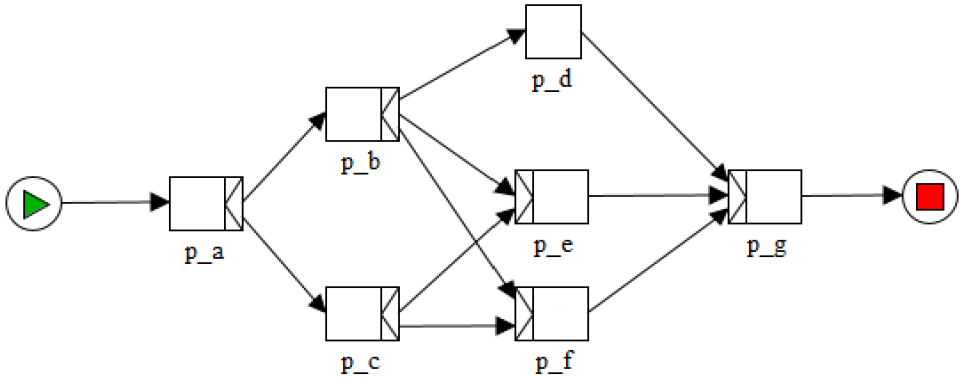
3.1.1 การกำหนดนิยามของกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลา

นิยามกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลา มี 8 ทูเพิล $YWL = (C, T, C_i, C_o, TD, K_{max}, K_{min}, A)$ ดังตารางที่ 3.1 โดยในตารางจะอธิบายส่วนประกอบ และคำอธิบายของ $C, T, C_i, C_o, TD, K_{max}, K_{min}$ และ A โดย K_{max}, K_{min} จะเป็นส่วนของข้อจำกัดช่วงเวลาที่ถูกเพิ่มเข้ามาในกระแสงานยอว์ล

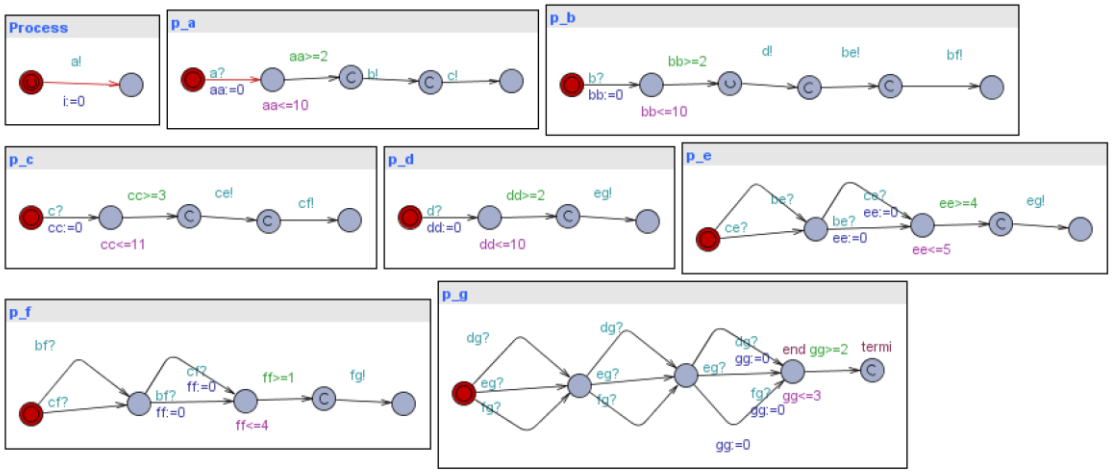
ตารางที่ 3.1 อธิบายองค์ประกอบของกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลา

ส่วนประกอบ	คำอธิบาย
C	เซตของเงื่อนไข (Conditions)
T	เซตของงาน (Atomic task)
C_i	เงื่อนไขในการนำเข้า (Input Conditions) $C_i \in C$
C_o	เงื่อนไขในการนำออก (Output Conditions) $C_o \in C$
TD	ฟังก์ชันที่ใช้ประดับเพื่อแสดงพฤติกรรมของงาน $TD : T \rightarrow \{$ *AND-Split, OR-Split, XOR-Split, AND-Join, OR-Join, XOR-Join}*
K_{max}	ฟังก์ชันที่อธิบายช่วงเวลาจำกัดขอบเขตบนของงาน T โดยที่ $N+$ เป็นจำนวนเต็มบวก $K_{max} : T \rightarrow N^+$
K_{min}	ฟังก์ชันที่อธิบายช่วงเวลาจำกัดขอบเขตล่างของงาน T โดยที่ $N+$ เป็นจำนวนเต็มบวก $K_{min} : T \rightarrow N^+$
A	เซตของการเชื่อมต่อ (x,y) โดยที่ $x,y \in (C \cup T)$ และ $A \subseteq (CxT) \cup (TxC)$

หลังจากนิยามกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาแล้ว จะทำการสร้างกระแสงานยอว์ลเพื่อศึกษาการทำงานเทียบกับไทมด์ออโตมาตา ดังรูป 3.2 จำลองการทำงานแบบขนานของกระแสงานยอว์ล โดยหลังจากจบการทำงานของงาน p_a แล้วจะส่งผ่านไปทำทั้งงาน p_b และงาน p_c ขนานกัน โดยที่งาน p_g จะทำได้ก็ต่อเมื่องาน p_d, p_e และ p_f ต้องเสร็จสิ้นแล้ว หลังจากที่ได้กระแสงานยอว์ลแล้วผู้วิจัยจึงนำไปเปรียบเทียบกับไทมด์ออโตมาตาว่าถ้ากรณีการทำงานเหมือนกันนั้นควรมีลักษณะเป็นอย่างไรได้ผลดังรูปที่ 3.3 โดยใช้แนวคิดในการออกแบบนั้นจะมองให้แต่ละงานในกระบวนการยอว์ลเป็น 1 ไทมด์ออโตมาตา ผลของการวิเคราะห์เปรียบเทียบสัญลักษณ์กระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลา และไทมด์ออโตมาตาจะได้ความสัมพันธ์เบื้องต้นระหว่างกระแสงานยอว์ลกับไทมด์ออโตมาตาเพื่อใช้เป็นส่วนหนึ่งในการออกแบบกฎการแปลงต่อไป



รูป 3.2 ตัวอย่างกระแสงานยอว์ลที่ได้จากการใช้เครื่องมือยอว์ล

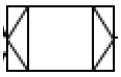


รูปที่ 3.3 ตัวอย่างไทมด์ออโตมาตาที่ได้จากการใช้เครื่องมือ UPPAAL

3.2 การวิเคราะห์โครงสร้างแฟ้มเอกซ์เอ็มแอลของกระแสงานยอว์ล และแฟ้มเอกซ์เอ็มแอลของไทมด์ออโตมาตา

เพื่อที่จะแปลงกระแสงานยอว์ลไปเป็นไทมด์ออโตมาตา ในกระบวนการนี้จำเป็นจะต้องวิเคราะห์โครงสร้างแฟ้มเอกซ์เอ็มแอลของกระแสงานยอว์ล และไทมด์ออโตมาตา แยกองค์ประกอบของแท็ก และคุณลักษณะต่าง ๆ เพื่อศึกษาว่าในแต่ละสัญลักษณ์ของกระแสงานยอว์ล และไทมด์ออโตมาตาสามารถสร้างขึ้นมาได้อย่างไร และสามารถดึงข้อมูลจากแฟ้มเอกซ์เอ็มแอลได้อย่างถูกต้อง เพื่อใช้ในการออกแบบ และสร้างโปรแกรมในการแปลงต่อไป โดยเริ่มแรกนั้นจะศึกษาความสัมพันธ์ระหว่างสัญลักษณ์กระแสงานยอว์ล และแท็กเอกซ์เอ็มแอลก่อน หลังจากนั้นจะเป็นส่วนของไทมด์ออโตมาตา โดยตารางที่ 3.2 จะเป็นการเปรียบเทียบแท็กเอกซ์เอ็มแอล และสัญลักษณ์ของกระแสงานยอว์ลว่ามีความสอดคล้องกันอย่างไร

ตารางที่ 3.2 ตัวอย่างความสัมพันธ์ระหว่างสัญลักษณ์กระแสงานยอว์ล และแท็กเอกซ์เอ็มแอล

สัญลักษณ์	แท็กเอกซ์เอ็มแอล	คำอธิบาย
Task 	<task id="p_a"> </task>	หน่วยงานต่าง ๆ ที่เป็นส่วนประกอบของกระบวนการ
	<name>p_a</name>	ชื่อของงานนั้น ๆ
	<flowsInto> <nextElementRef id="p_c" /> </flowsInto> <flowsInto><nextElementRef id="p_b" /></flowsInto>	ระบุงานถัดไปของกระแสงานนั้น ๆ
Join and split type 	<join code="xor" /> <split code="and" />	ระบุประเภทของ join และ split ของงานนั้น ๆ
Input Condition 	<inputCondition id="InputCondition"> <flowsInto> <nextElementRef id="p_a" /> </flowsInto> </inputCondition>	จุดเริ่มต้นของกระแสงานยอว์ล
Output Condition 	<outputCondition id="OutputCondition" />	จุดสิ้นสุดของกระแสงานยอว์ล

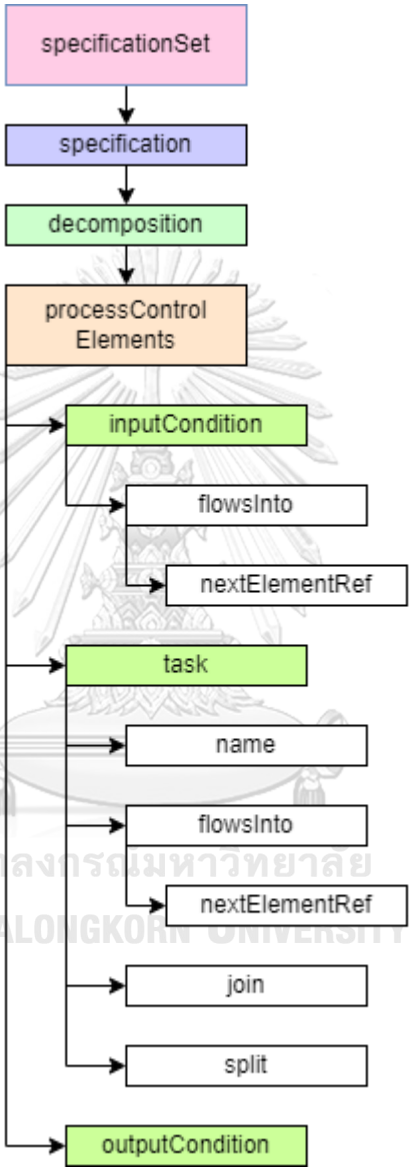
โดยตารางที่ 3.2 นั้นถูกแยก และวิเคราะห์มาจากแฟ้มเอกซ์เอ็มแอลของกระแสงานยอว์ลของรูปที่ 3.2 โดยแสดงเฉพาะงาน InputCondition, p_a และ p_b ดังรูปที่ 3.4 โดยในรูปนั้นจะแสดงข้อมูลรายละเอียดของแท็ก <inputCondition>, <task id=”p_a”> และ <task id=”p_b”> ว่ามีการเชื่อมต่อกับงานใดบ้าง มี *TD* การ join และ split แบบใด

```
<?xml version="1.0" encoding="UTF-8"?>
<specificationSet xmlns="http://www.yawlfoundation.org/yawlschema" xmlns:
  <specification uri="thesis_and">
    <documentation>No description provided</documentation>
    <metaData>
      <creator>Naronggorn</creator>
      <description>No description provided</description>
      <coverage>4.3.1.772</coverage>
      <version>0.1</version>
      <persistent>false</persistent>
      <identifier>UID_bf0f2188-c186-4d20-8ad9-92f1dfebfce6</identifier>
    </metaData>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" />
    <decomposition id="Net" isRootNet="true" xsi:type="NetFactsType">
      <processControlElements>
        <inputCondition id="InputCondition">
          <flowsInto>
            <nextElementRef id="p_a" />
          </flowsInto>
        </inputCondition>
        <task id="p_a">
          <name>p_a</name>
          <flowsInto>
            <nextElementRef id="p_c" />
          </flowsInto>
          <flowsInto>
            <nextElementRef id="p_b" />
          </flowsInto>
          <join code="xor" />
          <split code="and" />
          <resourcing>
            <offer initiator="user" />
            <allocate initiator="user" />
            <start initiator="user" />
          </resourcing>
        </task>
        <task id="p_b">
          <name>p_b</name>
          <flowsInto>
            <nextElementRef id="p_d" />
          </flowsInto>
          <flowsInto>
            <nextElementRef id="p_e" />
```

รูปที่ 3.4 ตัวอย่างเอกซ์เอ็มแอลของกระแสงานยอว์ล

จากความสัมพันธ์ระหว่างสัญลักษณ์กระแสงานยอว์ลและแท็กเอกซ์เอ็มแอลทำให้เราได้ทราบแท็กที่จำเป็นในการใช้แปลงเป็นไทม์ออโตมาตาเรียบร้อยแล้ว ทำการสรุปโครงสร้างของแท็กโดยสามารถอธิบายเป็นโครงสร้างที่

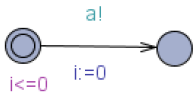
จำเป็นได้ดังรูปที่ 3.5 ซึ่งจะอธิบายถึงส่วนที่เราสนใจใช้ในการแปลงคือ แท็ก inputCondition, task ต่าง ๆ และ outputCondition ซึ่งอยู่ภายใต้แท็ก processControlElements และมีแท็กลูกต่าง ๆ ที่เกี่ยวข้อง เช่น flowInto เป็นข้อมูลที่ใช้บอกว่าหลังจากเสร็จสิ้นงานนั้นแล้วจะถูกส่งต่อไปที่งานใด join เป็นแท็กที่ใช้บอกว่าตรรกะของงานนั้น ๆ ฝั่งขาเข้าเป็นรูปแบบใด



รูปที่ 3.5 แสดงโครงสร้างเอกซ์เอ็มแอลที่จำเป็นในการแปลงของกระแสงานยอว์ล

หลังจากวิเคราะห์โครงสร้างเอกซ์เอ็มแอลของกระแสงานยอว์ลแล้ว ต่อไปทำการวิเคราะห์โครงสร้างเอกซ์เอ็มแอลของไทมด์ออโตมาตา โดยตารางที่ 3.3 จะเป็นการเปรียบเทียบแท็กเอกซ์เอ็มแอล และสัญลักษณ์ของไทมด์ออโตมาตา โดยจะศึกษาจากไทมด์ออโตมาตาที่มี 2 ตำแหน่ง ตำแหน่งแรกมี Invariant คือ *i<=0* และเส้นเชื่อมระหว่างตำแหน่งแรก และสองนั้นประกอบไปด้วย Synchronization คือ *a!* และ Assignment คือ *i:=0*

ตารางที่ 3.3 ความสัมพันธ์ระหว่างสัญลักษณ์**ไทมด์**ออโตมาตา และแท็กเอกซ์เอ็มแอล

สัญลักษณ์	แท็กเอกซ์เอ็มแอล	คำอธิบาย
	<template> </template>	ระบุข้อมูลของ 1 ไทมด์ออโตมาตา
	<name>start</name>	ชื่อไทมด์ออโตมาตา
	<location id="id0" x="0" y="0"> <label kind="invariant" x="-10" y="17">i<=0</label> </location>	ตำแหน่งของสัญลักษณ์ต่าง ๆ ในหน้าจอแสดงผลของเครื่องมือ UPPAAL ซึ่งมีการระบุ invariant
	<init ref =”id0”/>	ตำแหน่งเริ่มต้นของไทมด์ออโตมาตา
	<transition> <source ref="id0"/> <target ref="id1"/> <label kind="synchronisation" x="42" y="-32">a!</label> <label kind="assignment" x="34" y="9">i:=0</label> </transition>	ระบุการเปลี่ยนแปลงในกระบวนการนั้น ๆ เช่น การกำหนดค่าการซิงโครไนซ์เงื่อนไขต่าง ๆ เป็นต้น
	<declaration>clock i;</declaration>	การประกาศตัวแปรของไทมด์ออโตมาตา

โดยตารางที่ 3.3 ถูกวิเคราะห์มาจากตัวอย่างเอกซ์เอ็มแอลของไทมด์ออโตมาตาจากรูปที่ 3.6 ซึ่งมีการแบ่งไทมด์ออโตมาตาเป็นแต่ละไทมด์ออโตมาตอนโดยใช้แท็ก template และภายในแต่ละไทมด์ออโตมาตอนนั้นจะมี

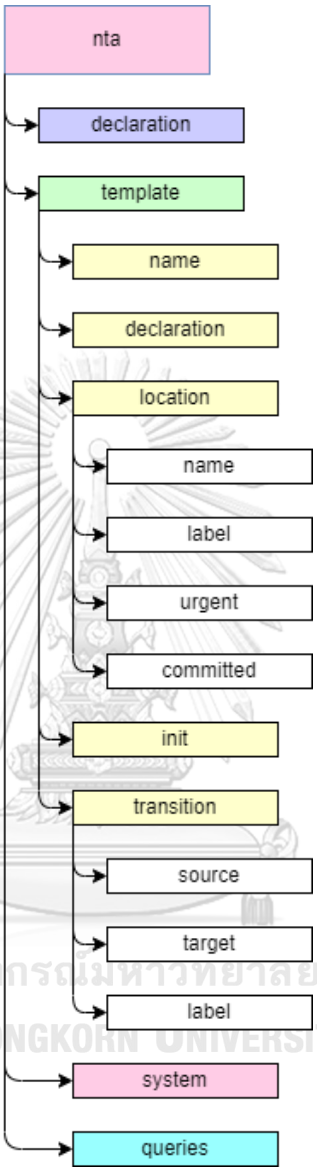
แท็ก name ระบุชื่อของไทมด์ออโตมาตา แท็ก declaration ระบุการนิยามการประกาศตัวแปรของออโตมาตานั้น ๆ แท็ก init ระบุตำแหน่งเริ่มต้นของไทมด์ออโตมาตา แท็ก location ที่อธิบายรายละเอียดองค์ประกอบของตำแหน่ง Name, Invariant, Urgent, Committed แท็ก transition ที่อธิบายรายละเอียดองค์ประกอบของเส้นเชื่อม Guard, Sync, Update โดยจะมีการระบุตำแหน่งตั้งต้น และปลายทางด้วย

