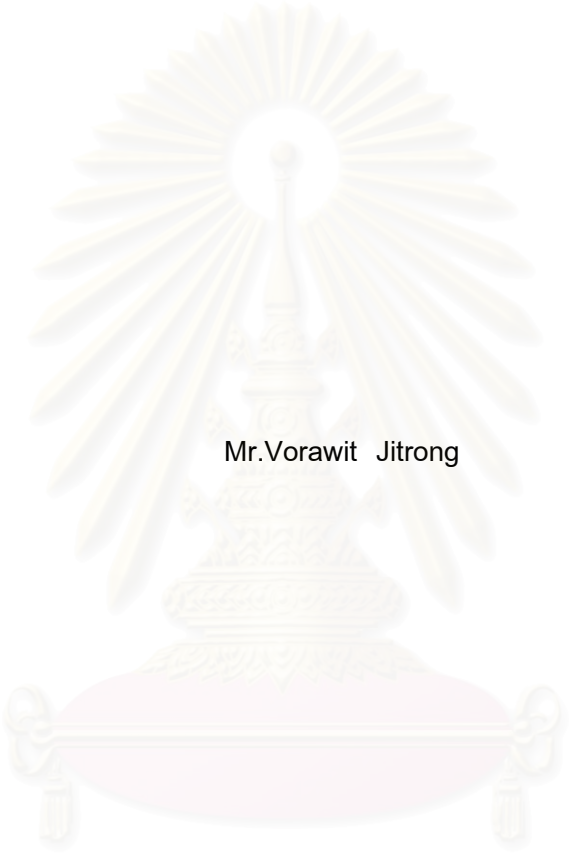
การวิเคราะห์มิวเทชันแบบหลายข้อผิดพลาดสำหรับข้อกำหนดรูปนัยแบบสัญกรณ์เซด

นายวรวิทย์ จิตรงค์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2549

MULTIPLE-FAULT MUTATION ANALYSIS FOR FORMAL SPECIFICATION IN Z NOTATION

Mr.Vorawit Jitrong



A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Computer Science
Department of Computer Engineering
Chulalongkorn University
Academic Year 2006

หัวข้อวิทยานิพนธ์	การวิเคราะห์มิวเทชันแบบหลายข้อผิดพลาดสำหรับข้อกำหนดรูปนัยแบบสัญกรณ์เซด
โดย	นายวรวิทย์ จิตรงค์
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษา	อาจารย์ ดร.อรรถสิทธิ์ สุรฤกษ์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.ดิเรก ลาวัณย์ศิริ)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์บุญชัย โสวรรณวณิชกุล)

..... อาจารย์ที่ปรึกษา
(อาจารย์ ดร.อรรถสิทธิ์ สุรฤกษ์)

..... กรรมการ
(อาจารย์ ดร.เศรษฐา ปานงาม)

..... กรรมการ
(อาจารย์ ดร.พิษณุ คนองชัยยศ)

นายวรวิทย์ จิตรงค์ : การวิเคราะห์มิวเทชันแบบหลายข้อผิดพลาดสำหรับข้อกำหนดรูปนัยแบบสัญกรณ์เซด (MULTIPLE-FAULT MUTATION ANALYSIS FOR FORMAL SPECIFICATION IN Z NOTATION). อาจารย์ที่ปรึกษา : อาจารย์ ดร.อรรถสิทธิ์ สุรฤกษ์, 78 หน้า

วิธีการวิเคราะห์มิวเทชันเป็นวิธีการวิเคราะห์ประสิทธิภาพของชุดทดสอบวิธีหนึ่งที่อาศัยการสร้างข้อผิดพลาดใส่เข้าไปในซอฟต์แวร์เสมือนเป็นการจำลองข้อผิดพลาดที่อาจจะเกิดขึ้น แต่ยังมีวิธีการวิเคราะห์มิวเทชันอีกวิธีหนึ่ง คือ การสร้างข้อผิดพลาดใส่เข้าไปในข้อกำหนดรูปนัย เพื่อนำข้อกำหนดรูปนัยที่มีการใส่ข้อผิดพลาดจำลองมาใช้กำหนดคุณสมบัติของชุดทดสอบ ข้อเสียคือ ใช้เวลาในการคำนวณสูง เนื่องจากจำนวนมิวแทนท์ของข้อกำหนด ดังนั้นงานวิจัยนี้จึงมีเป้าหมายในการลดจำนวนมิวแทนท์ของข้อกำหนด โดยที่ประสิทธิภาพของชุดทดสอบไม่ลดลง

งานวิจัยชิ้นนี้กล่าวถึงการปรับปรุงประสิทธิภาพการวิเคราะห์มิวเทชันในระดับของข้อกำหนดรูปนัย โดยใช้วิธีการวิเคราะห์มิวเทชันแบบหลายข้อผิดพลาดสำหรับข้อกำหนดรูปนัยแบบสัญกรณ์เซด เพื่อลดจำนวนมิวแทนท์ของข้อกำหนด โดยให้ข้อกำหนดมิวแทนท์หนึ่งๆ สามารถแทนหลายข้อกำหนดมิวแทนท์ และได้เสนออัลกอริทึมในการสร้างข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาด พร้อมทั้งได้ทำการพิสูจน์และทำการทดลองเพื่อยืนยันการลดลงของจำนวนมิวแทนน์ของข้อกำหนด

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....
สาขาวิชา วิทยาศาสตร์คอมพิวเตอร์ ลายมือชื่ออาจารย์ที่ปรึกษา .....
ปีการศึกษา 2549

4670470421 : MAJOR COMPUTER SCIENCE
KEY WORD : SOFTWARE TESTING / MUTATION ANALYSIS / SPECIFICATION SLICING / FORMAL SPECIFICATION / Z NOTATION

VORAWIT JITRONG : MULTIPLE-FAULT MUTATION ANALYSIS FOR FORMAL SPECIFICATION IN Z NOTATION. THESIS ADVISOR : ATHASIT SURARERKS, Ph.D., 78 pp.

A mutation analysis is one of the approaches for analyzing effectiveness of test data. This technique is fault-based testing, by inserting faults into the software as if to reproduce the faults which may occur in the specification. In addition, this approach can insert faults into the formal specification to specify the test data properties. However, since the specification provides many mutants, the mutation analysis takes high computational time. In this work, we focus on reducing the mutants of the specification without reducing the effectiveness of the test data.

This thesis proposes an improvement of mutation analysis for formal specification, using multiple-fault mutation analysis with Z notation. We concentrate on reducing the mutants of the specification, by assigning a mutant to cover more than one mutant. We propose an algorithm for generating the multiple-fault mutant. Moreover, the reduction of mutants is demonstrated with our theoretical and experimental results.

Department Computer Engineering Student's signature .....
Field of study Computer Science Advisor's signature .....
Academic year 2006

กิตติกรรมประกาศ

ขอขอบพระคุณมหาวิทยาลัยหอการค้าไทยที่อนุมัติให้ผู้วิจัยได้ลาศึกษาต่อและให้ทุนสนับสนุนแก่ผู้วิจัยตลอดการศึกษา

วิทยานิพนธ์ฉบับนี้สำเร็จได้ด้วยความเมตตา ความอนุเคราะห์ และความช่วยเหลืออย่างยิ่งจาก อาจารย์ ดร.อรรถสิทธิ์ สุรฤกษ์ อาจารย์ที่ปรึกษา ซึ่งเป็นผู้ให้ข้อคิด แนวทาง และคำปรึกษา ตลอดจนเป็นผู้ตรวจทานแก้ไข ทำให้วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วง

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์บุญชัย โสวรรณวณิชกุล ประธานกรรมการสอบวิทยานิพนธ์ อาจารย์ ดร.เศรษฐา ปานงาม อาจารย์ ดร.พิษณุ คนองชัยยศ กรรมการสอบวิทยานพนธ์ ที่ได้กรุณาให้คำแนะนำในการแก้ไขวิทยานิพนธ์ให้มีคุณภาพยิ่งขึ้น และขอขอบพระคุณคณาจารย์ในภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัยทุกท่านที่ประสิทธิ์ประสาทความรู้อันมีค่ายิ่งแก่ผู้วิจัย

ท้ายที่สุดนี้ขอขอบพระคุณ บิดา มารดา คณาจารย์ในสาขาวิชาวิทยาการคอมพิวเตอร์และในคณะวิทยาศาสตร์ มหาวิทยาลัยหอการค้าไทยที่เป็นกำลังใจสำคัญ และขอขอบคุณเพื่อนๆ น้องๆ ใน ELITE Lab ที่ผลักดันและให้ความช่วยเหลือในทุกๆ ด้านจนผู้วิจัยสามารถทำวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วง

สารบัญ

หน้า

บทที่ หน้า

สารบัญตาราง

สารบัญภาพ

ภาพที่ หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การทดสอบความถูกต้องของซอฟต์แวร์เป็นขั้นตอนที่สำคัญอย่างยิ่งในการพัฒนาซอฟต์แวร์ เพราะเป็นขั้นตอนในการรับประกันความถูกต้อง โดยการค้นหาข้อผิดพลาดที่สามารถเกิดขึ้นได้ในซอฟต์แวร์นั้น การทดสอบซอฟต์แวร์โดยส่วนใหญ่อาจทำได้ไม่สมบูรณ์มากนัก ด้วยปัจจัยข้อจำกัดทางด้านเวลาที่ใช้ในการทดสอบ โดยเวลาที่ใช้ในการทดสอบจะผันแปรตามขนาดของชุดทดสอบ ดังนั้นงานวิจัยนี้จึงมุ่งประเด็นไปที่การลดขนาดของชุดทดสอบโดยที่ประสิทธิภาพของชุดทดสอบไม่เปลี่ยนแปลง

การสร้างชุดทดสอบที่ประกอบไปด้วยกรณีทดสอบหลายกรณีนั้น ทำให้ได้ชุดทดสอบที่มีขนาดใหญ่ซึ่งหมายถึงชุดทดสอบที่มีจำนวนกรณีทดสอบมาก จำนวนกรณีทดสอบจะมากหรือน้อยขึ้นกับวิธีในการสร้างชุดทดสอบ ในปัจจุบันมีงานวิจัยเป็นจำนวนมากเสนอวิธีการสร้างชุดทดสอบที่มีประสิทธิภาพ ในที่นี้ประสิทธิภาพหมายถึงความสามารถในการตรวจจับข้อผิดพลาดที่มีอยู่ในซอฟต์แวร์ การวิเคราะห์มิวเทชัน (Mutation analysis) เป็นหนึ่งในวิธีวิเคราะห์ประสิทธิภาพของชุดทดสอบ โดยอาศัยการสร้างข้อผิดพลาด (Fault-base testing) ใส่เข้าไปในซอฟต์แวร์เสมือนการจำลองข้อผิดพลาด และตรวจสอบความสามารถของชุดทดสอบว่า สามารถตรวจจับข้อผิดพลาดจำลองทั้งหมดได้หรือไม่ วิธีการนี้ไม่ได้ใช้ในการสร้างชุดทดสอบ มีการวิเคราะห์มิวเทชันอีกรูปแบบหนึ่งคือ การใส่ข้อผิดพลาดเข้าไปในข้อกำหนดซอฟต์แวร์ (Software specification) ซึ่งเป็นข้อกำหนดที่ถูกบรรยายด้วยภาษารูปนัย (Formal language) ทั้งนี้เพื่อนำข้อกำหนดซอฟต์แวร์ที่มีข้อผิดพลาดจำลองมาใช้ในการกำหนดคุณสมบัติของชุดทดสอบ เพื่อสร้างชุดทดสอบที่มีประสิทธิภาพและนำไปทดสอบซอฟต์แวร์เมื่อซอฟต์แวร์ได้รับการพัฒนาขึ้นมา

ในการใส่ข้อผิดพลาดเข้าไปในข้อกำหนดซอฟต์แวร์นั้น มีข้อผิดพลาดมากมายที่สามารถสร้างขึ้นได้ สาเหตุที่สร้างข้อผิดพลาดเป็นจำนวนมากก็เพื่อให้ได้ชุดทดสอบที่มีความสามารถสูง หรืออีกนัยหนึ่งเพื่อให้ได้ชุดทดสอบที่ครอบคลุมการตรวจจับข้อผิดพลาดได้มาก ผลที่ตามมาคือ ได้ชุดทดสอบที่มีขนาดใหญ่ทำให้ใช้เวลาในการทดสอบนาน ในงานวิจัยนี้จะนำเสนอวิธีการลดกรณีในการสร้างชุดทดสอบที่สร้างจากวิธีการวิเคราะห์มิวเทชันเพื่อลดจำนวนกรณีทดสอบ โดยประสิทธิภาพของชุดทดสอบไม่ลดลง (ประสิทธิภาพของชุดทดสอบยังคงเดิมเมื่อเปรียบเทียบกับกรณีไม่ได้ลดกรณีทดสอบ)

1.2 วัตถุประสงค์ของการวิจัย

งานวิจัยนี้มีวัตถุประสงค์เพื่อนำเสนออัลกอริทึมการปรับปรุงการวิเคราะห์มิวเทชันสำหรับข้อกำหนดรูปนัยแบบสัญกรณ์เซด โดยการสร้างมิวแทนท์ของข้อกำหนดรูปนัยแบบหลายข้อผิดพลาด ผลลัพธ์ที่ได้คือ จำนวนของกรณีทดสอบที่ใช้ในการสร้างชุดทดสอบลดลง

1.3 ขอบเขตของการวิจัย

1.3.1 ภาษารูปนัยที่ใช้อ้างอิงจากคู่มืออ้างอิงสัญกรณ์เซด [1]

1.3.2 นำอัลกอริทึมการตัดแบ่งข้อกำหนดรูปนัยแบบสัญกรณ์เซดมาใช้ เพื่อแบ่งแยกองค์ประกอบของพฤติกรรมของระบบที่ไม่ขึ้นต่อกัน

1.3.3 กรณีทดสอบที่สร้างขึ้นถูกบรรยายในรูปของเซตและตัวดำเนินการบนเซต

1.3.4 พิสูจน์ว่า จำนวนกรณีทดสอบที่ได้จากการวิเคราะห์มิวเทชันแบบหลายข้อผิดพลาดต้องน้อยกว่าหรือเท่ากับจำนวนกรณีทดสอบด้วยวิธีวิเคราะห์มิวเทชันแบบข้อผิดพลาดเดียว

1.3.5 พิสูจน์ว่า ประสิทธิภาพของกรณีทดสอบต้องไม่ลดลง นั่นคือ กรณีทดสอบที่ได้ด้วยวิธีเดิมยังคงมีอยู่ในกรณีทดสอบที่ได้จากการวิเคราะห์มิวเทชันแบบหลายข้อผิดพลาดด้วยเช่นกัน

1.4 ขั้นตอนและวิธีดำเนินการวิจัย

1.4.1 ศึกษาและทำความเข้าใจการวิเคราะห์มิวเทชัน

1.4.2 ศึกษางานวิจัยที่เกี่ยวข้องกับการประยุกต์ใช้การวิเคราะห์มิวเทชันกับภาษารูปนัย

1.4.3 ศึกษาการบรรยายระบบซอฟต์แวร์โดยใช้ข้อกำหนดรูปนัยแบบสัญกรณ์เซด

1.4.4 ศึกษางานวิจัยที่เกี่ยวข้องในการตัดแบ่งข้อกำหนดรูปนัย

1.4.5 ศึกษางานวิจัยที่นำเอาการตัดแบ่งข้อกำหนดรูปนัยมาประยุกต์ใช้

1.4.6 ออกแบบอัลกอริทึมการวิเคราะห์มิวเทชันแบบหลายข้อผิดพลาดสำหรับข้อกำหนดรูปนัยแบบสัญกรณ์เซด

1.4.7 พิสูจน์ทฤษฎีที่เกี่ยวข้องในงานวิจัยนี้

1.4.8 สรุปผลการวิจัย และจัดทำรายงานวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับจากการวิจัย

ประโยชน์ที่คาดว่าจะได้รับจากงานวิจัยนี้ คือ การลดกรณีทดสอบในการสร้างชุดทดสอบที่สร้างจากการวิเคราะห์มิวเทชันสำหรับข้อกำหนดรูปนัยแบบสัญกรณ์เซด โดยการลดจำนวนข้อกำหนดรูปนัยที่มีการจำลองข้อผิดพลาด

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

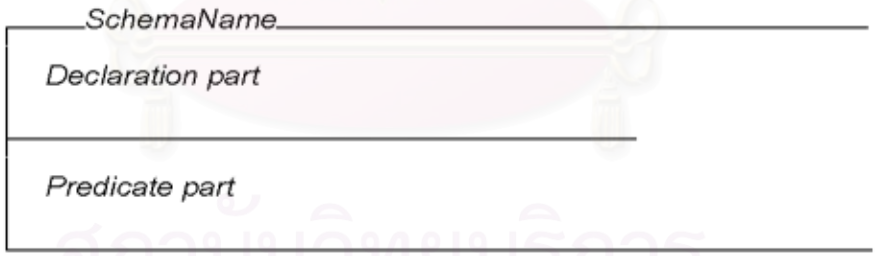
2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 สัญกรณ์เซด (Z Notation)

สัญกรณ์เซดเป็นภาษารูปนัย (Formal language) [1] ภาษาหนึ่งที่ใช้สัญกรณ์ (Notation) และทฤษฎีทางคณิตศาสตร์ที่เกี่ยวข้องกับทฤษฎีเซต (Set theory) และตรรกศาสตร์ภาคแสดง (Predicate logic) มาบรรยายการทำงานของระบบซอฟต์แวร์เพื่อลดความกำกวมและลดปัญหาความเข้าใจผิดระหว่างกลุ่มผู้พัฒนาซอฟต์แวร์ในขั้นตอนการเขียนข้อกำหนดของระบบ (System specification) หรือข้อกำหนดของโปรแกรม (Program specification)

การเขียนข้อกำหนดเพื่อบรรยายการทำงานของระบบซอฟต์แวร์ด้วยสัญกรณ์เซดนั้น จะพิจารณาจากพฤติกรรมการทำงานของระบบ ซึ่งการเขียนบรรยายมีโครงสร้างคล้ายกับภาษาการโปรแกรมเชิงโครงสร้าง (Structure programming language) นั่นคือ ข้อกำหนดที่ได้จะแสดงในรูปการเปลี่ยนแปลงค่าของตัวแปร รูปแบบการบรรยายข้อกำหนดรูปนัยแบบสัญกรณ์เซดแสดงได้ดังรูปที่ 2.1

รูปแบบที่ 1 กล่องเค้าร่าง (Schema box)



รูปแบบที่ 2 รูปแบบแนวนอน (Horizontal form)

SchemaName $\widehat{=}$ *[Declaration part | Predicate part]*

รูปที่ 2.1 รูปแบบการบรรยายเค้าร่าง (Schema) ในสัญกรณ์เซด

จากรูปที่ 2.1 เป็นการแสดงโครงสร้างทั่วไปของการเขียนข้อกำหนดแบบสัญกรณ์เซด ซึ่งจะเขียนอยู่ในรูปเค้าร่าง (Schema) ประกอบด้วย

- ชื่อเค้าร่าง (Schema name)
- ส่วนประกาศตัวแปร (Declaration part หรือ Signature part) ใช้แสดงโครงสร้างเชิงสถิต (Static) ของสิ่งที่กำลังบรรยาย นั่นคือ ตัวแปรและชนิดของ

ตัวแปร ซึ่งตามความหมายในสัญกรณ์เซดเป็นการประกาศให้ตัวแปรเป็นสมาชิกของเซต และนอกจากนี้ส่วนประกาศยังใช้แสดงเค้าร่างอื่นที่ถูกอ้างถึงหรือถูกรวมเข้ามาเพื่อนำบางส่วนมาใช้งาน โดยมีเครื่องหมายแสดงการอ้างถึง 2 ลักษณะ คือ การอ้างถึงเค้าร่างแบบที่มีการเปลี่ยนแปลงสถานะของตัวแปร (Δ) และการอ้างถึงเค้าร่างแบบที่ไม่เปลี่ยนแปลงสถานะของตัวแปร (Ξ) ตามด้วยชื่อของเค้าร่างที่ต้องการอ้างถึง

- ส่วนภาคแสดง (Predicate part) ใช้บรรยายสัจพจน์ (Axiom) ที่แสดงความสัมพันธ์ระหว่างตัวแปร โดยแบ่งเป็นภาคแสดงที่เป็นเงื่อนไขก่อน (Pre-condition) และภาคแสดงที่เป็นเงื่อนไขหลัง (Post-condition) ในส่วนนี้จะเขียนบรรยายด้วยสัญกรณ์และทฤษฎีทางคณิตศาสตร์ที่เกี่ยวกับเซต (Set) และตรรกศาสตร์ (Logic)

สัญลักษณ์ที่ใช้ประกอบกับตัวแปร

- ตัวแปรรับเข้า (Input variable) ชื่อตัวแปรจะลงท้ายด้วยเครื่องหมายคำถามหรือเครื่องหมายปรัศนี (Question mark : ?)
- ตัวแปรส่งออก (Output variable) ชื่อตัวแปรจะลงท้ายด้วยเครื่องหมายอุทานหรือเครื่องหมายอัศเจรีย์ (Exclamation mark : !)
- ตัวแปรที่มีการเปลี่ยนแปลงสถานะจะลงท้ายด้วยเครื่องหมายไพรม์หรือฝนทอง (Prime mark : ')

สัญกรณ์ที่มีใช้ในเซด เช่น

- $\mathbb{P}$ คือ เซตกำลัง (Power set)
- $\varnothing$ คือ เซตว่าง (Empty set)
- $\rightarrow$ คือ ฟังก์ชันแบบบางส่วน (Partial function)
- $\mapsto$ คือ ความสัมพันธ์จากสมาชิกตัวหน้าไปยังสมาชิกตัวหลัง (Maplet) ใช้แทนการเขียนคู่ลำดับ
- $\in$ คือ การเป็นสมาชิกของเซต
- $\cup$ คือ การนำเซตมารวมกัน
- $\cap$ คือ การรวมกันของสมาชิกที่ซ้ำกันของเซต
- $\setminus$ คือ เซตของสมาชิกของเซตตัวหน้า

จากตัวอย่างในรูปที่ 2.2 เป็นการแสดงส่วนของข้อกำหนดรูปนัยแบบสัญกรณ์เซดของระบบสมุดบันทึกวันเกิดซึ่งประกอบด้วย

- [*NAME*, *DATE*] คือ การกำหนดเซตของชื่อและวันที่ให้เป็นชนิดข้อมูลพื้นฐาน (Basic type)
- BB เป็นเค้าร่างวัตถุ (Object schema) ใช้กำหนดตัวแปรสถานะ (State variable) และเงื่อนไข (Constrain) ที่จะต้องสอดคล้องตลอดการทำงานของระบบ
- initBB เป็นเค้าร่างที่ใช้กำหนดค่าเริ่มต้น (Initialization schema)
- Add และ Delete เป็นเค้าร่างดำเนินการ (Operation schema)

รูปที่ 2.2 ส่วนของข้อกำหนดรูปนัยแบบสัญกรณ์เชดของระบบสมุดบันทึกวันเกิด [2]

2.1.2 การทดสอบโดยการอ้างอิงข้อผิดพลาด (Fault-based testing)

การทดสอบโดยการอ้างอิงข้อผิดพลาดเป็นวิธีหนึ่งในการทดสอบข้อผิดพลาดที่มีอยู่ในซอฟต์แวร์ ข้อดีของวิธีการนี้คือ สามารถยืนยันความบกพร่องที่เกิดจากข้อผิดพลาดได้ การทดสอบโดยการอ้างอิงสามารถระบุกลุ่มของข้อผิดพลาดได้เพราะว่าวิธีการนี้จะทำการทดสอบทุกๆ ข้อผิดพลาดที่เป็นไปได้ ซึ่งในการทดสอบที่ผ่านมานั้นผลการทดสอบจะมีประสิทธิผลต่อการทดสอบข้อผิดพลาดในกลุ่มของข้อผิดพลาดอื่นด้วย

กลุ่มของข้อผิดพลาด

ข้อผิดพลาดในที่นี้หมายถึงข้อผิดพลาดจากตัวแปรบูลลีน (Boolean variables) ข้อผิดพลาดจากตัวดำเนินการบูลลีน (Boolean operators) ข้อผิดพลาดจากตัวดำเนินการความสัมพันธ์ (Relational operators) หรือข้อผิดพลาดจากนิพจน์ทางคณิตศาสตร์ (Arithmetic expressions) นอกจากนี้ยังมีข้อผิดพลาดที่เกิดจากอนุประโยค (Clause) ต่างๆ ในการพิจารณาข้อผิดพลาดจะเริ่มจากการพิจารณาข้อผิดพลาดข้างต้น และตามด้วยข้อผิดพลาดที่มีผลกระทบต่อภาคแสดง ซึ่งก็คือข้อผิดพลาดจากอนุประโยคนั่นเอง ตัวอย่างของอนุประโยคต่างๆ มีดังนี้

