บทที่ 5

การทดสอบและสรุปผล

ในบทนี้จะกล่าวถึงการทดสอบเครื่องมือแปลงแผนภาพคลาสเป็นข้อกำหนดเชด และวิเคราะห์ข้อกำหนดเชดที่ได้โดยการสร้างเงื่อนไขก่อน การสร้างกรณีทดสอบ และการทดสอบทฤษฎีบท

5.1 ขั้นตอนการติดตั้งโปรแกรม

ทำการติดตั้งโปรแกรม CDZ โดยการเรียก SETUP.EXE แล้วปฏิบัติตามขั้นตอนที่โปรแกรมแนะนำเพื่อติดตั้งเครื่องมือ สำหรับวิธีการใช้งานโปรแกรม CDZ สามารถศึกษาได้จากภาคผนวก ก

5.2 สภาวะที่ใช้ทดสอบโปรแกรม

เครื่องคอมพิวเตอร์และระบบปฏิบัติการที่ใช้ในการทดสอบ มีรายละเอียดดังนี้

- คอมพิวเตอร์พีซี Pentium 4 1.6 กิกะเฮิร์ส
- หน่วยความจำหลัก 512 เมกะไบต์
- ฮาร์ดดิสก์ความจุ 60 กิกะไบต์
- ระบบปฏิบัติการไมโครซอฟท์วินโดว์ส 2000

5.3 ระบบที่ใช้ทดสอบโปรแกรม

ขั้นตอนการทดสอบเครื่องมือซอฟต์แวร์ มีลำดับดังนี้

- กำหนดระบบที่ใช้ทดสอบ ซึ่งมี 3 ระบบคือ ระบบสารสนเทศของสถานศึกษา (IS for a school) [7] ระบบฐานข้อมูลถังเลี้ยงปลา (Aquarium Database) [15] และระบบเครื่องฝากถอนเงินอัตโนมัติ (Automatic Teller Machine) [16]
- สร้างข้อกำหนดเชดของแต่ละระบบที่ใช้ทดสอบจากแผนภาพคลาสของระบบ
- ตรวจสอบวากยสัมพันธ์ของข้อกำหนดเชดที่ได้โดยใช้ Z/EVES
- ตรวจสอบเค้าร่างของข้อกำหนดเชดที่สร้างขึ้น ว่าเค้าร่างแต่ละเค้าร่างเป็นไปตามกฎการแปลงแผนภาพคลาสเป็นข้อกำหนดเชดในบทที่ 3 หรือไม่ และนับจำนวนเค้าร่างทั้งหมดในแต่ละระบบ
- วิเคราะห์ข้อกำหนดเชดที่ได้โดยใช้ Z/EVES โดยการสร้างเงื่อนไขก่อน การสร้างกรณีทดสอบและการทดสอบทฤษฎีบท การสร้างเงื่อนไขก่อนเป็นวิธีที่ใช้เพื่อตรวจสอบการดำเนินการพื้นฐานว่าสถานะก่อนของการดำเนินการนั้น ๆ มีความความถูกต้อง ไม่มีข้อผิดพลาด ซึ่งถ้าหากเงื่อนไขก่อนเป็น

เท็จแสดงว่าการนิยามมีความขัดแย้งกัน นั่นคือไม่มีสถานะก่อนที่เหมาะสม แต่ถ้าหากเงื่อนไขก่อนเป็นจริงแสดงว่าสถานะก่อนของการดำเนินการพื้นฐานนั้นถูกต้อง ส่วนการสร้างกรณีทดสอบเป็นการตรวจสอบผลลัพธ์ที่ได้เมื่อมีการทำกับการดำเนินการพื้นฐานต่าง ๆ เป็นลำดับกัน และการทดสอบทฤษฎีบทเป็นการตรวจสอบข้อกำหนดโดยการสร้างเป็นทฤษฎีบท โดยในงานวิจัยนี้จะทำการทดสอบค่าของตัวแปรเข้าเมื่อมีการทำกับการดำเนินการพื้นฐานต่าง ๆ เป็นลำดับกัน

สำหรับรายละเอียดในการทดสอบระบบทั้ง 3 ระบบ ดังนี้

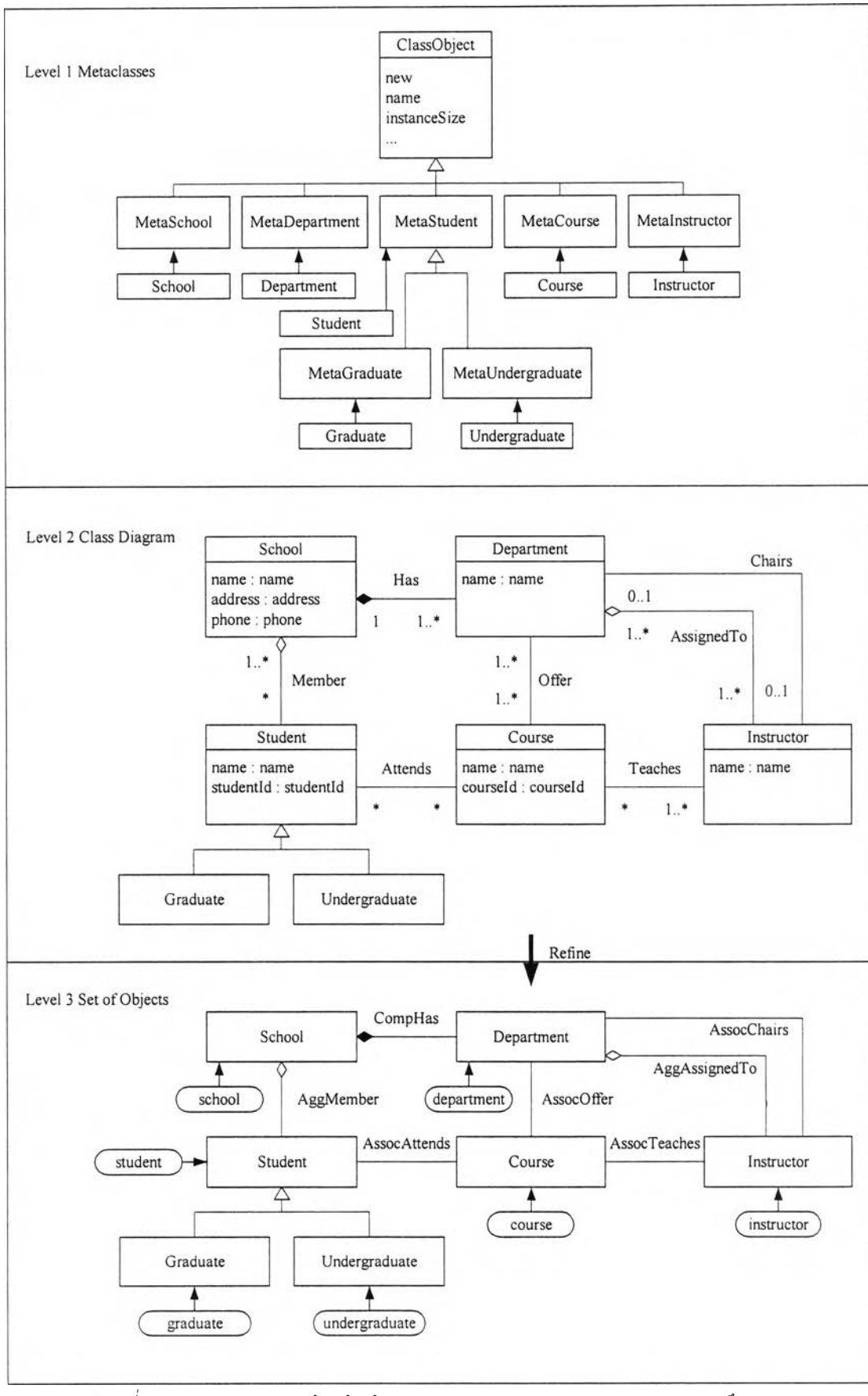
5.3.1 ระบบสารสนเทศของสถานศึกษา

ระบบสารสนเทศของสถานศึกษา เป็นระบบที่ใช้ในการบริหารงานเกี่ยวกับสถานศึกษา โดยนิสิตที่ได้ขึ้นทะเบียนเป็นนิสิตของสถานศึกษานั้นจะสามารถลงทะเบียนเรียนได้ โดยวิชาที่เปิดให้ลงทะเบียนเรียนนั้น อาจเปิดโดยแผนกต่าง ๆ หนึ่งหรือหลายแผนกก็ได้ และต้องมีผู้สอนอย่างน้อยหนึ่งคน ซึ่งผู้สอนแต่ละคนจะต้องสังกัดอย่างน้อยหนึ่งแผนก และอาจเป็นหัวหน้าแผนกใด ๆ ได้เพียงหนึ่งแผนกเท่านั้น นอกจากนี้นิสิตยังแบ่งได้เป็น 2 ชนิดคือ นิสิตหลักสูตรปริญญาบัณฑิต และนิสิตหลักสูตรที่สูงกว่าปริญญาบัณฑิต ดังในรูปที่ 5.1 ในระดับที่ 2 แสดงแผนภาพคลาสของระบบสารสนเทศของสถานศึกษา

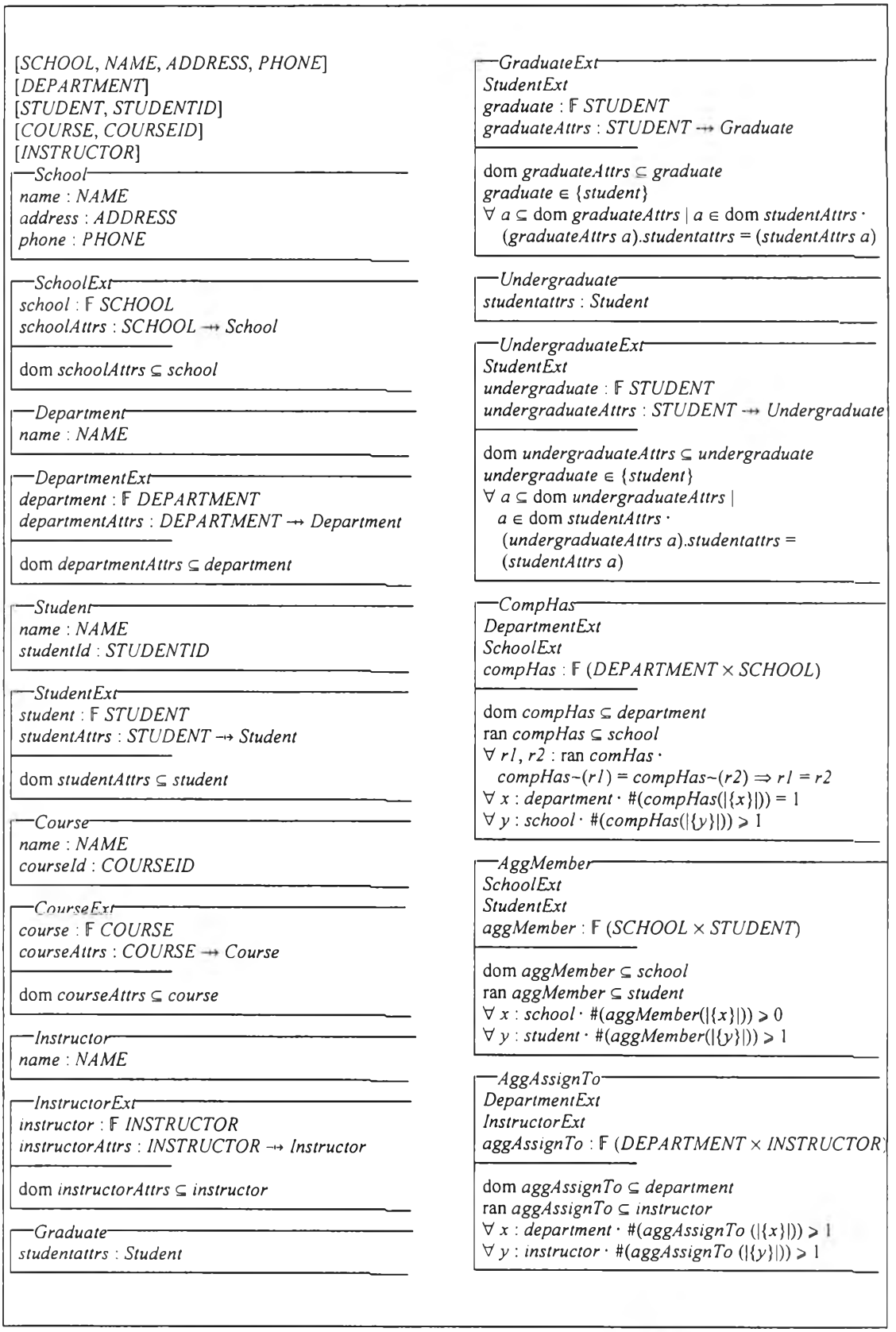
จากรูปที่ 5.1 สามารถสร้างเป็นข้อกำหนดเซตของระบบสารสนเทศของสถานศึกษาได้ดังรูปที่ 5.2 โดยข้อมูลนำเข้าและข้อกำหนดเซตที่ได้จากเครื่องมือซอฟต์แวร์ ที่อยู่ในรูปแทคของลาเท็กซ์แสดงในภาคผนวก ข