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE nta PUBLIC '-//Uppaal Team//DTD Flat System 1.1//EN' 'http:/
<nta>
    <declaration>// Place global declarations here.</declaration>
    <template>
        <name x="5" y="5">Template</name>
        <declaration>// Place local declarations here.
clock i;</declaration>
        <location id="id0" x="0" y="0">
            <label kind="invariant" x="-10" y="17">i&lt;=0</label>
        </location>
        <location id="id1" x="89" y="0">
        </location>
        <init ref="id0"/>
        <transition>
            <source ref="id0"/>
            <target ref="id1"/>
            <label kind="synchronisation" x="18" y="-17">a!</label>
            <label kind="assignment" x="18" y="0">i:=0</label>
        </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>
```

รูปที่ 3.6 ตัวอย่างแฟ้มเอกซ์เอ็มแอลของไทมด์ออโตมาตาจากเครื่องมือ UPPAAL

จากความสัมพันธ์ระหว่างสัญลักษณ์**ไทมด์**ออโตมาตา และแท็กเอกซ์เอ็มแอลทำให้เราได้ทราบแท็กที่จำเป็นในการใช้แปลงจากกระแสงานยอว์ล ทำการสรุปโครงสร้างของแท็กโดยสามารถอธิบายเป็นโครงสร้างที่จำเป็นได้ดังรูปที่ 3.7 ซึ่งจะอธิบายถึงส่วนที่เราสนใจใช้ในการแปลงคือ แท็ก template ซึ่งอยู่ภายใต้แท็ก nta และมีแท็กลูกต่าง ๆ ที่เกี่ยวข้อง เช่น name, location, init และ transition ที่เก็บข้อมูลของแต่ละไทมด์ออโตมาตาอยู่

เป็นต้น โดยในแต่ละแท็กของ location ก็จะมีข้อมูลของแต่ละ location อยู่ และแท็ก transition ก็จะมีข้อมูลของแต่ละเส้นเชื่อมอยู่

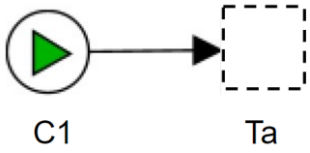
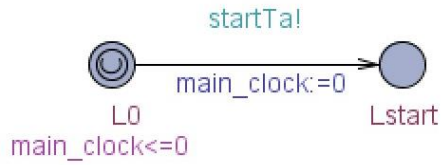
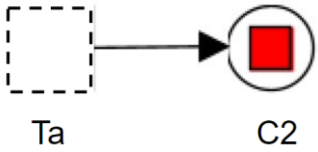
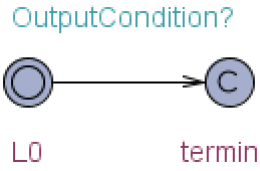
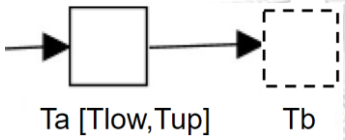
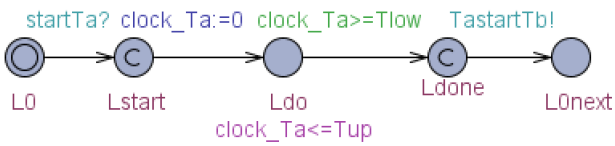
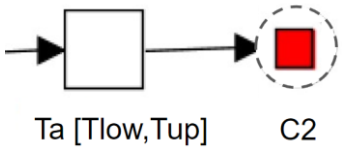
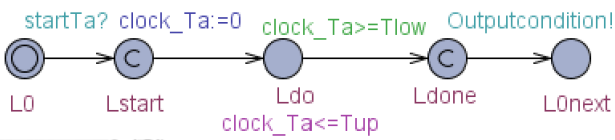
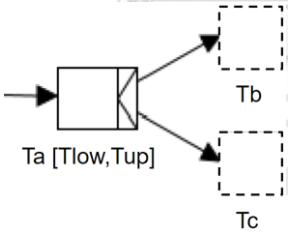
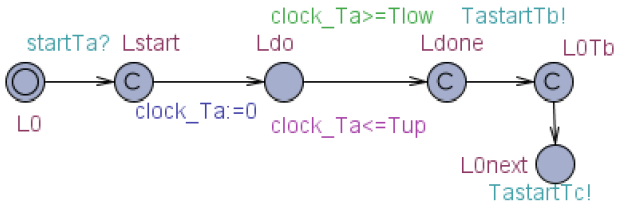
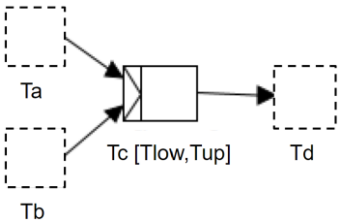
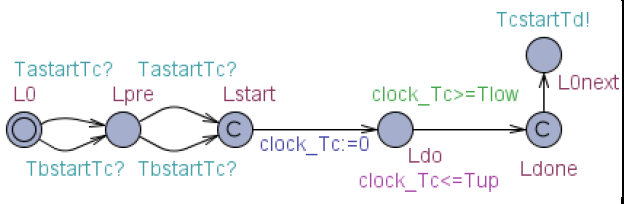
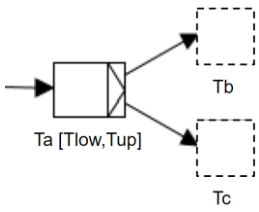
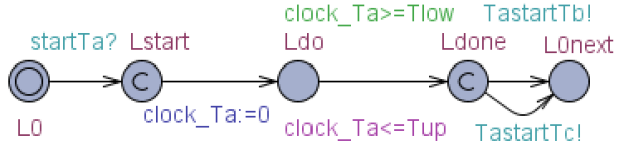


รูปที่ 3.7 แสดงโครงสร้างเอกซ์เอ็มแอลของไทมด์ออโตมาตาจากเครื่องมือ UPPAAL

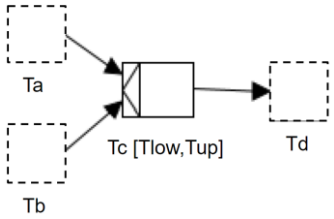
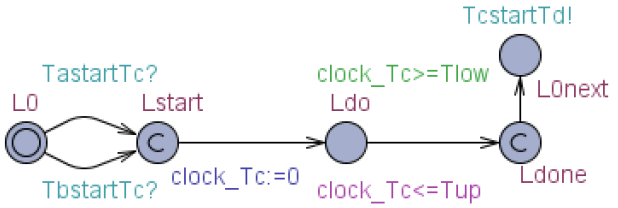
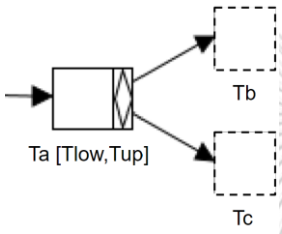
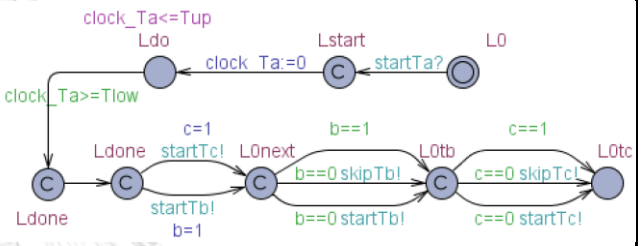
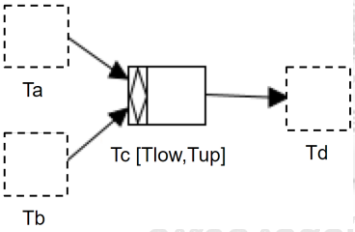
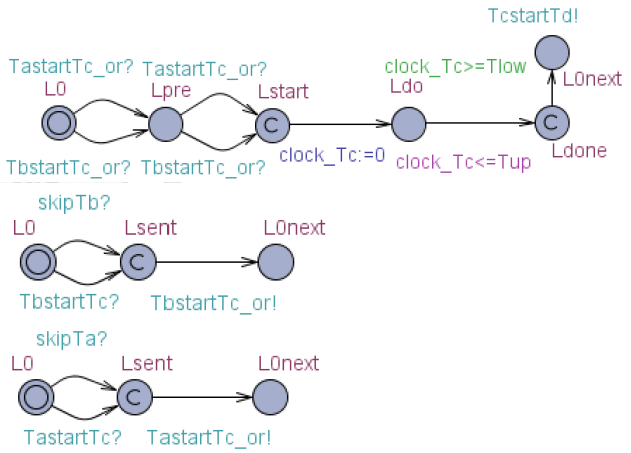
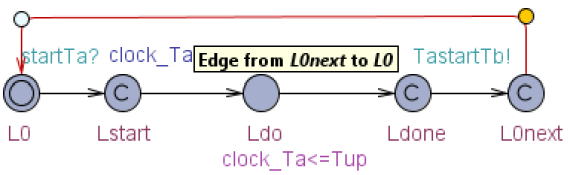
3.3 การออกแบบ และสร้างเกณฑ์ในการแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาเป็นไทมด์ออโตมาตา

หลังจากที่ได้ศึกษา และวิเคราะห์เปรียบเทียบกระแสงานยอว์ล และไทมด์ออโตมาตา ทั้งด้านพฤติกรรม และโครงสร้างแท็กของเอกซ์เอ็มแอลแล้วนำข้อมูลต่าง ๆ มาออกแบบเป็นกฎการแปลงจากกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาเป็นไทมด์ออโตมาตา ได้ดังตารางที่ 3.4 ซึ่งอธิบายลักษณะงานในกระแสงานยอว์ลชนิด Input condition, Output condition, Atomic task, Atomic task to outputcondition, AND-split task, AND-join task, XOR-split task, XOR-join task, OR-split task, OR-join task และ loop เพื่อใช้ในการสร้างซอฟต์แวร์สำหรับการแปลงกระแสงานยอว์ลเป็นไทมด์ออโตมาตาต่อไป

ตารางที่ 3.4 กฎการแปลงจากกระแสงานยอว์ลแบบมีข้อจำกัดช่วงเวลาไปเป็นไทมด์ออโตมาตา

ชื่อ	สัญลักษณ์ยอว์ล และช่วงเวลา	ไทมด์ออโตมาตา
Input condition		
Output condition		
Atomic task		
Atomic task to outputcondition		
AND-split		
AND-join		
XOR-split		

ตารางที่ 3.4 กฎการแปลงจากกระแสงานยอว์ลแบบมีข้อจำกัดช่วงเวลาไปเป็นไทมด์ออโตมาตา (ต่อ)

ชื่อ	สัญลักษณ์ยอว์ล และช่วงเวลา	ไทมด์ออโตมาตา
XOR-join		
OR-split		
OR-join		
Loop	Extra at end and initial location of Timed Automaton	

จากตารางที่ 3.4 กฎการแปลงจากกระแสงานยอว์ลแบบมีข้อจำกัดช่วงเวลาไปเป็นไทมด์ออโตมาตาสามารถอธิบายเป็นขั้นตอนเพื่อใช้ในการออกแบบการทำงานของซอฟต์แวร์รวมถึงกรณีที่ *TD* ทั้ง Join กับ Split มากกว่า 2 งานดังต่อไปนี้

กฎข้อที่ 1: การแปลง Input Condition ของกระแสงานยอว์ล

กำหนดให้กระแสงานยอว์ล *YML* เพื่อแปลงไปเป็นไทมด์ออโตมาตา *TA* มีขั้นตอนการแปลงดังต่อไปนี้ ที่แสดงในตารางที่ 3.4

- ถ้าพบ Input condition *C1* ที่เชื่อมไปยัง Atomic task *Ta* ใน *YML* แล้วสร้างไทมด์ออโตมาตา *TA* ที่ประกอบด้วย *L0* (Urgent location) ที่มีเส้นเชื่อมไปยังตำแหน่ง *Lstart* และกำหนดให้ *L0* เป็น Initial location มี Invariant เป็น *main_clock <= 0* เส้นเชื่อมระหว่าง *L0, Lstart* จะมีการส่ง Synchronization channel *startTa!* โดยสามารถประกาศตัวแปรได้ดังนี้ *main_clock := 0 (I0, startTa! , - , main_clock := 0, Lstart)* $\in E$, *L0, Lstart* $\in L$, *main_clock* $\in X$ และ *(I { main_clock <= 0} : L0* $\longrightarrow$ *{ main_clock == 0})*

กฎข้อที่ 2: การแปลง Output Condition ของกระแสงานยอว์ล

กำหนดให้กระแสงานยอว์ล *YML* เพื่อแปลงไปเป็นไทมด์ออโตมาตา *TA* มีขั้นตอนการแปลงดังต่อไปนี้ ที่แสดงในตารางที่ 3.4

- ถ้าพบ Output condition *C2* ที่เชื่อมมาจาก Atomic task *Ta* ใน *YML* แล้วสร้างไทมด์ออโตมาตา *TA* ที่ประกอบด้วย *L0* มีเส้นเชื่อมไปยังตำแหน่ง *Lterminal* และกำหนดให้ *L0* เป็น Initial location ตำแหน่ง *Lterminal* เป็น Committed location เส้นเชื่อมละหว่าง *L0, Lterminal* จะมีการรอรับ Synchronization channel ชื่อ *Outputcondition?* โดยสามารถประกาศตัวแปรได้ดังนี้ *(L0, Outputcondition? , - , - , Lstart)* $\in E$ *และ L0, Lterminal* $\in L$

กฎข้อที่ 3: การแปลง Atomic Task ของกระแสงานยอว์ล

กำหนดให้กระแสงานยอว์ล *YML* เพื่อแปลงไปเป็นไทมด์ออโตมาตา *TA* มีขั้นตอนการแปลงดังต่อไปนี้ ที่แสดงในตารางที่ 3.4

- ถ้าพบ Atomic task *Ta* ที่มีช่วงเวลา *[Tlow,Tup]* และเชื่อมไปยัง Atomic task *Tb* ใน *YML* แล้วสร้างไทมด์ออโตมาตา *TA* ที่ประกอบด้วย *Lstart (*Committed location) เชื่อมต่อเนื่องกับ *Ldo* และ *Ldone (*Committed location) โดยที่ *Ldo* กำหนด Invariant คือ *clock_Ta <= Tup* เส้นเชื่อมระหว่าง *Lstart* กับ *Ldo* กำหนดให้ *clock_Ta =0* และเส้นเชื่อมระหว่าง *Ldo* กับ *Ldone* กำหนด Guard conditions คือ *clock_Ta >= Tlow*
- เพิ่มตำแหน่ง *L0* (initial location) เชื่อมไปยัง *Lstart* โดยบนเส้นเชื่อมของ *L0* กับ *Lstart* มีการระบุการรับ Synchronization channel ชื่อว่า *startTa?*
- เพิ่มตำแหน่ง *L0next* เชื่อมต่อมาจาก *Ldone* โดยบนเส้นเชื่อมระหว่าง *Ldone* กับ *L0next* จะมีการส่ง Synchronization channel คือ *TastartTb!*

โดยสามารถประกาศตัวแปรได้ดังนี้

- *L0,Lstart, Ldo, Ldone, L0next* $\in$ *L*
- *clock_Ta* $\in$ *C*
- *(L0, startTa? , - , - , Lstart), (Lstart, -, -, clock_Ta := 0, Ldo), (Ldo, -, clock_Ta>=Tlow, - , Ldone) , (Ldone, TastartTb!, -, - , L0next)* $\in$ *E*

กฎข้อที่ 4: การแปลง Atomic task to outputcondition ของกระแสงานยอว์ล

กำหนดให้กระแสงานยอว์ล *YML* เพื่อแปลงไปเป็นไทมด์ออโตมาตา *TA* มีขั้นตอนการแปลงดังต่อไปนี้ ที่แสดงในตารางที่ 3.4

- ถ้าพบ Atomic task *Ta* ที่มีช่วงเวลา *[Tlow,Tup]* และเชื่อมไปยัง Output condition *C2* ใน *YML* แล้วสร้างไทมด์ออโตมาตา *TA* ที่ประกอบด้วย *Lstart (*Committed location) เชื่อมต่อเนื่องกับ *Ldo* และ *Ldone (*Committed location) โดยที่ *Ldo* กำหนด Invariant คือ *clock_Ta <= Tup* เส้นเชื่อมระหว่าง *Lstart* กับ *Ldo* กำหนดให้ *clock_Ta = 0* และเส้นเชื่อมระหว่าง *Ldo* กับ *Ldone* กำหนด Guard conditions คือ *clock_Ta >= Tlow*
- เพิ่มตำแหน่ง *L0* (Initial location) เชื่อมไปยัง *Lstart* โดยบนเส้นเชื่อมของ *L0* กับ *Lstart* มีการระบุการรับ Synchronization channel ชื่อว่า *startTa?*
- เพิ่มตำแหน่ง *L0next* เชื่อมต่อมาจาก *Ldone* โดยบนเส้นเชื่อมระหว่าง *Ldone* กับ *L0next* จะมีการส่ง Synchronization channel คือ *Outputcondition!*

โดยสามารถประกาศตัวแปรได้ดังนี้

- *L0*, *Lstart, Ldo, Ldone, L0next* $\in$ *L*
- *clock_Ta* $\in$ *C*
- *(L0, startTa? , - , - , Lstart), (Lstart, -, -, clock_Ta := 0, Ldo), (Ldo, -, clock_Ta>=Tlow, - , Ldone), (Ldone, Outputcondition!, -, - , L0next)* $\in$ *E*

กฎข้อที่ 5: การแปลง AND Split ของกระแสงานยอว์ล

กำหนดให้กระแสงานยอว์ล *YML* เพื่อแปลงไปเป็นไทมด์ออโตมาตา *TA* มีขั้นตอนการแปลงดังต่อไปนี้ ที่แสดงในตารางที่ 3.4

- ถ้าพบ Atomic task *Ta* ที่มีช่วงเวลา *[Tlow,Tup]* และเชื่อมไปยัง Atomic task *Tb* และ *Tc* ใน *YML* แล้วสร้างไทมด์ออโตมาตา *TA* ที่ประกอบด้วย *Lstart (*Committed location) เชื่อมต่อเนื่องกับ *Ldo* และ *Ldone (*Committed location) โดยที่ *Ldo* กำหนด Invariant คือ *clock_Ta <= Tup* เส้นเชื่อมระหว่าง *Lstart* กับ *Ldo* กำหนดให้ *clock_Ta = 0* และเส้นเชื่อมระหว่าง *Ldo* กับ *Ldone* กำหนด Guard conditions คือ *clock_Ta >= Tlow*

- เพิ่มตำแหน่ง *L0* (Initial location) เชื่อมไปยัง *Lstart* โดยบนเส้นเชื่อมของ *L0* กับ *Lstart* มีการระบุการรับ Synchronization channel ชื่อว่า *startTa?*
- เพิ่มตำแหน่ง *L0Tb (*Committed location), *L0next* เชื่อมต่อเนื่องมาจาก *Ldone* โดยบนเส้นเชื่อมระหว่าง *Ldone* กับ *L0Tb* จะมีการส่ง Synchronization channel คือ *TastartTb!* และเชื่อมระหว่าง *L0Tb* กับ *L0next* จะมีการส่ง Synchronization channel คือ *TastartTc!*

ในกรณีที่ AND Split = *n* tasks

- เพิ่ม *LTi ∈ L โดยที่ i=1* ถึง *n* เชื่อมต่อเนื่องจาก *Ldone* แล้วแต่ละเส้นเชื่อมจะมีการกำหนด Synchronization channel คือ *TastartTi!* โดยทุกตำแหน่งที่เพิ่มเข้ามากำหนดเป็น Committed location ยกเว้นตำแหน่งสุดท้าย

กฎข้อที่ 6: การแปลง AND Join ของกระแสงานยอว์ล

กำหนดให้กระแสงานยอว์ล *YML* เพื่อแปลงไปเป็นไทมด์ออโตมาตา *TA* มีขั้นตอนการแปลงดังต่อไปนี้ ที่แสดงในตารางที่ 3.4

- ถ้าพบ Atomic task *Tc* ที่มีช่วงเวลา *[Tlow,Tup]* เชื่อมมาจาก Atomic task *Ta* และ *Tb* ใน *YML* แล้วสร้างไทมด์ออโตมาตา *TA* ที่ประกอบด้วย *Lstart (*Committed location) เชื่อมต่อเนื่องกับ *Ldo* และ *Ldone (*Committed location) โดยที่ *Ldo* กำหนด Invariant คือ *clock_Tc <= Tup* เส้นเชื่อมระหว่าง *Lstart* กับ *Ldo* กำหนดให้ *clock_Tc = 0* และเส้นเชื่อมระหว่าง *Ldo* กับ *Ldone* กำหนด Guard conditions คือ *clock_Tc >= Tlow*