- ข้อผิดพลาดที่เกิดจากการอ้างอิงอนุประโยคอื่น (Clause Reference Fault : CRF) เป็นการแทนที่ด้วยอนุประโยคอื่น
- ข้อผิดพลาดจากการปฏิเสธอนุประโยค (Clause Negation Fault : CNF) เป็นการแทนที่ด้วยอนุประโยคที่เป็นนิเสธ
- ข้อผิดพลาดจากการแทรกอนุประโยค (Clause Insertion Fault : CIF) เป็นการเพิ่มอนุประโยคเข้าไป มีสองประเภทย่อยได้แก่
  - อนุประโยค "และ" (Clause Conjunction Fault : CCF)
  - อนุประโยค "หรือ" (Clause Disjunction Fault : CDF)
- ข้อผิดพลาดจากตัวดำเนินการความสัมพันธ์ (Relational Operator Fault : ROF) เป็นการแทนตัวดำเนินการความสัมพันธ์อื่น เช่น การแทนตัวดำเนินการความสัมพันธ์ที่เป็นตัวตรงข้าม นั่นคือ การใส่นิเสธเข้าไปในนิพจน์ความสัมพันธ์นั่นเอง
- ข้อผิดพลาดจากการบวก 1 หรือลบ 1 เข้ากับนิพจน์ (Off-By-1 Fault : OFF)
- ข้อผิดพลาดจากการแทนนิพจน์ด้วย 0 หรือ 1 (Stuck-At Fault : STF)

- ข้อผิดพลาดจากการเขียนอนุประโยคผิด (Miss Clause Fault : MCF) เป็นการใช้อนุประโยคที่มีการตกหล่นบางส่วน

การวิเคราะห์มิวเทชันจัดเป็นเทคนิคหนึ่งในการทดสอบโดยการอ้างอิงข้อผิดพลาดที่มีการใช้ตัวดำเนินการมิวเทชันใส่เข้าไปในโปรแกรมหรือข้อกำหนด แล้วเลือกกรณีทดสอบที่สามารถแยกความแตกต่างของตัวมิวแทนท์กับตัวต้นฉบับได้

สิ่งสำคัญประการหนึ่งสำหรับการทดสอบโดยการอ้างอิงข้อผิดพลาดคือ เงื่อนไขในการค้นหาข้อผิดพลาด (Fault detection condition) โดยการพิจารณาจากข้อบังคับหรือข้อจำกัดของตัวดำเนินการที่ใช้กับตัวแปร เงื่อนไขในการค้นหาข้อผิดพลาดเป็นเครื่องมือวิเคราะห์ที่มีประสิทธิผลและรัดกุมในการเรียนรู้ข้อผิดพลาดของข้อกำหนด สามารถแบ่งได้สองรูปแบบคือ

รูปแบบที่ 1 เงื่อนไขที่จำเป็น (Necessary condition) เป็นข้อบังคับของตัวแปรในนิพจน์ย่อย (Sub-expression) ในภาคแสดงที่ถูกวิเคราะห์มิวเทชัน ทำให้มีความแตกต่างไปจากต้นฉบับ แต่เงื่อนไขที่จำเป็นอาจไม่เพียงพอต่อการค้นหาข้อผิดพลาด เพราะไม่ส่งผลต่อตัวแปรส่งออก

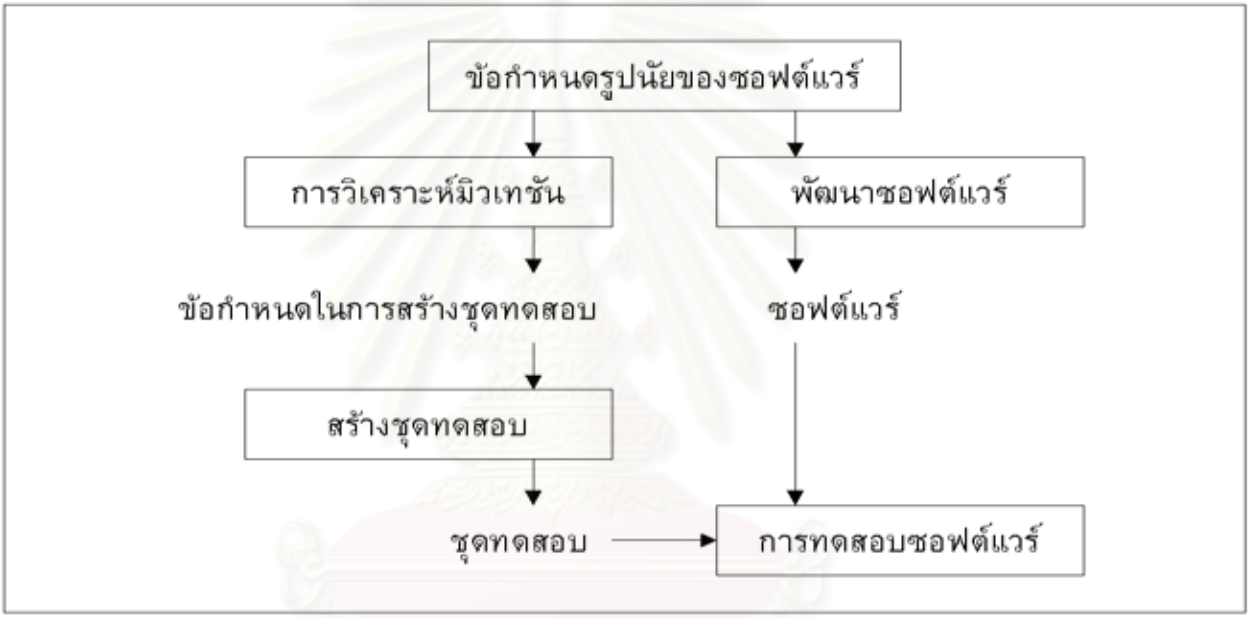
รูปแบบที่ 2 เงื่อนไขที่เพียงพอ (Sufficient condition) เป็นข้อบังคับค่าของตัวแปรส่งออกที่จะต้องแตกต่างจากต้นฉบับ

ความแตกต่างระหว่างสองรูปแบบนี้คือ เป็นการทดสอบมิวเทชันแบบอ่อน (Weak) กับแบบแรง (Strong) [3] และข้อกำหนดของเงื่อนไขที่จำเป็นใช้ง่ายและนิยมมากกว่าเงื่อนไขที่เพียงพอ

2.1.3 การวิเคราะห์มิวเทชัน (Mutation analysis)

วิธีการทดสอบซอฟต์แวร์ด้วยหลักการของการวิเคราะห์มิวเทชัน อาศัยการจำลองข้อผิดพลาดที่อาจเกิดขึ้นในระหว่างการพัฒนาซอฟต์แวร์ การวิเคราะห์มิวเทชันถูกนำมาใช้กับการทดสอบซอฟต์แวร์เป็นครั้งแรกในปี 1991 เสนอโดย เดอมิโล [4] การใส่ข้อผิดพลาดหนึ่งๆ เข้าไปในซอฟต์แวร์หรือโปรแกรม เปรียบเสมือนการจำลองเหตุการณ์ที่ซอฟต์แวร์หรือโปรแกรมทำงานผิดพลาด วัตถุประสงค์ของกระบวนการนี้คือ การปรับปรุงคุณภาพของชุดทดสอบ โดยชุดทดสอบที่ถูกสร้างขึ้นมานั้น ถ้าสามารถแยกความแตกต่างระหว่างซอฟต์แวร์ที่ไม่มีข้อผิดพลาดออกจากซอฟต์แวร์ที่ใส่ข้อผิดพลาดเข้าไปได้ แสดงว่า ชุดทดสอบนั้นมีประสิทธิภาพสูง ต่อมาแนวคิดวิธีการวิเคราะห์มิวเทชันได้ถูกนำมาพัฒนาใช้ในการสร้างชุดทดสอบเช่นกัน โดยใช้การวิเคราะห์มิวเทชันกับข้อกำหนดรูปนัยของซอฟต์แวร์ [5] นั่นคือ ใส่ข้อผิดพลาดเข้าไปที่ข้อกำหนดรูปนัย เพื่อตรวจสอบความแตกต่างที่เป็นไปได้ในกรณีที่

ข้อกำหนดผิดพลาดไปจากต้นฉบับ วัตถุประสงค์ที่แท้จริงคือ การค้นหากรณีที่เป็นไปได้ที่สามารถตรวจจับความแตกต่างดังกล่าว และกรณีเหล่านั้นจะถูกนำไปใช้เป็นคุณสมบัติของชุดทดสอบต่อไป โดยกระบวนการดังกล่าวตั้งอยู่บนพื้นฐานที่ว่า ถ้าซอฟต์แวร์ที่พัฒนาขึ้นมานั้นมีข้อผิดพลาดจริง ข้อผิดพลาดนั้นจะสอดคล้องกับข้อกำหนดรูปนัยที่ผิดพลาดด้วยเช่นกัน ดังนั้นชุดทดสอบที่สามารถตรวจจับข้อผิดพลาดของซอฟต์แวร์ได้จะเป็นชุดทดสอบที่เกิดจากความแตกต่างของข้อกำหนดรูปนัยที่ผิดพลาดด้วยเช่นเดียวกัน ชุดทดสอบที่ได้จากวิธีการวิเคราะห์มิวเทชันนี้จะเป็นชุดทดสอบที่ทดสอบว่า ซอฟต์แวร์ที่สร้างขึ้นสอดคล้องกับข้อกำหนดรูปนัยหรือไม่ โดยกระบวนการดังกล่าวสามารถแสดงได้ดังรูปที่ 2.3 ซึ่งต่างจากวิธีการของเดอมิโลที่ใช้ทดสอบประสิทธิภาพของชุดทดสอบ



รูปที่ 2.3 กระบวนการในการวิเคราะห์มิวเทชันบนข้อกำหนดรูปนัย

นอกจากนี้ยังมีงานวิจัยในปี 2000 ของ แบล็ค [6] ที่สนใจการวิเคราะห์มิวเทชันสำหรับโปรแกรมตรวจสอบตัวแบบด้วยเช่นกัน ซึ่งมีลักษณะการทำงานคล้ายกับการวิเคราะห์มิวเทชันสำหรับข้อกำหนดรูปนัย

นิยามพื้นฐานที่เกี่ยวข้องกับการวิเคราะห์มิวเทชันบนข้อกำหนดรูปนัยนั้น มีดังต่อไปนี้

นิยามที่ 1 *ข้อผิดพลาด (Fault) คือ สิ่งที่แสดงถึงการมีอยู่ของความผิดพลาด*

นิยามที่ 2 *ผลกระทบของข้อผิดพลาด (Incident) เป็นผลลัพธ์ที่ได้รับจากการมีอยู่ของข้อผิดพลาด ซึ่งผลกระทบนี้สามารถแสดงออกมาให้ผู้ใช้หรือผู้ทดสอบเห็นได้*

นิยามที่ 3 *ข้อกำหนดรูปนัย M ถูกเรียกว่า* ข้อกำหนดรูปนัยมิวแทนท์ *ที่เกิดจากข้อกำหนดรูปนัย S หาก M เกิดจากการเปลี่ยนแปลงโครงสร้างของข้อกำหนดรูปนัย S โดยการเปลี่ยนแปลงที่ได้นิยามไว้ในตัวดำเนินการมิวเทชัน*

ในการใส่ข้อผิดพลาดเข้าไปในข้อกำหนดรูปนัยแบบสัญกรณ์เซตนั้นจะกระทำเฉพาะส่วนของภาคแสดงที่ถูกกำหนดในเค้าร่างดำเนินการ งานวิทยานิพนธ์ปริญญาเอกของสต๊อก [5] ในปี 1993 ได้แบ่งกลุ่มของตัวดำเนินการมิวเทชันเพื่อใส่ข้อผิดพลาดที่สามารถใส่เข้าไปในข้อกำหนดรูปนัยออกเป็น 3 กลุ่ม คือ

กลุ่มที่ 1 ตัวดำเนินการมิวเทชันที่ใส่ข้อผิดพลาดที่ประเภทของตัวแปร (Type mutation) เป็นข้อผิดพลาดเนื่องจากการระบุประเภทของตัวแปรผิดไป โดยการนำประเภทของตัวแปรอื่นมาใส่แทน โดยไม่ทำให้ขัดต่อไวยกรณ์ของภาษาข้อกำหนดรูปนัยนั้น

กลุ่มที่ 2 ตัวดำเนินการมิวเทชันที่ใส่ข้อผิดพลาดที่ตัวแปร (Variable mutation) เป็นข้อผิดพลาดที่ใส่ให้กับตัวแปร แต่ไม่ใช่ประเภทของตัวแปร เช่น การสลับตำแหน่งของตัวแปรหรือการเปลี่ยนตัวแปร โดยไม่ทำให้ขัดต่อไวยกรณ์ของภาษาข้อกำหนดรูปนัยนั้น

กลุ่มที่ 3 ตัวดำเนินการมิวเทชันที่ใส่ข้อผิดพลาดที่ตัวดำเนินการ (Operator mutation) เป็นข้อผิดพลาดที่เกิดจากการเปลี่ยนตัวดำเนินการ เช่น การเปลี่ยนตัวดำเนินการยูเนียนเป็นตัวดำเนินการอินเตอร์เซคชัน เป็นต้น โดยไม่ทำให้ขัดต่อไวยกรณ์ของภาษาข้อกำหนดรูปนัยนั้น

ในปี 2000 แบล็ค [6, 7] ได้เสนอประเภทของตัวดำเนินการมิวเทชันเพื่อใส่ข้อผิดพลาดสำหรับภาษาตรวจสอบตัวแบบ (Model checker) สรุปได้ดังตารางที่ 2.1

ตารางที่ 2.1 แสดงตัวดำเนินการมิวเทชันของแบล็ค

ตัวดำเนินการ	ความหมาย
ORO : Operand Replacement Operator	แทนที่ตัวถูกดำเนินการ
SNO : Simple Expression Negation Operator	นิเสธของการบรรยายแบบง่าย
ENO : Expression Negation Operator	นิเสธคำบรรยาย
MCO : Missing Condition Operator	เงื่อนไขสูญหาย
STO : Stuck-At Operator	นิเสธ
ASO : Associative Shift Operator	เลื่อนการจัดหมู่
RRO : Relational Operator Replacement Operator	เปลี่ยนตัวดำเนินการความสัมพันธ์

2.2 งานวิจัยที่เกี่ยวข้อง

2.2.1 งานวิจัย Mutation Operators for Object-Z Specification [8]

งานวิจัยชิ้นนี้ได้ค้นหาความผิดพลาดที่อาจจะเกิดขึ้นในข้อกำหนดเชิงอ็อบเจ็กต์เซด และได้นำเสนอตัวดำเนินการมิวเทชันที่ใช้จำลองความผิดพลาดสำหรับข้อกำหนดเชิงอ็อบเจ็กต์เซดแบ่งเป็น 5 กลุ่ม โดยสองกลุ่มแรกเป็นตัวดำเนินการมิวเทชันสำหรับข้อกำหนดเซด นั่นคือ

กลุ่มตัวดำเนินการดั้งเดิม (Traditional mutation operators) ประกอบด้วย

1) การแทนที่ตัวดำเนินการ (Operator Replacement : OR)
2) การแทนที่ชนิดข้อมูล (Type Replacement : TR)
3) การแทนที่ตัวแปรอ้างอิง (Variable Reference Replacement : VRR)

กลุ่มคุณลักษณะเฉพาะของเซด (Z-specific features) ประกอบด้วย

1) การละค่าคงที่สถานะ (Omit State Invariant : OSI)
2) การละเงื่อนไขก่อน (Omit Precondition : OP)
3) การแทนที่เครื่องหมายซีต้าด้วยเดลต้า (Replace Ξ by Δ : RXD)
4) การเปลี่ยนแปลงจำนวนอาร์กิวเมนต์ (Renaming Arguments Number Change : RAN)
5) การเปลี่ยนแปลงลำดับอาร์กิวเมนต์ (Renaming Arguments Order Change : RAOC)

2.2.2 งานวิจัย Specification Slicing in Formal Methods of Software Development. [9]

งานวิจัยชิ้นนี้ได้นำเสนออัลกอริทึมการตัดแบ่งข้อกำหนดเพื่อช่วยในการแก้ไขจุดบกพร่อง (Debugging) การเพิ่มเติม (Modification) และการใช้ซ้ำ (Reuse) เนื่องจากผลลัพธ์ที่ได้คือ ข้อกำหนดรูปนัยที่เล็กลง ดังนั้นเมื่อต้องการแก้ไข ต่อเติมหรือนำกลับมาใช้ใหม่สามารถพิจารณาได้ง่ายขึ้นเพราะมีการตัดส่วนที่ไม่เกี่ยวข้องออกไป

2.2.3 งานวิจัย A Slicing approach for Parallel Component Adaptation. [10]

งานวิจัยชิ้นนี้ได้เสนออัลกอริทึมในการตัดแบ่งข้อกำหนดรูปนัย โดยใช้หลักการแบ่งแยกองค์ประกอบของซอฟต์แวร์ ดังแสดงในรูปที่ 2.4 ในงานวิจัยชิ้นนี้เสนอแนวทางในการแบ่งความสัมพันธ์ของตัวแปรออกเป็นกลุ่มสำหรับข้อกำหนดภาษา DRIO ซึ่งเป็นภาษาที่ใช้สำหรับบรรยายโปรแกรมตรวจสอบตัวแบบ (Model checker) ประเภทหนึ่งสำหรับซอฟต์แวร์ที่มีการประกอบกันจากหลายองค์ประกอบ (Component-based software)

D_s คือ เซตของตัวแปรรับเข้าของข้อกำหนด

R_s คือ เซตของตัวแปรส่งออกของข้อกำหนด

I_s คือ เซตของภาคแสดงที่เป็นเงื่อนไขก่อนของข้อกำหนด

O_s คือ เซตของภาคแสดงที่เป็นเงื่อนไขหลังของข้อกำหนด

INIT : $R_s \leftarrow$ criterion

$$O_s \leftarrow \{o \mid \forall o \in O, \exists r \in R_s, \text{constrains}(o,r)\}$$

$$D_s \leftarrow \varnothing$$

$$I_s \leftarrow \varnothing$$

1. $O_s' \leftarrow O_s \cup \{o \mid \forall o \in O, \exists x \in O_s, \text{dDepend}(o,r)\}$
2. Repeat step 1 until $O_s' = O_s$
3. $I_s' \leftarrow I_s \cup \{i \mid \forall i \in I, \exists x \in O_s, \text{cDepend}(x,i)\}$
4. $I_s' \leftarrow I_s \cup \{i \mid \forall i \in I, \exists y \in I_s, \text{dDepend}(i,y)\}$
5. Repeat step 1 until $I_s' = I_s$
6. $D_s' \leftarrow D_s \cup \{d \mid \forall d \in D, \exists o \in O_s, \text{requires}(o,d)\}$
7. $D_s' \leftarrow D_s \cup \{d \mid \forall d \in D, \exists i \in I_s, \text{constrains}(i,d)\}$
8. $D_s' \leftarrow D_s \cup \{r \mid \forall r \in R, \exists o \in O_s, \text{requires}(o,r)\}$
9. $R_s' \leftarrow R_s \cup \{r \mid \forall r \in R, \exists o \in O_s, \text{constrains}(o,r)\}$

รูปที่ 2.4 อัลกอริทึมการตัดแบ่งข้อกำหนดภาษา DRIO

บทที่ 3

การวิเคราะห์มิวเทชันแบบหลายข้อผิดพลาด

ในบทนี้จะนำเสนอแนวทางการลดจำนวนข้อกำหนดมิวแทนท์ที่ได้จากการวิเคราะห์มิวเทชันของข้อกำหนดรูปนัยแบบสัญกรณ์เชด โดยที่แต่ละข้อกำหนดมิวแทนท์เกิดจากการใส่ข้อผิดพลาดเข้าไปในข้อกำหนดรูปนัยแบบสัญกรณ์เชดของซอฟต์แวร์ที่ต้องการ และผลลัพธ์ที่ได้จากการวิเคราะห์มิวเทชันจะแสดงอยู่ในรูปการบรรยายด้วยเซตและตัวดำเนินการบนเซต

3.1 แนวคิด

ในงานวิจัยนี้มีวัตถุประสงค์ในการลดปริมาณกรณีทดสอบ (กรณีที่เป็นไปได้ที่สามารถตรวจจับความแตกต่าง) ที่ได้จากการวิเคราะห์มิวเทชันของข้อกำหนดรูปนัยแบบสัญกรณ์เชด โดยกรณีทดสอบแต่ละกรณีเกิดจากการใส่ข้อผิดพลาดเข้าไปในข้อกำหนดรูปนัยแบบสัญกรณ์เชด ดังนั้นในงานวิจัยนี้จึงมุ่งลดปริมาณของผลลัพธ์ที่ได้จากการวิเคราะห์มิวเทชัน ซึ่งคือ จำนวนมิวแทนท์ของข้อกำหนดรูปนัย โดยนำแนวความคิดของการใส่ข้อผิดพลาดหลายข้อผิดพลาดเข้าไปในมิวแทนท์เดียวกัน และเรียกว่า *มิวแทนท์แบบหลายข้อผิดพลาด* (*Multiple-fault mutation*) วิธีนี้จะรับประกันได้ว่าถึงแม้จำนวนมิวแทนท์ลดลง แต่ประสิทธิภาพของชุดทดสอบที่สร้างขึ้นจะไม่ลดลงไปด้วย

งานวิจัยนี้สามารถแบ่งเป็นขั้นตอนในการพิจารณาได้ดังนี้

1. การวิเคราะห์ผลกระทบจากการวิเคราะห์มิวเทชัน เนื่องจากวิธีการวิเคราะห์มิวเทชันสนใจตัวดำเนินการที่กระทำกับภาคแสดงของเค้าร่างที่กำหนดในข้อกำหนดรูปนัยเท่านั้น ดังนั้น ผลกระทบต่างๆ ที่เกิดขึ้นจะมีผลต่อตัวแปรเท่านั้น มิได้มีผลกระทบต่อเส้นทางการทำงานแต่อย่างใด ตัวแปรทั้งหมดในข้อกำหนดรูปนัยถูกอ้างอิงด้วยประเภทของตัวแปรเหล่านั้น ซึ่งตัวแปรทั้งหมดที่เป็นประเภทเดียวกันถูกจัดรวบรวมไว้ด้วยกันและแสดงในรูปของเซต ดังนั้นผลกระทบทั้งหมดจึงสามารถถูกตรวจจับได้ด้วยการศึกษาการเปลี่ยนแปลงค่าของตัวแปรที่ระบุในข้อกำหนดรูปนัย โดยค่าที่เป็นไปได้ของตัวแปรสามารถแสดงได้ด้วยเซต

2. การวิเคราะห์ผลกระทบระหว่างข้อผิดพลาด ในขั้นตอนนี้เป็นการวิเคราะห์ความสัมพันธ์ระหว่างตัวแปรต่างๆ ที่ได้รับผลกระทบจากตัวดำเนินการมิวเทชัน เป็นการศึกษาถึงผลกระทบแต่ละประเภทของตัวดำเนินการ และพิจารณาว่าตำแหน่งใดในข้อกำหนดรูปนัยที่สามารถถูกตัวดำเนินการมิวเทชันกระทำได้พร้อมกัน ในงานวิจัยนี้จะทำการแบ่งข้อกำหนดรูปนัยออกเป็นส่วนๆ ตามการขึ้นต่อกัน (Dependency) ของตัวแปรทั้งหมด และแยกพิจารณาเป็น

กลุ่ม ตัวแปรที่อยู่ในกลุ่มเดียวกันน่าจะสามารถถูกกระทำได้พร้อมกัน แต่ทั้งนี้ต้องขึ้นกับประเภทของตัวดำเนินการมิวเทชันด้วย

แนวคิดของการทำการแบ่งส่วนข้อกำหนดรูปนัยนั้น สามารถนำมาเป็นแนวทางในการพิจารณาความสัมพันธ์ของตัวแปรได้ส่วนหนึ่ง ในงานวิจัยนี้จะนำแนวทางการแบ่งส่วนของข้อกำหนดรูปนัยมาเป็นตัวแบบเริ่มต้นการวิเคราะห์ต่อไป

3. การวิเคราะห์ประสิทธิภาพของวิธีการสร้างกรณีทดสอบแบบหลายข้อผิดพลาด ในการวัดประสิทธิภาพของวิธีการสร้างกรณีทดสอบแบบหลายข้อผิดพลาดนั้น สามารถทำได้โดยการพิสูจน์ว่า อัลกอริทึมที่เสนอจะทำให้จำนวนกรณีทดสอบที่ได้น้อยกว่าหรือเท่ากับจำนวนกรณีทดสอบที่ได้จากการวิเคราะห์มิวเทชันแบบข้อผิดพลาดเดียว นอกจากนี้ในการพิสูจน์ว่าประสิทธิภาพไม่ลดลงในเชิงของการครอบคลุมข้อผิดพลาดสามารถแสดงได้ โดยการพิสูจน์ว่ากรณีทดสอบที่ได้ด้วยวิธีเดิมยังคงมีอยู่ในกรณีทดสอบที่ได้จากการวิเคราะห์มิวเทชันแบบหลายข้อผิดพลาดด้วยเช่นกัน