การตรวจสอบผลลัพธ์ของข้อกำหนดเซตที่สร้างขึ้น โดยตรวจสอบวากยสัมพันธ์ของข้อกำหนดเซตที่ได้ และวิเคราะห์ข้อกำหนดเซตที่สร้างจากระบบสารสนเทศของสถานศึกษา ได้ดังนี้

1) ตรวจสอบวากยสัมพันธ์ของข้อกำหนดเซตที่ได้โดยใช้ Z/EVES พบว่าข้อกำหนดที่ได้นั้นมีความถูกต้องตามวากยสัมพันธ์ของข้อกำหนดเซต
2) ระบบสารสนเทศของสถานศึกษามีจำนวนคลาสทั้งหมด 7 คลาส และมีความสัมพันธ์ทั้งหมด 7 ความสัมพันธ์ เมื่อตรวจสอบเค้าร่างของข้อกำหนดเซตที่สร้างขึ้น พบว่าเค้าร่างทุกเค้าร่างในระบบสารสนเทศของสถานศึกษา เป็นไปตามกฎการแปลงแผนภาพคลาสเป็นข้อกำหนดเซตในบทที่ 3 และเมื่อนับจำนวนเค้าร่างทั้งหมดในระบบได้ผลดังนี้
   - จำนวนเค้าร่างคลาสมีทั้งหมด 7 เค้าร่างคือ School Department Student Course Instructor Graduate และ Undergraduate
   - จำนวนเค้าร่างเพิ่มเติมของคลาสมีทั้งหมด 7 เค้าร่างคือ SchoolExt DepartmentExt StudentExt CourseExt InstructorExt GraduateExt และ UndergraduateExt
   - จำนวนเค้าร่างความสัมพันธ์มีทั้งหมด 7 เค้าร่างคือ CompHas AggMember AggAssignTo AssocChairs AssocAttends AssocOffer และ AssocTeaches

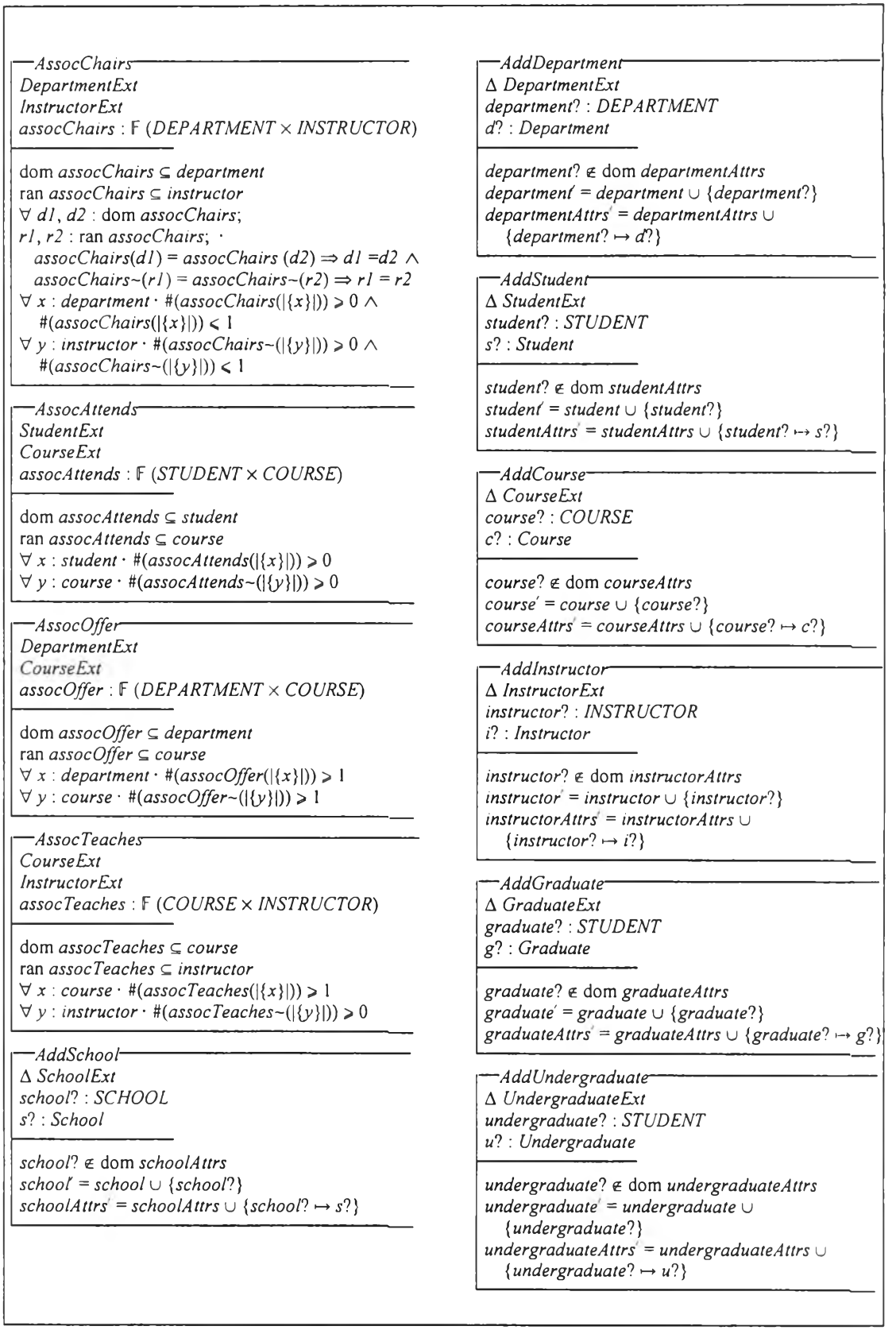


รูปที่ 5.1 ระดับของการสร้างข้อกำหนดเชิงของระบบสารสนเทศของสถานศึกษา

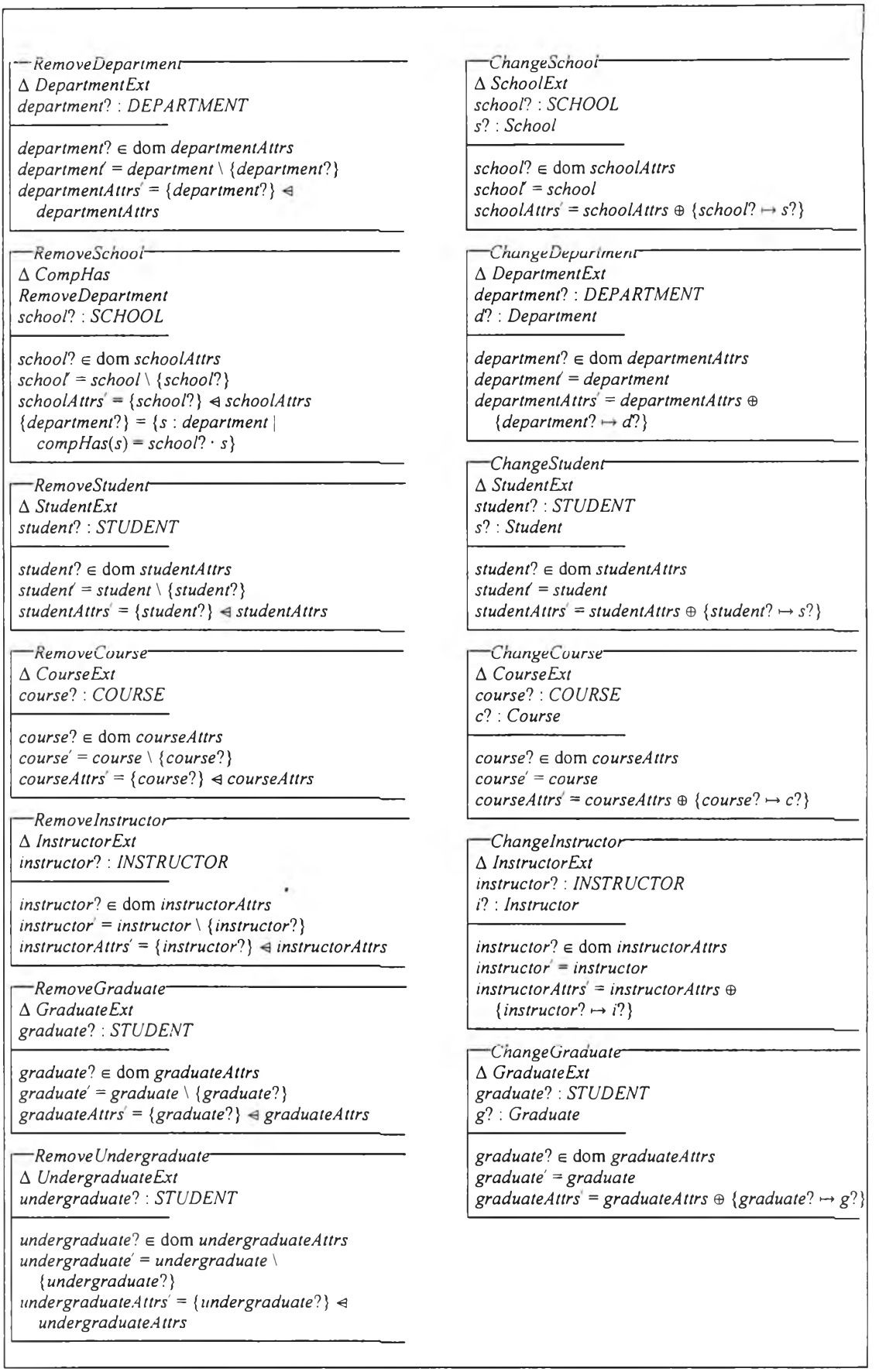


รูปที่ 5.2 ข้อกำหนดเซตของระบบสารสนเทศของสถานศึกษา

I21486203



รูปที่ 5.2 ข้อกำหนดเซตของระบบสารสนเทศของสถานศึกษา (ต่อ)



รูปที่ 5.2 ข้อกำหนดเชิงของระบบสารสนเทศของสถานศึกษา (ต่อ)

ChangeUndergraduate

Δ *UndergraduateExt*
undergraduate? : *STUDENT*
u? : *Graduate*

---

undergraduate? $\in$ dom *undergraduateAttrs*
undergraduate′ = *undergraduate*
undergraduateAttrs′ = *undergraduateAttrs* $\oplus$
{*undergraduate*? $\mapsto$ *u*?}