- เพิ่มตำแหน่ง *L0* (Initial location), *Lpre* ก่อนหน้า *Lstart* สร้างเส้นเชื่อมของ *L0* กับ *Lpre* และ *Lpre* กับ *Lstart* คู่ละ 2 เส้นโดยเส้นแรกมีการระบุการรับ Synchronization channel ชื่อว่า *TastartTc?* เส้นที่สองชื่อว่า *TbstartTc?.*
- เพิ่มตำแหน่ง *L0next* เชื่อมต่อมาจาก *Ldone* โดยบนเส้นเชื่อมระหว่าง *Ldone* กับ *L0next* จะมีการส่ง Synchronization channel คือ *TcstartTd!*

ในกรณี AND Join = *n* tasks

- เพิ่ม *Li ∈ L* โดยที่ *i=1* ถึง *n-1* ก่อนหน้า *Lstart* และเพิ่มเส้นเชื่อมตำแหน่งก่อนหน้า *Lstart* คู่ละ *n* เส้น โดยมี Synchronization channel คือ *TistartTc?* โดยที่ *i=1* ถึง *n* บนเส้นเชื่อมเส้นในแต่ละคู่ตำแหน่ง

กฎข้อที่ 7: การแปลง XOR Split ของกระแสงานยอว์ล

กำหนดให้กระแสงานยอว์ล *YML* เพื่อแปลงไปเป็นไทมด์ออโตมาตา *TA* มีขั้นตอนการแปลงดังต่อไปนี้ ที่แสดงในตารางที่ 3.4

- ถ้าพบ Atomic task *Ta* ที่มีช่วงเวลา *[Tlow,Tup]* และเชื่อมไปยัง Atomic task *Tb* และ *Tc* ใน *YML* แล้วสร้างไทมด์ออโตมาตา *TA* ที่ประกอบด้วย *Lstart (*Committed location) เชื่อมต่อเนื่องกับ *Ldo* และ *Ldone (*Committed location) โดยที่ *Ldo* กำหนด Invariant คือ *clock_Ta <= Tup* เส้นเชื่อมระหว่าง *Lstart* กับ *Ldo* กำหนดให้ *clock_Ta = 0* และเส้นเชื่อมระหว่าง *Ldo* กับ *Ldone* กำหนด Guard conditions คือ *clock_Ta >= Tlow*
- เพิ่มตำแหน่ง *L0* (Initial location) เชื่อมไปยัง *Lstart* โดยบนเส้นเชื่อมของ *L0* กับ *Lstart* มีการระบุการรับ Synchronization channel ชื่อว่า *startTa?*
- เพิ่มตำแหน่ง *L0next* หลังตำแหน่ง *Ldone* สร้างเส้นเชื่อมระหว่าง *Ldone* กับ *L0next* 2 เส้นโดยเส้นแรกจะมีการส่ง Synchronization channel คือ *TastartTb!* และเส้นที่สองคือ *TastartTc!*

ในกรณี XOR Split = *n* tasks

- เพิ่มเส้นเชื่อมระหว่าง *Ldone , L0next n* เส้น กำหนด Synchronization channel *TastartTi!* โดยที่ *i=1* ถึง *n* ในแต่ละเส้นตามลำดับ

กฎข้อที่ 8: การแปลง XOR Join ของกระแสงานยอว์ล

กำหนดให้กระแสงานยอว์ล *YML* เพื่อแปลงไปเป็นไทมด์ออโตมาตา *TA* มีขั้นตอนการแปลงดังต่อไปนี้ ที่แสดงในตารางที่ 3.4

- ถ้าพบ Atomic task *Tc* ที่มีช่วงเวลา *[Tlow,Tup]* เชื่อมมาจาก Atomic task *Ta* และ *Tb* ใน *YML* แล้วสร้างไทมด์ออโตมาตา *TA* ที่ประกอบด้วย *Lstart (*Committed location) เชื่อมต่อเนื่องกับ *Ldo* และ *Ldone (*Committed location) โดยที่ *Ldo* กำหนด Invariant คือ *clock_Tc <= Tup* เส้นเชื่อมระหว่าง *Lstart* กับ *Ldo* กำหนดให้ *clock_Tc = 0* และเส้นเชื่อมระหว่าง *Ldo* กับ *Ldone* กำหนด Guard conditions คือ *clock_Tc >=Tlow*
- เพิ่มตำแหน่ง *L0* (Initial location) ก่อนหน้า *Lstart* โดยสร้างเส้นเชื่อมสองเส้นระหว่าง *L0* กับ *Lstart* เส้นแรกจะมีการรับ Synchronization channel คือ *TastartTc?* และเส้นที่สองคือ *TbstartTc?.*
- เพิ่มตำแหน่ง *L0next* เชื่อมต่อมาจาก *Ldone* โดยบนเส้นเชื่อมระหว่าง *Ldone* กับ *L0next* จะมีการส่ง Synchronization channel คือ *TcstartTd!*

ในกรณี XOR Split = *n* tasks

- เพิ่มเส้นเชื่อมระหว่าง *L0* กับ *Lstart n* เส้น กำหนด Synchronization channel คือ *TistartTc!* โดยที่ *i=1* ถึง *n* ในแต่ละเส้นตามลำดับ

กฎข้อที่ 9: การแปลง OR Split ของกระแสงานยอว์ล

กำหนดให้กระแสงานยอว์ล *YML* เพื่อแปลงไปเป็นไทมด์ออโตมาตา *TA* มีขั้นตอนการแปลงดังต่อไปนี้ ที่แสดงในตารางที่ 3.4

- ถ้าพบ Atomic task *Ta* ที่มีช่วงเวลา *[Tlow,Tup]* และเชื่อมไปยัง Atomic task *Tb* และ *Tc* ใน *YML*แล้วสร้างไทมด์ออโตมาตา *TA* ที่ประกอบด้วย *Lstart (*Committed location) เชื่อมต่อเนื่องกับ *Ldo* และ *Ldone (*Committed location) โดยที่ *Ldo* กำหนด Invariant คือ *clock_Ta <= Tup* เส้นเชื่อมระหว่าง *Lstart* กับ *Ldo* กำหนดให้ *clock_Ta = 0* และเส้นเชื่อมระหว่าง *Ldo* กับ *Ldone* กำหนด Guard conditions คือ *clock_Ta >= Tlow*
- เพิ่มตำแหน่ง *L0* (Initial location) เชื่อมไปยัง *Lstart* โดยบนเส้นเชื่อมของ *L0* กับ *Lstart* มีการระบุการรับ Synchronization channel ชื่อว่า *startTa?*
- เพิ่มตำแหน่ง *L0next(*Committed location), *L0Tb (*Committed location) และ *L0Tc* หลังจาก *Ldone* โดยสร้างเส้นเชื่อมระหว่าง *Ldone, L0next* สองเส้น เส้นแรกกำหนด Synchronization channel คือ *startTb!, Tb =1* และเส้นที่สองคือ *startTc!* , *Tc =1*
- สร้างเส้นเชื่อมระหว่าง *L0next , L0Tb* สามเส้น เส้นแรกกำหนด Guard conditions คือ *Tb ==1* เส้นที่สองกำหนดคือ Guard conditions *Tb ==0*, Synchronization channel คือ *startTb!* เส้นที่สามกำหนด Guard conditions คือ *Tb ==0*, Synchronization channel คือ *skipTb!*
- สร้างเชื่อมระหว่าง *L0Tb , L0Tc* สามเส้น เส้นแรกกำหนด Guard conditions คือ *Tc ==1* เส้นที่สองกำหนด Guard conditions คือ *Tc ==0* , Synchronization channel คือ *startTc!* เส้นที่สามกำหนด Guard conditions *Tc ==0* , Synchronization channel คือ *skipTc !*

ในกรณี OR Split = *n* tasks

- เพิ่ม *L0Ti* $\in L$ โดยที่ *i=1* ถึง *n* ต่อจาก *L0next* โดยเพิ่มเส้นเชื่อมระหว่าง *Ldone* กับ *L0next* *n* เส้น กำหนด Synchronization channel คือ *startTi!* โดยที่ *i=1* ถึง *n* ในแต่ละเส้น และกำหนดตัวแปร *Ti =1* โดยที่ *i=1* ถึง *n* ในแต่ละเส้นตามลำดับ
- เพิ่มเส้นเชื่อมระหว่าง *L0Ti , L0Ti+1* อย่างละสามเส้น โดยที่ *i=1* ถึง *n-1* เส้นแรก กำหนด Guard conditions คือ *Ti+1 ==1* เส้นที่สอง กำหนด Guard conditions คือ *Ti+1 ==0*, Synchronization channel คือ *startTi+1!* เส้นที่สาม กำหนด Guard conditions คือ *Ti+1 ==0* , Synchronization channel คือ *skipTi+1*

กฎข้อที่ 10: การแปลง OR Join ของกระแสงานยอว์ล

กำหนดให้กระแสงานยอว์ล *YML* เพื่อแปลงไปเป็นไทมด์ออโตมาตา *TA* มีขั้นตอนการแปลงดังต่อไปนี้ ที่แสดงในตารางที่ 3.4

- ถ้าพบ Atomic task *Tc* ที่มีช่วงเวลา *[Tlow,Tup]* เชื่อมมาจาก Atomic task *Ta* และ *Tb* ใน *YML* แล้วสร้างไทมด์ออโตมาตา *TA* ที่ประกอบด้วย *Lstart (*Committed location) เชื่อมต่อเนื่องกับ *Ldo* และ *Ldone (*Committed location) โดยที่ *Ldo* กำหนด Invariant คือ *clock_Tc <= Tup* เส้นเชื่อมระหว่าง *Lstart* กับ *Ldo* กำหนดให้ *clock_Tc = 0* และเส้นเชื่อมระหว่าง *Ldo* กับ *Ldone* กำหนด Guard conditions คือ *clock_Tc >= Tlow*
- เพิ่มตำแหน่ง *L0* (Initial location), *Lpre* ก่อนหน้า *Lstart* สร้างเส้นเชื่อมของ *L0* กับ *Lpre* และ *Lpre* กับ *Lstart* คู่ละ 2 เส้นโดยเส้นแรกมีการระบุการรับ Synchronization channel ชื่อว่า *TastartTc_or?* เส้นที่สองชื่อว่า *TbstartTc_or?.*
- เพิ่มตำแหน่ง *L0next* เชื่อมต่อมาจาก *Ldone* โดยบนเส้นเชื่อมระหว่าง *Ldone* กับ *L0next* จะมีการส่ง Synchronization channel คือ *TcstartTd!*
- สร้าง Timed automaton *TA* ที่ประกอบด้วย *L0* เชื่อมต่อเนื่องกับ *Lsent, L0next* $\in L$. โดยเพิ่มเส้นเชื่อม *L0 , Lsent* อีก 1 เส้น กำหนดให้ Synchronization channel คือ *skipTa? , TastartTc?* ตามลำดับ และให้ *Lsent* เป็น Committed location
- สร้าง Timed automaton *TA* ที่ประกอบด้วย *L0* เชื่อมต่อเนื่องกับ *Lsent, L0next* $\in L$ โดยเพิ่มเส้นเชื่อม *L0 , Lsent* อีก 1 เส้น กำหนดให้ Synchronization channel คือ *skipTb?, TbstartTc?* ตามลำดับ และให้ *Lsent* เป็น Committed location

ในกรณี OR Join = *n* tasks

- เพิ่มตำแหน่ง *Li* $\in L$ โดยที่ *i=1* ถึง *n-1* และเส้นเชื่อม Location ก่อนหน้า *Lstart* อย่างละ *n* เส้น โดยมี Synchronization channel คือ *TistartTc?* โดยที่ *i=1* ถึง *n* บนเส้นเชื่อมเส้น
- สร้าง Timed automaton *TA* จำนวน *n TA* ที่ประกอบด้วย *L0* เชื่อมต่อเนื่องกับ *Lsent, L0next* $\in L$ โดยเพิ่มเส้นเชื่อม *L0 , Lsent* 2 เส้น กำหนด Synchronization channel คือ *skipTi?, TistartTc?* โดยที่ *i=1* ถึง *n* ตามลำดับและให้ *Lsent* เป็น Committed location

กฎข้อที่ 11: การแปลงให้รองรับการทำงานแบบ Loop ของกระแสงานยอว์ล

กำหนดให้กระแสงานยอว์ล *YML* เพื่อแปลงไปเป็นไทมด์ออโตมาตา *TA* มีขั้นตอนการแปลงดังต่อไปนี้ ที่แสดงในตารางที่ 3.4

- กฎทุกข้อยกเว้นข้อ 1 และ 2 จะมีการสร้างเส้นเชื่อมระหว่างตำแหน่งสุดท้ายกับตำแหน่งแรกของ *TA* โดยให้ตำแหน่งสุดท้ายเป็น Committed location

โดยเหตุผลการออกแบบที่เลือกให้ 1 สัญลักษณ์ของยอว์ลเป็น 1 ชุดสัญลักษณ์ของไทมด์ออโตมาตอน หลังจากการทดสอบแล้วสามารถอธิบายได้ดังตารางการเปรียบเทียบแนวคิดการออกแบบระหว่างให้ 1 สัญลักษณ์ของยอว์ลเป็นเป็น 1 ชุดสัญลักษณ์ของไทมด์ออโตมาตอนแนวคิดของคุณ Afsoon Soltani จาก Timed Automata for Workflow Modeling and Analysis [6] กับแนวคิดการออกแบบ 1 ชุดสัญลักษณ์ของยอว์ลเป็นเป็น 1 ชุดสัญลักษณ์ไทมด์ออโตมาตอนของคุณ Maruth-Ravibanjurdkul Transforming YAWL Workflows with Time Constraints to Timed Automata [7] ดังตารางที่ 3.5

ตารางที่ 3.5 เปรียบเทียบแนวทางการออกแบบกฎการแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาเป็นไทมด์ออโตมาตา

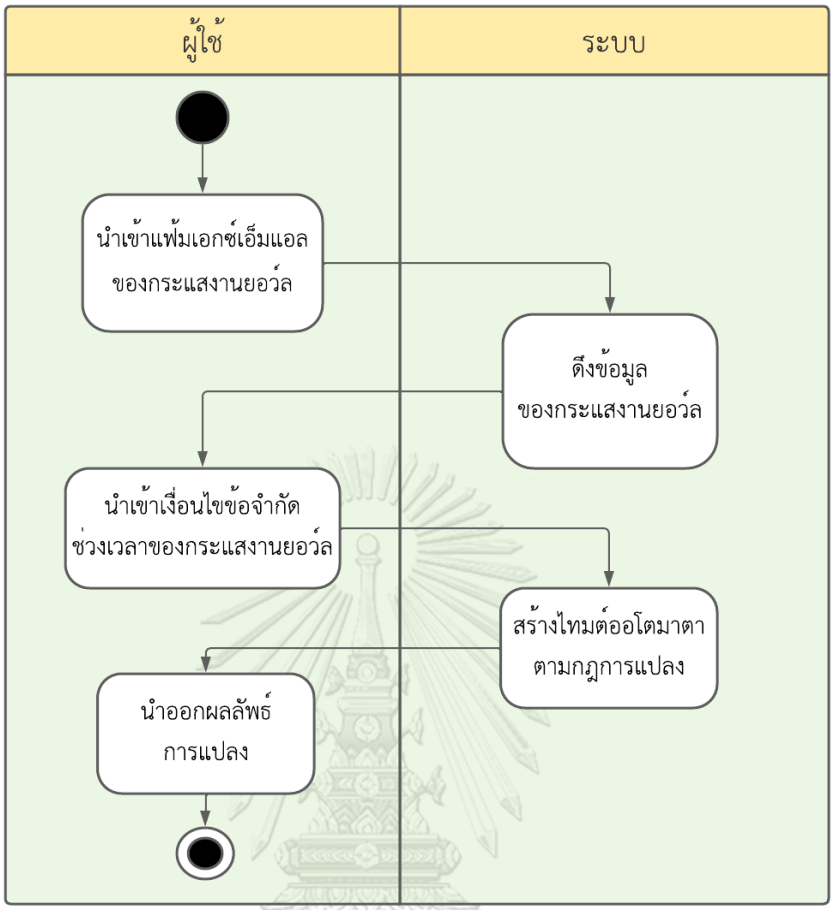
ความสามารถ	**งานที่เสนอ**	**Maruth-Ravibanjurdkul [7]**
รูปแบบการแปลง	1 สัญลักษณ์ของยอว์ลเป็น 1 ชุดสัญลักษณ์ของไทมด์ออโตมาตอน	1 ชุดสัญลักษณ์ของยอว์ลเป็นเป็น 1 ชุดสัญลักษณ์ไทมด์ออโตมาตอน
จำนวนสถานะไทมด์ออโตมาตา	ต้องใช้ความเข้าใจไทมด์ออโตมาเพื่อเข้าใจการทำงานของกระแสงานเนื่องจากมีจำนวนสถานะมากกว่า โดย เกิดจากการสร้างไทมด์ออโตมาตอนแยกอิสระ	ง่ายต่อการทำความเข้าใจและมองสัญลักษณ์ต่างๆในไทมด์ออโตมาเนื่องจากมีจำนวนสถานะใกล้เคียงกับกระแสงานยอว์ล
การกำหนด Invariant	สามารถกำหนด Invariant ให้แต่ละงานอิสระได้เลยเพราะถูกแยกให้แต่ละงานเป็น 1 ไทมด์ออโตมาตอนแล้ว ทำให้ไม่เกิดกรณีที่สถานะไม่ยอมส่งงานต่อ	ไม่สามารถกำหนด Invariant ให้กับไทมด์ออโตมาตอนหลักได้ในกรณีที่เกิดการส่งไปให้ Sub ไทมด์ออโตมาตอนทำให้สามารถเกิดกรณีที่ไทมด์ออโตมาตาไม่เปลี่ยนสถานะส่งต่องานได้
นาฬิกา	มีนาฬิกาท้องถิ่นมีแต่ละไทมด์ออโตมาตอนทำให้สามารถตรวจเช็คเวลาการทำงานของแต่ละงานของกระแสงานได้	แต่ละงานของกระแสงานใช้นาฬิการ่วมกัน ทำให้ไม่สามารถทวนสอบในแต่ละงานย่อยได้ว่าทำงานถูกต้องตามที่กำหนดหรือไม่

ตารางที่ 3.5 เปรียบเทียบแนวทางการออกแบบกฎการแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาเป็นไทมด์ออโตมา (ต่อ)

ความสามารถ	งานที่เสนอ	Maruth-Ravibanjurdkul [7]
