ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Lê Hồng Anh

PHƯƠNG PHÁP MÔ HÌNH HÓA VÀ KIỂM CHỨNG CÁC HỆ THỐNG HƯỚNG SỰ KIỆN

Chuyên ngành: Kỹ thuật Phần mềm
Mã số: 62.48.01.03

TÓM TẮT LUẬN ÁN TIẾN SĨ CÔNG NGHỆ THÔNG TIN

Hà Nội – 2015#

Công trình được hoàn thành tại: Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội

Người hướng dẫn khoa học: PGS. TS. Trương Ninh Thuận

PGS. TS. Phạm Bảo Sơn

Phản biện: PGS. TS. Nguyễn Đình Hóa .....

.....

Phản biện: PGS. TS. Huỳnh Quyết Thắng .....

.....

Phản biện: TS. Nguyễn Trường Thắng.....

.....

Luận án sẽ được bảo vệ trước Hội đồng cấp Đại học Quốc gia chấm luận án tiến sĩ họp tại .....

vào hồi giờ ngày tháng năm

Có thể tìm hiểu luận án tại:

- Thư viện Quốc gia Việt Nam
- Trung tâm Thông tin - Thư viện, Đại học Quốc gia Hà Nội

#

Chương 1. Tổng quan về luận án

1.1 Lý do lựa chọn đề tài

Mô hình hóa là một trong các phương pháp hiệu quả để quản lý độ phức tạp trong phát triển phần mềm, nó cho phép thiết kế và đánh giá các yêu cầu của hệ thống. Các kỹ thuật kiểm thử có thể được sử dụng trong phát triển phần mềm thông thường để kiểm tra liệu một phần mềm khi thực thi có thỏa mãn yêu cầu của người dùng. Tuy nhiên, kiểm thử là cách xác thực không đầy đủ vì nó chỉ có thể phát hiện được lỗi trong một vài trường hợp những không đảm bảo được hệ thống chạy đúng trong mọi trường hợp. Kiểm chứng phần mềm là một trong những phương pháp hiệu quả để tìm ra lỗi hoặc chứng minh phần mềm không có lỗi một cách toán học. Một vài kỹ thuật đã được đề xuất cho kiểm chứng phần mềm như kiểm chứng mô hình, chứng minh định lý, và phân tích chương trình. Trong các kỹ thuật này, chứng minh định lý có ưu điểm vì có khả năng kiểm chứng các chương trình có kích cỡ lớn và suy diễn qui nạp. Tuy nhiên, chứng minh định lý thường sinh ra nhiều chứng minh phức tạp và khó hiểu.

Kiến trúc phần mềm là một khái niệm được đề xuất để xây dựng các hệ thống phần mềm một cách hiệu quả. Mỗi dạng kiến trúc hoặc kiểu thiết kế thường có các phương pháp mô hình hóa và kiểm chứng phù hợp khác nhau. Kiến trúc hướng sự kiện là một trong những kiến trúc phổ biến trong phát triển phần mềm cung cấp cơ chế gọi dịch vụ không trực tiếp. Mỗi thành phần của kiến trúc này có thể sinh ra các sự kiện, sau đó hệ thống sẽ gọi các thủ tục đã đăng ký với các sự kiện này. Đây là một kiến trúc đầy hứa hẹn cho phép phát triển và mô hình các hệ thống ít ràng buộc nhau và đã được công nhận rộng rãi trong khoa học và ứng dụng. Có nhiều loại hệ thống hướng sự kiện bao gồm các hệ thống giao diện người dùng cho phép sử dụng các sự kiện để thực thi lệnh của người dùng, các hệ thống sinh ra luật sử dụng trong trí tuệ nhân tạo khi xảy ra một điều kiện nào đó (ví dụ như các hệ thống cảm ngữ cảnh), hay các đối tượng hoạt động khi thay đổi giá trị của các thuộc tính thì kích hoạt một số hành động (ví dụ như các hệ thống trigger CSDL). Trong kiến trúc hướng sự kiện, các luật dạng ECA được đề xuất như một cách tiếp cận để đặc tả các quan hệ khi các sự kiện xảy ra ở một điều kiện định trước. Luật ECA có dạng: On *Event* IF *conditions* DO *actions*, ta cũng có thể biểu diễn các luật này một cách không hình thức bằng các luật If-Then, nghĩa là **if** *Events* xảy ra trong điều kiện *condition*, **then** thực thi *action*. Các ưu điểm của phương pháp này đã được sử dụng và tích hợp trong các dạng hệ thống hướng sự kiện như hệ thống trigger CSDL hay các ứng dụng cảm ngữ cảnh. Đã có nhiều nghiên cứu về

phân tích các hệ thống hướng sự kiện cũng như là hình thức hóa các luật ECA. Tuy nhiên, các phương pháp mô hình hóa và kiểm chứng các hệ thống hướng sự kiện tổng quát là chưa đủ vì trên thực tế ta thường phát triển một dạng cụ thể của hệ thống hướng sự kiện. Các nghiên cứu hiện tại của kiểm chứng phần mềm cũng chủ yếu tập trung vào phân tích hệ thống với các mô tả chính xác. Chính vì những lý do trên, các phương pháp mô hình hóa và kiểm chứng phù hợp cho các hệ thống hướng sự kiện cụ thể như hệ thống CSDL và cảm ngữ cảnh hay các hệ thống với yêu cầu không chính xác là cần thiết. Nếu ta có thể kiểm chứng các tính chất quan trọng của hệ thống sớm, đồng thời giảm được độ phức tạp trong chứng minh thì sẽ tiết kiệm được chi phí phát triển và có khả năng áp dụng vào thực tế cao. Luận án đề xuất các phương pháp mới để đạt được các yêu cầu trên sử dụng phương pháp hình thức Event-B. Event-B được dựa trên lý thuyết tập hợp và phù hợp cho mô hình hóa các hệ thống lớn và phản ứng. Tính nhất quán của mô hình Event-B được bảo đảm bởi các chứng minh hình thức. Các công cụ hỗ trợ được cung cấp cho việc đặc tả và chứng minh Event-B dưới dạng các plug-in trên nền tảng Rodin. Vì vậy, sử dụng Event-B làm phương pháp hình thức để mô hình hóa và kiểm chứng các hệ thống hướng sự kiện là phù hợp và có nhiều ưu điểm.

1.2 Mục tiêu

Luận án đưa ra các cách tiếp cận mới và hiệu quả so với các nghiên cứu hiện tại. Thay vì phân tích một hệ thống hướng sự kiện tổng quát, luận án tập trung vào sử dụng Event-B để mô hình hóa các hệ thống hướng sự kiện đặc trưng như các hệ thống trigger CSDL, các hệ thống cảm ngữ cảnh. Luận án đề xuất các phương pháp mới không chỉ mô hình hóa các hành vi được biểu diễn bằng các luật If-Then mà còn hình thức hóa các tính chất quan trọng bằng các thành phần Event-B. Tính đúng đắn của các tính chất này được chứng minh một cách toán học bằng việc chứng minh các mệnh đề cần chứng minh được sinh ra bởi Event-B. Công cụ Rodin được sử dụng trong hỗ trợ quá trình mô hình hóa và chứng minh tự động. Luận án phát triển các công cụ mô hình hoá tự động để làm giảm chi phí và khó khăn trong phát triển các phần mềm hướng sự kiện. Luận án có mục tiêu phân tích các hệ thống hướng sự kiện được mô tả bằng các yêu cầu không chính xác. Luận án giới thiệu phương pháp mới dựa trên làm mịn để mô hình hóa và kiểm chứng một số tính chất quan trọng của hệ thống được mô tả bằng các yêu cầu không chính xác.

1.3 Các đóng góp mới của luận án

1. Luận án đề xuất một phương pháp mô hình hóa và kiểm chứng các hệ thống CSDL có thành phần trigger. Phương pháp này giới thiệu các bước chi tiết để chuyển đổi các khái niệm trong CSDL sang các ký hiệu Event-B. Quá trình chuyển đổi dựa trên cấu trúc tương tự giữa triggers và Event-B event. Với

phương pháp đề xuất, tính chất bảo toàn các ràng buộc được kiểm chứng và các vòng lặp vô hạn sinh ra bởi các trigger có thể được phát hiện bằng các chứng minh hình thức. Một công cụ hỗ trợ Trigger2B chuyển đổi bán tự động cũng được phát triển.

2. Luận án tiếp tục nghiên cứu ưu điểm của cơ chế hoạt động tương tự giữa các luật ECA và sự kiện Event-B để đề xuất phương pháp mô hình hóa và kiểm chứng các hệ thống cảm ngữ cảnh. Luận án phát hiện ra ưu điểm của cơ chế làm mịn trong Event-B để làm cho phương pháp đề xuất phù hợp với mô hình hóa tăng dần. Các tính chất quan trọng như là bảo toàn ràng buộc ngữ cảnh có thể được kiểm chứng qua các mệnh đề cần chứng minh.
3. Luận án đề xuất phương pháp mới để mô hình các hệ thống được mô tả bằng các yêu cầu không chính xác. Các hành vi của hệ thống này được biểu diễn dưới dạng luật If-Then mờ. Luận án giới thiệu một biểu diễn mới của các thuật ngữ mờ bằng các tập hợp và đưa ra một tập luật để chuyển đổi các luật If-Then mờ thành các phần tử Event-B. Luận án mở rộng các luật If-Then mờ với các yếu tố thời gian thời gian để mô hình hóa các hệ thống được định thời.
4. Luận án kế thừa tính làm mịn của Event-B, dựa trên phương pháp mô hình hóa các luật If-Then mờ và một số phương pháp suy diễn để phân tích một số tính chất quan trọng của các yêu cầu không chính xác như tính an toàn và tính hoạt động.