รูปที่ 5.2 ข้อกำหนดเซตของระบบสารสนเทศของสถานศึกษา (ต่อ)

- จำนวนเค้าร่างการเพิ่มมีทั้งหมด 7 เค้าร่างคือ AddSchool AddDepartment AddStudent AddCourse AddInstructor AddGraduate และ AddUndergraduate
- จำนวนเค้าร่างการลบมีทั้งหมด 7 เค้าร่างคือ RemoveDepartment RemoveSchool RemoveStudent RemoveCourse RemoveInstructor RemoveGraduate และ RemoveUndergraduate
- จำนวนเค้าร่างการแก้ไขมีทั้งหมด 7 เค้าร่างคือ ChangeSchool ChangeDepartment ChangeStudent ChangeCourse ChangeInstructor ChangeGraduate และ ChangeUndergraduate

3) วิเคราะห์ข้อกำหนดเซตที่ได้โดยใช้ Z/EVES โดยการสร้างเงื่อนไขก่อน การสร้างกรณีทดสอบ และการทดสอบทฤษฎีบท ซึ่งในที่นี้จะยกตัวอย่างการวิเคราะห์การดำเนินการพื้นฐาน AddStudent RemoveStudent และ ChangeStudent ดังนี้

AddStudent

Δ *StudentExt*
student? : *STUDENT*
s? : *Student*

---

student? $\notin$ dom *studnetAttrs*
student′ = *student* $\cup$ {*student*?}
studentAttrs′ = *studentAttrs* $\cup$ {*student*? $\mapsto$ *s*?}

RemoveStudent

Δ *StudentExt*
student? : *STUDENT*

---

student? $\in$ dom *studnetAttrs*
student′ = *student* $\setminus$ {*student*?}
studentAttrs′ = {*student*?} $\lhd$ *studentAttrs*

ChangeStudent

Δ *StudentExt*
student? : *STUDENT*
s? : *Student*

---

student? $\in$ dom *studnetAttrs*
student$'$ = *student*
studentAttrs$'$ = *studentAttrs* $\oplus$ {*stuent?* $\mapsto$ *s?*}

- การสร้างเงื่อนไขก่อนของ AddStudent

  Theorem preAddStudent

  $\forall$ StudentExt; student? : STUDENT; s? : Student •

  student? $\notin$ dom studentAttrs $\Rightarrow$ pre AddStudent

  ซึ่งเมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้จะมีค่าเป็น true

- การสร้างเงื่อนไขก่อนของ RemoveStudent

  Theorem preRemoveStudent

  $\forall$ StudentExt; student? : STUDENT •

  student? $\in$ dom studentAttrs $\wedge$

  dom studentAttrs \ {student?} $\in \mathbb{P}$(student \ {student?}) $\Rightarrow$

  pre RemoveStudent

  ซึ่งเมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้จะมีค่าเป็น true

- การสร้างเงื่อนไขก่อนของ ChangeStudent

  Theorem preChangeStudent

  $\forall$ StudentExt; student? : STUDENT; s? : Student •

  student? $\in$ dom studentAttrs $\Rightarrow$ pre ChangeStudent

  ซึ่งเมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้จะมีค่าเป็น true

เงื่อนไขก่อนทั้งหมดในระบบสารสนเทศของสถานศึกษา แสดงในตารางที่ 5.1

ตารางที่ 5.1 เงื่อนไขก่อนในระบบสารสนเทศของสถานศึกษา

การดำเนินการ	เงื่อนไขก่อน
AddSchool	school? ∉ dom schoolAttrs
AddDepartment	department? ∉ dom departmentAttrs
AddStudent	student? ∉ dom studentAttrs
AddCourse	course? ∉ dom courseAttrs
AddInstructor	instructor? ∉ dom instructorAttrs
AddGraduate	graduate? ∉ dom graduateAttrs ∧ graduate? ∉ graduate ∧ graduate? ∈ student
AddUndergraduate	undergraduate? ∉ dom undergraduateAttrs ∧ undergraduate? ∉ undergraduate ∧ undergraduate? ∈ student
RemoveDepartment	department? ∈ dom departmentAttrs ∧ dom departmentAttrs \ {department?} ∈ ℙ(department \ {department?})
RemoveSchool	school? ∈ dom schoolAttrs ∧ school? ∈ school
RemoveStudent	student? ∈ dom studentAttrs ∧ dom studentAttrs \ {student?} ∈ ℙ(student \ {student?})
RemoveCourse	course? ∈ dom courseAttrs ∧ dom courseAttrs \ {course?} ∈ ℙ(course \ {course?})
RemoveInstructor	instructor? ∈ dom instructorAttrs ∧ dom instructorAttrs \ {instructor?} ∈ ℙ(instructor \ {instructor?})
RemoveGraduate	graduate? ∈ \dom graduateAttrs ∧ graduate? ∈ graduate ∧ graduate? ∉ student
RemoveUndergraduate	undergraduate? ∈ dom undergraduateAttrs ∧ undergraduate? ∈ undergraduate ∧ undergraduate? ∉ student
ChangeSchool	school? ∈ dom schoolAttrs
ChangeDepartment	Department? ∈ dom departmentAttrs

ตารางที่ 5.1 เงื่อนไขก่อนในระบบสารสนเทศของสถานศึกษา (ต่อ)

การดำเนินการ	เงื่อนไขก่อน
ChangeStudent	student? ∈ dom studentAttrs
ChangeCourse	course? ∈ dom courseAttrs
ChangeInstructor	instructor? ∈ dom instructorAttrs
ChangeGraduate	graduate? ∈ dom graduateAttrs ∧ graduate? ∈ graduate ∧ graduate? ∉ student
ChangeUndergraduate	undergraduate? ∈ dom undergraduateAttrs ∧ undergraduate? ∈ undergraduate ∧ undergraduate? ∉ student

- การสร้างกรณีทดสอบของ AddStudent ซึ่งต้องมีการกำหนดสถานะเริ่มต้น (Initial state) ของเซต student และเซต studentAttrs ให้เป็นเซตว่าง ดังนี้

InitStudent

$StudentExt'$

$student' = \emptyset$

$studentAttrs' = \emptyset$

กรณีทดสอบที่ 1

TestAdd2Students ≙ InitStudent ⨾ AddStudent ⨾
AddStudent[student2?/student?, s2?/s?]

จากข้างต้นเป็นการใช้สัญกรณ์ ⨾ (Schema composition) เพื่อเป็นการระบุถึงลำดับของการดำเนินการ จากนั้นใช้คำสั่ง try ดังนี้

try TestAdd2Students[student? := OID1, s? := john,
student2? := OID2, s2? := jack];

ซึ่งเมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้จะเป็นการเพิ่มสมาชิกในเซตของ student และ studentAttrs ดังนี้

student' = {OID1} ∪ {OID2} ∧

studentAttrs' = {(OID1, john)} ∪ {(OID2, jack)} ∧

¬ OID2 = OID1

try TestAdd2Students[student? := OID1, s? := john,

student2? := OID1, s2? := jack];

เมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้จะมีค่าเป็น false เนื่องจากเงื่อนไขก่อนของการเพิ่มสมาชิกในเซตคือ student? ∉ dom studentAttrs

- การสร้างกรณีทดสอบของ RemoveStudent

กรณีทดสอบที่ 2

TestAddThenRemoveStudent ≙ InitStudent ⨾ AddStudent ⨾

RemoveStudent[student2?/student?]

try TestAddThenRemoveStudent[student? := OID1, s? := john,

student2? := OID1];

ซึ่งเมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้คือ เซตว่างของเซต student และ studentAttrs ดังนี้

student' = ∅ ∧ studentAttrs' = ∅

- การสร้างกรณีทดสอบของ ChangeStudent

กรณีทดสอบที่ 3

TestChangeStudent ≙ InitStudent ⨾ AddStudent ⨾

ChangeStudent[student2?/student?, s2?/s?]

try TestChangeStudent[student? := OID1, s? := john,

student2? := OID1, s2? := jack];

ซึ่งเมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้คือ เป็นการลบสมาชิกในเซตของ studentAttrs ที่มีสมาชิกลำดับที่หนึ่งของคู่ลำดับใน studentAttrs มีค่าเท่ากับตัวแปรเข้า student2? แล้วเพิ่มสมาชิกในเซต studentAttrs เป็นค่าของคู่ลำดับที่ต้องการแก้ไขค่า ดังนี้

student' = {OID1} ∧ studentAttrs' = {(OID1, john)} ⊕ {(OID1, jack)}

นั่นก็คือ

student' = {OID1} ∧ studentAttrs' = {(OID1, jack)}

- การทดสอบทฤษฎีบทของ AddStudent

  Theorem TwoStudentsOfAddingAreNotSame

  ∀ TestAdd2Students • student2? ≠ student?

  เมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้จะมีค่าเป็น true ทั้งนี้เป็นผลมาจากเงื่อนไขก่อนของ AddStudent คือ student? ∉ dom studentAttrs นั่นคือตัวแปรเข้า student? และ student2? ต้องไม่เท่ากัน

- การทดสอบทฤษฎีบทของ RemoveStudent

  Theorem TwoStudentsOfRemovingAreSame

  ∀ TestAddThenRemoveStudent • student2? = student?

  เมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้จะมีค่าเป็น true ทั้งนี้เป็นผลมาจากเงื่อนไขก่อนของ RemoveStudent คือ student? ∈ dom studentAttrs ∧ dom studentAttrs \ {student?} ∈ ℙ(student \ {student?}) นั่นคือตัวแปรเข้า student? และ student2? ต้องเท่ากัน

- การทดสอบทฤษฎีบทของ ChangeStudent

  Theorem TwoStudentsOfChangingAreSame

  ∀ TestChangeStudent • student2? = student?

  เมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้จะมีค่าเป็น true ทั้งนี้เป็นผลมาจากเงื่อนไขก่อนของ ChangeStudent คือ student? ∈ dom studentAttrs นั่นคือตัวแปรเข้า student? และ student2? ต้องเท่ากัน

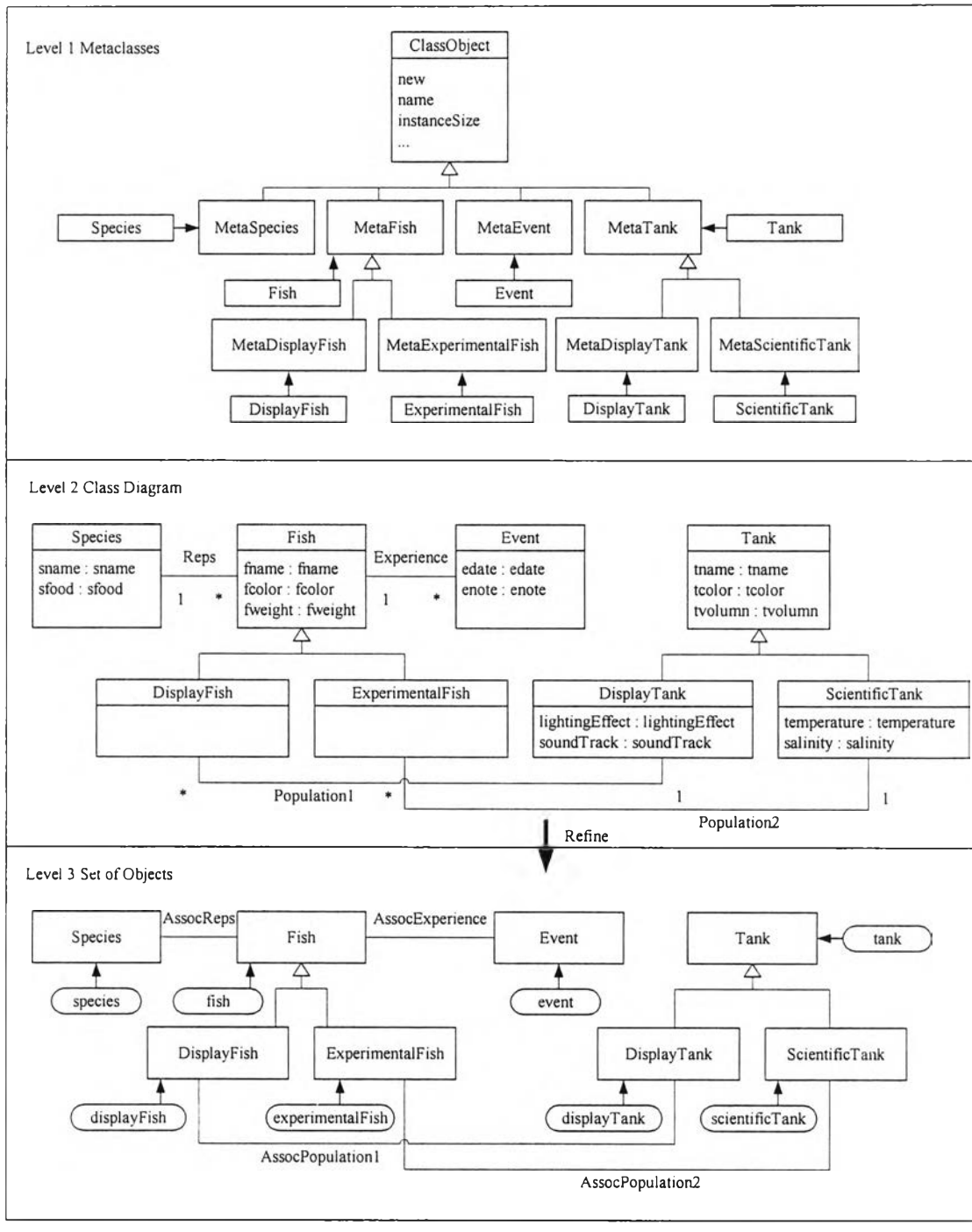
5.3.2 ระบบฐานข้อมูลถังเลี้ยงปลา

ระบบฐานข้อมูลถังเลี้ยงปลา เป็นระบบที่ใช้ในการจัดการเกี่ยวกับการเลี้ยง การขยายพันธุ์ และการจัดแสดงปลา โดยจะแบ่งปลาออกเป็น 2 ชนิดคือ ปลาสวยงามและปลาทดลอง ดังในรูปที่ 5.3 ในระดับที่ 2 แสดงแผนภาพคลาสของระบบฐานข้อมูลถังเลี้ยงปลา

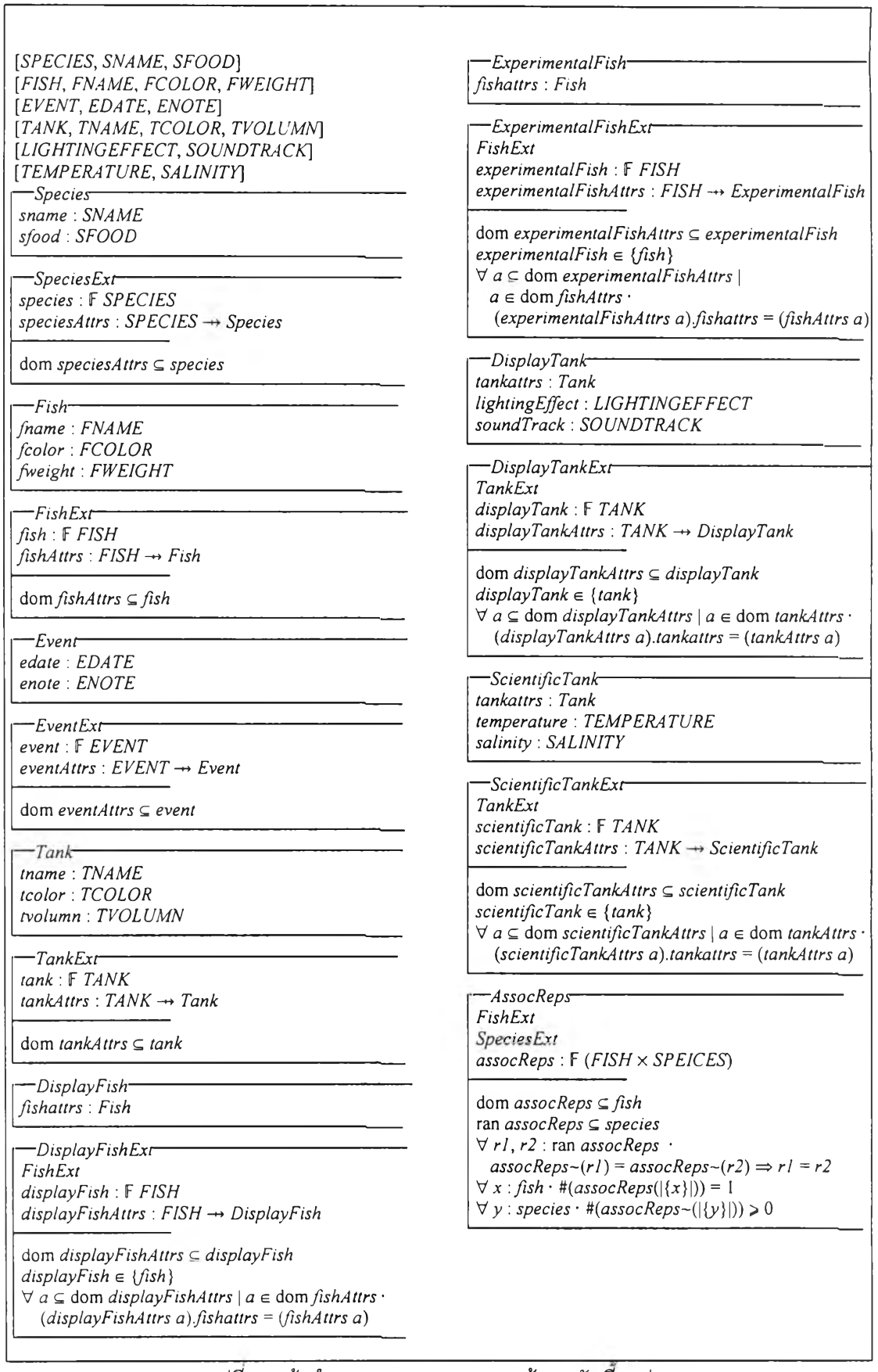
จากรูปที่ 5.3 สามารถสร้างเป็นข้อกำหนดเซตของระบบฐานข้อมูลถังเลี้ยงปลาได้ดังรูปที่ 5.4 โดยข้อมูลนำเข้าและข้อกำหนดเซตที่ได้จากเครื่องมือซอฟต์แวร์ ที่อยู่ในรูปแทคของลาเท็กซ์แสดงในภาคผนวก ค

การตรวจสอบผลลัพธ์ของข้อกำหนดเซตที่สร้างขึ้น โดยตรวจสอบวากยสัมพันธ์ของข้อกำหนดเซตที่ได้ และวิเคราะห์ข้อกำหนดเซตที่สร้างจากระบบฐานข้อมูลถังเลี้ยงปลา ได้ดังนี้

1) ตรวจสอบวากยสัมพันธ์ของข้อกำหนดเซตที่ได้โดยใช้ Z/EVES พบว่าข้อกำหนดที่ได้นั้นมีความถูกต้องตามวากยสัมพันธ์ของข้อกำหนดเซต
2) ระบบฐานข้อมูลถังเลี้ยงปลามีจำนวนคลาสทั้งหมด 8 คลาส และมีความสัมพันธ์ทั้งหมด 4 ความสัมพันธ์ เมื่อตรวจสอบเค้าร่างของข้อกำหนดเซตที่สร้างขึ้น พบว่าเค้าร่างทุกเค้าร่างในระบบฐานข้อมูลถังเลี้ยงปลา เป็นไปตามกฎการแปลงแผนภาพคลาสเป็นข้อกำหนดเซตในบทที่ 3 และเมื่อนับจำนวนเค้าร่างทั้งหมดในระบบได้ผลดังนี้
   - จำนวนเค้าร่างคลาสมีทั้งหมด 8 เค้าร่างคือ Species Fish Event Tank DisplayFish ExperimentalFish DisplayTank และ ScientificTank
   - จำนวนเค้าร่างเพิ่มเติมของคลาสมีทั้งหมด 8 เค้าร่างคือ SpeciesExt FishExt EventExt TankExt DisplayFishExt ExperimentalFishExt DisplayTankExt และ ScientificTankExt
   - จำนวนเค้าร่างความสัมพันธ์มีทั้งหมด 4 เค้าร่างคือ AssocReps AssocExperience AssocPopulation1 และ AssocPopulation2
   - จำนวนเค้าร่างการเพิ่มมีทั้งหมด 8 เค้าร่างคือ AddSpecies AddFish AddEvent AddTank AddDisplayFish AddExperimentalFish AddDisplayTank และ AddScientificTank
   - จำนวนเค้าร่างการลบมีทั้งหมด 8 เค้าร่างคือ RemoveSpecies RemoveFish RemoveEvent RemoveTank RemoveDisplayFish RemoveExperimentalFish RemoveDisplayTank และ RemoveScientificTank
   - จำนวนเค้าร่างการแก้ไขมีทั้งหมด 8 เค้าร่างคือ ChangeSpecies ChangeFish ChangeEvent ChangeTank ChangeDisplayFish ChangeExperimentalFish ChangeDisplayTank และ ChangeScientificTank



รูปที่ 5.3 ระดับของการสร้างข้อกำหนดเซตของระบบฐานข้อมูลถังเลี้ยงปลา



รูปที่ 5.4 ข้อกำหนดเซตของระบบฐานข้อมูลถังเลี้ยงปลา

AssocExperience

$EventExt$
$FishExt$
$assocExperience : \mathbb{F}\ (EVENT \times FISH)$

$\text{dom}\ assocExperience \subseteq event$
$\text{ran}\ assocExperience \subseteq fish$
$\forall\ r1, r2 : \text{ran}\ assocExperience \cdot$
$\quad assocExperience\sim(r1) = assocExperience\sim(r2) \Rightarrow$
$\quad r1 = r2$
$\forall\ x : event \cdot \#(assocExperience(\lvert\{x\}\rvert)) = 1$
$\forall\ y : fish \cdot \#(assocExperience\sim(\lvert\{y\}\rvert)) \geq 0$

AssocPopulation1

$DisplayFishExt$
$DisplayTankExt$
$assocPopulation1 : \mathbb{F}\ (FISH \times TANK)$

$\text{dom}\ assocPopulation1 \subseteq displayFish$
$\text{ran}\ assocPopulation1 \subseteq displayTank$
$\forall\ r1, r2 : \text{ran}\ assocPopulation1 \cdot$
$\quad assocPopulation1\sim(r1) = assocPopulation1\sim(r2)$
$\quad \Rightarrow r1 = r2$
$\forall\ x : displayFish \cdot \#(assocPopulation1(\lvert\{x\}\rvert)) = 1$
$\forall\ y : displayTank \cdot \#(assocPopulation1\sim(\lvert\{y\}\rvert)) \geq 0$

AssocPopulation2

$ExperimentalFishExt$
$ScientificTankExt$
$assocPopulation2 : \mathbb{F}\ (FISH \times TANK)$

$\text{dom}\ assocPopulation2 \subseteq experimentalFish$
$\text{ran}\ assocPopulation2 \subseteq scientificTank$
$\forall\ r1, r2 : \text{ran}\ assocPopulation2 \cdot$
$\quad assocPopulation2\sim(r1) = assocPopulation2\sim(r2)$
$\quad \Rightarrow r1 = r2$
$\forall\ x : experimentalFish \cdot$
$\quad \#(assocPopulation2(\lvert\{x\}\rvert)) = 1$
$\forall\ y : scientificTank \cdot$
$\quad \#(assocPopulation2\sim(\lvert\{y\}\rvert)) \geq 0$

AddSpecies

$\Delta\ SpeciesExt$
$species? : SPECIES$
$s? : Species$

$species? \notin \text{dom}\ speciesAttrs$
$species' = species \cup \{species?\}$
$speciesAttrs' = speciesAttrs \cup \{species? \mapsto s?\}$

AddFish

$\Delta\ FishExt$
$fish? : FISH$
$f? : Fish$

$fish? \notin \text{dom}\ fishAttrs$
$fish' = fish \cup \{fish?\}$
$fishAttrs' = fishAttrs \cup \{fish? \mapsto f?\}$

AddEvent

$\Delta\ EventExt$
$event? : EVENT$
$e? : Event$

$event? \notin \text{dom}\ eventAttrs$
$event' = event \cup \{event?\}$
$eventAttrs' = eventAttrs \cup \{event? \mapsto e?\}$

AddTank

$\Delta\ TankExt$
$tank? : TANK$
$t? : Tank$

$tank? \notin \text{dom}\ tankAttrs$
$tank' = tank \cup \{tank?\}$
$tankAttrs' = tankAttrs \cup \{tank? \mapsto t?\}$

AddDisplayFish

$\Delta\ DisplayFishExt$
$displayFish? : FISH$
$d? : DisplayFish$

$displayFish? \notin \text{dom}\ displayFishAttrs$
$displayFish' = displayFish \cup \{displayFish?\}$
$displayFishAttrs' = displayFishAttrs \cup$
$\quad \{displayFish? \mapsto d?\}$

AddExperimentalFish

$\Delta\ ExperimentalFishExt$
$experimentalFish? : FISH$
$e? : ExperimentalFish$

$experimentalFish? \notin \text{dom}\ experimentalFishAttrs$
$experimentalFish' = experimentalFish \cup$
$\quad \{experimentalFish?\}$
$experimentalFishAttrs' = experimentalFishAttrs \cup$
$\quad \{experimentalFish? \mapsto e?\}$

AddDisplayTank

$\Delta\ DisplayTankExt$
$displayTank? : TANK$
$d? : DisplayTank$

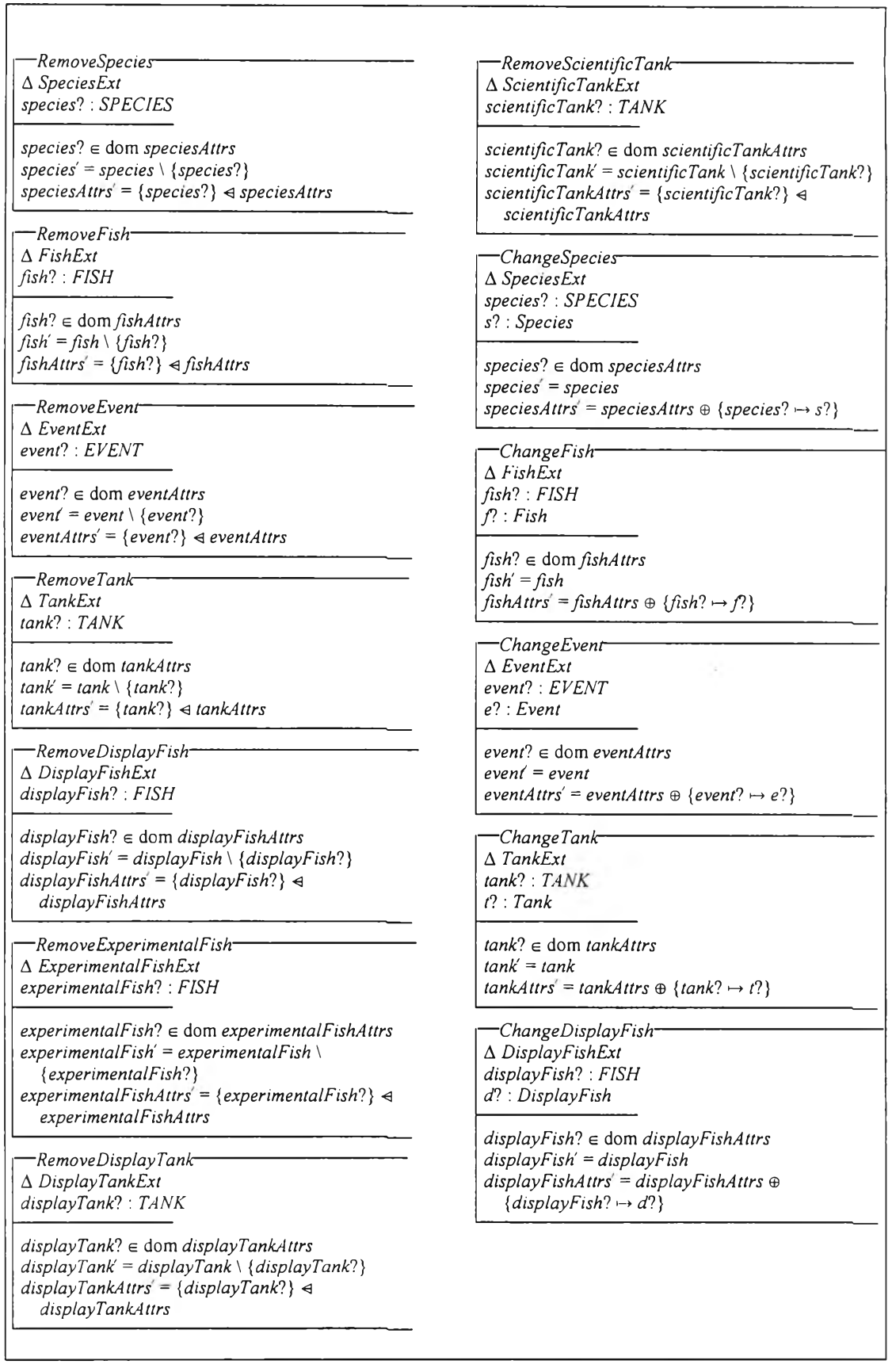
$displayTank? \notin \text{dom}\ displayTankAttrs$
$displayTank' = displayTank \cup \{displayTank?\}$
$displayTankAttrs' = displayTankAttrs \cup$
$\quad \{displayTank? \mapsto d?\}$

AddScientificTank

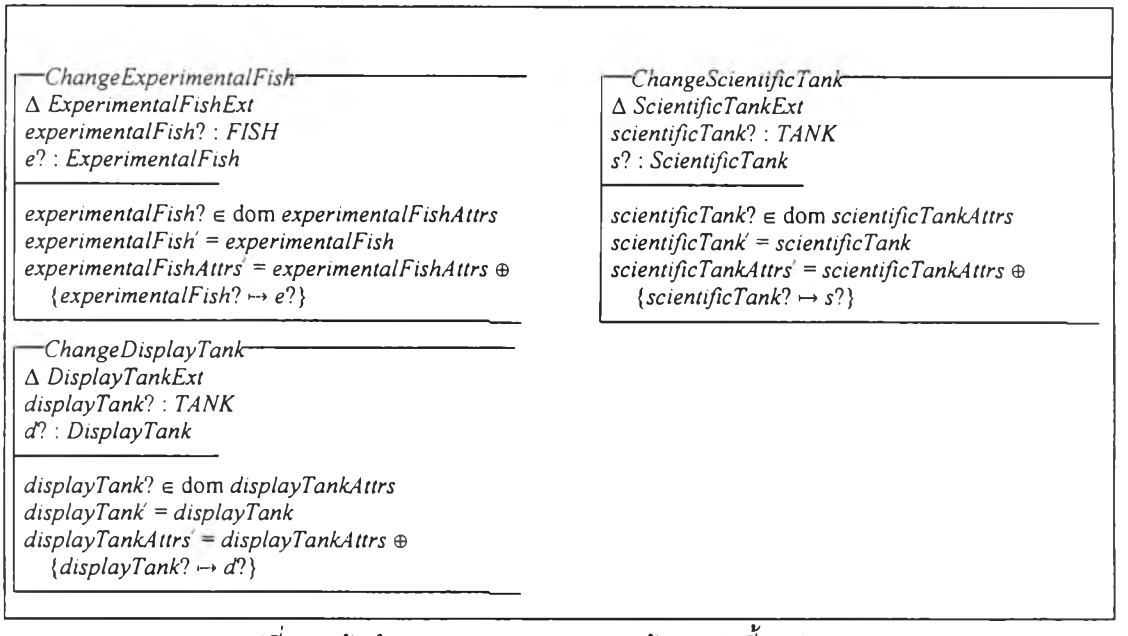
$\Delta\ ScientificTankExt$
$scientificTank? : TANK$
$s? : ScientificTank$

$scientificTank? \notin \text{dom}\ scientificTankAttrs$
$scientificTank' = scientificTank \cup \{scientificTank?\}$
$scientificTankAttrs' = scientificTankAttrs \cup$
$\quad \{scientificTank? \mapsto s?\}$

รูปที่ 5.4 ข้อกำหนดเซตของระบบฐานข้อมูลถังเลี้ยงปลา (ต่อ)

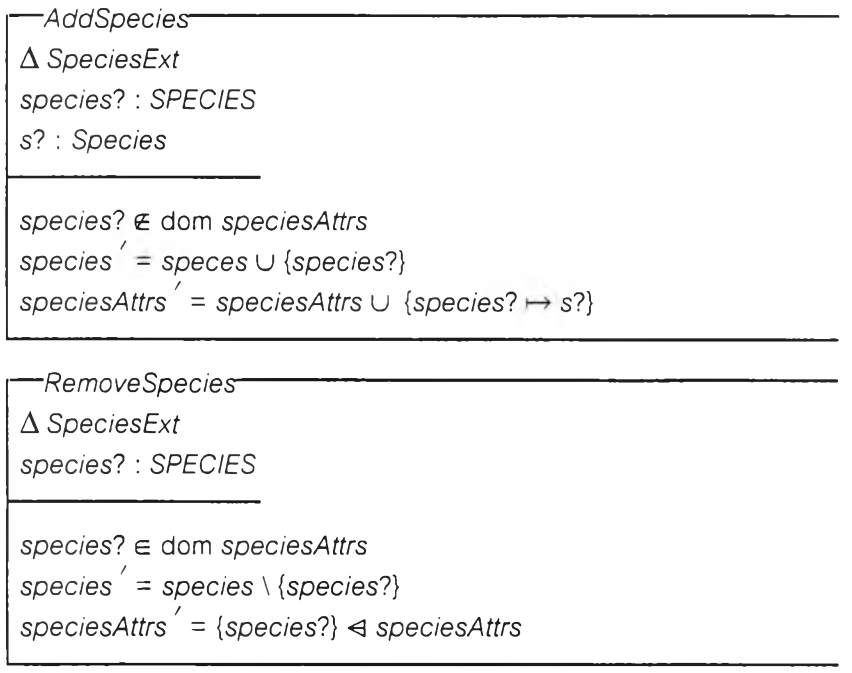


รูปที่ 5.4 ข้อกำหนดเซตของระบบฐานข้อมูลถังเลี้ยงปลา (ต่อ)



รูปที่ 5.4 ข้อกำหนดเซตของระบบฐานข้อมูลถังเลี้ยงปลา (ต่อ)

3) วิเคราะห์ข้อกำหนดเซตที่ได้โดยใช้ Z/EVES โดยการสร้างเงื่อนไขก่อน การสร้างกรณีทดสอบ และการทดสอบทฤษฎีบท ซึ่งในที่นี้จะยกตัวอย่างการวิเคราะห์การดำเนินการพื้นฐาน AddSpecies RemoveSpeceis และ ChangeSpecies ดังนี้



ChangeSpecies

Δ *SpeciesExt*
species? : *SPECIES*
s? : *Species*

---

species? $\in$ dom *speciesAttrs*
species$'$ = *species*
speciesAttrs$'$ = *speciesAttrs* $\oplus$ {*speices*? $\mapsto$ *s*?}

- การสร้างเงื่อนไขก่อนของ AddSpecies

  Theorem preAddSpecies

  $\forall$ SpeciesExt; species? : SPECIES; s? : Species •

  species? $\notin$ dom speciesAttrs $\Rightarrow$ pre AddSpecies

  ซึ่งเมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้จะมีค่าเป็น true

- การสร้างเงื่อนไขก่อนของ RemoveSpecies

  Theorem preRemoveSpecies

  $\forall$ SpeciesExt; species? : SPECIES •

  species? $\in$ dom speicesAttrs $\wedge$

  dom speciesAttrs \ {species?} $\in \mathbb{P}$(species \ {species?}) $\Rightarrow$

  pre RemoveSpecies

  ซึ่งเมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้จะมีค่าเป็น true

- การสร้างเงื่อนไขก่อนของ ChangeSpecies

  Theorem preChangeSpecies

  $\forall$ SpeciesExt; species? : SPECIES; s? : Species •

  species? $\in$ dom speciesAttrs $\Rightarrow$ pre ChangeSpeces

  ซึ่งเมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้จะมีค่าเป็น true

  เงื่อนไขก่อนทั้งหมดในระบบฐานข้อมูลถังเลี้ยงปลา แสดงในตารางที่ 5.2

ตารางที่ 5.2 เงื่อนไขก่อนในระบบฐานข้อมูลถังเลี้ยงปลา

การดำเนินการ	เงื่อนไขก่อน
AddSpecies	species? ∉ dom speciesAttrs
AddFish	fish? ∉ dom fishAttrs
AddEvent	event? ∉ dom eventAttrs
AddTank	tank? ∉ dom tankAttrs
AddDisplayFish	displayFish? ∉ dom displayFishAttrs ∧ displayFish? ∉ displayFish ∧ displayFish? ∈ fish
AddExperimentalFish	experimentalFish? ∉ dom experimentalFishAttrs ∧ experimentalFish? ∉ experimentalFish ∧ experimentalFish? ∈ fish
AddDisplayTank	displayTank? ∉ dom displayTankAttrs ∧ displayTank? ∉ displayTank ∧ displayTank? ∈ tank
AddScientificTank	scientificTank? ∉ dom scientificTankAttrs ∧ scientificTank? ∉ scientificTank ∧ scientificTank? ∈ tank
RemoveSpecies	species? ∈ dom speciesAttrs ∧ dom speciesAttrs \ {species?} ∈ ℙ(species \ {species?})
RemoveFish	fish? ∈ dom fishAttrs ∧ dom fishAttrs \ { fish?} ∈ ℙ(fish \ { fish?})
RemoveEvent	event? ∈ dom eventAttrs ∧ dom eventAttrs \ {event?} ∈ ℙ(event \ {event?})
RemoveTank	tank? ∈ dom tankAttrs ∧ dom tankAttrs \ \{tank?\} ∈ ℙ(tank \ {tank?})
RemoveDisplayFish	displayFish? ∈ dom displayFishAttrs ∧ displayFish? ∈ displayFish ∧ displayFish? ∉ fish
RemoveExperimentalFish	experimentalFish? ∈ dom experimentalFishAttrs ∧ experimentalFish? ∈ experimentalFish ∧ experimentalFish? ∉ fish

ตารางที่ 5.2 เงื่อนไขก่อนในระบบฐานข้อมูลถังเลี้ยงปลา (ต่อ)

การดำเนินการ	เงื่อนไขก่อน
RemoveDisplayTank	displayTank? ∈ dom displayTankAttrs ∧ displayTank? ∈ displayTank ∧ displayTank? ∉ tank
RemoveScientificTank	scientificTank? ∈ dom scientificTankAttrs ∧ scientificTank? ∈ scientificTank ∧ scientificTank? ∉ tank
ChangeSpecies	species? ∈ dom speciesAttrs
ChangeFish	fish? ∈ dom fishAttrs
ChangeEvent	event? ∈ dom eventAttrs
ChangeTank	tank? ∈ dom tankAttrs
ChangeDisplayFish	displayFish? ∈ dom displayFishAttrs ∧ displayFish? ∈ displayFish ∧ displayFish? ∉ fish
ChangeExperimentalFish	experimentalFish? ∈ dom experimentalFishAttrs ∧ experimentalFish? ∈ experimentalFish ∧ experimentalFish? ∉ fish
ChangeDisplayTank	displayTank? ∈ dom displayTankAttrs ∧ displayTank? ∈ displayTank ∧ displayTank? ∉ tank
ChangeScientificTank	scientificTank? ∈ dom scientificTankAttrs ∧ scientificTank? ∈ scientificTank ∧ scientificTank? ∉ tank

- การสร้างกรณีทดสอบของ AddSpecies ซึ่งต้องมีการกำหนดสถานะเริ่มต้น ของเซต species และเซต speicesAttrs ให้เป็นเซตว่าง ดังนี้

InitSpecies

$SpeciesExt'$

$species' = \emptyset$

$speciesAttrs' = \emptyset$

กรณีทดสอบที่ 4

TestAdd2Species ≙ InitSpecies ⨟ AddSpecies ⨟

AddSpecies[species2?/species?, s2?/s?]

จากนั้นใช้คำสั่ง try ดังนี้

try TestAdd2Species[species? := OID1, s? := dolphin,

speies2? := OID2, s2? := shark];

ซึ่งเมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้จะเป็นการเพิ่มสมาชิกในเซตของ species และ speciesAttrs ดังนี้

species' = {OID1} ∪ {OID2} ∧

speicesAttrs' = {(OID1, dolphin)} ∪ {(OID2, shark)} ∧

¬ OID2 = OID1

try TestAdd2Species[species? := OID1, s? := dolphin,

species2? := OID1, s2? := shark];

เมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้จะมีค่าเป็น false เนื่องจากเงื่อนไขก่อนของการเพิ่มสมาชิกในเซตคือ species? ∉ dom speciesAttrs

- การสร้างกรณีทดสอบของ RemoveSpecies

กรณีทดสอบที่ 5

TestAddThenRemoveSpecies ≙ InitSpeices ⨟ AddSpecies ⨟

RemoveSpecies[species2?/species?]

try TestAddThenRemoveSpecies[species? := OID1, s? := dolphin,

species2? := OID1];

ซึ่งเมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้คือ เซตว่างของเซต species และ speciesAttrs ดังนี้

species' = ∅ ∧ speciesAttrs' = ∅

- การสร้างกรณีทดสอบของ ChangeSpecies

กรณีทดสอบที่ 6

TestChangeSpecies $\widehat{=}$ InitSpecies ⨾ AddSpecies ⨾
ChangeSpecies[species2?/species?, s2?/s?]

try TestChangeSpecies[species? := OID1, s? := dolphin,
species2? := OID1, s2? := shark];

ซึ่งเมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้คือ เป็นการลบสมาชิกในเซตของ speciesAttrs ที่มีสมาชิกลำดับที่หนึ่งของคู่ลำดับใน speciesAttrs มีค่าเท่ากับตัวแปรเข้า species2? แล้วเพิ่มสมาชิกในเซต speciesAttrs เป็นค่าของคู่ลำดับที่ต้องการแก้ไขค่า ดังนี้

species' = {OID1} ∧ speciesAttrs' = {(OID1, dolphin)}⊕{(OID1, shark)}

นั่นก็คือ

species' = {OID1} ∧ speciesAttrs' = {(OID1, shark)}

- การทดสอบทฤษฎีบทของ AddSpecies

Theorem TwoSpeciesOfAddingAreNotSame

∀ TestAdd2Species • species2? ≠ species?

เมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้จะมีค่าเป็น true ทั้งนี้เป็นผลมาจากเงื่อนไขก่อนของ AddSpecies คือ species? ∉ dom speciesAttrs นั่นคือตัวแปรเข้า species? และ species2? ต้องไม่เท่ากัน

- การทดสอบทฤษฎีบทของ RemoveSpecies

Theorem TwoSpeciesOfRemovingAreSame

∀ TestAddThenRemoveSpecies • species2? = species?

เมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้จะมีค่าเป็น true ทั้งนี้เป็นผลมาจากเงื่อนไขก่อนของ RemoveSpecies คือ species? ∈ dom speciesAttrs ∧ dom speciesAttrs \ {species?} ∈ ℙ(species \ {species?}) นั่นคือตัวแปรเข้า species? และ species2? ต้องเท่ากัน

- การทดสอบทฤษฎีบทของ ChangeSpecies

  Theorem TwoSpeciesOfChangingAreSame

  ∀ TestChangeSpecies • species2? = species?

  เมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้จะมีค่าเป็น true ทั้งนี้เป็นผลมาจากเงื่อนไขก่อนของ ChangeSpecies คือ species? ∈ dom speciesAttrs นั่นคือ ตัวแปรเข้า species? และ species2? ต้องเท่ากัน

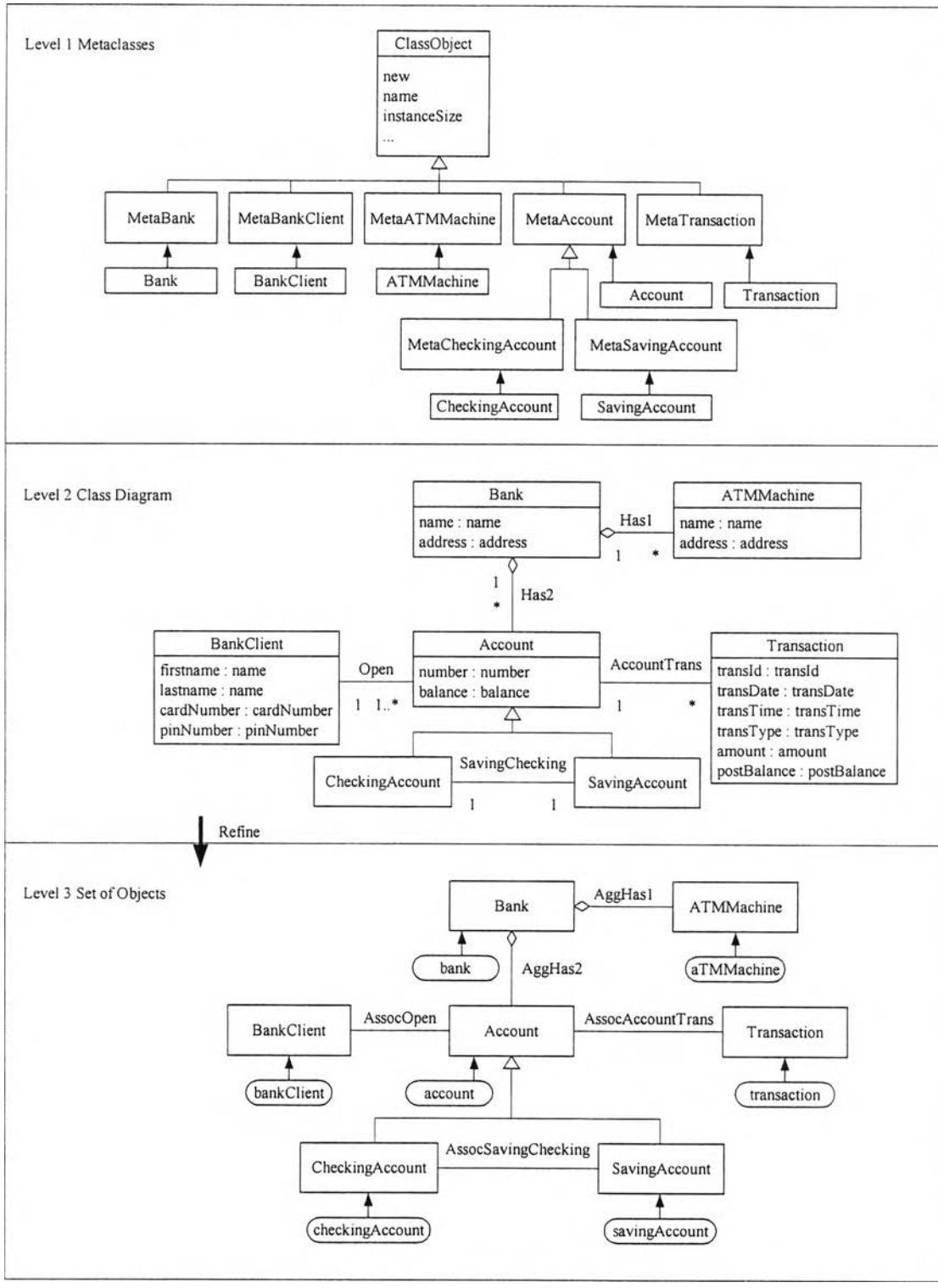
5.3.3 ระบบเครื่องฝากถอนเงินอัตโนมัติ

ระบบเครื่องฝากถอนเงินอัตโนมัติ เป็นระบบที่ใช้ในการจัดการเกี่ยวกับการทำรายการทางการเงินของธนาคาร ซึ่งลูกค้าของธนาคารจะต้องทำรายการต่าง ๆ ผ่านระบบเครื่องฝากถอนเงินอัตโนมัติ ดังในรูปที่ 5.5 ในระดับที่ 2 แสดงแผนภาพคลาสของระบบเครื่องฝากถอนเงินอัตโนมัติ

จากรูปที่ 5.5 สามารถสร้างเป็นข้อกำหนดเซตของระบบเครื่องฝากถอนเงินอัตโนมัติได้ดังรูปที่ 5.6 โดยข้อมูลนำเข้าและข้อกำหนดเซตที่ได้จากเครื่องมือซอฟต์แวร์ ที่อยู่ในรูปแทคของลาเท็กซ์แสดงในภาคผนวก ง

การตรวจสอบผลลัพธ์ของข้อกำหนดเซตที่สร้างขึ้น โดยตรวจสอบวากยสัมพันธ์ของข้อกำหนดเซตที่ได้ และวิเคราะห์ข้อกำหนดเซตที่สร้างจากระบบเครื่องฝากถอนเงินอัตโนมัติ ได้ดังนี้

1) ตรวจสอบวากยสัมพันธ์ของข้อกำหนดเซตที่ได้โดยใช้ Z/EVES พบว่าข้อกำหนดที่ได้นั้นมีความถูกต้องตามวากยสัมพันธ์ของข้อกำหนดเซต
2) ระบบเครื่องฝากถอนเงินอัตโนมัติมีจำนวนคลาสทั้งหมด 7 คลาส และมีความสัมพันธ์ทั้งหมด 5 ความสัมพันธ์ เมื่อตรวจสอบเค้าร่างของข้อกำหนดเซตที่สร้างขึ้น พบว่าเค้าร่างทุกเค้าร่างในระบบเครื่องฝากถอนเงินอัตโนมัติ เป็นไปตามกฎการแปลงแผนภาพคลาสเป็นข้อกำหนดเซตในบทที่ 3 และเมื่อนับจำนวนเค้าร่างทั้งหมดในระบบได้ผลดังนี้
   - จำนวนเค้าร่างคลาสมีทั้งหมด 7 เค้าร่างคือ Bank ATMMachine BankClient Account Transaction CheckingAccount และ SavingAccount
   - จำนวนเค้าร่างเพิ่มเติมของคลาสมีทั้งหมด 7 เค้าร่างคือ BankExt ATMMachineExt BankClientExt AccountExt TransactionExt CheckingAccountExt และ SavingAccountExt
   - จำนวนเค้าร่างความสัมพันธ์มีทั้งหมด 5 เค้าร่างคือ AggHas1 AggHas2 AssocOpen AssocAccountTrans และ AssocSavingChecking



รูปที่ 5.5 ระดับของการสร้างข้อกำหนดเชิงของระบบเครื่องฝากถอนเงินอัตโนมัติ

รูปที่ 5.6 ข้อกำหนดเซตของระบบเครื่องฝากถอนเงินอัตโนมัติ

AssocOpen

BankClientExt; *AccountExt*
assocOpen : $\mathbb{F}$ (*BANKCLIENT* × *ACCOUNT*)

---

dom *assocOpen* ⊆ *bankClient*
ran *assocOpen* ⊆ *account*
∀ *r1*, *r2* : dom *assocOpen* ·
  assocOpen(*r1*) = *assocOpen*(*r2*) ⇒ *r1* = *r2*
∀ *x* : *bankClient* · #(*assocOpen*(|{*x*}|)) ⩾ 1
∀ *y* : *account* · #(*assocOpen*~(|{*y*}|)) = 1

AssocAccountTrans

AccountExt
TransactionExt
assocAccountTrans :
  $\mathbb{F}$ (*ACCOUNT* × *TRANSACTION*)

---

dom *assocAccountTrans* ⊆ *account*
ran *assocAccountTrans* ⊆ *transaction*
∀ *r1*, *r2* : dom *assocAccountTrans* ·
  assocAccountTrans(*r1*) = *assocAccountTrans*(*r2*)
  ⇒ *r1* = *r2*
∀ *x* : *account* · #(*assocAccountTrans*(|{*x*}|)) ⩾ 0
∀ *y* : *transaction* · #(*assocAccountTrans*~(|{*y*}|))= 1

AssocSavingChecking

CheckingAccountExt
SavingAccountExt
assocSavingChecking :
  $\mathbb{F}$ (*ACCOUNT* × *ACCOUNT*)

---

dom *assocSavingChecking* ⊆ *checkingAccount*
ran *assocSavingChecking* ⊆ *savingAccount*
∀ *d1*, *d2* : dom *assocSavingChecking*;
r1, *r2* : ran *assocSavingChecking*; ·
  assocSavingChecking(*d1*) =
  assocSavingChecking (*d2*) ⇒ *d1* = *d2* ∧
  assocSavingChecking~(*r1*) =
  assocSavingChecking~(*r2*) ⇒ *r1* = *r2*
∀ *x* : *checkingAccount* ·
  #(*assocSavingChecking*(|{*x*}|)) = 1
∀ *y* : *savingAccount* ·
  #(*assocSavingChecking*~(|{*y*}|)) = 1

AddBank

Δ *BankExt*
bank? : *BANK*
b? : *Bank*

---

bank? ∉ dom *bankAttrs*
bank′ = *bank* ∪ {*bank*?}
bankAttrs′ = *bankAttrs* ∪ {*bank*? ↦ *b*?}

AddATMMachine

Δ *ATMMachineExt*
aTMMachine? : *ATMMACHINE*
a? : *ATMMachine*

---

aTMMachine? ∉ dom *aTMMachineAttrs*
aTMMachine′ = *aTMMachine* ∪ {*aTMMachine*?}
aTMMachineAttrs′ = *aTMMachineAttrs* ∪
  {*aTMMachine*? ↦ *a*?}

AddBankClient

Δ *BankClientExt*
bankClient? : *BANKCLIENT*
b? : *BankClient*

---

bankClient? ∉ dom *bankClientAttrs*
bankClient′ = *bankClient* ∪ {*bankClient*?}
bankClientAttrs′ = *bankClientAttrs* ∪
  {*bankClient*? ↦ *b*?}

AddAccount

Δ *AccountExt*
account? : *ACCOUNT*
a? : *Account*

---

account? ∉ dom *accountAttrs*
account′ = *account* ∪ {*account*?}
accountAttrs′ = *accountAttrs* ∪ {*account*? ↦ *a*?}

AddTransaction

Δ *TransactionExt*
transaction? : *TRANSACTION*
t? : *Transaction*

---

transaction? ∉ dom *transactionAttrs*
transaction′ = *transaction* ∪ {*transaction*?}
transactionAttrs′ = *transactionAttrs* ∪
  {*transaction*? ↦ *t*?}

AddCheckingAccount

Δ *CheckingAccountExt*
checkingAccount? : *ACCOUNT*
c? : *CheckingAccount*

---

checkingAccount? ∉ dom *checkingAccountAttrs*
checkingAccount′ = *checkingAccount* ∪
  {*checkingAccount*?}
checkingAccountAttrs′ = *checkingAccountAttrs* ∪
  {*checkingAccount*? ↦ *c*?}

AddSavingAccount

Δ *SavingAccountExt*
savingAccount? : *ACCOUNT*
s? : *SavingAccount*

---

savingAccount? ∉ dom *savingAccountAttrs*
savingAccount′ = *savingAccount* ∪
  {*savingAccount*?}
savingAccountAttrs′ = *savingAccountAttrs* ∪
  {*savingAccount*? ↦ *s*?}

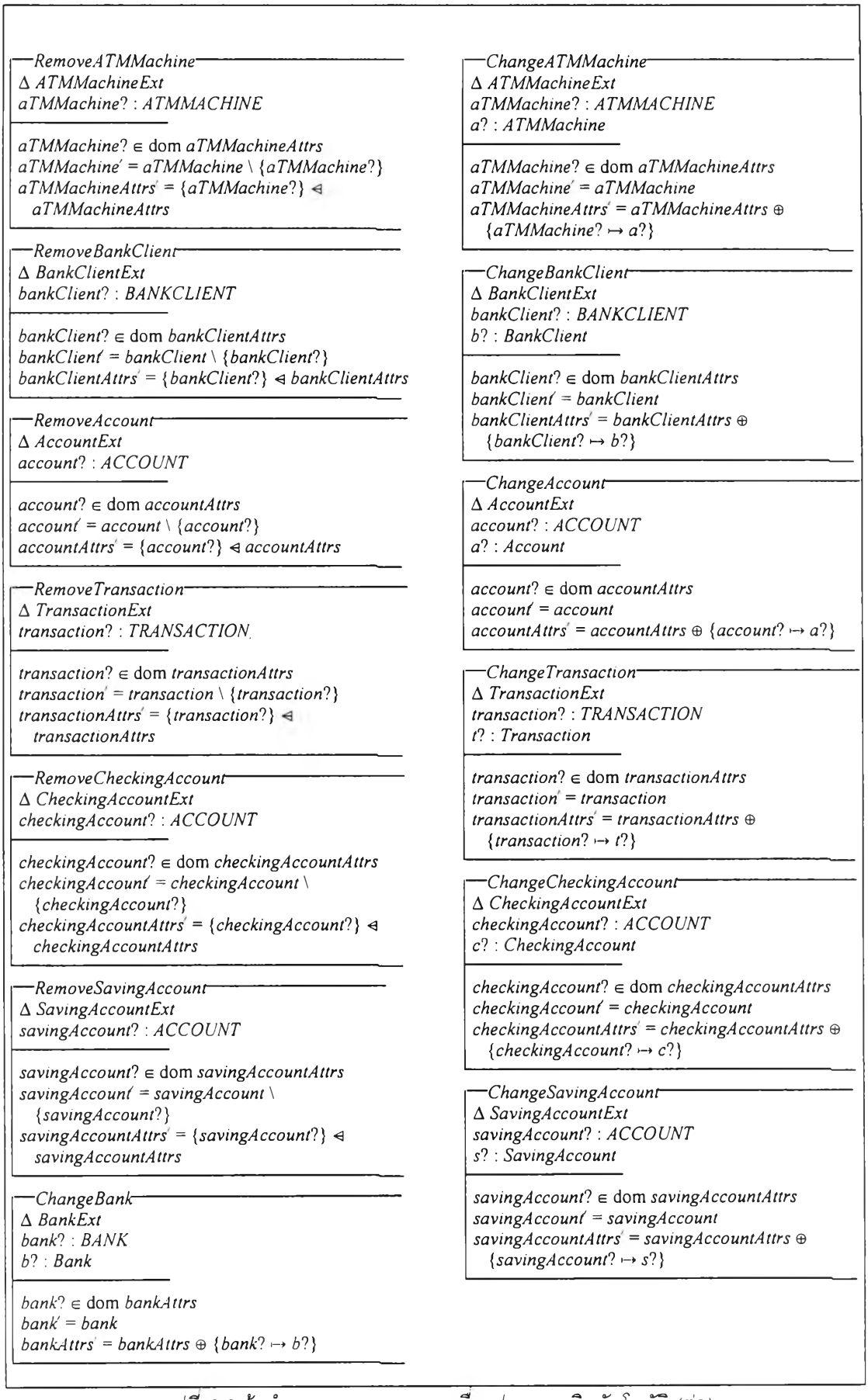
RemoveBank

Δ *BankExt*
bank? : *BANK*

---

bank? ∈ dom *bankAttrs*
bank′ = *bank* \ {*bank*?}
bankAttrs′ = {*bank*?} ⩤ *bankAttrs*