รูปแบบกระแสงาน	ในกรณีของ AND และ XOR สามารถรองรับกรณีไม่เป็น Well-formed ได้ เพราะเนื่องจากแต่ละงานแยกเป็นไทมด์ออโตมาตอนของงานนั้นอิสระ ทำให้ไม่จำเป็นต้องอยู่ในรูปแบบ Well-formed	กระแสงานยอว์ลต้องอยู่ในแบบ Well-formed เท่านั้น เพราะเนื่องจากตอนแยกงานไปให้ไทมด์ออโตมาตอนที่เป็น Sub แล้วจะต้องมีการรวมกลับมาที่ไทมด์ออโตมาตอนหลักเสมอ
กระแสงานแบบส่งสลับ	สามารถรองรับกระแสงานที่มีลักษณะการงานแบบส่งงานสลับกันตามรูปที่ 3.2 ได้ เพราะเนื่องจากแต่ละงานแยกเป็นไทมด์ออโตมาตอนของงานนั้นอิสระ	ไม่สามารถรองรับกระแสงานที่มีลักษณะการงานแบบส่งงานสลับกันตามรูปที่ 3.2 ได้ เพราะเนื่องจากจะมีปัญหาตอนส่งแยกไปแต่ละงานย่อยแล้ว ไม่สามารถกำหนดได้งานงานต่อไปงานใดจะเป็นงานหลักในไทมด์ออโตมาตอนหลัก

3.4 การออกแบบ และสร้างซอฟต์แวร์สำหรับการแปลงกระแสงานยอว์ลเป็นไทมด์ออโตมาตา

เมื่อวิเคราะห์โครงสร้างแฟ้มเอกซ์เอ็มแอลของยอว์ลและแฟ้มเอกซ์เอ็มแอลของไทมด์ออโตมาตา กับออกแบบ และสร้างเกณฑ์ในการแปลงกระแสงานยอว์ลเป็นไทมด์ออโตมาตาแบบมีข้อจำกัดช่วงเวลาแล้ว ผู้วิจัยนำเอาข้อมูลที่ได้มาเปรียบเทียบระหว่างกฎการแปลงกับโครงสร้างเอกซ์เอ็มแอลเพื่อทำการออกแบบ และสร้างซอฟต์แวร์สำหรับการแปลงกระแสงานยอว์ลเป็นไทมด์ออโตมาตาที่รองรับการนำเข้าข้อมูลข้อจำกัดช่วงเวลา ทำการออกแบบกระบวนการทำงานของซอฟต์แวร์ดังรูปที่ 3.8 ซึ่งแบ่งได้เป็น 5 กิจกรรมแยกเป็นฝั่งผู้ใช้กับระบบ

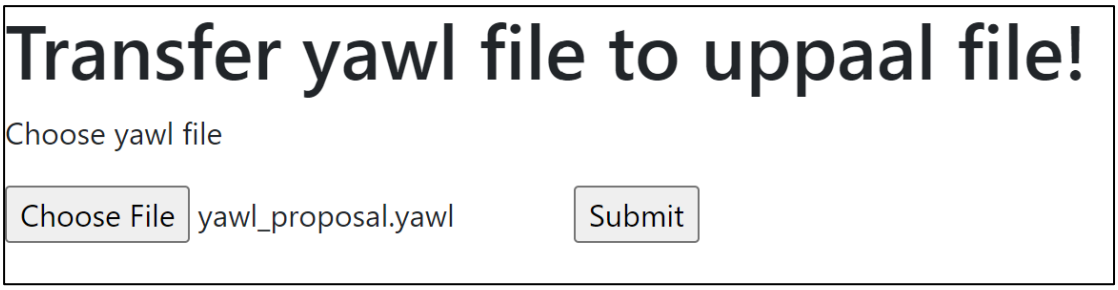


รูปที่ 3.8 กระบวนการทำงานของซอฟต์แวร์

จากรูปที่ 3.8 จะเริ่มต้นที่นำเข้าแฟ้มเอกซ์เอ็มแอลของกระแสงานยอว์ลเข้าสู่ระบบ ต่อมาระบบจะทำการดึงข้อมูลของกระแสงานยอว์ลให้อยู่ในรูปแบบ Dataframe หลังจากนั้นจะส่งกลับมาให้ผู้ใช้นำเข้าเงื่อนไขข้อจำกัดช่วงเวลาของกระแสงานยอว์ลเพื่อนำมากำกับใน Dataframe แล้วระบบจึงทำการแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาไปเป็นไทม์ออโตมาตาตามกฎการแปลง หลังจากแปลงเรียบร้อยแล้วทางผู้ใช้สามารถนำออกผลลัพธ์ของการแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาไปเป็นไทม์ออโตมาตาได้ โดยกระบวนการทั้งหมดสามารถอธิบายได้ดังต่อไปนี้

1) นำเข้าแฟ้มเอกซ์เอ็มแอลของกระแสงานยอว์ล

ในส่วนแรกนั้นจะเป็นการรับเข้าแฟ้มเอกซ์เอ็มแอลของกระแสงานยอว์ลเพื่อป้อนข้อมูลให้กับซอฟต์แวร์ในการแปลงกระแสงานยอว์ลเป็นไทม์ออโตมาตาต่อไปดังรูปที่ 3.9 โดยการกดปุ่ม Choose File เพื่อเลือกแฟ้มเอกซ์เอ็มแอลของกระแสงานยอว์ลที่ต้องการนำเข้ามาแปลง หลังจากนั้นกดปุ่ม Submit เพื่อนำเข้าแฟ้มเอกซ์เอ็มแอลของกระแสงานยอว์ลเข้าสู่ระบบ โดยในที่นี้ใช้แฟ้มของรูปที่ 3.2



รูปที่ 3.9 หน้าส่วนต่อประสานกับผู้ใช้งานในการเข้าในไฟล์กระแสงานยอว์ล

2) แปลงข้อมูลแฟ้มเอกซ์เอ็มแอลของกระแสงานยอว์ลให้อยู่ในรูปแบบ Dataframe

ในส่วนต่อมานั้น เป็นการดึงข้อมูลจากแฟ้มเอกซ์เอ็มแอลของกระแสงานยอว์ลให้อยู่ในรูป Dataframe โดย Dataframe จะเก็บตัวแปรแต่ละตัวไว้ในรูปของคอลัมน์ และค่าของตัวแปรแต่ละชุด (Observation) จะถูกเก็บไว้ในรูปของแถว เพื่อนำไปใช้ในการสร้างองค์ประกอบของไทมด์ออโตมาตาต่อไป ดังรูปที่ 3.9 ซึ่งอธิบายถึง Dataframe ที่สร้างมาจากการวนดึงข้อมูลจากแท็กลูกของแท็ก processControlElement ที่ละแท็กโดยคอลัมน์ชื่องานหรือกระบวนการ (process) คือ คุณลักษณะ id ของแท็ก inputCondition , task, outputCondition, คอลัมน์งานถัดไป (edge_out) คือ คุณลักษณะ id ของแท็ก nextElementRef, คอลัมน์ประเภทของงานขาเข้า (join) คือ คุณลักษณะ code ของแท็ก join และ คอลัมน์ประเภทของงานขาออก (split) คือ คุณลักษณะ code ของ tag join

	process name	edge_in	edge_out	join	split
0	InputCondition	[]	[p_a]		
1	p_a	[]	[p_c, p_b]	xor	and
2	p_b	[]	[p_d, p_e, p_f]	xor	and
3	p_c	[]	[p_e, p_f]	xor	and
4	p_d	[]	[p_g]	xor	and
5	p_e	[]	[p_g]	and	and
6	p_f	[]	[p_g]	and	xor
7	p_g	[]	[OutputCondition]	and	and
8	OutputCondition	[]	[]		

รูปที่ 3.9 Dataframe แสดงข้อมูลของแท็กลูกของแท็ก processControlElement

หลังจากได้ Dataframe ดังรูปที่ 3.9 แล้วทำการวนดังขึ้นมูลจากคอลัมน์งานถัดไป (edge_out) เพื่อสร้างข้อมูลในคอลัมน์งานก่อนหน้า (Edge in) เข้าไปใน Dataframe ดังรูปที่ 3.10 โดยกระบวนการ p_a จะอยู่ในคอลัมน์งานถัดไป (edge out) ของกระบวนการ InputCondition เท่านั้น ทำให้คอลัมน์งานขาเข้า (edge_in) ของกระบวนการ p_a คือ InputCondition เป็นต้น

	process name		edge_in	edge_out	join	split
0	InputCondition		[]	[p_a]		
1	p_a	p_a	[InputCondition]	[p_c, p_b]	xor	and
2	p_b	p_b	[p_a]	[p_d, p_e, p_f]	xor	and
3	p_c	p_c	[p_a]	[p_e, p_f]	xor	and
4	p_d	p_d	[p_b]	[p_g]	xor	and
5	p_e	p_e	[p_b, p_c]	[p_g]	and	and
6	p_f	p_f	[p_b, p_c]	[p_g]	and	xor
7	p_g	p_g	[p_d, p_e, p_f]	[OutputCondition]	and	and
8	OutputCondition		[p_g]	[]		

รูปที่ 3.10 Dataframe แสดงข้อมูลของกระแสงานยอว์ลหลังเพิ่มข้อมูลในคอลัมน์ edge_in

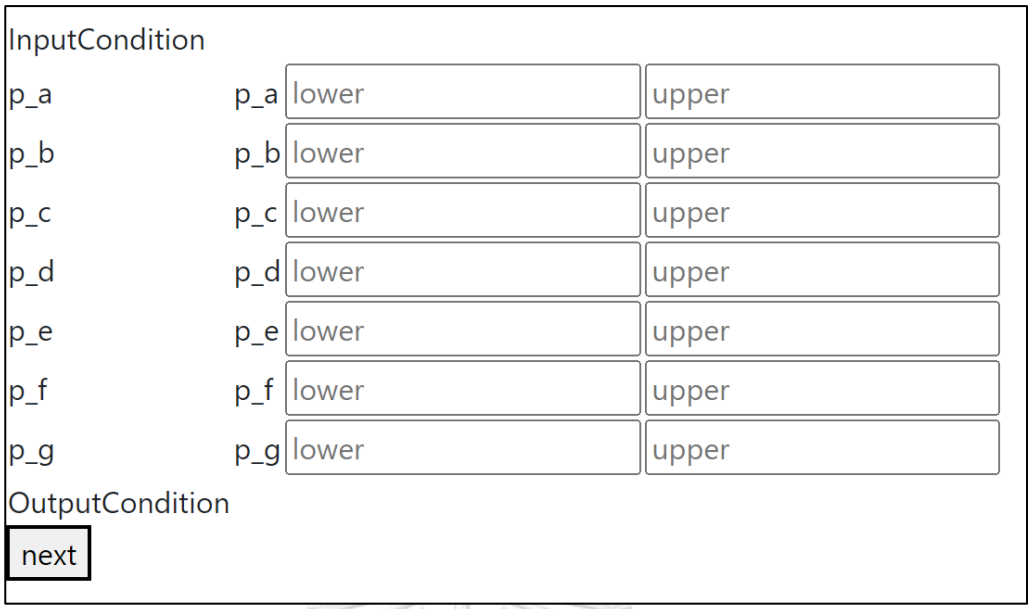
3) นำเข้าเงื่อนไขข้อจำกัดช่วงเวลาของกระแสงานยอว์ลเพื่อนำมากำกับใน Dataframe

หลังจากเปลี่ยนแฟ้มเอกซ์เอ็มแอลของกระแสงานยอว์ลให้อยู่ในรูป Dataframe แล้วทำการส่ง Dataframe กลับไปแสดงผลเพื่อรับข้อจำกัดด้านเวลาในแต่ละกระแสงานยอว์ล ดังรูปที่ 3.11 โดยจะมีการแสดง Dataframe ของกระแสงานยอว์ล และช่องว่างให้กรอกข้อมูลข้อจำกัดช่วงเวลาของแต่ละงานทั้ง lower และ upper

show all process!

Process	Edge in	Edge out	Join	Split
InputCondition		p_a		
p_a	InputCondition	p_c,p_b	xor	and
p_b	p_a	p_d,p_e,p_f	xor	and
p_c	p_a	p_e,p_f	xor	and
p_d	p_b	p_g	xor	and
p_e	p_b,p_c	p_g	and	and
p_f	p_b,p_c	p_g	and	xor
p_g	p_d,p_e,p_f	OutputCondition	and	and
OutputCondition	p_g			

รูปที่ 3.11 หน้าส่วนต่อประสานกับผู้ใช้งานในการเพิ่มข้อจำกัดช่วงเวลาเข้าไปในกระแสงานยอว์ล



รูปที่ 3.11 หน้าส่วนต่อประสานกับผู้ใช้งานในการเพิ่มข้อจำกัดช่วงเวลาเข้าไปในกระแสงานยอว์ล (ต่อ)

ในส่วนของต่อประสานนั้นจะมีข้อมูลรายละเอียดของกระแสงานยอว์ลอยู่ โดยมีช่องให้กรอกข้อจำกัดช่วงเวลาในรูปแบบของ [lower,upper] เพื่อเพิ่มข้อจำกัดช่วงเวลาของแต่ละงานยอว์ลเข้าไปใน Dataframe ที่ใช้ในการสร้างแฟ้มเอกซ์เอ็มแอลของไทมด์ออโตมาตาต่อไป โดยหลังจากรับค่าเรียบร้อยแล้ว และกดปุ่ม next จะได้ผล Dataframe ดังรูปที่ 3.12 โดยในรูปนั้นจะมีการเพิ่มคอลัมน์ upper และ lower ของแต่ละกระบวนการเข้าไปใน Dataframe

	process name		edge_in	edge_out	join	split	lower	upper
0	InputCondition		[]	[p_a]			NaN	NaN
1	p_a	p_a	[InputCondition]	[p_c, p_b]	xor	and	2	10
2	p_b	p_b	[p_a]	[p_d, p_e, p_f]	xor	and	2	10
3	p_c	p_c	[p_a]	[p_e, p_f]	xor	and	3	11
4	p_d	p_d	[p_b]	[p_g]	xor	and	2	10
5	p_e	p_e	[p_b, p_c]	[p_g]	and	and	2	5
6	p_f	p_f	[p_b, p_c]	[p_g]	and	xor	1	3
7	p_g	p_g	[p_d, p_e, p_f]	[OutputCondition]	and	and	2	3
8	OutputCondition		[p_g]	[]			NaN	NaN

รูปที่ 3.12 Dataframe แสดงข้อมูลของกระแสงานยอว์ลหลังจากเพิ่มข้อมูลข้อจำกัดช่วงเวลา

4) ทำการสร้างไทมด์ออโตมาตาตามกฎการแปลง

ในกระบวนการนี้ระบบจะทำการสร้างแฟ้มเอกซ์เอ็มแอลของไทมด์ออโตมาตาโดยเริ่มจากแท็ก nta และแท็ก declaration และหัวเอกสารแฟ้มเอกซ์เอ็มแอลของดังรายละเอียดในหัวข้อ 3.2 รูปที่ 3.7 หลังจากนั้นสร้าง

แท็ก template ของทุกแถวในคอลัมน์งานหรือกระบวนการ (process) โดยทำการวนสร้างแท็ก template ที่ละแถวดังรายละเอียดในหัวข้อ 3.2 รูปที่ 3.7 โดยใช้ข้อมูล Dataframe ของกระแสงานยอว์ล และกฎการแปลงที่ได้ถูกอธิบายในหัวข้อที่ 3.3

5) นำออกผลลัพธ์ของการแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาไปเป็นไทมด์ออโตมาตา

หลังจากสร้างไทมด์ออโตมาตาตามกฎการแปลงเรียบร้อยแล้ว จะถูกส่งกลับไปยังหน้าส่วนต่อประสานกับผู้ใช้งานสำหรับการโหลดแฟ้มเอกซ์เอ็มแอลของไทมด์ออโตมาตา ดังรูปที่ 3.13 โดยจะแสดงข้อมูลของกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลา และปุ่ม download เพื่อนำออกแฟ้มเอกซ์เอ็มแอลของไทมด์ออโตมาตาที่สามารถเปิดผ่านเครื่องมือ UPPAAL เพื่อใช้ในการทวนสอบ และจำลองต่อไป

Download!

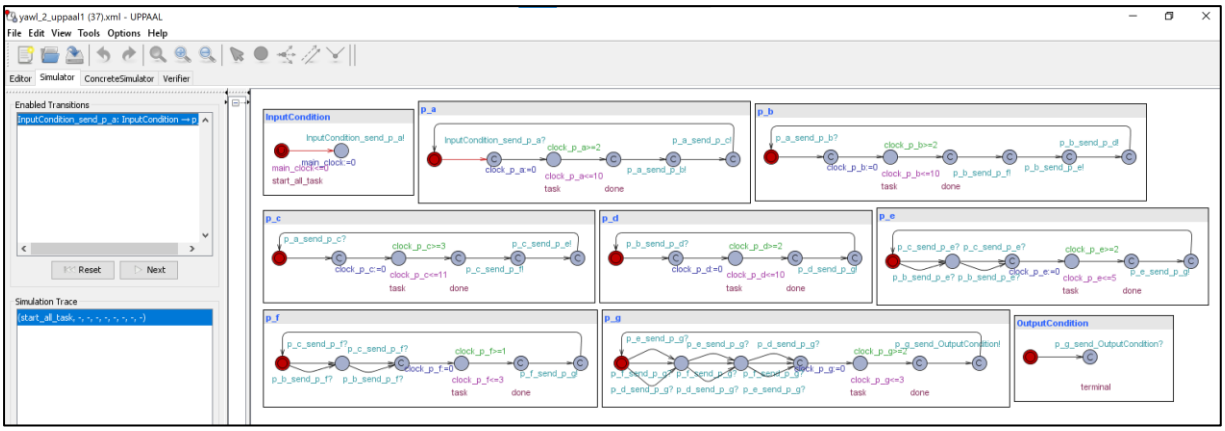
Process	Edge in	Edge out	Join	Split	Lower	Upper
InputCondition		p_a			NaN	NaN
p_a	InputCondition	p_c,p_b	xor	and	2	10
p_b	p_a	p_d,p_e,p_f	xor	and	2	10
p_c	p_a	p_e,p_f	xor	and	3	11
p_d	p_b	p_g	xor	and	2	10
p_e	p_b,p_c	p_g	and	and	2	5
p_f	p_b,p_c	p_g	and	xor	1	3
p_g	p_d,p_e,p_f	OutputCondition	and	and	2	3
OutputCondition	p_g				NaN	NaN

download

รูปที่ 3.13 หน้าส่วนต่อประสานกับผู้ใช้งานสำหรับการโหลดผลลัพธ์ของการแปลง

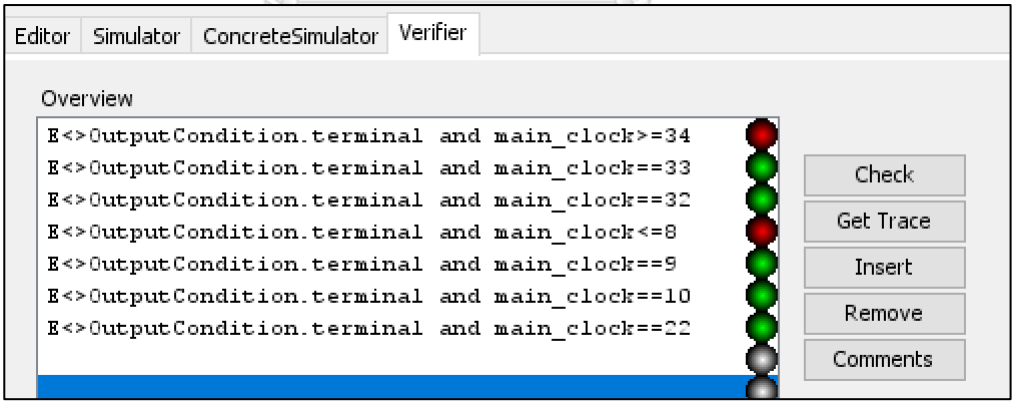
3.5 การทวนสอบผลลัพธ์ไทมด์ออโตมาตาใน UPPAAL [15]

การตรวจสอบความถูกต้องของแฟ้มเอกซ์เอ็มแอลผลลัพธ์หลักจากที่ได้รับการแปลง ผู้วิจัยจะนำแฟ้มเอกซ์เอ็มแอลเข้ามาในเครื่องมือ UPPAAL โดยใช้ฟังก์ชัน Simulator ในเครื่องมือโดยสังเกตพฤติกรรมของแต่ละงานเทียบกับกระแสงานยอว์ลของรูปที่ 3.2 ได้ดังรูปที่ 3.14 ซึ่งสามารถจำลองการเปลี่ยนแปลงผ่านปุ่ม Next โดยสามารถเลือก Enabled Transitions ได้ว่าจะให้เปลี่ยนผ่านไปยังเส้นทางใด และส่วนต่อไปจะอธิบายถึงการทวนสอบประสิทธิภาพด้านเวลาโดยใช้ฟังก์ชัน Verifier



รูปที่ 3.14 ตัวอย่างหน้าจอแสดงการจำลองพฤติกรรมของเครื่องมือ UPPAAL

การทวนสอบคุณสมบัติด้านต่าง ๆ โดยใช้ฟังก์ชัน Verifier ของกระแสงานนั้นสามารถแสดงดังรูปที่ 3.15 ซึ่งเป็นการทวนสอบแบบ Model checker จะกระทำในรูปแบบของแบบสอบถาม (Query) ของตรรกศาสตร์ต้นไม้การคำนวณเวลา (Timed computation tree logic : TCTL) โดยผลลัพธ์นั้นจะอยู่ในรูปแบบจริงแสดงวงกลมเป็นสีเขียวหรือเท็จแสดงในวงกลมเป็นสีแดง เช่น จากบรรทัดที่ 1 แบบสอบถามคือ E<>OutputCondition.terminal and main_clock >=34 หรือหมายความว่าบางเส้นทางแล้วในที่สุด ตำแหน่ง terminal ในไทมด์ออโตมาตา OutputCondition นั้นจะเป็นจริง และนาฬิกา main_clock ต้องมากกว่าหรือเท่ากับ 34 หน่วย โดยผลการทวนสอบพบว่าแสดงเป็นสีแดงหรือหมายถึงว่าเป็นเท็จ ไม่มีกรณีใดเลยที่ นาฬิกา main_clock จะมากกว่า 34 และตำแหน่ง terminal ในไทมด์ออโตมาตา OutputCondition นั้นเป็นจริงพร้อมกัน เป็นต้น



รูปที่ 3.15 หน้าจอแสดงการทวนสอบของเครื่องมือ UPPAAL

ซึ่งจากรูปที่ 3.15 นั้นเป็นการทวนสอบเครื่องมือในลักษณะ Robust boundary value testing ของรูปที่ 3.2 ซึ่งกระแสงานยอว์ลจะใช้เวลาในการเสร็จสิ้นกระแสงานอยู่ระหว่าง main_clock = [9,33] พบว่าผลลัพธ์ของเครื่องมือการแปลงมีความถูกต้องในด้านการจับเวลาการทำงานของกระแสงานยอว์ล โดยการทวนสอบด้านพฤติกรรมนั้นจะทวนสอบในกรณีศึกษาต่อไป

บทที่ 4