3.2 การวิเคราะห์ผลกระทบจากการวิเคราะห์มิวเทชัน

ในการวิเคราะห์ผลกระทบจากการวิเคราะห์มิวเทชันนั้น สามารถกล่าวได้ว่าผลกระทบทั้งหมดที่เกิดจากการใส่ข้อผิดพลาดเข้าไป จะทำให้ค่าของตัวแปรที่เกี่ยวข้องกับข้อผิดพลาดนั้นมีค่าเปลี่ยนแปลงไปได้ การศึกษาถึงผลการเปลี่ยนแปลงนี้จะพิจารณาจากเซตของค่าที่เป็นไปได้ทั้งหมดของตัวแปรแต่ละตัว เมื่อกำหนดให้ S_O หมายถึง เซตของค่าที่เป็นไปได้ทั้งหมดของตัวแปรต้นฉบับ และกำหนดให้ S_M หมายถึง เซตของค่าที่เป็นไปได้ทั้งหมดของตัวแปรที่เปลี่ยนแปลงไปเมื่อมีการใส่ข้อผิดพลาดเข้าไป สำหรับตัวแปรใดๆ การเปลี่ยนแปลงที่เกิดขึ้นซึ่งคือ การเปลี่ยนแปลงสมาชิกในเซตทั้งสอง (S_O และ S_M) สามารถสรุปเป็นรูปแบบได้สี่รูปแบบด้วยกัน ดังต่อไปนี้

รูปแบบที่ 1 ตัวแปรที่มีค่าที่เปลี่ยนแปลงไปไม่มีส่วนที่เกี่ยวข้องกับค่าที่เป็นไปได้ของตัวแปรต้นฉบับ รูปแบบนี้เรียกว่า *ไร้สมาชิกร่วม* (*Disjoint*) นั่นคือ $S_M \cap S_O = \varnothing$

รูปแบบที่ 2 ตัวแปรที่มีค่าที่เปลี่ยนแปลงไปเป็นส่วนหนึ่งของค่าที่เป็นไปได้ของตัวแปรต้นฉบับ รูปแบบนี้เรียกว่า *เซตย่อย* (*Subset*) นั่นคือ $S_M \subset S_O$

รูปแบบที่ 3 ตัวแปรที่มีค่าที่เปลี่ยนแปลงไปครอบคลุมค่าที่เป็นไปได้ของตัวแปรต้นฉบับ รูปแบบนี้เรียกว่า *ซูเปอร์เซต* (*Superset*) นั่นคือ $S_M \supset S_O$

รูปแบบที่ 4 ตัวแปรที่มีค่าที่เปลี่ยนแปลงไปมีส่วนที่เกี่ยวข้องกับค่าที่เป็นไปได้ของตัวแปรต้นฉบับ รูปแบบนี้เรียกว่า *ซ้อนเหลื่อม* (*Overlap*) นั่นคือ $S_M \cap S_O \neq \varnothing$) and ($\neg S_M \subset S_O$) and ($\neg S_M \supset S_O$))

ทั้งนี้หากผลกระทบจากการเปลี่ยนแปลงที่เกิดกับเซตของตัวแปรอยู่ในรูปแบบที่ 3 จะทำให้ไม่สามารถหาเงื่อนไขที่สามารถตรวจสอบความผิดพลาดที่เกิดจากการเปลี่ยนแปลงนั้นได้ ซึ่งในงานวิจัยนี้จะหลีกเลี่ยงการสร้างข้อกำหนดมิวแทนท์ที่อยู่ในรูปแบบที่ 3

ดังนั้นในการศึกษาถึงผลกระทบในที่นี้จะพิจารณาจากผลกระทบที่มีต่อเซตของตัวแปรทั้งสี่แบบนี้เท่านั้น

ตัวอย่างผลกระทบจากตัวดำเนินการมิวเทชัน แสดงได้ดังตัวอย่างที่ 3.1

ตัวอย่างที่ 3.1 *จากเค้าร่าง CheckIn ในข้อกำหนดรูปนัยแบบสัญกรณ์เชดต่อไปนี้*

```
001  [Staff]
002  QueryReply ::= is_in | is_out
003  ┌──Log──────────────────────────
004  │ users, in, out: ℙ Staff
005  ├──────────────────────────────
006  │ in ∩ out = ∅
007  │ in ∪ out = users
008  └──────────────────────────────
009  ┌──InitLog──────────────────────
010  │ Log
011  ├──────────────────────────────
012  │ users = ∅
013  │ in = ∅
014  │ out = ∅
015  └──────────────────────────────
016  ┌──CheckIn──────────────────────
017  │ ΔLog
018  │ name?: Staff
019  ├──────────────────────────────
020  │ name? ∈ out
```

021	$in' = in \cup \{name?\}$
022	$out' = out \setminus \{name?\}$
023	$users' = users$
024	

จากภาคแสดง $in' = in \cup \{name?\}$ ในบรรทัดที่ 021 ให้ความหมายว่า เซต *in* ซึ่งมีสมาชิกอยู่จำนวนหนึ่งเมื่อนำมายูเนียนกับเซต $\{name?\}$ ซึ่งไม่อยู่ในเซต *in* ทำให้เซต *in* ใหม่ที่ได้มีสมาชิกเพิ่มขึ้น

เมื่อกำหนดให้ตัวดำเนินการมิวเทชันเปลี่ยนภาคแสดงนี้เป็น

$$in' = in \cap \{name?\}$$

ผลลัพธ์ที่ได้จะมีผลกระทบกับเซต *in* นั่นคือ เท่ากับเซตว่าง หมายความว่า เซต *in* ซึ่งมีสมาชิกอยู่จำนวนหนึ่งเมื่อนำมาอินเตอร์เซ็กชันกับเซต $\{name?\}$ ซึ่งไม่อยู่ในเซต *in* ทำให้เซต *in* ใหม่ที่ได้เป็นเซตว่าง

หากพิจารณาถึงผลกระทบจากการวิเคราะห์มิวเทชัน จะได้ว่า เซต *in* ที่ได้หลังจากมีการเปลี่ยนแปลงจะเป็นเซตย่อยของเซต *in* ก่อนมีการเปลี่ยนแปลง นั่นคือ การเปลี่ยนแปลงจัดอยู่รูปแบบที่ 2

หรือเมื่อกำหนดให้ตัวดำเนินการมิวเทชันเปลี่ยนภาคแสดงนี้เป็น

$$in' = in \setminus \{name?\}$$

ผลลัพธ์ที่ได้จะมีผลกระทบกับเซต *in* นั่นคือ เท่ากับเซต *in* หมายความว่า เซต *in* ซึ่งมีสมาชิกอยู่จำนวนหนึ่งเมื่อนำเซต $\{name?\}$ ซึ่งไม่อยู่ในเซต *in* มาลบออก ทำให้เซต *in* ใหม่ที่ได้เท่ากับเซต *in* เดิม

หากพิจารณาถึงผลกระทบจากการวิเคราะห์มิวเทชัน จะได้ว่า เซต *in* ที่ได้หลังจากมีการเปลี่ยนแปลงจะเท่ากับเซต *in* ที่ได้ก่อนมีการเปลี่ยนแปลง นั่นคือ การเปลี่ยนแปลงจัดอยู่ในรูปแบบที่ 4 □

แนวทางในการใส่ข้อผิดพลาดมากกว่าหนึ่งข้อผิดพลาดเข้าไปในข้อกำหนดรูปนัยนั้น จะมีข้อจำกัดที่ห้ามมีการใส่ข้อผิดพลาดที่ทำให้ไม่สามารถตรวจจับได้ เช่น ข้อผิดพลาดที่สามารถหักล้างได้จากอีกข้อผิดพลาด ดังนั้นสองข้อผิดพลาดนี้ไม่สามารถนำมารวมกันได้ เป็นต้น

ตัวอย่างผลกระทบจากตัวดำเนินการมิวเทชันสองตัวดำเนินการ แสดงได้ดังตัวอย่างที่ 3.2

ตัวอย่างที่ 3.2 *อาศัยเค้าร่าง CheckIn ในตัวอย่างที่ 3.1*

จากภาคแสดง $in' = in \cup \{name?\}$ ในบรรทัดที่ 021 ให้ความหมายว่า เซต **in** ซึ่งมีสมาชิกอยู่จำนวนหนึ่งเมื่อนำมายูเนียนกับเซต $\{name?\}$ ซึ่งไม่อยู่ใน เซต **in** ทำให้เซต **in** ใหม่ที่ได้มีสมาชิกเพิ่มหนึ่งคน

เมื่อกำหนดให้ตัวดำเนินการมิวเทชันเปลี่ยนภาคแสดงนี้เป็น

$$in' \supset in \cap \{name?\}$$

จะได้ว่าเซต *in* ซึ่งมีสมาชิกอยู่จำนวนหนึ่งเมื่อนำมาอินเตอร์เซ็กชันกับเซต $\{name?\}$ ซึ่งไม่อยู่ในเซต *in* และไม่เป็นเซตว่าง ผลลัพธ์คือ เซตว่าง ซึ่งจะเป็นเซต ย่อยของเซต **in** ที่ได้หลังจากมีการเปลี่ยนแปลง ทำให้ภาคแสดงนี้เป็นจริงเสมอ ดังนั้นตัวดำเนินการมิวเทชันทั้งสองไม่สามารถใช้ร่วมกันได้ เป็นต้น □

ในงานวิจัยนี้จะหาอัลกอริทึมที่เหมาะสมในการใส่ข้อผิดพลาดเข้าไปในข้อกำหนดรูปนัยแบบสัญกรณ์เซตมากกว่าหนึ่งที่พร้อมกัน โดยไม่ให้เกิดผลลัพธ์หักล้างกัน ดังที่ปรากฏในตัวอย่างที่ 3.2

3.3 การวิเคราะห์ตัวดำเนินการมิวเทชัน

ในงานวิจัยนี้อาศัยตัวดำเนินการมิวเทชันการแทนที่ตัวดำเนินการ (Operator Replacement : OR) ซึ่งเกี่ยวข้องกับตัวดำเนินการมิวเทชันของข้อกำหนดรูปนัยแบบสัญกรณ์เซต ส่วนตัวดำเนินการที่นำมาใช้สร้างข้อกำหนดมิวแทนท์จะนำมาจากคู่มืออ้างอิงสัญกรณ์เซต [1] และที่ปรากฏอยู่ในโปรแกรม Z/EVES [11] โดยแสดงให้เห็นได้ดังตารางที่ 3.1 คอลัมน์ข้อกำหนดต้นฉบับจะแสดงตัวอย่างตัวดำเนินการในข้อกำหนดรูปนัยแบบสัญกรณ์เซต คอลัมน์ข้อกำหนดมิวแทนท์จะแสดงการเปลี่ยนตัวดำเนินการที่สามารถเกิดขึ้นได้จากข้อกำหนดต้นฉบับ และคอลัมน์เงื่อนไขการตรวจสอบคือ เงื่อนไขที่สามารถนำมาตรวจสอบความผิดพลาดจากการเปลี่ยนแปลงตัวดำเนินการในข้อกำหนดมิวแทนท์ ซึ่งในบางครั้งการเปลี่ยนแปลงตัวดำเนินการอาจส่งผลให้ไม่สามารถหาเงื่อนไขมาตรวจสอบความผิดพลาดนั้นได้ นั่นคือ ค่าที่เป็นไปได้ของตัวแปรที่เปลี่ยนแปลงไปครอบคลุมค่าที่เป็นไปได้ของตัวแปรต้นฉบับซึ่งจัดอยู่ในรูปแบบที่ 3 นั่นเอง

การพิจารณาหาเงื่อนไขที่สามารถตรวจสอบความผิดพลาดที่เกิดจากการเปลี่ยนตัวดำเนินการ (เสมือนเป็นข้อผิดพลาดจำลอง) ให้กับภาคแสดงในข้อกำหนดต้นฉบับเป็นข้อกำหนดมิวแทนท์ จะใช้การปฏิเสธเงื่อนไขที่ทำให้ผลลัพธ์ของภาคแสดงที่ใส่ข้อผิดพลาดแตกต่างจากผลลัพธ์ของภาคแสดงในข้อกำหนดต้นฉบับ โดยที่เงื่อนไขดังกล่าวยังคงสอดคล้องกับภาคแสดงในข้อกำหนดต้นฉบับ

การพิจารณาตัวดำเนินการมิวเทชันด้วยการแทนที่ 36 ดำเนินการมี ดังต่อไปนี้

ภาคแสดงที่ 1 $S \cup T$ จะสามารถเปลี่ยนภาคแสดงนี้ได้เป็น $S \cap T$ $S \setminus T$

1.1 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $S \cup T$ เหมือนกับภาคแสดง $S \cap T$ คือ $S = T$ ดังนั้นเงื่อนไขการตรวจสอบ คือ $S \neq T$

1.2 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $S \cup T$ เหมือนกับภาคแสดง $S \setminus T$ คือ $T = \varnothing$ ดังนั้นเงื่อนไขการตรวจสอบคือ $T \neq \varnothing$

ภาคแสดงที่ 2 $S \setminus T$ จะสามารถเปลี่ยนภาคแสดงนี้ได้เป็น $S \cap T$ $S \cup T$

2.1 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $S \setminus T$ เหมือนกับภาคแสดง $S \cap T$ คือ $S = \varnothing$ ดังนั้นเงื่อนไขการตรวจสอบ คือ $S \neq \varnothing$

2.2 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $S \setminus T$ เหมือนกับภาคแสดง $S \cup T$ คือ $T = \varnothing$ ดังนั้นเงื่อนไขการตรวจสอบคือ $T \neq \varnothing$

ภาคแสดงที่ 3 $S \cap T$ จะสามารถเปลี่ยนภาคแสดงนี้ได้เป็น $S \setminus T$ $S \cup T$

3.1 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $S \cap T$ เหมือนกับภาคแสดง $S \setminus T$ คือ $S = \varnothing$ ดังนั้นเงื่อนไขการตรวจสอบ คือ $S \neq \varnothing$

3.2 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $S \cap T$ เหมือนกับภาคแสดง $S \cup T$ คือ $S = T$ ดังนั้นเงื่อนไขการตรวจสอบคือ $S \neq T$

ภาคแสดงที่ 4 $S \subseteq T$ จะสามารถเปลี่ยนภาคแสดงนี้ได้เป็น $S = T$ $S \neq T$ $S \subset T$

4.1 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $S \subseteq T$ เหมือนกับภาคแสดง $S = T$ คือ $S = T$ เมื่อปฏิเสธเงื่อนไขนี้จะได้ $S \neq T$ แต่เงื่อนไขต้องสอดคล้องกับภาคแสดง $S \subseteq T$ ดังนั้นเงื่อนไขการตรวจสอบ คือ $S \subset T$

4.2 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $S \subseteq T$ เหมือนกับภาคแสดง $S \neq T$ คือ $S \subset T$ เมื่อปฏิเสธเงื่อนไขนี้จะได้ $S \not\subset T$ แต่เงื่อนไขต้องสอดคล้องกับภาคแสดง $S \subseteq T$ ดังนั้นเงื่อนไขการตรวจสอบ คือ $S = T$

4.3 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $S \subseteq T$ เหมือนกับภาคแสดง $S \subset T$ คือ $S \subset T$ เมื่อปฏิเสธเงื่อนไขนี้จะได้ $S \not\subset T$ แต่เงื่อนไขต้องสอดคล้องกับภาคแสดง $S \subseteq T$ ดังนั้นเงื่อนไขการตรวจสอบ คือ $S = T$

ภาคแสดงที่ 5 $S \neq T$ จะสามารถเปลี่ยนภาคแสดงนี้ได้เป็น $S = T$ $S \subseteq T$ $S \subset T$

5.1 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $S \neq T$ เหมือนกับภาคแสดง $S = T$ ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบ

5.2 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $S \neq T$ เหมือนกับภาคแสดง $S \subseteq T$ คือ $S \subset T$ เมื่อปฏิเสธเงื่อนไขนี้จะได้ $S \not\subset T$ แต่เงื่อนไขต้องสอดคล้องกับภาคแสดง $S \neq T$ ดังนั้นเงื่อนไขการตรวจสอบ คือ $S \not\subseteq T$

5.3 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $S \neq T$ เหมือนกับภาคแสดง $S \subset T$ คือ $S \subset T$ ดังนั้นเงื่อนไขที่ใช้ในการตรวจสอบ คือ $S \not\subset T$

ภาคแสดงที่ 6 $S \subset T$ จะสามารถเปลี่ยนภาคแสดงนี้ได้เป็น $S \neq T$ $S = T$ $S \subseteq T$

6.1 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $S \subset T$ เหมือนกับภาคแสดง $S \neq T$ จะเป็นจริงเสมอ ดังนั้นไม่สามารถหาเงื่อนไขการตรวจสอบได้

6.2 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $S \subset T$ เหมือนกับภาคแสดง $S = T$ ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

6.3 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $S \subset T$ เหมือนกับภาคแสดง $S \subseteq T$ คือ $S \subset T$ เมื่อปฏิเสธเงื่อนไขนี้จะได้ $S \not\subset T$ แต่เงื่อนไขต้องสอดคล้องกับภาคแสดง $S \subset T$ ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

ภาคแสดงที่ 7 $S = T$ จะสามารถเปลี่ยนภาคแสดงนี้ได้เป็น $S \neq T$ $S \subset T$ $S \subseteq T$

7.1 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $S = T$ เหมือนกับภาคแสดง $S \neq T$ ดังนั้นไม่สามารถหาเงื่อนไขการตรวจสอบได้

7.2 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $S = T$ เหมือนกับภาคแสดง $S \subset T$ ดังนั้นจึงไม่สามารหาเงื่อนไขการตรวจสอบได้

7.3 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $S = T$ เหมือนกับภาคแสดง $S \subseteq T$ คือ $S = T$ เมื่อปฏิเสธเงื่อนไขนี่จะได้ $S \neq T$ แต่เงื่อนไขต้องสอดคล้องกับภาคแสดง $S = T$ ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

ภาคแสดงที่ 8 R^+ จะสามารถเปลี่ยนภาคแสดงนี้ได้เป็น R R^- R^k R^*

8.1 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง R^+ เหมือนกับภาคแสดง R คือ $\text{dom } R \cap \text{ran } R = \{\}$ ดังนั้นเงื่อนไขการตรวจสอบ คือ $\text{dom } R \cap \text{ran } R \neq \{\}$

8.2 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง R^+ เหมือนกับภาคแสดง R^- คือ $(\text{dom } R \cup \text{ran } R) \setminus (\text{dom } R \cap \text{ran } R) = \{\}$ ดังนั้นเงื่อนไขการตรวจสอบ คือ $(\text{dom } R \cup \text{ran } R) \setminus (\text{dom } R \cap \text{ran } R) \neq \{\}$

8.3 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง R^+ เหมือนกับภาคแสดง R^k คือ $R^k = R^+$ ดังนั้นเงื่อนไขการตรวจสอบ คือ $R^k \neq R^+$

8.4 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง R^+ เหมือนกับภาคแสดง R^* ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

ภาคแสดงที่ 9 R จะสามารถเปลี่ยนภาคแสดงนี้ได้เป็น R^{-} R^{+} R^{*} R^{k}

9.1 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง R เหมือนกับภาคแสดง R^{-} คือ $R = R^{-}$ ดังนั้นเงื่อนไขการตรวจสอบ คือ $R \neq R^{-}$

9.2 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง R เหมือนกับภาคแสดง R^{+} ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

9.3 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง R เหมือนกับภาคแสดง R^{*} ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

9.4 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง R เหมือนกับภาคแสดง R^{k} ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

ภาคแสดงที่ 10 $a \leqslant b$ จะสามารถเปลี่ยนภาคแสดงนี้ได้เป็น $a<b$ $a>b$ $a \geqslant b$ $a=b$ $a \neq b$

10.1 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a \leqslant b$ เหมือนกับภาคแสดง $a<b$ คือ $a<b$ เมื่อปฏิเสธเงื่อนไขนี้จะได้ $a \geqslant b$ แต่เงื่อนไขจะต้องสอดคล้องกับภาคแสดง $a \leqslant b$ ดังนั้นเงื่อนไขการตรวจสอบ คือ $a=b$

10.2 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a \leqslant b$ เหมือนกับภาคแสดง $a>b$ ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

10.3 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a \leqslant b$ เหมือนกับภาคแสดง $a \geqslant b$ คือ $a=b$ เมื่อปฏิเสธเงื่อนไขนี้จะได้ $a \neq b$ แต่เงื่อนไขจะต้องสอดคล้องกับภาคแสดง $a \leqslant b$ ดังนั้นเงื่อนไขการตรวจสอบ คือ $a<b$

10.4 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a \leqslant b$ เหมือนกับภาคแสดง $a=b$ คือ $a=b$ ดังนั้นเงื่อนไขการตรวจสอบ คือ $a \neq b$ แต่เงื่อนไขจะต้องสอดคล้องกับภาคแสดง $a \leqslant b$ ดังนั้นเงื่อนไขการตรวจสอบ คือ $a<b$

10.5 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a \leqslant b$ เหมือนกับภาคแสดง $a \neq b$ คือ $a<b$ เมื่อปฏิเสธเงื่อนไขนี้จะได้ $a \geqslant b$ แต่เงื่อนไขจะต้องสอดคล้องกับภาคแสดง $a \leqslant b$ ดังนั้นเงื่อนไขการตรวจสอบ คือ $a=b$

ภาคแสดงที่ 11 $a<b$ จะสามารถเปลี่ยนภาคแสดงนี้ได้เป็น $a>b$ $a \geqslant b$ $a \leqslant b$ $a=b$ $a \neq b$

11.1 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a<b$ เหมือนกับภาคแสดง $a>b$ ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

11.2 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a<b$ เหมือนกับภาคแสดง $a \geqslant b$ ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

11.3 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a<b$ เหมือนกับภาคแสดง $a \leqslant b$ คือ $a<b$ เมื่อปฏิเสธเงื่อนไขนี้จะได้ $a \geqslant b$ ซึ่งไม่สอดคล้องกับภาคแสดง $a<b$ ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

11.4 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a<b$ เหมือนกับภาคแสดง $a=b$ ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

11.5 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a<b$ เหมือนกับภาคแสดง $a\neq b$ คือ $a<b$ เมื่อปฏิเสธเงื่อนไขนี้จะได้ $a\geqslant b$ ซึ่งไม่สอดคล้องกับภาคแสดง $a<b$ ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

ภาคแสดงที่ 12 $a>b$ จะสามารถเปลี่ยนภาคแสดงนี้ได้เป็น $a<b$ $a\leqslant b$ $a\geqslant b$ $a=b$ $a\neq b$

12.1 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a>b$ เหมือนกับภาคแสดง $a<b$ ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

12.2 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a>b$ เหมือนกับภาคแสดง $a\leqslant b$ ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

12.3 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a>b$ เหมือนกับภาคแสดง $a\geqslant b$ คือ $a>b$ เมื่อปฏิเสธเงื่อนไขนี้จะได้ $a\leqslant b$ ซึ่งไม่สอดคล้องกับภาคแสดง $a>b$ ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

12.4 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a>b$ เหมือนกับภาคแสดง $a=b$ ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

12.5 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a>b$ เหมือนกับภาคแสดง $a\neq b$ คือ $a>b$ เมื่อปฏิเสธเงื่อนไขนี้จะได้ $a\leqslant b$ ซึ่งไม่สอดคล้องกับภาคแสดง $a>b$ ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

ภาคแสดงที่ 13 $a\geqslant b$ จะสามารถเปลี่ยนภาคแสดงนี้ได้เป็น $a>b$ $a<b$ $a\leqslant b$ $a=b$ $a\neq b$

13.1 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a\geqslant b$ เหมือนกับภาคแสดง $a>b$ คือ $a>b$ เมื่อปฏิเสธเงื่อนไขนี้จะได้ $a\leqslant b$ แต่เงื่อนไขต้องสอดคล้องกับภาคแสดง $a\geqslant b$ ดังนั้นเงื่อนไขการตรวจสอบ คือ $a=b$

13.2 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a\geqslant b$ เหมือนกับภาคแสดง $a<b$ ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

13.3 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a\geqslant b$ เหมือนกับภาคแสดง $a\leqslant b$ คือ $a=b$ เมื่อปฏิเสธเงื่อนไขนี้จะได้ $a\neq b$ แต่เงื่อนไขต้องสอดคล้องกับภาคแสดง $a\geqslant b$ ดังนั้นเงื่อนไขการตรวจสอบ คือ $a>b$

13.4 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a\geqslant b$ เหมือนกับภาคแสดง $a=b$ คือ $a=b$ เมื่อปฏิเสธเงื่อนไขนี้จะได้ $a\neq b$ แต่เงื่อนไขต้องสอดคล้องกับภาคแสดง $a\geqslant b$ ดังนั้นเงื่อนไขการตรวจสอบ คือ $a>b$

13.5 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a \geqslant b$ เหมือนกับภาคแสดง $a \neq b$ คือ $a > b$ เมื่อปฏิเสธเงื่อนนี้จะได้ $a \leqslant b$ แต่เงื่อนไขต้องสอดคล้องกับภาคแสดง $a \geqslant b$ ดังนั้น เงื่อนไขการตรวจสอบ คือ $a = b$

ภาคแสดงที่ 14 $a = b$ จะสามารถเปลี่ยนภาคแสดงนี้ได้เป็น $a \geqslant b$ $a > b$ $a < b$ $a \leqslant b$ $a \neq b$

14.1 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a = b$ เหมือนกับภาคแสดง $a \geqslant b$ คือ $a = b$ เมื่อปฏิเสธเงื่อนไขนี้จะได้ $a \neq b$ ซึ่งไม่สอดคล้องกับภาคแสดง $a = b$ ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

14.2 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a = b$ เหมือนกับภาคแสดง $a > b$ ดังนั้น จึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

14.3 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a = b$ เหมือนกับภาคแสดง $a < b$ ดังนั้น จึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