รูปที่ 5.6 ข้อกำหนดเซตของระบบเครื่องฝากถอนเงินอัตโนมัติ (ต่อ)



รูปที่ 5.6 ข้อกำหนดเซตของระบบเครื่องฝากถอนเงินอัตโนมัติ (ต่อ)

- จำนวนเค้าร่างการเพิ่มมีทั้งหมด 7 เค้าร่างคือ AddBank AddATMMachine AddBankClient AddAccount AddTransaction AddCheckingAccount และ AddSavingAccount
- จำนวนเค้าร่างการลบมีทั้งหมด 7 เค้าร่างคือ RemoveBank RemoveATMMachine RemoveBankClient RemoveAccount RemoveTransaction RemoveCheckingAccount และ RemoveSavingAccount
- จำนวนเค้าร่างการแก้ไขมีทั้งหมด 7 เค้าร่างคือ ChangeBank ChangeATMMachine ChangeBankClient ChangeAccount ChangeTransaction ChangeCheckingAccount และ ChangeSavingAccount

3) วิเคราะห์ข้อกำหนดเซตที่ได้โดยใช้ Z/EVES โดยการสร้างเงื่อนไขก่อน การสร้างกรณีทดสอบ และการทดสอบทฤษฎีบท ซึ่งในที่นี้จะยกตัวอย่างการวิเคราะห์การดำเนินการพื้นฐาน AddAccount RemoveAccount และ ChangeAccount ดังนี้

AddAccount

Δ *AccountExt*
account? : *ACCOUNT*
a? : *Account*

---

account? $\notin$ dom *accountAttrs*
account$'$ = *account* $\cup$ {*account*?}
accountAttrs$'$ = *accountAttrs* $\cup$ {*account*? $\mapsto$ *a*?}

RemoveAccount

Δ *AccountExt*
account? : *ACCOUNT*

---

account? $\in$ dom *accountAttrs*
account$'$ = *account* $\setminus$ {*account*?}
accountAttrs$'$ = {*account*?} $\lhd$ *accountAttrs*

ChangeAccount

Δ *AccountExt*
account? : *ACCOUNT*
a? : *Account*

---

account? $\in$ dom *accountAttrs*
account$'$ = *account*
accountAttrs$'$ = *accountAttrs* $\oplus$ {*account*? $\mapsto$ *a*?}

- การสร้างเงื่อนไขก่อนของ AddAccount

  Theorem preAddAccount

  ∀ AccountExt; account? : ACCOUNT; a? : Account •

  account? ∉ dom accountAttrs ⇒ pre AddAccount

  ซึ่งเมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้จะมีค่าเป็น true

- การสร้างเงื่อนไขก่อนของ RemoveAccount

  Theorem preRemoveAccount

  ∀ AccountExt; account? : ACCOUNT •

  account? ∈ dom accountAttrs ∧

  dom accountAttrs \ {account?} ∈ ℙ(account \ {account?}) ⇒

  pre RemoveAccount

  ซึ่งเมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้จะมีค่าเป็น true

- การสร้างเงื่อนไขก่อนของ ChangeAccount

  Theorem preChangeAccount

  ∀ AccountExt; account? : ACCOUNT; a? : Account •

  account? ∈ dom accountAttrs ⇒ pre ChangeAccount

  ซึ่งเมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้จะมีค่าเป็น true

เงื่อนไขก่อนทั้งหมดในระบบเครื่องฝากถอนเงินอัตโนมัติ แสดงในตารางที่ 5.3

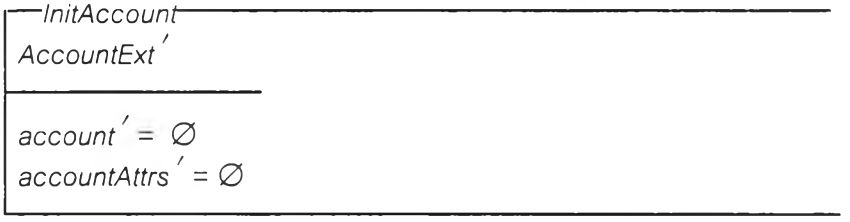
ตารางที่ 5.3 เงื่อนไขก่อนในระบบเครื่องฝากถอนเงินอัตโนมัติ

การดำเนินการ	เงื่อนไขก่อน
AddBank	bank? ∉ dom bankAttrs
AddATMMachine	aTMMachine? ∉ dom aTMMachineAttrs
AddBankClient	bankClient? ∉ dom bankClientAttrs
AddAccount	account? ∉ dom accountAttrs
AddTransaction	transaction? ∉ dom transactionAttrs
AddCheckingAccount	checkingAccount? ∉ dom checkingAccountAttrs ∧ checkingAccount? ∉ checkingAccount ∧ checkingAccount? ∈ account

ตารางที่ 5.3 เงื่อนไขก่อนในระบบเครื่องฝากถอนเงินอัตโนมัติ (ต่อ)

การดำเนินการ	เงื่อนไขก่อน
AddSavingAccount	savingAccount? ∉ dom savingAccountAttrs ∧ savingAccount? ∉ savingAccount ∧ savingAccount? ∈ account
RemoveBank	bank? ∈ dom bankAttrs ∧ dom bankAttrs \ {bank?} ∈ ℙ(bank \ {bank?})
RemoveATMMachine	aTMMachine? ∈ dom aTMMachineAttrs ∧ dom aTMMachineAttrs \ {aTMMachine?} ∈ ℙ(aTMMachine \ {aTMMachine?})
RemoveBankClient	bankClient? ∈ dom bankClientAttrs ∧ dom bankClientAttrs\{bankClient?} ∈ ℙ(bankClient\{bankClient?})
RemoveAccount	account? ∈ dom accountAttrs ∧ dom accountAttrs \ {account?} ∈ ℙ(account \ {account?})
RemoveTransaction	transaction? ∈ dom transactionAttrs ∧ dom transactionAttrs \ {transaction?} ∈ ℙ(transaction \ {transaction?})
RemoveCheckingAccount	checkingAccount? ∈ dom checkingAccountAttrs ∧ checkingAccount? ∈ checkingAccount ∧ checkingAccount? ∉ checkingAccount
RemoveSavingAccount	savingAccount? ∈ dom savingAccountAttrs ∧ savingAccount? ∈ savingAccount ∧ savingAccount? ∉ savingAccount
ChangeBank	bank? ∈ dom bankAttrs
ChangeATMMachine	aTMMachine? ∈ dom aTMMachineAttrs
ChangeBankClient	bankClient? ∈ dom bankClientAttrs
ChangeAccount	account? ∈ dom accountAttrs
ChangeTransaction	transaction? ∈ dom transactionAttrs
ChangeCheckingAccount	checkingAccount? ∈ dom checkingAccountAttrs ∧ checkingAccount? ∈ checkingAccount ∧ checkingAccount? ∉ account
ChangeSavingAccount	savingAccount? ∈ dom savingAccountAttrs ∧ savingAccount? ∈ savingAccount ∧ savingAccount? ∉ account

- การสร้างกรณีทดสอบของ AddAccount ซึ่งต้องมีการกำหนดสถานะเริ่มต้นของเซต account และเซต accountAttrs ให้เป็นเซตว่าง ดังนี้



กรณีทดสอบที่ 7

TestAdd2Accounts ≙ InitAccount ⨾ AddAccount ⨾
AddAccount[account2?/account?, a2?/a?]

จากนั้นใช้คำสั่ง try ดังนี้

try TestAdd2Accounts[account? := OID1, a? := account101,
account2? := OID2, a2? := account102];

ซึ่งเมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้จะเป็นการเพิ่มสมาชิกในเซตของ account และ accountAttrs ดังนี้

account' = {OID1} ∪ {OID2} ∧
accountAttrs' = {(OID1, account101)}∪ {(OID2, account102)}
∧ ¬ OID2 = OID1

try TestAdd2Accounts[account? := OID1, a? := account101,
account2? := OID1, a2? := account102];

เมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้จะมีค่าเป็น false เนื่องจากเงื่อนไขก่อนของการเพิ่มสมาชิกในเซตคือ account? ∉ dom accountAttrs

- การสร้างกรณีทดสอบของ RemoveAccount

กรณีทดสอบที่ 8

TestAddThenRemoveAccount ≙ InitAccount ⨾ AddAccount ⨾
RemoveAccount[account2?/account?]

try TestAddThenRemoveAccount[account? := OID1, a? := account101,
account2? := OID1];

ซึ่งเมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้คือ เซตว่างของเซต account และ accountAttrs ดังนี้

account' = ∅ ∧ accountAttrs' = ∅

- การสร้างกรณีทดสอบของ ChangeAccount

กรณีทดสอบที่ 9

TestChangeAccount ≙ InitAccount ⨾ AddAccount ⨾

ChangeAccount[account2?/account?, a2?/a?]

try TestChangeAccount[account? := OID1, a? := account101,

account2? := OID1, a2? := account102];

ซึ่งเมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้คือ เป็นการลบสมาชิกในเซตของ accountAttrs ที่มีสมาชิกลำดับที่หนึ่งของคู่ลำดับใน accountAttrs มีค่าเท่ากับตัวแปรเข้า account2? แล้วเพิ่มสมาชิกในเซต accountAttrs เป็นค่าของคู่ลำดับที่ต้องการแก้ไขค่า ดังนี้

account' = {OID1} ∧

accountAttrs' = {(OID1, account101)} ⊕ {(OID1, account102)}

นั่นก็คือ

account' = {OID1} ∧ accountAttrs' = {(OID1, account102)}

- การทดสอบทฤษฎีบทของ AddAccount

Theorem TwoAccountsOfAddingAreNotSame

∀ TestAdd2Account • account2? ≠ account?

เมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้จะมีค่าเป็น true ทั้งนี้เป็นผลมาจากเงื่อนไขก่อนของ AddAccount คือ account? ∉ dom accountAttrs นั่นคือตัวแปรเข้า account? และ account2? ต้องไม่เท่ากัน

- การทดสอบทฤษฎีบทของ RemoveAccount

Theorem TwoAccountOfRemovingAreSame

∀ TestAddThenRemoveAccount • account2? = account?

เมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้จะมีค่าเป็น true ทั้งนี้เป็นผลมาจากเงื่อนไขก่อนของ RemoveAccount คือ account? ∈ dom accountAttrs ∧ dom accountAttrs \ {account?} ∈ ℙ(account \ {account?}) นั่นคือตัวแปรเข้า account? และ account2? ต้องเท่ากัน

- การทดสอบทฤษฎีบทของ ChangeAccount

Theorem TwoAccountsOfChangingAreSame

∀ TestChangeAccount • account2? = account?

เมื่อใช้คำสั่ง prove by reduce ใน Z/EVES ผลลัพธ์ที่ได้จะมีค่าเป็น true ทั้งนี้เป็นผลมาจากเงื่อนไขก่อนของ ChangeAccount คือ account? ∈ dom accountAttrs นั่นคือตัวแปรเข้า account? และ account2? ต้องเท่ากัน