การทวนสอบแบบจำลองด้วยกรณีศึกษา

ในการทวนสอบการแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาไปเป็นไทมด์ออโตมาตา และกฎการแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาไปเป็นไทมด์ออโตมาตา เพื่อตรวจสอบว่าสามารถทำงานได้ตรงตามกฎการแปลงที่ได้ออกแบบไว้หรือไม่โดยจะทำการทดสอบกับ 3 กรณีศึกษา โดยในส่วนแรกจะอธิบายถึงสภาพแวดล้อมที่ใช้ในการทดสอบ ตามมาด้วยแนวทางในการทดสอบและสุดท้ายเป็นการทดสอบผ่านกรณีศึกษาและประเมินผล

4.1 สภาพแวดล้อมที่ใช้ในการทดสอบ

4.1.1 ฮาร์ดแวร์ (Hardware)

1) โน้ตบุ๊ค หน่วยประมวลผลอินเทลคอร์ไอ 9 เจนเนอร์เรชั่น 9 2.4 กิกะเฮิร์ท (Intel Core i9 9 th Gen 2.4 GHz)
2) หน่วยความจำสำรอง (RAM) 32.0 กิกะไบต์ (32.0 GB)
3) SSD 186 กิกะไบต์ (186 GB)

4.1.2 ซอฟต์แวร์ (Software)

1) ระบบปฏิบัติการ (Windows OS) Windows 10 Home
2) เครื่องมือยอว์ล 4.3.1
3) เครื่องมือ UPPAAL 4.1.24

4.2 แนวทางในการทดสอบ

แนวทางการทดสอบการแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาไปเป็นไทมด์ออโตมาตามีขั้นตอนการทดสอบดังนี้

1) ทดสอบการแปลงแบบจำลองกระแสงานยอว์ลไปเป็นไทมด์ออโตมาตาผ่านเว็บแอปพลิเคชัน ตรวจสอบความสมบูรณ์ของสัญลักษณ์ของไทมด์ออโตมาตาหลังการแปลงจากกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลา โดยมีขั้นตอนดังนี้

- นำเข้าแฟ้มเอกซ์เอ็มกระแสงานยอว์ลในเว็บแอปพลิเคชัน
- ผู้ใช้ระบุข้อมูลข้อจำกัดช่วงเวลา
- โหลดแฟ้มเอกซ์เอ็มแอลไทมด์ออโตมาตาที่ได้จากการแปลงนำไปเปิดบนเครื่องมือ UPPAAL 4.1.24 ตรวจสอบความสมบูรณ์ของสัญลักษณ์ และเส้นเชื่อมต่าง ๆ ของไทมด์ออโตมาตา

2) ตรวจสอบความถูกต้องของพฤติกรรมผ่าน Simulation และ Model Checker ดูว่าพฤติกรรมของไทมด์ออโตมาตาสามารถจำลองให้ครบทุกงานตามกระแสงานยอว์ล ที่สร้างจากแบบจำลองบนเครื่องมือยอว์ล ได้หรือไม่ในลักษณะครอบคลุมทุกกิ่งก้านที่เป็นไปได้ และทุกงานสามารถเกิดขึ้นได้อย่างน้อย 1 กรณี (Branch testing)

3) ยกตัวอย่างการวิเคราะห์ทวนสอบประสิทธิภาพด้านเวลาของไทมด์ออโตมาตาผ่าน Model Checker บนฟังก์ชัน Verifier บนเครื่องมือ UPPAAL 4.1.24 โดยอย่างน้อยที่สุดสามารถทราบระยะเวลาที่สั้นที่สุดของกระแสงานยอว์ลได้

4) สรุปผลการทดสอบ

4.3 การทดสอบและประเมิน

ทดสอบโดยใช้กรณีศึกษาพร้อมทั้งตรวจสอบการทำงานของผลลัพธ์การแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาไปเป็นไทมด์ออโตมาตา 3 กรณีศึกษาดังนี้

1) กรณีศึกษาการจัดซื้อ
2) กรณีศึกษาการเสนอขายสินค้า
3) กรณีศึกษาการสมัครบัตรเครดิต

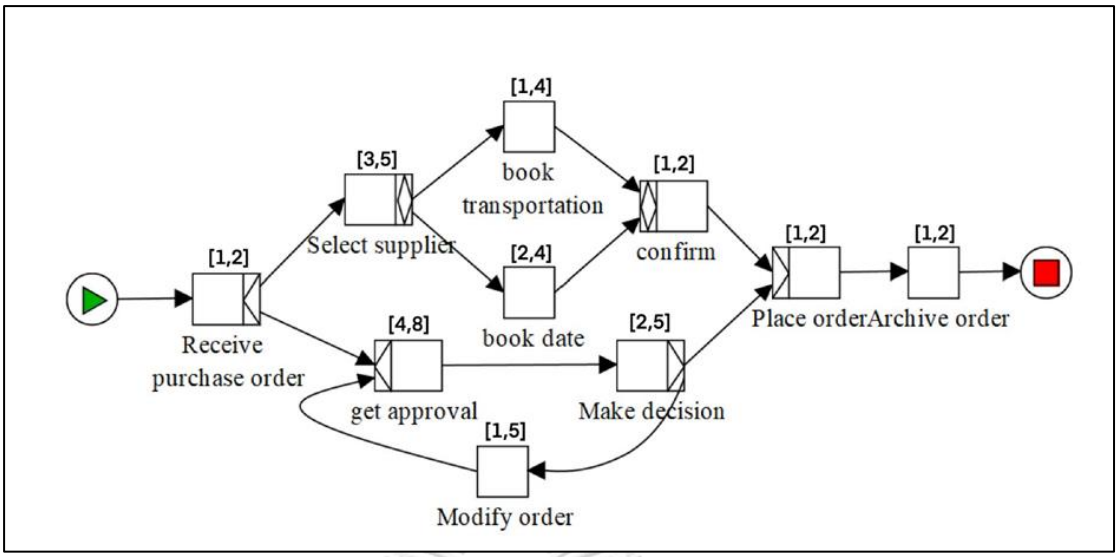
4.3.1 กรณีศึกษาที่ 1 กระแสงานยอว์ลเรื่องการจัดซื้อ

ในกรณีศึกษาที่ 1 กระแสงานยอว์ลเรื่องการจัดซื้อนั้นจะเป็นการจำลองการจัดซื้อสินค้าต่าง ๆ ผ่านระบบในโรงงานโดยต้องได้รับการอนุมัติก่อนถึงจะสามารถจัดซื้อได้ และสามารถเลือกระบุวิธีการส่งกับช่วงเวลาส่งได้อย่างใดอย่างหนึ่งหรือทั้งสองอย่างได้ โดยในกรณีศึกษานั้นจะมีขั้นตอนการศึกษาคือการสร้างแบบจำลองกระแสงานยอว์ลและแปลงไปเป็นไทมด์ออโตมาตาแบบมีข้อจำกัดช่วงเวลาหลังจากนั้นทำการตรวจสอบสอบพฤติกรรมผ่าน Simulation และ Model Checker เพื่อยืนยันว่าไทมด์ออโตมาตานั้นมีพฤติกรรมการทำงานสอดคล้องกับกระแสงานยอว์ล ต่อมาทำการวิเคราะห์ทวนสอบประสิทธิภาพด้านเวลา และสรุปผลของกรณีศึกษา ซึ่งอธิบายเพิ่มเติมดังนี้

4.3.1.1 การสร้างแบบจำลองกระแสงานยอว์ลและแปลงไปเป็นไทมด์ออโตมาตา

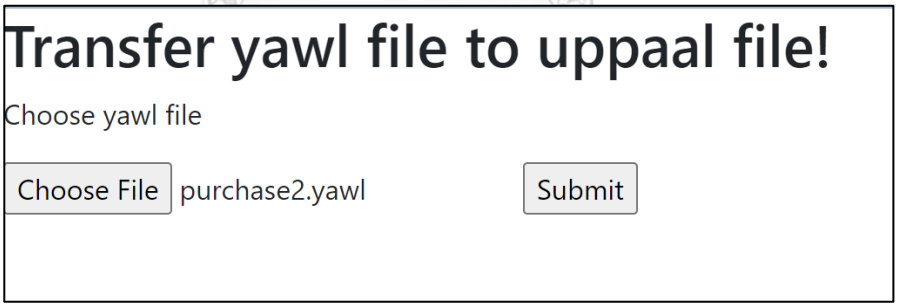
การสร้างแบบจำลองกระแสงานยอว์ล และแปลงไปเป็นไทมด์ออโตมาตา มีทั้งหมด 5 ขั้นตอนตามหัวข้อที่ 3.4 คือสร้างแบบจำลองผ่านเครื่องมือยอว์ล 4.3.1 ต่อมานำแฟ้มเอกซ์เอ็มแอลของกระแสงานยอว์ลเข้าเว็บแอปพลิเคชันหลังจากนั้นทำการใส่ข้อมูลข้อจำกัดช่วงเวลาแล้วนำออกเพื่อนำแฟ้มเอกซ์เอ็มแอลของไทมด์ออโตมาตามาตรวจสอบความสมบูรณ์ของสัญลักษณ์ของไทมด์ออโตมาตาหลังการแปลง และทวนสอบในเครื่องมือ UPPAAL ต่อไป โดยเหน่วยของเวลาเป็นชั่วโมง สามารถอธิบายเพิ่มเติมดังนี้

1) สร้างแบบจำลองการจัดซื้อ โดยใช้เครื่องมือยอว์ล 4.3.1 ดังรูปที่ 4.1 โดยงาน Receive purchase order นั้นจะมี AND split ซึ่งหลังจากเสร็จสิ้นงานนี้แล้วจะถูกส่งไปให้ทั้ง Select supplier และ Get approval ทำต่อโดย Select Supplier จะมี OR split และมี OR join ที่งาน Confirm ซึ่งที่งาน Get approval จะมี XOR join และงาน Make decision จะมี XOR split ทำให้สามารถเกิดการทำงานแบบวนซ้ำได้ โดยหลังจากหลุดจาก Loop แล้วจะมี AND join รวมงานเข้าด้วยกันที่งาน Place order ก่อนจบกระแสงานที่ Archive order



รูปที่ 4.1 กระแสงานยอว์ลการจัดซื้อ บนเครื่องมือยอว์ล 4.3.1

2) ทำการบันทึกแล้วนำเข้าแฟ้มเอกซ์เอ็มแอลของกระแสงานยอว์ลที่ได้จากเครื่องมือยอว์ล4.3.1 เข้ามายังเว็บแอปพลิเคชัน ดังรูปที่ 4.2 โดยกดปุ่ม Choose File เพื่อเลือกแฟ้มเอกซ์เอ็มแอล หลังจากนั้นกด Submit เพื่อนำแฟ้มเอกซ์เอ็มแอลเข้าระบบต่อไป



รูปที่ 4.2 การเลือกไฟล์ยอว์ลการจัดซื้อ เข้ามายังเว็บแอปพลิเคชัน

3) หลังจากที่กดปุ่ม Submit ตามรูปที่ 4.2 แล้วเว็บแอปพลิเคชันจะแสดงข้อมูลกระแสงานยอว์ลการจัดซื้อ กระบวนการทั้งหมด 12 กระบวนการ มีการแสดงข้อมูล Edge_in, Edge_out, ประเภทของ Join และ Split โดยผู้ใช้จะระบุข้อจำกัดช่วงเวลาของแต่ละงาน ดังรูปที่ 4.3 ซึ่งผู้ใช้งานมีการระบุข้อจำกัดช่วงเวลาครบทุกงานเป็นจำนวนเต็มแล้วกดปุ่ม next

show all process!

Process	Edge in	Edge out	Join	Split
InputCondition		Receive_purchase_order		
Receive_purchase_order	InputCondition	Select_supplier,get_approval	xor	and
Select_supplier	Receive_purchase_order	book_date,book_transportation	xor	or
get_approval	Receive_purchase_order,Modify_order	Make_decision	xor	and
Make_decision	get_approval	Place_order,Modify_order	xor	xor
book_date	Select_supplier	confirm	xor	and
book_transportation	Select_supplier	confirm	xor	and
Modify_order	Make_decision	get_approval	xor	and
Place_order	Make_decision	Archive_order	xor	and
confirm	book_date,book_transportation	Archive_order	or	and
Archive_order	Place_order,confirm	OutputCondition	and	and
OutputCondition	Archive_order			

next

InputCondition

Receive_purchase_order	1	2
Select_supplier	3	5
get_approval	4	8
Make_decision	2	5
book_date	2	4
book_transportation	1	4
Modify_order	1	5
Place_order	1	2
confirm	1	2
Archive_order	1	2

OutputCondition

next

รูปที่ 4.3 ระบุข้อจำกัดช่วงเวลาของแต่ละงานของกระแสงานยอว์ลการจัดซื้อ

4) หลังจากกดปุ่ม next ในรูปที่ 4.3 จากนั้นตรวจสอบข้อจำกัดช่วงเวลาของแต่ละงานแล้วกดปุ่ม Download ตามรูปที่ 4.4 ซึ่งจะแสดงข้อมูลกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาในรูปแบบ Dataframe โดยมีคอลัมน์ Process, Name, Edge_in, Edge_out, Join, Split, Lower, Upper และปุ่ม Download เพื่อโหลดไฟล์ผลลัพธ์การแปลงนำเข้าไปยัง เครื่องมือ UPPAAL 4.1.24 ต่อไป

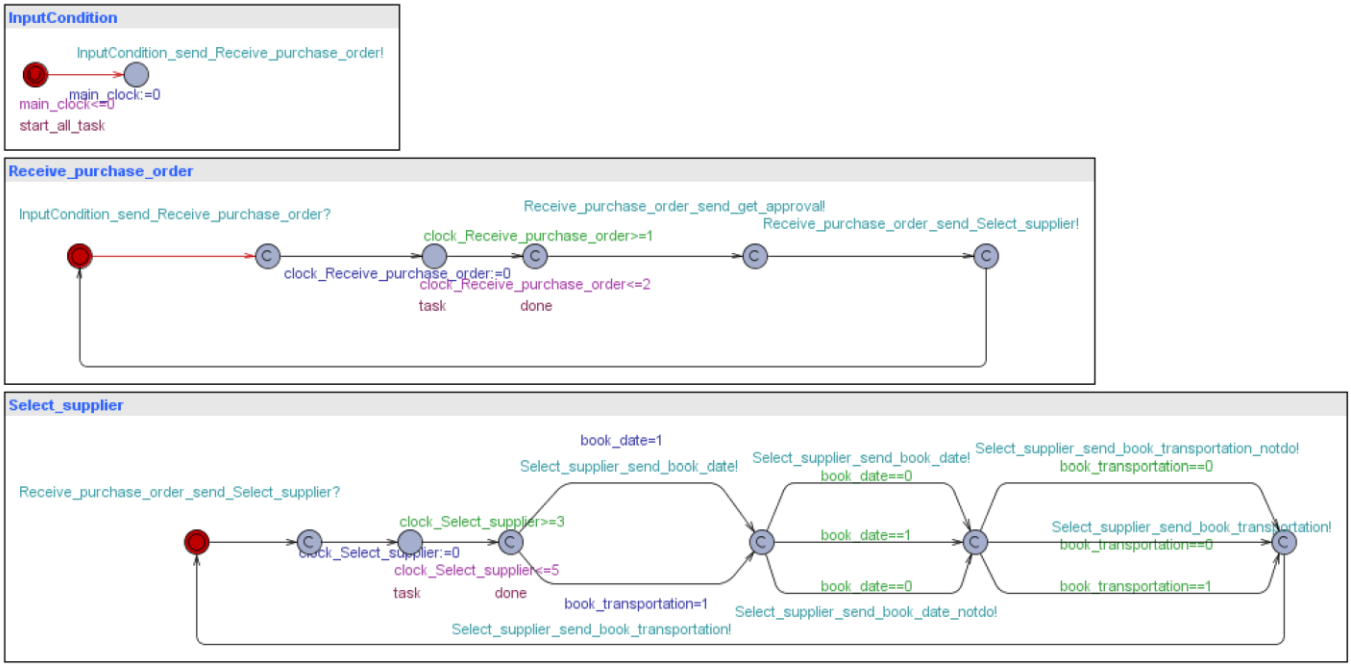
Download!

Process	Edge in	Edge out	Join	Split	Lower	Upper
InputCondition		Receive_purchase_order			NaN	NaN
Receive_purchase_order	InputCondition	Select_supplier,get_approval	xor	and	1	2
Select_supplier	Receive_purchase_order	book_date,book_transportation	xor	or	3	5
get_approval	Receive_purchase_order,Modify_order	Make_decision	xor	and	4	8
Make_decision	get_approval	Place_order,Modify_order	xor	xor	2	5
book_date	Select_supplier	confirm	xor	and	2	4
book_transportation	Select_supplier	confirm	xor	and	1	4
Modify_order	Make_decision	get_approval	xor	and	1	5
Place_order	Make_decision	Archive_order	xor	and	1	2
confirm	book_date,book_transportation	Archive_order	or	and	1	2
Archive_order	Place_order,confirm	OutputCondition	and	and	1	2
OutputCondition	Archive_order				NaN	NaN

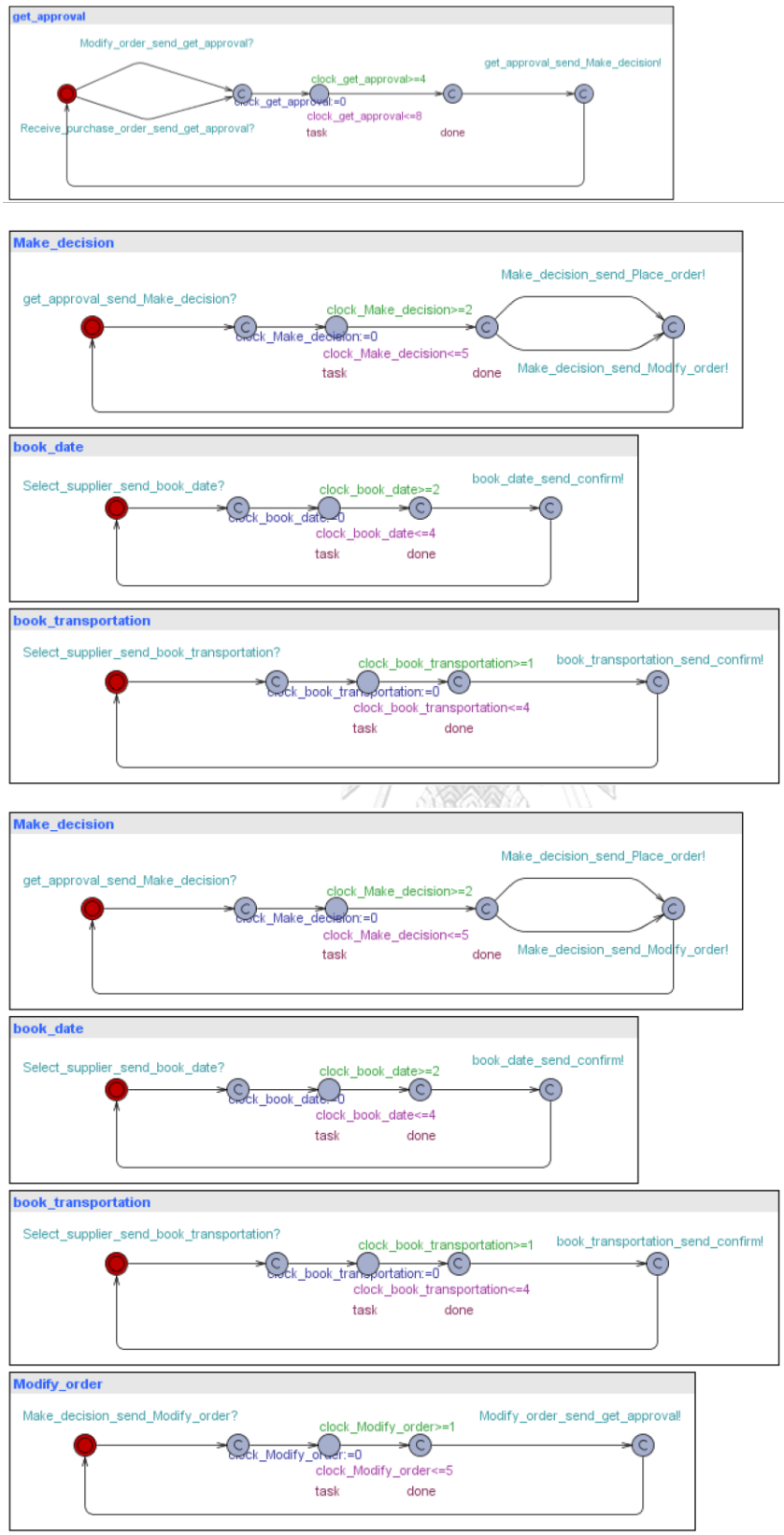
download

รูปที่ 4.4 ข้อมูลกระแสงานยอว์ลการจัดซื้อ และข้อจำกัดช่วงเวลาก่อนทำการแปลงเพื่อโหลดไฟล์ผลลัพธ์การแปลง

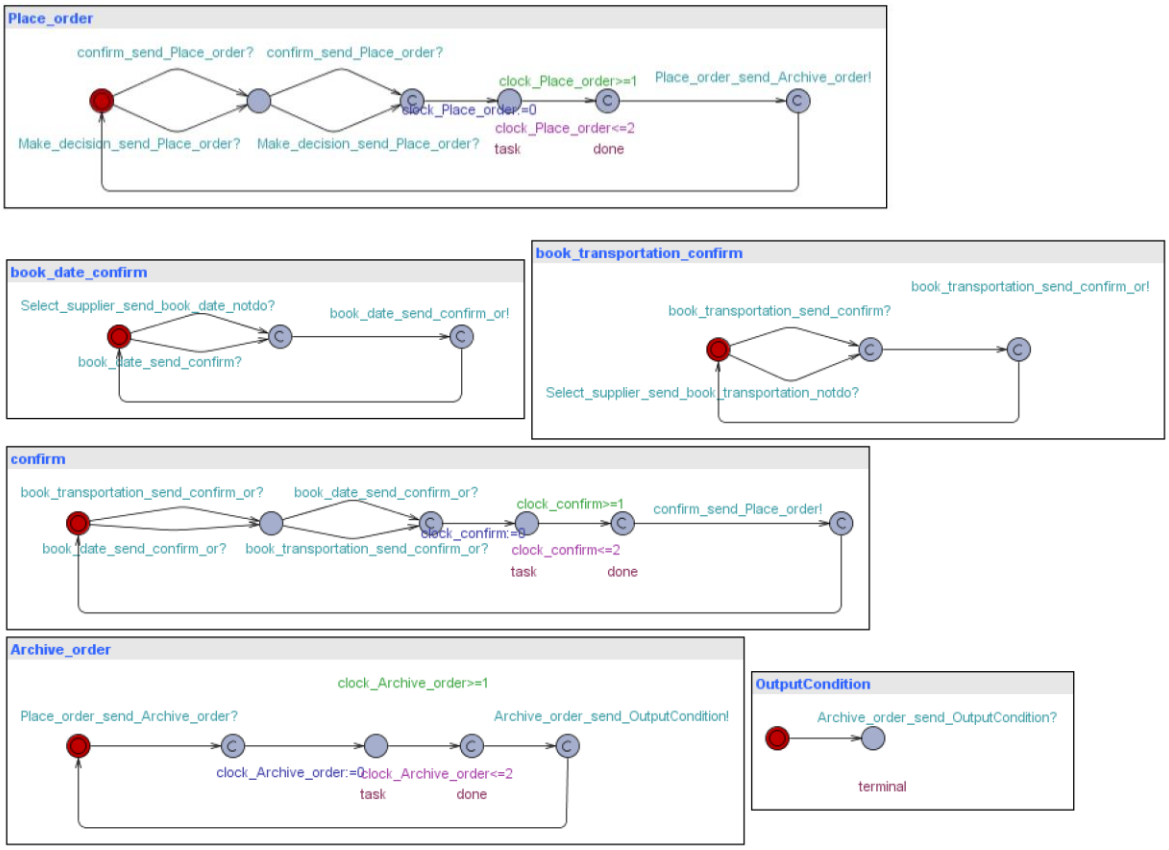
5) นำแฟ้มเอกซ์เอ็มแอลที่ได้จากการแปลงเปิดในเครื่องมือ UPPAAL 4.1.24 แล้วปรับแต่งเส้นเชื่อมตามที่ต้องการดังรูปที่ 4.5 จะเห็นได้ว่ามี ทั้งหมด 14 ไทมด์ออโตมาตา ซึ่งเกิดจาก InputCondition, OutputCondition และแต่ละงานของกระแสงานยอว์ล อย่างละ 1 รวมเป็น 12 ไทมด์ออโตมาตา และมีไทมด์ออโตมาตาพิเศษที่เกิดจาก OR split อีก 2 งานรวมเป็น 14 ไทมด์ออโตมาตา



รูปที่ 4.5 ไทมด์ออโตมาตาของการจัดซื้อ บนเครื่องมือ UPPAAL 4.1.24



รูปที่ 4.5 ไทมด์ออโตมาตาของการจัดซื้อ บนเครื่องมือ UPPAAL 4.1.24 (ต่อ)



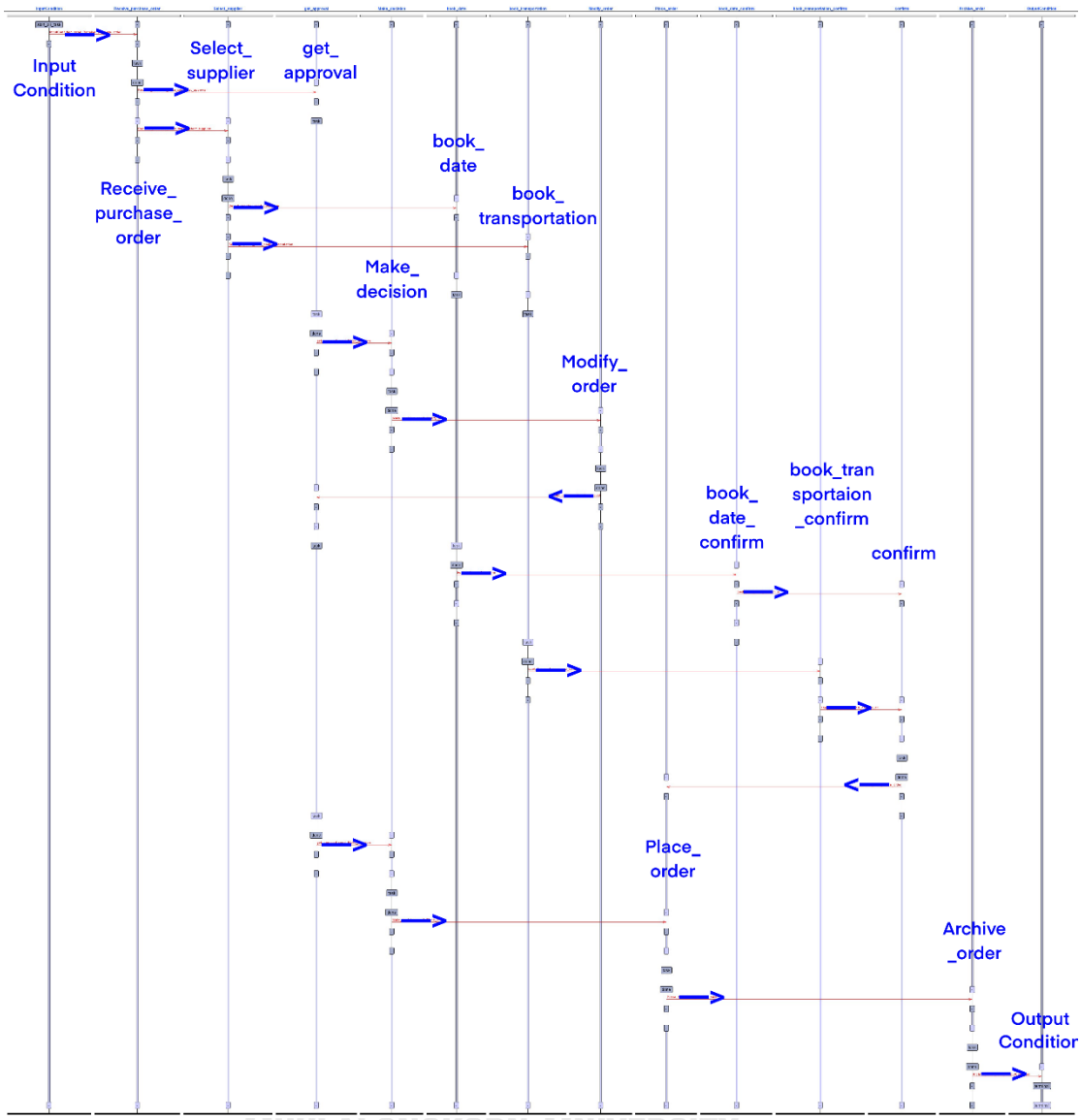
รูปที่ 4.5 ไทมด์ออโตมาตาของการจัดซื้อ บนเครื่องมือ UPPAAL 4.1.24 (ต่อ)

4.3.1.2 การตรวจสอบความถูกต้องของพฤติกรรมผ่าน Simulation และ Model Checker

ส่วนต่อมาหลังจากทำการแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาไปเป็นไทมด์ออโตมาตาแล้วจะทำการตรวจสอบสอบพฤติกรรมผ่าน Simulation และ Model Checker บนเครื่องมือ UPPAAL 4.1.24 เพื่อดูว่ามีไทมด์ออโตมาตาที่ได้จากการแปลงมีพฤติกรรมสอดคล้องกับกระแสงานยอว์ลหรือไม่ ในลักษณะครอบคลุมทุกกิ่งก้านที่เป็นไปได้ และทุกงานสามารถเกิดขึ้นได้อย่างน้อย 1 กรณี สามารถอธิบายเพิ่มเติมได้ดังต่อไปนี้

1) ตรวจสอบผ่าน Simulation บนเครื่องมือ UPPAAL 4.1.24

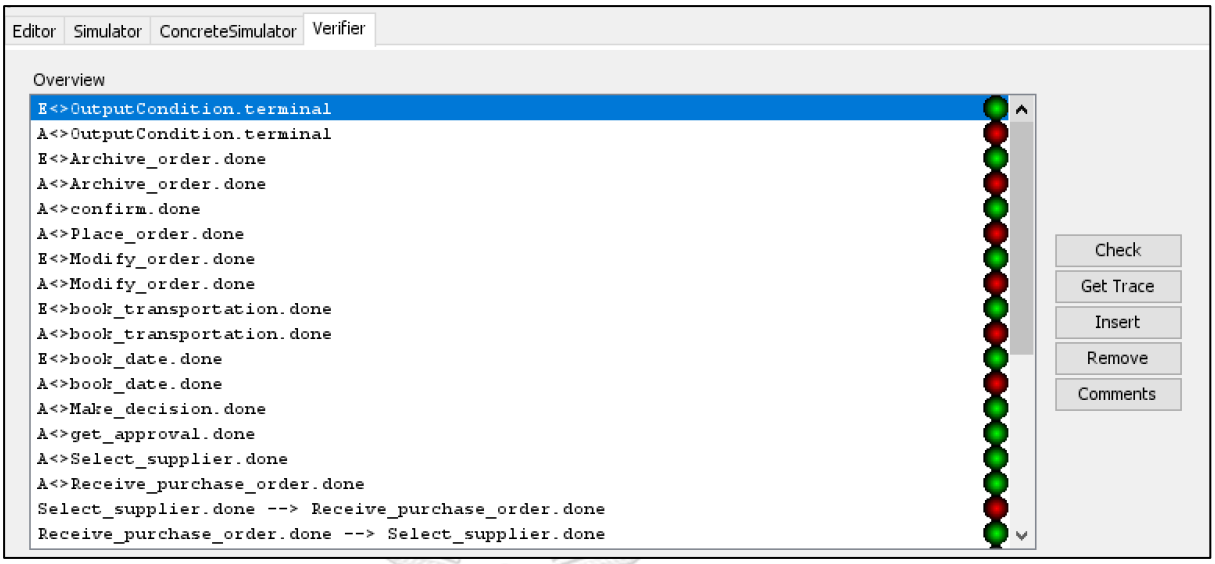
ทำการจำลองการทำงานให้ครบทุกงานครอบคลุมทุกกิ่งก้านที่เป็นไปได้ตามกระแสงานยอว์ลรูปที่ 4.1 พบว่าสามารถทำงานตามกระแสงานยอว์ลได้ครบทุกงาน เช่น Select_supplier มี OR split ส่งไปยัง Book_date หรือ Book_transportation ในที่นี้เลือกส่งทั้งสองงาน และ Get_approval มีการทำงานสองครั้ง ที่เกิดจากการวน Loop โดย Make_decision ที่มี XOR split ที่สามารถเลือกได้งานใดงานหนึ่งดังรูปที่ 4.6



รูปที่ 4.6 ผลการจำลองไทมด์ออโตมาตาของการจัดซื้อในเครื่องมือ UPPAAL 4.1.24

2) ตรวจสอบผ่าน Model Checker บนเครื่องมือ UPPAAL 4.1.24

ทำการทวนสอบทุกงาน โดยใช้สัญลักษณ์ A<> และ E<> เพื่อดูว่าทุกงานนั้นเกิดขึ้นในทุกเส้นทางหรือเกิดขึ้นบางเส้นทางของกระแสงานการจัดซื้อ โดยพบว่า Receive_purchase_order , Select_supplier, Get_approval, Make_decision, book_date, book_transportation, Modify_order, Place_order, Confirm, Archive_order สามารถเสร็จสิ้นได้อย่างน้อย 1 เส้นทางการทำงาน และ OutputCondition.terminal สามารถเกิดขึ้นได้ถ้าไม่เข้าเงื่อนไขวนลูปไม่รู้จบ และ Select_supplier ไม่สามารถเสร็จสิ้นก่อน Receive_purchase_order โดยสีเขียวหมายถึงเป็นจริง และที่แดงหมายถึงเป็นเท็จ ได้ดังรูปที่ 4.7 โดยสีเขียวหมายถึงเป็นจริง สีแดงหมายถึงเป็นเท็จ เช่น บรรทัดที่ 1 และ 2 พบว่าบางเส้นทางเท่านั้นที่ถึง OutputCondition