14.4 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a = b$ เหมือนกับภาคแสดง $a \leqslant b$ คือ $a = b$ เมื่อปฏิเสธเงื่อนไขนี้จะได้ $a \neq b$ ซึ่งไม่สอดคล้องกับภาคแสดง $a = b$ ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

14.5 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a = b$ เหมือนกับภาคแสดง $a \neq b$ ดังนั้น จึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

ภาคแสดงที่ 15 $a \neq b$ จะสามารถเปลี่ยนภาคแสดงนี้ได้เป็น $a \geqslant b$ $a > b$ $a < b$ $a \leqslant b$ $a = b$

15.1 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a \neq b$ เหมือนกับภาคแสดง $a \geqslant b$ คือ $a > b$ เมื่อปฏิเสธเงื่อนไขนี้จะได้ $a \leqslant b$ แต่เงื่อนไขต้องสอดคล้องกับภาคแสดง $a \neq b$ ดังนั้นเงื่อนไขการตรวจสอบ คือ $a < b$

15.2 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a \neq b$ เหมือนกับภาคแสดง $a > b$ คือ $a > b$ เมื่อปฏิเสธเงื่อนไขนี้จะได้ $a \leqslant b$ แต่เงื่อนไขต้องสอดคล้องกับภาคแสดง $a \neq b$ ดังนั้นเงื่อนไขการตรวจสอบ คือ $a < b$

15.3 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a \neq b$ เหมือนกับภาคแสดง $a < b$ คือ $a < b$ เมื่อปฏิเสธเงื่อนไขนี้จะได้ $a \geqslant b$ แต่เงื่อนไขต้องสอดคล้องกับภาคแสดง $a \neq b$ ดังนั้นเงื่อนไขการตรวจสอบ คือ $a > b$

15.4 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a \neq b$ เหมือนกับภาคแสดง $a \leqslant b$ คือ $a < b$ เมื่อปฏิเสธเงื่อนไขนี้จะได้ $a \geqslant b$ แต่เงื่อนไขต้องสอดคล้องกับภาคแสดง $a \neq b$ ดังนั้นเงื่อนไขการตรวจสอบ คือ $a > b$

15.5 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $a \neq b$ เหมือนกับภาคแสดง $a = b$ ดังนั้น ไม่สามารถหาเงื่อนไขการตรวจสอบ

ภาคแสดงที่ 16 x+y จะสามารถเปลี่ยนภาคแสดงนี้ได้เป็น x-y x*y x div y x mod y

16.1 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง x+y เหมือนกับภาคแสดง x-y คือ y=0 ดังนั้นเงื่อนไขการตรวจสอบ คือ y≠0

16.2 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง x+y เหมือนกับภาคแสดง x*y คือ x=(x-1)*y ดังนั้นเงื่อนไขการตรวจสอบ คือ x≠(x-1)*y

16.3 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง x+y เหมือนกับภาคแสดง x div y คือ x=(x div y)-y ดังนั้นเงื่อนไขการตรวจสอบ คือ x≠(x div y)-y

16.3 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง x+y เหมือนกับภาคแสดง x mod y คือ -y⩽x⩽-1 ดังนั้นเงื่อนไขการตรวจสอบ คือ (x<-y ∨ x>-1)

ภาคแสดงที่ 17 x-y จะสามารถเปลี่ยนภาคแสดงนี้ได้เป็น x+y x*y x div y x mod y

17.1 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง x-y เหมือนกับภาคแสดง x+y คือ y=0 ดังนั้นเงื่อนไขการตรวจสอบ คือ y≠0

17.2 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง x-y เหมือนกับภาคแสดง x*y คือ x=(x+1)*y ดังนั้นเงื่อนไขการตรวจสอบ คือ x≠(x+1)*y

17.3 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง x-y เหมือนกับภาคแสดง x div y คือ x=(x div y)+y ดังนั้นเงื่อนไขการตรวจสอบ คือ x≠(x div y)+y

17.4 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง x-y เหมือนกับภาคแสดง x mod y คือ y⩽x⩽2y-1 ดังนั้นเงื่อนไขการตรวจสอบ คือ (x<y ∨ x>2y-1)

ภาคแสดงที่ 18 x*y จะสามารถเปลี่ยนภาคแสดงนี้ได้เป็น x+y x-y x div y x mod y

18.1 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง x*y เหมือนกับภาคแสดง x+y คือ x=(x-1)*y ดังนั้นเงื่อนไขการตรวจสอบ คือ x≠(x-1)*y

18.2 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง x*y เหมือนกับภาคแสดง x-y คือ x=(x+1)*y ดังนั้นเงื่อนไขการตรวจสอบ คือ x≠(x+1)*y

18.3 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง x*y เหมือนกับภาคแสดง x div y คือ x=0 ดังนั้นเงื่อนไขการตรวจสอบ คือ x≠0

18.4 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง x*y เหมือนกับภาคแสดง x mod y คือ x=0 ดังนั้นเงื่อนไขการตรวจสอบ คือ x≠0

ภาคแสดงที่ 19 x div y จะสามารถเปลี่ยนภาคแสดงนี้ได้เป็น x+y x-y x*y x mod y

19.1 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง x div y เหมือนกับภาคแสดง x+y คือ x=(x+y)*y ดังนั้นเงื่อนไขการตรวจสอบ คือ x≠(x+y)*y

19.2 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง x div y เหมือนกับภาคแสดง x-y คือ x=(x-y)*y ดังนั้นเงื่อนไขการตรวจสอบ คือ x≠(x-y)*y

19.3 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง x div y เหมือนกับภาคแสดง x*y คือ x=0 ดังนั้นเงื่อนไขการตรวจสอบ คือ x≠0

19.4 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง x div y เหมือนกับภาคแสดง x mod y คือ x=0 ดังนั้นเงื่อนไขการตรวจสอบ คือ x≠0

ภาคแสดงที่ 20 x mod y จะสามารถเปลี่ยนภาคแสดงนี้ได้เป็น x+y x-y x*y x div y

20.1 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง x mod y เหมือนกับภาคแสดง x+y คือ ดังนั้นเงื่อนไขการตรวจสอบ คือ (x<-y ∨ x>-1)

20.2 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง x mod y เหมือนกับภาคแสดง x-y คือ ดังนั้นเงื่อนไขการตรวจสอบ คือ (x<y ∨ x>2y-1)

20.3 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง x mod y เหมือนกับภาคแสดง x*y คือ x=0 ดังนั้นเงื่อนไขการตรวจสอบ คือ x≠0

20.4 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง x mod y เหมือนกับภาคแสดง x div y คือ x=0 ดังนั้นเงื่อนไขการตรวจสอบ คือ x≠0

ภาคแสดงที่ 21 p∧q จะสามารถเปลี่ยนภาคแสดงนี้ได้เป็น p∨q p⟹q p⟺q

21.1 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง p∧q เหมือนกับภาคแสดง p∨q ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

21.2 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง p∧q เหมือนกับภาคแสดง p⟹q ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

21.3 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง p∧q เหมือนกับภาคแสดง p⟺q ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

ภาคแสดงที่ 22 p∨q จะสามารถเปลี่ยนภาคแสดงนี้ได้เป็น p∧q p⟹q p⟺q

22.1 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง p∨q เหมือนกับภาคแสดง p∧q คือ p=q=true เมื่อปฏิเสธเงื่อนไขนี้แต่ต้องสอดคล้องกับภาคแสดง p∨q ดังนั้นเงื่อนไขการตรวจสอบ คือ (p∧~q) ∨ (~p∧q)

22.2 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง p∨q เหมือนกับภาคแสดง p⟹q คือ p=q=true หรือ p=false และ q=True เมื่อปฏิเสธเงื่อนไขนี้แต่ต้องสอดคล้องกับภาคแสดง p∨q ดังนั้นเงื่อนไขการตรวจสอบ คือ (~p∨q) ∧ (p∨q)

22.3 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง p∨q เหมือนกับภาคแสดง p⟺q ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

ภาคแสดงที่ 23 $p \Leftrightarrow q$ จะสามารถเปลี่ยนภาคแสดงนี้ได้เป็น $p \wedge q$ $p \vee q$ $p \Longrightarrow q$

23.1 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $p \Leftrightarrow q$ เหมือนกับภาคแสดง $p \wedge q$ คือ $p=q=true$ เมื่อปฏิเสธเงื่อนไขนี้แต่ต้องสอดคล้องกับภาคแสดง $p \Leftrightarrow q$ ดังนั้น เงื่อนไขการตรวจสอบ คือ $\sim p \wedge \sim q$

23.2 เงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $p \Leftrightarrow q$ เหมือนกับภาคแสดง $p \vee q$ คือ $p=q=true$ เมื่อปฏิเสธเงื่อนไขนี้แต่ต้องสอดคล้องกับภาคแสดง $p \Leftrightarrow q$ ดังนั้น เงื่อนไขการตรวจสอบ คือ $\sim p \wedge \sim q$

23.3 ไม่มีเงื่อนไขที่ทำให้ผลลัพธ์จากภาคแสดง $p \Leftrightarrow q$ เหมือนกับภาคแสดง $p \Longrightarrow q$ ดังนั้นจึงไม่สามารถหาเงื่อนไขการตรวจสอบได้

ตัวดำเนินการมิวเทชันทั้งหมดสามารถสรุปได้ดังตารางที่ 3.1 ต่อไปนี้

ตารางที่ 3.1 แสดงตัวอย่างตัวดำเนินการที่สามารถนำมาสร้างข้อกำหนดมิวแทนท์

ลักษณะของภาคแสดง	ข้อกำหนดต้นฉบับ	ข้อกำหนดมิวแทนท์	เงื่อนไขการตรวจสอบ
1	$S \cup T$	$S \cap T$	$S \neq T$
2	$S \cup T$	$S \setminus T$	$T \neq \varnothing$
3	$S \setminus T$	$S \cap T$	$S \neq \varnothing$
4	$S \setminus T$	$S \cup T$	$T \neq \varnothing$
5	$S \cap T$	$S \setminus T$	$S \neq \varnothing$
6	$S \cap T$	$S \cup T$	$S \neq T$
7	$S \subseteq T$	$S = T$	$S \subset T$
8	$S \subseteq T$	$S \neq T$	$S = T$
9	$S \subseteq T$	$S \subset T$	$S = T$
10	$S \neq T$	$S = T$	ไม่สามารถหาได้
11	$S \neq T$	$S \subseteq T$	$S \not\subseteq T$
12	$S \neq T$	$S \subset T$	$S \not\subset T$
13	$S \subset T$	$S \neq T$	ไม่สามารถหาได้
14	$S \subset T$	$S = T$	ไม่สามารถหาได้
15	$S \subset T$	$S \subseteq T$	ไม่สามารถหาได้
16	$S = T$	$S \neq T$	ไม่สามารถหาได้
17	$S = T$	$S \subset T$	ไม่สามารถหาได้

ตารางที่ 3.1 แสดงตัวอย่างตัวดำเนินการที่สามารถนำมาสร้างข้อกำหนดมิวแทนท์ (ต่อ)

ลักษณะของภาคแสดง	ข้อกำหนดต้นฉบับ	ข้อกำหนดมิวแทนท์	เงื่อนไขการตรวจสอบ
18	$S=T$	$S\subseteq T$	ไม่สามารถหาได้
19	R^{+}	R	$\text{dom } R \cap \text{ran } R \neq \varnothing$
20	R^{+}	$R^{\sim}$	$(\text{dom } R \cup \text{ran } R) \setminus (\text{dom } R \cap \text{ran } R) \neq \varnothing$
21	R^{+}	R^{k}	$R^{k} \neq R^{+}$
22	R^{+}	R^{*}	ไม่สามารถหาได้
23	R	$R^{\sim}$	$R \neq R^{\sim}$
24	R	R^{+}	ไม่สามารถหาได้
25	R	R^{*}	ไม่สามารถหาได้
26	R	R^{k}	ไม่สามารถหาได้
27	$a\leqslant b$	$a<b$	$a=b$
28	$a\leqslant b$	$a>b$	ไม่สามารถหาได้
29	$a\leqslant b$	$a\geqslant b$	$a<b$
30	$a\leqslant b$	$a=b$	$a<b$
31	$a\leqslant b$	$a\neq b$	$a=b$
32	$a<b$	$a>b$	ไม่สามารถหาได้
33	$a<b$	$a\geqslant b$	ไม่สามารถหาได้
34	$a<b$	$a\leqslant b$	ไม่สามารถหาได้
35	$a<b$	$a=b$	ไม่สามารถหาได้
36	$a<b$	$a\neq b$	ไม่สามารถหาได้
37	$a>b$	$a<b$	ไม่สามารถหาได้
38	$a>b$	$a\leqslant b$	ไม่สามารถหาได้
39	$a>b$	$a\geqslant b$	ไม่สามารถหาได้
40	$a>b$	$a=b$	ไม่สามารถหาได้
41	$a>b$	$a\neq b$	ไม่สามารถหาได้
42	$a\geqslant b$	$a>b$	$a=b$
43	$a\geqslant b$	$a<b$	ไม่สามารถหาได้
44	$a\geqslant b$	$a\leqslant b$	$a>b$
45	$a\geqslant b$	$a=b$	$a>b$

ตารางที่ 3.1 แสดงตัวอย่างตัวดำเนินการที่สามารถนำมาสร้างข้อกำหนดมิวแทนท์ (ต่อ)

ลักษณะของภาคแสดง	ข้อกำหนดต้นฉบับ	ข้อกำหนดมิวแทนท์	เงื่อนไขการตรวจสอบ
46	a⩾b	a≠b	a=b
47	a=b	a⩾b	ไม่สามารถหาได้
48	a=b	a>b	ไม่สามารถหาได้
49	a=b	a<b	ไม่สามารถหาได้
50	a=b	a⩽b	ไม่สามารถหาได้
51	a=b	a≠b	ไม่สามารถหาได้
52	a≠b	a⩾b	a<b
53	a≠b	a>b	a<b
54	a≠b	a<b	a>b
55	a≠b	a⩽b	a>b
56	a≠b	a=b	ไม่สามารถหาได้
57	x+y	x-y	y≠0
58	x+y	x*y	x≠(x-1)*y
59	x+y	x div y	x≠(x div y)-y
60	x+y	x mod y	(x<-y ∨ x>-1)
61	x-y	x+y	y≠0
62	x-y	x*y	x≠(x-1)*y
63	x-y	x div y	x≠(x div y)+y
64	x-y	x mod y	(x<y ∨ x>2y-1)
65	x*y	x+y	x≠(x-1)*y
66	x*y	x-y	x≠(x+1)*y
67	x*y	x div y	x≠0
68	x*y	x mod y	x≠0
69	x div y	x+y	x≠(x+y)*y
70	x div y	x-y	x≠(x-y)*y
71	x div y	x*y	x≠0
72	x div y	x mod y	x≠0
73	x mod y	x+y	(x<-y ∨ x>-1)

ตารางที่ 3.1 แสดงตัวอย่างตัวดำเนินการที่สามารถนำมาสร้างข้อกำหนดมิวแทนท์ (ต่อ)

ลักษณะของภาคแสดง	ข้อกำหนดต้นฉบับ	ข้อกำหนดมิวแทนท์	เงื่อนไขการตรวจสอบ
74	x mod y	x-y	$(x<y \vee x>2y-1)$
75	x mod y	x*y	$x\neq 0$
76	x mod y	x div y	$x\neq 0$
77	$p\wedge q$	$p\vee q$	ไม่สามารถหาได้
78	$p\wedge q$	$p\Longrightarrow q$	ไม่สามารถหาได้
79	$p\wedge q$	$p\Leftrightarrow q$	ไม่สามารถหาได้
80	$p\vee q$	$p\wedge q$	$(p\wedge\sim q) \vee (\sim p\wedge q)$
81	$p\vee q$	$p\Longrightarrow q$	$(\sim p\vee q) \wedge (p\vee q)$
82	$p\vee q$	$p\Leftrightarrow q$	ไม่สามารถหาได้
83	$p\Leftrightarrow q$	$p\wedge q$	$\sim p\wedge\sim q$
84	$p\Leftrightarrow q$	$p\vee q$	$\sim p\wedge\sim q$
85	$p\Leftrightarrow q$	$p\Longrightarrow q$	ไม่สามารถหาได้

3.4 การวิเคราะห์ความสัมพันธ์และอัลกอริทึมการแบ่งกลุ่มตัวแปร

ในการใส่ข้อผิดพลาดเข้าไปในข้อกำหนดรูปนัยแบบสัญกรณ์เซตมากกว่าหนึ่งที่พร้อมกันโดยไม่ให้เกิดผลลัพธ์หักล้างกันนั้น แสดงว่า ข้อผิดพลาดที่ใส่ต้องเป็นอิสระจากกัน นั่นคือ เมื่อใส่ข้อผิดพลาดเข้าไปในภาคแสดงใดแล้วผลกระทบที่เกิดขึ้นกับตัวแปรจะต้องไม่ส่งผลถึงกัน ดังนั้นต้องทำการแบ่งข้อกำหนดรูปนัยออกเป็นส่วนๆ ตามการขึ้นต่อกันของตัวแปรทั้งหมดและแยกพิจารณาเป็นกลุ่ม ในการพิจารณาผลกระทบจะพิจารณาจากค่าของตัวแปร (v) ที่ปรากฏอยู่ในภาคแสดง p_i และ p_j โดยที่ $i\leq j$ และมีเส้นทางการทำงานจากภาคแสดง p_i ไปยังภาคแสดง p_j โดยที่ค่าของตัวแปร (v) ไม่ถูกเปลี่ยนแปลงก่อนถึงภาคแสดง p_j

ตัวอย่างการพิจารณาผลกระทบที่เกิดขึ้นกับตัวแปร แสดงได้จากตัวอย่างที่ 3.3

ตัวอย่างที่ 3.3 *จากเค้าร่าง CheckIn ในตัวอย่างที่ 3.1*

จากภาคแสดงในบรรทัดที่ 020 และบรรทัดที่ 021

020	$name? \in out$
021	$in' = in \cup \{name?\}$

เมื่อพิจารณาตัวแปร *name?* ที่ปรากฏอยู่ในภาคแสดงในบรรทัดที่ 020 และบรรทัดที่ 021 หากมีการเปลี่ยนแปลงค่าของแปร *name?* จากภาคแสดงในบรรทัดที่ 020 จะส่งผลกระทบต่อเซต {*name?*} ที่อยู่ในภาคแสดงในบรรทัดที่ 021 และอาจทำให้ค่าของตัวแปร **in** เปลี่ยนแปลงไปด้วย

□

เมื่อนำหลักการพิจารณาผลกระทบที่เกิดขึ้นกับตัวแปรมาใช้ในการแบ่งกลุ่มของตัวแปรจะได้อัลกอริทึมการแบ่งกลุ่ม ซึ่งแบ่งเป็น 2 ขั้นตอน ดังนี้

ขั้นตอนที่ 1 สร้างตารางแสดงผลกระทบที่เกิดขึ้นกับตัวแปร โดยมีขั้นตอนดังต่อไปนี้

ขั้นตอนที่ 1.1 พิจารณาตัวแปรในส่วนประกาศตัวแปรของแต่ละเค้าร่าง แล้วจัดแบ่งเป็น 3 กลุ่ม คือ กลุ่มตัวแปรสถานะ กลุ่มตัวแปรรับเข้า กลุ่มตัวแปรส่งออก

ขั้นตอนที่ 1.2 นำตัวแปรจากกลุ่มตัวแปรสถานะทั้งหมดที่ได้จากขั้นตอนที่ 1.1 มาเขียนลงตารางในแนวสดมภ์และแนวแถว ทั้งนี้เนื่องจากตัวแปรสถานะแต่ละตัวอาจจะส่งผลกระทบถึงกันได้เมื่อปรากฏอยู่ในเค้าร่างดำเนินการที่ต่างกัน

ขั้นตอนที่ 1.3 เพิ่มตัวแปรลงในตาราง โดยเพิ่มตัวแปรนำเข้าในแนวแถวและเพิ่มตัวแปรส่งออกในแนวสดมภ์ (ถัดจากตัวแปรสถานะ) ทั้งนี้เนื่องจากตารางแสดงผลกระทบที่เกิดขึ้นกับตัวแปรจะใช้พิจารณาตัวแปรในแนวแถวที่มีผลกระทบต่อตัวแปรในแนวสดมภ์

ขั้นตอนที่ 1.4 พิจารณาผลกระทบระหว่างตัวแปรในแต่ละภาคแสดงในส่วนภาคแสดงของแต่ละเค้าร่าง ยกเว้นเค้าร่างเริ่มต้น โดยพิจารณาจากผลลัพธ์หลังดำเนินการตามตัวดำเนินการที่ปรากฏในภาคแสดง ดังนั้นจะสามารถแบ่งตัวดำเนินการตามผลลัพธ์ของภาคแสดงได้ 2 ลักษณะ คือ

ลักษณะที่ 1 ตัวดำเนินการที่ส่งผลให้เกิดการเปลี่ยนแปลงหรือกำหนดค่าให้ตัวแปร เช่น ตัวดำเนินการกำหนดค่า ดังนั้นตัวแปรที่เกิดการเปลี่ยนแปลงจะเป็นตัวถูกกระทบเมื่อดำเนินการตามตัวดำเนินการนั้น

ลักษณะที่ 2 ตัวดำเนินการที่ไม่ส่งผลให้เกิดค่าที่เป็นไปได้ใหม่ นั่นคือ ให้ผลลัพธ์เป็นค่าความจริง เช่น ตัวดำเนินการเปรียบเทียบ ซึ่งเมื่อมีการเปลี่ยนแปลงค่าที่เป็นไปได้ของตัวแปรใดตัวแปรหนึ่งระหว่างตัวดำเนินการแล้ว จะไม่มีผลต่อค่าที่เป็นไปได้ของตัวแปรอีกตัวแปรหนึ่งตามไปด้วย

จากหลักการพิจารณาผลกระทบระหว่างตัวแปรข้างต้น ให้พิจารณาทีละตัวดำเนินการ หากดำเนินการตามตัวดำเนินการนั้นแล้วมีผลทำให้เกิดการเปลี่ยนแปลงหรือกำหนดค่าให้กับตัวแปร

ใดตัวแปรหนึ่งจะถือว่า ตัวแปรนั้นเป็นตัวถูกกระทบนั่นเอง ให้ใส่เครื่องหมาย ✓ ลงในตารางที่ได้จากขั้นตอนที่ 1.3 ตรงตำแหน่งช่องที่อยู่ในแนวสดมภ์เดียวกับตัวแปรที่ถูกกระทบและอยู่ในแนวแถวเดียวกับตัวแปรที่ส่งผลกระทบ

ผลลัพธ์ที่ต้องการ คือ ตารางแสดงผลกระทบที่เกิดขึ้นกับตัวแปร

ตัวอย่างแสดงการสร้างตารางผลกระทบที่เกิดขึ้นกับตัวแปร จะแสดงได้ดังตัวอย่างที่ 3.4

ตัวอย่างที่ 3.4 *จากข้อกำหนดรูปนัยแบบสัญกรณ์เซตต่อไปนี้*

```
001  [Name, Address, PhoneNumber]
002  ANSWER ::= ok | error
003  ┌──AddressPad──────────────────────────────
004  │ Person: ℙ Name
005  │ address: Name ⟶ Address
006  │ errorAddress: Address
007  ├──────────────────────────────────────────
008  │ Person = dom address
009  │ errorAddress ∉ ran address
010  └──────────────────────────────────────────
011  ┌──PhonePad────────────────────────────────
012  │ PhoneOwner: ℙ Name
013  │ phone: Name ⟶ PhoneNumber
014  │ errorNumber: PhoneNumber
015  ├──────────────────────────────────────────
016  │ PhoneOwner = dom phone
017  │ errorNumber ∉ ran phone
018  └──────────────────────────────────────────
019  ┌──Initialize──────────────────────────────
020  │ AddressPad'
021  │ answer!: ANSWER
022  ├──────────────────────────────────────────
023  │ Person' = ∅
```

024 $answer! = ok$

025

026 *AddToAddressPad*

027 $\Delta AddressPad$

028 $who?: Name$

029 $whoseAddress?: Address$

030 $answer!: ANSWER$

031

032 $Person' = Person \cup \{who?\}$

033 $address'$

034 $= \textbf{if } who? \notin Person \textbf{ then } address \cup \{(who? \mapsto whoseAddress?)\}$

035 **else** *address*

036 $answer! = \textbf{if } who? \notin Person \textbf{ then } ok \textbf{ else } error$

037

038 *AddToPhonePad*

039 $\Xi AddressPad$

040 $\Delta PhonePad$

041 $who?: Name$

042 $whosePhoneNumber?: PhoneNumber$

043 $answer!: ANSWER$

044

045 $PhoneOwner' = \textbf{if } who? \in Person \textbf{ then } PhoneOwner \cup \{who?\}$

046 **else** *PhoneOwner*

047 $phone' = \textbf{if } who? \in Person \setminus PhoneOwner$

048 $\textbf{then } phone \cup \{(who? \mapsto whosePhoneNumber?)\}$

049 **else** *phone*

050 $answer! = \textbf{if } who? \in Person \setminus PhoneOwner \textbf{ then } ok \textbf{ else } error$

051

052 *ErasePerson*

053 $\Delta AddressPad$

054 $\Delta PhonePad$

055 *who?: Name*

056 *answer!: ANSWER*

057 ———————————————

058 *Person'* = **if** *who?* ∈ *Person* **then** *Person* \ {*who?*} **else** *Person*