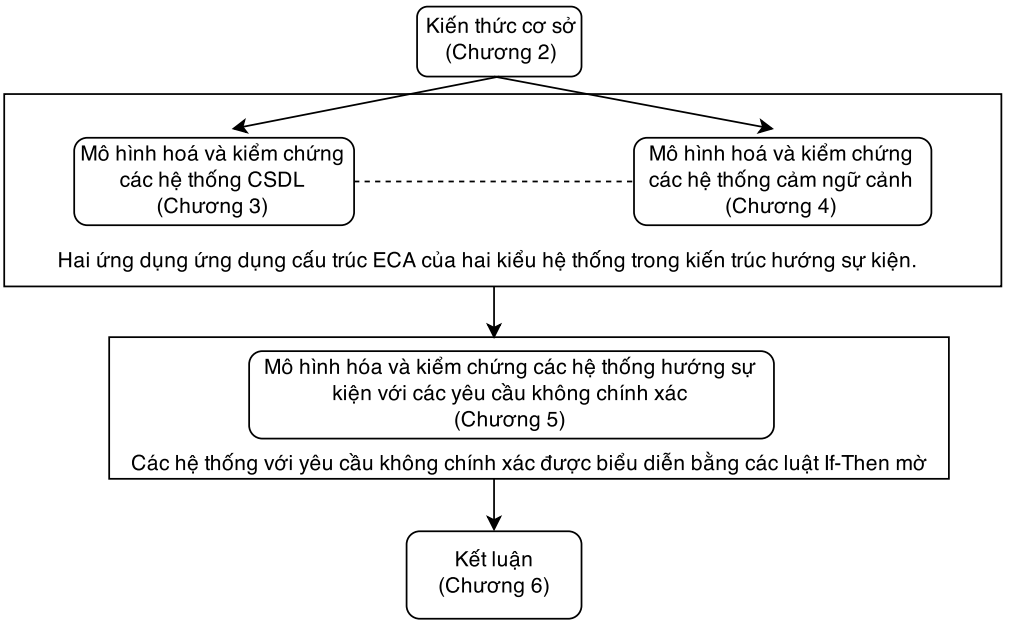
1.4 Cấu trúc luận án

Luận án được tổ chức như Hình 1.1. Chương 2 cung cấp các kiến thức nền tảng cho luận án. Chương 3 giới thiệu phương pháp mô hình hóa và kiểm chứng các hệ thống CSDL. Chương 4 tập trung vào vấn đề mô hình hóa và kiểm chứng các hệ thống cảm ngữ cảnh. Trong Chương 5, luận án đề xuất các phương pháp mô hình hóa và kiểm chứng các hệ thống hướng sự kiện được mô tả bằng các luật If-Then mờ. Chương 6 đánh giá các kết quả đạt được, các điểm còn tồn tại của luận án và đưa ra hướng phát triển.

Chương 2. Kiến thức cơ sở

2.1 Logic thời gian

Logic mệnh đề thời gian là sự mở rộng của logic mệnh đề trong đó mô tả một chuỗi các trạng thái ở những khoảng thời gian khác nhau gọi là thời điểm tức thời. Một thành phần cơ bản của PTL là một công thức logic bậc 1 được xây dựng từ các nguyên tố vị từ, các phép lượng hóa $\exists$, $\forall$; các toán tử $\wedge$, $\vee$,$\neg$; và các toán tử thời gian $\Box$ (“always”),$\Diamond$ (“eventually”), và $\circ$ (“next”), $\mathcal{U}$ (“until").



Hình 1.1: Cấu trúc luận án

2.2 Lý thuyết tập hợp

Ngôn ngữ lý thuyết tập hợp dựa trên các quan hệ cơ bản gọi là thành viên. a là một thành viên của B ($a \in B$), có nghĩa là B chứa a như một phần tử của nó. Có nhiều định nghĩa cơ bản khác của lý thuyết tập hợp như là tập mũ, quan hệ, hàm, vv..

2.3 Tập mờ và luật If-Then mờ

Để xử lý các hệ thống khó được biểu diễn bằng các mô tả chính xác Zadeh đã giới thiệu một khung làm việc logic không phải là logic hai giá trị truyền thống mà là đa giá trị hay được biên dịch bởi các tập mờ. Một tập mờ F được định nghĩa trên một tập X được biểu diễn bởi một cặp: $F = \{(x, \mu_F(x))\}$ trong đó $x \in X$ và $\mu_F(x) : X \to [0, 1]$ được định nghĩa như một hàm thuộc của một phần tử x trong F. Một fuzzy hedge là một toán tử biến đổi tập mờ $F(x)$ sang tập mờ $F(hx)$. Các hedges là các hàm sinh ra một tập lớn hơn các giá trị của các biến. Các luật If-Then mờ được viết dưới dạng đơn giản: "If a is A then b is B", đóng một vai trò quan trọng trong lý thuyết tập mờ. Nó là cách tiếp cận để phân tích các yêu cầu không chính xác của hệ thống.

2.4 Các phương pháp hình thức

Các phương pháp hình thức được sử dụng để đặc tả và kiểm chứng các hệ thống một cách toán học. Một phương pháp là hình thức nếu nó được có định nghĩa cơ sở dưới dạng toán học thường được đưa ra bởi một ngôn ngữ hình thức.

2.4.1 VDM

VDM là viết tắt của "The Vienna Development Method" là một phương pháp dựa trên mô hình mô tả các hệ thống phần mềm thành các mô hình. Các mô

hình này được đặc tả như các đối tượng và các hành động trên các đối tượng, trong đó các đối tượng biểu diễn các trạng thái input, output và bên trong của hệ thống. VDM có ngôn ngữ đặc tả VDM-SL bao gồm mô hình toán học được xây dựng từ các dữ liệu cơ bản như tập hợp, danh sách và ánh xạ cùng với các hành động thay đổi trạng thái của mô hình. VDM-SL có định nghĩa hình thức dựa trên Logic of Partial Functions (LPF).

2.4.2 Phương pháp Z

Đặc tả Z dựa trên lý thuyết tập hợp và tính toán vị từ bậc 1. Mỗi đối tượng có một kiểu riêng, được biểu diễn bởi một tập cực đại trong đặc tả hiện tại. Một khía cạnh của Z là sử dụng ngôn ngữ tự nhiên. Z sử dụng toán học để đưa ra vấn đề, tìm ra các giải pháp và chứng minh thiết kế lựa chọn phù hợp với đặc tả. Z cung cấp cơ chế làm mịn để xây dựng hệ thống dần dần. Một tài liệu đặc tả Z bao gồm các phần hình thức và phi hình thức.

2.4.3 Phương pháp B

B là một phương pháp đặc tả, thiết kế và viết mã cho các hệ thống phần mềm. Ý tưởng chính của B là bắt đầu bằng một mô hình trừu tượng, sau đó thêm dần các chi tiết vào các mô hình cụ thể tiếp theo. B đưa ra các khái niệm về máy trừu tượng đóng gói các thành phần toán học, hằng, tập hợp, biến và một tập hợp các hành động trên các biến này. Các thành phần này được chứa trong các môđun có tên để có thể được sử dụng ở các môđun khác.

2.5 Event-B

Event-B là một phương pháp hình thức mô hình hóa và phân tích mức mô hình. Đặc tính chủ chốt của Event-B là sử dụng lý thuyết tâp hợp làm ký hiệu mô tả, sử dụng cơ chế làm mịn để biểu diễn các hệ thống ở các mức trừu tượng khác nhau và sử dụng các chứng minh toán học để kiểm chứng sự đồng nhất giữa các mức làm mịn. Một cấu trúc mô hình Event-B cơ bản bao gồm một máy và ngữ cảnh. Ngữ cảnh Event-B mô tả phần tĩnh trong đó các tính chất liên quan và các giả thuyết. Một ngữ cảnh có thể bao gồm các tập hợp, hằng và các giả thiết. Tập mang s được biểu diễn bởi tên và khác rỗng. Các tập mang có tên khác thì hoàn toàn độc lập nhau. Hằng c được xác định bởi các giả thiết $P(s, c)$, các giả thiết này cũng phụ thuộc vào tập mang s. Máy Event-B được xác định bằng một tập các mệnh đề bao gồm các biến, bất biến, định lý và sự kiện. Biến v biểu diễn trạng thái của mô hình. Bất biến $I(v)$ sinh ra các luật đề đảm bảo các biến v luôn thỏa mãn. Mỗi sự kiện có dạng evt = any x where $G(x, v)$ then $A(x, v, v')$ end trong đó x,v là các biến cục bộ của sự kiện, $G(x, v)$ là điều kiện xảy ra sự kiện và $A(x, v, v')$ là hành động của sự kiện. Một sự kiện được kích hoạt nếu điều kiện của nó thoả mãn. Một hành động của sự kiện có thể có nhiều phép gán. Mỗi phép gán có thể là (1) phép gán đơn định ($x := E(t, v)$), (2) phép gán rỗng (skip), hoặc (3) phép gán không đơn

định ($x :| P(t, v, x')$). Để giải quyết sự phức tạp khi mô hình hóa hệ thống, Event-B cung cấp cơ chế làm mịn cho phép xây dựng hệ thống từng bước bằng việc thêm vào các chi tiết để đạt được mô hình chính xác hơn. Một máy làm mịn thường có nhiều biến hơn máy trừu tượng vì ta cần nhiều trạng thái biểu diễn mô hình chi tiết hơn. Trong làm mịn chồng, các biến trừu tượng vẫn tồn tại trong máy làm mịn, trong làm mịn theo chiều dọc các biến trừu tượng được thay thế bởi các biến cụ thể. Kết nối giữa biến trừu tượng v và biến cụ thể w được biểu diễn bằng các bất biến $J(v, w)$. Để kiểm tra xem một máy Event-B có thỏa mãn một tập các tính chất không, Event-B định nghĩa ra các mệnh đề cần chứng minh. Một vài mệnh đề cần chứng minh liên quan đến luận án là bảo toàn bất biến (invariant preservation - INV), hội tụ (convergence-VAR), không tắc nghẽn (deadlockfreeness-DLKF).