รูปที่ 4.7 ผลการทวนสอบพฤติกรรมของไทมด์ออโตมาตาของการจัดซื้อ

4.3.1.3 ตัวอย่างการวิเคราะห์ผลลัพธ์ผ่าน Model Checker บนเครื่องมือ UPPAAL

หลังจากทำการแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาไปเป็นไทมด์ออโตมาตา และตรวจสอบสอบพฤติกรรมผ่าน Simulation และ Model Checker แล้ว ส่วนนี้จะเป็นการทำการวิเคราะห์ประสิทธิภาพด้านเวลาของไทมด์ออโตมาตาผ่าน Model Checker ฟังก์ชัน Verifier บนเครื่องมือ UPPAAL ดังนี้ โดยผลลัพธ์ของการวิเคราะห์สามารถแสดงได้ดังในรูปที่ 4.8

1) มีการยืนยันข้อมูลการส่ง วันจัดส่ง ภายในเวลา 13 ชั่วโมงเสมอ สามารถตรวจสอบได้ดังนี้

ในทุกเส้นทางที่เป็นไปได้ Confirm จะเสร็จสิ้นระหว่าง main_clock == [6,13] สามารถเขียนตรรกศาสตร์ต้นไม้การคำนวณเวลาได้คือ A[]confirm.done imply (main_clock>=6 and main_clock<=13) พบว่าผลลัพธ์เป็นสีเขียวหมายถึงเป็นจริง หรือหมายความว่าในทุกเส้นทางที่เป็นไปได้ Confirm ที่ตำแหน่ง Done จะถูกส่งต่อระหว่าง main_clock = [6,13]

2) มีโอกาสหรือไม่ ที่การยืนยันข้อมูลการส่ง วันจัดส่ง จะเสร็จสิ้นก่อน 6 ชั่วโมง สามารถตรวจสอบได้ดังนี้

ในทุกเส้นทางที่เป็นไปได้ Confirm จะไม่สามารถเสร็จสิ้นก่อน main_clock == 6 ได้ สามารถเขียนตรรกศาสตร์ต้นไม้การคำนวณเวลา ได้คือ E<>confirm.done and main_clock<6 ผลลัพธ์เป็นเท็จ หรือหมายความว่า ไม่มีเส้นทางใดเลยที่ Confirm อยู่ที่ตำแหน่ง Done และเวลาน้อยกว่า 6

3) ไม่สามารถทำ Modify_order พร้อมกับ Place_order ได้จริงหรือไม่ สามารถตรวจสอบได้ดังนี้

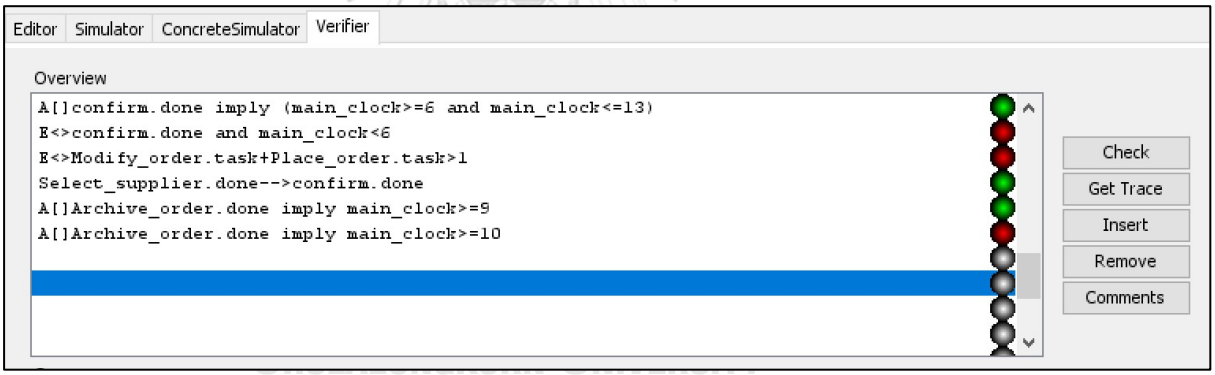
ในทุกเส้นทางที่เป็นไปได้ Modify_order กับ Place_order ไม่สามารถเกิดขึ้นพร้อมกันได้ สามารถเขียนตรรกศาสตร์ต้นไม้การคำนวณเวลา ได้คือ E<>Modify_order.task+Place_order.task>1 ผลลัพธ์เป็นเท็จ หมายความว่าในทุกเส้นทางที่เป็นไปได้ Modify_order กับ Place_order ไม่สามารถเกิดขึ้นพร้อมกันได้

4) สามารถยืนยันข้อมูลการส่ง วันจัดส่ง ได้หลังจากเลือกผู้ขายเท่านั้น สามารถตรวจสอบได้ดังนี้

ในทุกเส้นทางที่เป็นไปได้ Select_supplier ต้องเสร็จสิ้นก่อน Confirm เสมอ สามารถเขียนตรรกศาสตร์ต้นไม้การคำนวณเวลา ได้คือ Select_supplier.done --> Confirm.done ผลลัพธ์เป็นจริง หมายความว่าในทุกเส้นทางที่เป็นไปได้ Select_supplier ต้องเสร็จสิ้นก่อน confirm เสมอ

5) เป็นไปได้หรือไม่ ที่จะเสร็จสิ้นกระบวนการจัดซื้อทั้งหมดก่อน 10 ชั่วโมง เสมอสามารถตรวจสอบได้ดังนี้

ในทุกเส้นทางที่เป็นไปได้ Archive_order เสร็จสิ้นหลังจาก main_clock == 9 เสมอ สามารถเขียนตรรกศาสตร์ต้นไม้การคำนวณเวลา ได้คือ A[] Archive_order.done imply main_clock>=9 และ A[] Archive_order.done imply main_clock >=10 ผลลัพธ์เป็นจริง และเท็จตามลำดับ หมายความว่า ทุกเส้นทางที่เป็นไปได้ Archive_order ตำแหน่ง Done จะถูกส่งต่อที่เวลา มากกว่าหรือเท่ากับ 9 เท่านั้นเนื่องเพราะบรรทัดสุดท้ายเป็นเท็จเพราะมีบางกรณีเกิดขึ้นได้



รูปที่ 4.8 ผลการทวนสอบของไทมด์ออโตมาตาของการจัดซื้อ

4.3.1.4 สรุปผลกรณีศึกษาที่ 1 กระแสงานยอว์ลเรื่องการจัดซื้อ

จากกรณีศึกษาที่ 1 กระแสงานยอว์ลเรื่องการจัดซื้อ พบว่าสามารถแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาไปเป็นไทมด์ออโตมาตาผ่านเว็บแอปพลิเคชันได้ถูกต้องเส้นเชื่อมในไทมด์ออโตมาตาครบทุกตำแหน่ง มีครบทุกงานตามกระแสงานยอว์ล ต่อมาตรวจสอบสอบพฤติกรรมผ่าน Simulation และ Model Checker บนเครื่องมือ UPPAAL พบว่าไทมด์ออโตมาตาที่ได้จากการแปลงมีพฤติกรรมสอดคล้องกับกระแสงานยอว์ล ผลการทวนสอบประสิทธิภาพด้านเวลาพบว่า เส้นทางที่เร็วที่สุดที่สามารถจบงานของกระแสงานยอว์ลเรื่องการจัดซื้ออยู่ที่ main_clock == 9 และที่งาน Confirm มีสามารถจบงานเร็วที่สุดที่ main_clock == 6 และช้าที่สุดที่ main_clock == 13

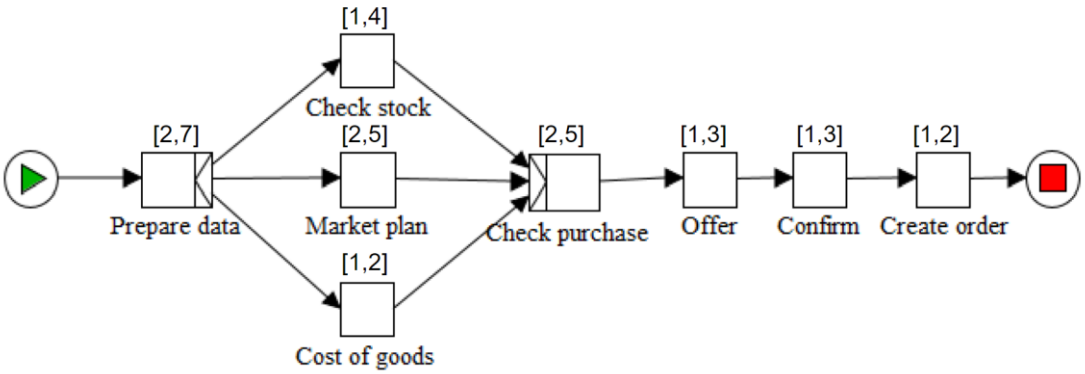
4.3.2 กรณีศึกษาที่ 2 กระแสงานยอว์ลเรื่องการเสนอขายสินค้า

กรณีศึกษาที่ 2 กระแสงานยอว์ลเรื่องการเสนอขายสินค้าของร้านค้าส่งนั้นจะมีการรวมรวบข้อมูลต่าง ๆ ทั้งด้านการตลาด ต้นทุนสินค้า จุดคุ้มทุน ก่อนที่จะออกไปนำเสนอขาย โดยในกรณีศึกษานั้นขั้นต่อการศึกษาคือการสร้างแบบจำลองกระแสงานยอว์ล และแปลงไปเป็นไทมด์ออโตมาตาแบบมีข้อจำกัดช่วงเวลาหลังจากนั้นทำการตรวจสอบสอบพฤติกรรมผ่าน Simulation และ Model Checker เพื่อยืนยันว่าไทมด์ออโตมาตานั้นมีพฤติกรรมการทำงานสอดคล้องกับกระแสงานยอว์ล ต่อมาทำการวิเคราะห์ทวนสอบประสิทธิภาพด้านเวลา และสรุปผลของกรณีศึกษา โดยกำหนดให้หน่วยของเวลาเป็นชั่วโมง ซึ่งอธิบายเพิ่มเติมดังนี้

4.3.2.1 การสร้างแบบจำลองกระแสงานยอว์ล และแปลงไปเป็นไทมด์ออโตมาตา

การสร้างแบบจำลองกระแสงานยอว์ล และแปลงไปเป็นไทมด์ออโตมาตา มีทั้งหมด 5 ขั้นตอนตามหัวข้อที่ 3.4 คือสร้างแบบจำลองผ่านเครื่องมือยอว์ล 4.3.1 ต่อมานำแฟ้มเอกซ์เอ็มแอลของกระแสงานยอว์ลเข้าเว็บแอปพลิเคชันหลังจากนั้นทำการใส่ข้อมูลข้อจำกัดช่วงเวลาแล้วนำออก เพื่อนำแฟ้มเอกซ์เอ็มแอลของไทมด์ออโตมาตามาตรวจสอบความสมบูรณ์ของสัญลักษณ์ของไทมด์ออโตมาตาหลังการแปลง และทวนสอบในเครื่องมือ UPPAAL ต่อไป สามารถอธิบายเพิ่มเติมดังนี้

1) สร้างแบบจำลองการจัดซื้อ โดยใช้เครื่องมือยอว์ล 4.3.1 ดังรูปที่ 4.9 โดยงาน Prepare data จะมี AND split ซึ่งหลังจากเสร็จสิ้นงานนี้แล้วจะถูกส่งไปให้ทั้ง Check stock, Market plan, Cost of goods ให้ทำงานพร้อมก่อนรวมข้อมูลส่งไปที่งาน Check purchase สรุปข้อมูลก่อนส่งไปยังงาน Offer เพื่อเสนอขายสินค้าต่อไป

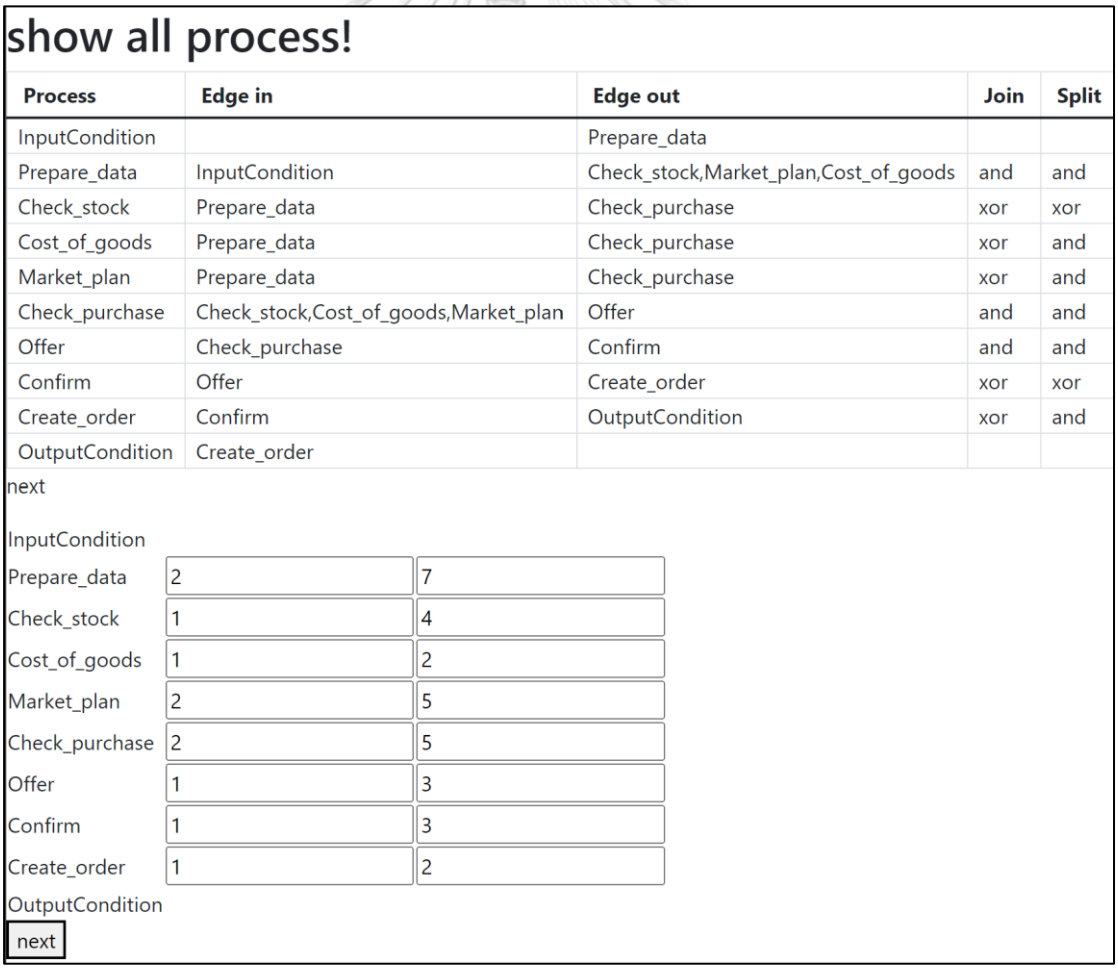


รูปที่ 4.9 กระแสงานยอว์ลการเสนอขายสินค้าบนเครื่องมือยอว์ล 4.3.1

2) ทำการบันทึกแล้วนำเข้าแฟ้มเอกซ์เอ็มแอลของกระแสงานยอว์ลที่ได้จากเครื่องมือยอว์ล 4.3.1 เข้ามายังเว็บแอปพลิเคชัน ดังรูปที่ 4.10 โดยกดปุ่ม Choose File เพื่อเลือกแฟ้มเอกซ์เอ็มแอล หลังจากนั้นกด Submit เพื่อนำแฟ้มเอกซ์เอ็มแอลเข้าระบบต่อไป

รูปที่ 4.10 การเลือกไฟล์ยอว์ลการเสนอขายสินค้าเข้ามายังเว็บแอปพลิเคชัน

3) หลังจากที่กดปุ่ม Submit ตามรูปที่ 4.10 แล้วเว็บแอปพลิเคชันจะแสดงข้อมูลกระแสงานยอว์ลการจัดซื้อกระบวนการทั้งหมด 10 กระบวนการ มีการแสดงข้อมูล Edge_in, Edge_out, ประเภทของ Join และ Split โดยผู้ใช้จะระบุข้อจำกัดช่วงเวลาของแต่ละงาน ดังรูปที่ 4.11 ซึ่งผู้ใช้งานมีการระบุข้อจำกัดช่วงเวลาครบทุกงานเป็นจำนวนเต็มแล้วกดปุ่ม next



รูปที่ 4.11 ระบุข้อจำกัดช่วงเวลาของแต่ละงานของกระแสงานยอว์ลการเสนอขายสินค้า

4) หลังจากกดปุ่ม next ในรูปที่ 4.11 จากนั้นตรวจสอบข้อจำกัดช่วงเวลาของแต่ละงานแล้วกดปุ่ม Download ตามรูปที่ 4.12 ซึ่งจะแสดงข้อมูลกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาในรูปแบบ Dataframe โดยมีคอลัมน์ Process, Name, Edge_in, Edge_out, Join, Split, Lower, Upper และปุ่ม Download เพื่อโหลดไฟล์ผลลัพธ์การแปลงนำเข้าไปยัง เครื่องมือ UPPAAL 4.1.24 ต่อไป

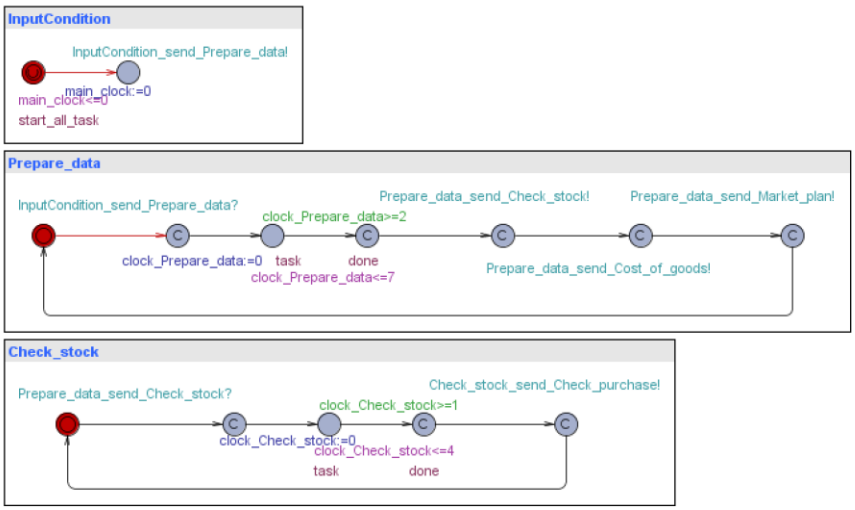
Download!

Process	Edge in	Edge out	Join	Split	Lower	Upper
InputCondition		Prepare_data			NaN	NaN
Prepare_data	InputCondition	Check_stock,Market_plan,Cost_of_goods	and	and	2	7
Check_stock	Prepare_data	Check_purchase	xor	xor	1	4
Cost_of_goods	Prepare_data	Check_purchase	xor	and	1	2
Market_plan	Prepare_data	Check_purchase	xor	and	2	5
Check_purchase	Check_stock,Cost_of_goods,Market_plan	Offer	and	and	2	5
Offer	Check_purchase	Confirm	and	and	1	3
Confirm	Offer	Create_order	xor	xor	1	3
Create_order	Confirm	OutputCondition	xor	and	1	2
OutputCondition	Create_order				NaN	NaN

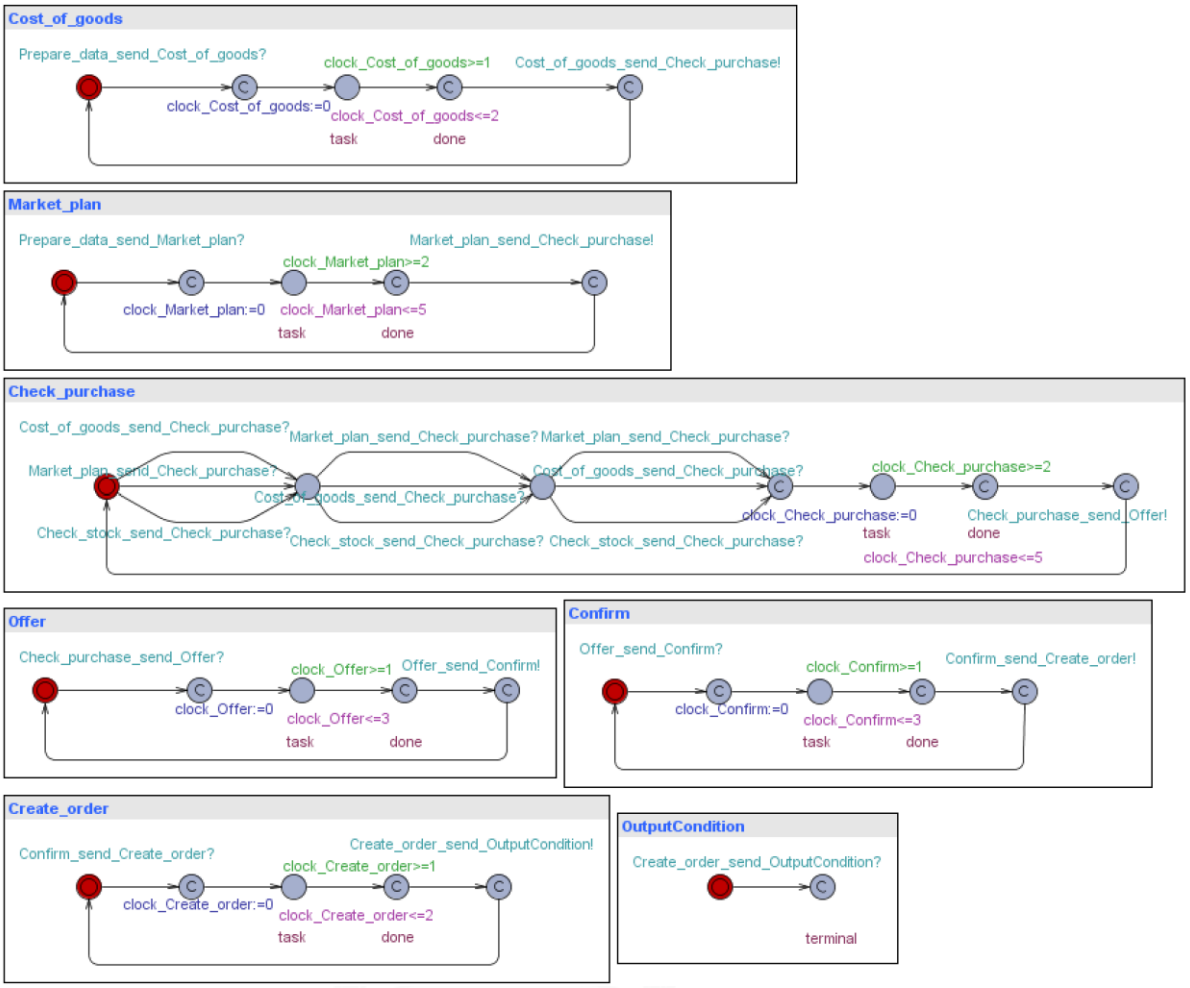
download

รูปที่ 4.12 ข้อมูลกระแสงานยอว์ลการเสนอขายสินค้า และข้อจำกัดช่วงเวลาก่อนทำการแปลงเพื่อโหลดไฟล์ผลลัพธ์การแปลง

5) นำแฟ้มเอกซ์เอ็มแอลที่ได้จากการแปลงเปิดในเครื่องมือ UPPAAL 4.1.24 แล้วปรับแต่งเส้นเชื่อมตามที่ต้องการดังรูปที่ 4.13 จะเห็นได้ว่ามี ทั้งหมด 10 ไทม์ออโตมาตา ซึ่งเกิดจาก InputCondition, OutputCondition และแต่ละงานของกระแสงานยอว์ล อย่างละ 1 รวมเป็น 10 ไทม์ออโตมาตา



รูปที่ 4.13 ไทม์ออโตมาตาการเสนอขายสินค้าบนเครื่องมือ UPPAAL 4.1.24



รูปที่ 4.13 ไทมด์ออโตมาตาการเสนอขายสินค้าบนเครื่องมือ UPPAAL 4.1.24 (ต่อ)

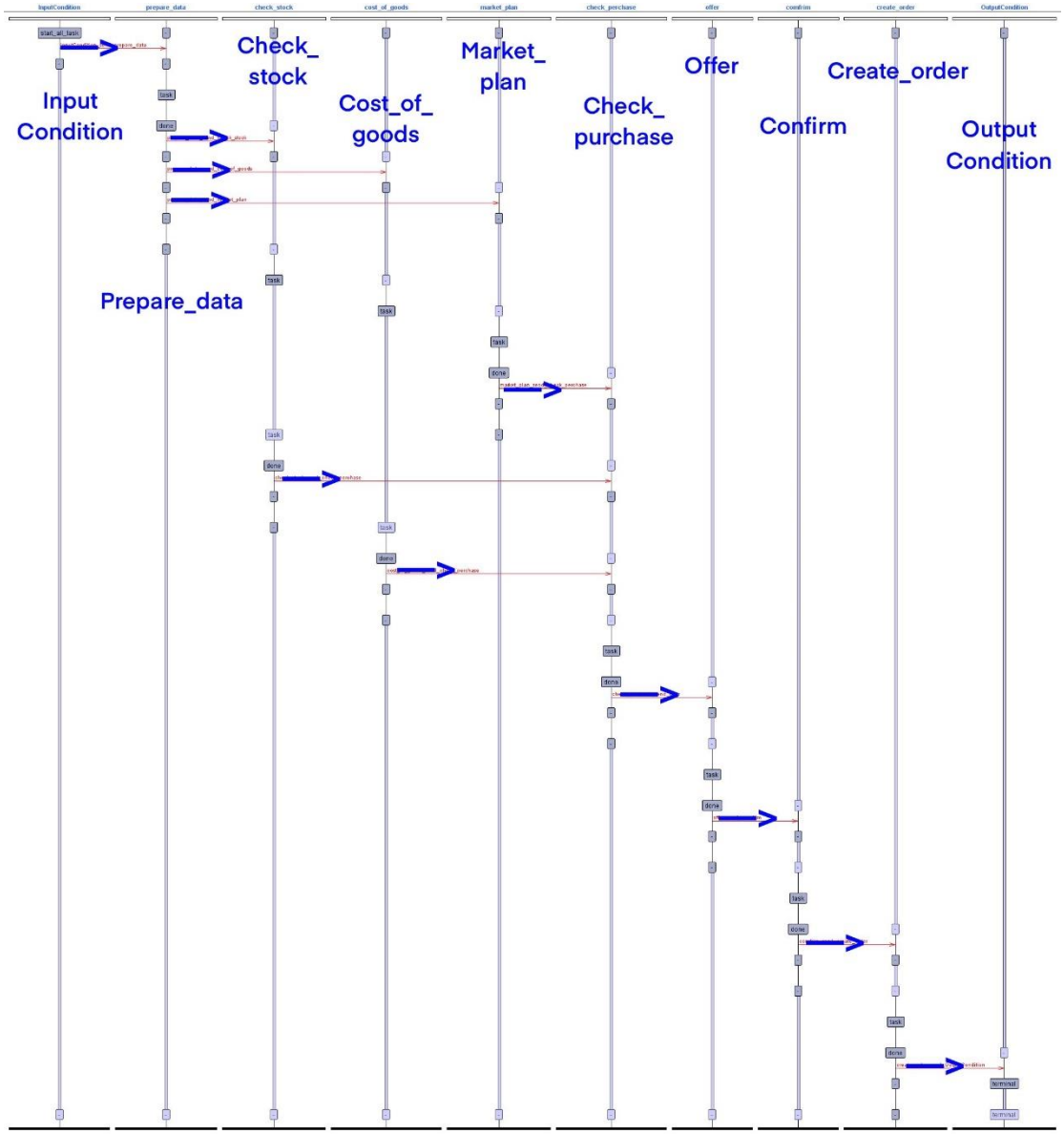
4.3.2.2 การตรวจสอบความถูกต้องของพฤติกรรมผ่าน Simulation และ Model Checker

ส่วนต่อมาหลังจากทำการแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาไปเป็นไทมด์ออโตมาตาแล้วจะทำการตรวจสอบสอบพฤติกรรมผ่าน Simulation และ Model Checker บนเครื่องมือ UPPAAL 4.1.24 เพื่อดูว่ามีไทมด์ออโตมาตาที่ได้จากการแปลงมีพฤติกรรมสอดคล้องกับกระแสงานยอว์ลหรือไม่ ในลักษณะครอบคลุมทุกกิ่งก้านที่เป็นไปได้ และทุกงานสามารถเกิดขึ้นได้อย่างน้อย 1 กรณี สามารถอธิบายเพิ่มเติมได้ดังต่อไปนี้ได้

1) ตรวจสอบผ่าน Simulation บนเครื่องมือ UPPAAL 4.1.24

ทำการจำลองให้ครบทุกงานตามกระแสงานยอว์ลในรูป 4.9 พบว่าสามารถทำงานตามกระแสงานยอว์ลได้ครบครอบคลุมทุกงานตามลำดับ เช่น Prepare_data มี AND split ต้องส่งต่อไปทุก ๆ งานให้ Check_stock,

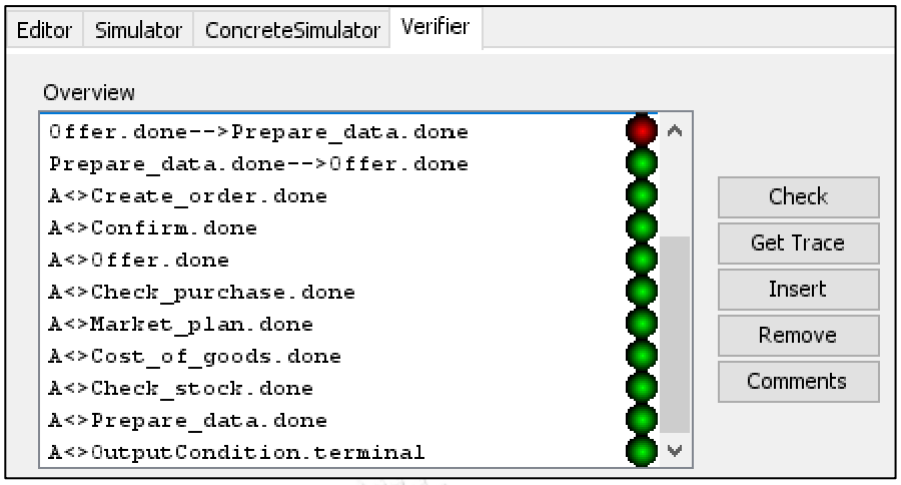
Cost_of_goods และ Market_plan หลังจากนั้น Check_purchase มี AND join ต้องรอทุกงานก่อนหน้าให้เสร็จสิ้นก่อนถึงสามารถเริ่มงานได้ เป็นต้น ดังรูปที่ 4.14



รูปที่ 4.14 ผลการจำลองไทมด์ออโตมาตาของการเสนอขายสินค้าในเครื่องมือ UPPAAL 4.1.24

2) ตรวจสอบผ่าน Model Checker บนเครื่องมือ UPPAAL 4.1.24

การทวนสอบทุกงาน โดยใช้สัญลักษณ์ A<> เพื่อดูว่าทุกงานนั้นต้องเกิดขึ้นในทุก ๆ ครั้งของกระแสงานการจัดซื้อ โดยพบว่าทุกเส้นทางจะมีการทำ Prepare_data, Check_stock, Cost_of_goods, Market_plan, Check_purchase, Offer, Confirm, Create_order เสมอ โดยที่ Prepare_data จะต้องเสร็จสิ้นก่อนเสมอถึงงาน Offer จะสามารถเสร็จสิ้นได้ ดังรูปที่ 4.15 ซึ่งแสดงให้เห็นว่าทุกงานต้องเสร็จสิ้นทุกครั้งในทุก ๆ เส้นทางที่เป็นไปได้ แสดงเป็นสีเขียวเป็นจริง และ Prepare_data จะต้องเสร็จสิ้นก่อนเสมอถึงงาน Offer เป็นจริงที่บรรทัดที่สอง



รูปที่ 4.15 ผลการทวนสอบพฤติกรรมของไทมด์ออโตมาตาของการเสนอขายสินค้า

4.3.2.3 ตัวอย่างการวิเคราะห์ผลลัพธ์ผ่าน Model Checker บนเครื่องมือ UPPAAL

หลังจากทำการแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาไปเป็นไทมด์ออโตมาตา และตรวจสอบสอบพฤติกรรมผ่าน Simulation และ Model Checker แล้ว ส่วนนี้จะเป็นการวิเคราะห์ประสิทธิภาพด้านเวลาของไทมด์ออโตมาตาผ่าน Model Checker ฟังก์ชัน Verifier บนเครื่องมือ UPPAAL ดังนี้ โดยผลลัพธ์ของการวิเคราะห์สามารถแสดงได้ดังในรูปที่ 4.16