059 *PhoneOwner'* = **if** *who?* ∈ *PhoneOwner* **then** *PhoneOwner* \ {*who?*}

060 **else** *PhoneOwner*

061 *answer!* = **if** *who?* ∈ *Person* **then** *ok* **else** *error*

062 ———————————————

เมื่อพิจารณาข้อกำหนดข้างต้นเพื่อสร้างตารางผลกระทบที่เกิดขึ้นกับตัวแปรตามขั้นตอนที่ 1 จะได้กลุ่มตัวแปรสถานะประกอบด้วยตัวแปร *Person address errorAddress PhoneOwner phone errorNumber* กลุ่มตัวแปรรับเข้าประกอบด้วย *who? whoseAddress? whosePhoneNumber?* และกลุ่มตัวแปรส่งออกประกอบด้วย *answer!* และจะได้ตารางแสดงผลกระทบที่เกิดขึ้นกับตัวแปรจากข้อกำหนดดังกล่าวดังตารางที่ 3.2 โดยที่ตัวแปรสถานะจะแสดงเป็นตัวอักษรตัวเข้ม ตัวแปรนำเข้าและตัวแปรส่งออกจะแสดงเป็นตัวอักษรปกติ

ตารางที่ 3.2 แสดงผลกระทบที่เกิดขึ้นกับตัวแปร

	Person	**address**	**errorAddress**	**PhoneOwner**	**Phone**	**errorNumber**	answer!
Person	✓	✓		✓	✓		✓
address		✓					
errorAddress							
PhoneOwner				✓	✓		✓
Phone					✓		
errorNumber							
who?	✓	✓		✓	✓		✓
whoseAddress?		✓					
whosePhoneNumber?					✓		

□

ขั้นตอนที่ 2 การจัดกลุ่มของตัวแปรที่ไม่มีผลกระทบถึงกัน มีขั้นตอนดังนี้

ขั้นตอนที่ 2.1 เริ่มการพิจารณาจากกลุ่มของตัวแปรที่อยู่ในแนวสดมภ์ทั้งหมดและกลุ่มของตัวแปรในแนวแถวเฉพาะกลุ่มตัวแปรนำเข้า โดยเริ่มจากตัวแปรตัวแรกจากกลุ่มตัวแปรสถานะให้นำมาจัดให้อยู่ในกลุ่มที่ 1

ขั้นตอนที่ 2.2 นำตัวแปรสถานะตัวถัดไปมาพิจารณาถึงผลกระทบที่มีกับตัวแปรในกลุ่ม โดยอาศัยตารางที่ได้จากขั้นตอนที่ 1 วิธีการคือ เริ่มจากการมองตัวแปรที่กำลังพิจารณาในแนวแถวเพื่อหาช่องที่ตรงกับตัวแปรในกลุ่มในแนวสดมภ์ และดูว่ามีเครื่องหมาย ✓ หรือไม่

หากมีเครื่องหมาย ✓ แสดงว่า ตัวแปรที่กำลังพิจารณาอยู่ส่งผลกระทบต่อตัวแปรในกลุ่ม ให้จัดตัวแปรที่กำลังพิจารณานี้อยู่ในกลุ่มใหม่

แต่หากไม่มีเครื่องหมาย ✓ แสดงว่า ตัวแปรที่กำลังพิจารณาอยู่ไม่ส่งผลกระทบต่อตัวแปรในกลุ่ม ให้จัดตัวแปรที่กำลังพิจารณานี้อยู่ลงในกลุ่มได้

ขั้นตอนที่ 2.3 นำตัวแปรถัดไปมาพิจารณาเหมือนขั้นตอนที่ 2.2 จนครบทุกตัวแปร และหากเป็นตัวแปรนำเข้าให้พิจารณาว่า ตัวแปรนำเข้านั้นส่งผลกระทบถึงตัวแปรใดในกลุ่ม โดยการพิจารณานั้นให้พิจารณาผลกระทบที่มีต่อตัวแปรแต่ละตัวในแต่ละกลุ่มที่ถูกสร้างขึ้นก่อนหน้าจนทำให้ตัวแปรที่กำลังพิจารณาสามารถถูกจัดเข้ากลุ่มใดกลุ่มหนึ่งได้

ผลลัพธ์ที่ต้องการ คือ ตารางแสดงกลุ่มของตัวแปรที่เป็นอิสระจากกัน นั่นคือ ค่าของตัวแปรที่เป็นไปได้ในแต่ละกลุ่มไม่มีผลกระทบถึงกัน ดังแสดงในตัวอย่างที่ 3.4

ตัวอย่างที่ 3.4 *จากตารางผลกระทบที่เกิดขึ้นกับตัวแปรในตัวอย่างที่ 3.3*

เมื่อนำตารางที่ 3.2 มาพิจารณาจัดกลุ่มของตัวแปรที่สมาชิกในกลุ่มไม่มีผลกระทบถึงกัน ตามขั้นตอนการสร้างกลุ่มของตัวแปรที่ไม่มีผลกระทบถึงกัน จะได้ผลลัพธ์แสดงดังตารางที่ 3.3

ตารางที่ 3.3 แสดงกลุ่มของตัวแปรที่ไม่มีผลกระทบถึงกัน

กลุ่มที่ 1	กลุ่มที่ 2	กลุ่มที่ 3
Person	***address***	*who?*
errorAddress	*phone*	
PhoneOwner	*answer!*	
errorNumber		
whoseAddress?		
whosePhoneNumber?		

□

3.5 อัลกอริทึมการสร้างข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาด

3.5.1 นิยามของข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาด

ข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดเกิดจากการใส่ข้อผิดพลาดในหลายตำแหน่งเพื่อให้ข้อกำหนดมิวแทนท์เดียวสามารถแทนหลายข้อกำหนดมิวแทนท์ จะมีนิยามดังนี้

นิยามที่ 3.1 *กำหนดให้ S เป็นข้อกำหนดรูปนัย และข้อกำหนดรูปนัย C จะถูกเรียกว่าเป็น* ข้อกำหนดรูปนัยมิวแทนท์แบบหลายข้อผิดพลาด (Composite mutant formal specification) *ของ S ก็ต่อเมื่อ*

สำหรับจำนวนเต็มบวก k ใดๆ และ $p_1, p_2, p_3 \ldots, p_k$ เป็นภาคแสดงของ S จะต้องสามารถหาข้อกำหนดรูปนัย $M_1, M_2, M_3, \ldots, M_k$ ที่สอดคล้องกับ

$$S \underset{p_1}{\overset{u_1}{\Longrightarrow}} M_1 \underset{p_2}{\overset{u_2}{\Longrightarrow}} M_2 \underset{p_3}{\overset{u_3}{\Longrightarrow}} M_3 \Longrightarrow \ldots \underset{p_k}{\overset{u_k}{\Longrightarrow}} M_k = C$$

โดย $S \underset{p_j}{\overset{u_j}{\Longrightarrow}} M_j$ *เป็นการใส่ข้อผิดพลาด u_j ที่ภาคแสดง p_j*

3.5.2 อัลกอริทึมในการสร้างข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาด

อัลกอริทึมที่ใช้ในการสร้างข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาด กำหนดให้ S เป็นข้อกำหนดต้นฉบับ

ขั้นตอนที่ 1 สร้างข้อผิดพลาดที่จะนำไปใส่ให้กับข้อกำหนดต้นฉบับ S โดยชนิดของข้อผิดพลาดจะถูกนิยามไว้ในตัวดำเนินการมิวเทชัน

ขั้นตอนที่ 2 ใช้อัลกอริทึมการแบ่งกลุ่มในหัวข้อที่ 3.4 เพื่อให้ได้กลุ่มของตัวแปรแต่ละกลุ่มที่ไม่มีผลกระทบถึงกัน สมมติแบ่งกลุ่มได้ n กลุ่ม ($V_1, V_2, V_3, \ldots, V_n$) จำนวนตัวแปรในแต่ละกลุ่ม $|V_i|$

ขั้นตอนที่ 3 วิเคราะห์แต่ละข้อผิดพลาดที่จะใส่ให้กับข้อกำหนดต้นฉบับว่า เมื่อใส่ข้อผิดพลาดนี้แล้ว ข้อกำหนดมิวแทนท์ที่ได้จะสามารถหาเงื่อนไขการตรวจสอบได้หรือไม่ หากไม่สามารถหาเงื่อนไขการตรวจสอบได้หรือเป็นข้อผิดพลาดที่หักล้างกันจะไม่พิจารณาข้อผิดพลาดนี้

ขั้นตอนที่ 4 ทำการแบ่งกลุ่มของภาคแสดงตามตัวแปรทั้งหมดโดยพิจารณาทีละภาคแสดง ในการพิจารณาจะดูที่ผลกระทบของภาคแสดงที่มีต่อตัวแปร ทำให้สามารถแบ่งภาคแสดงออกได้เป็นสองกรณี

กรณีที่หนึ่ง ภาคแสดงที่ให้ค่าความจริง จะถือว่าภาคแสดงนี้มีผลกระทบต่อตัวแปรที่ปรากฏในภาคแสดงนั้นตัวใดตัวหนึ่งเพียงตัวเดียว

กรณีที่สอง ภาคแสดงที่มีการเปลี่ยนแปลงหรือกำหนดค่าในตัวแปร จะถือว่าภาคแสดงนั้นมีผลกระทบต่อตัวแปรที่มีการเปลี่ยนแปลงค่าหรือถูกกำหนดค่า

ทำการจัดกลุ่มของภาคแสดงทั้งหมดตามตัวแปรที่ถูกผลกระทบของภาคแสดงนั้น ซึ่งจะเห็นว่าในแต่ละกลุ่มของตัวแปรที่ไม่มีผลกระทบต่อกัน (ซึ่งได้จากขั้นตอนที่ 2) จะประกอบด้วยภาคแสดงต่างๆ ที่แบ่งย่อยตามตัวแปรต่างๆ ในกลุ่มนั้น

กำหนดให้กลุ่มของตัวแปร $V_i = \{ v_{i,1}, v_{i,2}, \ldots, v_{i,t} \}$
กำหนดให้กลุ่มของภาคแสดงสำหรับตัวแปร $v_{i,j}$ แต่ละตัวแทนด้วย $P_{i,j} = \{ p_{i,j,1}, p_{i,j,2}, \ldots , p_{i,j,m} \}$
ทำการสร้างกลุ่มของข้อกำหนดมิวแทนท์ (ตามตารางที่ 3.1) สำหรับแต่ละ $P_{i,j}$ กำหนดให้เป็น $U_{i,j} = \{ u_{i,j,1}, u_{i,j,2}, \ldots, u_{i,j,s} \}$

ขั้นตอนที่ 5 สร้างข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดตามอัลกอริทึมด้านล่างนี้ โดยการเลือกข้อผิดพลาดที่ใส่ให้กับภาคแสดงที่ได้จากตัวแปรในกลุ่มเดียวกัน ทำจนกระทั่งครบทุกกลุ่มของตัวแปร

Input : V_i คือกลุ่มของตัวแปร
$P_{i,j}$ คือกลุ่มของภาคแสดงของ $v_{i,j}$
$U_{i,j}$ คือกลุ่มของข้อกำหนดมิวแทนท์
Output : Mutant = $\{C_{1,1}, C_{1,2}, \ldots, C_{n,\max\left|U_{n,|Vn|}\right|}\}$

```
begin
    for i=1 to n do
        max = max(|U_{i,1}|, |U_{i,2}|, ..., |U_{i,|Vi|}|)
        for j=1 to max do
            for k=1 to |V_i| do
                if u_{i,k,j} then
                    C_{i,j} = insert u_{i,k,j} at predicate p_{i,j} to C_{i,i}
                endif;
```

```
            endfor;
            Mutant = Mutant ∪ {C_ij}
        endfor;
    endfor;
end
```

ผลลัพธ์ที่ต้องการ คือ ชุดของข้อกำหนดมิวแทนท์ที่เกิดจากชุดของข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาด

ตัวอย่างของข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดถูกแสดงไว้ในตัวอย่างที่ 3.5

ตัวอย่างที่ 3.5 *จากเค้าร่าง CheckIn ในตัวอย่างที่ 3.1*

```
016  ┌──CheckIn──────────────────────────────
017  │  ΔLog
018  │  name?: Staff
019  ├──────────────────────────────────────
020  │  name? ∈ out
021  │  in' = in ∪ {name?}
022  │  out' = out \ {name?}
023  │  users' = users
024  └──────────────────────────────────────
```

ข้อกำหนดมิวแทนท์ M_1 ได้มาจากการเปลี่ยนภาคแสดงที่บรรทัด 021 จาก *in' = in ∪ {name?}* เป็น *in' = in ∩ {name?}* จะได้

```
016  ┌──CheckIn──────────────────────────────
017  │  ΔLog
018  │  name?: Staff
019  ├──────────────────────────────────────
020  │  name? ∈ out
021  │  in' = in ∩ {name?}          // Operator Replacement
022  │  out' = out \ {name?}
023  │  users' = users
024  └──────────────────────────────────────
```

ข้อกำหนดมิวแทนท์ M_2 ได้มาจากการเปลี่ยนภาคแสดงที่บรรทัด 022 จาก $out' = out \setminus \{name?\}$ เป็น $out' = out \cup \{name?\}$ จะได้

```
016  ____CheckIn_____________________________________
017  | ΔLog
018  | name?: Staff
019  |_____________________________________
020  | name? ∈ out
021  | in' = in ∪ {name?}
022  | out' = out ∪ {name?}          // Operator Replacement
023  | users' = users
024  |_____________________________________
```

ข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาด C ได้มาจากการเปลี่ยนภาคแสดงที่บรรทัด 021 จาก $in' = in \cup \{name?\}$ เป็น $in' = in \cap \{name?\}$ และเปลี่ยนภาคแสดงที่บรรทัด 022 จาก $out' = out \setminus \{name?\}$ เป็น $out' = out \cup \{name?\}$

```
016  ____CheckIn_____________________________________
017  | ΔLog
018  | name?: Staff
019  |_____________________________________
020  | name? ∈ out
021  | in' = in ∩ {name?}            // Operator Replacement
022  | out' = out ∪ {name?}          // Operator Replacement
023  | users' = users
024  |_____________________________________
```

จะสังเกตได้ว่า ข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาด C เกิดจากข้อผิดพลาดของทั้ง M_1 และ M_2 □

3.6 บทพิสูจน์

งานวิจัยนี้ได้เสนอแนวทางการพิสูจน์และได้ออกแบบการทดลองเพื่อยืนยันแนวความคิดในการลดจำนวนข้อกำหนดมิวแทนท์ โดยการสร้างข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาด ดังต่อไปนี้

3.6.1 ผลกระทบที่เกิดจากข้อผิดพลาดที่ใส่ให้กับข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาด

ในการสร้างข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาด โดยการใส่ข้อผิดพลาดหลายข้อผิดพลาดให้กับข้อกำหนดต้นฉบับนั้น ข้อผิดพลาดที่ใส่เข้าไปจะต้องเป็นอิสระจากกัน นั่นคือ ผลกระทบของข้อผิดพลาดต่างๆ ที่ใส่เข้าไปในข้อกำหนดต้นฉบับจำเป็นต้องกระทบต่อผลลัพธ์ของข้อกำหนดต้นฉบับที่ต่างกัน ในหัวข้อนี้จะแสดงการพิสูจน์ว่า อัลกอริทึมการสร้างข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดสามารถสร้างข้อกำหนดมิวแทนท์ที่เกิดจากการใส่ข้อผิดพลาดที่เป็นอิสระจากกัน

ทฤษฎีบทที่ 3.1 *กำหนดให้ S เป็นข้อกำหนดรูปนัย และ C เป็นข้อกำหนดรูปนัยมิวแทนท์แบบหลายข้อผิดพลาดที่ได้มาจากอัลกอริทึมการสร้างข้อกำหนดรูปนัยมิวแทนท์แบบหลายข้อผิดพลาด*

$$S \underset{p_1}{\overset{u_1}{\Longrightarrow}} M_1 \underset{p_2}{\overset{u_2}{\Longrightarrow}} M_2 \underset{p_3}{\overset{u_3}{\Longrightarrow}} M_3 \Longrightarrow \ldots \underset{p_k}{\overset{u_k}{\Longrightarrow}} M_k = C$$

โดย u_j คือ ข้อผิดพลาดที่ถูกใส่ให้กับภาคแสดง p_j และผลกระทบของข้อผิดพลาด u_j ส่งผลกระทบต่อผลลัพธ์ตัวที่ j เท่านั้น

หากมีกรณีทดสอบ t ใดๆ สามารถแยกความแตกต่างของผลลัพธ์ตัวที่ j (O_j) ของข้อกำหนดต้นฉบับเมื่อได้รับการทดสอบกับกรณีทดสอบ t (S(t)) จากข้อกำหนดมิวแทนท์เมื่อได้รับการทดสอบด้วยกรณีทดสอบ t เดียวกัน ($M_j(t)$) โดยข้อกำหนดมิวแทนท์นี้เกิดจาก

$$S \underset{p_1}{\overset{u_1}{\Longrightarrow}} M_j, 1 \le j \le k$$

แล้วข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดเมื่อได้รับการทดสอบกับกรณีทดสอบ t (C(t)) จะต้องให้ผลลัพธ์ที่แตกต่างจาก S(t) ที่ผลลัพธ์ตัวที่ j เท่านั้น

พิสูจน์ สมมุติให้กรณีทดสอบ t สามารถแยกความแตกต่างของผลลัพธ์ O_j ของ S(t) จาก $M_j(t)$ ได้ โดยการพิสูจน์แบบขัดแย้งสมมุติให้ C(t) ให้ผลลัพธ์ที่ไม่แตกต่างไปจาก S(t) ที่ผลลัพธ์ O_j จะต้องพิสูจน์ให้ได้ว่าเกิดข้อขัดแย้งขึ้น

เนื่องจากข้อกำหนดมิวแทนท์ M_j ต่างจากข้อกำหนดต้นฉบับ S ที่ภาคแสดง p_j ด้วยการใส่ข้อผิดพลาด u_j และข้อผิดพลาด u_j นี้ยังถูกใส่ให้กับข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาด C

จากสมมุติฐานที่กำหนดว่า สมมุติให้ C(t) ให้ผลลัพธ์ที่ไม่แตกต่างไปจาก S(t) ที่ผลลัพธ์ O_j ดังนั้นเราสามารถบอกได้ว่า C จะต้องมีภาคแสดงที่ได้รับการใส่ข้อผิดพลาดภาคแสดงหนึ่ง (S_o) ที่ $S_o \neq S_j$ โดยที่ผลกระทบของข้อผิดพลาดที่คำสั่ง S_o กระทบต่อ O_j เช่นเดียวกับข้อผิดพลาดที่ใส่ที่ S_j

สรุปได้ว่า S_o และ S_j จะต้องอยู่ในส่วนของข้อกำหนดที่ส่งผลกระทบต่อตัวแปรส่งออก O_j ทำให้ขัดแย้งกับอัลกอริทึมการสร้างข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดที่กล่าวไว้ว่า ให้เลือกข้อผิดพลาดหนึ่งข้อผิดพลาดมาจากแต่ละกลุ่ม

■

3.6.2 ประสิทธิผลของกรณีทดสอบ

ในหัวข้อนี้เป็นการพิสูจน์ถึงประสิทธิผลของชุดกรณีทดสอบที่ได้รับมาจากการวิเคราะห์มิวเทชันแบบหลายข้อผิดพลาดว่ามีประสิทธิผลที่ไม่ลดลง

ทฤษฎีบทที่ 3.2 *ชุดของกรณีทดสอบ T ใดๆ ที่สามารถกำจัดข้อกำหนดมิวแทนท์ในวิธีการแบบเดิมได้แล้ว จะมีความสามารถเพียงพอที่จะกำจัดข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาด*

พิสูจน์ กำหนดให้

- $F = \{u_1, u_2, u_3, \ldots, u_n\}$ เป็นชุดของข้อผิดพลาดที่ใส่ให้กับข้อกำหนดต้นฉบับ S
- C เป็นข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดใดๆ ก็ตามที่สร้างมาจากการใส่ข้อผิดพลาดที่มีอยู่ใน F หลายข้อผิดพลาดให้กับ S
- $T = \{t_1, t_2, t_3, \ldots, t_m\}$ เป็นชุดของกรณีทดสอบที่สามารถตรวจจับผลกระทบของทุกข้อผิดพลาดที่มีอยู่ใน F

เราสามารถพิสูจน์ทฤษฎีบทที่ 3.2 ได้โดยสมมุติว่า เมื่อเรานำทุกๆ t_i ใน T มาทดสอบกับ C แล้วยังมีผลลัพธ์ที่ O_j ของ $S(t_i)$ เท่ากับ O_j ของ $C(t_i)$

ถ้า $u_j \in F$ เป็นข้อผิดพลาดที่ส่งผลกระทบต่อ O_j และถูกใส่ให้กับ C จากข้อสมมุติที่ตั้งไว้คือ ทุกๆ t_i ใน T เมื่อนำมาทดสอบกับ C แล้วยังมีผลลัพธ์ที่ O_j ของ $S(t_i)$ เท่ากับ O_j ของ $C(t_i)$ และจากทฤษฎีบทที่ 3.1 ที่กล่าวว่า t_i จะต้องตรวจจับ u_j ได้ที่ O_j เท่านั้น

จะได้ว่า ทุกๆ t_i ใน T ไม่สามารถตรวจจับผลกระทบของข้อผิดพลาด u_j ซึ่งขัดแย้งกับข้อกำหนดที่ว่า T เป็นชุดทดสอบที่สามารถตรวจจับผลกระทบของข้อผิดพลาดได้ทุกข้อผิดพลาดใน F

■

3.6.3 การทดลองเพื่อวัดเปอร์เซ็นต์การลดลงของข้อกำหนดมิวแทนท์

งานวิจัยชิ้นนี้ได้เสนอการทดลองเพื่อวัดประสิทธิภาพการสร้างข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดโดยเลือกทดลองกับข้อกำหนดรูปนัยแบบสัญกรณ์เซด

ขั้นตอนการทดลองเริ่มจากการนิยามตัวดำเนินการมิวเทชัน

ขั้นตอนถัดมา คือ วิเคราะห์ข้อผิดพลาดว่า มีข้อผิดพลาดใดที่ทำให้เกิดข้อกำหนดมิวแทนท์ที่หักล้างกันและไม่สามารถหาเงื่อนไขการตรวจสอบได้ ดังนั้นจำนวนข้อผิดพลาดที่เหลืออยู่จะเป็นจำนวนของข้อกำหนดมิวแทนท์ในวิธีการวิเคราะห์มิวเทชันแบบเดิม

จากนั้นสร้างข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดโดยอาศัยอัลกอริทึมการสร้างข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาด แล้วเปรียบเทียบจำนวนข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดที่ได้กับจำนวนข้อกำหนดมิวแทนท์ในวิธีการแบบเดิม เพื่อวัดเปอร์เซ็นต์การลดลง (Percentage saved)

การคำนวณเปอร์เซ็นต์ที่ลดลงของจำนวนมิวแทนท์ สามารถคำนวณได้จากสูตรดังต่อไปนี้

$$\text{เปอร์เซ็นต์ที่ลดลง} = \frac{M - E - C}{M - E} \times 100$$

M คือ จำนวนมิวแทนท์ที่ได้รับจากวิธีการวิเคราะห์มิวเทชันแบบเดิม

E คือ จำนวนมิวแทนท์ที่หักล้างกันและที่ไม่สามารถหาเงื่อนไขการตรวจสอบได้

C คือ จำนวนมิวแทนท์ของข้อกำหนดแบบหลายข้อผิดพลาด

เปอร์เซ็นต์การลดลงของจำนวนของข้อกำหนดมิวแทนท์ที่อาศัยการสร้างข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดจะขึ้นกับจำนวนข้อผิดพลาดที่ใส่ให้กับภาคแสดงที่ไม่มีผลกระทบถึงกัน หากข้อผิดพลาดที่ใส่ให้กับข้อกำหนดที่ไม่มีผลกระทบถึงกันมีจำนวนมาก การลดลงของข้อกำหนดมิวแทนท์ในวิธีการสร้างข้อกำหนดมิวแทน์แบบหลายข้อผิดพลาดจะมีเปอร์เซ็นต์สูงขึ้น

บทที่ 4

การทดลอง

ในบทนี้ได้นำเสนอการทดลองเพื่อวัดประสิทธิภาพของการสร้างข้อกำหนดรูปนัยแบบหลายข้อผิดพลาด โดยอาศัยแนวคิดที่กล่าวไว้ก่อนหน้านี้ และในการทดลองนั้นจะใช้ข้อกำหนดรูปนัยแบบสัญกรณ์เซดเป็นตัวอย่างในการทดลอง

4.1 รายละเอียดของข้อกำหนดรูปนัยแบบสัญกรณ์เซด

ในหัวข้อนี้จะแสดงข้อมูลเกี่ยวกับจำนวนเค้าร่างดำเนินการและจำนวนตัวแปรในข้อกำหนดรูปนัยที่นำมาใช้ทดลองดังตารางที่ 4.1 ส่วนรายละเอียดของข้อกำหนดดังกล่าวจะแสดงไว้ในภาคผนวก ก.