2.6 Rodin tool

Luận án sử dụng công cụ Rodin phiên bản 2.8 chạy trên môi trường Eclipse để mô hình hóa và chứng minh trong Event-B, có giao diện phong phú cho phép xây dựng các mô hình Event-B một cách thuận lợi, đồng thời tự động sinh ra các mệnh đề cần chứng minh. Rodin đưa ra cơ chế chứng minh tự động hoặc cho phép người dùng tương tác qua các cửa sổ mệnh đề và mục tiêu.

2.7 Các hệ thống hướng sự kiện

Có nhiều loại hệ thống hướng sự kiện như giao diện của phần mềm, các hệ thống sinh luật được sử dụng trong trí tuệ nhân tạo khi điều kiện đúng thì thực thi một hành động (hệ thống cảm ngữ cảnh), hay trong các đối tượng hoạt động khi thay đổi giá trị hay thuộc tính của đối tượng (trigger CSDL).

2.7.1 Hệ thống CSDL và triggers

Các hệ thống CSDL quan hệ hiện đại sử dụng các luật hoạt động như trigger để đáp ứng lại các sự kiện xảy ra bên trong và bên ngoài của CSDL. Trigger là một đoạn mã tự động thực thi khi có một sự kiện xác định xảy ra trong hệ thống CSDL. Cấu trúc của một trigger có dạng ECA : *rule_name:: Event(e) IF condition DO action*. Trigger CSDL phần lớn có thể chia thành hai loại: Data Manipulation Language(DML) và Data Definition Language (DDL). Loại đầu tiên được thực thi khi có thao tác với dữ liệu, loại thứ hai được kích hoạt khi có các sự kiện DDL xảy ra như tạo bảng hoặc login, commit, roll-back..

2.7.2 Các hệ thống cảm ngữ cảnh

Thuật ngữ "cảm ngữ cảnh" được đưa ra lần đầu tiên bởi Bill Schilit, tác giả đã định nghĩa ngữ cảnh là vị trí, là định danh của các đối tượng và sự thay đổi của các đối tượng này sau đó làm cho chương trình thích nghi với ngữ cảnh. Luận án tập trung vào các hệ thống cảm ngữ cảnh sử dụng dữ liệu ngữ cảnh một cách trực tiếp từ các cảm biến vật lý. Hệ thống có thể cảm nhận được nhiều loại ngữ cảnh trong môi trường nó hoạt động như vị trí, độ tăng tốc, nhiệt độ, độ ẩm,

thời thiết.. Và việc xử lý của hệ thống này là phụ thuộc vào ngữ cảnh tức là phản ứng với các thay đổi của ngữ cảnh.

Chương 3. Mô hình hóa và kiểm chứng các hệ thống trigger CSDL

3.1 Giới thiệu

Một trigger là một đoạn mã có cú pháp, không mã hóa nhưng không có ngữ nghĩa. Chính vì vậy, ta chỉ có thể kiểm tra liệu một trigger hoạt động có dẫn đến xung đột ràng buộc dữ liệu hay các triggers có thể dẫn đến các vòng lặp vô hạn không bằng cách thực thi chúng hoặc điều tra từng bước một. Vì những lý do này, các nghiên cứu về một khung làm việc hình thức cho mô hình hóa và kiểm chứng các hệ thống trigger CSDL là cần thiết. Trong chương này, luận án đề xuất một phương pháp mới để mô hình hóa và kiểm chứng hệ thống CSDL có trigger sử dụng Event-B ở giai đoạn thiết kế. Ý tưởng chính của phương pháp xuất phát từ sự tương đồng về cơ chế hoạt động của trigger và sự kiện Event-B. Đầu tiên, chúng tôi đề xuất một tập luật để chuyển đổi một hệ thống CSDL có trigger sang mô hình Event-B, sau đó kiểm tra một cách hình thức liệu hệ thống có thỏa mãn ràng buộc dữ liệu và phát hiện các vòng lặp vô hạn bằng việc chứng minh các mệnh đề cần chứng minh của máy Event-B. Ưu điểm của phương pháp là hệ thống CSDL có thể chuyển đổi sang mô hình Event-B một cách trực tiếp. Các tính chất quan trọng được chứng minh một cách toán học thông qua chứng minh mệnh đề cần chứng minh, vì vậy tính đúng đắn của các tính chất này được đảm bảo. Phương pháp này có giá trị vì nó đảm bảo được hệ thống tránh được một số lỗi nghiêm trọng ở thời điểm thiết kế. Một công cụ Trigger2B ứng dụng phương pháp đề xuất đã được phát triển nhằm hỗ trợ quá trình mô hình hoá tự động. Điều này cũng khắc phục nhược điểm về độ phức tạp khi mô hình hóa của các phương pháp hình thức khiến chúng trở nên khó áp dụng trong phát triển phần mềm.

3.2 Mô hình hóa và kiểm chứng hệ thống triggers

3.2.1 Mô hình hóa hệ thống CSDL

Một hệ thống CSDL thường được thiết kế bao gồm các phần tử như bảng, khung nhìn với các ràng buộc và các triggers. Khi người dùng thực thi các câu lệnh Insert, Delete và Update, những sự thay đổi này có thể làm kích hoạt triggers đồng thời cũng phải tuân theo các ràng buộc định trước. Các qui tắc chuyển đổi một hệ CSDL sang Event-B được tổng kết ở Bảng 3.1.

3.2.2 Hình thức hóa triggers

Bảng 3.2 mô tả cách chuyển một trigger sang một sự kiện Event-B trong đó điều kiện của sự kiện là hợp của loại trigger và điều kiện xảy ra của trigger. Phần

Bảng 3.1: Qui tắc chuyển đổi giữa CSDL và Event-B

	CSDL	**Event-B**
Luật 1	$db = \langle T, C, G \rangle$	DB_M, DB_C
Luật 2	$t = \langle r_1, .., r_m \rangle$	$T = TYPE_1 \times TYPE_2 \times .. \times TYPE_n$
Luật 3	$r_i = \langle f_{i1}, .., f_{in} \rangle$	$t \in \mathbb{P}(T)$
Luật 4	Khoá chính	$f : TYPE_1 \twoheadrightarrow T$
Luật 5	Ràng buộc C	Invariant $\mathcal{I}$
Luật 6	Trigger E	Event Evt

thực thi của trigger được chuyển đổi thành phần thân của sự kiện Event-B. Ta xét trường hợp đơn giản là nội dung trigger chỉ chứa các câu lệnh DML đơn. Mã hóa nội dung của trigger được mô tả trong Bảng 3.3

Bảng 3.2: Hình thức hóa một trigger

IF (e) ON (c)	WHEN $(e \wedge c)$
ACTION (a)	THEN (a) END

3.2.3 Kiểm chứng các tính chất của hệ thống

Sau khi chuyển đổi, với các đặc điểm của Event-B và công cụ hỗ trợ, ta có thể kiểm chứng được các tính chất sau:

- Vòng lặp vô hạn: Vì một trigger có thể kích hoạt trigger khác, nên nó có thể dẫn đến vòng lặp vô hạn. Tình trạng này có thể xảy ra khi sau một chuỗi các sự kiện thì trạng thái của hệ thống không thay đổi. Có hai cách để phát hiện tính chất này sau khi mô hình hóa bằng phương pháp đã đề xuất. Cách thứ nhất, ta chứng minh mệnh đề cần chứng minh deadlock-freeness (DLKF), mệnh đề này là tuyển của các biểu thức điều kiện của các sự kiện Event-B, biểu diễn một cách hình thức: $I(v), P(c) \vdash G_1(v) \vee ... \vee G_n(v)$, trong đó v là biến, $I(v)$ là bất biến, $G_i(v)$ là điều kiện của sự kiện. Trong một số trường hợp, mệnh đề DLKF không suy diễn được từ tập các bất biến $I(v)$ và hằng ràng buộc, ta sẽ chứng minh tại một thời điểm luôn có 1 sự kiện được kích

Bảng 3.3: Mã hóa hành động của trigger

INSERT INTO T VALUES (value1,..,valuen)	ANY r WHEN $(r \in T \wedge e \wedge c)$ THEN $T := T \cup r$ END
DELETE FROM T WHERE $\langle column1 = some\_value \rangle$	ANY v WHEN $(v \in TYPE_1 \wedge e \wedge c)$ THEN $t := t - f(v)$ END
UPDATE T SET column1=value, column2=value2 WHERE $\langle column1 = some\_value \rangle$	ANY $v1, v2$ WHEN $v1 \in TYPE_1 \wedge v2 \in TYPE_2 \wedge e \wedge c$ THEN $t := \{1 \mapsto value1, 2 \mapsto value2\} \oplus t$ END

hoạt bằng chứng minh hội của các điều kiện luôn đúng trước và sau khi thực thi sự kiện, tức là chứng minh mệnh đề cần chứng minh INV.