1) มีโอกาสการเช็คจัดซื้อจะเสร็จสิ้นภายใน 8 ชั่วโมง สามารถตรวจสอบได้ดังนี้

ในบางเส้นทางที่เป็นไปได้ Check_purchase จะเสร็จสิ้นก่อนหรือเท่ากับ Main_clock == 8 สามารถเขียนตรรกศาสตร์ต้นไม้การคำนวณเวลาได้คือ E[]Check_purchase.done imply main_clock<=8 พบว่าผลลัพธ์เป็นสีเขียวหมายถึงเป็นจริงหรือหมายความว่าในบางเส้นทางที่เป็นไปได้ งาน Check_purchase ที่ตำแหน่ง done จะถูกส่งต่อก่อนหรือเท่ากับ Main_clock ==8

2) ควรใช้เวลาอย่างน้อย 5 ชั่วโมงในการทำงานตั้งแต่ต้นจนเสร็จสิ้นการเช็คจัดซื้อ สามารถตรวจสอบได้ดังนี้

ในทุกเส้นทางที่เป็นไปได้ Check_purchase จะไม่สามารถเสร็จสิ้นก่อนหรือเท่ากับ Main_clock == 5 ได้ สามารถเขียนตรรกศาสตร์ต้นไม้การคำนวณเวลาได้คือ E<> Check_purchase.done and main_clock<=5 ผลลัพธ์เป็นเท็จ หรือหมายความว่า ไม่มีเส้นทางใดเลยที่ Confirm อยู่ที่ตำแหน่ง Done และเวลาน้อยกว่าหรือเท่ากับ 5

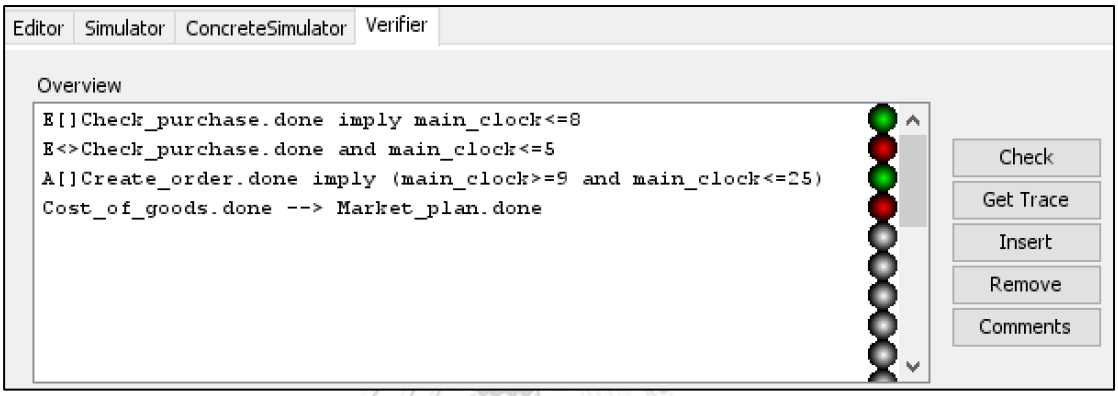
3) ต้องมีการสร้างออร์เดอร์ภายใน 25 ชั่วโมงเสมอ สามารถตรวจสอบได้ดังนี้

ในทุกเส้นทางที่เป็นไปได้ Create_order จะเสร็จสิ้นระหว่าง Main_clock == [9,25] สามารถเขียนตรรกศาสตร์ต้นไม้การคำนวณเวลาได้คือ A[]Create_order.done imply (main_clock>=9 and

main_clock<=25) ผลลัพธ์เป็นจริง หรือหมายความว่าในทุกเส้นทางที่เป็นไปได้ Create_order ตำแหน่ง Done จะถูกส่งต่อระหว่าง Main_clock == 9 ถึง Main_clock == 25 เสมอ

4) ต้องวิเคราะห์จุดคุ้มทุนเสร็จก่อนการวางแผนการตลาดเสมอ สามารถตรวจสอบได้ดังนี้

ในทุกเส้นทางที่เป็นไปได้ Cost_of_goods ต้องเสร็จสิ้นก่อน Market_plan เสมอ สามารถเขียนตรรกศาสตร์ต้นไม้การคำนวณเวลาได้คือ Cost_of_goods.done --> market_plan.done ผลลัพธ์เป็นเท็จ หรือหมายความในทุกเส้นทางที่เป็นไปได้มีอย่างน้อย 1 เส้นทางที่ Cost_of_goods เสร็จสิ้นหลัง Market_plan



รูปที่ 4.16 ผลการทวนสอบของไทมด์ออโตมาตาของการเสนอขายสินค้า

4.3.2.4 สรุปผลกรณีศึกษาที่ 2 กระแสงานยอว์ลเรื่องการเสนอขายสินค้า

จากกรณีศึกษาที่ 2 กระแสงานยอว์ลเรื่องการเสนอขายสินค้า พบว่าสามารถแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาไปเป็นไทมด์ออโตมาตาผ่านเว็บแอปพลิเคชันได้ถูกต้องเส้นเชื่อมในไทมด์ออโตมาตาครบทุกตำแหน่ง มีครบทุกงานตามกระแสงานยอว์ล ต่อมาตรวจสอบสอบพฤติกรรมผ่าน Simulation และ Model Checker บนเครื่องมือ UPPAAL พบว่าไทมด์ออโตมาตาที่ได้จากการแปลงมีพฤติกรรมสอดคล้องกับกระแสงานยอว์ล ผลการทวนสอบประสิทธิภาพด้านเวลาพบว่า เส้นทางที่เร็วที่สุดที่สามารถจบงานของกระแสงานยอว์ลเรื่องการจัดซื้ออยู่ที่ Main_clock == 9 และช้าที่สุดที่ Main_clock == 25

4.3.3 กรณีศึกษาที่ 3 กระแสงานยอว์ลเรื่องการสมัครบัตรเครดิต

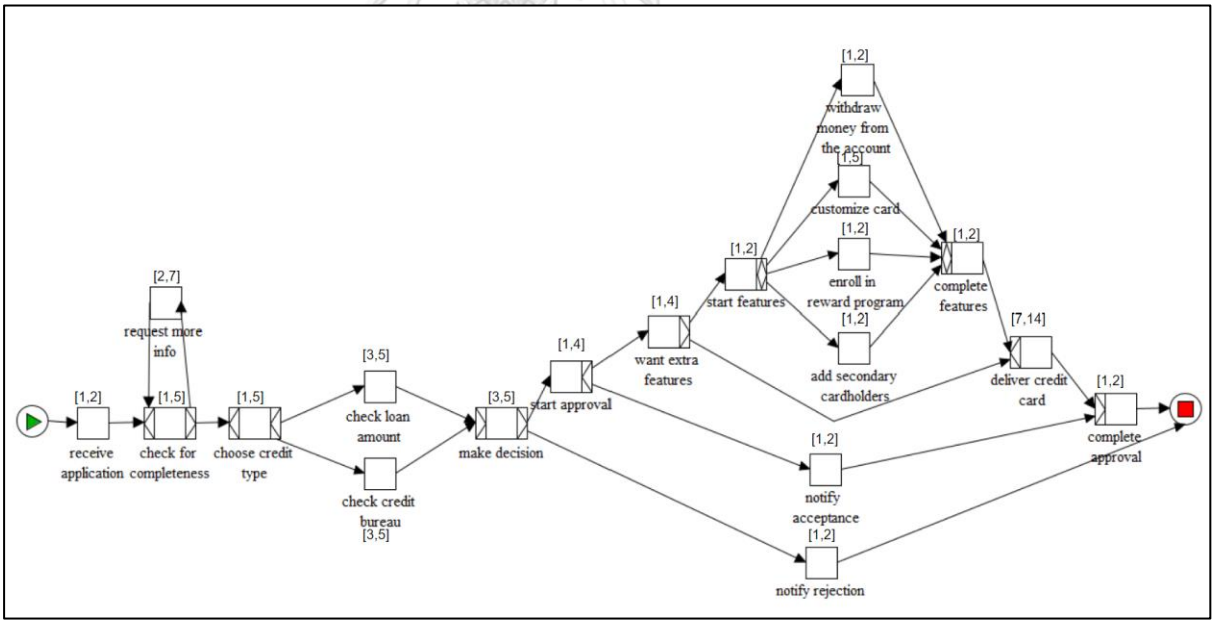
ในกรณีศึกษาที่ 3 กระแสงานยอว์ลเรื่องการสมัครบัตรเครดิตนั้นจะเป็นการจำลองกระบวนการของการสมัครบัตรเครดิต มีการรับข้อมูลการสมัคร เช็คความถูกต้อง เลือกประเภทบัตร วงเงิน แล้วให้ทางผู้ออกบัตรเครดิตอนุมัติ โดยในกรณีศึกษานั้นจะมีขั้นต่อการศึกษาคือการสร้างแบบจำลองกระแสงานยอว์ล และแปลงไปเป็นไทมด์ออโตมาตาแบบมีข้อจำกัดช่วงเวลาหลังจากนั้นทำการตรวจสอบสอบพฤติกรรมผ่าน Simulation และ Model Checker เพื่อยืนยันว่าไทมด์ออโตมาตานั้นมีพฤติกรรมการทำงานสอดคล้องกับกระแสงานยอว์ล ต่อมาทำการ

วิเคราะห์ทวนสอบประสิทธิภาพด้านเวลา และสรุปผลของกรณีศึกษา โดยกำหนดให้หน่วยของเวลาเป็นชั่วโมง ซึ่งอธิบายเพิ่มเติมดังนี้

4.3.3.1 การสร้างแบบจำลองกระแสงานยอว์ลและแปลงไปเป็นไทมด์ออโตมาตา

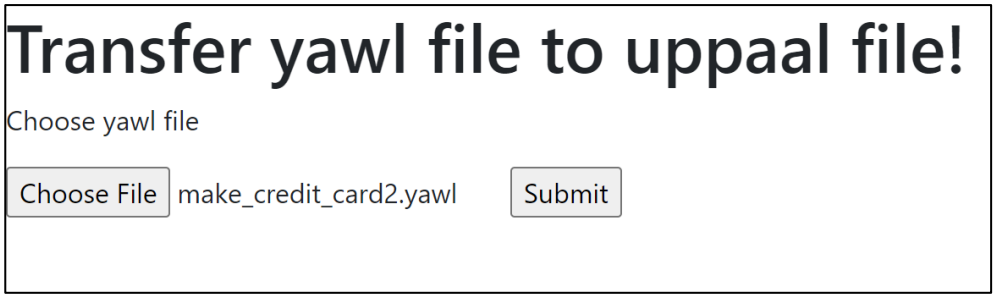
การสร้างแบบจำลองกระแสงานยอว์ล และแปลงไปเป็นไทมด์ออโตมาตา มีทั้งหมด 5 ขั้นตอนตามหัวข้อที่ 3.4 คือสร้างแบบจำลองผ่านเครื่องมือยอว์ล 4.3.1 ต่อมานำแฟ้มเอกซ์เอ็มแอลของกระแสงานยอว์ลเข้าเว็บแอปพลิเคชันหลังจากนั้นทำการใส่ข้อมูลข้อจำกัดช่วงเวลาแล้วนำออก เพื่อนำแฟ้มเอกซ์เอ็มแอลของไทมด์ออโตมาตามาตรวจสอบความสมบูรณ์ของสัญลักษณ์ของไทมด์ออโตมาตาหลังการแปลง และทวนสอบในเครื่องมือ UPPAAL ต่อไป สามารถอธิบายเพิ่มเติมดังนี้

1) สร้างแบบจำลองการจัดซื้อ โดยใช้เครื่องมือยอว์ล 4.3.1 ดังรูปที่ 4.17 โดยงาน Check for completeness จะมี XOR join และ XOR split สามารถทำงานแบบ Loop ตรวจสอบความถูกต้องของข้อมูลได้ งาน Start approval จะมี AND split ส่งให้งาน Notify acceptance และงาน Want extra features โดยงาน Start features จะมี OR split สามารถเลือกทำงาน Withdraw money from the account, Customize card, Enroll in reward program หรือ Add secondary cardholders ได้หลังจากนั้นจะรวบรวมข้อมูลที่ Complete features ที่มี OR join



รูปที่ 4.17 กระแสงานยอว์ลการสมัครบัตรเครดิตบนโปรแกรมยอว์ล 4.3.1

2) ทำการบันทึกแล้วนำเข้าแฟ้มเอกซ์เอ็มแอลของกระแสงานยอว์ลที่ได้จากเครื่องมือยอว์ล 4.3.1 เข้ามายังเว็บแอปพลิเคชัน ดังรูปที่ 4.18 โดยกดปุ่ม Choose File เพื่อเลือกแฟ้มเอกซ์เอ็มแอล หลังจากนั้นกด Submit เพื่อนำแฟ้มเอกซ์เอ็มแอลเข้าระบบต่อไป



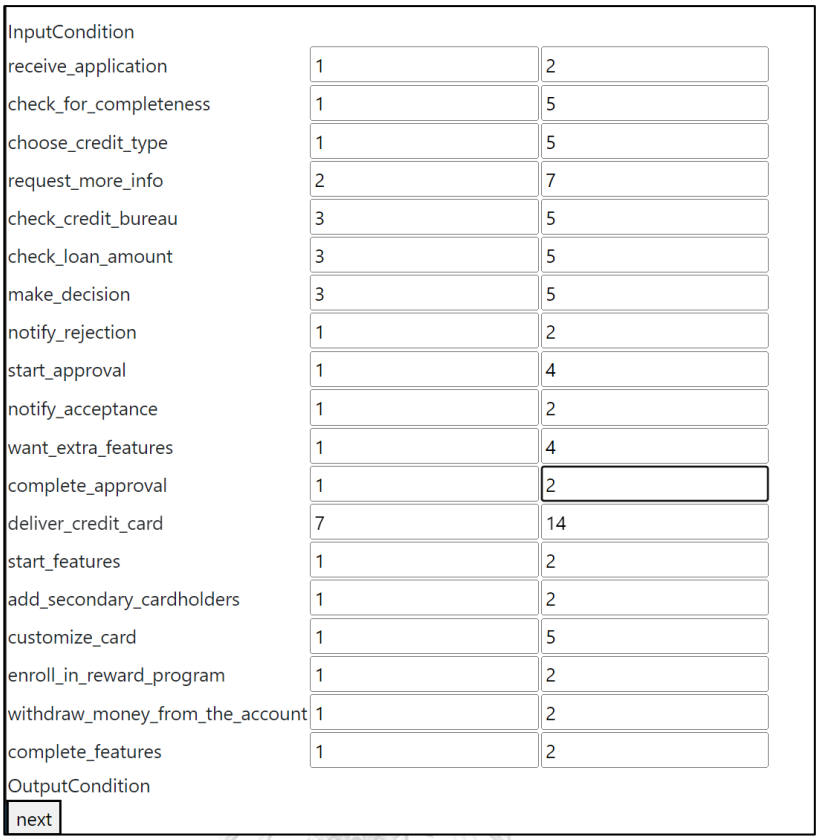
รูปที่ 4.18 การเลือกไฟล์ยอว์ลการสมัครบัตรเครดิตเข้ามายังเว็บแอปพลิเคชัน

3) หลังจากที่กดปุ่ม Submit ตามรูปที่ 4.18 แล้วเว็บแอปพลิเคชันจะแสดงข้อมูลกระแสงานยอว์ลการจัดซื้อกระบวนการทั้งหมด 21 กระบวนการ มีการแสดงข้อมูล Edge_in, Edge_out, ประเภทของ Join และ Split โดยผู้ใช้จะระบุข้อจำกัดช่วงเวลาของแต่ละงาน ดังรูปที่ 4.19 ซึ่งผู้ใช้งานมีการระบุข้อจำกัดช่วงเวลาครบทุกงานเป็นจำนวนเต็มแล้วกดปุ่ม next

show all process!

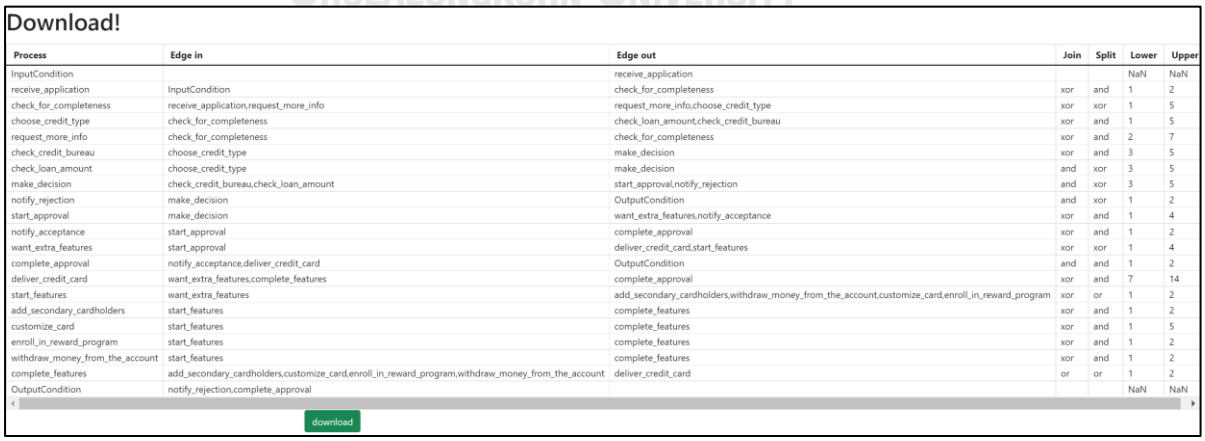
Process	Edge in	Edge out	Join	Split
InputCondition		receive_application		
receive_application	InputCondition	check_for_completeness	xor	and
check_for_completeness	receive_application,request_more_info	request_more_info,choose_credit_type	xor	xor
choose_credit_type	check_for_completeness	check_loan_amount,check_credit_bureau	xor	and
request_more_info	check_for_completeness	check_for_completeness	xor	and
check_credit_bureau	choose_credit_type	make_decision	xor	and
check_loan_amount	choose_credit_type	make_decision	and	xor
make_decision	check_credit_bureau,check_loan_amount	start_approval,notify_rejection	and	xor
notify_rejection	make_decision	OutputCondition	and	xor
start_approval	make_decision	want_extra_features,notify_acceptance	xor	and
notify_acceptance	start_approval	complete_approval	xor	and
want_extra_features	start_approval	deliver_credit_card,start_features	xor	xor
complete_approval	notify_acceptance,deliver_credit_card	OutputCondition	and	and
deliver_credit_card	want_extra_features,complete_features	complete_approval	xor	and
start_features	want_extra_features	add_secondary_cardholders,withdraw_money_from_the_account,customize_card,enroll_in_reward_program	xor	or
add_secondary_cardholders	start_features	complete_features	xor	and
customize_card	start_features	complete_features	xor	and
enroll_in_reward_program	start_features	complete_features	xor	and
withdraw_money_from_the_account	start_features	complete_features	xor	and
complete_features	add_secondary_cardholders,customize_card,enroll_in_reward_program,withdraw_money_from_the_account	deliver_credit_card	or	or
OutputCondition	notify_rejection,complete_approval			

รูปที่ 4.19 ระบุข้อจำกัดช่วงเวลาของแต่ละงานของกระแสงานยอว์ลการสมัครบัตรเครดิต



รูปที่ 4.19 ระบุข้อจำกัดช่วงเวลาของแต่ละงานของกระแสงานยอว์ลการสมัครบัตรเครดิต (ต่อ)

4) หลังจากกดปุ่ม next ในรูปที่ 4.19 จากนั้นตรวจสอบข้อจำกัดช่วงเวลาของแต่ละงานแล้วกดปุ่ม Download ตามรูปที่ 4.20 ซึ่งจะแสดงข้อมูลกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาในรูปแบบ Dataframe โดยมีคอลัมน์ Process, Name, Edge_in, Edge_out, Join, Split, Lower, Upper และปุ่ม Download เพื่อโหลดไฟล์ผลลัพธ์การแปลงนำเข้าไปยังเครื่องมือ UPPAAL 4.1.24 ต่อไป

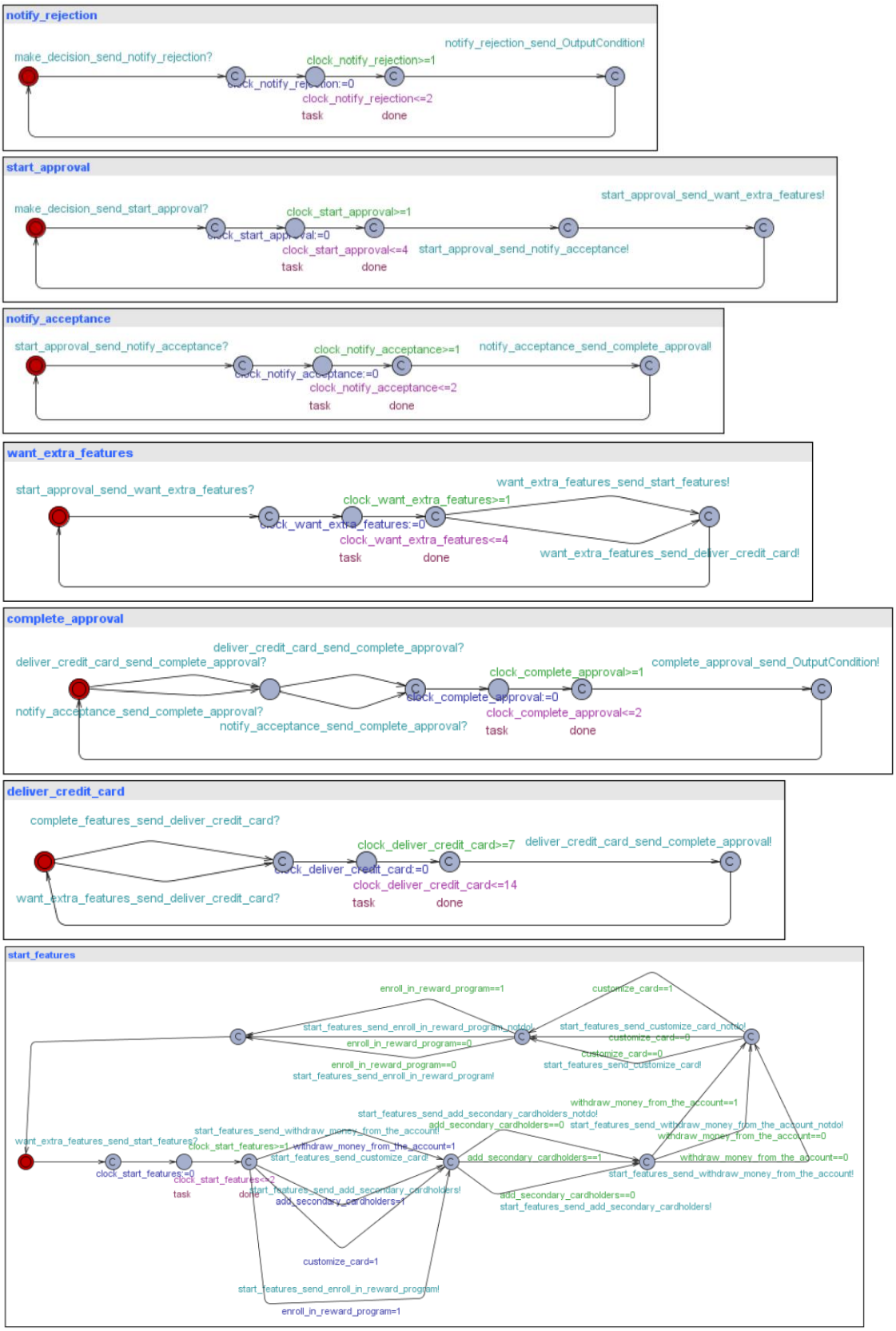


Process	Edge in	Edge out	Join	Split	Lower	Upper
InputCondition		receive_application			NaN	NaN
receive_application	InputCondition	check_for_completeness	xor	and	1	2
check_for_completeness	receive_application,request_more_info	request_more_info,choose_credit_type	xor	xor	1	5
choose_credit_type	check_for_completeness	check_loan_amount,check_credit_bureau	xor	and	1	5
request_more_info	check_for_completeness	check_for_completeness	xor	and	2	7
check_credit_bureau	choose_credit_type	make_decision	xor	and	3	5
check_loan_amount	choose_credit_type	make_decision	and	xor	3	5
make_decision	check_credit_bureau,check_loan_amount	start_approval,notify_rejection	and	xor	3	5
notify_rejection	make_decision	OutputCondition	and	xor	1	2
start_approval	make_decision	want_extra_features,notify_acceptance	xor	and	1	4
notify_acceptance	start_approval	complete_approval	xor	and	1	2
want_extra_features	start_approval	deliver_credit_card,start_features	xor	xor	1	4
complete_approval	notify_acceptance,deliver_credit_card	OutputCondition	and	and	1	2
deliver_credit_card	want_extra_features,complete_features	complete_approval	xor	and	7	14
start_features	want_extra_features	add_secondary_cardholders,withdraw_money_from_the_account,customize_card,enroll_in_reward_program	xor	or	1	2
add_secondary_cardholders	start_features	complete_features	xor	and	1	2
customize_card	start_features	complete_features	xor	and	1	5
enroll_in_reward_program	start_features	complete_features	xor	and	1	2
withdraw_money_from_the_account	start_features	complete_features	xor	and	1	2
complete_features	add_secondary_cardholders,customize_card,enroll_in_reward_program,withdraw_money_from_the_account	deliver_credit_card	or	or	1	2
OutputCondition	notify_rejection,complete_approval				NaN	NaN

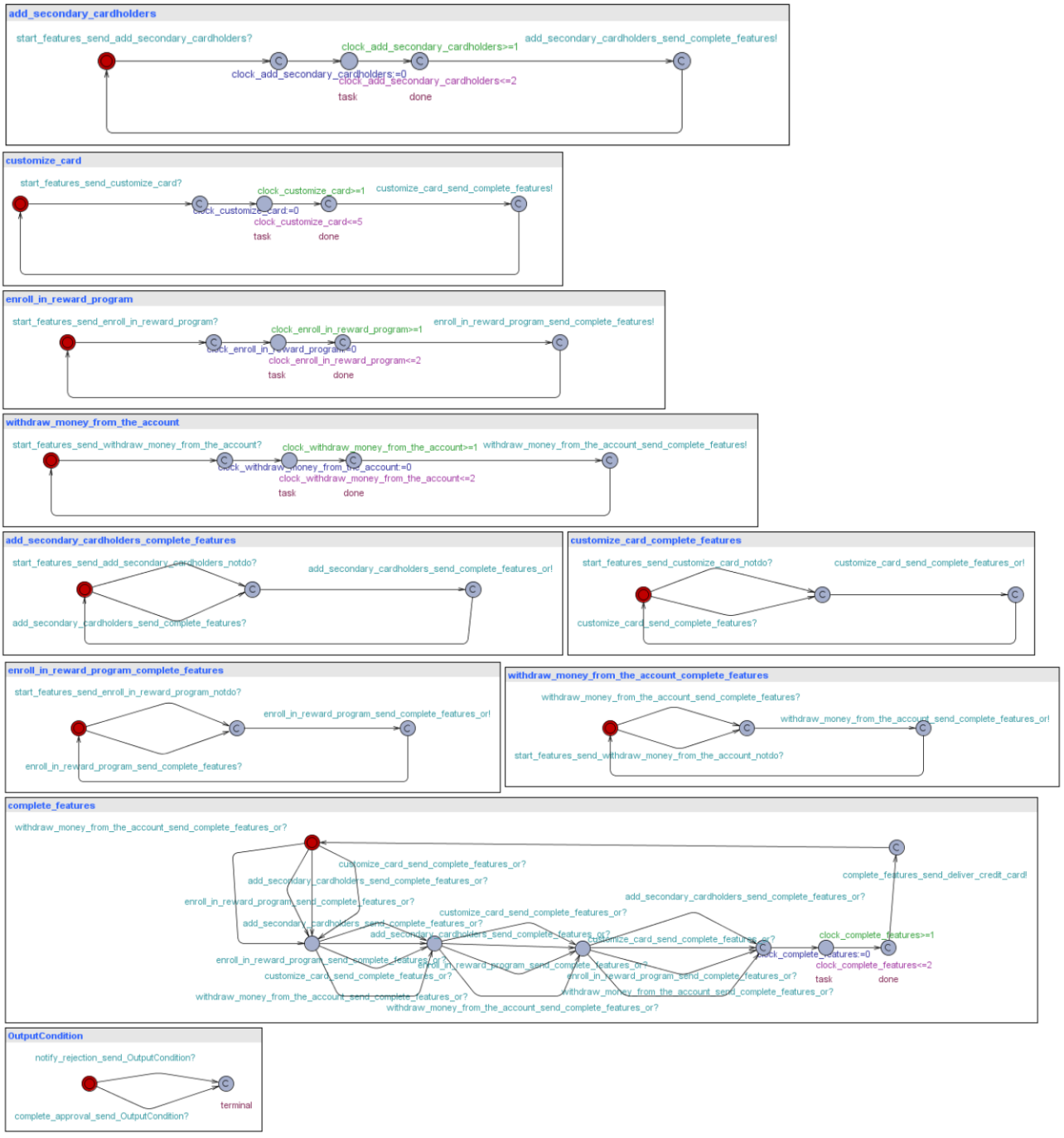
รูปที่ 4.20 ข้อมูลกระแสงานยอว์ลการสมัครบัตรเครดิต และข้อจำกัดช่วงเวลาก่อนทำการแปลงเพื่อโหลดไฟล์ผลลัพธ์การแปลง

5) นำแฟ้มเอกซ์เอ็มแอลที่ได้จากการแปลงเปิดในเครื่องมือ UPPAAL 4.1.24 แล้วปรับแต่งเส้นเชื่อมตามที่ต้องการดังรูปที่ 4.21 จะเห็นได้ว่ามี ทั้งหมด 25 ไทมด์ออโตมาตา ซึ่งเกิดจาก InputCondition, OutputCondition และแต่ละงานของกระแสงานยอว์ล อย่างละ 1 รวมเป็น 21 ไทมด์ออโตมาตา และมีไทมด์ออโตมาตาพิเศษที่เกิดจาก OR split อีก 4 งานรวมเป็น 25 ไทมด์ออโตมาตา

รูปที่ 4.21 ไทมด์ออโตมาตาของการสมัครบัตรเครดิตบนเครื่องมือ UPPAAL 4.1.24



รูปที่ 4.21 ไทมด์ออโตมาตาของการสมัครบัตรเครดิตบนเครื่องมือ UPPAAL 4.1.24 (ต่อ)



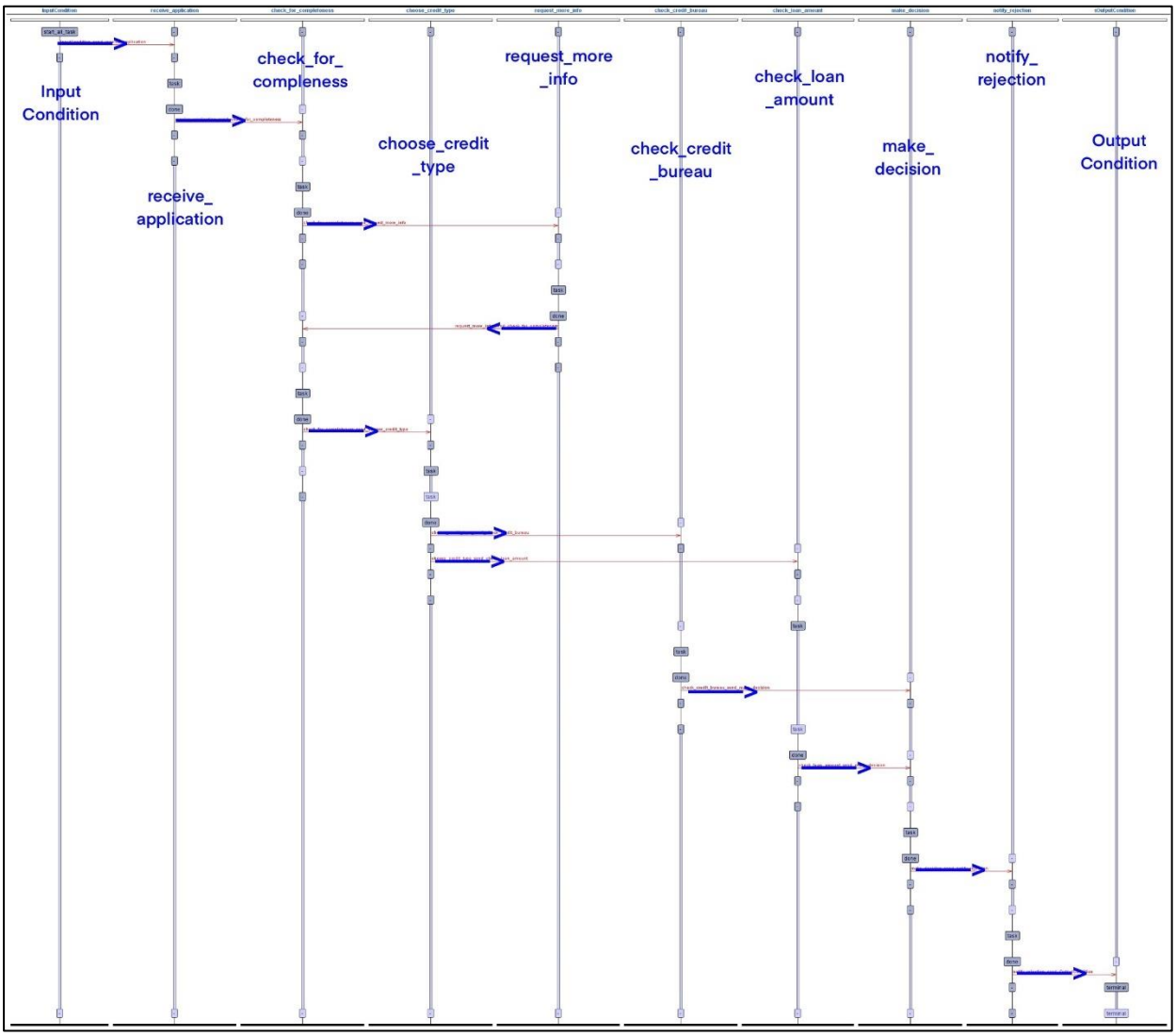
รูปที่ 4.21 ไทมด์ออโตมาตาของการสมัครบัตรเครดิตบนเครื่องมือ UPPAAL 4.1.24 (ต่อ)

4.3.3.2 การตรวจสอบความถูกต้องของพฤติกรรมผ่าน Simulation และ Model Checker

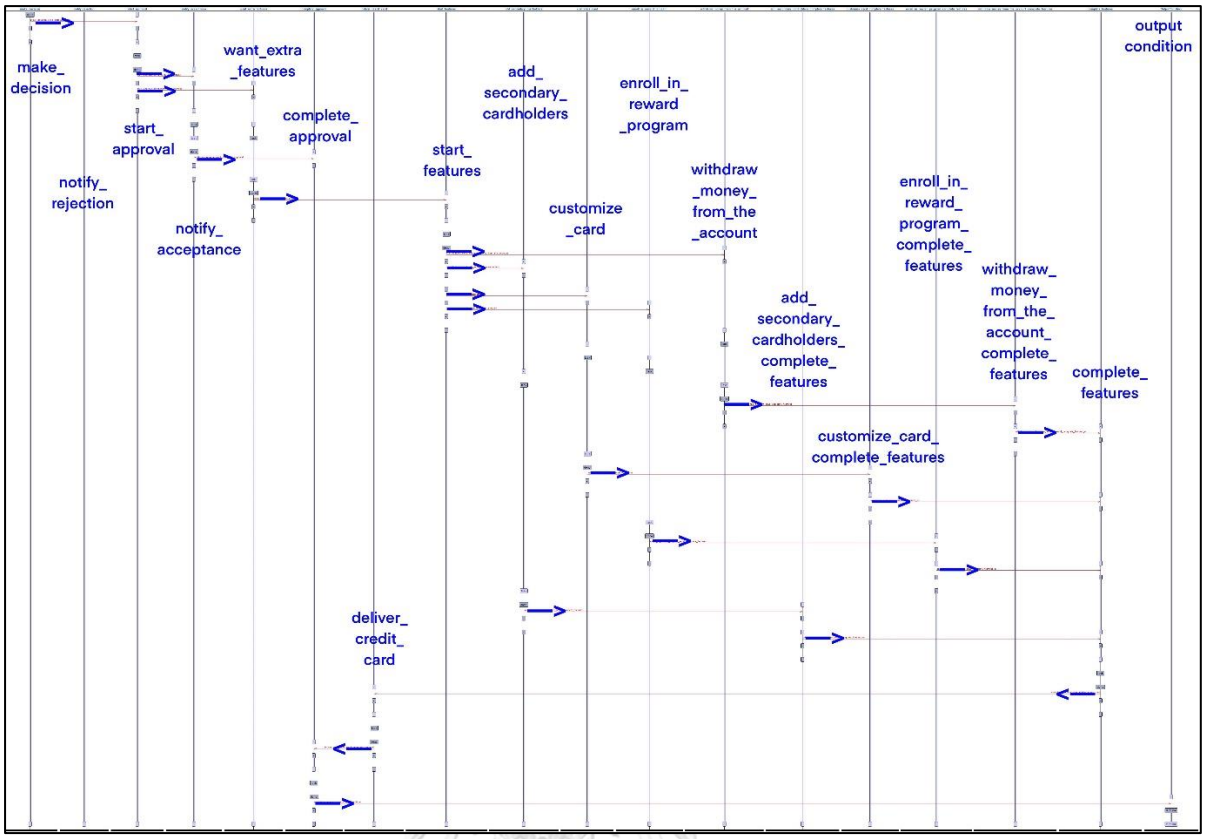
ส่วนต่อมาหลังจากทำการแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาไปเป็นไทมด์ออโตมาตาแล้ว จะทำการตรวจสอบสอบพฤติกรรมผ่าน Simulation และ Model Checker บนเครื่องมือ UPPAAL 4.1.24 เพื่อดูว่ามีไทมด์ออโตมาตาที่ได้จากการแปลงมีพฤติกรรมสอดคล้องกับกระแสงานยอว์ลหรือไม่ ในลักษณะครอบคลุมทุกกิ่งก้านที่เป็นไปได้ และทุกงานสามารถเกิดขึ้นได้อย่างน้อย 1 กรณี สามารถอธิบายเพิ่มเติมได้ดังต่อไปนี้

1) ตรวจสอบผ่าน Simulation บนเครื่องมือ UPPAAL 4.1.24

ทำการจำลองให้ครบทุกงานตามกระแสงานยอว์ล พบว่าสามารถทำงานตามกระแสงานยอว์ลได้ครบทุกงานโดยจะแบ่งเป็นสองแบบจำลอง แบบที่หนึ่ง Make_decision ซึ่งมี XOR split เลือก Notify_rejection ดังรูปที่ 4.22 และแบบที่สอง Make_decision เลือก Start_approval ดังรูปที่ 4.23 โดยถ้าเลือก Notify_rejection หรือหมายความว่า ไม่สามารถทำบัตรเครดิตได้ ก็จะถูกส่งไปจบงานทันที ส่วนถ้าเลือก Start_approval ก็จะถูกส่งไป Notify_acceptance และ Want_extra_features เพื่อสอบถามคุณสมบัติเพิ่มเติมต่อไป



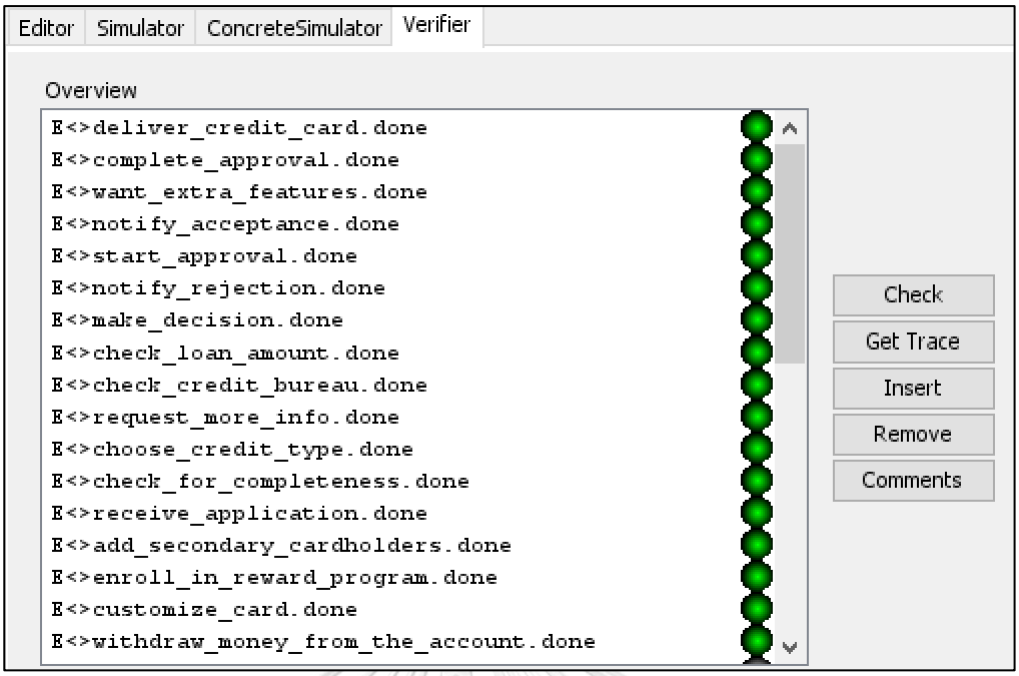
รูปที่ 4.22 ผลการจำลองไทมด์ออโตมาตาของการสมัครบัตรเครดิต กรณี make_decision เลือก notify_rejection ในเครื่องมือ UPPAAL 4.1.24



รูปที่ 4.23 ผลการจำลองไทมด์ออโตมาตาของการสมัครบัตรเครดิต กรณี make_decision เลือก start_approval ในเครื่องมือ UPPAAL 4.1.24

2) ตรวจสอบผ่าน Model Checker บนเครื่องมือ UPPAAL 4.1.24

ทำการทวนสอบโดยใช้สัญลักษณ์ E<> เพื่อดูว่าทุกงานนั้นต้องเกิดขึ้นในอย่างน้อย 1 เส้นทางของกระแสงานการสมัครบัตรเครดิต โดยพบว่า deliver_credit_card, complete_approval, want_extra_features, notify_acceptance, start_approval, notify_rejection, make_decision, check_loan_amount, check_credit_bureau, request_more_info, choose_credit_type, check_for_completeness, receive_application, add_secondary_cardholders, enroll_in_reward_program, customize_card, withdraw_money_from_the_account สามารถเสร็จสิ้นได้อย่างน้อย 1 เส้นทางการทำงาน ดังรูปที่ 4.24 พบว่าแสดงเป็นสีเขียวทั้งหมด หมายความว่าเป็นจริงทั้งหมด



รูปที่ 4.24 ผลการทวนสอบพฤติกรรมของไทมด์ออโตมาตาของการสมัครบัตรเครดิต

4.3.3.3 ตัวอย่างการวิเคราะห์ผลลัพธ์ผ่าน Model Checker บนเครื่องมือ UPPAAL

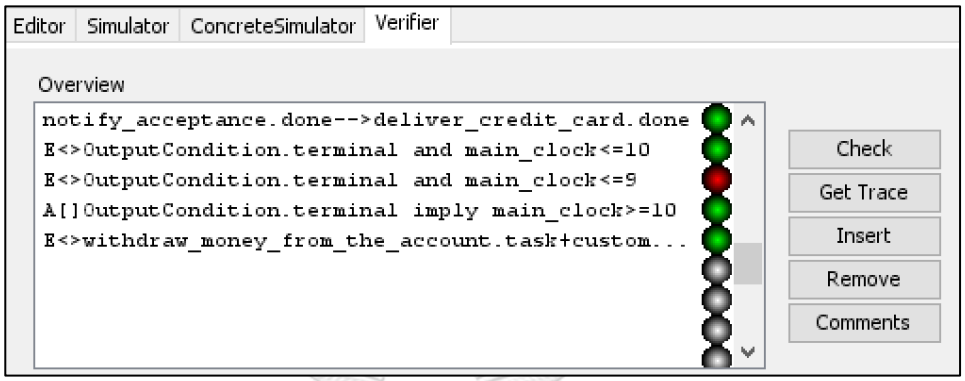
หลังจากทำการแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาไปเป็นไทมด์ออโตมาตา และตรวจสอบสอบพฤติกรรมผ่าน Simulation และ Model Checker แล้ว ส่วนนี้จะเป็นการวิเคราะห์ประสิทธิภาพด้านเวลาของไทมด์ออโตมาตาผ่าน Model Checker ฟังก์ชัน Verifier บนเครื่องมือ UPPAAL ดังนี้ โดยผลลัพธ์ของการวิเคราะห์สามารถแสดงได้ดังในรูปที่ 4.25

1) มีโอกาสที่จะเสร็จสิ้นการสมัครบัตรเครดิตภายใน 10 ชั่วโมง สามารถตรวจสอบได้ดังนี้

ในทุกเส้นทางที่เป็นไปได้ OutputCondition จะ Terminal ไม่เกิน Main_clock==10 หรือใช้เวลาน้อยกว่า Main_clock==10 สามารถเขียนตรรกศาสตร์ต้นไม้การคำนวณเวลาได้คือ E<> OutputCondition .terminal and main_clock <=10 ร่วมกับ E <> OutputCondition.terminal and main_clock<=9 หรือ A[]OutputCondition.terminal imply main_clock>=10 พบว่าผลลัพธ์เป็นสีเขียว สีแดง และสีเขียวตามลำดับ หรือจริง เท็จ จริง ซึ่งหมายความว่า มีบางเส้นทางที่ OutputCondition อยู่ตำแหน่ง Terminal และเวลาน้อยกว่าหรือเท่ากับ 10 และไม่มีเส้นทางใดเลยที่ OutputCondition อยู่ตำแหน่ง Terminal และเวลาน้อยกว่าหรือเท่ากับ 9 สรุปได้ว่า OutputCondition จะ Terminal เร็วที่สุดที่ Main_clock==10

2) ทางบริษัทจะต้องส่งแจ้งเตือนผลการสมัครบัตรที่ผ่านแล้วพร้อมข้อมูลก่อนส่งมอบบัตรเครดิต สามารถตรวจสอบได้ดังนี้

ในทุกเส้นทางที่เป็นไปได้ Notify_acceptance ต้องเสร็จสิ้นก่อน Deliver_credit_card เสมอ สามารถเขียนตรรกศาสตร์ต้นไม้การคำนวณเวลาได้คือ notify_acceptance.done-->deliver_credit_card.done พบว่าผลลัพธ์เป็นจริง หรือหมายความว่าระบบจะต้องส่งแจ้งเตือนเสร็จสิ้นก่อนที่บัตรเคดิตจะถูกส่งออกไปให้ลูกค้า



รูปที่ 4.25 ผลการทวนสอบของไทมด์ออโตมาตาของการเสนอขายสินค้า

4.3.2.4 สรุปผลกรณีศึกษาที่ 3 กระแสงานยอว์ลเรื่องการสมัครบัตรเครดิต

จากกรณีศึกษาที่ 3 กระแสงานยอว์ลเรื่องการสมัครบัตรเครดิต พบว่าสามารถแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาไปเป็นไทมด์ออโตมาตาผ่านเว็บแอปพลิเคชันได้ถูกต้อง เส้นเชื่อมในไทมด์ออโตมาตาครบทุกตำแหน่ง มีครบทุกงานตามกระแสงานยอว์ล ต่อมาตรวจสอบสอบพฤติกรรมผ่าน Simulation และ Model Checker บนเครื่องมือ UPPAAL พบว่าไทมด์ออโตมาตาที่ได้จากการแปลงมีพฤติกรรมสอดคล้องกับกระแสงานยอว์ล ผลการทวนสอบประสิทธิภาพด้านเวลาพบว่า เส้นทางที่เร็วที่สุดที่สามารถจบงานของกระแสงานยอว์ลเรื่องการสมัครบัตรเครดิตอยู่ที่ Main_clock == 10 และระบบจะต้องส่งแจ้งเตือนเสร็จสิ้นก่อนที่บัตรเคดิตจะถูกส่งออกไปให้ลูกค้าเสมอ

บทที่ 5
สรุปผลการดำเนินงานวิจัย

5.1 สรุปผลงานวิจัย

จากงานวิจัยนี้มีจุดประสงค์เพื่อสร้างกฎการแปลงกระแสงานยอว์ลไปเป็นไทมด์ออโตมาตาแบบมีข้อจำกัดช่วงเวลา และเพื่อออกแบบพัฒนาเครื่องมือที่ใช้สำหรับการแปลงกระแสงานยอว์ลไปเป็นไทมด์ออโตมาตาพบว่า ได้ออกแบบกฎการแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาไปเป็นไทมด์ออโตมาตา โดยงานวิจัยนี้ได้เสนอกฎการแปลงสัญลักษณ์ของกระแสงานยอว์ลไว้ทั้งหมด 11 กฎ ได้แก่ Input Condition, Output Condition, Atomic task, Atomic task to outputcondition, AND-split, AND-join, XOR-split, XOR-join, OR-split, OR-join, Loop จากนั้นทำการพัฒนา เว็บแอปพลิเคชัน ในการเพิ่มข้อจำกัดช่วงเวลา และแปลงไปเป็นไทมด์ออโตมาตา ที่สามารถเปิดในเครื่องมือ UPPAAL ได้ โดยการนำเข้าไฟล์ยอว์ลเป็นไฟล์นำเข้าซึ่งได้มาจากเครื่องมือยอว์ล ไฟล์จะอยู่ในรูปแบบแฟ้มเอกสารเอกซ์เอ็มแอลเมื่อกระแสงานยอว์ลถูกนำเข้าเว็บแอปพลิเคชัน จะให้เพิ่มข้อจำกัดช่วงเวลาและแปลงไปเป็นไทมด์ออโตมาตาจะได้ไฟล์ไทมด์ออโตมาตา ในรูปแบบแฟ้มเอกสารเอกซ์เอ็มแอล จากนั้นนำไฟล์ไทมด์ออโตมาตาเปิดในเครื่องมือ UPPAAL สำหรับทวนสอบความถูกต้องของกฎการแปลง ว่าพฤติกรรมก่อน และหลังแปลงมีพฤติกรรมการทำงานที่สอดคล้องกันหรือไม่ โดยผู้วิจัยได้ทดสอบกับกรณีศึกษา 3 กรณีที่ครอบคลุมกฎการแปลงทั้งหมด พบว่าสามารถแปลงจากกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาไปเป็นไทมด์ออโตมาตาได้ และครอบคลุมกฎการแปลงทุกข้อ โดยมีพฤติกรรมการทำงานที่สอดคล้องกัน จากนั้นนำไทมด์ออโตมาตามาวิเคราะห์และทวนสอบพบว่าสามารถรองรับตรรกศาสตร์ต้นไม้การคำนวณเวลาตามที่ UPPAAL รองรับได้

5.2 ข้อจำกัดของการแปลง

1) เมื่อแปลงกระแสงานยอว์ลที่มีข้อจำกัดช่วงเวลาไปเป็นไทมด์ออโตมาตาแล้วนำไฟล์ที่ได้การจากแปลงไปเปิดในเครื่องมือ UPPAAL ผู้ใช้จะต้องทำการจัดตำแหน่งของเส้นเชื่อมใหม่ เพราะการแปลงยังไม่สามารถจัดตำแหน่งเส้นให้อยู่ในรูปแบบที่สวยงามได้

2) สัญลักษณ์ Or split และ Or join ต้องใช้คู่กันเสมอ (อยู่ในรูป Well-formed)

5.3 ประโยชน์ที่ได้รับ

1) สามารถตรวจสอบด้านประสิทธิภาพของเวลาแบบมีช่วงเวลาของกระแสงานยอว์ล

2) ได้วิธีการแปลงกระแสงานยอว์ลไปเป็นไทมด์ออโตมาตาที่เปิดได้โดยเครื่องมือ UPPAAL

3) สามารถใช้เครื่องมือ UPPAAL ในการตรวจสอบ และทวนสอบกระแสงานของระบบการจัดการกระบวนการทางธุรกิจจากเครื่องมือยอว์ล ได้

5.4 ข้อเสนอแนะ

1) ออกแบบกฎการแปลงให้รองรับสัญลักษณ์ Multiple Instance Composite Task, Multple instances of an atomic task

2) พัฒนาให้สามารถสร้าง ตรรกศาสตร์ต้นไม้การคำนวณเวลาหลังจากการแปลงได้เลย เพื่อช่วยในการทวนสอบ และวิเคราะห์ผล

3) พัฒนาให้ไทมด์ออโตมาตาที่ได้มีความซับซ้อนน้อยลง เพื่อง่ายต่อการทำความเข้าใจมากขึ้น

4) พัฒนาให้ OR join และ OR split รองรับกระแสงานที่ไม่เป็น Well-formed

บรรณานุกรม

1. Benedict, T. *ABPMP Standards for Business Process Management (BPM)*. Available from: https://www.abpmp.org/page/BPM_Profession.
2. *Business Process Management*. Available from: https://www.goodmaterial.co/business-process-management/.
3. Alur, R. and D.L. Dill, *A theory of timed automata.* Theoretical computer science, 1994. 126(2): p. 183-235.
4. Behrmann, G., A. David, and K.G. Larsen, *A tutorial on Uppaal* 4.0. Department of computer science, Aalborg university, 2006.
5. Alur, R. *Timed automata*. in *International Conference on Computer Aided Verification*. 1999. Springer.
6. Soltani, A. *Timed Automata for Workflow Modeling and Analysis*. 2014.
7. Maruth, R., *Transforming YAWL Workflows with Time Constraints to Timed Automata*.
8. Proenza, J., *The UPPAAL Model Checker*.
9. *UPPAAL*. Available from: http://www.uppaal.com/.
10. Emerson, E.A., *Temporal and modal logic*, in *Formal Models and Semantics*. 1990, Elsevier. p. 995-1072.
11. Adams, M., A.V. Hense, and A.H. ter Hofstede, *YAWL: An open source Business Process Management System from science for science.* SoftwareX, 2020. 12: p. 100576.
12. *Better than BPMN*. Available from: https://www.yaug.org/.
13. *YAWL*. Available from: http://www.yawlfoundation.org/yawlbook.
14. Boonyawat, S. and W. Vatanawood. *Transforming YAWL Workflows with Time Constraints to Generalized Stochastic Petri Nets*. in *Proceedings of the* 2019 3rd *International Conference on Software and e-Business*. 2019.
15. Chandratarat, P., *Transforming WS-BPEL into Timed Automata*.

ประวัติผู้เขียน

ชื่อ-สกุล	ณรงค์กร วงศ์สิทธิไพฑูรย์
วัน เดือน ปี เกิด	22 เมษายน 2539
สถานที่เกิด	นครราชสีมา
วุฒิการศึกษา	จุฬาลงกรณ์มหาวิทยาลัย
ที่อยู่ปัจจุบัน	555 หมู่4 ต.บ้านใหม่ อ.เมือง จ.นครราชสีมา 30000