ตารางที่ 4.1 ข้อมูลของข้อกำหนดรูปนัยที่ใช้ในการทดลอง

ข้อกำหนดรูปนัยแบบสัญกรณ์เซดของระบบ	จำนวน			
	เค้าร่างดำเนินการ	ตัวแปรสถานะ	ตัวแปรนำเข้า	ตัวแปรส่งออก
A	8	6	3	3
B	11	6	2	-
C	6	2	3	3
D	5	4	1	3
E	5	2	2	3

4.2 การทดลองเพื่อวัดเปอร์เซ็นต์การลดลงของข้อกำหนดมิวแทนท์

จากขั้นตอนการทดลองที่ได้ออกแบบไว้ เมื่อนำข้อกำหนดรูปนัยแบบสัญกรณ์เซดทั้ง 5 ข้อกำหนดมาสร้างข้อกำหนดมิวแทนท์แบบเดิมและแบบหลายข้อผิดพลาด พร้อมทั้งคำนวณหาเปอร์เซ็นต์การลดลงของข้อกำหนดมิวแทนท์ จะได้ผลลัพธ์ดังตารางที่ 4.2

ตารางที่ 4.2 เปอร์เซ็นต์การลดลงของจำนวนข้อกำหนดมิวแทนท์

ข้อกำหนดรูปนัยแบบสัญกรณ์เซดของระบบ	จำนวนข้อกำหนดมิวแทนท์			
	แบบเดิม	ไม่มีเงื่อนไขตรวจสอบ	แบบหลายข้อผิดพลาด	เปอร์เซ็นต์การลดลง
A	53	18	20	42.86

ตารางที่ 4.2 เปอร์เซ็นต์การลดลงของจำนวนข้อกำหนดมิวแทนท์ (ต่อ)

ข้อกำหนดรูปนัยแบบสัญกรณ์เซตของระบบ	จำนวนข้อกำหนดมิวแทนท์			
	แบบเดิม	ไม่มีเงื่อนไขตรวจสอบ	แบบหลายข้อผิดพลาด	เปอร์เซ็นต์การลดลง
B	368	218	80	46.67
C	72	47	17	32.00
D	68	37	18	41.94
E	67	37	27	10.00
เฉลี่ย				38.50

จากผลการทดลองที่ได้รับซึ่งแสดงให้เห็นในตารางที่ 4.2 จะเห็นได้ว่า จำนวนข้อกำหนดมิวแทนท์ลดลงได้มากที่เท่ากับ 46.67% ในข้อกำหนดรูปนัยของระบบ B และค่าเฉลี่ยของเปอร์เซ็นต์การลดลงของทั้ง 5 ข้อกำหนดเท่ากับ 38.50%

เมื่อพิจารณาเปอร์เซ็นต์ที่ลดลงของข้อกำหนดมิวแทนท์ โดยการสร้างข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดนั้น จะเห็นว่า เปอร์เซ็นต์ที่ลดลงจะขึ้นอยู่กับการใส่ข้อผิดพลาดให้กับภาคแสดงที่ค่าของตัวแปรไม่ส่งผลกระทบถึงกัน ดังนั้นหากมีภาคแสดงที่ค่าของตัวแปรไม่ส่งผลกระทบถึงกันจำนวนมาก จะทำให้เปอร์เซ็นต์การลดลงของจำนวนของข้อกำหนดมิวแทนท์ด้วยวิธีการสร้างข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดมีค่าสูงตามไปด้วย

นอกจากนี้ยังมีปัจจัยอื่นที่ส่งผลต่อเปอร์เซ็นต์การลดลงของข้อกำหนดมิวแทนท์ นั่นคือ ลักษณะของการเขียนบรรยายข้อกำหนด ซึ่งมีด้วยกัน 2 ประการ

ประการที่ 1 จำนวนตัวแปรที่ไม่ส่งผลกระทบถึงกัน เพราะหากมีจำนวนตัวแปรที่ไม่ส่งผลกระทบถึงกันจำนวนมาก จะส่งผลให้ข้อกำหนดมิวแทนท์หนึ่งสามารถเป็นใส่ข้อผิดพลาดเข้าไปในภาคแสดงที่ไม่ทำให้เกิดผลกระทบต่อค่าในตัวแปรเหล่านั้น นั่นคือข้อกำหนดมิวแทนท์หนึ่งสามารถแทนหลายข้อกำหนดมิวแทนท์เดียวได้มากขึ้นนั่นเอง

ประการที่ 2 จำนวนภาคแสดงในแต่ละเค้าร่าง เนื่องจากภาคแสดงในส่วนภาคแสดงของเค้าร่างเป็นการเขียนบรรยายพฤติกรรมของระบบ ดังนั้นอาจจะมีการเขียนบรรยายด้วยภาคแสดงจำนวนมากและซ้ำกันในแต่ละเค้าร่าง ทั้งนี้เพื่อเพิ่มความละเอียดในการบรรยายพฤติกรรมของระบบ ทำให้ได้ภาคแสดงจำนวนมาก ส่งผลให้โอกาสที่จะใส่ข้อผิดพลาดให้กับแต่ละภาคแสดงทำได้มากขึ้นจนนำไปสู่การสร้างข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดเพื่อลดข้อกำหนดมิวแทนท์ได้มากตามไปด้วย

ดังนั้นสามารถสรุปได้ว่า ในการสร้างข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาด สามารถลดจำนวนข้อกำหนดมิวแทนท์ได้ ส่งผลต่อการกำหนดคุณสมบัติของชุดทดสอบที่จะนำมาใช้ในกระบวนการทดสอบโปรแกรมเมื่อถูกพัฒนาขึ้น โดยที่ยังคงรักษาประสิทธิภาพของชุดทดสอบที่มีอยู่เดิมได้

บทที่ 5

สรุปและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

ปัญหาสำคัญของการวิเคราะห์มิวเทชันคือ จำนวนครั้งในการทดสอบแต่ละข้อกำหนดมิวแทนท์ซึ่งมีจำนวนมาก ดังนั้นเพื่อทำการปรับปรุงประสิทธิภาพของการวิเคราะห์มิวเทชัน งานวิจัยชิ้นนี้จึงได้เสนอแนวทางการปรับปรุงประสิทธิภาพการวิเคราะห์มิวเทชันโดยการลดจำนวนข้อกำหนดมิวแทนท์ โดยการสร้างข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดซึ่งสามารถเป็นตัวแทนหลายข้อกำหนดมิวแทนท์ และข้อดีที่ได้รับจากการปรับปรุงในงานวิจัยนี้คือ ยังคงรักษาประสิทธิผลของชุดทดสอบที่มีอยู่เดิมได้ตามทฤษฎีบทที่ 3.2

ในงานวิจัยนี้ได้นำเสนอแนวทางการลดกรณีทดสอบโดยการลดจำนวนของข้อกำหนดมิวแทนท์ และทฤษฎีบทพร้อมทั้งพิสูจน์แนวทางดังกล่าว เพื่อยืนยันว่าสามารถทำการลดจำนวนข้อกำหนดมิวแทนท์ได้

จากการทดลองพบว่า การลดลงของจำนวนของข้อกำหนดมิวแทนท์โดยเฉลี่ยของข้อกำหนดรูปนัยแบบสัญกรณ์เซตทั้ง 5 ข้อกำหนดคิดเป็นเปอร์เซ็นต์เท่ากับ 59.94

ปัจจัยที่ส่งผลต่อการลดลงของจำนวนของข้อกำหนดมิวแทนท์เมื่อใช้วิธีการสร้างข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดมีด้วยกัน 2 ปัจจัย คือ

1. จำนวนข้อผิดพลาดที่สามารถใส่ให้กับภาคแสดงที่ทำให้ค่าที่เป็นไปได้ของตัวแปรไม่ส่งผลกระทบถึงกัน ดังนั้นหากข้อกำหนดรูปนัยมีตัวแปรที่ไม่ส่งผลกระทบถึงกันเป็นจำนวนมาก การลดลงของจำนวนของข้อกำหนดมิวแทนท์จะมีมากเมื่อนำมาสร้างข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาด
2. ลักษณะของข้อกำหนดรูปนัยแบบสัญกรณ์เซต นั่นคือ หากในการเขียนข้อกำหนดรูปนัยแบบสัญกรณ์เซตมีการเขียนบรรยายด้วยภาคแสดงจำนวนมากและแต่ละภาคแสดงประกอบด้วยตัวดำเนินการหลากหลาย ทั้งนี้เพื่อเพิ่มความละเอียดให้กับข้อกำหนด โอกาสที่จะใส่ข้อผิดพลาดได้อย่างหลากหลายก็จะมีมากขึ้น ส่งผลให้การสร้างข้อกำหนดมิวแทนท์ซึ่งแทนหลายข้อกำหนดมิวแทนท์เป็นไปได้สูงขึ้น

5.2 ปัญหาและอุปสรรค

การวิเคราะห์มิวเทชันสำหรับข้อกำหนดรูปนัยแบบสัญกรณ์เซต คือ การใส่ข้อผิดพลาดที่อาจจะเกิดขึ้นได้ลงในข้อกำหนด โดยใช้ตัวดำเนินการมิวเทชันที่ได้นิยามไว้ และในการใส่ข้อผิดพลาดให้กับข้อกำหนดต้นฉบับนั้น จะต้องไม่ทำให้ผิดไวยกรณ์ของข้อกำหนดรูปนัย ในงานวิจัยชิ้นนี้ใช้โปรแกรม Z/EVES ช่วยในการตรวจสอบไวยกรณ์ของข้อกำหนดรูปนัยแบบสัญกรณ์เซต โดยที่ความสามารถของโปรแกรมดังกล่าวยังมีอยู่อย่างจำกัด นั่นคือ หน้าที่การทำงานของเครื่องมือยังไม่ครอบคลุมสัญกรณ์ที่มีใช้ในข้อกำหนดรูปนัยแบบสัญกรณ์เซต หรือเครื่องมือขาดความยืดหยุ่นในการนำเข้าสัญกรณ์บางอย่างที่เป็นข้อความ เช่น **if then else ran dom** เป็นต้น ผู้ใช้งานต้องเรียกใช้เครื่องมือที่มีให้เท่านั้นโดยไม่สามารถพิมพ์ได้เอง อีกทั้งในการทดสอบข้อกำหนดรูปนัยแบบสัญกรณ์เซตโดยการใส่กรณีทดสอบเพื่อดูผลลัพธ์ที่เกิดขึ้น ผู้ใช้จะต้องเขียนข้อกำหนดรูปนัยแบบสัญกรณ์เซตด้วยภาษาลาเท็กซ์ ทำให้ต้องใช้เวลาในการแปลงจากเขียนข้อกำหนดในรูปแบบเค้าร่างไปเป็นข้อกำหนดในแบบภาษาลาเท็กซ์

นอกจากนี้ในขั้นตอนการพิจารณาผลกระทบที่เกิดขึ้นกับตัวแปร เพื่อนำมาสร้างตารางผลกระทบระหว่างตัวแปร ยังไม่สามารถทำได้แบบอัตโนมัติเหมือนดังเช่นในโปรแกรมภาษาระดับสูง เนื่องจากบางภาคแสดงที่ปรากฏตัวแปรส่งออก (มีเครื่องหมายอัศเจรีย์หลังตัวแปร) มีการดำเนินการด้วยตัวดำเนินการบางอย่าง เช่น ภาคแสดง r! x r! = n? ซึ่งต่างจากโปรแกรมภาษาระดับสูง เช่น ภาษาซี ภาษาปาสคาล ที่กำหนดให้เครื่องหมายเท่ากับ (=) เป็นเครื่องหมายกำหนดค่าที่ได้จากนิพจน์ทางด้านขวาให้กับตัวแปรทางด้านซ้ายอย่างชัดเจน ส่งผลให้การแปลภาษาโปรแกรมกระทำได้ง่ายขึ้น ดังนั้นในการพิจารณาผลกระทบที่เกิดขึ้นกับตัวแปรจะต้องพิจารณาทั้งด้านซ้ายและด้านขวาของตัวดำเนินการซึ่งผู้วิจัยจะต้องเป็นผู้พิจารณา

และการใส่แต่ละข้อผิดพลาดให้กับภาคแสดงในข้อกำหนดรูปนัย ผู้ใช้จะต้องพิจารณาว่า ตัวดำเนินการที่ใช้ในภาคแสดงใช้ดำเนินการเกี่ยวกับอะไร เช่น ตัวดำเนินการเท่ากับซึ่งใช้เปรียบเทียบได้ทั้งตัวแปรที่หมายถึงเซต (กลุ่มของค่าที่เป็นไปได้) และตัวแปรที่หมายถึงค่าเพียงค่าเดียว เป็นต้น ทั้งนี้เพื่อให้การใส่ข้อผิดพลาดกระทำได้ถูกต้อง

5.3 ข้อเสนอแนะ

จากปัญหาและอุปสรรคที่เกี่ยวกับเครื่องมือที่นำมาใช้ในการตรวจสอบไวยกรณ์ของข้อกำหนดรูปนัยแบบสัญกรณ์เซต ผู้วิจัยเห็นว่า ควรมีการพัฒนาเพื่อขยายขีดความสามารถของโปรแกรมให้มีความยืดหยุ่นและสะดวกต่อการใช้งานมากยิ่งขึ้น

และนอกจากการวิเคราะห์มิวเทชันสำหรับข้อกำหนดรูปนัยแบบสัญกรณ์เซต โดยการใส่ข้อผิดพลาดที่อาจจะเกิดขึ้นได้ลงในข้อกำหนดรูปนัย โดยใช้ตัวดำเนินการมิวเทชันที่ได้

นิยามไว้แล้ว ยังมีตัวดำเนินการมิวเทชันอื่นอีกที่สามารถนำมาใช้สร้างกรณีทดสอบ หมายความว่า กรณีทดสอบนั้นได้มาจากหลายส่วน [9] (ไม่ใช่เฉพาะจากการเปลี่ยนตัวดำเนินการ) นั่นคือ จะต้องมาจากส่วนการประกาศในเค้าร่าง ดังนั้นวิธีการวิเคราะห์มิวเทชันสำหรับข้อกำหนดรูปนัยแบบหลายข้อผิดพลาดนี้อาจนำมาใช้ในการปรับปรุงวิเคราะห์มิวเทชันเพื่อลดกรณีทดสอบได้ด้วย

รายการอ้างอิง

[1] J. M. Spivey. The Z Notation: A Reference Manual (Second Edition) (n.p.) : Prentice-Hall, **1992**.

[2] Fangjun Wu, Tong Yi. "Slicing Z specifications", ACM SIGPLAN Notices, 39, 8 (August **2004**).

[3] W.E. Howden. "Weak mutation testing and completeness of test sets", *IEEE Transactions On Software Engineering*, 8 : 371-379, 1982.

[4] R. A. Demillo and A. J. Offutt. "Constraint-Based Automatic Test Data Generation", *IEEE Transaction on Software Engineering*. **(1991)** : 900-910.

[5] P. A. Stock. "Applying formal methods to software testing", *Doctoral Dissertation*, The University of Queensland, Department of Computer Science, **1993**.

[6] P. E. Black, V. Okun, and Y. Yesha. "Mutation of model checker specifications for test generation and evaluation", *Mutation Testing for the New Century*. **(2000)** : 14-20.

[7] P. E. Black, V. Okun, and Y. Yesha. "Mutation operators for specifications", *Proceedings of 15th IEEE International Conference on Automated Software Engineering*. **(2000)** : 81-88.

[8] Fangjun Wu, Tong Yi. "Mutation Operator for Object-Z Specification", Proceedings of the 10th IEEE Internatioanl Conference on Engineering of Comples Computer systems (ICECCS'05). **2005**.

[9] T. Oda, K. Araki. "Specification slicing in formal methods of software development", *In Proceedings of the 17th Annual International Computer Software and Applications Conference, IEEE Computer Society Press.* **(1993)** : 313-319.

[10] M. Brandon, A. Perry. "A slicing approach for parallel component adaptation", *Technical Report ITTC-FY2003-TR-29120-01*. **2002**.

[11] Meisels, Saaltink. "The Z/EVES Reference Manual (for Version 1.5)", Developing Secure and Dependable Systems Using Advanced Software Technology, (September **1997**).

[12] J. Chang, D.J. Richardson. "Static and Dynamic Specification Slicing", *Proceedings of the Fourth Irvine Software Symposium*. **1994**.

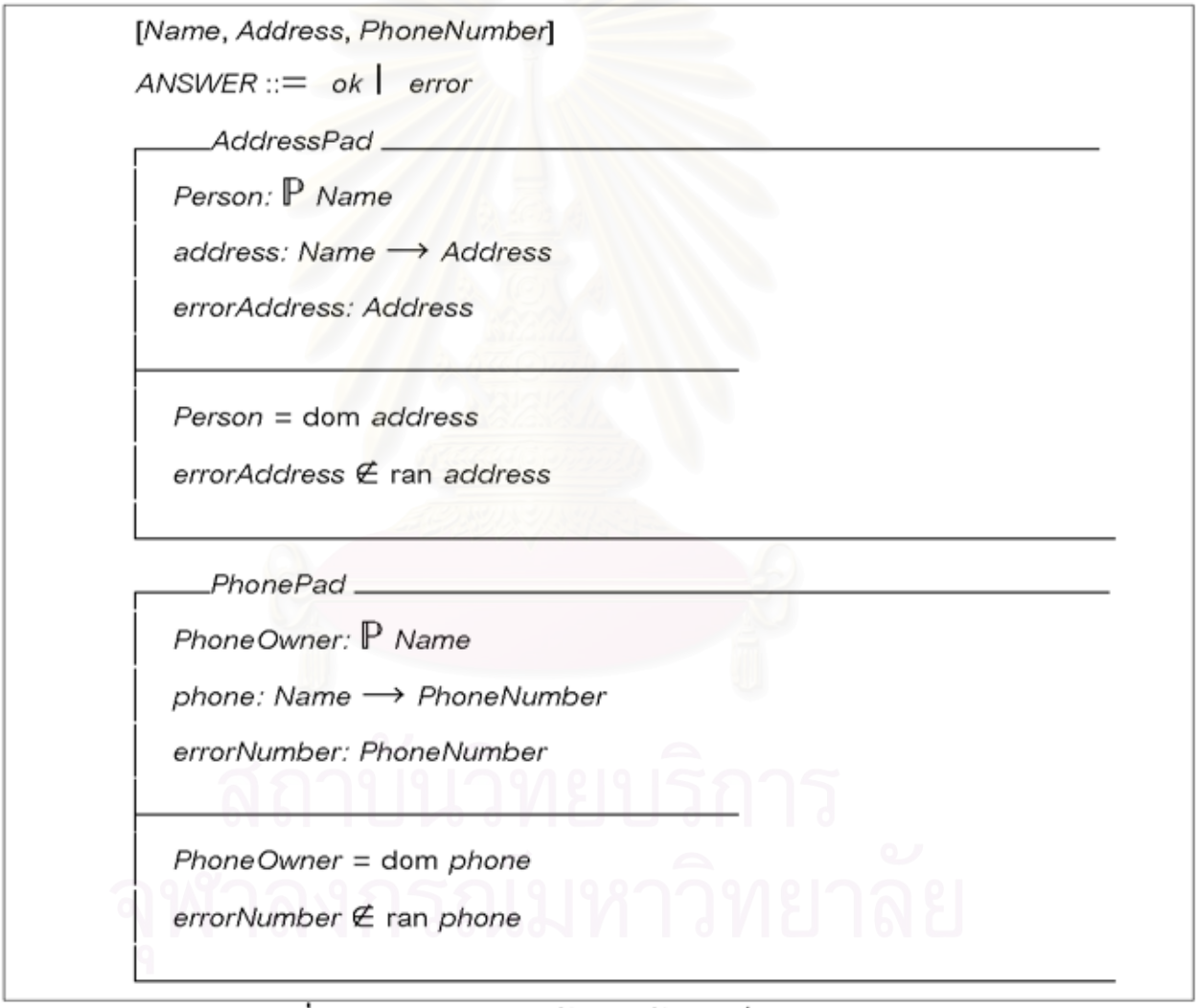
ภาคผนวก

ภาคผนวก ก.

รายละเอียดของข้อกำหนดรูปนัยแบบสัญกรณ์เซดที่ใช้ในการทดลอง

ข้อกำหนดรูปนัยแบบสัญกรณ์เซดที่นำมาใช้ทดลองในงานวิจัยนี้มี 5 ข้อกำหนด ดังต่อไปนี้

1. ข้อกำหนดรูปนัยแบบสัญกรณ์เซดของระบบ A (ระบบ Pocket Book) [6] ได้แสดงไว้ดังรูปที่ ก.1



รูปที่ ก.1 ข้อกำหนดรูปนัยแบบสัญกรณ์เซดของระบบ A

Initialize

$AddressPad'$

$answer! : ANSWER$

$Person' = \varnothing$

$answer! = ok$

AddToAddressPad

$\Delta AddressPad$

$who? : Name$

$whoseAddress? : Address$

$answer! : ANSWER$

$Person' = Person \cup \{who?\}$

$address'$

$= \textbf{if } who? \notin Person \textbf{ then } address \cup \{(who? \mapsto whoseAddress?)\}$

$\textbf{else } address$

$answer! = \textbf{if } who? \notin Person \textbf{ then } ok \textbf{ else } error$

AddToPhonePad

$\Xi AddressPad$

$\Delta PhonePad$

$who? : Name$

$whosePhoneNumber? : PhoneNumber$

$answer! : ANSWER$

$PhoneOwner' = \textbf{if } who? \in Person \textbf{ then } PhoneOwner \cup \{who?\}$

$\textbf{else } PhoneOwner$

รูปที่ ก.1 ข้อกำหนดรูปนัยแบบสัญกรณ์เซตของระบบ A (ต่อ)

$phone'$

$= \textbf{if } who? \in Person \setminus PhoneOwner$

$\textbf{then } phone \cup \{(who? \mapsto whosePhoneNumber?)\}$

$\textbf{else } phone$

$answer! = \textbf{if } who? \in Person \setminus PhoneOwner \textbf{ then } ok \textbf{ else } error$

ErasePerson

$\Delta AddressPad$

$\Delta PhonePad$

$who?: Name$

$answer!: ANSWER$

$Person' = \textbf{if } who? \in Person \textbf{ then } Person \setminus \{who?\} \textbf{ else } Person$

$PhoneOwner' = \textbf{if } who? \in PhoneOwner \textbf{ then } PhoneOwner \setminus \{who?\}$

$\textbf{else } PhoneOwner$

$answer! = \textbf{if } who? \in Person \textbf{ then } ok \textbf{ else } error$

LookupAddress

$\Xi AddressPad$

$who?: Name$

$whoseAddress!: Address$

$answer!: ANSWER$

$whoseAddress! = \textbf{if } who? \in Person \textbf{ then } address\ who? \textbf{ else } errorAddress$

$answer! = \textbf{if } who? \in Person \textbf{ then } ok \textbf{ else } error$

LookupPhoneNumber

$\Xi PhonePad$

$who?: Name$

$whosePhone!: PhoneNumber$

รูปที่ ก.1 ข้อกำหนดรูปนัยแบบสัญกรณ์เซตของระบบ A (ต่อ)

$$answer! : ANSWER$$

$$whosePhone! = \textbf{if } who? \in PhoneOwner \textbf{ then } phone\ who? \textbf{ else } errorNumber$$

$$answer! = \textbf{if } who? \in PhoneOwner \textbf{ then } ok \textbf{ else } error$$

รูปที่ ก.1 ข้อกำหนดรูปนัยแบบสัญกรณ์เซตของระบบ A (ต่อ)

2. ข้อกำหนดรูปนัยแบบสัญกรณ์เซตของระบบ B (ระบบ Elevator) [4] ได้แสดงไว้ดังรูปที่ ก.2

$$MaxFloor == 10$$

$$Direction ::= up \mid down$$

$$DoorState ::= open \mid closed$$

Elevator

$$CurrentFloor : \mathbb{N}_1$$

$$Requests : \mathbb{P}\, \mathbb{N}_1$$

$$UpCalls : \mathbb{P}\, \mathbb{N}_1$$

$$DownCalls : \mathbb{P}\, \mathbb{N}_1$$

$$Dir : Direction$$

$$Door : DoorState$$

$$CurrentFloor \leqslant MaxFloor$$

$$max\ Requests \leqslant MaxFloor$$

$$max\ UpCalls \leqslant MaxFloor$$

$$max\ DownCalls \leqslant MaxFloor$$

InitElevator

$$Elevator$$

$$CurrentFloor = 1$$

$$Requests = \varnothing$$

รูปที่ ก.2 ข้อกำหนดรูปนัยแบบสัญกรณ์เซตของระบบ B

$UpCalls = \varnothing$

$DownCalls = \varnothing$

$Dir = up$

$Door = open$

ElevatorButtonEvent

$\Delta Elevator$

$Floor?: \mathbb{N}_1$

$Floor? \leqslant MaxFloor$

$CurrentFloor' = CurrentFloor$

$Requests' = Requests \cup \{Floor?\}$

$UpCalls' = UpCalls$

$DownCalls' = DownCalls$

$Dir' = Dir$

$Door' = Door$

FloorButtonEvent

$\Delta Elevator$

$Floor?: \mathbb{N}_1$

$CallDir?: Direction$

$Floor? \leqslant MaxFloor$

$CurrentFloor' = CurrentFloor$

$CallDir? = down \implies UpCalls' = UpCalls \land DownCalls' = DownCalls \cup \{Floor?\}$