- Bảo toàn ràng buộc: Với phương pháp chuyển đổi đề xuất, một trigger không vi phạm các luật này nếu $I(v)$, $G(w, v)$, $S(w, v, v') \vdash I(v')$. Đây cũng chính là mệnh đề cần chứng minh INV của một máy Event-B.

3.3 Case study: Ứng dụng quản lý nhân sự

3.3.1 Mô tả

Trong một hệ thống CSDL cho ứng dụng quản lý nhân sự có hai bảng EMPLOYEES (gồm hai trường id và level) và bảng BONUS (gồm hai trường id và amount). Ứng dụng có yêu cầu ràng buộc về dữ liệu như sau: *Trường amount của BONUS* $\geq$*10 nếu level của nhân viên* $>$*5*. Hệ thống CSDL này được thiết kết hai triggers thực hiện nhiệm vụ sau:
Trigger 1. Nếu bảng EMPLOYESS được cập nhật thì bonus của nhân viên tương ứng tăng lên 10. Trigger 2. Nếu trường amount của nhân viên ở BONUS được tăng thêm giá trị $\geq$*10 thì level của nhân viên tăng lên 1.*

3.3.2 Mô hình hóa

Một phần đặc tả Event-B của hệ thống CSDL trên được thể hiện ở Hình 3.1, Hình 3.2 và Hình 3.3.

CONTEXT *TRIGGER_C*
SETS
    TYPES
    TABLE_NAMES
CONSTANTS
    TBL_EMPL
    TBL_BONUS
AXIOMS
    axm1 : partition(TYPES, {insert}, {update}, {delete})
    axm2 : TBL_EMPL $= \mathbb{N} \times \mathbb{N}$
    axm3 : TBL_BONUS $= \mathbb{N} \times \mathbb{N}$
    axm4 : partition(TABLE_NAMES, {employees}, {bonus})
END

Hình 3.1: Ngữ cảnh Event-B

3.3.3 Kiểm chứng các thuộc tính

- Bảo toàn ràng buộc: Tính chất này được hình thức hóa bởi bất biến SYS_CTR : $\forall\, eid.eid \in dom(empl) \wedge pk\_empl(eid) > 5 \Rightarrow pk\_bonus(eid) > 10$.
  Ta cần chứng minh bất biến này bảo toàn trước và sau khi các sự kiện thực thi, mệnh đề cần chứng minh tương ứng của $trigger1$ được mô tả như Bảng 3.4. Hai sự kiện $Trigger1$ và $Trigger2$ của máy DB_M sinh ra các mệnh đề cần chứng minh tương ứng là trigger1/SYS_CTR/INV, trigger2/SYS_CTR/INV.
- Vòng lặp vô hạn: Ở mục 3.2.3, bất biến INF_LOOP là mệnh đề tuyển của các điều kiện của sự kiện được sử dụng để phát hiện vòng lặp vô hạn. Nếu

MACHINE *TRIGGER_M*
SEES *TRIGGER_C*
VARIABLES
bonus
empl
f_bonus
f_empl
type
INVARIANTS
inv1 : $bonus \in \mathbb{P}(TBL\_BONUS)$
inv2 : $empl \in \mathbb{P}(TBL\_EMPL)$
inv3 : $type \in TYPES$
inv4 : $f\_bonus \in \mathbb{N} \pfun \mathbb{N}$
inv5 : $f\_empl \in \mathbb{N} \pfun \mathbb{N}$
SYS_CTR : $\forall eid.eid \in dom(empl) \land pk\_empl(eid) > 5 \Rightarrow pk\_bonus(eid) > 10$
INF_LOOP : $(type = update \land table = BONUS) \lor (type = update \land table = EMPL)$
END

Hình 3.2: Máy Event-B

Event *trigger1* $\widehat{=}$
any
eid
when
grd1 : $type = update$
grd2 : $table = EMPL$
grd3 : $eid \in dom(empl)$
then
act1 : $type := update$
act3 : $table := BONUS$
act5 : $bonus := \{eid \mapsto (pk\_bonus(eid) + 10)\} \oplus bonus$
act6 : $pk\_bonus(eid) := pk\_bonus(eid) + 10$
end
Event *trigger2* $\widehat{=}$
any
eid
when
grd1 : $type = update$
grd2 : $table = BONUS$
grd3 : $pk\_bonus(eid) \geq 10$
then
act1 : $type := update$
act2 : $table := EMPL$
act3 : $empl := \{eid \mapsto (pk\_empl(eid) + 1)\} \oplus empl$
end

Hình 3.3: Hình thức hóa trigger

bất biến này bảo toàn với hai sự kiện thì hai triggers sẽ dẫn đến vòng lặp vô hạn. Mệnh đề cần chứng minh của tính chất này được thể hiện ở Bảng 3.5.

3.4 Công cụ hỗ trợ: Trigger2B

Từ phương pháp đề xuất ở Mục 3.2, chúng tôi đã cài đặt một công cụ Trigger2B để hỗ trợ mô hình hóa và kiểm chứng một hệ thống CSDL có trigger. Công cụ này có thể sinh ra các file dưới định dạng XML để có thể sử dụng làm đầu vào cho các bộ chứng minh và công cụ hỗ trợ khác của Event-B (Rodin).

BẢNG 3.4: Mệnh đề cần chứng minh bảo toàn ràng buộc

$$\begin{array}{l}\forall\, nid.nid \in dom(empl\_rec) \land pk\_empl(nid) > 5 \Rightarrow pk\_bonus(nid) > 10\\ emplid \in dom(empl\_rec)\\ type = update\\ table = EMPL\\ \vdash\\ \forall\, nid.nid \in dom(empl\_rec) \land pk\_empl(nid) > 5\\ \Rightarrow (pk\_bonus \oplus \{emplid \mapsto pk\_bonus(emplid) + 10\})(nid) > 10\end{array}$$

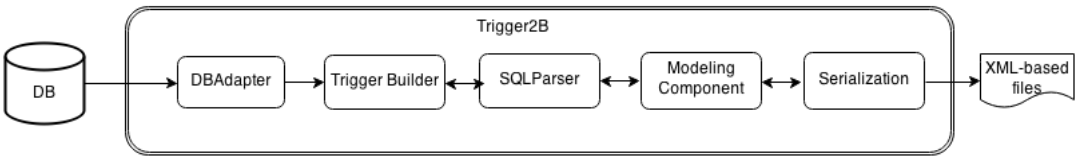
BẢNG 3.5: Mệnh đề cần chứng minh vòng lặp vô hạn

$$\begin{array}{l}\forall\, nid.(nid \in dom(empl\_rec) \land\\ type = update \land table = BONUS \land\\ pk\_bonus(nid) \geqslant 10) \lor (type = update \land table = EMPL)) \land\\ emplid \in dom(bonus\_rec)\\ table = BONUS \land pk\_bonus(emplid) \geqslant 10\\ \vdash\\ \forall\, nid.(nid \in dom(\{emplid \mapsto pk\_empl(emplid) + 1\} \oplus empl\_rec) \land\\ update = update \land EMPL = BONUS \land\\ pk\_bonus(nid) \geqslant 10) \lor\\ (update = update \land EMPL = EMPL)\end{array}$$

3.4.1 Kiến trúc

Kiến trúc của công cụ được mô tả như Hình 3.4 bao gồm 05 môđun như sau:

- DBAdapter: trích xuất thông tin từ các nguồn CSDL khác nhau
- Trigger Builder: hỗ trợ xây dựng trigger từ thông tin trích xuất bởi DBAdapter.
- SQLParser: phân tích nội dung của các trigger dưới dạng SQL.
- Modeling: chuyển đổi cây cú pháp phân tích được sang mô hình Event-B.
- Serialization: kết xuất ra các định dạng khác nhau có thể sử dụng làm đầu vào cho các bộ chứng minh Event-B.



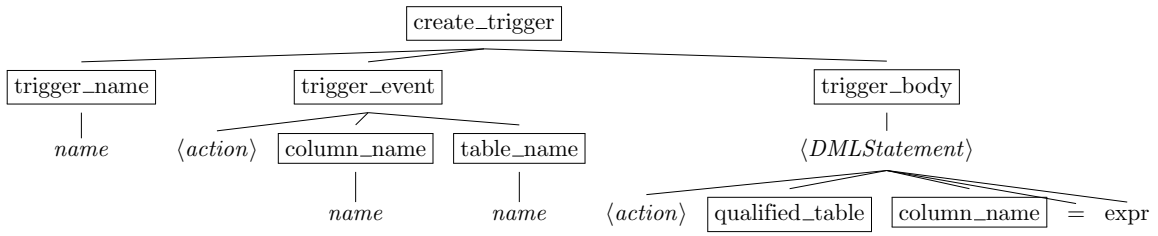
HÌNH 3.4: Kiến trúc của Trigger2B

3.4.2 Cài đặt

Trong mục này, luận án trình bày chi tiết phương pháp cài đặt với các module quan trọng như SQLParser, Modeling Component. Hình 3.5 mô tả một cây cú pháp được sinh ra khi phân tích nội dung của các trigger.

Giải thuật duyệt và chuyển đổi cây cú pháp thành mô hình Event-B dựa trên các luật đã đề xuất được mô tả như sau:

Input: Parsed syntax tree(t)



Hình 3.5: Cây cú pháp sinh bởi bộ phân tích ANLTR-v3

Output: Event-B model (M, C)

```
1   begin
2       node = root(t)
3       while (isVisited(node))
4           if node.type = create_trigger then
5               e=createNewEvent(M)
6               if node.type = trigger_name then
7                   e.name = node.name
8               elseif node.type = trigger_event
9                   for child in nodes.childs
10                      if node.type = action then
11                          addGuard(e,type=node.value)
12                      if node.type = tabletable_name then
13                          addGuard(e,table=node.child.value)
14              elseif node.type = trigger_body
15                          addAction(e,getExp(node.childs))
16          end
17          visit_next(node)
18      end
```

Chương 4. Mô hình hóa và kiểm chứng các hệ thống cảm ngữ cảnh

4.1 Giới thiệu

Cảm ngữ cảnh của một ứng dụng liên quan tới khả năng đáp ứng, phản hồi và độ nhạy cảm của ứng dụng với các thay đổi của ngữ cảnh. Theo nghĩa hẹp thì một hệ thống cảm ngữ cảnh có thể xem như một hệ thống hướng sự kiện nghĩa là nó nhận các sự kiện gửi khi có sự thay đổi về ngữ cảnh và phản hồi lại các sự kiện này với các tri thức về ngữ cảnh đang có. Hành vi của các hệ thống cảm ngữ cảnh thường là phức tạp và không chắc chắn. Đã có nhiều kết quả nghiên cứu với nhiều cách tiếp cận khác nhau như mô hình hóa vai trò đối tượng, mô hình hóa dựa trên ontology, mô hình hóa dựa trên logic. Một số nghiên cứu cũng được ra các khung làm việc cho mô hình hóa ngữ cảnh. Trong chương này, luận

án đề xuất sử dụng Event-B để mô hình hóa và kiểm chứng các hệ thống cảm ngữ cảnh. Các đóng góp của đề xuất này là (1) Biểu diễn tự nhiên các hệ thống cảm ngữ cảnh bằng các thành phần Event-B. Đề xuất một tập luật định nghĩa các thành phần cảm ngữ cảnh một cách hình thức. Đây là phương pháp dựa trên làm mịn cho phép xây dựng hệ thống theo từng bước. (2) Sau khi hình thức hóa, các tính chất quan trọng của hệ thống được kiểm chứng qua các mệnh đề cần chứng minh của cơ chế làm mịn mà không cần qua bước chuyển đổi trung gian nào.

4.2 Hình thức hóa tính chất cảm ngữ cảnh

4.2.1 Mô hình hóa hệ thống cảm ngữ cảnh

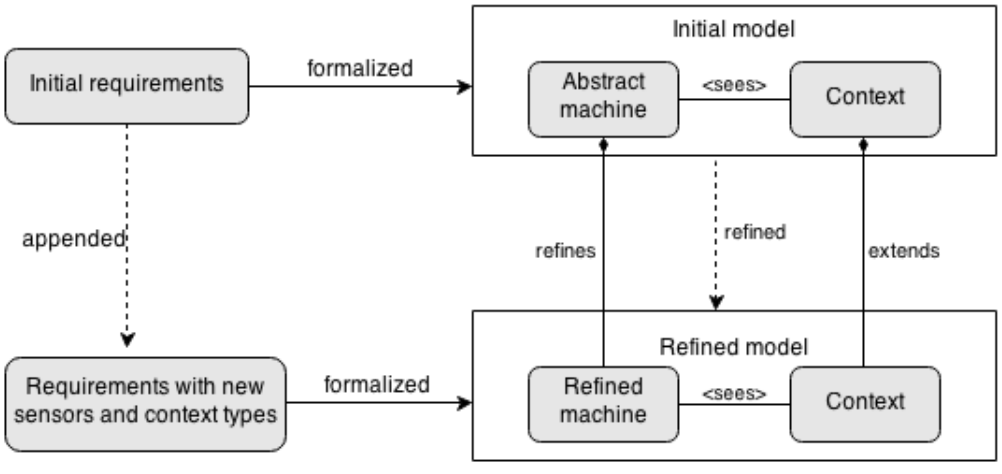
Các luật chuyển đổi giữa một hệ thống cảm ngữ cảnh và một mô hình Event-B được trình bày ở Bảng 4.1

Bảng 4.1: Chuyển đổi giữa thành phần cảm ngữ cảnh và Event-B

	Hệ thống cảm ngữ cảnh	**Event-B**
Luật 1	Dữ liệu ngữ cảnh CD	Sets, Constants
Luật 2	Luật ngữ cảnh $r = \langle e, c, a \rangle$	Events
Luật 3	Trigger môi trường E	Events
Luật 4	Ràng buộc ngữ cảnh CC	Invariants

4.2.2 Mô hình hóa tăng dần sử dụng cơ chế làm mịn

Trên thực tế, phát triển các hệ thống cảm ứng thường được xây dựng từ các yêu cầu cơ bản, sau đó được xây dựng dần khi có thêm các yêu cầu về các thực thể ngữ cảnh và tri thức suy diễn. Hệ thống cũng có thêm các luật ngữ cảnh để xử lý các dữ liệu này. Lúc này, hệ thống mới vẫn phải thỏa mãn những ràng buộc đã được thiết lập. Vì vậy, ta cần có một phương pháp mô hình hóa phù hợp cho qui trình phát triển tiến hóa này. Hình 4.1 mô tả một phương pháp mô hình hóa tiến hóa dựa trên phương pháp đã đề xuất ở Mục 4.2.1.



Hình 4.1: Mô hình hóa tăng dần sử dụng làm mịn

Cơ chế làm mịn cho phép mô hình hóa các hệ thống cảm ngữ cảnh theo từng bước. Event-B có hai cơ chế làm mịn chồng và làm mịn theo chiều dọc. Với cơ chế làm mịn chồng, các biến trừu tượng vẫn tồn tại trong các máy cụ thể cùng với các biến bổ sung. Chính vì vậy phương pháp dựa trên cơ chế này phù hợp với một hệ thống cảm ngữ cảnh được mở rộng bằng việc bổ sung các cảm biến.

- Phần dữ liệu tĩnh: khi một cảm biến được thêm vào hệ thống, ta có thể sẽ phải giải quyết với các kiểu dữ liệu khác. Phương pháp đề xuất mô hình hóa nó như một ngữ cảnh mới mở rộng từ các ngữ cảnh ở mô hình trừu tượng.
- Các hành vi động: Ta mô hình hóa các hành vi chung của hệ thống với các mô tả ban đầu bằng các máy trừu tượng, sau đó làm mịn các máy này bằng các máy cụ thể. Trong các máy đã làm mịn các biến mới được thêm có thể tham chiếu đến các thành phần dữ liệu ngữ cảnh. Các sự kiện của một máy làm mịn có thể kế thừa các sự kiện của máy trừu tượng.

4.3 Case study: Hệ thống Adaptive Cruise Control

4.3.1 Mô tả ban đầu

Hệ thống ACC điều khiển tốc độ của ôtô dựa trên điều kiện lái xe đồng thời được có chức năng cảm ngữ cảnh như nhận biết vật cản phía trước nhờ sử dụng cảm biến trước xe. Ôtô có tốc độ đối đa và được thiết lập khi bắt đầu khởi động xe. Nếu không phát hiện vật cản phía trước thì sẽ tăng và giảm tốc độ nếu phát hiện vật cản. Nếu xe dừng và không có vật cản thì tốc độ xe lại được thiết lập là tối đa. Hệ thống ACC luôn thỏa mãn ràng buộc ngữ cảnh là tốc độ luôn trong khoảng an toàn tức là không vượt quá tốc độ cho phép.

4.3.2 Mô hình hóa hệ thống ACC

Trong kịch bản này, hệ thống có ba cảm biến, theo phương pháp được đề xuất ở Mục 4.2, chúng ta đặc tả hệ thống ban đầu với một máy và ngữ cảnh là ACC_M0 và *Target* (Hình 4.2).

4.3.3 Làm mịn: Bổ sung cảm biến thời tiết và độ nhẵn mặt đường

Hai cảm biến này được bổ sung vào hệ thống và hoạt động tương tự như cảm biến phát hiện chướng ngại vật, gửi dữ liệu về hệ thống. Các luật ngữ cảnh được mở rộng để xử lý các sự kiện gửi dữ liệu về hai cảm biến này như sau: “Khi một ôtô di chuyển trong điều kiện trời mưa hoặc mặt đường không tốt thì ACC giảm tốc độ ôtô". Với hệ thống mới này, hệ thống cần đảm bảo trong điều kiện thời tiết hoặc đường xấu thì tốc độ nhỏ hơn tốc độ tối đa.

Mô hình làm mịn: Dựa trên phương thức đã đề xuất, ngữ cảnh *Weather_Road* mở rộng ngữ cảnh *Target* biểu diễn dữ liệu ngữ cảnh của các cảm biến mới. Ta thêm hai sự kiện cho máy làm mịn. Sự kiện thứ nhất biểu diễn luật mới và không làm mịn sự kiện nào ở máy trừu tượng. Nó biểu diễn hành vi của hệ thống khi các cảm biến gửi dữ liệu biểu diễn trạng thái thời tiết có mưa hay đường

CONTEXT Target
CONSTANTS
TARGET_DETECTION
MAX_SPEED
INC
AXIOMS
axm1 : TARGET_DETECTION = BOOL
axm2 : MAX_SPEED ∈ ℕ
axm3 : INC < MAX_SPEED
axm4 : INC ∈ ℕ
END

EVENTS
Event *TargetDetected* ≙
when
grd1 : target_det = TRUE
grd2 : speed > INC
then
act1 : speed := speed − INC
end
Event *TargetUndetected* ≙
when
grd1 : target_det = FALSE
grd2 : speed < MAX_SPEED − INC
then
act1 : speed := speed + INC
end
END

HÌNH 4.2: Đặc tả Event-B cho hệ thống ban đầu

xấu. Trong khi sự kiện *TargetUndetected* làm mịn sự kiện của máy trừu tượng. Ràng buộc ngữ cảnh được hình thức hóa thành bất biến *cxt_ct* (Hình 4.3).

4.3.4 Kiểm chứng tính chất bảo toàn ngữ cảnh

Các ràng buộc ngữ cảnh được chuyển đổi thành các mệnh đề bất biến. Vì vậy, ta có thể chứng minh tính đúng đắn của hệ thống bằng chứng minh các mệnh đề cần chứng minh INV.

BẢNG 4.2: Chứng minh tính bảo toàn ngữ cảnh

$target\_det = TRUE \Rightarrow speed < MAX\_SPEED$ $target\_det = TRUE$ $speed > INC$ $\vdash$ $target\_det = TRUE \Rightarrow speed - INC < MAX\_SPEED$	ctx_ct/INV

Các mệnh đề cần chứng minh cho các bất biến của cả máy trừu tượng và làm mịn như sau:

- Machine *ACC_M0*: "*TargetDetected/ctx_ct/INV*" (Hình 4.2) and "*TargetUndetected/ctx_ct/INV*"
- Machine *ACC_M1*: *TargetUndetected/ctx_ct/INV*, *RainSharp/ctx_ct/INV*

Chương 5. Mô hình hóa và kiểm chứng các hệ thống với yêu cầu không chính xác

5.1 Giới thiệu

Các phương pháp hình thức đưa ra các khung làm việc để đặc tả và kiểm chứng tính đúng đắn của các hệ thống nói chung và các hệ thống hướng sự kiện nói riêng thường đòi hỏi các yêu cầu mô tả chính xác. Tuy nhiên, chúng ta thường

CONTEXT Weather_Road
EXTENDS Target
CONSTANTS
RAINING
SHARP
AXIOMS
axm1 : RAINING = BOOL
axm2 : SHARP = BOOL
END

MACHINE ACC_M1
REFINES ACC_M0
SEES Weather_Road
VARIABLES
isRain
speed
target_det
isSharp
INVARIANTS
inv1 : isRain ∈ RAINING
cxt_ct : isRain = TRUE ∨ isSharp = TRUE ⇒ speed < MAX_SPEED
inv3 : isSharp ∈ SHARP
EVENTS
Event *TargetUndetected* ≙
extends *TargetUndetected*
when
grd1 : $target\_det = FALSE$
grd2 : $speed < MAX\_SPEED - INC$
grd3 : isRain = FALSE
grd4 : isSharp = FALSE
then
act1 : $speed := speed + INC$
end
Event *RainSharp* ≙
when
grd1 : isRain = TRUE ∨ isSharp = TRUE
then
act1 : speed := speed − INC
end
END

Hình 4.3: Mô hình làm mịn của hệ thống ACC

phải đối mặt với những mô tả không chính xác có các thuật ngữ không rõ ràng, nhập nhằng hay không chắc chắn. Chính vì vậy, những khung làm việc hình thức có thể sử dụng để phân tích và biểu diễn các yêu cầu không chính xác là cần thiết. Phương pháp với tập mờ được đề xuất bởi Zadeh là một trong những khung làm việc như vậy, trong đó các luật If-Then mờ được sử dụng để biểu diễn các yêu cầu không chính xác. Các yêu cầu này được biểu diễn dưới dạng các luật If-Then mờ có thể là đủ biểu diễn nhưng vẫn có yêu cầu các kỹ thuật tiếp theo để kiểm chứng các tính chất một cách hình thức trên nhiều khía cạnh để phát hiện và giải quyết các xung đột. Chương 5 sử dụng lợi thế của cơ chế làm mịn Event-B để mô hình hóa và kiểm chứng các hệ thống hướng sự kiện được mô tả bởi các tập luật If-Then mờ. Theo như hiểu biết của chúng tôi thì đây là những kết quả cụ thể đầu tiên trong kiểm chứng các tính chất an toàn (safety) và kết quả (eventuality) của các yêu cầu không chính xác. Đóng góp

của chương là (1) Đưa ra một tập luật chuyển đổi từ các luật If-Then mờ sang đặc tả Event-B (2) Sử dụng cơ chế làm mịn của Event-B để hình thức hóa các luật If-Then mờ và mở rộng các luật này với yếu tố thời gian. (3) Đưa ra một tập luật chuyển đổi để hình thức hóa tính kết quả sang đặc tả Event-B (4) Sử dụng công cụ Rodin để kiểm chứng các thuộc tính an toàn và kết quả của hệ thống.

5.2 Mô hình hóa các yêu cầu mờ sử dụng Event-B

Trong phần này, luận án giới thiệu một cách tiếp cận để biểu diễn các thuật ngữ mờ bằng lý thuyết tập hợp cổ điển. Dựa trên các biểu diễn này, chúng tôi đề xuất một tập luật để chuyển đổi các luật If-Then mờ sang Event-B. Sau đó sử dụng cơ chế làm mịn để mô hình hóa các hệ định thời.

5.2.1 Biểu diễn các thuật ngữ mờ bằng tập hợp cổ điển

Hệ quả 5.1. Một tập các yêu cầu mờ "If x is δY then m is γP" có thể được đặc tả bằng các tập hợp cổ điển.

5.2.2 Mô hình hoá các trạng thái rời rạc

Một hệ thống bao gồm một tập các yêu cầu $FR_i, i = \overline{1, n}$ được mô hình hóa bởi một mô hình Event-B $FR_B = \langle FR\_C, FR\_M \rangle$, trong đó FR_C và FR_M là ngữ cảnh và máy Event-B tương ứng. Chúng tôi đề xuất các luật được sử dụng để ánh xạ các yêu cầu không chính xác vào các phần tử Event-B. Nguyên tắc quan trọng là với phương pháp này ta có thể bảo toàn cấu trúc và biểu diễn toàn bộ các yêu cầu mờ sử dụng ký hiệu Event-B. **Các luật chuyển đổi:**

- Luật 1. hedges δ_i, γ_i generators Y_i và giá trị P_i trong một tập các yêu cầu có thể được chuyển đổi thành ba tập hợp tương ứng δ, γ, Y, và P. Các tập này được đưa ra trong ngữ cảnh FR_C.
- Luật 2. Các biến x_i, m_i trong mỗi FR_i được ánh xạ đến các biến x_i, m_i của máy Event-B FR_M.
- Luật 3. Mỗi biến x_i được biểu diễn thành $\delta \times Y$, trong đó $m_i \in \gamma \times P$.
- Luật 4. Mỗi yêu cầu FR_i được mô hình hóa bởi một sự kiện ev_i trong máy Event-B FR_M

Hình 5.1 là đặc tả của mô hình Event-B đích sau khi thực hiện các luật chuyển đổi.

5.2.3 Mô hình hóa các trạng thái liên tục

Trước tiên ta định nghĩa các luật If-Then mờ định thời có dạng như sau: IF x(t) is A THEN y(t) is B. Áp dụng phương pháp mô hình hoá các hệ lai bằng Event-B, nếu một yêu cầu mờ FR_i chứa một biến phụ thuộc thời gian nào đó, thì ta sẽ làm mịn sự kiện tương ứng của máy trừu tượng. Đồng thời, ta cũng đã giới thiệu trong Luật 3 là nếu một biến x được đặc tả như một thành viên của $\delta \times Y$, trong đó δ là một tập hợp của các fuzzy hedges, Y là một tập mờ.

CONTEXT *FR_C*
SETS
γ
δ
P
Y
END

MACHINE *FR_M*
SEES *FR_C*
VARIABLES
x_i
m_i
INVARIANTS
inv1 : $x_i \in \mathbb{P}(\delta \times Y)$
inv2 : $m_i \in \mathbb{P}(\gamma \times P)$
EVENTS
Event $FR_i \mathrel{\widehat{=}}$
when
grd1 : $x_i = \{\delta_i \mapsto Y_i\}$
then
act1 : $m_i := \{\gamma_i \mapsto P_i\}$
act2 : $x_i := \{\delta_j \mapsto Y_j\}$
end
END

HÌNH 5.1: Đặc tả Event-B cho mô hình hoá các trạng thái rời rạc

Tương tự, ta thêm các biến liên tục x_c của máy làm mịn được khai báo như sau $x_c \in \mathbb{R} \nrightarrow \delta \times P$.

- Luật 5: Nếu biến x_i hoặc m_i (trong một yêu cầu FR_i) gắn với trục thời gian, thì sự kiện tương ứng sẽ được làm mịn. Một biến $t \in \mathbb{R}$ để biểu diễn thời gian sẽ được thêm vào máy làm mịn.
- Luật 6: Các biến phụ thuộc thời gian (trong Luật 2) được thay thế bởi các hàm thời gian và bất biến ràng buộc trong máy làm mịn.

5.3 Kiểm chứng các hệ thống yêu cầu không chính xác

Các tính chất phổ biến của các hệ thống là tính an toàn và tính hoạt động. Tính an toàn bảo đảm để hệ thống không rơi vào trạng thái xấu trong khi các tính chất hoạt động đảm bảo một trạng thái tốt của hệ thống (tính dừng, kết quả, tiến trình). Trong mục này, luận án sẽ giới thiệu một phương pháp dựa trên làm mịn mới để mô hình hóa và kiểm chứng cả hai tính chất an toàn và kết quả.

5.3.1 Tính hội tụ trong Event-B

Tính chất hội tụ của máy Event-B là sự hội cụ các sự kiện. Tức là một tập các sự kiện không thể chạy không ngừng. Điều đó có nghĩa là có sự kiện khác sẽ xảy ra. Những sự kiện như vậy gọi là sự kiện hội tụ. Để chứng minh tính chất này, Event-B đưa ra cơ chế sử dụng variant V ánh xạ đến một biến trạng thái v đến một số nguyên, các sự kiện này được chứng minh là giảm giá trị của biến v. Chi tiết hơn, e là một sự kiện hội tụ, v và v' là trạng thái trước/sau khi thực hiện e, ta chứng minh rằng $V(v') < V(v)$. Hai mệnh đề cần chứng minh này được sinh ra cho mỗi sự kiện hội tụ trong đó mệnh đề cần chứng minh VAR

bảo đảm giá trị variant sẽ giảm và NAT đảm bảo là variant là một số nguyên sau khi thực thi sự kiện.

5.3.2 Phân tích tính an toàn và hoạt động trong Event-B

Event-B cung cấp một cách biểu diễn các thuộc tính an toàn một cách trực tiếp bằng các bất biến. Vì vậy, ta có thể chứng minh tính đúng đắn của các tính chất này bằng chứng minh mệnh đề INV. Event-B không hỗ trợ đặc tả các tính chất hoạt động một cách trực tiếp nhưng ta có thể áp dụng một số kết quả nghiên cứu mới nhất về suy diễn trong Event-B để kiểm chứng các tính chất như *existence* ($\Box\Diamond P$), *progress* ($\Box(P_1 \Rightarrow \Diamond P_2)$), *persistence* ($\Diamond\Box P$), trong đó P là một công thức logic bậc một, $\Diamond$ và $\Box$ là các toán tử chuẩn của Linear Temporal Logic (LTL).

5.3.3 Kiểm chứng tính chất an toàn

Hệ quả 5.2. Với các luật chuyển đổi đã đề xuất, các tính chất an toàn được bảo toàn bởi các hành động trong yêu cầu hệ thống không chính xác.

5.3.4 Kiểm chứng tính chất kết quả

Luận án đề xuất một phương pháp dựa trên làm mịn với việc bổ sung thêm các luật để mở rộng ngữ cảnh và làm mịn máy Event-B như sau:

- Luật 7. Giá trị mờ P, Y và hedges δ, γ được chuyển thành các hàm tương ứng $deg_P : P \to \mathbb{N}$, $deg_Y : Y \to \mathbb{N}$, $deg_{H\delta} : \delta \to \mathbb{N}$, và $deg_{H\gamma} : \gamma \to \mathbb{N}$.
- Luật 8. Thêm một variant ánh xạ tới biến có xuất hiện trong biểu thức tính chất kết quả Q.
- Luật 9. Làm mịn các sự kiện biểu diễn cho các yêu cầu If-Then bằng hai sự kiện: sự kiện thường và sự kiện hội tụ.
- Luật 10. Thêm mệnh đề $\neg Q(x_i)$ vào các điều kiện của mỗi sự kiện hội tụ và mệnh đề $Q(x_i)$ vào sự kiện thường.
- Luật 11. Tính chất tắc nghẽn được mã hóa như một định lý của máy Event-B.

Định nghĩa 5.2.[Tính hội tụ] Các luật mờ là hội tụ từ một trạng thái $Q(x)$ nếu mỗi luật làm giảm giá trị của biến x. Tính hội tụ được định nghĩa hình thức như sau: $FR_i, Q(x) \vdash x' < x$, trong đó x' là giá trị sau khi thực hiện luật FR_i.

Định nghĩa 5.3.[Không tắc nghẽn] Các luật mờ là không tắc nghẽn ở trạng thái $Q(x)$ nếu luôn có một mệnh đề IF của một luật thỏa mãn. Định nghĩa một cách hình thức là: $Q(x) \Rightarrow \bigvee_{i=1}^{n} (\exists\, x_i . x_i = \delta\, Y_i)$

Hệ quả 5.4. Với các luật chuyển đổi đã đề xuất, nếu một tập luật If-Then mờ là hội tụ và không tắc nghẽn từ một biểu thức logic trạng thái $Q(x)$ trong đó x là một biến thì thuộc tính trạng thái $\neg Q(x)$ sẽ luôn xảy ra. Một cách hình thức, ta có $\{FR\} \vdash \Box\Diamond\neg Q(x)$.

5.4 Case study: Container Crane Control

5.4.1 Mô tả

Cần cẩu container được sử dụng để nhấc và thả hàng được sử dụng trong các cảng. Một tập hợp các yêu cầu mờ mô tả hệ thống như sau:

- FR_1 Nếu cần cẩu đang ở vị trị ban đầu, thì tốc độ ở mức nhanh.
- FR_2. Nếu khoảng cách từ cần cẩu đến container xa, thì tốc độ ở mức trung bình.
- FR_3. Nếu khoảng cách đến container là trung bình thì tốc độ được điều chỉnh ở mức thấp.
- FR_4. Nếu khoảng cách là gần thì tốc độ là rất chậm.
- FR_5. Nếu cần cẩu ở trên container thì tốc độ bằng 0.

Hệ thống có một thuộc tính an toàn để đảm bảo tốc độ của motor không ở tốc độ cao khi khoảng cách không xa ($\mathcal{I}$). Đồng thời hệ thống cũng cần thỏa mãn tính chất từ vị trí ban đầu, một lúc nào đó cần cẩu sẽ ở trên container($\mathcal{Q}$). Biểu diễn một cách hình thức, ta cần kiểm chứng $\{FR\} \vdash \mathcal{I}$ và $\{FR\} \vdash \Box\Diamond\neg\mathcal{Q}$.

5.4.2 Mô hình hóa hệ thống Container Crane Control

5.4.2.1 Mô hình hóa các hành vi rời rạc

Áp dụng các luật chuyển đổi đã trình bày trong Mục 5.2.2, tập các yêu cầu của hệ thống được chuyển đổi sang mô hình Event-B như sau:

- Áp dụng *Luật 1*: Fuzzy hedges, generators và các giá trị trong tập các yêu cầu được chuyển thành các tập HEDGES, DISTANCE và POWER trong ngữ cảnh Event-B $Crane\_C0$.
- Áp dụng *Luật 2*: Các hàm thuộc của hedges vàvà giá trị mờ được biểu diễn như các hàm số nguyên. Ví dụ: $h\_deg : HEDGES \rightarrow \mathbb{N}$. Ta có axiom cho giá trị này như sau $h\_deg(very) = 3 \wedge h\_deg(quite) = 2 \wedge h\_deg(precise) = 1$.
- Áp dụng *Luật 3*: Các biến trong yêu cầu được chuyển thành các thành phần Event-B như *distance* and *power*. Kiểu của hai biến này được biểu diễn như các bất biến $inv1$ and $inv2$.
- Áp dụng *Luật 4*: Mỗi yêu cầu không chính xác FR_i của hệ thống được chuyển thành một sự kiên evt_i, $i = \overline{1,5}$.

5.4.2.2 Mô hình hóa các hành vi liên tục

Ta làm mịn mô hình chuyển đổi ở mục trước bằng việc xem xét hệ thống một cách chi tiết và cụ thể hơn. Trên thực tế, mỗi di chuyển của cần cẩu gắn với trục thời gian vì nó di chuyển một cách liên lục trong khi tốc độ thì được điều chỉnh rời rạc. Ta áp dụng các luật được đưa ra trong mục 5.2.3 như sau:

- Áp dụng *Luật 5*: Năm sự kiện được làm mịn trong máy làm mịn $Crane\_M1$, biến đếm thời gian t cũng được bổ sung.

- Áp dụng *Luật 6*: Biến *dis* được thay bằng dis_c (vì khoảng cách đến container là phụ thuộc thời gian). Ràng buộc của biến mới của máy làm mịn và biến cũ thể hiện qua bất biến $inv3$.

5.4.2.3 Làm mịn: Mô hình hóa tính chất kết quả

CONTEXT Cranel_C1
EXTENDS Crane_C0
CONSTANTS
deg_HED, deg_POWER, d_DIS
AXIOMS
axm4 : deg_HED : HEDGES → ℕ
axm5 : deg_HED(very) = 3 ∧ deg_HED(quite) = 2
∧ deg_HED(precise) = 1
END

INVARIANTS
inv1 : d ∈ ℕ
DELF : d = 0 ⇒ ran(dist) = {start} ∨ ran(dist) = {far} ∨
ran(dist) = {medium} ∨ ran(dist) = {close} ∨ ran(dist) =
{above}
EVENTS

Event *evt4_CE* ≙
Status convergent
extends evt_4
when
grd1 : distance = {precise ↦ close}
grd2 : d = deg_DIS(close)
grd3 : ¬d = deg_DIS(above)
then
act1 : power := {very ↦ slow}
act2 : distance := {precise ↦ above}
act2 : d := deg_DIS(above)
end

Event *evt4_OE* ≙
Status ordinary
extends evt_4
when
grd1 : distance = {precise ↦ close}
grd2 : d = deg_DIS(close)
grd3 : d = deg_DIS(above)
then
act1 : power := {very ↦ slow}
act2 : distance := {precise ↦ above}
act2 : d := deg_DIS(above)
end

5.4.3 Kiểm chứng các thuộc tính

Tính chất an toàn của hệ thống đã được hình thức hóa thành một bất biến $inv4 : ran(dist) = \{close\} \Rightarrow \neg ran(power) = \{fast\}$. Một mệnh đề cần chứng minh này được sinh ra cho một sự kiện của máy $Crane\_M0$. Bảng 5.1 thể hiện mệnh đề cần chứng minh $inv4$ của sự kiện $evt4$

Bảng 5.1: INV PO of event $evt4$

$ran(dis) = \{close\} \Rightarrow \neg ran(speed) = \{fast\}$ $dis = \{precise \mapsto close\}$ $\vdash$ $ran\left(\{precise \mapsto above\}\right) = \{close\} \Rightarrow \neg ran\left(\{very \mapsto slow\}\right) = \{fast\}$

Ta cần phải chứng minh cần cẩu sẽ đến vị trí ở phía trên container, có nghĩa là $Crane\_M1 \vdash \Box\Diamond(d = deg\_DIS(above))$. Tính chất không tắc nghẽn của máy được biểu diễn thành một định lý $DELF$ in $Crane\_M1$ (Bảng 5.2).

Bảng 5.2: Mệnh đề cần chứng minh tính chất không tắc nghẽn

$d = deg\_DIS(above)$ $\Rightarrow$ $d = deg\_DIS(start) \vee d = deg\_DIS(far)$ $d = deg\_DIS(medium) \vee d = deg\_DIS(close)$ $d = deg\_DIS(above)$	DELF/THM

Để kiểm tra tính hội tụ, ta chứng minh các mệnh đề cần chứng minh cho các sự kiện hội tụ, cụ thể là evt_i/NAT và evt_i/VAR. Bảng 5.3 là mệnh đề cần chứng minh sự kiện $evt4$ của máy $Crane\_M1$ giảm giá trị của variant d.

Bảng 5.3: Mệnh đề cần chứng minh tính hội tụ

$dis = \{precise \mapsto close\}$ $\neg d = deg\_DIS(close)$ $d = deg\_DIS(close)$ $\vdash$ $d - (deg\_DIS(close) - deg\_DIS(above)) < d$	evt4_CE/VAR

Chương 6. Kết luận

6.1 Các kết quả đạt được

Luận án đã đạt được các mục tiêu ban đầu đề ra. Trong phần đầu tiên của luận án, thay vì làm việc trên một mô hình tham chiếu của kiến trúc hướng sự kiện trừu tượng hơn và mô tả một lớp hơn của hệ thống, luận án tập trung vào các ứng dụng đặc trưng là hệ thống cơ sở dữ liệu sử dụng triggers và cảm ngữ cảnh. Hai ứng dụng này có tính chất và chức năng riêng. Tuy nhiên, hai hệ thống có một đặc điểm chung là trigger và các quy tắc ngữ cảnh có cấu trúc tương tự nhau và ở dạng ECA. Phương pháp đề xuất của luận án được dựa trên sự tương đồng giữa cấu trúc ECA và một sự kiện Event-B. Vì lý do này, quá trình xây

dựng mô hình là tự nhiên và dễ dàng. Hơn nữa, sau khi hình thức hóa hệ thống, chúng ta có thể kiểm chứng các tính chất của hệ thống mà không cần thêm bất kỳ bước trung gian nào. Một công cụ Trigger2B cũng đã được xây dựng để hỗ trợ quá trình chuyển đổi tự động từ hệ thống CSDL sang mô hình Event-B.

Trong phần thứ hai, luận án có những đóng góp cụ thể vào việc phân tích hệ thống hướng sự kiện được đặc tả bởi các yêu cầu không chính xác. Mặc dù yêu cầu không chính xác thường được tìm thấy trong quá trình phát triển phần mềm, nhưng từ trước đến nay những nghiên cứu chủ yếu giải quyết vấn đề mô hình và kiểm chứng các yêu cầu chính xác. Luận án trình bày phương pháp mới dựa trên qui tắc làm mịn, trong đó các yêu cầu được mô tả bởi If-Then mờ được chuyển đổi sang một tập hợp các sự kiện Event-B. Luận án cũng đề xuất phương pháp dựa trên làm mịn để có thể kiểm chứng tính chất an toàn và kết quả của hệ thống.

6.2 Hạn chế

- Phương pháp đề xuất cho mô hình và kiểm chứng các hệ thống cơ sở dữ liệu không hỗ trợ cho suy diễn trực tiếp về thuộc tính kết thúc, trong khi nó là một trong những đặc tính mong muốn mà người phát triển phần mềm muốn kiểm tra. Phương pháp xử lý các trường hợp đơn giản của một chuỗi các câu lệnh DML không chứa lồng nhau và cú pháp trigger đầy đủ.
- Do thiếu sự hỗ trợ kiểu dữ liệu nguyên thủy trong Event-B, luận án chỉ có thể làm phong phú thêm mô hình dữ liệu ngữ cảnh bằng cách kết hợp với các Plug-in mới. Ngữ cảnh là dữ liệu thường rất phức tạp và có nhiều loại dữ liệu. Hơn nữa, trong thực tế một ứng dụng cảm ngữ cảnh thường chứa dữ liệu thời gian có liên quan. Tuy nhiên, Event-B không hỗ trợ logic thời gian, do đó mô hình và kiểm chứng các ứng dụng như vậy sẽ phải đối mặt với một số vấn đề. Phương pháp đề xuất cần phải được mở rộng để mô hình biến phụ thuộc thời gian.
- Phương pháp mô hình hóa và kiểm chứng yêu cầu hệ thống không chính xác cả hai trường hợp hành vi rời rạc và liên tục của hệ thống. Tuy nhiên, do một số hạn chế của Rodin, luận đã giới thiệu một loại xấp xỉ sử dụng $\mathbb{N}$ thay vì $\mathbb{R}$. Hơn nữa, các thuộc tính liên quan đến thời gian chưa được kiểm chứng. Mô tả hành vi của hệ thống bằng các luật If-Then mờ là chưa đủ cho mọi trường hợp. Bên cạnh đó tính tình huống, có một số tính chất hoạt động quan trọng của hệ thống như tính tiến trình (progress), tính bền bỉ (persitence) chưa được phân tích.

6.3 Hướng phát triển

Một trong những hướng nghiên cứu luận án là phát triển công cụ Trigger2B thành một plugin của nền tảng Rodin. Chúng tôi cũng sẽ xử lý các triggers phức tạp hơn với nhiều câu lệnh DML lồng nhau kết hợp với vòng lặp. Trong

trường hợp lệnh phức hợp lồng nhau, chúng tôi có thể cần phải áp dụng các kỹ thuật tổ hợp để mô hình thành phần là các sự kiện phức hợp. Suy diễn về tính dừng của các trigger và các loại trigger khác cũng sẽ được xem xét trong các nghiên cứu tiếp theo.

Chúng tôi sẽ mở rộng phương pháp mô hình hóa các hệ thống cảm ngữ cảnh để tạo ra và xác định ngữ nghĩa cho các loại dữ liệu khác nhau của bối cảnh thường xuyên được sử dụng như: thời gian, địa điểm. Phương pháp đề xuất sẽ được mở rộng để mô hình hóa mối quan hệ phức tạp hơn của các ngữ cảnh. Công việc tương lai của chúng tôi cũng sẽ tập trung vào phân tích thuộc tính liên quan đến thời gian của các hệ thống cảm ngữ cảnh và mở rộng các kiểu dữ liệu ngữ cảnh cho phong phú hơn bằng việc tích hợp các plug-in.

Các phương pháp hiện tại để chứng minh tính hoạt động (liveness) được thực hiện ở quá trình làm mịn cuối cùng. Vì vậy, một cải tiến mà làm để phương pháp có thể để chứng minh tính hoạt động ở mọi giai đoạn làm mịn cũng là một trong các hướng nghiên cứu. Mở rộng các vấn đề về lý thuyết trong suy diễn của Event-B để có thể kiểm chứng các tính chất hoạt động quan trọng khác như tính chất bền bỉ và tiến trình là hướng nghiên cứu trong tương lai của luận án.

CÁC CÔNG TRÌNH TRÌNH KHOA HỌC CÔNG BỐ

1. Hong Anh Le and Ninh Thuan Truong. Modeling and Verifying WS-CDL Using Event-B. In Proc. ICCASA 2012. LNICST Vol 109, pp. 290-299, Springer, 2013.

2. Hong Anh Le and Ninh Thuan Truong: Modeling and Verifying DML Triggers Using Event-B, In Proc. ACIIDS 2013. LNCS Vol 7083, Vol 2, pp. 539-548, Spinger, 2013.

3. Hong Anh Le, Loan Dinh Thi and Ninh Thuan Truong: Modeling and Verifying Imprecise Requirements of Systems Using Event-B. In Proc. KSE 2013. AISC Vol 244, pp. 313-325, Springer, 2013.

4. Hong Anh Le and Ninh Thuan Truong: Formal Modeling and Verification of Context-Aware Systems Using Event-B. In Proc. ICCASA 2013. LNICST Vol 128, pp. 250-259, Springer 2014 (**The best paper award**).

5. Hong Anh Le and Ninh Thuan Truong: Formal Modeling and Verification of Context-Aware Systems Using Event-B. In EAI Endorsed Transactions on Context-Aware Systems and Applications. Vol2, e4, 2014. ISSN 2409-0026.

6. Hong Anh Le, Ninh Thuan Truong and Shin Nakajima: Verifying Eventuality Properties of Imprecise System Requirements using Event-B. In Proc. The 30th ACM/SIGAPP Symposium On Applied Computing - Software Engineering Track, Salamanca, Spain. Volume 2, pp. 1651-1654, ACM Press.