$CallDir? = up \implies DownCalls' = DownCalls \land UpCalls' = UpCalls \cup \{Floor?\}$

$Requests' = Requests$

รูปที่ ก.2 ข้อกำหนดรูปนัยแบบสัญกรณ์เซตของระบบ B (ต่อ)

$Dir' = Dir$

$Door' = Door$

BasicMoveUp

$\Delta Elevator$

$Dir = up$

$Requests \cup UpCalls \neq \varnothing$

$CurrentFloor \leqslant max\,(Requests \cup UpCalls)$

$CurrentFloor' = min\,\{x: \mathbb{N}_1 \mid x \in Requests \cup UpCalls \wedge x > CurrentFloor\}$

$Requests' = Requests \setminus \{CurrentFloor'\}$

$UpCalls' = UpCalls \setminus \{CurrentFloor'\}$

$Dir' = up$

$Door' = Door$

BasicMoveDown

$\Delta Elevator$

$Dir = down$

$Requests \cup DownCalls \neq \varnothing$

$CurrentFloor \geqslant min\,(Requests \cup DownCalls)$

$CurrentFloor' = max\,\{x: \mathbb{N}_1 \mid x \in Requests \cup DownCalls \wedge x < CurrentFloor\}$

$Requests' = Requests \setminus \{CurrentFloor'\}$

$UpCalls' = UpCalls$

$DownCalls' = DownCalls \setminus \{CurrentFloor'\}$

$Dir' = Dir$

$Door' = Door$

รูปที่ ก.2 ข้อกำหนดรูปนัยแบบสัญกรณ์เซตของระบบ B (ต่อ)

ChangeUpToDown

$\Delta Elevator$

$Dir = up$

$Requests \cup UpCalls \neq \varnothing \land CurrentFloor > max\,(Requests \cup UpCalls)$

$\lor\ Requests \cup UpCalls = \varnothing$

$Requests \cup DownCalls \neq \varnothing$

$CurrentFloor' = max\,(Requests \cup DownCalls)$

$Requests' = Requests \setminus \{CurrentFloor'\}$

$UpCalls' = UpCalls$

$DownCalls' = DownCalls \setminus \{CurrentFloor'\}$

$Dir' = down$

$Door' = Door$

ChangeDownToUp

$\Delta Elevator$

$Dir = down$

$Requests \cup DownCalls \neq \varnothing \land$

$CurrentFloor < min\,(Requests \cup DownCalls)$

$\lor\ Requests \cup DownCalls \neq \varnothing$

$Requests \cup UpCalls \neq \varnothing$

$CurrentFloor' = min\,(Requests \cup UpCalls)$

$Requests' = Requests \setminus \{CurrentFloor'\}$

$UpCalls' = UpCalls \setminus \{CurrentFloor'\}$

$DownCalls' = DownCalls$

$Dir' = up$

$Door' = Door$

รูปที่ ก.2 ข้อกำหนดรูปนัยแบบสัญกรณ์เซตของระบบ B (ต่อ)

RestartMovingUp

$\Delta Elevator$

$Dir = up$

$UpCalls \neq \varnothing$

$CurrentFloor > max\ UpCalls$

$DownCalls \cup Requests = \varnothing$

$CurrentFloor' = min\ UpCalls$

$Requests' = Requests$

$UpCalls' = UpCalls \setminus \{CurrentFloor\}$

$DownCalls' = DownCalls$

$Dir' = up$

$Door' = Door$

RestartMovingDown

$\Delta Elevator$

$Dir = down$

$DownCalls \neq \varnothing$

$CurrentFloor < min\ DownCalls$

$UpCalls \cup Requests = \varnothing$

$CurrentFloor' = max\ DownCalls$

$Requests' = Requests$

$UpCalls' = UpCalls$

$DownCalls' = DownCalls \setminus \{CurrentFloor\}$

$Dir' = down$

$Door' = Door$

รูปที่ ก.2 ข้อกำหนดรูปนัยแบบสัญกรณ์เซตของระบบ B (ต่อ)

NoRequestsOrCalls

$\Xi Elevator$

$Requests \cup UpCalls \cup DownCalls = \varnothing$

OpenDoor

$\Delta Elevator$

$CurrentFloor' = CurrentFloor$

$Requests' = Requests$

$UpCalls' = UpCalls$

$DownCalls' = DownCalls$

$Dir' = Dir$

$Door' = open$

CloseDoor

$\Delta Elevator$

$Requests \cup DownCalls \cup UpCalls \neq \varnothing$

$CurrentFloor' = CurrentFloor$

$Requests' = Requests$

$UpCalls' = UpCalls$

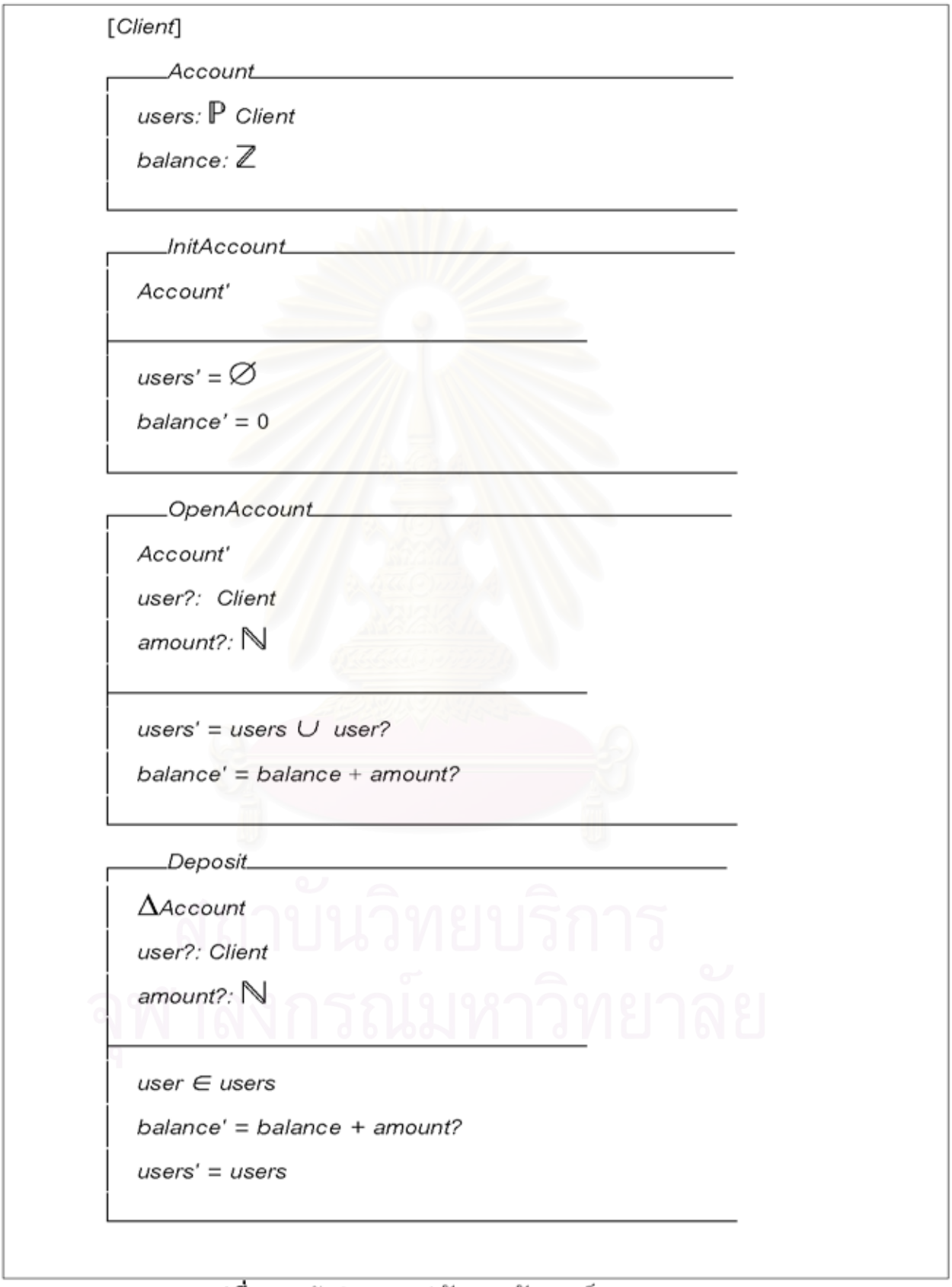
$DownCalls' = DownCalls$

$Dir' = Dir$

$Door' = closed$

รูปที่ ก.2 ข้อกำหนดรูปนัยแบบสัญกรณ์เซตของระบบ B (ต่อ)

3. ข้อกำหนดรูปนัยแบบสัญกรณ์เซตของระบบ C ได้แสดงไว้ดังรูปที่ ก.3



รูปที่ ก.3 ข้อกำหนดรูปนัยแบบสัญกรณ์เซตของระบบ C

Withdraw

$\Delta Account$

user?: Client

amount?: $\mathbb{N}$

---

user $\in$ *users*

balance' = balance - amount?

users' = users

CloseAccount

$\Delta Account$

user?: Client

return!: $\mathbb{N}$

---

user? $\in$ *users*

users' = users $\setminus$ *{user?}*

return! = balance

balance' = 0

ComputeInterest

$\Xi Account$

user?: Client

rate?: $\mathbb{N}$

interest!: $\mathbb{N}$

---

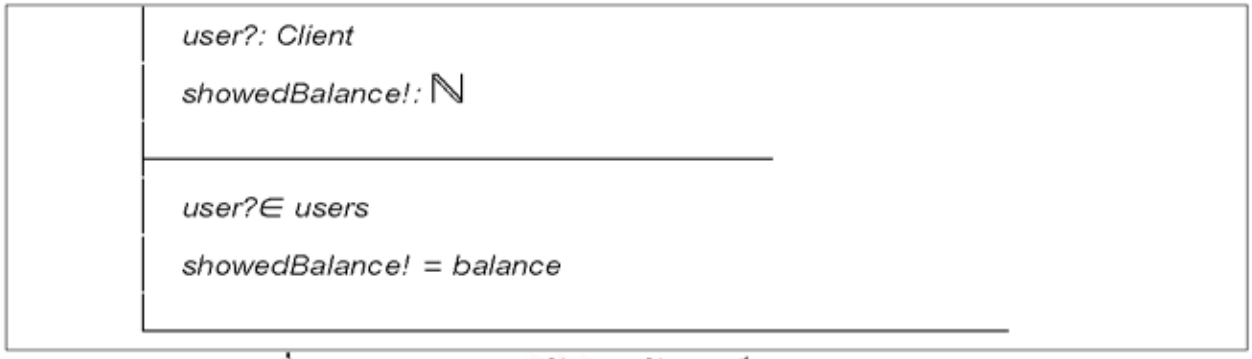
user? $\in$ *users*

interest! = balance rate?*

QueryBalance

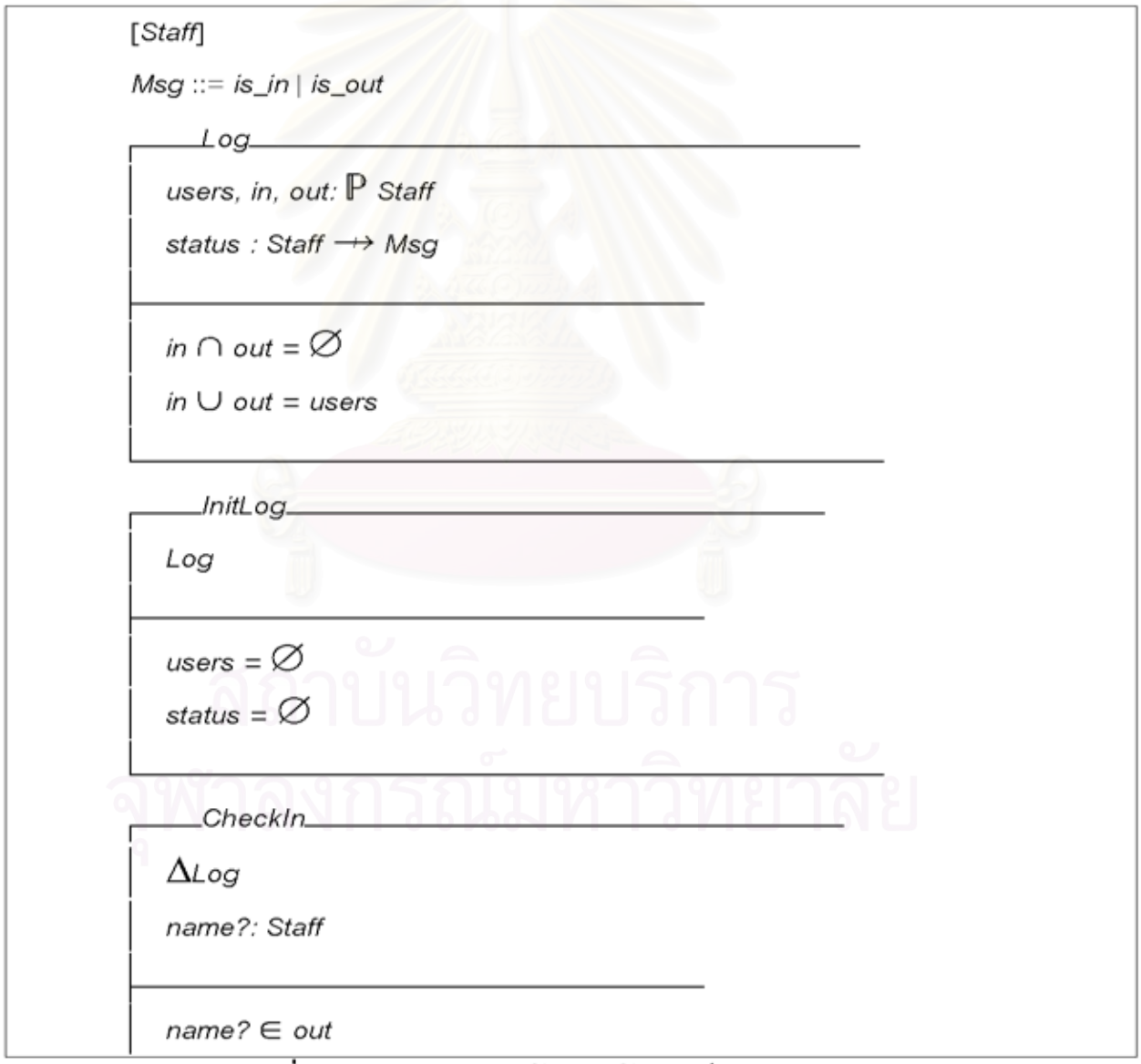
$\Xi Account$

รูปที่ ก.3 ข้อกำหนดรูปนัยแบบสัญกรณ์เซดของระบบ C (ต่อ)



รูปที่ ก.3 ข้อกำหนดรูปนัยแบบสัญกรณ์เซตของระบบ C (ต่อ)

4. ข้อกำหนดรูปนัยแบบสัญกรณ์เซตของระบบ D ได้แสดงไว้ดังรูปที่ ก.4



รูปที่ ก.4 ข้อกำหนดรูปนัยแบบสัญกรณ์เซตของระบบ D

$in' = in \cup \{name?\}$

$\#in' = \#in + 1$

$out' = out \setminus \{name?\}$

$\#out' = \#out - 1$

$status(name?) = is\_in$

CheckOut

ΔLog

$name?: Staff$

$name? \in in$

$in' = in \setminus \{name?\}$

$\#in' = \#in - 1$

$out' = out \cup \{name?\}$

$\#out' = \#out + 1$

$status(name?) = is\_out$

QueryInStaff

ΞLog

$name?: Staff$

$rep!: Msg$

$rep! =$ **if** $name? \in in$ **then** is_in **else** is_out

---

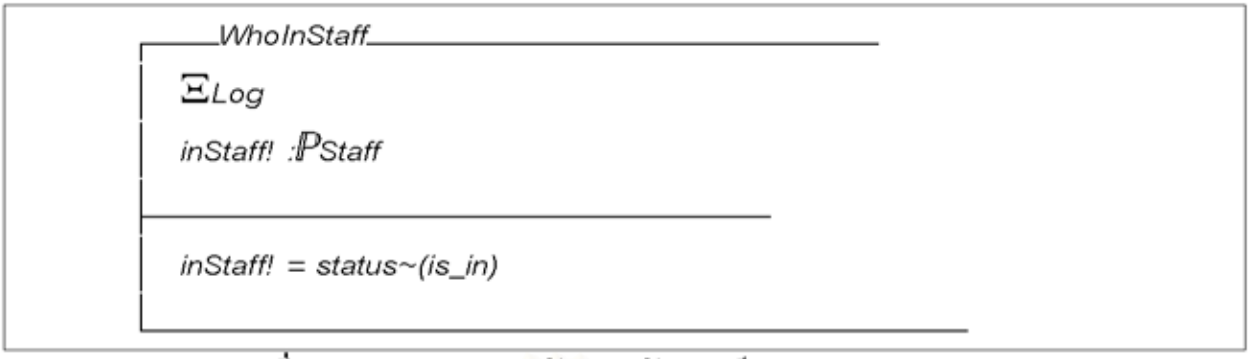
QueryNumberOfInStaff

ΞLog

$amount! : \mathbb{N}$

$amount! = \# in$

รูปที่ ก.4 ข้อกำหนดรูปนัยแบบสัญกรณ์เซดของระบบ D (ต่อ)



รูปที่ ก.4 ข้อกำหนดรูปนัยแบบสัญกรณ์เซตของระบบ D (ต่อ)

5. ข้อกำหนดรูปนัยแบบสัญกรณ์เซตของระบบ E ได้แสดงไว้ดังรูปที่ ก.5

รูปที่ ก.5 ข้อกำหนดรูปนัยแบบสัญกรณ์เซตของระบบ E

$\#seats' = \#seats - 1$

$\#clients' = \#clients + 1$

$clients' = clients \cup \{name? \mapsto seat?\}$

$seats = seats \setminus \{seat?\}$

CancelFlight

$\Delta Flight$

$name? : NAME$

$seat? : SEAT$

$name? \in dom\ clients$

$\#seats' = \#seats + 1$

$\#clients' = \#clients - 1$

$clients' = clients \setminus \{name? \mapsto seat?\}$

$seats = seats \cup \{seat?\}$

QuerySeatNumber

$\Xi Flight$

$name? : NAME$

$seatNumber! : SEAT$

$name? \in dom\ clients$

$seatNumber! = ran\ client(name?)$

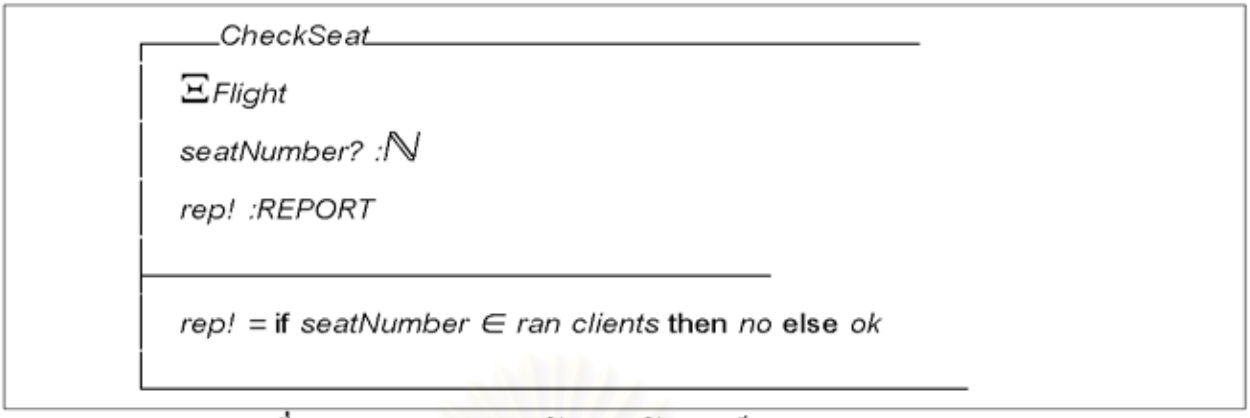
CheckAvailableSeat

$\Xi Flight$

$amount! : \mathbb{N}$

$amount! = \#seats$

รูปที่ ก.5 ข้อกำหนดรูปนัยแบบสัญกรณ์เซตของระบบ E (ต่อ)



รูปที่ ก.5 ข้อกำหนดรูปนัยแบบสัญกรณ์เซตของระบบ E (ต่อ)

ภาคผนวก ข.

ตัวอย่างการทดลองเพื่อหาจำนวนของข้อกำหนดมิวแทนท์

ในส่วนนี้จะเป็นการแสดงการหาจำนวนของข้อกำหนดมิวแทนท์แบบข้อผิดพลาดเดียว (แบบเดิม) จำนวนของข้อผิดพลาดที่ใส่ให้กับภาคแสดงแล้วไม่สามารถหาเงื่อนไขมาตรวจสอบข้อผิดพลาดนั้นได้ จำนวนของข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาด และการคำนวณหาเปอร์เซ็นต์การลดลงของข้อกำหนดมิวแทนท์จากข้อกำหนดรูปนัยแบบสัญกรณ์เซดของระบบ A มีขั้นตอนดังนี้

ขั้นตอนที่ 1 สร้างตารางแสดงผลกระทบที่เกิดขึ้นกับตัวแปร
จากข้อกำหนดจะได้ตัวแปรทั้งสามกลุ่ม ดังนี้
ตัวแปรสถานะ คือ Person address errorAddress PhoneOwner phone errorNumber
ตัวแปรนำเข้า คือ why? whoseAddress? whosePhoneNumber?
ตัวแปรส่งออก คือ answer! whoseAddress! whosePhone!
เมื่อนำมาสร้างตารางผลกระทบที่เกิดขึ้นกับตัวแปรจะได้ดังตารางที่ ข.1

ตารางที่ ข.1 แสดงผลกระทบที่เกิดขึ้นกับตัวแปร

	Person	address	errorAddress	PhoneOwner	Phone	errorNumber	answer!	whoseAddress!	whosePhone!
Person	✓	✓		✓	✓		✓	✓	
address	✓	✓						✓	
errorAddress								✓	
PhoneOwner				✓	✓		✓		✓
Phone				✓	✓				✓
errorNumber									✓
who?	✓	✓		✓	✓		✓	✓	✓
whoseAddress?		✓							
whosePhoneNumber?					✓				

ขั้นตอนที่ 2 ทำการแบ่งกลุ่มตัวแปรตามอัลกอริทึมการแบ่งกลุ่ม
นำตารางที่ ข.1 มาพิจารณาในการแบ่งกลุ่มตัวแปรจะได้ผลลัพธ์ดังตารางที่ ข.2

ตารางที่ ข.2 แสดงกลุ่มของตัวแปรที่สมาชิกในกลุ่มไม่มีผลกระทบถึงกัน

กลุ่มที่ 1	กลุ่มที่ 2	กลุ่มที่ 3	กลุ่มที่ 4
Person	***Address***	*whoseAddress!*	*who?*
errorAddress	***phone***	*whosePhone!*	
PhoneOwner	***answer!***		
errorNumber			
whoseAddress?			
whosePhoneNumber?			

ขั้นตอนที่ 3 วิเคราะห์แต่ละข้อผิดพลาดว่า สามารถหาเงื่อนไขการตรวจสอบได้หรือไม่

ในการวิเคราะห์ข้อผิดพลาดจะต้องทราบก่อนว่าภาคแสดงใดที่สามารถทำการวิเคราะห์มิวเทชันได้

- จากข้อกำหนด ภาคแสดงในแต่ละเค้าร่างที่นำมาใช้สำหรับการวิเคราะห์มิวเทชันมีดังนี้

เค้าร่าง *AddToAddressPad*

001 $Person' = Person \cup \{who?\}$

002 $address' =$ **if** $who? \notin Person$ **then** $address \cup \{(who? \mapsto whoseAddress?)\}$
else *address*

003 $answer! =$ **if** $who? \notin Person$ **then** *ok* **else** *error*

เค้าร่าง *AddToPhonePad*

004 $PhoneOwner' =$ **if** $who? \in Person$ **then** $PhoneOwner \cup \{who?\}$
else *PhoneOwner*

005 $phone' =$ **if** $who? \in Person \setminus PhoneOwner$
then $phone \cup \{(who? \mapsto whosePhoneNumber?)\}$ **else** *phone*

006 $answer! =$ **if** $who? \in Person \setminus PhoneOwner$ **then** *ok* **else** *error*

เค้าร่าง *ErasePerson*

007 $Person' = \textbf{if } who? \in Person \textbf{ then } Person \setminus \{who?\} \textbf{ else } Person$

008 $PhoneOwner' = \textbf{if } who? \in PhoneOwner \textbf{ then } PhoneOwner \setminus \{who?\}$ **else** *PhoneOwner*

009 $answer! = \textbf{if } who? \in Person \textbf{ then } ok \textbf{ else } error$

เค้าร่าง *LookupAddress*

010 $whoseAddress! = \textbf{if } who? \in Person \textbf{ then } address\ who? \textbf{ else } errorAddress$

011 $answer! = \textbf{if } who? \in Person \textbf{ then } ok \textbf{ else } error$

เค้าร่าง *LookupPhoneNumber*

012 $whosePhone! = \textbf{if } who? \in PhoneOwner \textbf{ then } phone\ who? \textbf{ else } errorNumber$

013 $answer! = \textbf{if } who? \in PhoneOwner \textbf{ then } ok \textbf{ else } error$

- ข้อผิดพลาดที่เป็นไปได้ที่สามารถใส่เข้าไปในแต่ละภาคแสดง

เมื่อเปลี่ยนตัวดำเนินการในแต่ละภาคแสดง (เสมือนเป็นการจำลองข้อผิดพลาดที่อาจจะเกิดขึ้นได้) จะได้จำนวนของข้อกำหนดมิวแทนท์แบบข้อผิดพลาดเดียวจากแต่ละภาคแสดง (ระบุตามหมายเลขบรรทัด)

ตารางที่ ข.3 แสดงข้อผิดพลาดที่เป็นไปได้ที่สามารถใส่เข้าไปในแต่ละภาคแสดง

บรรทัดที่	ภาคแสดงที่ใส่ข้อผิดพลาด	จำนวนมิวแทนท์
001	$Person' = Person \cap \{who?\}$ $Person' = Person \setminus \{who?\}$ $Person' \neq Person \cup \{who?\}$ $Person' \subset Person \cup \{who?\}$ $Person' \subseteq Person \cup \{who?\}$	5
002	$address' = \textbf{if } who? \notin Person \textbf{ then } address \cap \{(who? \mapsto whoseAddress?)\} \textbf{ else } address$ $address' = \textbf{if } who? \notin Person \textbf{ then } address \setminus \{(who? \mapsto whoseAddress?)\} \textbf{ else } address$	6

ตารางที่ ข.3 แสดงข้อผิดพลาดที่เป็นไปได้ที่สามารถใส่เข้าไปในแต่ละภาคแสดง (ต่อ)

บรรทัดที่	ภาคแสดงที่ใส่ข้อผิดพลาด	จำนวนมิวแทนท์
	$address' = \textbf{if } who? \in Person \textbf{ then } address \cup \{(who? \mapsto whoseAddress?)\} \textbf{ else } address$ $address' \neq \textbf{if } who? \notin Person \textbf{ then } address \cup \{(who? \mapsto whoseAddress?)\} \textbf{ else } address$ $address' \subset \textbf{if } who? \notin Person \textbf{ then } address \cup \{(who? \mapsto whoseAddress?)\} \textbf{ else } address$ $address' \subseteq \textbf{if } who? \notin Person \textbf{ then } address \cup \{(who? \mapsto whoseAddress?)\} \textbf{ else } address$	
003	$answer! = \textbf{if } who? \in Person \textbf{ then } ok \textbf{ else } error$ $answer! \neq \textbf{if } who? \notin Person \textbf{ then } ok \textbf{ else } error$	2
004	$PhoneOwner' = \textbf{if } who? \in Person \textbf{ then } PhoneOwner \cap \{who?\} \textbf{else } PhoneOwner$ $PhoneOwner' = \textbf{if } who? \in Person \textbf{ then } PhoneOwner \setminus \{who?\} \textbf{else } PhoneOwner$ $PhoneOwner' = \textbf{if } who? \notin Person \textbf{ then } PhoneOwner \cup \{who?\} \textbf{else } PhoneOwner$ $PhoneOwner' \neq \textbf{if } who? \in Person \textbf{ then } PhoneOwner \cup \{who?\} \textbf{else } PhoneOwner$ $PhoneOwner' \subset \textbf{if } who? \in Person \textbf{ then } PhoneOwner \cup \{who?\} \textbf{else } PhoneOwner$ $PhoneOwner' \subseteq \textbf{if } who? \in Person \textbf{ then } PhoneOwner \cup \{who?\} \textbf{else } PhoneOwner$	6
005	$phone' = \textbf{if } who? \in Person \setminus PhoneOwner \textbf{ then } phone \cap \{(who? \mapsto whosePhoneNumber?)\} \textbf{ else } phone$ $phone' = \textbf{if } who? \in Person \setminus PhoneOwner \textbf{ then } phone \setminus \{(who? \mapsto whosePhoneNumber?)\} \textbf{ else } phone$ $phone' = \textbf{if } who? \in Person \cup PhoneOwner \textbf{ then } phone \cup \{(who? \mapsto whosePhoneNumber?)\} \textbf{ else } phone$	8

ตารางที่ ข.3 แสดงข้อผิดพลาดที่เป็นไปได้ที่สามารถใส่เข้าไปในแต่ละภาคแสดง (ต่อ)

บรรทัดที่	ภาคแสดงที่ใส่ข้อผิดพลาด	จำนวนมิวแทนท์
	$phone' = \textbf{if } who? \in Person \cap PhoneOwner \textbf{ then } phone \cup \{(who? \mapsto whosePhoneNumber?)\} \textbf{ else } phone$ $phone' = \textbf{if } who? \notin Person \setminus PhoneOwner \textbf{ then } phone \cup \{(who? \mapsto whosePhoneNumber?)\} \textbf{ else } phone$ $phone' \neq \textbf{if } who? \in Person \setminus PhoneOwner \textbf{ then } phone \cup \{(who? \mapsto whosePhoneNumber?)\} \textbf{ else } phone$ $phone' \subset \textbf{if } who? \in Person \setminus PhoneOwner \textbf{ then } phone \cup \{(who? \mapsto whosePhoneNumber?)\} \textbf{ else } phone$ $phone' \subseteq \textbf{if } who? \in Person \setminus PhoneOwner \textbf{ then } phone \cup \{(who? \mapsto whosePhoneNumber?)\} \textbf{ else } phone$	
006	$answer! = \textbf{if } who? \in Person \cup PhoneOwner \textbf{ then } ok \textbf{ else } error$ $answer! = \textbf{if } who? \in Person \cap PhoneOwner \textbf{ then } ok \textbf{ else } error$ $answer! = \textbf{if } who? \notin Person \setminus PhoneOwner \textbf{ then } ok \textbf{ else } error$ $answer! \neq \textbf{if } who? \in Person \setminus PhoneOwner \textbf{ then } ok \textbf{ else } error$	4
007	$Person' = \textbf{if } who? \in Person \textbf{ then } Person \cup \{who?\} \textbf{ else } Person$ $Person' = \textbf{if } who? \in Person \textbf{ then } Person \cap \{who?\} \textbf{ else } Person$ $Person' = \textbf{if } who? \notin Person \textbf{ then } Person \setminus \{who?\} \textbf{ else } Person$ $Person' \neq \textbf{if } who? \in Person \textbf{ then } Person \setminus \{who?\} \textbf{ else } Person$ $Person' \subset \textbf{if } who? \in Person \textbf{ then } Person \setminus \{who?\} \textbf{ else } Person$ $Person' \subseteq \textbf{if } who? \in Person \textbf{ then } Person \setminus \{who?\} \textbf{ else } Person$	6
008	$PhoneOwner' = \textbf{if } who? \in PhoneOwner \textbf{ then } PhoneOwner \cup \{who?\} \textbf{ else } PhoneOwner$ $PhoneOwner' = \textbf{if } who? \in PhoneOwner \textbf{ then } PhoneOwner \cap \{who?\} \textbf{ else } PhoneOwner$ $PhoneOwner' = \textbf{if } who? \notin PhoneOwner \textbf{ then } PhoneOwner \setminus \{who?\} \textbf{ else } PhoneOwner$ $PhoneOwner' \neq \textbf{if } who? \in PhoneOwner \textbf{ then } PhoneOwner \setminus \{who?\} \textbf{ else } PhoneOwner$	6

ตารางที่ ข.3 แสดงข้อผิดพลาดที่เป็นไปได้ที่สามารถใส่เข้าไปในแต่ละภาคแสดง (ต่อ)

บรรทัดที่	ภาคแสดงที่ใส่ข้อผิดพลาด	จำนวนมิวแทนท์
	PhoneOwner' $\subset$ **if** *who?* $\in$ *PhoneOwner* **then** *PhoneOwner* $\setminus$ $\{$*who?*$\}$ **else** *PhoneOwner* *PhoneOwner'* $\subseteq$ **if** *who?* $\in$ *PhoneOwner* **then** *PhoneOwner* $\setminus$ $\{$*who?*$\}$ **else** *PhoneOwner*	
009	*answer!* = **if** *who?* $\notin$ *Person* **then** *ok* **else** *error* *answer!* $\neq$ **if** *who?* $\in$ *Person* **then** *ok* **else** *error*	2
010	*whoseAddress!* = **if** *who?* $\notin$ *Person* **then** *address who?* **else** *errorAddress* *whoseAddress!* $\neq$ **if** *who?* $\in$ *Person* **then** *address who?* **else** *errorAddress*	2
011	*answer!* = **if** *who?* $\notin$ *Person* **then** *ok* **else** *error* *answer!* $\neq$ **if** *who?* $\in$ *Person* **then** *ok* **else** *error*	2
012	*whosePhone!* = **if** *who?* $\notin$ *PhoneOwner* **then** *phone who?* **else** *errorNumber* *whosePhone!* $\neq$ **if** *who?* $\in$ *PhoneOwner* **then** *phone who?* **else** *errorNumber*	2
013	*answer!* = **if** *who?* $\notin$ *PhoneOwner* **then** *ok* **else** *error* *answer!* $\neq$ **if** *who?* $\in$ *PhoneOwner* **then** *ok* **else** *error*	2

รวมจำนวนของข้อกำหนดมิวแทนท์แบบข้อผิดพลาดเดียวเท่ากับ 53 ข้อกำหนดมิวแทนท์

เมื่อพิจารณาเฉพาะภาคแสดงที่ใส่ข้อผิดพลาดและไม่สามารถหาเงื่อนไขการตรวจสอบข้อผิดพลาดนั้นได้ จะได้ดังตารางที่ ข.4

ตารางที่ ข.4 แสดงภาคแสดงที่ใส่ข้อผิดพลาดและไม่สามารถหาเงื่อนไขการตรวจสอบได้

บรรทัดที่	ภาคแสดงที่ใส่ข้อผิดพลาด	จำนวนมิวแทนท์
001	*Person'* $\neq$ *Person* $\cup$ $\{$*who?*$\}$ *Person'* $\subset$ *Person* $\cup$ $\{$*who?*$\}$ *Person'* $\subseteq$ *Person* $\cup$ $\{$*who?*$\}$	3

ตารางที่ ข.4 แสดงภาคแสดงที่ใส่ข้อผิดพลาดและไม่สามารถหาเงื่อนไขการตรวจสอบได้ (ต่อ)

บรรทัดที่	ภาคแสดงที่ใส่ข้อผิดพลาด	จำนวนมิวแทนท์
002	$address' \neq$ if $who? \notin Person$ then $address \cup \{(who? \mapsto whoseAddress?)\}$ else $address$ $address' \subset$ if $who? \notin Person$ then $address \cup \{(who? \mapsto whoseAddress?)\}$ else $address$ $address' \subseteq$ if $who? \notin Person$ then $address \cup \{(who? \mapsto whoseAddress?)\}$ else $address$	3
004	$PhoneOwner' \neq$ if $who? \in Person$ then $PhoneOwner \cup \{who?\}$ else $PhoneOwner$ $PhoneOwner' \subset$ if $who? \in Person$ then $PhoneOwner \cup \{who?\}$ else $PhoneOwner$ $PhoneOwner' \subseteq$ if $who? \in Person$ then $PhoneOwner \cup \{who?\}$ else $PhoneOwner$	3
005	$phone' \neq$ if $who? \in Person \setminus PhoneOwner$ then $phone \cup \{(who? \mapsto whosePhoneNumber?)\}$ else $phone$ $phone' \subset$ if $who? \in Person \setminus PhoneOwner$ then $phone \cup \{(who? \mapsto whosePhoneNumber?)\}$ else $phone$ $phone' \subseteq$ if $who? \in Person \setminus PhoneOwner$ then $phone \cup \{(who? \mapsto whosePhoneNumber?)\}$ else $phone$	3
007	$Person' \neq$ if $who? \in Person$ then $Person \setminus \{who?\}$ else $Person$ $Person' \subset$ if $who? \in Person$ then $Person \setminus \{who?\}$ else $Person$ $Person' \subseteq$ if $who? \in Person$ then $Person \setminus \{who?\}$ else $Person$	3
008	$PhoneOwner' \neq$ if $who? \in PhoneOwner$ then $PhoneOwner \setminus \{who?\}$ else $PhoneOwner$ $PhoneOwner' \subset$ if $who? \in PhoneOwner$ then $PhoneOwner \setminus \{who?\}$ else $PhoneOwner$ $PhoneOwner' \subseteq$ if $who? \in PhoneOwner$ then $PhoneOwner \setminus \{who?\}$ else $PhoneOwner$	3

จำนวนของข้อกำหนดมิวแทนท์ที่ใส่ข้อผิดพลาดที่เป็นไปได้และไม่สามารถหาเงื่อนไขการตรวจสอบได้เท่ากับ 18 ข้อกำหนด

ขั้นตอนที่ 4 แบ่งกลุ่มของภาคแสดงตามตัวแปรทั้งหมด

ตารางที่ ข.5 แสดงกลุ่มของภาคแสดงตามตัวแปรในกลุ่มที่ 1

ตัวแปร	ภาคแสดง
Person	$Person' = Person \cap \{who?\}$ $Person' = Person \setminus \{who?\}$ $Person' = \textbf{if } who? \in Person \textbf{ then } Person \cup \{who?\} \textbf{ else } Person$ $Person' = \textbf{if } who? \in Person \textbf{ then } Person \cap \{who?\} \textbf{ else } Person$ $Person' = \textbf{if } who? \notin Person \textbf{ then } Person \setminus \{who?\} \textbf{ else } Person$
errorAddress	-
PhoneOwner	$PhoneOwner' = \textbf{if } who? \in Person \textbf{ then } PhoneOwner \cap \{who?\} \textbf{else } PhoneOwner$ $PhoneOwner' = \textbf{if } who? \in Person \textbf{ then } PhoneOwner \setminus \{who?\} \textbf{else } PhoneOwner$ $PhoneOwner' = \textbf{if } who? \notin Person \textbf{ then } PhoneOwner \cup \{who?\} \textbf{else } PhoneOwner$ $PhoneOwner' = \textbf{if } who? \in PhoneOwner \textbf{ then } PhoneOwner \cup \{who?\} \textbf{ else } PhoneOwner$ $PhoneOwner' = \textbf{if } who? \in PhoneOwner \textbf{ then } PhoneOwner \cap \{who?\} \textbf{ else } PhoneOwner$ $PhoneOwner' = \textbf{if } who? \notin PhoneOwner \textbf{ then } PhoneOwner \setminus \{who?\} \textbf{ else } PhoneOwner$
errorNumber	-
whoseAddress?	-
whosePhoneNumber?	-

ตารางที่ ข.6 แสดงกลุ่มของภาคแสดงตามตัวแปรในกลุ่มที่ 2

ตัวแปร	ภาคแสดง
address	$address' = \text{if } who? \notin Person \text{ then } address \cap \{(who? \mapsto whoseAddress?)\} \text{ else } address$ $address' = \text{if } who? \notin Person \text{ then } address \setminus \{(who? \mapsto whoseAddress?)\} \text{ else } address$ $address' = \text{if } who? \in Person \text{ then } address \cup \{(who? \mapsto whoseAddress?)\} \text{ else } address$
phone	$phone' = \text{if } who? \in Person \setminus PhoneOwner \text{ then } phone \cap \{(who? \mapsto whosePhoneNumber?)\} \text{ else } phone$ $phone' = \text{if } who? \in Person \setminus PhoneOwner \text{ then } phone \setminus \{(who? \mapsto whosePhoneNumber?)\} \text{ else } phone$ $phone' = \text{if } who? \in Person \cup PhoneOwner \text{ then } phone \cup \{(who? \mapsto whosePhoneNumber?)\} \text{ else } phone$ $phone' = \text{if } who? \in Person \cap PhoneOwner \text{ then } phone \cup \{(who? \mapsto whosePhoneNumber?)\} \text{ else } phone$ $phone' = \text{if } who? \notin Person \setminus PhoneOwner \text{ then } phone \cup \{(who? \mapsto whosePhoneNumber?)\} \text{ else } phone$
answer!	$answer! = \text{if } who? \in Person \text{ then } ok \text{ else } error$ $answer! \neq \text{if } who? \notin Person \text{ then } ok \text{ else } error$ $answer! = \text{if } who? \in Person \cup PhoneOwner \text{ then } ok \text{ else } error$ $answer! = \text{if } who? \in Person \cap PhoneOwner \text{ then } ok \text{ else } error$ $answer! = \text{if } who? \notin Person \setminus PhoneOwner \text{ then } ok \text{ else } error$ $answer! \neq \text{if } who? \in Person \setminus PhoneOwner \text{ then } ok \text{ else } error$ $answer! = \text{if } who? \notin Person \text{ then } ok \text{ else } error$ $answer! \neq \text{if } who? \in Person \text{ then } ok \text{ else } error$ $answer! = \text{if } who? \notin Person \text{ then } ok \text{ else } error$ $answer! \neq \text{if } who? \in Person \text{ then } ok \text{ else } error$ $answer! = \text{if } who? \notin PhoneOwner \text{ then } ok \text{ else } error$ $answer! \neq \text{if } who? \in PhoneOwner \text{ then } ok \text{ else } error$

ตารางที่ ข.7 แสดงกลุ่มของภาคแสดงตามตัวแปรในกลุ่มที่ 3

ตัวแปร	ภาคแสดง
whoseAddress!	*whoseAddress!* = **if** *who?* $\notin$ *Person* **then** *address who?* **else** *errorAddress* *whoseAddress!* $\neq$ **if** *who?* $\in$ *Person* **then** *address who?* **else** *errorAddress*
whosePhone!	*whosePhone!* = **if** *who?* $\notin$ *PhoneOwner* **then** *phone who?* **else** *errorNumber* *whosePhone!* $\neq$ **if** *who?* $\in$ *PhoneOwner* **then** *phone who?* **else** *errorNumber*

ตารางที่ ข.8 แสดงกลุ่มของภาคแสดงตามตัวแปรในกลุ่มที่ 4

ตัวแปร	ภาคแสดง
who?	-

ขั้นตอนที่ 5 สร้างข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดตามอัลกอริทึมการสร้างข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาด

ตารางที่ ข.9 แสดงภาคแสดงที่ใส่ข้อผิดพลาดเข้าไปในข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดจากตัวแปรในกลุ่มที่ 1

ข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาด	ภาคแสดงที่ใส่ข้อผิดพลาด
1	*Person'* = *Person* $\cap$ {*who?*} *PhoneOwner'* = **if** *who?* $\in$ *Person* **then** *PhoneOwner* $\cap$ {*who?*}**else** *PhoneOwner*
2	*Person'* = *Person* $\setminus$ {*who?*} *PhoneOwner'* = **if** *who?* $\in$ *Person* **then** *PhoneOwner* $\setminus$ {*who?*}**else** *PhoneOwner*
3	*Person'* = **if** *who?* $\in$ *Person* **then** *Person* $\cup$ {*who?*} **else** *Person* *PhoneOwner'* = **if** *who?* $\notin$ *Person* **then** *PhoneOwner* $\cup$ {*who?*}**else** *PhoneOwner*

ตารางที่ ข.9 แสดงภาคแสดงที่ใส่ข้อผิดพลาดเข้าไปในข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดจากตัวแปรในกลุ่มที่ 1 (ต่อ)

ข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาด	ภาคแสดงที่ใส่ข้อผิดพลาด
4	$Person' = \textbf{if } who? \in Person \textbf{ then } Person \cap \{who?\} \textbf{ else } Person$ $PhoneOwner' = \textbf{if } who? \in PhoneOwner \textbf{ then } PhoneOwner \cup \{who?\} \textbf{ else } PhoneOwner$
5	$Person' = \textbf{if } who? \notin Person \textbf{ then } Person \setminus \{who?\} \textbf{ else } Person$ $PhoneOwner' = \textbf{if } who? \in PhoneOwner \textbf{ then } PhoneOwner \cap \{who?\} \textbf{ else } PhoneOwner$
6	$PhoneOwner' = \textbf{if } who? \notin PhoneOwner \textbf{ then } PhoneOwner \setminus \{who?\} \textbf{ else } PhoneOwner$

จำนวนของข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดจากกลุ่มที่ 1 เท่ากับ 6 ข้อกำหนด

ตารางที่ ข.10 แสดงภาคแสดงที่ใส่ข้อผิดพลาดเข้าไปในข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดจากตัวแปรในกลุ่มที่ 2

ข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาด	ภาคแสดงที่ใส่ข้อผิดพลาด
1	$address' = \textbf{if } who? \notin Person \textbf{ then } address \cap \{(who? \mapsto whoseAddress?)\} \textbf{ else } address$ $phone' = \textbf{if } who? \in Person \setminus PhoneOwner \textbf{ then } phone \cap \{(who? \mapsto whosePhoneNumber?)\} \textbf{ else } phone$ $answer! = \textbf{if } who? \in Person \textbf{ then } ok \textbf{ else } error$
2	$address' = \textbf{if } who? \notin Person \textbf{ then } address \setminus \{(who? \mapsto whoseAddress?)\} \textbf{ else } address$ $phone' = \textbf{if } who? \in Person \setminus PhoneOwner \textbf{ then } phone \setminus \{(who? \mapsto whosePhoneNumber?)\} \textbf{ else } phone$ $answer! \neq \textbf{if } who? \notin Person \textbf{ then } ok \textbf{ else } error$

ตารางที่ ข.10 แสดงภาคแสดงที่ใส่ข้อผิดพลาดเข้าไปในข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดจากตัวแปรในกลุ่มที่ 2 (ต่อ)

ข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาด	ภาคแสดงที่ใส่ข้อผิดพลาด
3	$address' = \textbf{if } who? \in Person \textbf{ then } address \cup \{(who? \mapsto whoseAddress?)\} \textbf{ else } address$ $phone' = \textbf{if } who? \in Person \cup PhoneOwner \textbf{ then } phone \cup \{(who? \mapsto whosePhoneNumber?)\} \textbf{ else } phone$ $answer! = \textbf{if } who? \in Person \cup PhoneOwner \textbf{ then } ok \textbf{ else } error$
4	$phone' = \textbf{if } who? \in Person \cap PhoneOwner \textbf{ then } phone \cup \{(who? \mapsto whosePhoneNumber?)\} \textbf{ else } phone$ $answer! = \textbf{if } who? \in Person \cap PhoneOwner \textbf{ then } ok \textbf{ else } error$
5	$phone' = \textbf{if } who? \notin Person \setminus PhoneOwner \textbf{ then } phone \cup \{(who? \mapsto whosePhoneNumber?)\} \textbf{ else } phone$ $answer! = \textbf{if } who? \notin Person \setminus PhoneOwner \textbf{ then } ok \textbf{ else } error$
6	$answer! \neq \textbf{if } who? \in Person \setminus PhoneOwner \textbf{ then } ok \textbf{ else } error$
7	$answer! = \textbf{if } who? \notin Person \textbf{ then } ok \textbf{ else } error$
8	$answer! \neq \textbf{if } who? \in Person \textbf{ then } ok \textbf{ else } error$
9	$answer! = \textbf{if } who? \notin Person \textbf{ then } ok \textbf{ else } error$
10	$answer! \neq \textbf{if } who? \in Person \textbf{ then } ok \textbf{ else } error$
11	$answer! = \textbf{if } who? \notin PhoneOwner \textbf{ then } ok \textbf{ else } error$
12	$answer! \neq \textbf{if } who? \in PhoneOwner \textbf{ then } ok \textbf{ else } error$

จำนวนของข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดจากกลุ่มที่ 2 เท่ากับ 12 ข้อกำหนด

ตารางที่ ข.11 แสดงภาคแสดงที่ใส่ข้อผิดพลาดเข้าไปในข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดจากตัวแปรในกลุ่มที่ 3

ข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาด	ภาคแสดงที่ใส่ข้อผิดพลาด
1	*whoseAddress!* = **if** *who?* $\notin$ *Person* **then** *address who?* **else** *errorAddress* *whosePhone!* = **if** *who?* $\notin$ *PhoneOwner* **then** *phone who?* **else** *errorNumber*
2	*whoseAddress!* $\neq$ **if** *who?* $\in$ *Person* **then** *address who?* **else** *errorAddress* *whosePhone!* $\neq$ **if** *who?* $\in$ *PhoneOwner* **then** *phone who?* **else** *errorNumber*

จำนวนของข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดจากกลุ่มที่ 3 เท่ากับ 2 ข้อกำหนด

รวมจำนวนข้อกำหนดมิวแทนท์แบบหลายข้อผิดพลาดจากทุกกลุ่มเท่ากับ 20 ข้อกำหนด

ประวัติผู้เขียนวิทยานิพนธ์

นายวรวิทย์ จิตรงค์ เกิดเมื่อวันที่ 8 พฤศจิกายน 2518 ที่จังหวัดตรัง จบการศึกษาระดับมัธยมศึกษาตอนต้นและตอนปลายจากโรงเรียนวิเชียรมาตุ อำเภอเมือง จังหวัดตรัง เข้าศึกษาต่อในระดับปริญญาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยหอการค้าไทย จนสำเร็จการศึกษาเมื่อปีการศึกษา 2539 และได้เข้าทำงานในตำแหน่งอาจารย์ประจำหลักสูตรปริญญาตรี สาขาวิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยหอการค้าไทย ต่อมาปีการศึกษา 2546 ลาศึกษาต่อในระดับบัณฑิตศึกษา สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย