Mục lục

Danh sách hình vẽ

Danh sách bảng

Bảng các từ viết tắt

Từ viết tắt	**Từ gốc**	**Giải nghĩa-Tạm dịch**
AA	Asynchronous Automata	Ô-tô-mát bất đồng bộ
ADA	Asynchronous Duration Automata	Ô-tô-mát khoảng bất đồng bộ
CBSE	Component-based Software Enginnering	Công nghệ phần mềm dựa trên thành phần
CBSD	Component-based Software Development	Phát triển phần mềm dựa trên thành phần
DTS	Distributed Transition Systems	Hệ dịch chuyển phân tán
DDTS	Duration Distributed Transition Systems	Hệ dịch chuyển phân tán khoảng
TCIA	Timed Concurrent Interface Automata	Ô-tô-mát giao diện tương tranh thời gian
UTP	Unifying Theories of Programming	Lý thuyết hợp nhất về lập trình (tạm dịch)
TA	Timed Automata	Ô-tô-mát thời gian
LTL	Linear Temporal Logic	Logic thời gian tuyến tính
TLTL	Timed Linear Temporal Logic	Logic thời gian tuyến tính có ràng buộc thời gian
rCOS	Refinement of Component and Object Systems	Làm mịn thành phần và các hệ thống đối tượng
COM	Component Object Model	Mô hình đối tượng thành phần
DCOM	Distributed Component Object Model	Mô hình đối tượng thành phần phần tán
CORBA	Common Object Request Broker Architecture	
ORB	Object Request Broker	
OMG	Object Management Group	Tập đoàn quản lý đối tượng
COTS	Component Off The Shelf	Thành phần thương mại có sẵn
TW	Timed Word	Từ thời gian

Từ viết tắt	**Từ gốc**	**Giải nghĩa-Tạm dịch**
RTS	Real-time systems	Các hệ thời gian thực
CCS	Calculus of Communicating Systems	Tính toán các hệ thống giao tiếp
CTL	Computational Tree Logic	Logic cây tính toán
TCTL	Timed Computational Tree Logic	Logic cây tính toán có thời gian
$Proc$	Process	Ký hiệu tiến trình
$wtot$	word to trace	Ký hiệu hàm chuyển từ "từ" sang "Vết"
$ttow$	trace to word	Ký hiệu hàm chuyển từ "Vết" sang "từ"
BA	Büchi Automata	Ô-tô-mát Bu-khi
$conf$	Configuration	Ký hiệu cấu hình
CG	Configuration Graph	Ký hiệu đồ thị cấu hình
WCET	Worst-case Execution Time	Thời gian thực thi yếu nhất
$dtot$	duration to timed	Ký hiệu hàm chuyển thời khoảng sang thời điểm
$tTrL$	timed Trace Language	Ngôn ngữ Vết thời gian
$intv$	inteval	Ký hiệu khoảng thời gian
$pref$	prefix	Ký hiệu tiền tố
$Proj$	Project	Ký hiệu phép chiếu
dur	duration	Ký hiệu khoảng
Ctr	Contract	Hợp đồng
$Comp$	Component	Thành phần
$Behav$	Behavior	Hành vi
$ActComp$	Active Component	Thành phần chủ động
$SysCtr$	System Contract	Hợp đồng hệ thống

Lời cam đoan

Tôi xin cam đoan đây là công trình nghiên cứu do tôi thực hiện dưới sự hướng dẫn của TS. Đặng Văn Hưng và PGS.TS. Nguyễn Việt Hà tại bộ môn Công nghệ Phần mềm, Khoa Công nghệ Thông tin, Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội. Các số liệu và kết quả trình bày trong luận án là trung thực, chưa được công bố bởi bất kỳ tác giả nào hay ở bất kỳ công trình nào khác.

Tác giả

Lời cảm ơn

Luận án này được thực hiện tại Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội dưới sự hướng dẫn khoa học của TS Đặng Văn Hưng và PGS.TS. Nguyễn Việt Hà. Nghiên cứu sinh xin bày tỏ lòng biết ơn sâu sắc tới các Thầy về định hướng khoa học, sự quan tâm, hướng dẫn và các chỉ bảo kịp thời cho các hướng nghiên cứu, tạo điều kiện thuận lợi trong suốt quá trình nghiên cứu tại trường.

Nghiên cứu sinh cũng xin cảm ơn tới các thày cô trong Bộ môn Công nghệ Phần mềm. Trong quá trình thực hiện luận án, nghiên cứu sinh đã nhận được sự giúp đỡ nhiệt tình và sự động viên kịp thời của các thầy cô, các nhà khoa học. Đây là nguồn động lực lớn để tôi có thể hoàn thành luận án.

Nghiên cứu sinh xin trân trọng cảm ơn Lãnh đạo Trường Đại học Công nghệ, Đại học Quốc Gia Hà Nội đã tạo những điều kiện tốt nhất để nghiên cứu sinh có được môi trường nghiên cứu tốt nhất và hoàn thành chương trình nghiên cứu của mình. Xin chân thành cám ơn Khoa Công nghệ Thông tin, Phòng Đào tạo và đào tạo sau đại học và các nhà khoa học thuộc trường Đại học Công nghệ cũng như các nghiên cứu sinh khác về sự hỗ trợ trên phương diện hành chính, hợp tác có hiệu quả trong suốt quá trình nghiên cứu khoa học của mình.

Nghiên cứu sinh xin gửi lời cảm ơn tới Ban Lãnh đạo Trường Đại học Dân lập Hải Phòng, Khoa Công nghệ Thông tin và các bạn đồng nghiệp vì đã tạo nhiều điều kiện thuận lợi hỗ trợ cho nghiên cứu sinh có thời gian và toàn tâm thực hiện triển khai đề tài nghiên cứu của luận án. Nghiên cứu sinh cũng xin trân trọng cảm ơn các nhà khoa học, tác giả các công trình công bố đã trích dẫn trong luận án vì đã cung cấp nguồn tư liệu quý báu, những kiến thức liên quan trong quá trình nghiên cứu hoàn thành luận án.

Cuối cùng là sự biết ơn tới Bố Mẹ, vợ con, các anh chị em trong gia đình và những người bạn thân thiết đã liên tục động viên để duy trì nghị lực, sự cảm thông, chia sẻ về thời gian, sức khỏe và các khía cạnh của cuộc sống trong cả quá trình hoàn thành luận án.

Tóm tắt

Chất lượng dịch vụ của một hệ thống bao gồm thời gian tiến hành, tài nguyên tiêu thụ và độ tin cậy của dịch vụ, trong đó thì chất lượng dịch vụ về thời gian đang được quan tâm nhiều, thể hiện rằng thời gian cung ứng dịch vụ tốt hơn. Ràng buộc thời gian trong các hệ thống thường được phân chia thành hai loại là ràng buộc thời gian cứng (hard) và mềm (soft). Luận án quan tâm tới các ràng buộc thời gian cứng. Để chất lượng dịch vụ tốt, các phương thức trong hệ thống cần được tiến hành song song (tăng tốc độ đáp ứng) nếu có thể và phải có ràng buộc thời gian rõ ràng. Ràng buộc thời gian thể hiện thời gian tối thiểu và tối đa mà phương thức cần để có thể cung cấp dịch vụ, tức là không được gọi phương thức quá "dày"[1], nếu không có thể sẽ gây ra tình trạng dịch vụ không đáp ứng được. Luận án quan tâm tới phương pháp đặc tả hệ thống có chứa chất lượng dịch vụ về thời gian.

Đối tượng nghiên cứu của luận án là các hệ thống phần mềm dựa trên thành phần có tính tương tranh và có ràng buộc về thời gian. Tính tương tranh là một thuộc tính của hệ thống trong đó một số dịch vụ của hệ thống được cho phép truy cập một cách song song. Ràng buộc về thời gian trong luận án là các yêu cầu về thời gian thực thi của các hành động trong hệ thống, mỗi hành động sẽ được gắn với một khoảng thời gian cho việc thực thi của nó.

Mục đích của luận án là phát triển một phương pháp hình thức để đặc tả và kiểm chứng các giao diện của các thành phần phần mềm có tính tương tranh và ràng buộc về thời gian. Sau đó, luận án áp dụng phương pháp được đề xuất vào việc đặc tả, phân tích và kiểm chứng các mô hình khác nhau của các hệ thống phần mềm dựa trên thành phần.

Các kết quả của luận án đạt được như sau. Luận án đề xuất lý thuyết Vết thời gian để hỗ trợ đặc tả các ràng buộc về thời gian trên các hệ thống tương tranh thời gian thực. Vết thời gian là một sự mở rộng về thời gian của Vết Mazurkiewicz bằng việc bổ sung vào Vết Mazurkiewicz một hàm gán nhãn thời gian. Với việc mở rộng này, Vết thời gian có thể dễ dàng đặc tả các hành vi của hệ thống tương tranh có ràng buộc thời gian. Trong lý thuyết này, luận án còn đề xuất khái niệm Vết khoảng. Vết khoảng là các Vết Mazurkiewicz mà mỗi

[1]Mật độ cao trên một đơn vị thời gian

ký hiệu (hành động) trong bảng chữ cái phụ thuộc được gán một ràng buộc là một khoảng thời gian. Vết khoảng được sử dụng để biểu diễn các ràng buộc thời gian của các hệ thống mà mỗi hành động của các hệ thống này có ràng buộc về khoảng thời gian hoạt động và cung cấp dịch vụ. Vết khoảng và Vết thời gian có mối quan hệ với nhau, Vết khoảng là biểu diễn ngắn gọn của một tập các Vết thời gian. Luận án cũng đưa vào ô-tô-mát khoảng bất đồng bộ làm công cụ đoán nhận lớp ngôn ngữ Vết thời gian chính quy để sử dụng trong các bài toán về kiểm chứng hệ thống. Một kết quả trong luận án là bài toán kiểm tra tính rỗng của ô-tô-mát khoảng bất đồng bộ là quyết định được dù độ phức tạp không phải là đa thức. Để hỗ trợ việc biểu diễn đặc tả các thuộc tính cần kiểm chứng của các hệ thống, trong lý thuyết Vết thời gian, luận án đưa vào lôgic thời gian thực tuyến tính đặc tả thuộc tính của các Vết thời gian. Lôgic này là một mở rộng về thời gian của lôgic thời gian tuyến tính (LTL - Linear Temporal Logic). Mối quan hệ giữa ô-tô-mát khoảng bất đồng bộ và lôgic này cũng được đề cập và chứng minh. Như vậy, với lý thuyết Vết thời gian đề xuất, các hệ thống tương tranh có ràng buộc thời gian sẽ dễ dàng được đặc tả và kiểm chứng bằng các ô-tô-mát khoảng bất đồng bộ và các công thức của lôgic thời gian thực tuyến tính.

Để minh chứng cho tính hiệu quả của phương pháp được đề xuất, luận án áp dụng phương pháp này vào việc đặc tả, phân tích và kiểm chứng cho ba mô hình ứng dụng thiết kế hệ thống dựa trên thành phần. Với mỗi mô hình, các hành vi của hệ thống được đặc tả thông qua các Vết thời gian. Như vậy, các mô hình này có thể đặc tả được các tính chất tương tranh và ràng buộc về thời gian của các hệ thống cần kiểm chứng. Thứ nhất, luận án giới thiệu một mô hình hệ thống tương tranh thời gian dựa trên lý thuyết rCOS (Refinement of Component and Object Systems). Nghiên cứu này sử dụng Vết thời gian trong đặc tả các thể thức giao diện thành phần. Các tính toán về phép ghép nối, phương pháp làm mịn thành phần được đưa ra và chứng minh. Thứ hai, luận án đề xuất một mô hình thiết kế dựa trên giao diện cho các hệ tương tranh. Trong mô hình này, luận án sử dụng ô-tô-mát giao diện tương tranh thời gian để đặc tả mỗi thành phần. Các kết quả trong nghiên cứu đã chỉ ra rằng phương pháp mới đảm bảo tất cả các yêu cầu của lý thuyết thiết kế dựa trên giao diện. Thứ ba, luận án đã giới thiệu một phương pháp là một mô hình hỗ trợ đặc tả và kiểm chứng cho hệ thống phân tán. Ý tưởng của phương pháp là mở rộng hệ dịch chuyển phân tán, sử dụng Vết thời gian để đặc tả ngôn ngữ và chỉ ra ra mối quan hệ tương đương giữa Vết thời gian và hệ dịch chuyển phân tán tương đương về ngôn ngữ.

Các kết quả trong luận án đã được công bố qua các công trình đã được xuất bản và có đóng góp phần nào vào việc nghiên cứu, đặc tả và kiểm chứng các hệ thống có tính tương tranh và ràng buộc về thời gian.

Chương 1

Giới thiệu

1.1 Đặt vấn đề

Công nghệ phần mềm dựa trên thành phần là một trong những sáng kiến kỹ thuật quan trọng nhất trong công nghệ phần mềm vì nó được coi là một cách tiếp cận mở, hiệu quả trong việc giảm chi phí và thời gian phát triển phần mềm trong khi vẫn được đảm bảo được chất lượng của sản phẩm [59]. Với cách tiếp cận này, hệ thống được xây dựng bằng việc ghép nối các thành phần phần mềm có sẵn. Mỗi thành phần riêng lẻ là một gói phần mềm, một dịch vụ Web hoặc một mô-đun được đóng gói và chúng giao tiếp với nhau thông qua các giao diện. Tuy nhiên, phương pháp này vẫn còn một số vấn đề được đặt ra [59].
Thứ nhất là làm sao đảm bảo được các thành phần hoạt động được khi ghép nối với nhau? Thứ hai là làm thế nào để có thể mở rộng các thành phần từ thành phần có sẵn, và quan trọng nhất là làm thế nào để đảm bảo chất lượng hệ thống tức là hệ thống sau khi xây dựng phải thỏa mãn các ràng buộc được đưa ra trong quá trình đặc tả ban đầu?

Để giải quyết các vấn đề trên, một trong các giải pháp thông dụng và hiệu quả được sử dụng là áp dụng các phương pháp hình thức. Đây là phương pháp sử dụng các mô hình toán học cho việc đặc tả, phát triển và kiểm chứng các hệ thống phần mềm và phần cứng [8, 22, 44, 57]. Cách tiếp cận này đặc biệt quan trọng đối với các hệ thống có yêu cầu chất lượng cao, chẳng hạn hệ thống điều khiển lò phản ứng hạt nhân hay điều khiển tên lửa, hệ thống điều khiển cần gạt ga tàu,v.v. Trong các hệ thống này, vấn đề an toàn hay an ninh có vai trò quan trọng để góp phần đảm bảo rằng quá trình phát triển hệ thống sẽ không có lỗi.

Các phương pháp hình thức đặc biệt hiệu quả tại giai đoạn đầu của quá trình phát triển (tại giai đoạn phân tích thiết kế), nhưng cũng có thể được sử dụng cho các giai đoạn sau của quá trình xây dựng và phát triển hệ thống.

Để một hệ thống có hiệu quả cao về chất lượng dịch vụ (chất lượng dịch vụ tốt), các phương thức cần được tiến hành song song để tăng khả năng và tốc độ tính toán (Vết Mazurkiewicz đã đặc tả được đặc tính này). Mặt khác, thực tế là chúng ta không nên gọi phương thức quá "dày", nếu không có thể sẽ gây ra tình trạng dịch vụ không đáp ứng được. Một ví dụ nhỏ trên thực tế về vấn đề này mà chúng ta hay gặp phải là thao tác mở tệp tin (hoặc chương trình) trên máy tính (hệ điều hành windows). Chúng ta nhấn đúp chuột để chạy một chương trình nào đó, nếu chưa thấy chương trình thực hiện, nhiều người sẽ lại nhấn đúp để mở lên và thao tác này được lặp lại nhiều lần cho tới khi trên màn hình hiển thị chương trình cần mở. Lúc này sẽ có nhiều bản sao của chương trình chạy lên (do nhấn nhiều lần) và người dùng lại phải đóng bớt đi (thường để lại một bản). Nguyên nhân của việc này là chương trình đó cần một khoảng thời gian để thực hiện nhưng người dùng không biết (không chờ) mà làm thao tác mở liên tục (gọi chương trình quá "dày"). Như vậy chúng ta cần có ràng buộc về thời gian cho mỗi phương thức thể hiện thời gian tối thiểu và tối đa phương thức cần để thực hiện và đáp ứng yêu cầu dịch vụ. Do đó, thêm vào các ràng buộc thời gian cho vết Mazurkiewicz là hợp lý và phương pháp đặc tả hệ thống sử dụng vết Mazurkiewicz mở rộng về thời gian là có chứa chất lượng dịch vụ (về khía cạnh thời gian).

Đối tượng nghiên cứu trong luận án là các hệ thống phần mềm dựa trên thành phần có tính tương tranh và có ràng buộc thời gian hay gọi một cách ngắn gọn là hệ tương tranh có ràng buộc (có yếu tố) thời gian nhằm tận dụng các thế mạnh của cách tiếp cận phát triển phần mềm dựa trên thành phần và các phương pháp hình thức trong phát triển phần mềm. Trong các hệ thống, tính tương tranh là một thuộc tính của hệ thống trong đó một số dịch vụ của hệ thống được cho phép truy cập một cách song song. Ràng buộc về thời gian trong luận án là các yêu cầu về thời gian thực thi của các hành động trong hệ thống, mỗi hành động sẽ được gắn với một khoảng thời gian cho việc thực thi của nó. Như vậy, hệ tương tranh có ràng buộc thời gian trong luận án này là các hệ thống mà các hành động có thể thực hiện một cách song song, có sử dụng dịch vụ của nhau và thêm các ràng buộc về khoảng thời gian thực hiện của hành

động. Đây là một lớp bài toán nhỏ trong công nghệ phát triển phần mềm dựa trên thành phần nhưng lại có ứng dụng rất lớn và đòi hỏi độ tin cậy cao. Do đó, mục tiêu của luận án là phát triển một phương pháp hình thức để đặc tả và kiểm chứng các giao diện của các thành phần phần mềm có tính tương tranh và ràng buộc về thời gian. Sau đó, luận án áp dụng phương pháp được đề xuất vào việc đặc tả, phân tích và kiểm chứng các mô hình khác nhau của các hệ thống phần mềm dựa trên thành phần.

Hiện nay đã có một số các phương pháp đề xuất để giải quyết bài toán đặc tả và kiểm chứng hệ thống nhưng hiếm thấy phương pháp nào đặc tả đồng thời tính tương tranh và ràng buộc thời gian. Các phương pháp chung thường sử dụng các mô hình là các hệ dịch chuyển trạng thái hay các ô-tô-mát để đặc tả các hoạt động của một thành phần (hệ thống). Những phương pháp này đã có những kết quả tốt và được sử dụng rộng rãi nhưng không tập trung và hỗ trợ vào đặc tả thời gian cũng như tính tương tranh mà quan tâm tới việc làm thế nào để có thể mô hình được hệ thống như trong [56, 71, 72]. Một số phương pháp khác sử dụng các mô hình ngẫu nhiên [6], trong đó các nghiên cứu tập trung vào việc phát triển lý thuyết chuỗi Markov, chuỗi Markov thời gian và mô hình xác suất nhằm xử lý các hệ thống xác suất (bao gồm cả các ràng buộc thời gian và tương tranh).
Đây là một kiểu hệ thống có nhiều ứng dụng lớn, tuy nhiên mục tiêu của luận án không tập trung vào các hệ thống này.

Để đặc tả các hệ tương tranh, công cụ được sử dụng phổ biến là lý thuyết Vết Mazurkiewicz [30, 55]. Các khái niệm về Vết Mazurkiewicz (gọi tắt là Vết) đã được giới thiệu để mô tả hành vi không liên tục của hệ thống song song thông qua các quan sát liên tục của nó. Vết biểu diễn quá trình đồng thời theo cùng một cách như các chuỗi biểu diễn một cách tuần tự. Lý thuyết về Vết được sử dụng như một công cụ để suy luận về quy trình không liên tục; là một vấn đề của thực tế sử dụng lý thuyết này người ta có thể có được một tính toán của các tiến trình tương tranh tương tự như những cái có sẵn cho hệ thống tuần tự. Trong lý thuyết này, các khái niệm cơ bản được đề xuất là quan hệ phụ thuộc-độc lập, ngôn ngữ Vết, ô-tô-mát đoán nhận ngôn ngữ Vết cũng như lôgic thời gian tuyến tính biểu diễn Vết được đặc biệt quan tâm và có nhiều nghiên cứu hỗ trợ, mở rộng như trong [14, 31, 50, 68]. Điều này khẳng định lý thuyết Vết là công cụ có hiệu quả trong đặc tả các hệ tương tranh bởi tính đơn giản,

tiện dụng của phương pháp. Tuy nhiên công cụ này có hạn chế là không hỗ trợ đặc tả các ràng buộc thời gian. Đây là một kiểu ràng buộc được yêu cầu trong nhiều ứng dụng quan trọng và phổ biến. Những hệ có ràng buộc này được gọi là các hệ thời gian thực hoặc đơn giản hơn là các hệ thống có ràng buộc thời gian.

Do đó, nghiên cứu về đặc tả các hệ có các ràng buộc thời gian được nhiều nhà nghiên cứu quan tâm xem xét và đề xuất các phương pháp đặc tả và kiểm chứng. Hầu hết các đề xuất này đều dựa trên lý thuyết về các ô-tô-mát thời gian (TA - Timed Automata) [4, 11]. Lý thuyết này được coi là công cụ không thể thiếu khi nghiên cứu về các hệ thống thời gian thực. Ứng dụng ô-tô-mát thời gian, người phát triển hệ thống sẽ mô tả mỗi thành phần bằng một ô-tô-mát và hệ thống sẽ là một mạng các ô-tô-mát được ghép nối với nhau theo một cách thức phù hợp. Các kết quả về lý thuyết cũng như các thuật toán chứng minh đã được nghiên cứu và giới thiệu. Tuy nhiên, phương pháp này và các mở rộng như trong [46, 35] lại rất khó khăn trong việc mô tả các ràng buộc về tính tương tranh. Chi tiết về lý thuyết Vết và ô-tô-mát thời gian sẽ được giới thiệu trong chương 2. Đây là hai lý thuyết nền tảng trong nghiên cứu của luận án này. Một số nghiên cứu khác đã sử dụng khái niệm Vết thời gian (timed traces) để mô tả hành vi hệ thống như trong [12, 73] và một số nghiên cứu liên quan. Trong các nghiên cứu này, Vết thời gian được định nghĩa nhằm giải các bài toán về kiểm chứng mô hình (Model checking) như việc sử dụng Vết thời gian được định nghĩa trên tập các đồng hồ của ô-tô-mát thời gian nhằm giải bài toán về giảm bùng nổ không gian trạng thái vùng trong ô-tô-mát thời gian hoặc giải các bài toán kiểm chứng cho mô hình mạch bất đồng bộ. Các nghiên cứu này khác với nghiên cứu trong luận án về mục tiêu. Trong luận án, mục đích các nghiên cứu là đề xuất phương pháp đặc tả các giao diện của các thành phần. Tuy nhiên, điều dễ thấy là các nghiên cứu này và nghiên cứu trong luận án có thể hỗ trợ nhau nhằm phát huy hết lợi ích mà lý thuyết Vết mang lại.

Gần đây, trong [27], các tác giả đã đề xuất một phương pháp đặc tả và kiểm chứng các giao diện thời gian thực. Trong nghiên cứu này, hệ thống được hình thành bởi ghép nối các giao diện thời gian thực với nhau, mỗi giao diện biểu diễn một thành phần hệ thống. Giao diện thời gian thực là giao diện với ràng buộc thời gian liên quan giữa thời điểm đầu ra với thời điểm đầu vào. Tại một thời điểm trong quá trình thực hiện, một hành vi của giao diện theo một hợp đồng được thực hiện với môi trường về chức năng của nó cũng như thời gian

thực hiện để đảm bảo thực hiện hợp đồng. Hợp đồng này được xác định như là một thiết kế theo thời gian sử dụng các ký hiệu trong UTP (Unifying Theories of Programming), và phụ thuộc vào lịch sử tính toán của giao diện. Nghiên cứu đưa ra mô hình phụ thuộc này như là một chức năng bộ phận từ lịch sử tính toán của giao diện tới hợp đồng theo thời gian thực. Các tác giả cũng đưa ra phương pháp để tạo giao diện mới từ việc ghép nối các giao diện lại với nhau, cách làm mịn các giao diện cũng như chỉ ra cách biểu diễn hữu hạn cho các giao diện. Tuy nhiên, hạn chế lớn nhất trong phương pháp này là chưa hỗ trợ đặc tả tính tương tranh trong hệ thống mà phương pháp chỉ quan tâm nhiều tới giải quyết mối quan hệ đầu vào đầu ra của yêu cầu giao diện. Một ứng dụng khác của Vết thời gian được đề xuất trong [23] để giải bài toán kiểm chứng động. Các tác giả sử dụng khái niệm hộp đen để mô tả các mô hình hệ thống, trong đó một hộp đen là hệ thống thực và một hộp đen là mô hình của hệ thống đó. Hai hệ thống này được thực hiện song song với cùng điều kiện đầu vào. Một bộ quan sát tự động sẽ nhận các đầu ra và đối sánh với nhau ngay tức thì. Các kết quả nghiên cứu trong này đã đưa ra một thuật toán cho bộ quan sát đối sánh hai Vết thời gian của hai mô hình trên.

Xuất phát từ các nhận định và nghiên cứu trên, luận án nghiên cứu và đề xuất một phương pháp hiệu quả trong việc đặc tả các giao diện thành phần của các hệ thống phần mềm dựa trên thành phần mà có các ràng buộc về thời gian và tính tương tranh. Luận án tập trung nghiên cứu bài toán về đặc tả và kiểm chứng các hệ thống tương tranh, các hệ thống có ràng buộc thời gian cũng như các phương pháp đã được sử dụng để giải bài toán đó. Trên cơ sở đó, với đối tượng nghiên cứu trong luận án, luận án tập trung nghiên cứu các phương pháp tối ưu nhất, xem xét các vấn đề và nghiên cứu áp dụng, mở rộng để đề xuất ra phương pháp mới, hiệu quả nhằm đạt được mục tiêu nghiên cứu. Luận án cũng áp dụng phương pháp đề xuất vào một số mô hình thiết kế thông dụng để minh chứng cho khả năng ứng dụng của phương pháp.

1.2 Các kết quả chính của luận án

Với mục đích nghiên cứu phương pháp đặc tả và kiểm chứng các giao diện thành phần cho các hệ thống dựa trên thành phần có chứa chất lượng dịch vụ và tính tương tranh, luận án đã đạt được các kết quả chính như sau.

Luận án đề xuất lý thuyết Vết thời gian để hỗ trợ đặc tả các ràng buộc về thời gian trên các hệ thống tương tranh thời gian thực. Vết thời gian là một sự mở rộng về thời gian của Vết Mazurkiewicz bằng việc bổ sung vào Vết Mazurkiewicz một hàm gán nhãn thời gian. Với việc mở rộng này, Vết thời gian có thể dễ dàng đặc tả các hành vi của hệ thống tương tranh có ràng buộc thời gian. Trong lý thuyết này, luận án còn đề xuất khái niệm Vết khoảng. Vết khoảng là các Vết Mazurkiewicz mà mỗi ký hiệu (hành động) trong bảng chữ cái phụ thuộc được gán một ràng buộc là một khoảng thời gian. Vết khoảng được sử dụng để biểu diễn các ràng buộc thời gian của các hệ thống mà mỗi hành động của các hệ thống này có ràng buộc về khoảng thời gian hoạt động và cung cấp dịch vụ. Vết khoảng và Vết thời gian có mối quan hệ với nhau, Vết khoảng là biểu diễn ngắn gọn của một tập các Vết thời gian. Luận án cũng đưa vào ô-tô-mát khoảng bất đồng bộ làm công cụ đoán nhận lớp ngôn ngữ Vết thời gian chính quy để sử dụng trong các bài toán về kiểm chứng hệ thống. Một kết quả trong luận án là bài toán kiểm tra tính rỗng của ô-tô-mát khoảng bất đồng bộ là quyết định được dù độ phức tạp không phải là đa thức. Để hỗ trợ việc biểu diễn đặc tả các thuộc tính cần kiểm chứng của các hệ thống, trong lý thuyết Vết thời gian, luận án đưa vào lôgic thời gian thực tuyến tính đặc tả thuộc tính của các Vết thời gian. Lôgic này là một mở rộng về thời gian của lôgic thời gian tuyến tính (LTL - Linear Temporal Logic). Mối quan hệ giữa ô-tô-mát khoảng bất đồng bộ và lôgic này cũng được đề cập và chứng minh. Như vậy, với lý thuyết Vết thời gian đề xuất, các hệ thống tương tranh có ràng buộc thời gian sẽ dễ dàng được đặc tả và kiểm chứng bằng các ô-tô-mát khoảng bất đồng bộ và các công thức của lôgic thời gian thực tuyến tính.

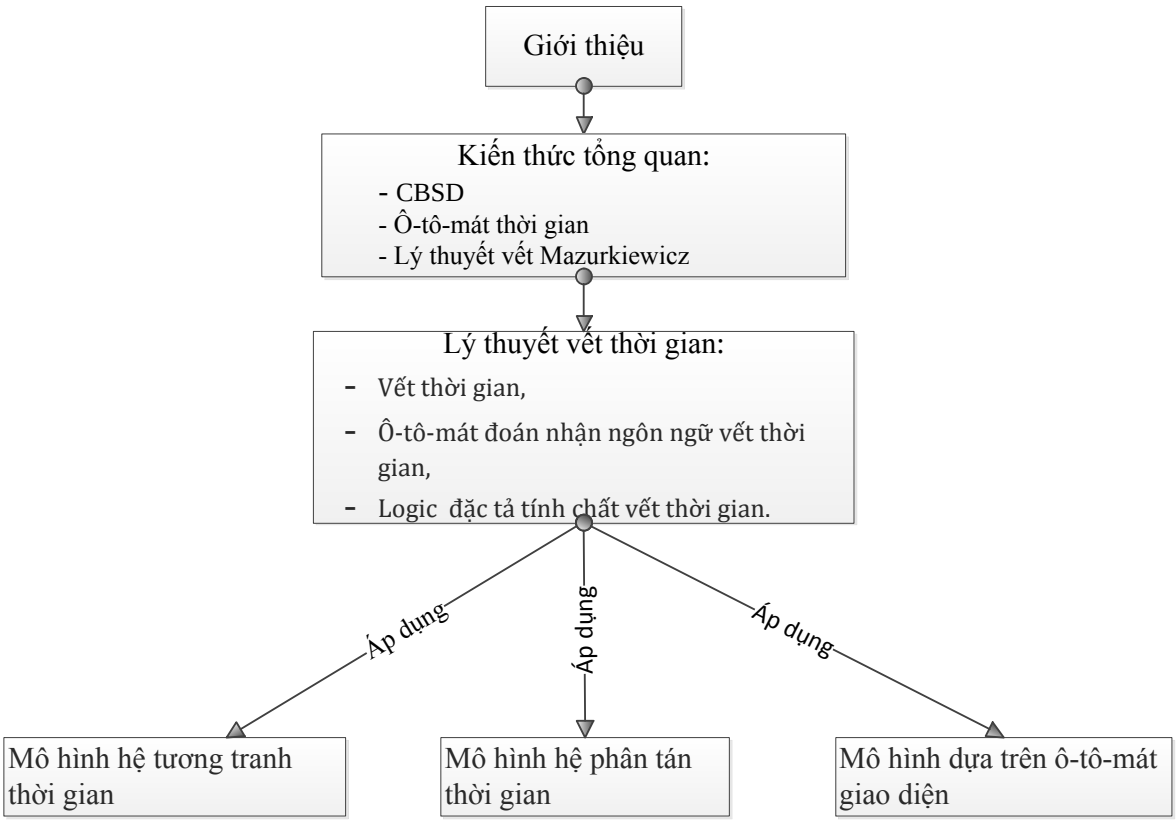
Để minh chứng cho tính hiệu quả của phương pháp được đề xuất, luận án áp dụng phương pháp này vào việc đặc tả, phân tích và kiểm chứng cho ba mô hình ứng dụng thiết kế hệ thống dựa trên thành phần. Với mỗi mô hình, các hành vi của hệ thống được đặc tả thông qua các Vết thời gian. Như vậy, các mô hình này có thể đặc tả được các tính chất tương tranh và ràng buộc về thời gian của các hệ thống cần kiểm chứng, cụ thể:

i. Luận án sử dụng các Vết thời gian để đặc tả các giao thức trong các giao diện thành phần nhằm hỗ trợ đặc tả các truy cập đồng thời có yếu tố thời gian tới các dịch vụ của một thành phần bằng việc mở rộng mô hình rCOS [74] về thời gian. Trong mô hình này, hệ thống được xây dựng bởi các thành

phần kết hợp với nhau. Một hợp đồng thành phần được định nghĩa bao gồm các đặc tả giao diện, các phương thức khởi tạo giá trị ban đầu, các đặc tả phương thức, các ràng buộc bất biến và các đặc tả giao thức tương tác là các Vết thời gian. Khi đó, một thành phần được định nghĩa như là một sự thực thi của một hợp đồng. Các kết quả chỉ ra mô hình trong nghiên cứu này hỗ trợ sự phân biệt giữa các ràng buộc chức năng và phi chức năng, và kiểm chứng hình thức kết hợp của các hệ thống thời gian dựa trên thành phần. Mô hình này cũng chỉ ra thuật toán tìm các giao thức cho các thành phần kết hợp và bài toán mở rộng hệ thống thông qua các phép toán làm mịn và ghép nối các thành phần.

ii. Tiếp theo, luận án đưa ra một mô hình thiết kế hệ thống dựa trên giao diện [3] cho các hệ tương tranh có ràng buộc thời gian, trong đó đề xuất khái niệm ô-tô-mát giao diện tương tranh có ràng buộc thời gian. Ô-tô-mát này một hạn chế của ô-tô-mát khoảng bất đồng bộ khi chỉ quan tâm các hành động đầu vào và đầu ra của hệ thống. Trong mô hình này, mỗi thành phần của hệ thống sẽ được đặc tả bởi một ô-tô-mát và hệ thống sẽ là một tích song song các ô-tô-mát. Các vấn đề về khả năng ghép nối các thành phần, cách thức ghép nối các thành phần, môi trường cho thành phần và làm mịn các thành phần cũng được đề xuất. Mô hình này cũng chứng minh việc đảm bảo hai tính chất cơ bản là thực thi độc lập và thiết kế gia tăng trong lý thuyết về thiết kế dựa trên giao diện.

iii. Cuối cùng, luận án cũng chỉ ra phương pháp đề xuất (lý thuyết Vết thời gian) cũng có khả năng áp dụng mở rộng cho các hệ thống tương tự như các hệ phân tán có ràng buộc thời gian bằng việc đưa ra mô hình hệ thống phân tán tương tranh có yếu tố thời gian. Nghiên cứu này là một mở rộng về thời gian của hệ dịch chuyển phân tán [50] mà ở đó ngôn ngữ đoán nhận của hệ là các Vết thời gian. Trong mô hình này, luận án cũng đưa ra một lôgic thời gian tuyến tính được thể hiện trực tiếp trên Vết thời gian theo các cấu hình Foata để đặc tả các thuộc tính lôgic hệ thống. Các kết quả nghiên cứu đã chứng minh và chỉ ra rằng bài toán kiểm chứng hệ thống là quyết định được.



Hình 1.1: Cấu trúc luận án

1.3 Bố cục của luận án

Luận án được bố cục gồm các chương sau. Chương 2 trình bày tóm tắt các nghiên cứu nền tảng cho các nghiên cứu tiếp theo của luận án. Trong chương này, các lý thuyết về công nghệ phần mềm nói chung, ô-tô-mát thời gian và lý thuyết Vết Mazurkiewicz được giới thiệu. Các lý thuyết này là lý thuyết cơ bản cho các nghiên cứu phát triển trong luận án.

Chương 3 đưa ra lý thuyết Vết thời gian dựa trên Vết Mazurkiewicz. Trong chương này, luận án đề xuất một nghiên cứu về mở rộng về thời gian trên các Vết dựa trên các lợi ích quan trọng của chúng trong việc đặc tả hệ thống tương tranh. Luận án phát triển thêm vào đó yếu tố thời gian với các ràng buộc khoảng cho các hành động của hệ thống. Với sự gia tăng này, Vết Mazurkiewicz trở thành Vết thời gian và phù hợp cho đặc tả các thể thức giao diện của các thành phần. Các nghiên cứu trong luận án còn đưa ra một ô-tô-mát đoán nhận ngôn ngữ

Vết thời gian được gọi là ô-tô-mát khoảng bất đồng bộ. Chương này cũng đưa ra một lôgic cho phép hỗ trợ các đặc tả lôgic của hệ thống và được thể hiện trực tiếp trên Vết thời gian. Như vậy Vết thời gian là đủ để đặc tả một lớp lớn các bài toán ứng dụng liên quan.

Chương 4 trình bày một ứng dụng của lý thuyết Vết trong việc đặc tả hệ thống tương tranh có yếu tố thời gian dựa trên việc sử dụng Vết thời gian cho đặc tả các thể thức giao diện thành phần được mở rộng từ lý thuyết rCOS. Trong chương này, chúng tôi định nghĩa lại các khái niệm về thiết kế, giao diện, thành phần, các phép toán làm mịn, ghép nối của rCOS. Như vậy, với việc làm này, lý thuyết rCOS có thể mở rộng để đặc tả các hệ tương tranh có yếu tố thời gian.

Chương 5 là một phát triển của lý thuyết Vết trên cơ sở xây dựng một phương pháp phát triển hệ tương tranh có yếu tố thời gian. Trong chương này chúng tôi đề xuất mô hình hệ tương tranh có yếu tố thời gian là một tập các thành phần tương thích được ghép nối với nhau mà mỗi thành phần là một thành phần tương tranh có yếu tố thời gian. Lý thuyết về thiết kế dựa trên thành phần của chúng tôi dựa trên lý thuyết về thiết kế dựa trên giao diện của Thomat Henzinger và các cộng sự. Để làm điều này, chúng tôi đã hạn chế về ô-tô-mát khoảng bất đồng bộ thành ô-tô-mát giao diện tương tranh thời gian bất đồng bộ để đặc tả các thành phần tương tranh có yếu tố thời gian. Các định nghĩa và phép toán trên thành phần được chúng tôi đề cập đầy đủ trong chương này.

Chương 6 một lần nữa ứng dụng Vết thời gian vào trong việc đặc tả hệ có yếu tố thời gian. Trong chương này, luận án đề xuất mở rộng hệ phân tán dựa trên việc mô hình bằng các hệ dịch truyển phân tán. Chúng tôi đã sử dụng Vết thời gian như là ngôn ngữ của hệ phân tán bằng việc đề xuất hệ dịch chuyển thời gian phân tán để đặc tả hệ thống và đề xuất lôgic thời gian trên các cấu hình Foata để đặc tả các thuộc tính lôgic của hệ thống. Chúng tôi cũng chỉ ra thuật toán kiểm chứng hệ thống bằng cách đặc tả trên và chứng minh rằng bài toán kiểm chứng là quyết định được.

Các kết luận về luận án và các nghiên cứu tiếp theo của luận án được chúng tôi trình bày trong chương 7. Cấu trúc tổng quan các phần trong luận án được thể hiện trong Hình 1.1.

Chương 2

Kiến thức nền tảng

Chương này giới thiệu một số kiến thức nền tảng phục vụ cho các nghiên cứu chuyên sâu của luận án theo mục tiêu đề ra. Các kiến thức này bao gồm các nội dung sau:

i. Các khái quát về công nghệ phần mềm dựa trên thành phần. Kiến thức tổng quát này đưa ra cái nhìn cơ bản và toàn diện về sự phát triển của công nghệ phần mềm dựa trên thành phần bao gồm các công nghệ hiện nay, vấn đề đảm bảo chất lượng, các mô hình đảm bảo cũng như các công đoạn quan trọng trong quá trình phát triển phần mềm. Từ đó, nhà thiết kế sẽ biết được việc đảm bảo công đoạn nào ra sao, thế nào và cái nào quyết định chính cũng như các vấn đề cần phải giải quyết.

ii. Phương pháp phổ biến trong đặc tả các thành phần có chứa các ràng buộc thời gian là sử dụng công cụ là lý thuyết ô-tô-mát thời gian [4, 11]. Đây là công cụ nổi tiếng và được ứng dụng hoặc được coi là nền tảng trong nghiên cứu các hệ có yếu tố thời gian.

iii. Lý thuyết Vết Mazurkiewicz [30, 68]. Đây là lý thuyết được ứng dụng trong giải quyết bài toán đặc tả được các tính chất tương tranh của các thành phần. Các kết quả trong luận án sẽ dựa trên lý thuyết này để phát triển nhằm đạt được mục tiêu đề ra.

Chi tiết các nội dung trên được trình bày trong các phần dưới đây.

2.1 Công nghệ phần mềm dựa trên thành phần

2.1.1 Giới thiệu

Hệ thống phần mềm hiện đại ngày càng trở nên quy mô lớn, phức tạp và không dễ nắm bắt kiểm soát. Hệ quả tất yếu là chi phí phát triển cao, năng suất thấp, không thể quản lý chất lượng phần mềm và có nguy cơ cao để chuyển sang công nghệ mới [63]. Điều này đã dẫn tới một nhu cầu cho việc tìm kiếm một mô hình phát triển phần mềm mới, hiệu quả, và chi phí phải chăng. Một trong những giải pháp hứa hẹn nhất hiện nay là phát triển phần mềm dựa trên phương pháp tiếp cận dựa trên thành phần. Cách tiếp cận này được dựa trên ý tưởng rằng các hệ thống phần mềm có thể được phát triển bằng cách chọn các thành phần có sẵn thích hợp và sau đó lắp ráp chúng với một kiến trúc phần mềm đã được xác định [25]. Phương pháp tiếp cận phát triển phần mềm mới này là rất khác nhau từ cách tiếp cận truyền thống trong đó các hệ thống phần mềm có thể thực hiện từ đầu và được gọi là công nghệ phần mềm dựa trên thành phần (CBSE - **C**omponent-**B**ased **S**oftware **E**ngineering-CBSE). Những thành phần thương mại có sẵn (**C**omponent **O**ff-the-**S**helf - COTS) có thể được phát triển bởi các nhà phát triển khác nhau bằng cách sử dụng các ngôn ngữ khác nhau và các nền tảng khác nhau. Điều này có thể được minh họa trong Hình 2.1, các thành phần COTS có thể được kiểm tra từ một kho lưu trữ thành phần, và lắp ráp thành một hệ thống phần mềm theo mục tiêu có trước.

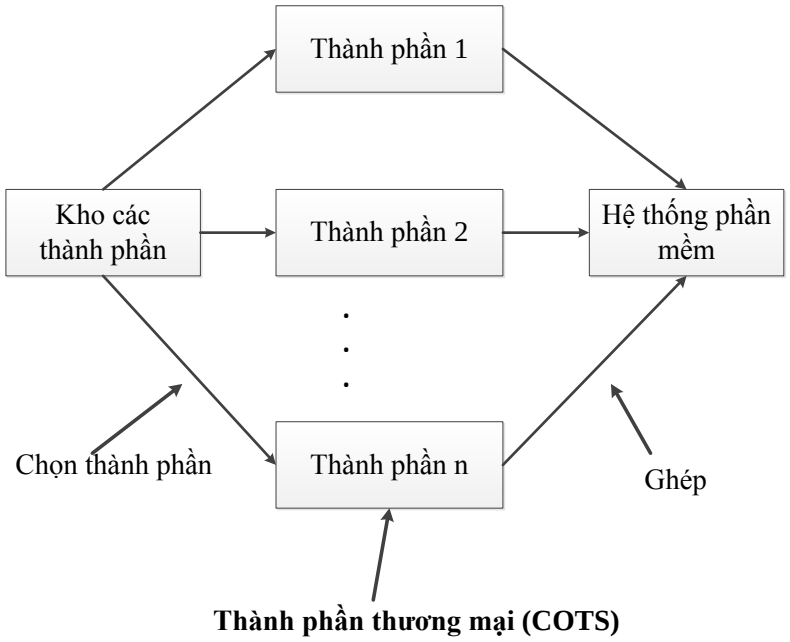
Phát triển phần mềm dựa trên thành phần (**C**omponent-**B**ased **S**oftware **D**evelopment-CBSD) có thể làm giảm đáng kể chi phí phát triển và thời gian để đưa ra thị trường. Nó cũng góp phần cải thiện việc bảo trì, tăng độ tin cậy và chất lượng tổng thể của hệ thống phần mềm [36]. Cách tiếp cận này không chỉ nhận được sự quan tâm trong cộng đồng nghiên cứu mà còn trong ngành công nghiệp phần mềm. Chu kỳ sống và mô hình kỹ thuật phần mềm của CBSD có nhiều khác nhau so với các kỹ thuật truyền thống. Đây là những công việc mà CBSE được tập trung nghiên cứu và phát triển.

Cho đến nay, các công nghệ thành phần phần mềm là một công nghệ mới nổi, nó vẫn còn nhiều vấn đề trước khi đưa chúng đến được sự phát triển hoàn thiện. Không có tiêu chuẩn hoặc hướng dẫn nào trong lĩnh vực mới này, và thậm chí không có một định nghĩa thống nhất về "thành phần". Tuy nhiên, một thành phần có ba tính năng chính[15]:

i. Một thành phần là một phần độc lập và có thể thay thế của một hệ thống thực hiện một chức năng một cách rõ ràng,

ii. Một thành phần hoạt động trong bối cảnh của một kiến trúc được xác định rõ, và

iii. Một thành phần giao tiếp với các thành phần khác bởi giao diện của nó.



Hình 2.1: Mô hình phát triển CBSE

Để đảm bảo rằng một phần mềm dựa trên thành phần có thể chạy đúng cách và hiệu quả thì kiến trúc hệ thống là yếu tố quan trọng nhất. Theo cả hai cộng đồng nghiên cứu [37] và ngành công nghiệp thực hành[1], kiến trúc hệ thống của các hệ thống phần mềm dựa trên thành phần phải là một kiến trúc nhiều lớp và mô-đun (Hình 2.2). Lớp kiến trúc là hệ thống ứng dụng hỗ trợ doanh nghiệp. Lớp thứ hai bao gồm các thành phần tham gia vào một doanh nghiệp cụ thể hoặc một miền ứng dụng, bao gồm các thành phần có thể sử dụng trong hơn một ứng dụng duy nhất. Lớp thứ ba là thành phần kinh doanh qua trung gian bao gồm các phần mềm phổ biến và các giao diện cho các thực thể khác được thiết lập. Cuối cùng, lớp thấp nhất của các thành phần phần mềm hệ thống bao

[1] http://www4.ibm.com/software/ad/sanfrancisco

gồm các thành phần cơ bản mà giao tiếp với các hệ thống điều hành cơ bản và phần cứng.

Các công nghệ dựa trên thành phần hiện tại đã được sử dụng để thực hiện xây dựng các hệ thống phần mềm khác nhau, chẳng hạn như phần mềm thành phần phân tán hướng đối tượng [72] và các ứng dụng Web cho doanh nghiệp [64]. Ngoài ra còn có một số sản phẩm thương mại được tham gia vào cuộc cách mạng thành phần phần mềm, chẳng hạn như BEA, Microsoft, IBM và Sun. Một ví dụ nổi bật là dự án của IBM SanFrancisco. Nó cung cấp một cơ sở hạ tầng đối tượng phân tán có thể tái sử dụng và thiết lập một cách phong phú của thành phần ứng dụng cho các nhà phát triển ứng dụng[2].

2.1.2 Các công nghệ xây dựng hệ thống phần mềm dựa trên thành phần hiện nay

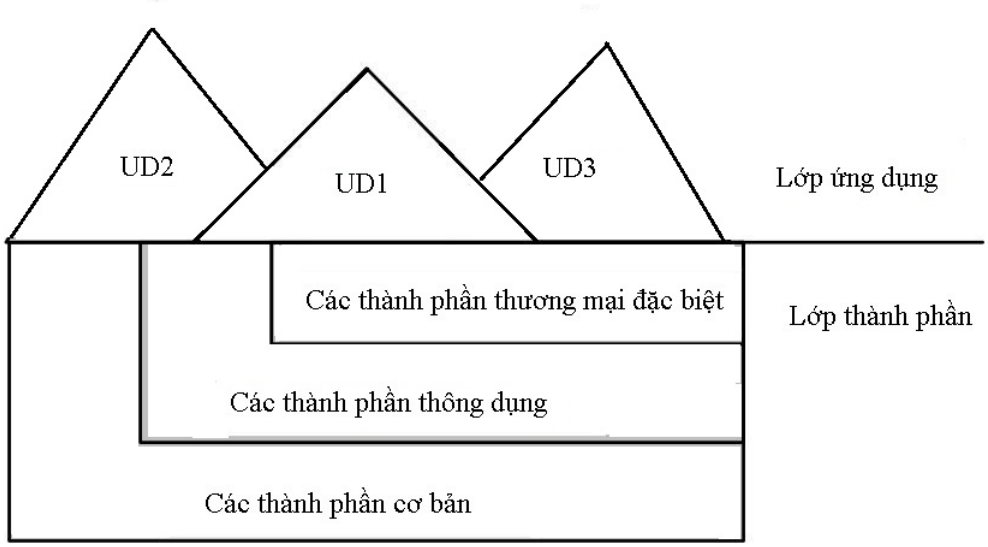
Một số phương pháp tiếp cận, như Visual Basic Controls (VBX), các điều khiển ActiveX, các thư viện lớp, JavaBeans,... đưa ra cách tiếp cận về ngôn ngữ liên quan của chúng như Visual Basic, C++, Java, và các công cụ hỗ trợ để chia sẻ và phân tán các phần ứng dụng. Nhưng tất cả các phương pháp tiếp cận đều dựa vào một số dịch vụ cơ bản để cung cấp các thông tin giao tiếp và phối hợp cần thiết cho các ứng dụng. Cơ sở hạ tầng của các thành phần (đôi khi được gọi là một mô hình thành phần) hoạt động như "hệ thống ống nước" cho phép giao tiếp giữa các thành phần [15]. Trong số các cơ sở hạ tầng công nghệ thành phần đã được phát triển, có ba nhà phát triển đã trở thành chuẩn là: OMG CORBA, Microsoft Component Object Model (COM) và Distributed COM (DCOM), và của Sun JavaBeans và Enterprise JavaBeans. Dưới đây sẽ giới thiệu tổng quan về ba công nghệ này.

CORBA

CORBA (viết tắt của Common Object Request Broker Architecture) là một chuẩn mở cho khả năng tương tác ứng dụng được định nghĩa và được hỗ trợ bởi tập đoàn quản lý đối tượng (Object Management Group - OMG), một tổ chức của hơn 400 nhà cung cấp phần mềm và người sử dụng[3]. Nói một cách đơn giản,

[2]http://www4.ibm.com/software/ad/sanfrancisco

[3]http://www.omg.org/corba/whatiscorba.html



Hình 2.2: Kiến trúc CBSE

CORBA quản lý chi tiết về khả năng tương tác thành phần, và cho phép các ứng dụng giao tiếp với nhau với các địa điểm và thiết kế khác nhau. Giao diện là cách duy nhất mà các ứng dụng hoặc các thành phần qua nó giao tiếp với nhau.

Phần quan trọng nhất của một hệ thống CORBA là ORB(Object Request Broker). ORB là phần trung gian cho việc thiết lập mối quan hệ khách-chủ (client-server) giữa các thành phần. Sử dụng một ORB, một máy khách có thể gọi một phương thức trên một đối tượng máy chủ, có vị trí là hoàn toàn trong suốt với người dùng. ORB có trách nhiệm ngăn chặn một cuộc gọi và tìm kiếm một đối tượng có thể thực hiện các yêu cầu, vượt qua các tham số của nó, gọi phương thức của nó, và trả lại kết quả. Các máy khách không cần phải biết được nơi đối tượng được gọi, ngôn ngữ lập trình, hệ điều hành của nó, hoặc bất kỳ khía cạnh khác của hệ thống không liên quan đến giao diện. Bằng cách này, ORB cung cấp khả năng tương tác giữa các ứng dụng trên các máy khác nhau trong môi trường phân tán không đồng nhất và kết nối một cách liên tục với các hệ thống nhiều đối tượng.

CORBA được sử dụng rộng rãi trong hệ thống phân tán hướng đối tượng [72] bao gồm các hệ thống phần mềm dựa trên thành phần bởi vì nó cung cấp một chương trình phân tán phù hợp và môi trường có yếu tố thời gian trên các ngôn ngữ lập trình, hệ điều hành, và các mạng phân tán phổ biến.

COM và DCOM

Được giới thiệu vào năm 1993, COM là mô hình đối tượng thành phần (Component Object Model-COM) là một kiến trúc chung cho phần mềm thành phần[4]. Nó cung cấp phụ thuộc nền tảng, dựa trên Windows và Windows NT, và các ứng dụng dựa trên thành phần với ngôn ngữ độc lập. COM định nghĩa thành phần và máy khách của chúng tương tác như thế nào. Sự tương tác này được định nghĩa sao cho các máy khách và các thành phần có thể kết nối mà không cần bất cứ thành phần hệ thống trung gian nào. Đặc biệt, COM cung cấp một tiêu chuẩn rằng các thành phần và các máy khách của chúng phải tuân theo để đảm bảo khả năng tương tác động. Điều này cho phép cập nhật phần mềm trực tuyến và sử dụng lại phần mềm qua ngôn ngữ [56] .

Như một phần mở rộng của COM, COM phân tán (Distributed COM - DCOM), là một giao thức cho phép các thành phần phần mềm giao tiếp trực tiếp qua mạng một cách đáng tin cậy, an toàn và hiệu quả. DCOM được thiết kế để sử dụng trên nhiều vận chuyển mạng, bao gồm cả giao thức Internet như HTTP. Khi một máy khách và thành phần của nó cư trú trên các máy khác nhau, DCOM chỉ đơn giản là thay thế các thông tin liên lạc liên tiến trình địa phương với một giao thức mạng. Cả máy khách và thành phần đều không thấy được những thay đổi của các kết nối vật lý.

Mô hình thành phần dựa trên Java của Sun

Mô hình thành phần dựa trên Java của Sun (Sun Microsystems's JavaBeans) bao gồm hai phần: phần JavaBeans để phát triển thành phần phía máy khách và Enterprise JavaBeans (EJB) cho phát triển thành phần phía máy chủ. Kiến trúc thành phần JavaBeans hỗ trợ các ứng dụng của nhiều nền tảng, cũng như các thành phần tái sử dụng, các thành phần phía máy khách và máy chủ[5].

Nền tảng Java cung cấp một giải pháp hiệu quả cho tính di động và các vấn đề bảo mật thông qua việc sử dụng mã Java bytecode và khái niệm về Java applets đáng tin cậy và không đáng tin cậy. Java cung cấp một công nghệ tích hợp chung, thuận lợi cho phát triển ứng dụng doanh nghiệp, bao gồm các máy chủ nhiều nhà cung cấp liên điều hành (interoperating across multivendor servers);

[4]http://www.microsoft.com/isapi

[5]http://developer.java.sun.com/developer

giao dịch và ngữ cảnh an ninh; cung cấp dịch vụ máy khách đa ngôn ngữ, và hỗ trợ ActiveX thông qua cầu nối DCOM/CORBA.

JavaBeans và EJB mở rộng tất cả các thế mạnh vốn có của Java bao gồm cả tính di động và an ninh vào lĩnh vực phát triển dựa trên thành phần. Tính di động, bảo mật và độ tin cậy của Java rất thích hợp cho phát triển các đối tượng máy chủ mạnh mẽ độc lập với hệ điều hành, máy chủ Web và máy chủ quản lý cơ sở dữ liệu.

So sánh giữa các công nghệ hiện nay

So sánh giữa các công nghệ thành phần hiện tại có thể được tìm thấy trong nhiều tài liệu khác nhau. Ở đây luận án chỉ đưa ra tóm tắt các tính năng khác nhau và được chỉ ra trong trong Bảng 2.1 theo tài liệu [18]. Các tiêu chí được đưa ra so sánh trong giữa các công nghệ bao gồm: Môi trường phát triển, chuẩn giao diện, khả năng tương thích và tính di động, sửa đổi và duy trì, các dịch vụ cung cấp, nền tảng phụ thuộc, ngôn ngữ phụ thuộc và thế mạnh ứng dụng.

2.1.3 Đảm bảo chất lượng cho các hệ thống phần mềm dựa trên thành phần

Vòng đời của các hệ thống phần mềm dựa trên thành phần

Hệ thống phần mềm thành phần được phát triển bằng cách lựa chọn các thành phần khác nhau và lắp ráp chúng lại với nhau chứ không phải là lập trình một hệ thống tổng thể từ đầu, do đó vòng đời của các hệ thống phần mềm dựa trên thành phần là khác so với các hệ thống phần mềm truyền thống. Vòng đời của hệ thống phần mềm dựa trên thành phần có thể được tóm tắt gồm các bước như sau [25]:

i. Phân tích các yêu cầu,

ii. Lựa chọn kiến trúc phần mềm, xây dựng, phân tích, và đánh giá,

iii. Xác định và tùy biến thành phần,

iv. Tích hợp hệ thống,

v. Kiểm thử hệ thống, và

vi. Bảo trì phần mềm.

Bảng 2.1: Bảng so sánh giữa các công nghệ

DV	CORBA	EJB	COM/DCOM
Môi trường phát triển	kém phát triển	mới nổi	Được hỗ trợ bởi một phạm vi rộng của môi trường phát triển mạnh mẽ môi trường
Chuẩn giao diện	Không	Dựa trên COM, Java	Chuẩn các tương tác thành phần
Khả năng tương thích và tính di động	đặc biệt mạnh trong chuẩn hóa các cam kết ràng buộc ngôn ngữ, nhưng không di động	Di động theo ngôn ngữ Java nhưng không tương thích	Không có bất kỳ khái niệm về chuẩn mức nguồn của chuẩn ràng buộc ngôn ngữ.
Sửa đổi và duy trì	CORBA IDL để định nghĩa các giao diện thành phần, cần sửa đổi bổ sung và bảo trì	Không liên quan đến các file IDL định nghĩa các giao diện giữa các thành phần và container.Dễ dàng thay đổi và bảo trì.	Microsoft IDL cho việc định nghĩa các giao diện thành phần, cần sửa đổi bổ sung và bảo trì
Các dịch vụ cung cấp	Một bộ đầy đủ các dịch vụ được chuẩn hóa, thiếu thực thi	KHông chuẩn cũng không thực thi	Gần đây, bổ sung một số dịch vụ quan trọng
Nền tảng phụ thuộc	Độc lập	Độc lập	Độc lập
Ngôn ngữ phụ thuộc	Độc lập	Phụ thuộc	Độc lập
Thực hiện	Mạnh mẽ nhất cho điện toán doanh nghiệp truyền thống	Mạnh nhất trên Web máy khách chung.	Mạnh nhất trên các ứng dụng máy tính để bàn truyền thống

Các kiến trúc của phần mềm định nghĩa một hệ thống về khía cạnh tính toán các thành phần và tương tác giữa các thành phần. Tập trung vào kết hợp và lắp ráp các thành phần có khả năng được phát triển riêng biệt, và thậm chí là độc lập. Xác định thành phần, tuỳ biến và tích hợp là một hoạt động quan trọng trong vòng đời của các hệ thống dựa trên thành phần. Nó bao gồm hai phần chính: Một là ước lượng thành phần COTS ứng cử viên dựa trên các yêu cầu chức năng và chất lượng sẽ được sử dụng để đánh giá thành phần đó, và hai là tuỳ biến của những thành phần ứng cử COTS nên được sửa đổi trước khi được tích hợp vào hệ thống phần mềm thành phần mới. Tích hợp là để tạo ra các quyết định quan trọng về việc làm thế nào để cung cấp thông tin liên lạc và phối hợp giữa các thành phần khác nhau của hệ thống phần mềm mục tiêu.

Để bảo đảm chất lượng cho các hệ thống phần mềm dựa trên thành phần, nên chỉ ra vòng đời và các hoạt động chính của nó để phân tích các thành phần và đạt được hệ thống phần mềm chất lượng cao dựa trên thành phần. Các công nghệ bảo đảm chất lượng cho các hệ thống phần mềm dựa trên thành phần đã xuất hiện sớm, như các đặc tính cụ thể của các hệ thống thành phần khác với các hệ thống truyền thống thế nào. Mặc dù một số bảo đảm chất lượng kỹ thuật chẳng hạn như mô hình phân tích độ tin cậy cho các hệ thống phần mềm phân tán và phương pháp tiếp cận dựa trên thành phần cho công nghệ phần mềm đã được nghiên cứu, nhưng vẫn còn những chuẩn và hướng dẫn không rõ ràng cho các hệ thống phần mềm thành phần dựa trên thành phần. Việc xác định các đặc điểm bảo đảm chất lượng, cùng với các mô hình, công cụ và các số liệu, là các nhu cầu cấp thiết.

Đặc tính chất lượng của các thành phần

Khi nhiều công việc vẫn chưa được thực hiện để phát triển phần mềm dựa trên thành phần, câu hỏi và câu trả lời cho công nghệ bảo đảm chất lượng phát triển phần mềm dựa trên thành phần phải chỉ ra hai phần không thể tách rời: Làm thế nào để xác nhận chất lượng của một thành phần? Và làm thế nào để xác nhận chất lượng của các hệ thống phần mềm dựa trên các thành phần? Để trả lời các câu hỏi, các mô hình cần được thúc đẩy để xác định kiểm soát chất lượng tổng thể của các thành phần và các hệ thống số liệu được tìm thấy để đo kích thước, độ phức tạp, khả năng tái sử dụng và độ tin cậy của các thành phần và các hệ thống và công cụ cần được quyết định kiểm tra các thành phần và hệ thống hiện có.

Để đánh giá một thành phần, chúng ta phải xác định làm thế nào để chứng nhận chất lượng của thành phần. Các đặc tính chất lượng của các thành phần là nền tảng để đảm bảo chất lượng của các thành phần, và do đó là nền tảng để đảm bảo chất lượng của toàn bộ hệ thống phần mềm dựa trên thành phần. Nhiều nghiên cứu đã đề xuất một danh sách các đặc điểm về chất lượng của các thành phần gồm: chức năng; giao diện; khả năng sử dụng; khả năng kiểm thử; bảo trì; và độ tin cậy. Các thước đo phần mềm có thể được đề xuất để đo đạc độ phức tạp của phần mềm và đảm bảo chất lượng của nó. Các số phép đo thường được sử dụng để phân loại các thành phần bao gồm:

i. ***Kích cỡ phần mềm***: Điều này ảnh hưởng đến chi phí tái sử dụng và chất lượng. Nếu nó quá nhỏ, lợi ích sẽ không vượt quá chi phí quản lý nó. Nếu nó là quá lớn, rất khó để có thể có chất lượng cao,

ii. ***Sự phức tạp***: Điều này cũng ảnh hưởng đến chi phí tái sử dụng và chất lượng. Một thành phần quá tầm thường là không có lợi nhuận để tái sử dụng trong khi một thành phần quá phức tạp khó có thể kế thừa chất lượng cao,

iii. ***Tần số tái sử dụng***: Số lượng các sự cố mà một thành phần được sử dụng là một chỉ số vững chắc của tính hữu dụng của nó, và

iv. ***Độ tin cậy***: Xác suất của hoạt động thất bại của một thành phần theo các kịch bản hoạt động.

2.2 Ô-tô-mát thời gian

Trong các hệ thống rời rạc, các dịch chuyển của hệ thống được xem xét như là các yếu tố nguyên tử (atomic primitives), các sự kiện xảy ra ngay lập tức. Đôi khi chúng ta muốn một mức trừu tượng hơn, chúng ta muốn đặc tả một cách thực tế rằng các sự kiện khi hoạt động sẽ tiêu tốn một lượng thời gian nhất định nào đó. Một cách thức để thấy được là giả sử các tiến trình là liên tục (ví dụ như hành động đổ đầy nước vào tét đựng nước) và ta áp dụng một mức trừu tượng rời rạc: "Hành động đổ nước vào tét mất t đơn vị thời gian". Các hành vi rời rạc được xem như là chuỗi các sự kiện mà không có thông tin đo đạc về thời gian, chỉ có thứ tự hoặc thứ tự bộ phận giữa các sự kiện. Một hành vi thời gian liên quan tới việc nhúng chuỗi này vào trục thời gian và ta có khái niệm từ thời gian (timed words -TW). Để đặc tả các hệ thống có thời gian như vậy, nhiều mô hình thời gian đã được đề xuất như Petri Nets, đại số tính toán tiến trình thời gian, và các lôgic có yếu tố thời gian (timed logics) [48, 67]. Lý thuyết

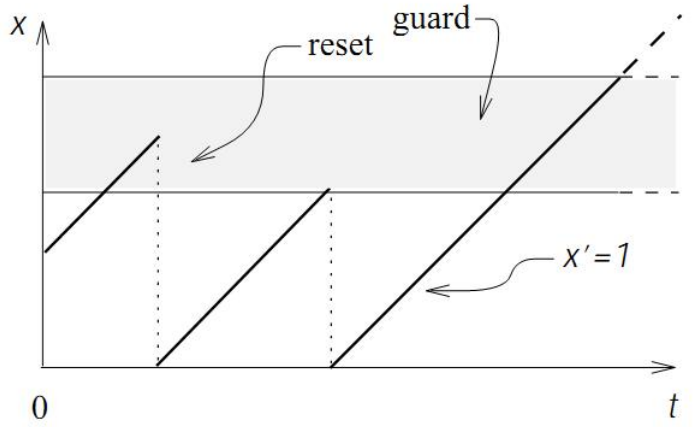
về ô-tô-mát thời gian (Timed Automata - TA) được đề xuất bởi Alur và Dill trong [4] đã mở ra một hướng tiếp cận mới với nhiều sức mạnh và các phát triển công cụ đặc tả với TA được coi là cốt lõi của ngôn ngữ đặc tả [10]. TA cung cấp một nền tảng cơ bản cho các phương pháp hình thức trong việc đặc tả và kiểm chứng tính đúng đắn của các hệ thống thời gian thực (Real-time systems - RTS). Trong phần này, chúng tôi giới thiệu những nét cơ bản về lý thuyết về ô-tô-mát thời gian, các kết quả liên quan và công cụ kiểm chứng mô hình UPPAAL[11].

2.2.1 Giới thiệu

Các lý thuyết về ô-tô-mát thời gian cung cấp một nền tảng hình thức để mô hình và phân tích hành vi của hệ thống có yếu tố thời gian. Đây là các hệ thống mà có chức năng thực hiện đúng đắn và đảm bảo rằng các ràng buộc hạn chế về thời gian là thỏa mãn, chẳng hạn như yếu tố thời gian thực hiện, khoảng thời gian làm nhiệm vụ, độ trễ liên lạc, v.v. Ô-tô-mát thời gian lần đầu tiên được đề xuất bởi A.Dill trong [4] như là một phần mở rộng của phương pháp tiếp cận lý thuyết ô-tô-mát cổ điển để đặc tả các hệ thống có yếu tố thời gian. Kể từ đó, các lý thuyết về ô-tô-mát thời gian trở thành một lĩnh vực chuyên sâu của các nghiên cứu khoa học máy tính. Để minh chứng cho tính hiệu quả và khả năng ứng dụng của lý thuyết này, rất nhiều các nghiên cứu về lý thuyết đã phát triển mở rộng, đi sâu và đã xây dựng công cụ hỗ trợ là UPPAAL[11].

Một ô-tô-mát thời gian là một máy hữu hạn trạng thái được trang bị một bộ các đồng hồ. Đồng hồ là một hàm gán các giá trị thực dùng để ghi lại thời gian trôi qua giữa các sự kiện. Tất cả các đồng hồ được đồng bộ hóa, đó là, tất cả đều được giả định hoạt động ở cùng một tốc độ. Sự gián đoạn có thể xảy ra khi quá trình dịch chuyển được thực hiện. Trong trường hợp này, đồng hồ được phép được thiết lập lại một giá trị mới và trở thành giá trị ban đầu của các giai đoạn liên tục tiếp theo.

Quá trình chuyển đổi có liên quan với một hoặc một số các ràng buộc (guard) là một vị từ trên các đồng hồ. Ràng buộc guard đảm bảo điều kiện khi một quá trình dịch chuyển có thể được thực hiện. Các hành vi của một đồng hồ như là một hàm của thời gian được minh họa trong Hình 2.3. Trong đó, giá trị đồng hồ x tại một thời điểm t là một số thực không âm, x có thể được đặt lại giá trị bằng 0 (reset) ở bất kỳ thời điểm nào phụ thuộc vào yêu cầu bài toán cụ thể. Các ràng buộc về thời gian của hệ thống (guard) sẽ được đặc tả thông qua điều kiện ràng buộc trên các giá trị đồng hồ này .



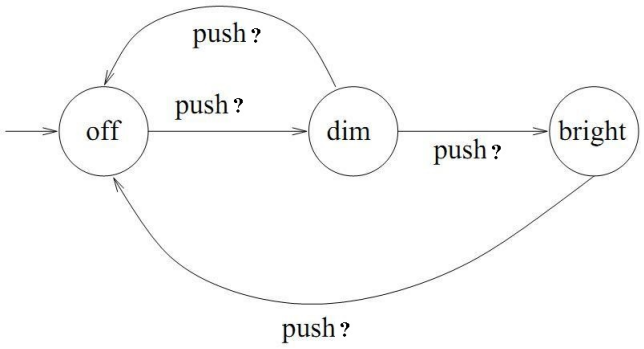
Hình 2.3: Đồng hồ là một hàm thời gian

2.2.2 Ô-tô-mát thời gian

Trong mục này, trước tiên luận án giới thiệu tổng quan về ô-tô-mát thời gian bao gồm các khái niệm, hoạt động của TA và mô hình mạng các TA. Tiếp theo, luận án giới thiệu về bài toán kiểm chứng mô hình bằng TA cũng như các kết quả liên quan. TA kích hoạt các dịch chuyển từ vị trí này sang vị trí khác khi các điều kiện về thời gian được thỏa mãn. Những dịch chuyển này có thể hoặc thực hiện đầu vào hoặc đầu ra trên các kênh mà sẽ đồng bộ với các TA khác làm việc song song hoặc chúng thực hiện các hành động bên trong mà không thể hiện ra bên ngoài. Chúng ta bắt đầu với việc xem xét ví dụ sau.

Ví dụ 2.1 (Điều khiển đèn giao thông) *Chúng ta muốn mô hình một bộ điều khiển đèn giao thông với cách thức hoạt động như sau: Trước tiên, đèn ở trạng thái tắt (off). Khi công tắc được nhấn một lần, đèn chuyển sang trạng thái mờ. Nếu công tắc được nhấn hai lần liên tiếp (đủ nhanh) thì đèn sáng. Ngược lại, nếu công tắc nhấn một lần thì chỉ sau một khoảng thời gian nào đó, đèn trở lại trạng thái tắt.*

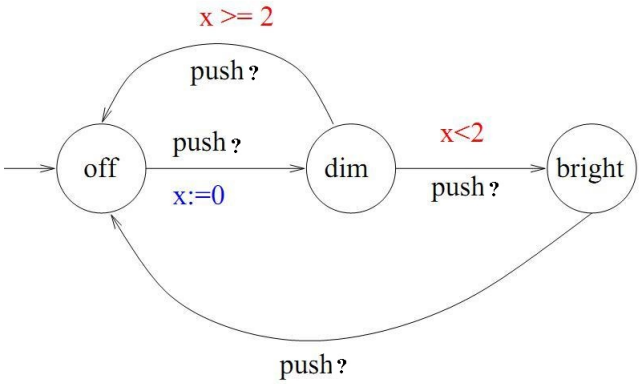
Chúng ta sẽ thử mô hình hệ thống này bằng một ô-tô-mát với ba trạng thái là off, dim, bright ứng với tắt, mờ và sáng . Trạng thái khởi động là off. Có bốn dịch chuyển được đánh dấu bởi ký hiệu ? là ký hiệu được dùng để biểu diễn dịch chuyển đợi một đầu vào từ môi trường. Ô-tô-mát này có biểu diễn bằng đồ thị như Hình 2.4.



Hình 2.4: Mô hình điều khiển đèn không có thời gian

Một vấn đề với mô hình này là các khái niệm liên quan tới thời gian là "nhanh" và "sau một khoảng thời gian" không thể được biểu diễn. Thay vào đó, ô-tô-mát kế thừa tính không đơn định trong trạng thái dim: khi công tắc được bật, nó sẽ chuyển sang hoặc sáng hoặc tắt.

TA có thể làm được việc này bằng việc mở rộng ô-tô-mát với các đồng hồ để thể hiện thời gian liên tục. Ban đầu, các đồng hồ khởi tạo với giá trị 0. Sau đó, các giá trị đồng hồ tăng lên một cách liên tục. Các dịch chuyển có thể phụ thuộc vào giá trị hiện thời của đồng hồ. Các giá trị của đồng hồ cũng có thể được thiết lập lại về 0. Đối với bộ điều khiển đèn, chúng ta có thể lấy một đồng hồ x và mở rộng ô-tô-mát theo TA của K.G. Larsen[49]: Khi công tắc được bật lần đầu tiên, đồng hồ được thiết lập về giá trị 0 bằng hàm gán $x := 0$. Chỉ khi nhấn lần thứ hai trong vòng 2 giây sau lần một thì đèn chuyển sang trạng thái sáng. Nếu thời gian lớn hơn 2 giây, đèn chuyển sang trạng thái tắt. Đây cũng là trường hợp khi đèn ở trạng thái sáng. Chú ý rằng các điều kiện không giao nhau về thời gian đã làm chuyển tính không đơn định tại trạng thái của đèn.



Hình 2.5: Mô hình điều khiển đèn có thời gian

Trước khi định nghĩa TA, chúng ta xét một số định nghĩa trước mà liên quan tới định nghĩa TA trong phần dưới đây.

Hoạt động của TA

Cho S là một tập, gọi S^* là tập các chuỗi hữu hạn các thành phần trong S. Chúng ta quan tâm miền thời gian Ţ là tập $\mathbb{Q}_+$ các số hữu tỉ dương hoặc $\mathbb{R}_+$ các số thực dương và Σ là tập các hành động (actions). Một chuỗi thời gian trên Ţ là một chuỗi hữu hạn không giảm $\tau = (t_i)_{1 \leq i \leq p} \in \text{Ţ}^*$. Khi đó ta có định nghĩa về từ thời gian (timed words) như sau.

Định nghĩa 2.1 (Từ thời gian) *Một từ gian gian $\omega = (a_i, t_i)_{1 \leq i \leq p}$ là một thành phần của $(\Sigma \times \text{Ţ})^*$, tức là $\omega \in (\Sigma \times \text{Ţ})^*$ và thường được biểu diễn dưới dạng (σ, τ) với σ là một từ trên Σ^* và τ là một chuỗi thời gian trên Ţ^* và độ dài của từ σ và τ là tương đương nhau, tức là $\sigma = (a_i)_{1 \leq i \leq p}, \tau = (t_i)_{1 \leq i \leq p}$.*

Ví dụ 2.2 *Cho $\Sigma = \{a, b, c, d\}$, $\omega = \{(a, 1)(b, 3)(a, 5)(d, 6)(c, 8)\}$ là một từ thời gian trên bảng chữ cái Σ. Ta cũng có thể viết $\omega = (\sigma, \tau)$ với $\sigma = \{abadc\}, \tau = \{13568\}$.*

Giá trị đồng hồ và các phép toán trên đồng hồ

Để có thể đặc tả và tính toán các ràng buộc về thời gian, TA sử dụng tập các biến được gọi là đồng hồ. Gọi X là tập hữu hạn các biến đồng hồ, một giá trị đồng hồ trên X là một ánh xạ $\nu : X \longrightarrow \text{Ţ}$ gán mỗi đồng hồ một giá trị thời gian. Tập tất cả các giá trị đồng hồ trên X được ký hiệu là Ţ^X. Giá trị khởi đầu cho đồng hồ là ánh xạ $\nu_0 : X \longrightarrow \text{Ţ}$ với $\nu_0(x) = 0$ đối với tất cả các $x \in X$. Với mỗi $t \in \text{Ţ}$ chúng ta định nghĩa $(\nu + t)(x) = \nu(x) + t, \forall x \in X$ là giá trị của đồng hồ x, tức là $\nu(x)$ sau t đơn vị thời gian. Cho tập con Y của X $(Y \subseteq X)$, phép thiết lập lại giá trị (Reset) đồng hồ, ký hiệu $\nu[Y := 0]$, thiết lập lại giá trị của các đồng hồ trong Y bằng 0 được xác định như sau:

- $\nu[Y := 0](x) = 0$ nếu $x \in Y$, và
- $\nu[Y := 0](x) = \nu(x)$ nếu $x \notin Y$.

Các ràng buộc đồng hồ

Cho X là một tập hữu hạn các biến đồng hồ, chúng ta giới thiệu tập các ràng buộc đồng hồ trên X. Các ràng buộc này được sử dụng để đặc tả các ràng buộc về thời gian trên hệ thống được mô hình bằng các TA.

Định nghĩa 2.2 (Ràng buộc đồng hồ) *Tập các ràng buộc trên tập đồng hồ X được kí hiệu là $\Phi(X)$ với mỗi ràng buộc ϕ trên tập đồng hồ X được định nghĩa như sau: $\phi ::= x \sim c | x - y \sim c | \phi \wedge \phi$ với $x, y \in X, c \in \mathbb{Z}, \sim \in \{<, \leq, =, >, \geq\}$. Một ràng buộc đồng hồ k (thường được gọi là k-bounded) là một ràng buộc đồng hồ mà liên quan chỉ tới hằng số c trong khoảng -k đến +k. Một ràng buộc có dạng $x - y \sim c$ được gọi là ràng buộc chéo. Tập các ràng buộc k-bounded được kí hiệu là $C^k(X)$.*

Ví dụ 2.3 *Cho x là một đồng hồ, một biểu thức $x < 5 \wedge x > 0$ là một ràng buộc đồng hồ.*

Nếu ν là một giá trị đồng hồ (một ánh xạ giá trị của đồng hồ), g là một ràng buộc đồng hồ được xác định như trên, chúng ta viết $\nu \models g$ khi ν thỏa các giá trị ràng buộc đồng hồ g. Chúng ta nói ν thỏa $x \sim c$ (tương tự $x - y \sim c$) bất cứ khi nào $\nu(x) \sim c$ (tương tự $x - y \sim c$). Chúng ta ký hiệu $\|g\| = \{\nu \in \text{Ţ}^X | \nu \models g\}$.

Định nghĩa hình thức của một Ô-tô-mát thời gian

Bây giờ, chúng ta có thể định nghĩa một cách hình thức về ô-tô-mát thời gian như dưới đây.

Định nghĩa 2.3 (Ô-tô-mát thời gian) *Một ô-tô-mát thời gian (TA) trên Ţ là một bộ $\mathcal{A} = (Q, Q^0, X, \Sigma, I, E, F)$, với*

- *Σ là một tập hữu hạn các hành động,*
- *Q là tập hữu hạn các trạng thái của ô-tô-mát,*
- *$Q^0 \subseteq Q$ là tập các trạng thái khởi đầu,*
- *X là tập hữu hạn các đồng hồ,*
- *$I : Q \longrightarrow \Phi(X)$ là tập các ràng buộc bất biến trên các trạng thái của ô-tô-mát,*
- *$F \subseteq Q$ là tập các trạng thái kết thúc, và*
- *$E \subseteq Q \times \Sigma \times \Phi(X) \times 2^X \times Q$ là một tập hữu hạn các dịch chuyển trạng thái.*

Ngữ nghĩa của TA

Cấu hình của TA được kí hiệu là (q, ν) với $q \in Q$ và $\nu \in \text{Ţ}^X$. Ngữ nghĩa của TA là một hệ dịch chuyển với mỗi trạng thái là một cấu hình và quan hệ chuyển được định nghĩa theo luật sau: Cho δ là một số thực dương ($\delta \in \mathbb{R}_+$), a là một

ký hiệu hành động, $a \in \Sigma$, $(q, \nu) \xrightarrow{\delta} (q, \nu + \delta)$ nếu $\nu \models I(q)$ và $\nu + \delta \models I(q)$, $(q, \nu) \xrightarrow{a} (p, \nu[Y := 0])$ nếu $(q, a, \phi, Y, p) \in E$ và $\nu + \nu \models \phi$.

Một thực thi (run) của một ô-tô-mát trên một từ thời gian ω là một dãy các dịch chuyển có dạng: $\rho = (q_0, \nu_0) \xrightarrow{t_0} q_0, \nu_0' \xrightarrow{a_0} q_1, \nu_1 \xrightarrow{t_1 - t_0} (q_1, \nu_1') \xrightarrow{a_1} ... \xrightarrow{a_{n-1}} (q_n, \nu_n)$. Một thực thi ρ của TA $\mathcal{A}$ được gọi là một thực thi chấp nhận (accepted run) nếu $q_n \in F$ Một từ thời gian ω được gọi là được đoán nhận bởi TA $\mathcal{A}$ nếu tồn tại một thực thi chấp nhận ρ trên $\mathcal{A}$. Và ta có định nghĩa về ngôn ngữ thời gian của ô-tô-mát thời gian như sau.

Định nghĩa 2.4 (Ngôn ngữ được đoán nhận bởi TA) *Tập tất cả các từ thời gian được đoán nhận bởi ô-tô-mát thời gian $\mathcal{A}$ được kí hiệu là $L(\mathcal{A})$ là ngôn ngữ được đoán nhận bởi ô-tô-mát thời gian $\mathcal{A}$. Ngôn ngữ thời gian L được gọi là chính quy nếu tồn tại một TA $\mathcal{A}$ đoán nhận ngôn ngữ L, tức là $L(\mathcal{A}) = \mathcal{L}$.*

Theo [4] ta có kết quả về tính đóng của ngôn ngữ với các phép toán hợp và giao và được thể hiện qua định lý sau.

Định lý 2.1 *Ngôn ngữ chính quy thời gian là đóng đối với phép toán hợp và giao.*

Mạng các TA

Hệ thống thời gian thực (Real-Timed System - RTS) chủ yếu bao gồm một số các thành phần ghép nối song song với nhau và tương tác với các thành phần khác theo thời gian. Để mô hình những hệ thống như vậy, chúng ta xem xét mạng các TA được xây dựng từ các TA đơn lẻ bằng hai phép toán kết hợp là ghép song song và phép hạn chế (restriction). Bất kỳ khái niệm nào của phép kết nối song song của các tiến trình có thể được chuyển về phép ghép nối các TA.

Ở đây, chúng ta chọn thiết lập tính toán giao tiếp hệ thống của R. Milner (Calculus of Communicating Systems - CCS) bởi vì nó cho phép được thực thi trong bộ kiểm chứng UPPAAL mà chúng ta sử dụng cho kiểm chứng tự động các thuộc tính của TA.

Ý tưởng của CCS là các tiến trình song song hoặc trong trường hợp của TA, giao tiếp một một và có bắt tay. Cuối cùng, các hành động bổ sung $a!$ và $a?$ của hai ô-tô-mát có thể đồng bộ để nắm lấy hành động bên trong τ, nhưng chúng cũng được thực hiện một cách riêng biệt để được chuẩn bị cho các đồng bộ sau này. Điều này là quan trọng cho ghép nối song song để trở thành một phép toán

hai ngôi trên TA. Để thực thi đồng bộ kênh, a phải được khai báo như là kênh cục bộ (local).

Định nghĩa 2.5 (Ghép nối song song) *Phép kết nối song song, ký hiệu là* $\|$ *của hai TA* $\mathcal{A}_i = (Q_i, Q_i^0, X_i, \Sigma_i, I_i, E_i, F_i)|i = 1, 2$ *với tập các đồng hồ không giao nhau* $(X_1 \cap X_2 = \emptyset)$ *là một TA* $\mathcal{A} = \mathcal{A}_1 \| \mathcal{A}_2 = (Q_1 \times Q_2, Q_1^0 \times Q_2^0, X_1 \cup X_2, \Sigma_1 \cup \Sigma_2, E, I, (F_1, F_2)$, *sao cho:*

- $I(q_1, q_2) = I(q_1) \wedge I(q_2)$,
- *Nếu* $a \in \Sigma_1 \cap \Sigma_2$ *thì với* $(q_i, a, \phi_i, Y_i, p_i) \in E_i$
  $((q_1, q_2), a, \phi_1 \wedge \phi_2, Y_1 \cup Y_2, (p_1, p_2)) \in E$,
- *Nếu* $a \in \Sigma_1 \setminus \Sigma_2$ *thì với* $(q_1, a, \phi_1, Y_1, p_1) \in E_i$
  $((q_1, q_2), a, \phi_1, Y_1, (p_1, q_2)) \in E$, *và*
- *Nếu* $a \in \Sigma_2 \setminus \Sigma_1$ *thì với* $(q_2, a, \phi_2, Y_2, p_2) \in E_i$
  $((q_1, q_2), a, \phi_2, Y_2, (q_1, p_2)) \in E$.

Từ định nghĩa trên ta có thể suy luận trực tiếp kết quả về tính giao hoán và kết hợp với phép nối song song qua hệ quả sau:

Hệ quả 2.1 *Phép ghép nối song song các TA có tính chất giao hoán và kết hợp.*

Một ví dụ về hệ thống được ghép nối bởi hai TA được chỉ ra trong Hình 2.8.

Bài toán kiểm chứng

Trong nghiên cứu về ô-tô-mát, bài toán kiểm tra tính rỗng của ngôn ngữ đoán nhận là một bài toán quan trọng và thường dành được nhiều sự tập trung nghiên cứu của các nhà khoa học. Trong phần này, chúng ta sẽ tìm hiểu các nghiên cứu về thuộc tính quan trọng liên quan tới câu hỏi về tính rỗng là tính đến được của trạng thái, chúng ta cũng chỉ ra rằng, đối với TA, khả năng đến được của các cấu hình là quyết định được.

Trạng thái đến được: Chúng ta xét bài toán sau:
Cho: một TA $\mathcal{A}$ và một trạng thái điều khiển l.
Câu hỏi: Liệu l có thể được dịch chuyển tới được không? tức là liệu có tồn tại một dãy dịch chuyển cấu hình bắt đầu từ (q_0, ν_0) đến (l, ν_n) trong hệ dịch chuyển trạng thái được định nghĩa bên trên.

Để giải bài toán này, cách đơn giản là xây dựng hệ dịch chuyển trạng thái. Sau đó, áp dụng các thuật toán tìm đường để tìm đường đi từ một đỉnh này tới một đỉnh khác trên hệ dịch chuyển trạng thái. Tuy nhiên, vấn đề đáng quan tâm

ở đây là số các cấu hình là vô hạn do các giá trị của đồng hồ là số thực. Và do đó, bài toán không thể quyết định được trong trường hợp này. Tuy nhiên, cũng theo R. Alur và D. Dill trong [4], bài toán kiểm tra tính rỗng của ngôn ngữ đoán nhận bởi một TA là quyết định được. Để giải quyết vấn đề vô hạn của cấu hình, R. Alur và D. Dill đã đưa ra khái niệm và xây dựng một cấu trúc vùng tương đương mà ở đó có sự tương đương về mặt ngôn ngữ giữa dịch chuyển vùng và hệ dịch chuyển cấu hình. Chúng ta sẽ xem xét tổng quan về ý tưởng và thuật toán của R. Alur và D. Dill như sau:

Xây dựng các vùng: Do hệ dịch chuyển trạng thái của TA là vô hạn trạng thái nên ý tưởng được đưa ra là chia không gian trạng thái này thành các vùng tương đương, phù hợp. Ô-tô-mát được chia này là hữu hạn trạng thái. Quan hệ tương đương này được gọi là tương đương vùng và được định nghĩa như sau:

Định nghĩa 2.6 (Tương đương vùng) *Hai trạng thái được gọi là tương đương vùng nếu chúng cùng thỏa mãn các ràng buộc đồng hồ.*

Cho $\mathcal{A}$ là một TA. Giả sử tất cả các ràng buộc trong $\mathcal{A}$ là số nguyên và gọi c_x là hằng số lớn nhất mà đồng hồ x có được. Gọi $\mu, \nu \in Ti^X$ là các giá trị đồng hồ, chúng ta gọi μ tương đương vùng với ν, kí hiệu $\mu \approx \nu$ nếu:

- $(\forall x)$ hoặc $\mu(x) > c_x$ và $\nu(x) > c_x$ hoặc $\lfloor \mu(x) \rfloor = \lfloor \nu(x) \rfloor$,
- $(\forall xy)$ nếu $\mu(x) \leq c_x$ và $\mu(x) < leqc_y$ Thì $fr(\mu(x)) \leq fr(\mu(y)) \Leftrightarrow fr(\nu(x)) \leq fr(\nu(y))$, và
- $(\forall x)$ nếu $\mu(x) \leq c_x$ thì $fr(\mu(x)) = 0 \Leftrightarrow fr(\nu(x)) = 0$.

trong đó, với r là một số thực ta kí hiệu $\lfloor r \rfloor$ là phần nguyên của r, còn $fr(r)$ là phần thập phân của r. $(l, \mu) \approx (p, \nu)$ nếu $l = p$ và $\mu \approx \nu$. Một vùng là một lớp tương đương theo phép $\approx$.

Với việc chia vùng tương đương trên TA, ta có kết quả về tính quyết định của ô-tô-mát được thể hiện qua định lý sau theo [4].

Định lý 2.2 *Bài toán về trạng thái tới được của ô-tô-mát thời gian là quyết định được.*

Hệ quả sau có được từ việc suy dẫn trực tiếp từ định lý trên.

Hệ quả 2.2 *Bài toán kiểm tra tính rỗng của ngôn ngữ của một ô-tô-mát thời gian là quyết định được*

Ví dụ về bài toán kiểm chứng

Chúng ta hãy xem xét một ví dụ về một bộ điều khiển tự động đóng mở thanh chắn (cổng) tại một nút giao thông đường sắt và đường bộ. Hệ thống này bao gồm ba thành phần: TRAIN, GATE và CONTROLLER được mô tả bởi ba ô-tô-mát thời gian tương ứng như trong Hình 2.6. Đây là ví dụ điển hình cho minh họa mạng các TA và được trích dẫn từ tài liệu [4].

Tàu được mô hình bằng ô-tô-mát thời gian TRAIN. Tập các sự kiện của tàu là $approach, exit, in, out, id_T$. Tàu bắt đầu ở trạng thái s_0. Các hành động id_T biểu diễn cho sự kiện trạng thái chờ, tức là tàu không được vào cửa. Các tàu giao tiếp với bộ điều khiển với hai sự kiện *approach* (tiếp cận) và *exit* (thoát). Các sự kiện *in* (trong) và *out* (ngoài) đánh dấu những sự kiện vào ra của đoàn tàu qua đường sắt. Các tàu được yêu cầu để gửi tín hiệu tiếp cận ít nhất 2 phút trước khi nó qua đường. Do đó, sự chậm trễ tối thiểu giữa *approach* và *in* là 2 phút. Hơn nữa, chúng ta biết rằng sự chậm trễ tối đa giữa *approach* và *in* là 5 phút. Cả hai yêu cầu thời gian được thể hiện bằng cách sử dụng một đồng hồ duy nhất x.

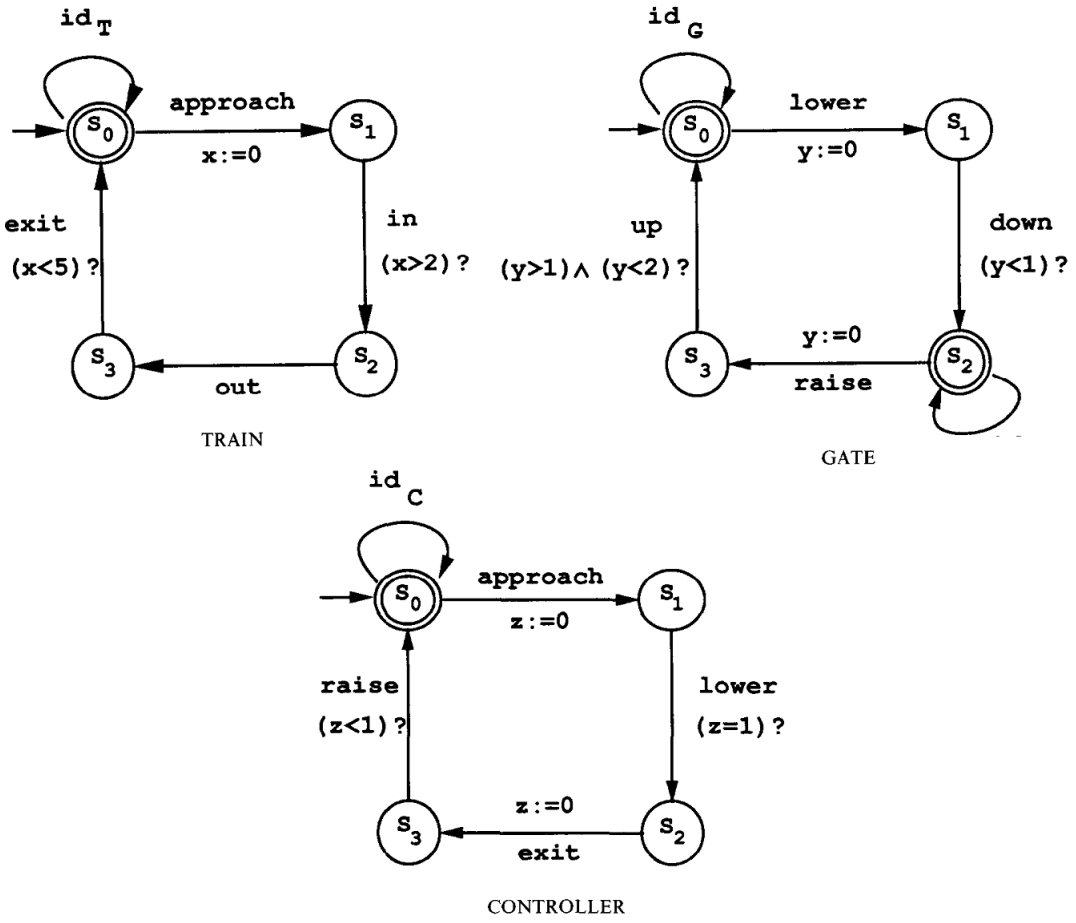
Ô-tô-mát thời gian mô tả cổng là GATE. Sự kiện này đặt là $raise, lower, up, down, id_G$. Các cổng được mở ở trạng thái s_0 và đóng tại trạng thái s_2. Nó giao tiếp với bộ điều khiển thông qua các tín hiệu *lower* (thấp hơn) và *raise* (nâng cao). Các sự kiện *up* (lên) và *down* (xuống) biểu khai hành động đóng mở của cổng. Cổng trả về tín hiệu *lower* bằng cách đóng cổng trong vòng 1 phút, và tín hiệu *raise* trong vòng 1-2 phút. Các cổng có thể giữ trạng thái *idl* (chờ) tại trạng thái s_0 hoặc s_2 mãi mãi.

Ô-tô-mát thời gian mô tả bộ điều khiển là CONTROLLER. Các sự kiện tập hợp là $approach, exit, raise, lower, id_C$. Bộ điều khiển chờ ở trạng thái s_0. Bất cứ khi nào nó nhận được tín hiệu *approach* từ tàu, nó phản hồi bằng cách gửi các tín hiệu *lower* đến cổng. Thời gian đáp ứng là 1 phút. Bất cứ khi nào nó nhận được tín hiệu *exit*, nó phản hồi với một tín hiệu *raise* đến cổng trong vòng 1 phút. Khi đó, hệ thống sẽ là một ghép nối song song của ba ô-tô-mát TRAIN, GATE và CONTROLLER, tức là TRAIN||GATE||CONTROLLER. Ví dụ về hệ thống điều khiển thanh chắn đường tàu được minh họa trong Hình 2.6, đây là ví dụ điển hình cho minh họa mạng các TA và được trích dẫn từ tài liệu [4].

Các thuộc tính cần kiểm chứng bao gồm thuộc tính an toàn (safety) và liveness của hệ thống và được biểu diễn bởi các ô-tô-mát trong Hình 2.7.

1. Safety: Bất cứ khi nào tàu vào bên trong, cổng nên được đóng lại.

2. Real-time Liveness: Các cửa không bao giờ đóng lâu hơn 10 phút.



Hình 2.6: Mô hình hệ thống điều khiển thanh chắn tàu

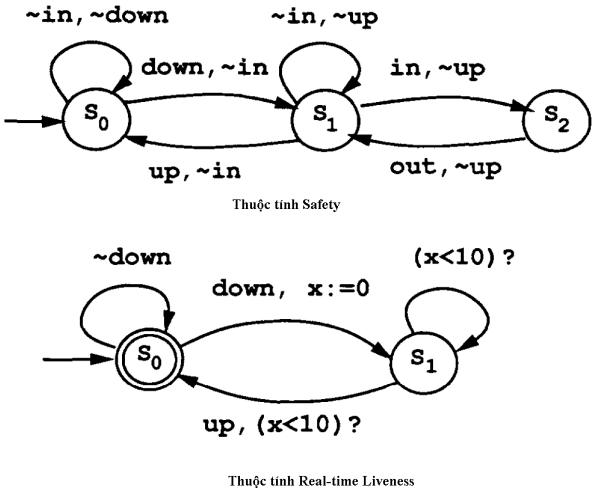
Do đó, đến nay, TA trở thành một công cụ hiệu quả cho việc đặc tả và kiểm chứng các hệ thống có yếu tố thời gian mà kết quả của nó là sự ra đời của một số bộ kiểm chứng mô hình có đặc tả bằng TA mà điển hình là bộ kiểm chứng mô hình UPPAAL. Bên cạnh đó, bài toán kiểm chứng cho đặc tả bằng TA vẫn đang được nghiên cứu nhằm tối ưu các thuật toán đã có trước. Phần tiếp theo luận án sẽ giới thiệu những đặc điểm cơ bản nhất về công cụ kiểm chứng UPPAAL.

2.2.3 Công cụ UPPAAL

Trong thời gian vài năm trở lại đây, có một số công cụ được phát triển dựa trên TA để mô hình và kiểm chứng các hệ thống có yếu tố thời gian (RTS), đáng chú ý nhất là Kronos[6] và UPPAAL [10]. Trong phần này, luận án sẽ giới thiệu về công cụ UPPAAL, đây là một bộ kiểm chứng mô hình cho việc mô hình, mô phỏng và kiểm chứng các ô-tô-mát thời gian. Công cụ này được giới thiệu đầu

[6]http://www-verimag.imag.fr/DIST-TOOLS/TEMPO/kronos/

tiên năm 1995, và kể từ đó nó đã được phát triển và duy trì trong sự cộng tác giữa Uppsala University và Aalborg University.



Hình 2.7: Thuộc tính Safety và Real-time Liveness của bài toán mô hình hệ điều khiển đóng mở thanh chắn tàu

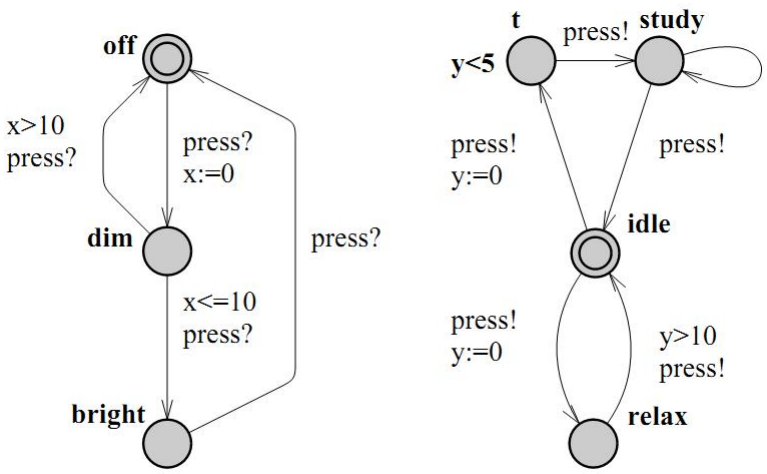
Mô hình hệ thống với UPPAAL

Thành phần quan trọng chính trong ngôn ngữ mô hình của UPPAAL là các ô-tô-mát thời gian. Thêm vào đó, ngôn ngữ đã được mở rộng hơn để dễ dàng mô hình công việc và hướng dẫn kiểm chứng trong không gian trạng thái lớn. Điều quan trọng nhất ở đây là được chia sẻ các biến nguyên, các kênh khẩn cấp và các vị trí cam kết.

Mô hình mạng các Ô-tô-mát thời gian

Một mạng các TA là một sự ghép nối song song của một tập các TA, được goi là các tiến trình, kết hợp chúng vào một hệ thông đơn lẻ bởi toán tử ghép nối song song CCS với tất cả các hành động bên trong được ẩn giấu.

Giao tiếp đồng bộ giữa các tiến trình bằng việc bắt tay đồng bộ giữa chúng và sử dụng các hành động đầu vào và đầu ra; giao tiếp bất đồng bộ được thực hiện bởi các biến chia sẻ được mô tả sau đây. Để đặc tả bắt tay đồng hộ, bảng kí hiệu hành động Σ được giả thiết bao gồm các kí hiệu hành động đầu vào kí hiệu là $a?$ và đầu ra, kí hiệu là $a!$ và các hành động bên trong biểu diễn bởi kí hiệu phân biệt τ. Một ví dụ về hệ thống được ghép nối bởi hai TA được chỉ ra trong Hình 2.8.



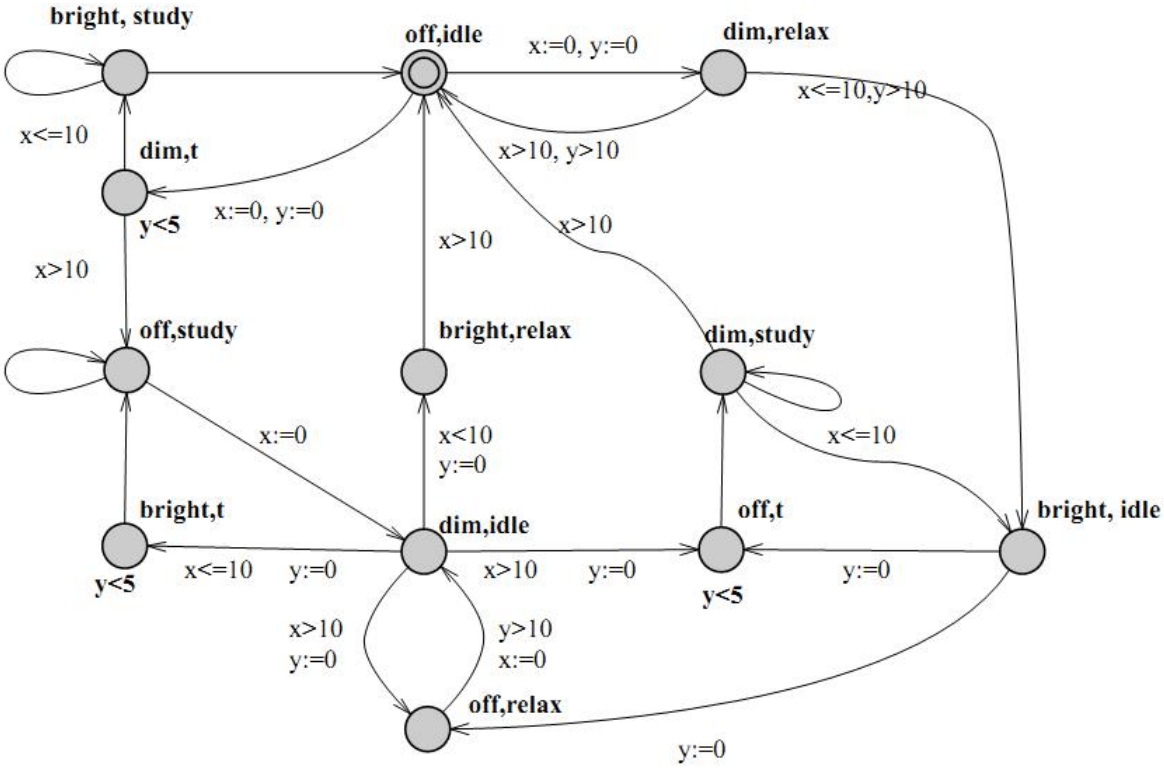
Hình 2.8: Mạng các ô-tô-mát thời gian

Mạng TA mô hình một công tắc bật đèn phụ thuộc thời gian (bên trái) và người dùng (bên phải). Người dùng và công tắc giao tiếp với nhau sử dụng nhãn press (nhấn). Người dùng có thể nhấn công tắc (press!) và công tắc đợi được nhấn (press?). Ô-tô-mát mô hình công tắc đèn được diễn tả như sau: Đèn có ba trạng thái là tắt (off), mờ (dim) và sáng (bright). Khi người dùng nhấn công tắc (press) một lần (lần đầu) thì đèn chuyển sang trạng thái mờ, trong khoảng 10 (giây), nếu không nhấn thêm lần nữa thì đèn sẽ chuyển sang trạng thái tắt, ngược lại thì đèn sẽ sang trạng thái sáng. Khi đèn sáng nếu nhấn công tắc thì đèn sẽ chuyển sang trạng thái tắt. Ô-tô-mát mô hình hoạt động của người sử dụng được diễn tả như sau: người dùng ban đầu tại trạng thái nghỉ (idle), khi muốn thư giãn (relax) sẽ nhấn công tắc một lần để đèn chuyển sang trạng thái mờ. Sau khoảng thời gian lớn hơn 10 (giây), người dùng không muốn thư giãn nữa thì nhấn công tắc đèn lần nữa để về trạng thái idle. Khi muốn học, người dùng sẽ nhấn công tắc đèn hai lần trong khoảng nhỏ hơn 5(giây) để đèn sáng và chuyển sang trạng thái học tập. Người dùng cũng có thể nhấn thêm công tắc 1 lần nữa để tắt đèn và chuyển sang trạng thái idle. Ô-tô-mát tích, tức là ô-tô-mát mô tả hệ thống được kết hợp được chỉ ra trong Hình 2.9.

Các biến nguyên được chia sẻ: Các đồng hồ có thể được xem xét như là các biến có kiểu là đồng hồ. Trong UPPAAL, có thể sử dụng các biến nguyên và mảng các số nguyên, mỗi biến với một miền ràng buộc và một giá trị khởi tạo ban đầu. Các vị từ trên các biến nguyên này có thể được sử dụng như là các ràng buộc trên các cạnh. Trong phiên bản hiện thời của UPPAAL, cú pháp liên quan tới các biến nguyên cũng giống như cú pháp chuẩn của C. Các ràng buộc

nguyên và các khởi tạo lại đồng hồ là theo chuẩn biểu thức C với giới hạn sao cho các ràng buộc không có hiệu ứng phụ.

Ngữ nghĩa của mạng có thể được định nghĩa một cách phù hợp. Phép gán đồng hồ trong các cấu hình trạng tháo có thể được mở rộng để lưu trữ các giá trị của các biến nguyên thêm vào các đồng hồ. Bởi vì độ trễ không ảnh hưởng tới các biến nguyên, các dịch chuyển theo độ trễ (dịch chuyển giữ nguyên vị trí) cũng giống như đối với các mạng không có các biến nguyên. Các dịch chuyển hành động được mở rộng theo một cách tự nhiên, tức là một dịch chuyển hành động được kích hoạt, phép gán đồng hồ được mở rộng phải thỏa tất cả các ràng buộc số nguyên trên các cạnh tương ứng và khi một dịch chuyển xảy ra phép gán được cập nhật theo số nguyên và đồng hồ thiết lập lại.



Hình 2.9: Ô-tô-mát tích của hai ô-tô-mát trong Hình 2.8

Có một vấn đề với biến đang cập nhật trong một sự đồng bộ dịch chuyển với một trong các tiến trình tham gia vào trong dịch chuyển cập nhật một biến được sử dụng bởi biến khác. Trong UPPAAL, đối với một dịch chuyển đồng bộ, sự thiết lập trên các cạnh với một nhãn đầu ra được thực hiện trước khi các thiết lập trên cạnh với một nhãn đầu vào. Điều này phá hủy sự đối xứng của các hành động đầu vào và đầu ra. Nhưng nó đưa ra một ngữ nghĩa rõ ràng và tự nhiên cho các biến cập nhật. Luật dịch chuyển cho sự đồng bộ hóa được sửa

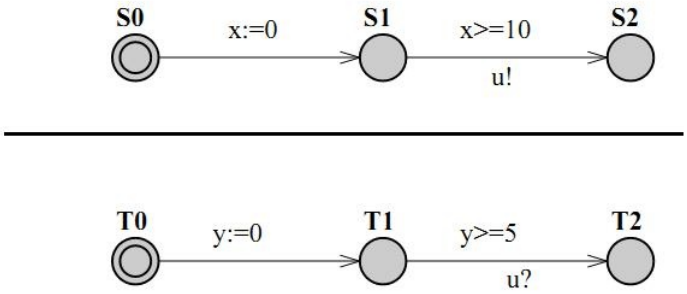
đổi một cách phù hợp như sau:

$(l, u) \xrightarrow{\tau} (l[l'_i/l_i][l'_j/l_j], u')$ nếu tồn tại $(i \neq j)$ sao cho:

i. $l_i \xrightarrow{g_i, a?, r_i} l'_j$, $l_j \xrightarrow{g_j, a!, r_j} l'_i$ và $u \in g_i \wedge g_j$, và

ii. $u' = [r_1 \to 0]([r_i \to 0]u)$ và $u' \in I(l[l'_i/l_i][l'_j l_j], u')$.

Kênh khẩn cấp: Để mô hình các dịch chuyển đồng bộ khẩn, UPPAAL hỗ trợ một khái niệm kênh khẩn cấp. Một kênh khẩn cấp làm việc giống như một kệnh thông thường nhưng với một ngoại lệ là nếu sự đồng bộ trên kênh là có thể thì hệ thống có thể không trì hoãn (delay). Việc vào ra với các dịch chuyển hành động khác là có thể (enabled), tuy nhiên vẫn được cho phép. Với mục đích giữ các ràng buộc đồng hồ biểu diễn sử dụng các vùng lồi (convex zones) các ràng buộc đồng hồ không được cho phép trên các cạnh đang đồng bộ trên các kênh khẩn cấp. Để hiểu rõ tại sao giới hạn này là cần thiết, ta có thể xem xét mạng trong Hình 2.10



Hình 2.10: Ví dụ một mạng với các vùng thời gian không lồi

Cả hai tiến trình có thể là độc lập từ trạng thái đầu tiên của chúng tới trạng thái thứ hai. Trong trạng thái thứ hai, tiến trình đầu tiên phải trễ (trì hoãn) ít nhất 10 đơn vị thời gian trước khi nó được phép đồng bộ với kênh khẩn cấp. Trong trạng tháo thứ hai, tiến trình khác phải làm trễ ít nhất 5 đơn vị thời gian trước khi nó được cho phép đồng bộ trên kênh khẩn cấp. Ngay khi cả hai tiến trình phải dành thời gian tối thiểu được yêu cầu trong trạng thái thứ hai, chúng nên đồng bộ và di chuyển tới trạng thái thứ ba. Vấn đề ở đây là trong [S2,T2] với vùng, ví dụ là $(x \geq 10 \wedge y = 5) \vee (y \geq 5 \wedge x = 10)$, là một vùng không lồi.

Với vấn đề này, bài toán có thể được giải bằng việc chia vùng không lồi thành hai vùng lồi. Nhưng trong trường hợp tổng quát, việc chia nhỏ là một hành động

mất nhiều chi phí tính toán. Trong UPPAAL, chúng ta đã quyết định để tránh các hành động như vậy để tăng hiệu quả. Vì thế các ràng buộc số nguyên được cho phép trên các cạnh liên quan tới các đồng bộ hóa trên các kênh khẩn cấp.

Kiểm chứng với UPPAAL

Mô hình kiểm chứng của UPPAAL được thiết kế để kiểm tra một tập con của công thức TCTL (Timed Computational Tree Logic) [4] cho các mạng các TA. Công thức có các dạng biểu diễn như sau:

- $A[]\phi - Invariantly\phi$,
- $E <> \phi - Possible\phi$,
- $A <> \phi - Always\ Eventually\phi$,
- $E[]\phi - Always\ Possible\phi$, và
- $\phi - - > \psi - \phi$ always lead to ψ.

Với ϕ, ψ là các các thuộc tính cục bộ mà có thể được kiểm tra một cách cục bộ tại một trạng thái, tức là các biểu diễn nhị phân (boolean) trên các vị từ trên các vị trí và các biến nguyên và các ràng buộc đồng hồ trong $B(C)$ [11].

Hệ dịch chuyển trạng thái của một mạng có thể được mở ra trong một cây vô hạn chứa các trạng thái và các dịch chuyển. Các ngữ nghĩa của các công thức được định nghĩa trên cây này. Ký tự A và E được sử dụng để định lượng trên các đường dẫn. A được sử dụng để biểu diễn rằng thuộc tính được đưa ra cần được giữ trên tất cả các đường đi trên cây trong khi đó E biểu diễn rằng có ít nhất một đường đi trên cây mà thuộc tính được thỏa mãn.

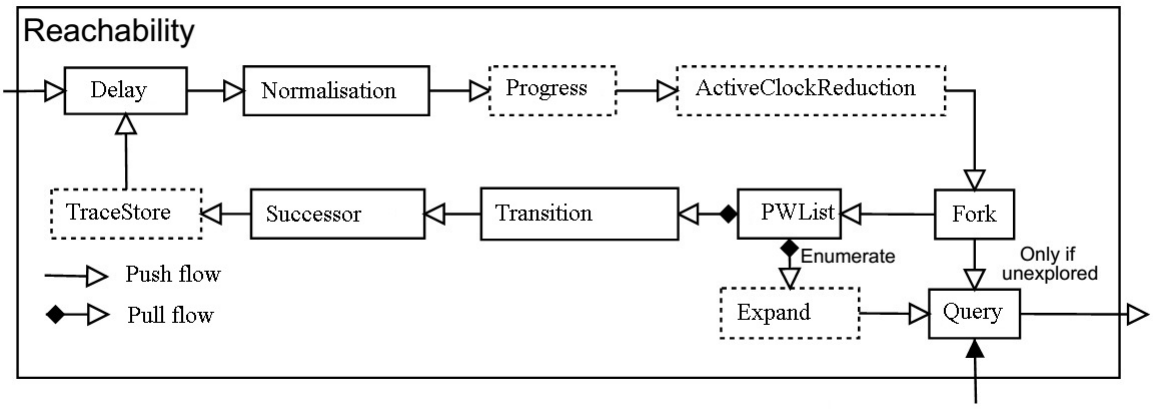
Ký hiệu [] và <> được sử dụng để định lượng trên các trạng thái trong đường đi. [] biểu diễn rằng tất cả các trạng thái trên đường đi thỏa thuộc tính trong khi<> để chỉ rằng có ít nhất một trạng thái trên đường thỏa thuộc tính.

Kiến trúc UPPAAL

Để cung cấp một hệ thống mà vừa hiệu quả lại dễ sử dụng, UPPAAL chia thành hai thành phần, một giao diện đồ họa người dùng được viết bằng JAVA và một máy kiểm chứng được viết bằng C++. Máy và giao diện ngường dùng giao tiếp sử dụng một giao thức, cho phép kiểm chứng được thực thi hoặc trên máy trạm hoặc trên một server trên mạng.

Để thực hiện phân tích giải bài toán đến được, máy kiểm chứng UPPAAL được tổ chức như là một đường ống [11]. Một phác họa đường ống này được chỉ ra trong Hình 2.11. Đây là kiến trúc đơn giản cho việc kích hoạt và không kích hoạt các tối ưu tại thời điểm thực thi bằng việc chèn thêm và gỡ bỏ các công đoạn một cách động. Ngoài việc thực thi thuật toán trên zone và kiến trúc đường ống, trong UPPAAL, một số các tối ưu được thực thi [11]:

- Hệ thống ràng buộc tối thiểu để giảm tiêu tốn bộ nhớ bằng việc bỏ các thông tin dư thừa trong zones trước khi lưu trữ.
- Lựa chọn lưu trữ các trạng thái trong PASSED, với phân tích tĩnh được sử dụng để phát hiện các trạng thái mà có thể bị bỏ qua một cách an toàn từ PASSED mà không làm mất sự hủy bỏ.
- Nén và chia sẻ các dữ liệu trạng thái để giảm tiêu thụ bộ nhớ của PASSED và WAIT.
- Giảm hoạt động đồng hồ, mà sử dụng phân tích live-range để quyết định khi nào giá trị của một đồng hồ là không liên quan. Điều này không chỉ giảm kích thước các trạng thới đơn lẻ mà còn nhận biếtkhông gian trạng thái.
- Supertrace và Hash Compaction nơi các trạng thái đã được thăm được lưu trữ chỉ như các chữ ký hash, và Convex-hull approximation nơi thân lồi được sử dụng để xấp xỉ phép hợp các zones để giảm tiêu tốn bộ nhớ tại các kết quả có nguy cơ không thuyết phục.



Hình 2.11: Kiến trúc hệ thống của UPPAAL

2.3 Lý thuyết Vết và ứng dụng trong đặc tả hệ thống tương tranh

2.3.1 Giới thiệu

Ngày nay, khi khoa học công nghệ và xã hội ngày càng phát triển thì con người càng phụ thuộc vào các hệ thống hỗ trợ hoạt động từ các thiết bị mang tính cơ khí như máy giặt tới cao cấp hơn như các hệ thống điều khiển máy bay, vũ trụ, v.v. Vấn đề đặt ra là làm thế nào để đảm bảo các hệ thống này luôn hoạt động theo đúng chức năng của nó, tức là không phát sinh lỗi? Câu hỏi này dẫn tới sự ra đời của các phương pháp hình thức. Các phương pháp này dựa trên lý thuyết toán học ứng dụng cho việc đặc tả và kiểm chứng tính đúng đắn của hệ thống.

Có rất nhiều các phương pháp hình thức được phát triển để hỗ trợ cho việc đặc tả và kiểm chứng một hệ thống phần mềm. Chúng đã góp phần thúc đẩy sự phát triển mạnh mẽ của công nghệ phần mềm. Các phần mềm hiện nay được phát triển trên cơ sở đảm bảo sự tin cậy, tính đúng đắn và thẩm định tốt hơn.

Trong quá trình phát triển, các hệ thống ngày càng phức tạp đòi hỏi các phương pháp đặc tả và kiểm chứng chúng cũng thay đổi theo một cách phù hợp. Đặc biệt, khi công nghệ truyền thông phát triển, các hệ thống tương tranh, phân tán cũng dần được hình thành và phát triển theo kéo theo nó là sự đòi hỏi về các nghiên cứu về phương pháp và công cụ dùng để đặc tả và kiểm chứng các hệ thống này. Các công việc này đã dẫn tới việc hình thành và phát triển lý thuyết Vết khởi đầu từ mạng Petri. Đến nay, các kết quả nghiên cứu liên quan tới lý thuyết Vết và các ứng dụng vẫn đang rất phát triển, trong đó đặc biệt quan tâm tới việc kiểm chứng một số tính chất trên Vết và biểu diễn lôgic thời gian tuyến tính đặc tả Vết. Lôgic thời gian tuyến tính (Linear Temporal Logic - LTL) là một công cụ được dùng để đặc tả hành vi của hệ thống trong các hệ phân tán [62]. Cách tiếp cận truyền thống về kiểm chứng chương trình tự động là kiểm chứng mô hình đặc tả bằng LTL. Đặc điểm cơ bản của LTL là ý nghĩa công thức của nó được thể hiện trên chuỗi. Một chuỗi sẽ mô tả một tính toán của một hệ thống, tức là một chuỗi các trạng thái được thăm của hệ thống hoặc một chuỗi các hành đuộng được thực hiện bởi hệ thống trong quá trình tính toán.

Thực tế rằng, các tính toán của một hệ phân tán tạo thành các Vết Mazurkiewicz. Rất nhiều thuộc tính được biểu diễn bằng LTL có tính chất "không hoặc tất cả", tức là hoặc tất cả các thành viên của một lớp tương đương các tính toán sẽ có thuộc tính mong ước hoặc không (Dẫn tới khóa chết là một thuộc tính như vậy). Để kiểm chứng những thuộc tính như vậy, chúng ta phải

kiểm tra chúng chỉ cần cho một thành viên của lớp tương đương. Tuy nhiên, có một vấn đề trong đặc tả các thuộc tính trong LTL trên từ (word) là không hỗ trợ Vết. Từ quan điểm thực tế, sẽ là không thuận tiên khi cho phép một người dùng sau của một công cụ kiểm chứng xây dựng công thức cho các ràng buộc như vậy. Người dùng nên được hướng dẫn để đặc tả chỉ các ràng buộc tương ứng với cấu trúc kế thừa của một hệ thống cơ bản. Vấn đề này có thể được giải quyết bằng việc phát triển LTL mà có thể được thể hiện trực tiếp trên các Vết Mazurkiewicz. Trong các lôgic này, mọi đặc tả được đảm bảo có thuộc tính "tất cả hoặc không" và do đó có thể là mục tiêu hướng tới các phương pháp giảm bớt không gian trạng thái dựa trên nguyên lý thứ tự bộ phận trong các tiến trình kiểm chứng.

Có nhiều phát triển LTL trên Vết như [58, 62, 14, 68]. Trong phần này, luận án sẽ giới thiệu những khái niệm và kết quả nghiên cứu cơ bản nhất đã được đề xuất về Vết Mazukiewicz và lôgic trên Vết.

2.3.2 Vết Mazurkiewicz

Trong phần này, luận án sẽ trình bày các khái niệm và các kết quả nghiên cứu về lý thuyết Vết Mazurkiewicz (sau này viết tắt là Vết) được tổng hợp từ các tài liệu [13, 30, 31, 34].

Bảng chữ cái

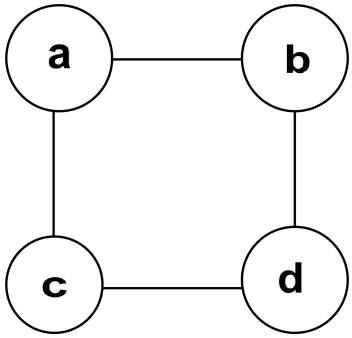
Trước tiên, luận án nhắc lại một số khái niệm về Vết Mazurkiewicz được giới thiệu trong [30, 31, 34].

Định nghĩa 2.7 *Ký hiệu Σ là một tập hữu hạn các phần tử và được gọi là bảng chữ cái, $D \subset \Sigma \times \Sigma$ là một quan hệ hai ngôi có tính phản xạ và đối xứng trên Σ và được gọi là quan hệ phụ thuộc. Quan hệ độc lập I là phần bù của D. Một bảng chữ cái phụ thuộc là một cặp (Σ, D).*

Chúng ta gọi (Σ, I) là bảng chữ cái độc lập, do quan hệ độc lập I là phần bù của quan hệ độc lập D nên trong một số trường hợp, để cho thuận tiện, chúng ta có thể sử dụng bảng chữ cái độc lập mà không ảnh hưởng tới nội dung và kết quả nghiên cứu. Cho $A \subseteq \Sigma$, tập các chữ cái phụ thuộc trên A được ký hiệu là $D(A) = \{b \in \Sigma | (a, b) \in D \text{ với } a \in A\}$.

Ví dụ 2.4 *Cho bảng chữ cái phụ thuộc (Σ, D) với tập $\Sigma = \{a, b, c, d\}$ và quan hệ phụ thuộc $D = \{(a, c), (c, a), (a, b), (b, a), (c, d), (d, c), (b, d), (d, b), (a, a), (b, b), (c, c), (d, d)\}$.*

Đồ thị phụ thuộc của (Σ, D) ký hiệu là $G(V, E)$ [51] là đồ thị mà có tập đỉnh $V = \Sigma$ và tập cạnh $E = \{(a, b) \in \Sigma \times \Sigma | (a, b) \in D$ và $a \neq b\}$ được chỉ rõ trong Hình 2.12.



Hình 2.12: Đồ thị phụ thuộc của bảng chữ cái phụ thuộc

Ta thấy rằng, đồ thị phụ thuộc của một bảng chữ cái hoàn toàn độc lập sẽ không có cạnh và một trong một bảng chữ cái hoàn toàn phụ thuộc sẽ là một đồ thị đầy đủ. Với một bảng chữ cái phụ thuộc bắc cầu, đồ thị phụ thuộc của nó bao gồm các thành phần kết nối là đầy đủ.

Trong định nghĩa về bảng chữ cái độc lập/phụ thuộc, bảng chữ cái hoàn toàn phụ thuộc sẽ được sử dụng để mô tả hệ thống tuần tự trong khi bảng chữ cái hoàn toàn độc lập sẽ đại diện cho các hệ thống song song hoạt động tự chủ, tức là các tiến trình không giao nhau hoặc tương tác với nhau.

Quan sát một hệ thống (phức tạp) từ một điểm bên ngoài, người ta có thể thấy rằng hệ thống có khả năng thực hiện hành động. Hơn nữa, người ta có thể nhận ra sự phụ thuộc của những hành động này. Như vậy, xem một tập đơn lẻ của hành động (cùng với một mối quan hệ phụ thuộc/độc lập) là phù hợp cho một cái nhìn bên ngoài vào một hệ thống có cấu trúc bên trong mà ta không biết.

Tuy nhiên, nếu ta biết được cấu trúc bên trong của một hệ thống, chúng ta cần xem xét hệ thống ở một khía cạnh thích hợp hơn. Chúng ta có thể hiểu các hệ thống như một tập n các tiến trình mà mỗi tiến trình có khả năng thực hiện các hành động chỉ trong Σ_i. Những hành động này được sử dụng cho truyền thông giữa các tiến trình hoặc tương ứng với hành động bên trong của một tiến trình. Điều này dẫn chúng ta đến khái niệm về một bảng phân tán, khái niệm do Zielonka đề xuất trong [75].

Định nghĩa 2.8 (Bảng chữ cái phân tán) *Một bảng chữ cái phân tán (trên n tiến trình) là một n-bộ $\widetilde{\Sigma} = \{\Sigma_1, ..., \Sigma_n\}$ của các bảng chữ cái Σ_i (không cần thiết phải giao nhau).*

Kí hiệu $Proc = \{1, 2, ..., n\}$ là một tập các chỉ số tiến trình. Trong giải thích của chúng ta, một hành động a trong Σ_i và Σ_j đối với hai giá trị khác nhau $i, j \in Proc$ có thể được sử dụng cho một giao tiếp của các quá trình khác nhau i và j, tức là hai tiến trình có chung một hành động được thực hiện là a. Để chỉ ra sự tương ứng của bảng chữ cái phân tán và độc lập, chúng ta giới thiệu các phép toán *alphabet* and *pr*.

Định nghĩa 2.9 *Cố định bảng chữ cái $\widetilde{\Sigma}$, khi đó bảng chữ cái $\Sigma = alphabet_{\widetilde{\Sigma}}$ được định nghĩa như sau: $alphabet_{\widetilde{\Sigma}} = \Sigma \widehat{=} \Sigma_1 \cup \Sigma_2 \cup ... \cup \Sigma_n$ là tập các ký hiệu khác nhau của các bảng chữ cái thành phần, $pr_{\widetilde{\Sigma}}(a) = \{i \in Proc(\widetilde{\Sigma}) | a \in \Sigma_i\}$ là tập các tiến trình mà có hành động a được thực thi, và $D(\widetilde{\Sigma}) = \{(a, b) \in \Sigma \times \Sigma | pr(a) \cap pr(b) = \emptyset\}$ là quan hệ phụ thuộc giữa các hành động trên bảng chữ cái phân tán $\widetilde{\Sigma}$.*

Từ định nghĩa trên, ta có $(\Sigma, D(\widetilde{\Sigma}))$ là một bảng chữ cái phụ thuộc. Mặt khác, cho (Σ, D), chúng ta có một bảng chữ cái phân tán duy nhất $(\widetilde{\Sigma})$ sao cho $(\Sigma, D) = (\Sigma, D(\widetilde{\Sigma}))$ [51].

Ví dụ 2.5 *Cho bảng chữ cái phụ thuộc (Σ, D) trong ví dụ 2.4, ứng với mỗi quan hệ phụ thuộc $(a, b) \in D$ là một tiến trình. Như vậy ta có bảng chữ cái phân tán với 4 tiến trình $(\{a, b\}, \{a, c\}, \{c, d\}, \{b, d\})$ và ngược lại.*

Vết Mazurkiewicz

Hành vi của hệ thống có thể được mô tả bởi các hành động được thực thi. Bản chất của một chuỗi hành động trong hệ thống là nó có thể chỉ thực thi tuần tự các hành động. Vì thế, một thực thi của hệ thống có thể được mô tả bởi một chuỗi các hành động nối tiếp nhau. Cho một hệ thống tương tranh và cố định các phụ thuộc của các hành động, các hành động khác nhau của một sự thực thi trong hệ thống không hình thành một trật tự tuyến tính mà là một thứ tự bộ phận. Vì thế, chúng ta định nghĩa một thực thi của một hệ thống tương tranh là một thứ tự bộ phận. Cho Σ là bảng chữ cái (biểu diễn các hành động trong hệ thống) và D là một quan hệ phụ thuộc (biểu diễn quan hệ phụ thuộc nhau giữa các hành động trong một hệ thống) như định nghĩa phía trên, Vết Mazurkiewicz được định nghĩa như sau[51]:

Định nghĩa 2.10 (Vết Mazurkiewicz) *Một Vết Mazurkiewicz (gọi tắt là Vết) là một lớp tương đương của một thứ tự bộ phận được gán nhãn $T = \langle V, \leq, \lambda \rangle$ với V là tập các nút, $\leq$ là một quan hệ thứ tự bộ phận trên V, và $\lambda : V \longrightarrow \Sigma$ là một hàm gán nhãn mỗi nút của T với một ký hiệu hành động thuộc bảng chữ cái Σ thỏa mãn điều kiện sau:*

- *Với mỗi $x \in V$ tập các đỉnh trước x theo quan hệ $\leq$, được kí hiệu là $\downarrow x$, $\downarrow x \widehat{=} \{y \in V | y \leq x\}$ là hữu hạn*[7]*, và*
- *Với mọi $x, y \in V$, nếu $(\lambda(x), \lambda(y)) \in D$ thì $x \leq y$ hoặc $y \leq x$, và $x \lessdot y$ kéo theo $(\lambda(x), \lambda(y)) \in D$, với $\lessdot$ là một quan hệ nhỏ hơn trực tiếp trong T được định nghĩa là $x \lessdot y$ nếu và chỉ nếu $x < y$ và $\forall z \in V, x \leq z \leq y$ kéo theo $x = z$ hoặc $y = z$.*

Vết T biểu diễn sự thực thi của một hệ tương tranh, trong đó tập đỉnh V kết hợp với hàm λ thể hiện thứ hành động được thực thi trong hệ thống, quan hệ $\leq$ trên V thể hiện thứ tự trước sau của các hành động được thực hiện trong một thực thi của hệ tương tranh mà được biểu diễn bởi vết T. Nếu V là hữu hạn, Vết T được gọi là Vết hữu hạn. Tập tất cả các Vết trên (Σ, D) kí hiệu là $Tr(\Sigma, D)$. Đặt $alph(T) \widehat{=} \lambda(V)$ là bảng chữ cái của T. Phép nối tiếp của Vết $T_1 = \langle\langle V_1, \leq_1, \lambda_1\rangle$ và $T_2 = \langle V_2, \leq_2, \lambda_2\rangle$ được định nghĩa là $T_1.T_2 \ \widehat{=} \ \langle V, \leq, \lambda\rangle$ với V là hợp V_1 và V_2, $\lambda \ \widehat{=} \ \lambda_1 \cup \lambda_2$, và $\leq$ có quan hệ đóng bắc cầu $\leq_1 \cup \leq_2 \cup ((V_1 \times V_2) \cap \lambda^{-1}(D))$. Tập các Vết hữu hạn trên (Σ, D) trở thành một nửa nhóm với phần tử rỗng là Vết rỗng $\langle\emptyset, \emptyset, \emptyset\rangle$ là phần tử đơn vị. Khi $T = R.S$ chúng ta nói R là tiền tố và S hậu tố của T. Hình 2.13 là một ví dụ của Vết Mazurkiewicz. Các ví dụ khác có thể xem trong [51].

Cho một Vết T chúng ta kí hiệu tập các đỉnh tối thiểu của nó là $\min(T) \widehat{=} \{v \in V | \nexists v'.v' < v\}$. Một nhát cắt của T là một tập lớn nhất của các đỉnh không so sánh được trong V. Vì thế, $\min(T)$ cũng là một nhát cắt của T. Một cách tương tự, $\max(T) \widehat{=} \{v \in V | \nexists v'.v < v'\}$ cũng là một nhát cắt của T. Cho $\omega \in \Sigma^\omega$, $proj(w)$ biểu diễn phép chiếu của từ ω trên bảng chữ cái Σ, và $pref(\omega)$ biểu diễn tập tất cả các tiền tiền tố của từ ω. Một từ trong Σ^ω được gắn với một Vết trên (Σ, D) bằng ánh xạ $wtot\colon \Sigma^\omega \to Tr(\Sigma, D)$ được định nghĩa như sau:

Cho $\omega \in \Sigma^\omega, wtot(\omega)$ là Vết $\langle V, \leq, \lambda\rangle$ với:

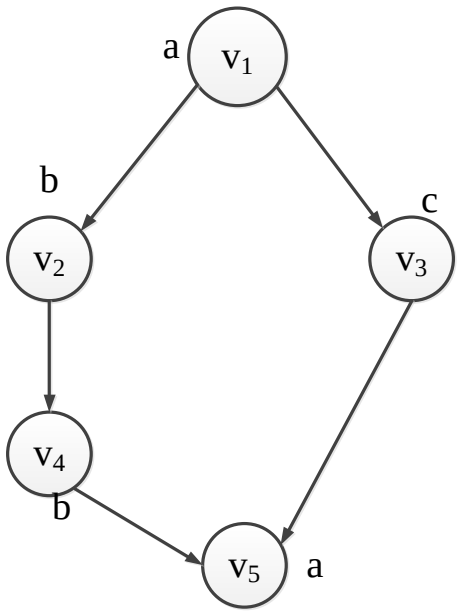
- $V = pref(\omega) - \{\epsilon\}$,
- $\leq$ là thứ tự nhỏ hơn bộ phận trên V thỏa mãn: cho $\tau a, \tau' b \in V$ nếu τa là tiền tố của $\tau' b$ và nếu $(a, b) \in D$ thì $\tau a \leq \tau' b$, và
- $\lambda(\tau a) = a$.

Theo kết quả trong [51], *wtot* là được định nghĩa tốt(well-defined). Vì thế, ω là một tuyến tính hóa (xem trong [50]) của $wtot(\omega)$ và $wtot(\omega)$ biểu diễn sự phụ thuộc giữa các chữ cái xuất hiện trong ω theo quan hệ phụ thuộc D (xem trong ví

[7]Ở đây, luận án sử dụng kí hiệu "$\widehat{=}$" có nghĩa là "được định nghĩa là"

dụ 2.6). Thứ tự bộ phận này thể hiện một hành vi đồng thời của một ô-tô-mát. Để cho đầy đủ, chúng ta sẽ định nghĩa ánh xạ $ttow\colon Tr(\Sigma, D) \to \Sigma^{\infty}$ như sau.

$$ttow(\langle V, \leq, \lambda\rangle)\widehat{=}\{\lambda(\delta)|\delta \text{ là một sự tuyến tính hóa của } (V, \leq)\} \text{ với}$$
$$(\lambda)(e_1e_2\ldots)\widehat{=}\lambda(e_1)\lambda(e_2)\ldots$$



Hình 2.13: Một đồ thị biểu diễn của Vết Mazurkiewicz

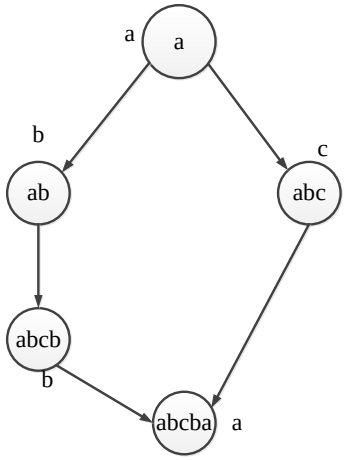
Ánh xạ *ttow* có thể được mở rộng để định nghĩa trên ngôn ngữ Vết như sau. Cho bất kỳ ngôn ngữ Vết L trên $(\Sigma, D), ttow(L)\widehat{=}\bigcup_{T\in L} ttow(T)$.

Ví dụ 2.6 *Cho $\omega = abcba$ với bảng chữ cái phụ thuộc cho trong ví dụ 2.4, khi đó $pref(\omega) = a, ab, abc, abcb, abcba$ Vết có được từ ánh xạ $wtot(\omega)$ được chỉ ra trong Hình 2.14.*

Định nghĩa 2.11 (Ngôn ngữ chính quy nhất quán) *Một ngôn ngữ chính quy (được đoán nhận bởi một Büchi ô-tô-mát) $L \subseteq \Sigma^{\omega}$ là nhất quán nếu và chỉ nếu với bất kỳ $\omega \in L$, và với bất kỳ $\omega' \in \Sigma^{\omega}$ chúng ta có: $wtot(\omega) = wtot(\omega')$ kéo theo $\omega' \in L$. Đối với ngôn ngữ chính quy L như vậy, chúng ta cũng gọi L là một ngôn ngữ ω-chính quy nhất quán trên Σ.*

Từ định nghĩa trên ta dễ dàng suy ra $ttow(wtot(L)) = L$.

Định nghĩa 2.12 (Ngôn ngữ Vết chính quy) *Ta gọi một ngôn ngữ Vết $L \subseteq Tr(\Sigma, D)$ là chính quy khi và chỉ khi $\bigcup\{ttow(T)|T \in L\}$ là một ngôn ngữ chính quy.*



Hình 2.14: Ánh xạ *wtot*() cho từ *abcba*

Cấu hình và đồ thị cấu hình

Một hệ thống tương tranh (hay đồng thời) được nghiên cứu theo khía cạnh các thực thi của nó. Chúng ta muốn phân tích một thực thi trong khi nó đang mở ra từng bước một. Cho một chuỗi, một tiền tố của chuỗi là một phần của thực thi sau một khoảng thời gian trôi qua. Có cái gì tương tự với khái niệm trong Vết?

Gọi $T = \langle V, \leq, \lambda \rangle$ là một Vết trên (Σ, D). Chúng ta đã định danh một sự kiện $v \in V$ để biểu diễn một bước thực thi của hệ thống tương ứng với hành động $\lambda(v)$. Theo định nghĩa thể hiện, chúng ta yêu cầu tất cả các sự kiện v' trước v, tức là $v' \in \downarrow v \backslash \{v\}$ đã phải thực thi trước rồi. Vì thế, một cách tự nhiên để mô hình một phần của một thực thi T bởi một tập các sự kiện Ç mà chứa tất cả các sự kiện cũng như lịch sử của nó, tức là với mọi $v \in Ç$, chúng ta có $\downarrow v \subseteq Ç$. Chúng ta gọi Ç là một cấu hình của chuỗi thực thi. Ta có định nghĩa hình thức như sau.

Định nghĩa 2.13 (Cấu hình) *Gọi $T = \langle V, \leq, \lambda \rangle$ là một Vết trên (Σ, D) và Ç là tập các nút $Ç \subseteq V$, lịch sử của Ç, ký hiệu là $\downarrow Ç$, được định nghĩa như sau:*

$$\downarrow Ç = \bigcup_{v \in Ç} \downarrow v.$$

Một cấu hình của T là một tập hữu hạn $Ç \subseteq V$ sao cho $\downarrow Ç = Ç$.

Chúng ta định nghĩa *conf(T)* là tập tất cả các cấu hình của Vết T. Các cấu hình sẽ đóng vai trò cốt yếu trong lý thuyết Vết. Những quan hệ của chúng là đối tượng cơ bản cho việc đặc tả và phân tích các hệ thống đồng thời.

Định nghĩa 2.14 *Gọi $T = \langle V, \leq, \lambda \rangle$ là một Vết và $conf(T)$ là tập các cấu hình của nó. Tất cả các cấu hình $Ç \in conf(T)$ tạo ra một Vết hữu hạn $T' = \langle Ç, \leq |_{Ç \times Ç}, \lambda|_{Ç} \rangle$. T' được gọi là tiền tố của T. Chúng ta ký hiệu $prf(T)$ là tập các tiền tố của Vết T.*

Tiền tố của một chuỗi có thể có thứ tự tuyến tính theo một cách rõ ràng, tức là các hành động xuất hiện trong tiền tố đó là phụ thuộc nhau. Bằng trực giác, một tiền tố là lớn hơn tiền tố khác khi và chỉ khi tiền tố đầu biểu diễn hệ thống tại thời điểm trước tiền tố thứ hai. Các cấu hình của $conf(T)$ được định nghĩa tương tự các tiền tố của các chuỗi hữu hạn. Tập các cấu hình của T có thể được sắp thứ tự theo một cách tự nhiên. Cho hai cấu hình $Ç, Ç' \in conf(T)$, chúng ta nói $Ç'$ lớn hơn Ç nếu và chỉ nếu $Ç \subset Ç'$, và do đó, nếu và chỉ nếu $Ç'$ mô tả cấu hình của Vết trong đó tất cả các sự kiện của Ç và các sự kiện khác nữa đã xảy ra. Điều này sinh ra một thứ tự bộ phận.

Hơn thế, các cấu hình có thể được trang bị một quan hệ chuyển $\longrightarrow_T \subseteq conf(T) \times \Sigma \times conf(T)$. Cho hai cấu hình $Ç, Ç'$ thuộc $conf(T)$, chúng ta nói $Ç'$ là một a-bước chuyển (a-successor) của Ç và được ký hiệu bởi $Ç \xrightarrow{a}_T Ç'$ khi và chỉ khi Ç có thể được thêm sự kiện a để trở thành $Ç'$.

Định nghĩa 2.15 *Cho T là một Vết và $conf(T)$ là tập các cấu hình của nó. Ta định nghĩa $\longrightarrow_T \subseteq conf(T) \times \Sigma \times conf(T)$ bởi $Ç \xrightarrow{a}_T Ç'$ khi và chỉ khi tồn tại v thuộc V sao cho $\lambda(v) = a, v \notin Ç$ và $Ç' = Ç \cup \{v\}$. Đồ thị cấu hình của T ký hiệu là $CG(T) = (conf(T), \longrightarrow_T)$.*

2.3.3 Ô-tô-mát đoán nhận ngôn ngữ Vết

Trong phần này, luận án sẽ tìm hiểu một số ô-tô-mát có thể dùng để đoán nhận ngôn ngữ Vết chính quy. Có nhiều loại ô-tô-mát khác nhau nhưng nhìn chung có thể chia làm hai loại (hai khái niệm) cơ bản là ô-tô-mát Alternating (tạm dịch là ô-tô-mát xen kẽ)và ô-tô-mát bất đồng bộ. Trong phần này, luận án giới thiệu tổng quan về hai loại ô-tô-mát này và trình bày chi tiết hơn về ô-tô-mát bất đồng bộ, các khái niệm chi tiết về ô-tô-mát Alternating có thể đọc hiểu trong tài liệu tham khảo [16, 20, 58, 75].

Ô-tô-mát Alternating Büchi

Alternating Ô-tô-mát mở rộng ô-tô-mát không đơn định bằng các lựa chọn chung (universal). Hàm chuyển không biểu diễn tập các trạng thái kế tiếp nữa mà là một sự kết hợp nhị phân. Alternating trong ngữ cảnh ô-tô-mát được nghiên cứu

trong [16, 20] và sau đó được nghiên cứu mở rộng cho việc đoán nhận ngôn ngữ Vết như trong [58].

Định nghĩa 2.16 *Cho tập cố định các biến X, ký hiệu tt và ff là giá trị true và false tương ứng, gọi $B^+(X)$ là tập các công thức nhị phân trên X là tập nhỏ nhất thỏa mãn:*

- $X \subseteq B^+(X)$,
- $tt, ff \in B^+(X)$, *và*
- $\varphi, \psi \in B^+(X) \Rightarrow \varphi \wedge \psi \in B^+(X), \varphi \vee \psi \in B^+(X)$.

Định nghĩa 2.17 (Ô-tô-mát Alternating Büchi) *Một ô-tô-mát Alternating Büchi $\mathcal{A} = (Q, \delta, q_0, F)$ trên bảng chữ cái Σ trong đó:*

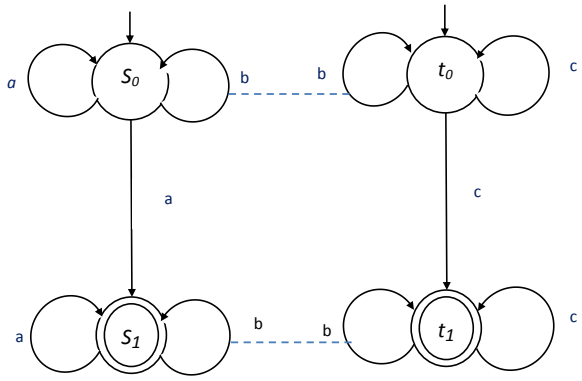
- Q *là tập hữu hạn khác rỗng các trạng thái,*
- $q_0 \in Q$ *là trạng thái khởi tạo,*
- $F \subseteq Q$ *là tập các trạng thái kết thúc, và*
- $\delta : Q \times \Sigma \longrightarrow B^+(Q)$ *là hàm chuyển trạng thái.*

Ô-tô-mát bất đồng bộ

Ô-tô-mát bất đồng bộ (Asynchronous Automata -AA) được nghiên cứu để mô hình các hệ thống tương tranh, ngôn ngữ nó đoán nhận là tập các Vết trên bảng chữ cái phân tán [75]. Do đó đây là ô-tô-mát được quan tâm nghiên cứu và phát triển công cụ cho việc đặc tả và kiểm chứng các hệ thống tương tranh. Cho $\omega \in \Sigma^\omega, proj_i(\omega)$ ký hiệu phép chiếu của từ ω trên Σ_i; và $pref(\omega)$ ký hiệu tập tất cả các tiền tố của ω. Cho tập $\{S_j\}_j \in Proc$, và $a \in \Sigma$ với $loc(a) = \{j_i, j_2, ..., j_k\}$, S_a là tích Đề-Các $\{S_{j1} \times S_{j2} \times ... \times S_{jk}\}$. Mỗi S_i là một tập trạng thái địa phương thứ i ký hiệu là i-local. F_i, G_i với mỗi $i \in Proc$ là các tập con của S_i với $i \in Proc$. Ký hiệu $S = S_1 \times S_2 \times ... \times S_{Proc}$,$F = F_1 \times F_2 \times ... \times F_{Proc}$, $G = G_1 \times G_2 \times ... \times G_{Proc}, S_{in} \subseteq S$ là tập các trạng thái ban đầu.

Bảng 2.2: Bảng chuyển của ô-tô-mát bất đồng bộ của Hình 2.15

$$
\begin{array}{rcl}
(s_0, *) & \xrightarrow{a} & (s_0, *) \\
(s_0, *) & \xrightarrow{a} & (s_1, *) \\
(s_1, *) & \xrightarrow{a} & (s_1, *) \\
(s_1, t_0) & \xrightarrow{b} & (s_1, t_0) \\
(s_1, t_1) & \xrightarrow{b} & (s_1, t_1) \\
(*, t_0) & \xrightarrow{c} & (*, t_0) \\
(*, t_0) & \xrightarrow{c} & (*, t_1) \\
(*, t_1) & \xrightarrow{c} & (*, t_1)
\end{array}
$$



Hình 2.15: Một ô-tô-mát bất đồng bộ

Định nghĩa 2.18 (Ô-tô-mát bất đồng bộ) *Một ô-tô-mát bất đồng bộ trên* $\widetilde{\Sigma}$ *là một cấu trúc* $\mathcal{A} = \langle S, \{\rightarrow_a\}_{a\in\Sigma}, S_{in}, F, G\rangle$.

Để định nghĩa hành vi của $\mathcal{A}$ trước tiên, chúng ta đi định nghĩa một quan hệ chuyển toàn cục $\rightarrow_{\mathcal{A}} \subseteq Sproc \times \Sigma \times Sproc$ của $\mathcal{A}$ như sau: $(s, a, s') \in \rightarrow_{\mathcal{A}}$ khi và chỉ khi tồn tại một dịch chuyển $s_a \rightarrow_a s'_a$ sao cho $s(i) = s_a(i)$ và $s'(i) = s'_a(i)$ đối với mọi $i \in loc(a)$, và $s(j) = s'(j)$ với $j \notin loc(a)$. Khi $(s, a, s') \in \rightarrow_{\mathcal{A}}$ chúng ta viết $s \xrightarrow{a} s'$.

Ví dụ 2.7 *Một ô-tô-mát bất đồng bộ trên ({a,b}, {b,c}) được biểu diễn trong Hình 2.15 với các dịch chuyển đồng bộ b-synchronizing được mô tả bằng nét đứt. Bảng chuyển được mô tả trong Bảng 2.2*

Một thực thi (run) trên từ ω thuộc Σ^ω là một ánh xạ ρ: $pref(\omega) \rightarrow S_{Proc}$ được định nghĩa như sau:

- $\rho(\epsilon) \in S_{in}$, và
- đối với mỗi tiền tố τa của ω, $\rho(\tau) \xrightarrow{a}_{\mathcal{A}} \rho(\tau a)$.

Một thực thi ρ được gọi là thực thi chấp nhận (accepting run) khi và chỉ khi với mỗi $j \in Proc$ hoặc:

- $Proj_j(\omega)$ là hữu hạn, và $\rho(\omega')(j) \in F_j$ với $\omega' \in pref(\omega)$ và $Proj_j(\omega') = Proj_j(\omega)$, hoặc
- $Proj_j(\omega)$ là vô hạn và $\rho(\tau)(j) \in G_j$, với $\tau \in pref(\omega)$ được lặp lại vô hạn lần.

Khi ρ là một thực thi chấp nhận trên ω, chúng ta nói rằng ω được đoán nhận bởi $\mathcal{A}$. Tập tất cả các từ được đoán nhận bởi $\mathcal{A}$ hình thành ngôn ngữ được đoán nhận bởi $\mathcal{A}$ và được ký hiệu là $L_{sym}(\mathcal{A})$. Kết quả sau có được dựa trên kết quả của Gastin và Petit [71] là một sự tổng quát hóa cho từ vô hạn từ kết quả của Zielonka trong [75].

Định lý 2.3 *Cho $L \subseteq \Sigma^\omega$, khi đó L là một ngôn ngữ ω-chính quy nhất quán trên $\widetilde{\Sigma}$ khi và chỉ khi L được đoán nhận bởi một ô-tô-mát bất đồng bộ trên $\widetilde{\Sigma}$.*

Do đó, nếu một từ $\omega \in \Sigma^\omega$ được đoán nhận bởi $\mathcal{A}$ thì bất kỳ $\omega ttow(wtot(\omega))$ cũng được đoán nhận bởi $\mathcal{A}$. Thực tế, $\mathcal{A}$ đoán nhận Vết $wtot(\omega)$. Chúng ta định nghĩa ngôn ngữ Vết được đoán nhận bởi $\mathcal{A}$ là $TrL(\mathcal{A}) \hat{=} wtot(L_{sym}(\mathcal{A}))$.

2.3.4 Logic trên Vết

Phần này sẽ giới thiệu về lôgic thời gian tuyến tính (LTL) trên Vết. Các khái niệm và kết quả được tham khảo trong các tài liệu [14, 58, 31].

Cú pháp

Tập các công thức của LTL trên một bảng chữ cái độc lập (Σ, I) được ký hiệu là $LTL_t(\Sigma, I)$ và được cho theo cú pháp sau:

$$\varphi ::= tt | \langle a \rangle \varphi | \varphi U \varphi | \neg \varphi | \varphi \vee \varphi, a \in \Sigma.$$

Chúng ta thường sử dụng ký hiệu như $\eta, \varphi, \psi, ...$ cho biểu diễn công thức. Khi ngữ cảnh là rõ ràng, chúng ta có thể viết $LTL_t(\Sigma, I)$ bằng $LTL(\Sigma, I)$, tức là bỏ ký hiệu _t (hiểu là Vết) đi. Mặc dù cú pháp của công thức LTL_t là tương tự như LTL được biểu diễn ngữ nghĩa trên từ (word) (LTL_w) nhưng về ý nghĩa thì có nhiều khác nhau. Công thức của $LTL(\Sigma, I)$ được thể hiện trên các cấu hình của các Vết trên (Σ, I). Ta có định nghĩa chi tiết về ngữ nghĩa trong phần dưới đây.

Ngữ nghĩa

Định nghĩa 2.19 *Cho Vết $T \in Tr(\Sigma, I)$, một cấu hình $Ç \in conf(T)$, và một công thức $\phi \in LTL(\Sigma, I)$, ngữ nghĩa của khái niệm $T, Ç \models \phi$ được định nghĩa một cách đệ quy như sau:*

- $T, Ç \models tt$,
- $T, Ç \models \neg\psi$ *nếu và chỉ nếu* $T, Ç \not\models \psi$,
- $T, Ç \models \psi \vee \varphi$ *nếu và chỉ nếu* $T, Ç \models \psi$ *hoặc* $T, Ç \models \varphi$,
- $T, Ç \models \langle a\rangle\psi$ *nếu và chỉ nếu tồn tại một cấu hình* $Ç' \in conf(T)$ *sao cho* $Ç \xrightarrow{a}_T Ç'$ *và* $T, Ç' \models \psi$, *và*
- $T, Ç \models \psi U \varphi$ *nếu và chỉ nếu tồn tại một cấu hình C' ∈ conf(T) với* $Ç \subseteq Ç'$ *sao cho* $T, Ç' \models \varphi$ *và với mọi Ç" ∈ conf(T) với* $Ç \subseteq Ç" \subseteq Ç'$, *chúng ta có* $T, Ç \models \psi$.

Nếu $T, \emptyset \models \psi$ ta viết bằng $T \models \psi$ và ta nói T mô hình ψ , T là một mô hình cho ψ hoặc T thỏa ψ. Hơn thế, ta nói rằng ψ có khả năng thỏa mãn nếu tồn tại một vêt $T \in Tr(\Sigma, I)$ sao cho $T \models \psi$. Tất cả các mô hình của một công thức $\psi \in LTL_t(\Sigma, I)$ tạo thành một tập con của $Tr(\Sigma, I)$ do đó có thể gọi là một ngôn ngữ và ký hiệu là $L(\psi)$.

Ý nghĩa của các phép toán lôgic trong này là như lôgic thông thường. Chỉ có một chút khác biệt trong phép kết hợp $\langle\_\rangle$ và U. Chúng ta sẽ tìm hiểu nó thông qua ví dụ sau. Một công thức đơn giản $\langle\_\rangle$ của LTL_t là $\varphi = \langle a\rangle\langle b\rangle\langle c\rangle tt$. Cho một Vết T, nó được coi là thỏa mãn trong cấu hình rỗng nếu và chỉ nết có một cấu hình a-successor [51] thỏa $\langle b\rangle\langle c\rangle tt$. Bằng trực giác, một phép kết hợp nhị phân $\langle\_\rangle$ mô tả một khởi tạo phân đoạn của cấu trúc của đồ thị cấu hình của một Vết T.

Định nghĩa 2.20 (Công thức tương đương) *Hai công thức $\psi, \psi' \in LTL(\Sigma, I)$ được gọi là tương đương khi và chỉ khi với mọi Vết $T \in Tr(\Sigma, I)$ và mọi cấu hình $Ç \in conf(T)$, ta có:*
$T, Ç \models \psi$ khi và chỉ khi $T, Ç \models \psi'$.

Tiếp theo chúng ta xét thêm về công thức trong LTL_t. Mặc dù quan hệ độc lập của một bảng chữ cái độc lập cho trước không ảnh hưởng gì trong cú pháp của công thức LTL_t nhưng nó vẫn có ý nghĩa về mặt ngữ nghĩa. Ví dụ hai công thức $\langle a\rangle\langle b\rangle\varphi$ và $\langle b\rangle\langle a\rangle\varphi$ là tương đương nếu như a và b là độc lập nhau. Trong trường hợp hoàn toàn độc lập, cấu hình rỗng của một Vết có thể được hiểu như

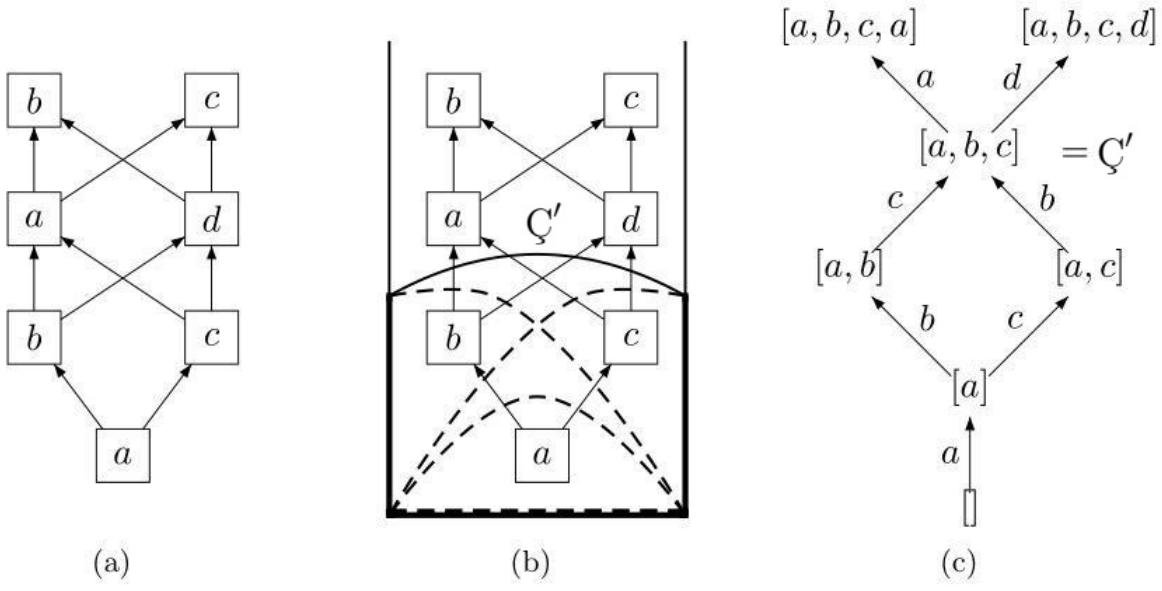
là tiền tố rỗng của một từ. Hơn thế, mọi cấu hình có một cấu hình tiếp dẫn trực tiếp duy nhất.

Bây giờ chúng ta sẽ tìm hiểu chi tiết công thức có toán tử *Until*, tức là $\varphi U \psi$. Ta biết rằng $T, C \models \psi U \varphi$ nếu và chỉ nếu tồn tại một cấu hình $C' \in conf(T)$ với Ç $\subseteq$ Ç′ sao cho T, Ç′ $\models \varphi$ và với mọi Ç" $\in conf(T)$ với Ç $\subseteq$ Ç" $\subseteq$ Ç′, chúng ta có T, Ç $\models \psi$.

Chúng ta xem xet Vết được chỉ ra trong Hình 2.16(a) và công thức $\psi = \langle a \rangle tt \wedge \langle d \rangle tt$. Rõ ràng là cấu hình Ç′ trong 2.16 (b) thỏa mã ψ và Ç′ là duy nhất. Vì thế đối với một công thức φ bất kỳ, công thức $\varphi U \psi$ có được đối với một Vết cho trước khi và chỉ khi tất cả các cấu hình nét đứt thỏa φ. Tình huống này cũng được thể hiện trong Hình 2.16(c).

Một số kết quả

Phần này sẽ chỉ ra một số kết quả tổng hợp được trong các tài liệu liên quan mà không cần chứng minh. Một số kết quả dưới đây được lấy từ tài liệu [14], trong đó các tác giả chứng minh rằng về mặt biểu diễn, lôgic thời gian tuyến tính trên Vết là bằng và đầy đủ với lôgic vị từ cấp một trên Vết. Đây là kết quả quan trọng vì nó chứng minh được tính đúng đắn, đầy đủ của lôgic trên Vết.



Hình 2.16: Ý nghĩa của Until

Định lý 2.4 *Bài toán về tính thỏa được của LTL_t là quyết định được.*

Định lý 2.5 *Biểu diễn của LTL_t là đầy đủ tương ứng với biểu diễn của FO trên Vết.*

Hệ quả 2.3 *Với mọi công thức $\varphi \in LTL_t$, tồn tại một công thức $\psi \in FO_t$ sao cho $L(\varphi) = L(\psi)$.*

Hệ quả 2.4 *Với mọi công thức $\varphi \in FO_t$, tồn tại một công thức $\psi \in LTL_t$ sao cho $L(\varphi) = L(\psi)$.*

Trong [62], tác giả đã chỉ ra được (xây dựng được) một ô-tô-mát đoán nhận ngôn ngữ của công thức lôgic LTL trên Vết. Các kết quả dưới đây sẽ cho ta mối quan hệ giữa một công thức lôgic LTL trên Vết và ô-tô-mát trên Vết.

Định lý 2.6 *Cho φ là một công thức của $LTL_t(\Sigma, I)$ và ký hiệu ô-tô-mát Alternating trên bảng chữ cái Σ là $\mathcal{A}_\varphi$. Khi đó $\omega \in L(\mathcal{A}_\varphi)$ khi và chỉ khi $T_\omega, \emptyset \models \varphi$ đối với mọi $\omega \in \Sigma^\omega$.*

Đối với ô-tô-mát bất đồng bộ, ta biết rằng ngôn ngữ Vết được đoán nhận bởi một ô-tô-mát bất đồng bộ (đã có thuật toán xây dựng) và ngôn ngữ của một công thức $\varphi \in LTL_t$ là một ngôn ngữ Vết. Do đó ta có kết quả sau [75].

Định lý 2.7 *Cho một công thức LTL_t φ và một ô-tô-mát bất đồng bộ $\mathcal{A}$, tồn tại một thuật toán quyết định xem $T \models \varphi$ với mọi $T \in TrL(\mathcal{A})$.*

Định lý trên nói rằng, cho trước một công thức lôgic LTL trên Vết, việc kiểm tra ngôn ngữ Vết của công thức có là tập con của ngôn ngữ của ô-tô-mát bất đồng bộ hay không là quyết định được. Ý nghĩa cơ bản của định lý này là nếu mô hình của một hệ thống là một ô-tô-mát bất đồng bộ $\mathcal{A}$ và một thuộc tính của hệ thống là một công thức lôgic trên Vết, sẽ tồn tại một cách thức kiểm tra xem thuộc tính này của hệ thống có thỏa mãn bởi mô hình của nó hay không. Điều này hỗ trợ tích cực cho việc xây dựng một phương pháp và công cụ đặc tả hệ thống (một model checker) mà hỗ trợ kiểm chứng một cách hoàn toàn tự động.

Phần này giới thiệu một công cụ dùng để đặc tả các hệ thống tương tranh là Vết Mazurkiewicz và lôgic trên Vết. Đây là một công cụ được đánh giá cao và có nhiều nghiên cứu bởi khả năng biểu diễn ngắn gọn và hỗ trợ kiểm chứng tự động thông qua một biểu diễn hữu hạn là các ô-tô-mát đoán nhận ngôn ngữ Vết. Các vấn đề trong đề tài được tổng hợp từ nhiều nguồn tài liệu khác nhau có độ tin cậy cao. Các kết quả được trình bày cơ bản và rõ ràng. Tuy Vết là công cụ rất tốt cho đặc tả các hệ thống tương tranh nhưng nó vẫn chưa hỗ trợ đặc tả hệ thống có yếu tố thời gian, là các hệ thống có ràng buộc về có yếu tố thời

gian thi của các hành động. Do đó, việc mở rộng lý thuyết Vết cho vấn đề ràng buộc thời gian là có ý nghĩa và cần được quan tâm nghiên cứu và phát triển.

2.4 Kết luận

Công nghệ phần mềm dựa trên thành phần là một cách tiếp cận mới trong phát triển phần mềm hiện nay. Với cách tiếp cận này, một hệ thống sẽ được xây dựng dựa trên việc ghép nối các thành phần với nhau. Mỗi thành phần riêng lẻ là một gói phần mềm, một dịch vụ Web hoặc một mô-đun được đóng gói và chúng giao tiếp với nhau thông qua các giao diện. Như vậy, các tiêu chí về sản xuất phần mềm nhanh, giá thành hạ, linh hoạt, chất lượng cao, dễ mở rộng là dễ dàng nhận thấy trong cách tiếp cận này. Tuy nhiên, phương pháp này vẫn còn một số vấn đề đặt ra. Thứ nhất là làm sao đảm bảo được các thành phần hoạt động được khi ghép nối với nhau? Thứ hai là làm thế nào để có thể mở rộng các thành phần từ thành phần có sẵn và quan trọng nhất là làm thế nào để đảm bảo chất lượng hệ thống tức là hệ thống thỏa mãn các ràng buộc đặc tả?

Qua nghiên cứu một số phương pháp thông dụng trong các phương pháp hình thức kiểm chứng phần mềm, luận án thấy rằng các phương pháp này là rất tốt, có hiệu quả trong việc đặc tả một lớp rộng các bài toán. Ngoài ra, các phương pháp này cũng phát triển đến việc tạo ra các công cụ phần mềm hỗ trợ. Tuy nhiên, với các hệ thống đòi hỏi đặc tả cả ràng buộc thời gian và tính tương tranh thì các phương pháp hiện thời chưa đề cập đến hoặc còn nhiều hạn chế. Do vậy, nghiên cứu phương pháp có khả năng đặc tả và kiểm chứng các giao diện thành phần hệ thống có tính tương tranh và ràng buộc thời gian là công việc nghiên cứu có nhiều ý nghĩa thực tế. Các kết quả nghiên cứu của luận án sẽ giải quyết phần nào bài toán này và được trình bày trong các chương tiếp theo.

Chương 3

Lý thuyết Vết thời gian

Như đã trình trong chương 1, lý thuyết Vết Mazurkiewicz là một công cụ hiệu quả trong việc mô hình và đặc tả các tiến trình song song (không tuần tự) cho các hệ tương tranh. Hạn chế lớn nhất của lý thuyết Vết là không hỗ trợ đặc tả các yêu cầu ràng buộc thời gian. Do vậy, trong chương này, luận án giới thiệu nghiên cứu đề xuất mở rộng về thời gian của lý thuyết Vết. Chúng tôi gọi lý thuyết mở rộng này là lý thuyết Vết thời gian. Sử dụng Vết thời gian, chúng ta có thể đặc tả không những tính tương tranh mà còn các ràng buộc thời gian trên các hành động của hệ thống. Như vậy, với lý thuyết này, chúng ta có thể đặc tả và kiểm chứng một cách hiệu quả các hệ thống tương tranh và có thêm ràng buộc thời gian trên cơ sở của các kết quả của lý thuyết Vết.

Nội dung chính của chương bao gồm việc đưa ra các khái niệm về Vết thời gian, Vết khoảng, ngôn ngữ Vết thời gian và một số kết quả liên quan. Đặc biệt, luận án đưa vào ô-tô-mát khoảng bất đồng bộ như là một công cụ đoán nhận lớp các ngôn ngữ Vết thời gian. Mặt khác, khi sử dụng Vết thời gian vào đặc tả các hệ tương tranh có yếu tố thời gian, chúng ta cần một kỹ thuật đặc tả các thuộc tính lôgic của hệ thống và kiểm tra xem liệu một thuộc tính có được thỏa bởi đặc tả của hệ thống hay không. Các kỹ thuật thông dụng là sử dụng lôgic để đặc tả như LTL hay CTL (Computational Tree Logic). Với hệ thống có yếu tố thời gian, chúng ta cần một lôgic thời gian. Việc mở rộng lý thuyết lôgic này để có thể đặc tả tính chất Vết thời gian[1] cũng được giới thiệu trong nghiên cứu này.

[1]Ngữ nghĩa của lôgic sẽ được thể hiện trực tiếp trên Vết thời gian

3.1 Giới thiệu

Lý thuyết Vết đã được sử dụng như là một công cụ đắc lực cho việc mô tả các hệ thống tương tranh [32, 68, 58] hay các hệ song song, phân tán [60, 61, 66]. Trong các nghiên cứu này, hành vi của hệ thống được đặc tả bởi các Vết, các thuộc tính logic được biểu diễn qua các công thức LTL mô tả tính chất Vết. Bài toán kiểm chứng hệ thống cũng được giải quyết qua các phương pháp của lý thuyết ô-tô-mát [75]. Khi các hệ thống có thêm các ràng buộc thời gian thì việc mở rộng lý thuyết Vết để có thể mô hình hệ thống này cũng được quan tâm nghiên cứu và đã có những đóng góp nhất định. Các kết quả trong [23, 73, 12] đã khẳng định công việc nghiên cứu và phát triển lý thuyết Vết thời gian là có ý nghĩa. Tuy nhiên, các nghiên cứu này tập trung vào giải quyết bài toán kiểm chứng mô hình trong khi mục tiêu của luận án là đề xuất phương pháp đặc tả các giao diện của các hệ thống tương tranh thời gian.

Do đó trong chương này, luận án giới thiệu một nghiên cứu đề xuất phương pháp đặc tả hệ thống có ràng buộc về tính tương tranh và thời gian. Ý tưởng cơ bản của phương pháp là mở rộng về thời gian trên lý thuyết Vết để có thể dễ dàng đặc tả tính tương tran và ràng buộc thời gian. Trong nghiên cứu này, ràng buộc cho một hành động của hệ thống được mô tả như sau. Một hành động a trong hệ thống không những có đặc tả về chức năng mà còn đặc tả phi chức năng bao gồm đặc tả về có yếu tố thời gian thi yếu nhất (WCET - Worst-case Execution Time). Trong đó, đặc tả WCET có dạng $l_a \leq dur \leq u_a$, với l_a, u_a là số thực không âm, $l_a \leq u_a$, và dur là khoảng (ràng buộc thời gian thực thi) của phương thức được đặc tả. Như vậy với mỗi hành động, ta sẽ biểu diễn WCET bằng một hàm gán thời gian tương ứng $J(a) = dur$.

Khi đưa thời gian vào, một giao thức thể hiện trong thành phần sẽ phải thỏa mãn ba ràng buộc sau:

1. *Ràng buộc thứ tự*: các hành động tương tác nên tuân theo một số ràng buộc trên thứ tự chúng xảy ra,

2. *Ràng buộc thời gian:* có nhiều kiểu ràng buộc trên thời gian. Trong một thành phần, ràng buộc tới hạn quan trọng nhất là các phương thức không thể được gọi quá liên tục nếu chúng không được thực hiện song song. Điều này có nghĩa là có một số thời gian tối thiểu giữa các hành động mà chúng hoạt động trong một chuỗi, và

3. *Ràng buộc trên các lời gọi song song từ các luồng khác nhau*: Các dịch vụ nào có thể được gọi song song với dịch vụ nào.

Ba điều kiện này là ba điều kiện bình thường có thể quan sát và dễ dàng nhận thấy. Đây là các yêu cầu được đặc tả và cần tuân thủ trong đặc tả thể thức của giao diện, đảm bảo chất lượng dịch vụ của hệ thống: Thứ nhất là ràng buộc thứ tự, các hành động tương tác nên tuân theo một số ràng buộc trên thứ tự chúng xảy ra, tức là thứ tự các lời gọi cần được tuân thủ, chẳng hạn như hành động mở tệp tin cần được thực hiện trước hành động đọc và ghi (điều kiện 1). Vì có chất lượng dịch vụ nên các hành động không nên gọi quá liên tục (chúng sẽ không đáp ứng kịp). Do đó, giữa các lời gọi hành động nên có một khoảng thời gian để cho hành động đó khởi động, thực hiện và cung cấp dịch vụ. Yêu cầu này dẫn tới một ràng buộc thời gian cho mỗi hành động về thời gian tối thiểu và tối đa cần thiết để hành động có thể cung cấp dịch vụ (trong trường hợp không cần ràng buộc thì thời gian này là $[0,\infty]$). Chẳng hạn hành động mở tệp tin để ghi thì phải được gọi sau hành động đóng tệp, nếu gọi liên tục mở tệp thì hệ thống có thể không đáp ứng được do hệ thống còn phải thực hiện hành động đóng trước. Do đó hành động mở tệp tin cần có một khoảng thời gian để nó thực hiện dịch vụ của mình (điều kiện 2). Với điều kiện thứ 3: Ràng buộc trên các lời gọi song song từ các luồng khác nhau. Điều kiện này thể hiện rằng một số hành động độc lập nhau có thể thực hiện song song mà không cần phải chờ đợi nhau, nhằm tăng tốc độ thực hiện hành động. Chẳng hạn hai hành động mở tệp tin và hành động đóng một cửa sổ đang hiển thị. Các kết quả trong các công trình nghiên cứu trong danh mục công trình khoa học [2] đã chỉ ra rằng Vết thời gian là phù hợp cho đặc tả hệ thống tương tranh có ràng buộc thời gian.

3.2 Vết thời gian và ô-tô-mát khoảng bất đồng bộ

Vết thời gian và ô-tô-mát khoảng bất đồng bộ (được viết tắt là ADA - Asynchronous Duration Automata) đã được nghiên cứu đề xuất trong các công trình khoa học được công bố trong công trình số 1 và số 2 thuộc danh mục công trình khoa học của tác giả liên quan tới luận án. Các nghiên cứu đó đã chỉ ra các lợi ích quan trọng của Vết thời gian trong việc hỗ trợ đặc tả các hệ thống tương tranh có yếu tố thời gian. Các lợi ích đó bao gồm tính đơn giản, ngắn gọn, có biểu diễn hữu hạn bằng ô-tô-mát và có thể được định nghĩa bằng lôgic thời gian tuyến tính. Trong phần này, luận án giới thiệu lại một số khái niệm và kết quả quan trọng được áp dụng vào nghiên cứu này.

[2] Danh mục được liệt kê trong phần cuối luận án, trước tài liệu tham khảo

3.2.1 Vết thời gian

Cho Σ là tập hữu hạn các ký hiệu[3], $D \subseteq \Sigma \times \Sigma$ là quan hệ phụ thuộc giữa các ký hiệu trên bảng chữ cái Σ. Ta biết rằng, một Vết là một lớp tương đương của một thứ tự bộ phận được gán nhãn $T = \langle V, \leq, \lambda \rangle$ trên bảng chữ cái phụ thuộc (Σ, D) với V là tập các nút, $\leq$ là một quan hệ thứ tự bộ phận trên V, và $\lambda : V \longrightarrow \Sigma$ là một hàm gán nhãn. Bây giờ ta sẽ đưa ra các khái niệm về Vết thời gian như là một sự mở rộng về thời gian của Vết. Gọi thời gian là liên tục và được biểu diễn như là tập các số thực không âm $\mathbb{R}^{\geq 0}$. Kí hiệu $\leq$ cũng biểu diễn thứ tự tự nhiên trong $\mathbb{R}^{\geq 0}$. Trường hợp các từ thời gian, chúng ta thêm một hàm gán nhãn θ cho mỗi đỉnh của Vết một điểm (giá trị) thời gian trong $\mathbb{R}^{\geq 0}$.

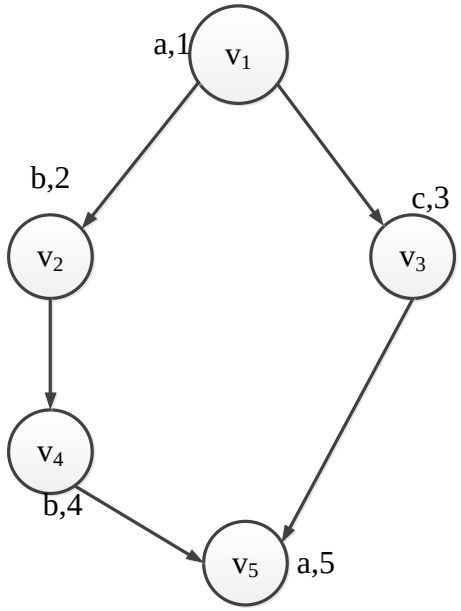
Định nghĩa 3.1 (Vết thời gian) *Một Vết thời gian trên bảng chữ cái phụ thuộc (Σ, D) là một bộ (T, θ) với $T = \langle V, \leq, \lambda \rangle$ là một Vết trên bảng chữ cái phụ thuộc (Σ, D), và $\theta : V \longrightarrow \mathbb{R}^{\geq 0}$ là hàm gán nhãn thời gian gán mỗi đỉnh $v \in V$ một giá trị là một số thực không âm biểu diễn thời điểm mà hành động tương ứng nút đó thực hiện. Hàm θ thỏa mãn các điều kiện sau:*

- *Nếu $v < v'$ thì $\theta(v) < \theta(v')$ (thời gian có tính trước sau), và*
- *Nếu T là vô hạn, với bất kỳ $t > 0$ có một nhát cắt C của T sao cho $\min\{\theta(v) \mid v \in C\} > t$ (thời gian luôn tiến triển).*

Ví dụ 3.1 *Cho $D = \{a, b\} \times \{a, b\} \cup \{a, c\} \times \{a, c\}$ là quan hệ phụ thuộc trên bảng chữ cái $\Sigma = \{a, b, c\}$,$V = \{v_1, v_2, v_3, v_4, v_5\}$, quan hệ thứ tự $\leq$ được cho như sau: $v_1 \leq v_2, v_1 \leq v_3, v_2 \leq v_4, v_3 \leq v_5, v_4 \leq v_5$, hàm gán đỉnh*
$\lambda(v_1) = a, \lambda(v_2) = b, \lambda(v_3) = c, \lambda(v_4) = b, \lambda(v_5) = a$, Vết $T = \langle V, \leq, \lambda \rangle$, hàm gán giá trị thời gian θ được cho như sau:
$\theta(v_1) = 1, \theta(v_2) = 2, \theta(v_3) = 3, \theta(v_4) = 4, \theta(v_5) = 5$. Sơ đồ thứ tự bộ phận cho Vết thời gian (T, θ) được chỉ ra trong Hình 3.1.

Một tập các Vết thời gian trên bảng chữ cái phụ thuộc (Σ, D) được gọi là ngôn ngữ Vết thời gian trên (Σ, D). Nếu V là hữu hạn, Vết T được gọi là Vết thời gian hữu hạn. Cho (T, θ) một Vết thời gian, tập các từ thời thỏa Vết thời gian là tập các tuyến tính hóa của T kết hợp với hàm gán thời gian θ. Như vậy, nếu một Vết mà quan hệ độc lập là rỗng thì Vết thời gian chính là từ thời gian.

[3]Thường là các ký hiệu hành động của hệ thống



Hình 3.1: Sơ đồ thứ tự bộ phận Vết thời gian được cho trong ví dụ 3.1

Đặt *intv* là tập tất cả các khoảng thời gian trên $\mathbb{R}^{\geq 0}, intv = \{[l,u] \mid l \in \mathbb{R}^{\geq 0} \wedge u \in \mathbb{R}^{\geq 0} \cup \{\infty\}\}$. Như đã trình bày trong phần giới thiệu, mỗi hành động của hệ thống có ràng buộc về thời gian thực thi WCET. Để đặc tả ràng buộc này, luận án sử dụng một hàm gán khoảng thời gian $J : \Sigma \longrightarrow intv$ là hàm gán một khoảng thời gian cho mỗi hành động $a \in \Sigma$. Hàm J kết hợp với bảng chữ cái phụ thuộc cho ta một khái niệm về bảng chữ cái mới được gọi là bảng chữ cái phụ thuộc khoảng và được định nghĩa là một bộ ba thành phần (Σ, D, J).

Định nghĩa 3.2 (Vết khoảng) *Gọi $T = \langle V, \leq, \lambda \rangle$ là một Vết trên (Σ, D), J là hàm gán ràng buộc khoảng thời gian được xác định như trên, cặp (T, J) được gọi là một Vết khoảng trên (Σ, D).*

Ký hiệu $\lessdot_v = \{u | u \lessdot v\}$ là tập các nút thực hiện ngay trước nút v, vết khoảng (T, J) biểu diễn một lớp các Vết thời gian theo hai cách thường được sử dụng trong các ứng dụng như sau.

i. Một Vết thời gian (T, θ) được gọi là thỏa mãn Vết khoảng (T, J) theo điều kiện sau khi và chỉ khi $\forall v \in V, \forall v' \in \lessdot_v, \lambda(v') \in \Sigma \Rightarrow \theta(v) - \theta(v') \in J(\lambda(v))$. Ta ký hiệu $Postw(T, J) = \{(T, \theta) | \forall v \in V, \forall v' \in \lessdot_v, \lambda(v') \in \Sigma \Rightarrow \theta(v) - \theta(v') \in J(\lambda(v))\}$ là tập các Vết thời gian thỏa Vết khoảng theo điều kiện sau.

ii. Một Vết thời gian (T, θ) được gọi là thỏa mãn Vết thời khoảng (T, J) theo điều kiện trước khi và chỉ khi $\forall v \in V, \forall v' \in \lessdot_v, \lambda(v') \in \Sigma \Rightarrow \theta(v) - \theta(v') \in J(\lambda(v'))$.

Ta ký hiệu $Pretw(T,J) = \{(T,\theta) | \forall v \in V, \forall v' \in \lessdot_v, \lambda(v') \in \Sigma \Rightarrow \theta(v) - \theta(v') \in J(\lambda(v'))\}$ là tập các Vết thời gian thỏa Vết khoảng theo điều kiện trước.

Với cách xác định như trên, thỏa theo điều kiện sau thể hiện rằng các hành động trước hành động hiện tại phải cung cấp dịch vụ để thỏa mãn yêu cầu ràng buộc của hành động hiện tại (chẳng hạn các hệ cung cấp dịch vụ sẽ phải thỏa mãn yêu cầu của khách hàng thuê dịch vụ đó). Thỏa theo điều kiện trước thể hiện hành động hiện tại (sau) muốn thực hiện được mà yêu cầu dịch vụ từ hành động trước nó thì phải thỏa ràng buộc của hành động trước (trong trường hợp cung cấp dịch vụ, khách hàng thuê dịch vụ phải đồng ý với điều khoản ràng buộc mà dịch vụ đó cung cấp). Gọi $dtot(T,J)$ là tập các Vết thời gian (T,θ) thỏa Vết khoảng (T,J), $dtot(T,J) \widehat{=} \{(T,\theta) \mid (T,\theta)$ thỏa mãn (T,J) theo điều kiện sau hoặc điều kiện trước$\}$[4]. Trong luận án này, nếu không nói gì thêm và không mất tính tổng quát về phương pháp, luận án sử dụng quan niệm thỏa theo điều kiện sau. Các ứng dụng khác nhau có thể điều chỉnh việc chọn lựa điều kiện thỏa cho phù hợp.

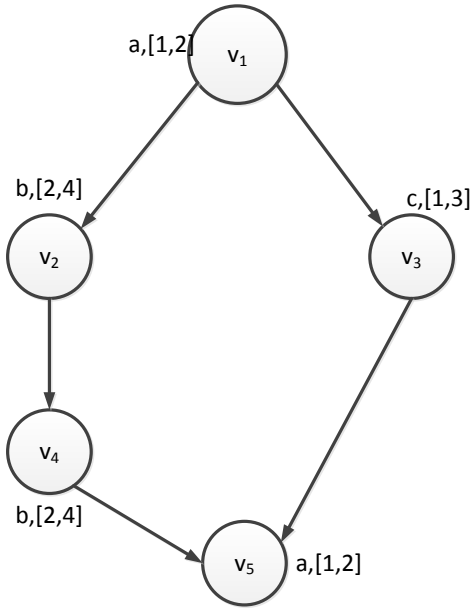
Cho trước một bảng chữ cái phụ thuộc khoảng (Σ, D, J) và một ngôn ngữ Vết L trên (Σ, D), khi đó một ngôn ngữ Vết khoảng trên bảng chữ cái phụ thuộc khoảng (Σ, D, J), kí hiệu là $dTrL(\Sigma, D, J)$, được định nghĩa là tập các Vết khoảng trên bảng chữ cái (Σ, D, J), tức là $dTrL(\Sigma, D, J) \widehat{=} \bigcup_{T \in L}(T, J)$. Ngôn ngữ Vết thời gian trên bảng chữ cái phụ thuộc khoảng (Σ, D, J), kí hiệu là $tTrL(L, J)$, là tập các Vết thời gian thỏa mãn các Vết khoảng của ngôn ngữ Vết khoảng. Cho L là một ngôn ngữ Vết và J là hàm gán thời gian được định nghĩa như trên, khi đó ta kí hiệu cặp (L, J) là ngôn ngữ Vết khoảng trên ngôn ngữ Vết L, $(L, J) \widehat{=} \bigcup_{T \in L}(T, J)$. Ta định nghĩa ngôn ngữ Vết thời gian của ngôn ngữ Vết khoảng là tập tất cả các Vết thời gian thỏa các Vết khoảng của ngôn ngữ Vết khoảng.

Định nghĩa 3.3 *Ngôn ngữ Vết thời gian của ngôn ngữ Vết khoảng (L, J) được ký hiệu là $tTrL(L, J)$, $tTrL(L, J) \widehat{=} \bigcup_{T \in L} dtot(T, J)$.*

Ví dụ 3.2 đưa ra một minh họa cho Vết khoảng.

Ví dụ 3.2 *Cho $T = \langle V, \leq, \lambda \rangle$ là một Vết với $D = \{a, b\}^2 \cup \{a, c\}^2$ là quan hệ phụ thuộc trên bảng chữ cái $\Sigma = \{a, b, c\}$,$V = \{v_1, v_2, v_3, v_4, v_5\}$, quan hệ thứ tự $\leq$ được cho như sau: $v_1 \leq v_2, v_1 \leq v_3, v_3 \leq v_5, v_2 \leq v_5$, hàm gán đỉnh $\lambda(v_1) = a, \lambda(v_2) = b, \lambda(v_3) = c, \lambda(v_4) = b, \lambda(v_5) = a$, hàm gán giá trị thời khoảng J được cho như sau: $J(a) = [1, 2], J(b) = [2, 4], J(c) = [1, 3]$. Sơ đồ thứ tự bộ phận minh họa cho Vết khoảng (T, J) được chỉ ra trong Hình 3.2.*

[4]Tùy từng bài toán ta sẽ có cách lựa chọn phép thỏa một cách thích hợp



Hình 3.2: Sơ đồ thứ tự bộ phận của một Vết khoảng được cho trong ví dụ 3.2

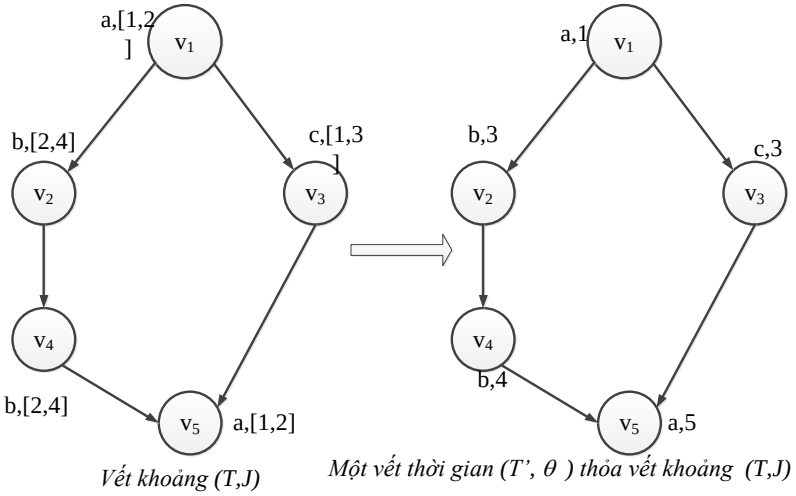
Một Vết thời gian (T', θ) thỏa Vết khoảng (T, J) như trong ví dụ 3.2 được chỉ ra trong Hình 3.3.

Hiển nhiên rằng, một Vết thời gian có thể nắm giữ được tất cả ba đặc trưng của các giao thức tương tranh có yếu tố thời gian được đề cập trong phần giới thiệu, và hơn thế nữa, bảng chữ cái khoảng có thể biểu diễn ràng buộc trên sự thực thi của các hành động của hệ thống. Như vậy, ngôn ngữ Vết khoảng có thể đặc tả đầy đủ ràng buộc thời gian và tương tranh của hệ thống. Hành vi của hệ thống sẽ là ngôn ngữ Vết thời gian thỏa ngôn ngữ Vết khoảng được dùng để đặc tả hệ thống.

3.2.2 Ô-tô-mát khoảng bất đồng bộ

Ta biết rằng ô-tô-mát bất đồng bộ đoán nhận ngôn ngữ Vết [75], do đó một cách tự nhiên, ta sẽ mở rộng ô-tô-mát này với thời gian để nó có thể đoán nhận các Vết thời gian của chúng ta. Gọi $Proc = \{1, 2, ..., n\}$ là tập các chỉ số các tiến trình. Gọi bảng chữ cái phụ thuộc là (Σ, D) sao cho $\Sigma = \bigcup_{j \in Proc} \Sigma_j$ và $D = \{(a, b) | \exists j. \{a, b\} \subset \Sigma_j\}$. Điều này có nghĩa là hai hành động trong Σ là độc lập nếu chúng thuộc hai tiến trình khác nhau trong hệ thống. Ta kí hiệu $\widetilde{\Sigma} \widehat{=} \{\Sigma_i\}_{i \in Proc}$ là bảng chữ cái phân tán.

Một ô-tô-mát bất đồng bộ trên $\widetilde{\Sigma}$ là một cấu trúc $\mathcal{A} = (S, \{\rightarrow_a\}_{a \in \Sigma}, S_{in}, F, G)$. Hành vi của ô-tô-mát bất đồng bộ được định nghĩa chi tiết trong chương 2. Tiếp



Hình 3.3: Sơ đồ thứ tự bộ phận của Vết khoảng (T, J) và và Vết thời gian (T', θ) thỏa (T, J)

theo, chúng ta sẽ định nghĩa về ô-tô-mát khoảng bất đồng bộ (được ký hiệu là ADA - Asynchronous Duration Automata). Một ô-tô-mát khoảng bất đồng bộ là một ô-tô-mát bất đồng bộ mà với mỗi hành động của nó được gán thêm ràng buộc khoảng thời gian thực thi (được thể hiện qua hàm gán thời gian J như đã trình bày bên trên). Như vậy, cho J được định nghĩa như trên, chúng ta định nghĩa một ô-tô-mát khoảng bất đồng bộ như là một ô-tô-mát bất đồng bộ được trang bị thêm một hàm gán ràng buộc thời gian J với $J(a) = (J_i(a))_{i \in loc(a)})$ với mỗi dịch chuyển a $(a - transition)$.

Định nghĩa 3.4 (Ô-tô-mát khoảng bất đồng bộ) *Một ô-tô-mát khoảng bất đồng bộ là một bộ $(\mathcal{A}, J)$ với $\mathcal{A}$ là một ô-tô-mát bất đồng bộ và J là hàm gán thời gian được xác định như trên.*

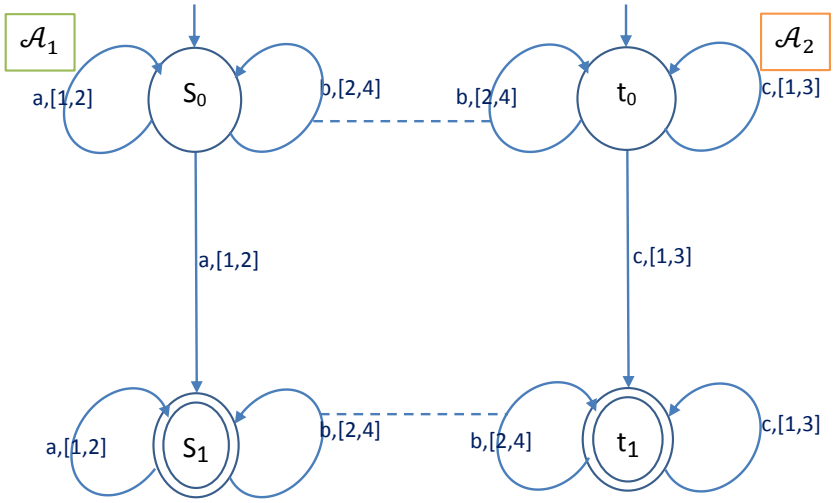
Hành vi của ADA được hiểu thông qua hành vi của AA. Ta biết rằng AA đoán nhận ngôn ngữ là các Vết, khi đó các Vết này kết hợp với hàm gán thời gian J cho ta ngôn ngữ Vết khoảng (L, J) với L là ngôn ngữ Vết được đoán nhận bởi AA. Từ đó ta có ngôn ngữ Vết thời gian thỏa ngôn ngữ Vết khoảng (L, J) này (được xác định như định nghĩa phía trước, định nghĩa 3.3) và ta có thể xác định được hành vi của ADA thông qua các từ thời gian mà nó đoán nhận là các từ thời gian thỏa Vết thời gian. Bây giờ chúng ta định nghĩa một từ thời gian được đoán nhận bởi $(\mathcal{A}, J)$ trực tiếp trên Vết khoảng.

Một thực thi trên từ thời gian $\omega \in (\Sigma \times \mathbb{R}^{\leq 0})^\omega$ là một ánh xạ $\rho : pref(\omega) \to S_{Proc}$ được định nghĩa một cách đệ quy như sau:

- $\rho(\epsilon) \in S_{in}$,
- với mỗi tiền tố τ(a,t) của $\omega, \rho(\tau) \xrightarrow{a}_{\mathcal{A}} \rho(\tau a)$ và $t - time_i(\tau) \in J_i(a)$ với mọi $i \in loc(a)$ ở đây đối với một từ thời gian $\tau = w'(b,t')w''$ sao cho $b \in \Sigma_i$ và w'' không có kí hiệu a trong Σ_i, chúng ta định nghĩa $time_i(\tau)\widehat{=}t'$.

Một ví dụ minh họa cho ADA được chỉ ra trong ví dụ 3.3.

Ví dụ 3.3 *Cho một Ô-tô-mát bất đồng bộ như trong Hình 2.15, ta thêm hàm gán thời gian J cho các hành động như sau: $J(a) = [1,2], J(b) = [2,4], J(c) = [1,3]$ để trở thành một ADA như trong Hình 3.4.*



Hình 3.4: Một ADA với hàm gán thời gian J trong ví dụ 3.3

Cho ω là một từ thời gian, ta ký hiệu $Proj_j(\omega)$ từ thời gian được sinh ta khi ta chiếu ω lên tiến trình j, tức từ thời gian có được bằng cách là giữ lại tất cả các ký hiệu xuất hiện trong ω và trên tiến trình j cùng thứ tự của nó.

Một thực thi ρ trên từ thời gian ω được gọi là thực thi chấp nhận (accepting run) khi và chỉ khi với mỗi $j \in Proc$ hoặc:

- $Proj_j(\omega)$ là hữu hạn, và $\rho(\omega')(j) \in F_j$ với $\omega' \in pref(\omega)$ và $Proj_j(\omega') = Proj_j(\omega)$, hoặc
- $Proj_j(\omega)$ là vô hạn và $\rho(\tau)(j) \in G_j$, với $\tau \in pref(\omega)$ được lặp lại vô hạn lần.

Khi ρ là một thực thi chấp nhận trên ω, chúng ta nói rằng ω được đoán nhận bởi $(\mathcal{A}, J)$. Tập tất cả các từ được đoán nhận bởi $(\mathcal{A}, J)$ hình thành ngôn ngữ

được đoán nhận bởi $(\mathcal{A}, J)$ và được ký hiệu là $L(\mathcal{A}, J)$. Ví dụ 3.4 minh họa cho thực thi của ADA trong ví dụ 3.3.

Ví dụ 3.4 *Cho ADA $(\mathcal{A}, J)$ như trong ví dụ 3.3 một thực thi trên từ thời gian $\omega = (a,1)(b,3)(b,4)(c,5)(a,6)$ là hàm ρ như sau: $\rho(\epsilon) = (S_0, t_0), \rho(\omega) = (S_1, t_0), \rho((b,3)(b,4)(c,5)(a,6)) = (S_1, t_0), \rho((b,4)(c,5)(a,6)) = (S_1, t_0), \rho((c,5)(a,6)) = (S_1, t_1), \rho((a,6)) = (S_1, t_1)$ Dãy dịch chuyển tương ứng là $(S_0, t_0) \xrightarrow{a}_{\mathcal{A}} (S_1, t_0) \xrightarrow{b}_{\mathcal{A}} (S_1, t_0) \xrightarrow{b}_{\mathcal{A}} (S_1, t_0) \xrightarrow{c}_{\mathcal{A}} (S_1, t_1) \xrightarrow{a}_{\mathcal{A}} (S_1, t_1)$.*

Ta biết rằng, nếu một từ $\omega \in \Sigma^\omega$ được đoán nhận bởi $\mathcal{A}$ thì bất kỳ ω, $ttow(wtot(\omega))$ cũng được đoán nhận bởi $\mathcal{A}$. Ngôn ngữ Vết được đoán nhận bởi $\mathcal{A}$ là $TrL(\mathcal{A}) \hat{=} wtot(L_{sym}(\mathcal{A}))$.

Chúng ta đưa ra định nghĩa về ngôn ngữ Vết thời gian được đoán nhận bởi ô-tô-mát khoảng bất đồng bộ như sau.

Định nghĩa 3.5 *Ngôn ngữ Vết thời gian được đoán nhận bởi một ô-tô-mát khoảng bất đồng bộ $(\mathcal{A}, J)$ được định nghĩa như sau: $tL(\mathcal{A}, J) \widehat{=} \bigcup_{T \in TrL(\mathcal{A})} dtot(T, J)$.*

Như vậy, các ADA đoán nhận các từ thời gian dựa trên các dịch chuyển trạng thái của ô-tô-mát bất đồng bộ tương ứng và thỏa các điều kiện ràng buộc về thời gian cho mỗi hành động. Nói cách khác, các từ thời gian được đoán nhận bởi ADA là các từ thời gian thỏa các Vết khoảng được tạo bởi các Vết từ ngôn ngữ do ô-tô-mát đồng bộ tương ứng đoán nhận kết hợp với hàm gán thời gian J. Từ định nghĩa trên và định nghĩa về phép toán chuyển từ vết thời gian thành từ thơi gian ($tword()$) ta có kết quả sau.

Hệ quả 3.1 $L(\mathcal{A}, J) = tword(\bigcup_{T \in TrL(\mathcal{A})} dtot(T, J))$.

Phần tiếp theo luận án sẽ chứng minh một tính chất quan trọng của ADA là bài toán kiểm tra tính rỗng. Đây là một trong những bài toán quan trọng nhất trong việc nghiên cứu về các ô-tô-mát. Nó có ý nghĩa to lớn trong giải quyết bài toán kiểm chứng hệ thống có quyết định được hay không. Kết quả được biểu diễn qua định lý sau.

Định lý 3.1 *Cho ADA $(\mathcal{A}, J)$, bài toán kiểm tra tính rỗng của $(\mathcal{A}, J)$ là quyết định được.*

Chứng minh: Ta biết rằng, bản chất của một ADA là một mạng các ô-tô-mát thời gian một đồng hồ được ghép nối đồng bộ với nhau. Theo các kết quả trong [7], phép ghép nối này tạo ra một ô-tô-mát thời gian. Như vậy, một ADA là một ô-tô-mát thời gian. Mặt khác, theo hệ quả 2.2 thì bài toán kiểm tra ngôn

ngữ rỗng của ô-tô-mát thời gian là quyết định được, do đó bài toán kiểm tra tính rỗng của ADA cũng quyết định được. ■

Cuối cùng, chúng ta sẽ chứng minh một kết quả là mối quan hệ giữa ngôn ngữ Vết khoảng và ADA. Kết quả này được biểu diễn qua định lý sau.

Định lý 3.2 *Cho 1 ngôn ngữ Vết khoảng (L, J) với L là ngôn ngữ Vết chính quy trên bảng chữ cái phụ thuộc (Σ, D) và J là một hàm gán ràng buộc thời gian được xác định như trên, luôn tồn tại một ADA $(calA, J)$ sao cho $tL(\mathcal{A}, J) = tTr(L, J)$.*

Chứng minh: Để chứng minh định lý, trước tiên ta phải chứng minh rằng nếu L là một ngôn ngữ Vết chính quy thì luôn tồn tại một ô-tô-mát bất đồng bộ đoán nhận ngôn ngữ vết chính quy L. Thật vậy, theo kết quả trong [68, 75, 60] thì cho một ngôn ngữ Vết chính quy luôn tồn tại một AA đoán nhận ngôn ngữ chính quy này. Do đó, giả sử L là ngôn ngữ Vết chính quy trên bảng chữ cái phụ thuộc (Σ, D), ta luôn tồn tìm được một ô-tô-mát bất đồng bộ $\mathcal{A}$ đoán nhận ngôn ngữ vết L, tức là $TrL(\mathcal{A}) = L$. Tiếp theo, ta chứng minh rằng ô-tô-mát bất đồng bộ này kết hợp với hàm gán khoảng thời gian J chính là ADA cần tìm, tức là $tL(\mathcal{A}, J) = tTr(L, J)$. Theo định nghĩa ADA, cặp $(\mathcal{A}, J)$ là một ADA, với kết quả trên ta đã có $TrL(\mathcal{A}) = L$. Mặt khác, theo định nghĩa về ngôn ngữ được đoán nhận bởi ADA ta có $tL(\mathcal{A}, J) \widehat{=} \bigcup_{T \in TrL(\mathcal{A})} dtot(T, J)$. Theo định nghĩa ngôn ngữ vết thời gian của ngôn ngữ vết khoảng thì $tTr(L, J) \widehat{=} \bigcup_{T \in L} dtot(T, J)$. Do đó $tL(\mathcal{A}, J) \widehat{=} \bigcup_{T \in L} dtot(T, J) = tTr(L, J)$.

■

Từ các kết quả trên, chúng ta có thể sử dụng ô-tô-mát hữu hạn bất đồng bộ $\mathcal{A}$ để biểu diễn một ngôn ngữ Vết L, và cùng với một hàm gán khoảng thời gian $J : \Sigma \longrightarrow intv$ để biểu diễn một ngôn ngữ Vết thời gian.

3.3 Lôgic trên Vết thời gian

Lôgic thời gian tuyến tính đã được sử dụng để đặc tả các thuộc tính lôgic cho các hệ tương tranh như trong [34] và các công trình trước của họ. Lôgic này được mở rộng để có thể đặc tả các tính chất của các Vết. Tuy nhiên, các nghiên cứu này chưa hỗ trợ đặc tả thêm các ràng buộc thời gian của các hành động trong hệ thống. Do đó trong phần này, luận án đề xuất mở rộng về thời gian để đặc tả các thuộc tính của ngôn ngữ Vết thời gian.

Cú pháp và ngữ nghĩa

Xuất phát từ nghiên cứu lôgic thời gian tuyến tính đặc tả Vết, luận án đưa ra một lôgic tương tự có thêm đặc tả ràng buộc thời gian được gọi là lôgic thời gian thực tuyến tính (ký hiệu là TLTL_Σ - Timed Linear Temporal Logic). Lôgic này có cú pháp như sau:

$$\varphi ::= \top \mid a \mid \mathsf{EX}_I\varphi \mid \varphi \mathsf{U}_I \varphi \mid \neg\varphi \mid \varphi \vee \varphi$$

Với a là phần tử thuộc bảng chữ cái Σ, I thuộc $intv$, và $\top$ là ký hiệu *true*. Các toán tử EX_I, $\varphi\mathsf{U}_I$ là toán tử Next và toán tử Until tương tự như trên lôgic thời gian tuyến tính (LTL). Cho trước một Vết thời gian $(T, \theta) = (\langle V, \leq, \lambda\rangle, \theta)$, để tìm hiểu ngữ nghĩa của lôgic, trước tiên luận án nhắc lại một số khái niệm. Chúng ta xem xét một nhát cắt trên V như là một lát cắt tức thời của hệ tương tranh và định nghĩa ngữ nghĩa của TLTL_Σ dựa trên một nhát cắt của T. Để làm việc này, chúng ta cần một số khái niệm sau. Gọi C là một nhát cắt trên V. Ký hiệu $\Uparrow C$ là tập $\{y \in V | \exists x \in C.x < y\}$. Với hai nhát cắt C, C' ta định nghĩa $C \leq C'$ nếu và chỉ nếu $\forall x \in C \exists y \in C'.x \leq y$, và $C < C'$ nếu và chỉ nếu $C \leq C'$ và $C \neq C'$. Ngữ nghĩa của TLTL_Σ được đưa ra như sau:

$T, C \models \top$

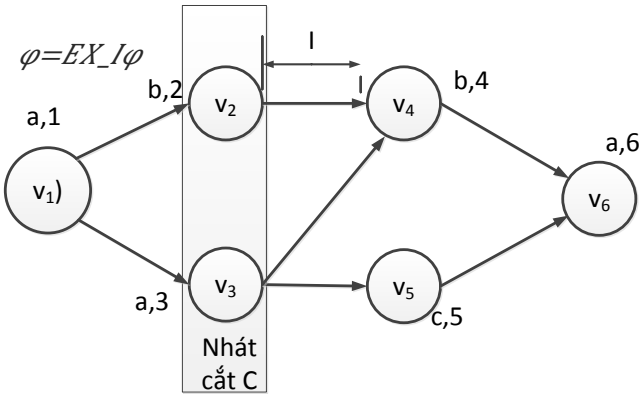
$T, C \models a$ nếu $\exists v \in C.\lambda(v) = a$

$T, C \models \mathsf{EX}_I\varphi$ nếu $\exists v \in V.(\,^{\lessdot}v \subseteq C \wedge T, (C - \,^{\lessdot}v \cup \{v\}) \models \varphi \wedge$
$\forall v' \in \,^{\lessdot}v.(\theta(v) - \theta(v')) \in I)$

$T, C \models \varphi\mathsf{U}_I\psi$ nếu tồn tại $v_1, \ldots, v_k \in V$ sao cho:
$C_0 = C, \,^{\lessdot}v_{i+1} \subseteq C_i, C_{i+1} = C_i - \,^{\lessdot}v_{i+1} \cup \{v_{i+1}\},\ T, C_i \models \varphi$
for $i = 0, \ldots, k-1$, and $T, C_k \models \psi$,
$\forall z \in (C_k - C) \forall v \in (C \cap \downarrow z).(\theta(z) - \theta(v) \in I)$[5]

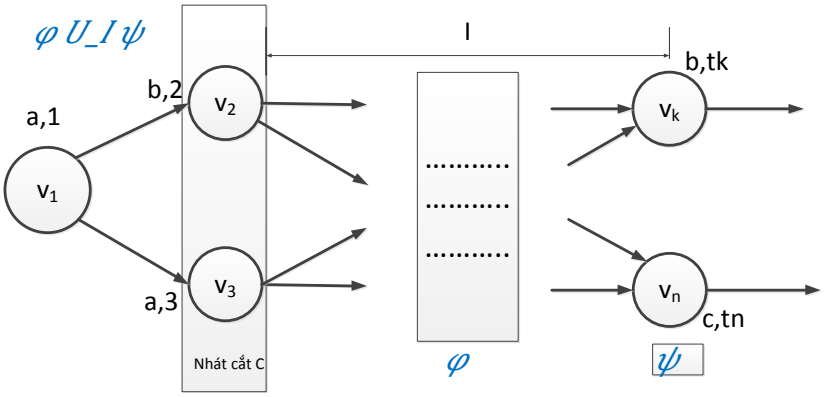
$T, C \models \neg\varphi$ nếu $T, C \not\models \varphi$

$T, C \models \varphi \vee \psi$ nếu $T, C \models \varphi$ hoặc $T, C \models \psi$



Hình 3.5: Ngữ nghĩa của toán tử EX_I

Do ta ẩn giấu yếu tố thời gian vào bảng chữ cái (trở thành bảng chữ cái thời gian) nên ngữ nghĩa của TLTL được định nghĩa tương tự như trường hợp không có thời gian. Đây là điểm rất quan trọng để chúng ta có thể lợi dụng các kết quả hữu ích của các nghiên cứu đã có trên Vết (không có thời gian). Ngữ nghĩa của hai toán tử $EX_I\varphi$ và $\varphi \cup_I \psi$ được minh họa trong Hình 3.5 và Hình 3.6.



Hình 3.6: Ngữ nghĩa của toán tử U_I

Gọi $\min(T) \widehat{=} \{v \in V | \nexists v' \in V.v' < v\}$, Vết thời gian T được gọi là thỏa công thức φ của $TLTL_\Sigma$ khi và chỉ khi $T, \min(T) \models \varphi$. Trong trường hợp này ta gọi T là mô hình của φ. Hơn thế, ta gọi công thức φ là thỏa được nếu tồn tại Vết thời gian T sao cho $T, \emptyset \models \varphi$. Tập tất cả các Vết thỏa công thức φ được kí hiệu là $ttL(\varphi)$ và gọi là ngôn ngữ được định nghĩa bởi φ.

Công thức $\mathsf{F}\varphi$ (Tương lai) và $\mathsf{G}\varphi$ (Toàn thể) trong TLTL_Σ được xác định thông qua các toán tử khác như sau.

$$\mathsf{F}\varphi \;\widehat{=}\; \top\mathsf{U}\varphi$$
$$\mathsf{G}\varphi \;\widehat{=}\; \neg\mathsf{F}\neg\varphi$$

Cho trước một ngôn ngữ Vết thời gian L và một công thức $\varphi \in TLTL_\Sigma$, bài toán kiểm chứng mô hình là bài toán kiểm tra xem liệu $L \subseteq L(\varphi)$ hay không. Bài toán này được giải một cách đơn giản dựa trên lý thuyết ô-tô-mát. Tuy nhiên, chúng ta cần phải chứng minh và tìm ra mối quan hệ giữa ngôn ngữ Vết thời gian, lôgic thời gian tuyến tính trên Vết và ô-tô-mát khoảng bất đồng bộ. Để làm được điều này, chúng ta phải giải bài toán về tính thỏa được của công thức $TLTL$, tức là cho một công thức $\phi \in TLTL$ liệu có tồn tại một ADA đoán nhận ngôn ngữ được sinh bởi công thức này hay không.

Tính thỏa được của TLTL

Tiếp theo chúng ta quan tâm tới tập các Vết thời gian được đoán nhận bởi ADA. Chúng ta sẽ chứng minh mối quan hệ giữa công thức TLTL và ADA.

Trước tiên, chúng ta xem xét lớp con $TLTL_1$ của công thức $TLTL$ mà ở đó không có ràng buộc cho toán tử U, tức là công thức có dạng:

$$\varphi ::= \top \mid a \mid \mathsf{EX}_I\varphi \mid \varphi\mathsf{U}_{[0,\infty)}\varphi \mid \neg\varphi \mid \varphi \vee \varphi$$

Đối với một công thức $TLTL\varphi$ gọi $ttL(\varphi)$ là biểu diễn tập các Vết thời gian thỏa công thức φ. Tương tự trường hợp không có thời gian, chúng ta có kết quả thể hiện dưới định lý sau:

Định lý 3.3 *Với mọi công thức $TLTL_1$ bất kỳ φ luôn tồn tại một ô-tô-mát khoảng bất đồng bộ $(\mathcal{A}, J)$ sao cho $ttL(\varphi) = tTrL(\mathcal{A}, J)$ và ngược lại.*

Chứng minh: Đặt $untimed(\varphi)$ là công thức LTL bỏ đi ràng buộc thời gian cho toán tử EX, tức là bỏ qua ràng buộc thời gian I trên toán tử EX_I. Theo kết quả trong [68, 75, 60] thì luôn tồn tại một AA đoán nhận ngôn ngữ vết thỏa công thức $untimed(\varphi)$, gọi AA này là $\mathcal{A}$, tức là $TrL(\mathcal{A}) = TrL(untimed(\varphi))$. Tiếp theo, với mỗi khoảng thời gian I cho EX_I trong φ, ta thêm ràng buộc thời gian I tương ứng với mỗi ký hiệu hành động a thuộc đỉnh v, tức là $\lambda(v) = a$ thỏa điều kiện của phép toán EX_I theo định nghĩa về ngữ nghĩa của công thức TLTL. Ta xây dựng J là tập các ràng buộc tương ứng này. Như vậy, ta có ADA $(\mathcal{A}, J)$ mà ngôn ngữ vết thời gian $tTr(\mathcal{A}, J) \widehat{=} \bigcup_{T \in TrL(\mathcal{A})} dtot(T, J)$ và ngôn ngữ vết thời gian thỏa công thức φ sẽ là ngôn ngữ vết thời gian thỏa ngôn ngữ khoảng (L, J) với $L = untimed(\varphi)$, tức là $ttL(\varphi) = \bigcup_{T \in TrL(untimed(\varphi))} dtot(T, J) = \bigcup_{T \in TrL(\mathcal{A})} dtot(T, J) = tTrL(\mathcal{A}, J)$. Từ

đó ta có điều phải chứng minh. ■

Vì một ô-tô-mát khoảng bất đồng bộ $(\mathcal{A}, J)$ mà $tTrL(\varphi) = ttL((\mathcal{A}, J))$ có thể được xây dựng một cách hiệu quả, và vì bài toán kiểm tra tính rỗng của ô-tô-mát khoảng bất đồng bộ là quyết định được nên ta có tính thoản được của $TLTL_1$ là quyết định được.

Hệ quả 3.2 *Cho một công thức $TLTL_1$ φ và một ô-tô-mát khoảng bất đồng bộ $(\mathcal{A}, J)$, luôn tồn tại một thuật toán để chỉ ra liệu $T \models \varphi$ với $T \in tL((\mathcal{A}, J))$.*

3.4 Các nghiên cứu liên quan

Vết Mazurkiewicz và các mở rộng khác của nó đã và đang được quan tâm nghiên cứu để đặc tả và kiểm chứng các hệ thống, đặc biệt là các hệ thống có tính tương tranh [68, 74, 31]. Đối với một số hệ thống tương tranh có thêm các ràng buộc về thời gian, một số đề xuất mở rộng về thời gian của Vết cũng được giới thiệu tuy chỉ dừng ở việc giải quyết một số bài toán nhỏ trong bài toán về kiểm chứng mô hình. Trong [73, 12], Tomohiro Yoneda và đồng nghiệp đề xuất kỹ thuật giảm thứ tự bộ phận cho bài toán kiểm chứng lý thuyết Vết thời gian. Các tác giả đã đưa ra khái niệm về Vết thời gian, là một chuỗi hữu hạn hoặc vô hạn các sự kiện được gán các nhãn thời gian trên các mạch đồng bộ [70]. Với cách định nghĩa này, Vết thời gian có thể đặc tả được các ràng buộc về thời gian của bài toán đặc tả các mạch đồng bộ có thời gian. Phương pháp này là một mở rộng của phương pháp kiểm chứng lý thuyết Vết để nó có thể mô tả được các mạch thời gian cũng như các đặc tả thời gian một cách đúng đắn. Phương pháp đề xuất kỹ thuật sử dụng tính đồng thời giữa các sự kiện để giảm không gian trạng thái trong bài toán kiểm chứng, do đó ta có thể sử dụng kỹ thuật này để kiểm chứng các mạch lớn hơn (mà không sợ làm tăng không gian trạng thái). Các tác giả đã hực nghiệm kỹ thuật với các mạch STARI. Kết quả thực nghiệm đã chỉ ra tính hiệu quả của phương pháp trong việc giảm bùng nổ không gian trạng thái trong ô-tô-mát thời gian. Tuy nhiên, lý thuyết Vết thời gian trong này được định nghĩa một cách cụ thể, cho một bài toán nhất định, không tổng quát và không đưa ra các khái niệm liên quan như ngôn ngữ, ô-tô-mát đoán nhận cũng như phương pháp để có thể sử dụng Vết thời gian cho đặc tả các bài toán có ràng buộc thời gian khác. Luận án đưa ra một định nghĩa đầy đủ và bao quát hơn về lý thuyết Vết thời gian, đưa ra một công cụ hỗ trợ hiệu quả cho việc đặc tả các giao diện thành phần của hệ thống dựa trên thành phần có tính tương

tranh và ràng buộc thời gian.

Các nghiên cứu trong [23] đã sử dụng Vết thời gian trong việc giải quyết bài toán kiểm chứng động đối với các hệ thống có ràng buộc thời gian. Kiểm chứng động là kiểm tra xem các thực thi của hệ thống có thỏa mãn hay vi phạm các thuộc tính hay không. Kết quả trong nghiên cứu này đưa ra kỹ thuật kiểm chứng sử dụng hai "hộp đen" để đối sánh các Vết thời gian tương ứng trong quá trình hoạt động. Một "hộp đen" là hệ thống thực và một là mô hình của hệ thống thực. Hai mô hình này được thực hiện cùng lúc, cùng sinh ra các Vết thời gian là các thực thi của hệ thống. Bộ theo dõi sẽ nắm giữ được các Vết này và tiến hành đối sánh. Đây là một ứng dụng của Vết thời gian cho bài toán kiểm chứng, là minh chứng cho thấy tính hiệu quả của lý thuyết Vết thời gian đề xuất trong luận án.

3.5 Kết luận

Chương này luận án đã giới thiệu một phương pháp dựa trên mở rộng về thời gian của lý thuyết Vết Mazurkiewicz. Luận án đề xuất khái niệm Vết thời gian, ngôn ngữ Vết thời gian, Vết khoảng, mối quan hệ giữa Vết khoảng và Vết thời gian cũng như ý nghĩa của Vết khoảng. Kết quả trong nghiên cứu này cũng chỉ ra rằng với mỗi ngôn ngữ Vết thời gian chính quy luôn tồn tại một ô-tô-mát khoảng bất đồng bộ đoán nhận và ngược lại. Bên cạnh đó, để hỗ trợ cho việc đặc tả thuộc tính lôgic của hệ thống, luận án giới thiệu lôgic thời gian tuyến tính trên Vết thời gian, luận án đã đưa ra và chứng minh bài toán thỏa được của lôgic này là quyết định được. Như vậy, với các kết quả này, chúng ta hoàn toàn có khả năng đặc tả và kiểm chứng các hệ tương tranh có yếu tố thời gian một cách đơn giản, ngắn gọn và tiện lợi. Các kết quả này được công bố trong các công trình nghiên cứu [1] và [5] trong danh mục công trình khoa học đã công bố. Một số nghiên cứu cũng đã ứng dụng lý thuyết Vết thời gian vào giải một số bài toán. Đây là những minh chứng cho tính hiệu quả của lý thuyết Vết và các mở rộng về thời gian. Một cách tiếp cận khác, các nghiên cứu trong luận án áp dụng Vết thời gian trên một số mô hình thiết kế hệ thống để các mô hình này có thể đặc tả được các tính chất tương tranh và thời gian. Chi tiết các mô hình này sẽ được giới thiệu trong các chương tiếp theo dưới đây.

Chương 4

Một mô hình cho hệ thống tương tranh có ràng buộc thời gian dựa trên các khái niệm và kỹ thuật rCOS

Chương này luận án giới thiệu một mô hình được đề xuất cho các hệ thống dựa trên thành phần có các ràng buộc về thời gian và tính tương tranh. Luận án sử dụng các Vết thời gian để đặc tả các giao thức trong các giao diện thành phần để hỗ trợ đặc tả các truy cập đồng thời có yếu tố thời gian tới các dịch vụ của một thành phần. Luận án định nghĩa một hợp đồng bao gồm đặc tả phương thức và định nghĩa một thành phần như là một sự thực thi của một hợp đồng. Mô hình nghiên cứu trong chương này hỗ trợ sự phân biệt giữa các ràng buộc chức năng và phi chức năng, và kiểm chứng hình thức kết hợp của các hệ thống dựa trên thành phần có ràng buộc thời gian. Các kết quả nghiên cứu cũng chỉ ra thuật toán tìm các giao thức khi kết hợp các thành phần với nhau để tạo thành phần mới với nhiều chức năng hơn.

4.1 Giới thiệu

Các kỹ thuật dựa trên hướng đối tượng và thành phần đang trở lên phổ biến và được sử dụng rộng rãi trong mô hình và thiết kế các hệ thống phần mềm

phức tạp. Chúng cung cấp các hỗ trợ hiệu quả cho việc phân rã ứng dụng thành các đối tượng và các thành phần, mà có thể được nhận ra bởi việc sử dụng lại và mở rộng các thiết kế và các thực thi đã có. Việc phân tích và kiểm chứng những hệ thống như vậy có thể cũng dễ dàng hơn bởi vì các kiến trúc kết hợp cả các thành phần. Hiện nay có nhiều các công nghệ dựa trên thành phần mà điển hình là Component CORBA, EBJ, J2EE, COM, và .NET.,v.v. Các ngôn ngữ hình thức và bán hình thức như UML [65], JML [17], và BIP [9] đang trở lên thông dụng để hỗ trợ phát triển hướng mô hình. Tuy nhiên, những mô hình này hoặc không hỗ trợ mức trừy tượng cao hoặc chỉ tập trung và hợp đồng và thiết kế chức năng của thành phần và đối tượng, và vẫn chưa cung cấp đủ cho đặc tả và phân tích về chất lượng dịch vụ của hệ thống từ những thành phần và đặc biệt là các ràng buộc về thời gian và tính đồng thời (hay tương tranh). Bài toán nghiên cứu trong chương này của luận án là đề xuất phương pháp đặc tả và kiểm chứng các hệ thống tương tranh, có ràng buộc thời gian hướng thành phần. Kỹ thuật đưa ra phải hỗ trợ đặc tả các thành phần hệ thống tại mức trừu tượng cao bao gồm cả chất lượng dịch vụ và có thể cung cấp cho việc phát triển một ngôn ngữ mẫu mà có thể kết hợp với nhiều ngôn ngữ lập trình khác nhau để hỗ trợ lập trình dựa trên thành phần.

Kỹ thuật đặc tả rCOS [45, 53] được phát triển tại UNU-IIST là một kỹ thuật đáp ứng được các yêu cầu của bài toán đặt ra ngoại trừ yêu cầu về ràng buộc thời gian và tính tương tranh. Do đó, trong nghiên cứu này, luận án đề xuất một mô hình dựa trên các khái niệm và kỹ thuật của rCOS và có thể giải quyết các vấn đề thiếu sót như đề đề cập trên về rCOS. Một bài toán đơn giản thể hiện ứng dụng của phương pháp được phát biểu như sau: Ta có một thành phần A với hai phương thức cung cấp dịch vụ là tính giá trị căn bậc hai và căn bậc năm của một số và có một ràng buộc về thời gian cung cấp dịch vụ là $[a, b]$. Một thành phần B có một phương thức cần tính toán một biểu thức có chứa căn bậc hai và căn bậc năm. Hỏi liệu thành phần B có thể sử dụng các dịch vụ do thành phần A cung cấp hay không, kiểm tra nó như thế nào? Câu trả lời được cho trong ví dụ 4.1 của chương này.

Phương pháp tiếp cận của luận án có thể được mô tả như sau. Luận án giới thiệu về quan hệ phụ thuộc giữa các phương phức của thành phần, và xem xét một giao thức tương tác như là một tập các Vết thời gian theo quan hệ phụ thuộc đã giới thiệu. Bằng cách này, tính đồng thời và các ràng buộc về thời gian thực thi của các hành động trong thành phần có thể được đặc tả bởi một sự đan xen truyền thống trong giao diện của thành phần. Được trang bị với các đặc tả về tính đồng thời và thời gian trong giao diện thành phần, mô hình thành phần

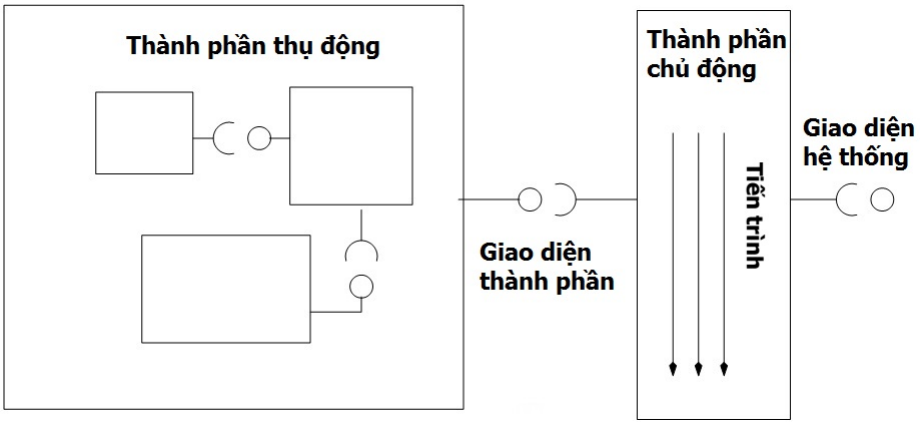
của trong nghiên cứu này hỗ trợ tốt cho bài toán phân tích chất lượng dịch vụ trong phát triển phần mềm [38]. Nhìn chung, một trong các khó khăn trong sử dụng mô hình thành phần cho phát triển hệ thống là làm thế nào để tính toán các giao thức đồng thời và các vị từ đồng thời cho các thành phần được kết hợp từ các thành phần con. Luận án sẽ làm rõ vấn đề này trong một số trường hợp.

Cấu trúc chương được tổ chức như sau. Trong phần tiếp theo, luận án đề xuất một kiến trúc cho các hệ thống thành phần. Sau đó, luận án giới thiệu khái niệm về biểu thức chính quy thời gian đồng thời như là một cơ sở cho đặc tả các ngôn ngữ Vết thời gian được sử dụng như là giao thức tương tác các thành phần. Mô hình chính được luận án giới thiệu trong phần 4. Các phần sau sẽ thảo luận tính đúng đắn của đặc tả giao thức và các kết luận.

4.2 Kiến trúc thành phần và các giao thức tương tác

Trong phần này, luận án đề xuất một kiến trúc dựa trên thành phần cho các hệ thống phần mềm. Với các thành phần tự chủ, thông qua giao diện của chúng, chúng ta có thể đặt thêm thành phần sử dụng các đầu nối để hình thành các thành phần ngày một lớn hơn với nhiều dịch vụ hơn cho tới khi chúng trở thành một thành phần đóng mà có thể cung cấp đủ các dịch vụ cho thực thi các ràng buộc. Đối với thành phần đóng, đây là thành phần mà không có ràng buộc gì trên giao diện, tức là không cần bất kỳ các dịch vụ ngoài nào để thực hiện các dịch vụ của chúng. Hệ thống của chúng ta xây dựng theo cách này có hai phần riêng biệt: Phần thụ động là một thành phần đóng được ghép từ tập các thành phần và phần chủ động là một tập các tiến trình tương tác với các kích thích bên ngoài với một số ràng buộc về thời gian và sử dụng dịch vụ từ phần thụ động để thỏa mãn các yêu cầu từ các nhân tố bên ngoài của hệ thống. Kiến trúc hệ thống này được mô tả trong Hình 4.1.

Mỗi tiến trình tương ứng với một luồng trong JAVA, ở đây cho phép cơ chế đa luồng. Điều này có nghĩa là có một cơ hội cho nhiều hơn một luồng truy cập một dịch vụ từ một thành phần con trong phần thụ động một cách trực tiếp hoặc qua các dịch vụ khác. Làm thế nào để nắm được trường hợp này để tăng sự thực thi và đặc biệt tránh khóa chết gây ra bởi truy cập đồng thời tới một nguồn chung luôn là bài toán nhiều thách thức cũng như cho bất kỳ điều khiển đồng thời nào trong các hệ thông tính toán. Nếu điều này xảy ra, thậm chí khi các tiền điều kiện của một dịch vụ được thỏa mãn, dịch vụ có thể bị ngắt bởi vì khóa chết hoặc thậm chí dịch vụ ngắt nhưng có yếu tố thời gian thi tối thiểu của nó không thể dự đoán được. Vì thế không phải tất cả các chuỗi truy cập



Hình 4.1: Kiến trúc hệ thống

dịch vụ nên được cho phép. Một hợp đồng nên bao gồm đặc tả của các chuỗi có khả năng cho phép của chuỗi truy cập cũng như các đặc tả về tiền điều kiện và hậu điều kiện đối với thành phần. Đặc tả này, cũng cho phép thực hiện như là một giả thiết từ các thành phần đối với môi trường mà chúng cần để thực hiện, và được thực hiện bởi giao thức tương tác. Vì thế, một giao thức tương tác được xem như là một tiền điều kiện trên thứ tự tạm thời của các truy cập vào dịch vụ của thành phần và phục vụ như là một phần của hợp đồng giữa các thành phần và môi trường của nó. Một thành phần có thể cung cấp các dịch vụ tốt chỉ khi nếu môi trường của nó cho phép giao thức được đặc tả trong hợp đồng tương tác với thành phần.

Như đã trình bày trong chương 3, trong thành phần có yếu tố thời gian, một phương thức (dịch vụ) m không những có đặc tả về chức năng mà còn đặc tả phi chức năng bao gồm đặc tả về có yếu tố thời gian thi yếu nhất (WCET). Như vậy, sử dụng Vết thời gian là phù hợp cho đặc tả các hành vi của hệ thống.

Trong phần tiếp theo, luận án giới thiệu khái niệm về biểu thức chính quy thời gian đồng thời như là một sự mở rộng của biểu thức chính quy cổ điển dựa trên ngôn ngữ Vết của Mazurkiewicz và các khái niệm về Vết thời gian.

4.3 Vết thời gian và biểu diễn của nó

Một ngôn ngữ chính quy (được đoán nhận bởi một ô-tô-mát hữu hạn) $L \subseteq \Sigma^*$ là nhất quán (consistent) khi và chỉ khi với bất kỳ $\omega \in L$, và bất kỳ $\omega' \in \Sigma^*$ chúng ta có nếu $wtot(\omega) = wtot(\omega')$ thì $\omega' \in L$. Với ngôn ngữ như vậy, chúng ta gọi L là một ngôn ngữ chính quy nhất quán trên (Σ, D). Rõ ràng là với một ngôn ngữ L

như thế, ta có $ttow(wtot(L)) = L$.

Kết quả sau có được theo các nghiên cứu của Zielonka trong [75] và được thể hiện qua định lý sau.

Định lý 4.1 *Lấy $L \subseteq \Sigma^*$, L là một ngôn ngữ chính quy nhất quán trên (Σ, D) nếu và chỉ nếu L được đoán nhận bởi một ô-tô-mát hữu hạn bất đồng bộ.*

Vì vậy, chúng ta có thể sử dụng ô-tô-mát hữu hạn bất đồng bộ $\mathcal{A}$ để biểu diễn một ngôn ngữ Vết L, và cùng với một hàm gán ràng buộc khoảng thời gian $J : \Sigma \longrightarrow intv$ để biểu diễn một ngôn ngữ Vết thời gian[1]. Chúng ta sẽ sử dụng một tập các Vết thời gian Mazurkiewicz mà có biểu diễn hữu hạn như là một đặc tả cho các giao thức trong hợp đồng của thành phần. Một cách biểu diễn đơn giản bao gồm:

- Cho một bảng chữ cái phụ thuộc khoảng (Σ, D, J) biểu diễn các hành động của hệ thống với các tính chất phụ thuộc nhau và các ràng buộc khoảng thời gian cho mỗi hành động, và
- Cho một biểu thức chính quy R trên bảng chữ cái (Σ, D, J) để biểu diễn một ngôn ngữ chính quy $L(R)$.

Một ngôn ngữ Vết khoảng (và một ngôn ngữ Vết thời gian) có thể được sinh ra từ việc biểu diễn sử dụng ánh xạ $wtot$. Chúng ta sẽ sử dụng cách biểu diễn này, và gọi cặp $((\Sigma, D, J), R)$ là một biểu thức chính quy Vết thời gian. Đối với mục đích của nghiên cứu này, luận án định nghĩa phép ghép hai ngôn ngữ Vết thời gian như sau. Cho bảng chữ cái phụ thuộc khoảng (Σ, D, J) và $(\Sigma_i, D_i, J_i), (i = 1, 2)$ là các bảng chữ cái phụ thuộc khoảng thỏa mãn $\Sigma = \Sigma_1 \cup \Sigma_2, D_i = D|_{\Sigma_i}, J_i = J|_{\Sigma_i}$. L_i được ký hiệu là một ngôn ngữ Vết thời gian biểu diễn bởi một biểu thức chính quy Vết thời gian $((\Sigma_i, D_i, J_i), R_i)$. Phép ghép trên D của L_1 và L_2, biểu diễn bởi $L_1 L_2$ là ngôn ngữ Vết thời gian biểu diễn bởi $((\Sigma, D, J), R_1 R_2)$.

4.4 Mô hình thành phần

Trong nghiên cứu này, luận án sửa đổi và mở rộng mô hình thành phần trong rCOS được phát triển bởi He, Liu và Li trong [53, 45] dựa trên các khái niệm đưa ra trong phần trước. Mô hình kết quả ở đây sẽ gồm một sự đơn giản hóa và sự mở rộng cho mô hình của rCOS và phiên bản thời gian đã được D.V. Hung định nghĩa [40], với một số chi tiết về giao thức tương tác giữa các thành phần và các truy xuất đồng thời với các ràng buộc thời gian cho các dịch vụ được

[1] Được trình bày trong chương 3, mục 3.2

cung cấp bởi các thành phần. Mô hình này là khác so với một trong các mô hình trước của D.V Hung và cộng sự [43].

4.4.1 Thiết kế

Như đã nói trong phần trước, một thành phần cung cấp các dịch vụ cho các khách hàng của nó. Một dịch vụ hoặc là dữ liệu hoặc là phương thức. Để đặc tả chức năng cho các phương thức, luận án sử dụng các khái niệm UTP truyền thống (He và Hoare [39]) trong đó một phương thức được đặc tả như là một quan hệ với ký hiệu của nó có dạng *op(in, out)*, trong đó *in* và *out* là tập các biến. Đặc tả thời gian chỉ là một ràng buộc trên WCET của các phương thức.

Định nghĩa 4.1 (Thiết kế) *Một thiết kế là một bộ $\langle \alpha, FP, FN \rangle$, với α ký hiệu tập các (chương trình) biến được sử dụng bởi phương thức, FP ký hiệu đặc tả chức năng của phương thức được viết trong UTP, và FN ký hiệu đặc tả có yếu tố thời gian thi.*

- *FP là một mệnh đề có dạng $(p \vdash R)$,*

  trong đó p là các tiền điều kiện của phương thức mà là giả thiết trên giá trị đầu vào của các biến trong $(\alpha \setminus out)$ mà phương thức có thể dựa trên đố để kịch hoạt, và R là một hậu điều kiện liên quan tới các quan sát khởi đầu tới các quan sát cuối cùng(được biểu diễn bởi các biến chính trong tập $\{x' \mid x \in (\alpha \cup out)\}$), và

- *FN có hình thức $l \leq dur \leq u$, với l, u là các số thực không âm, $l \leq u$.*

Làm mịn các thiết kế

Trong luận án, các khái niệm về quan hệ làm mịn cho thiết kế được xác định như trong UTP và trong [74]. Một thiết kế $D_1 = \langle \alpha, FP_1, l_1 \leq dur \leq u_1 \rangle$ được làm mịn bởi thiết kế $D_2 = \langle \alpha, FP_2, l_2 \leq dur \leq u_2 \rangle$ (ký hiệu bởi $D_1 \sqsubseteq D_2$) nếu và chỉ nếu: $(\forall v, v' \bullet FP_2 \Rightarrow FP_1)$, $l_1 \leq l_2, u_2 \leq u_1$. Trong đó v, v' là các véc tơ các biến chương trình.

Chú ý rằng đối với đặc tả phi chức năng, một sự làm mịn nên có khoảng ràng buộc của WCET nằm trong khoảng ràng buộc của thiết kế mà nó làm mịn.

Ghép tuần tự

Nếu $D_1 = \langle alpha, FP_1, l_1 \leq dur \leq u_1 \rangle$ và $D_2 = \langle alpha, FP_2, l_2 \leq dur \leq u_2 \rangle$ là các thiết kế, thì $D_1; D_2 \mathrel{\widehat{=}} \langle \alpha, FP, l \leq dur \leq u \rangle$,

Trong đó $FP \mathrel{\widehat{=}} \exists m \bullet FP_1(m) \wedge FP_2(m)$ với giả thiết rằng $FP_1 = FP_1(v')$ và $FP_2 = FP_2(v)$, và $l = l_1 + l_2, u = u_1 + u_2$.

Bây giờ và sau này, luận án sử dụng $F[x_1/x]$ để diễn tả việc thay thế x_1 bằng x trong biểu thức F.

4.4.2 Giao diện và hợp đồng

Thành phần quan trọng của một thành phần là giao diện. Giao diện đặc tả những dịch vụ mà nó cung cấp và dịch vụ mà nó yêu cầu. Hợp đồng của một thành phần là một đặc tả giao diện của thành phần đó. Từ việc thảo luận một cách không hình thức trước, luận án đưa ra định nghĩa hình thức của hai khái niệm trong lập trình dựa trên thành phần như sau.

Ký hiệu Fd là một khai báo đặc trưng là một tập các biến, Md là một khai báo phương thức, mỗi phương thức $m \in Md$ có dạng $op(in, out)$, với in và out là tập các biến.

Định nghĩa 4.2 (Giao diện) *Một giao diện là một cặp $I = (I_p, I_r)$, với $I_p = \langle Fd_p, Md_p \rangle$, và $I_r = (Fd_r, Md_r)$. I_p được gọi là giao diện cung cấp của I và I_r là giao diện yêu cầu của I.*

Định nghĩa 4.3 (Hợp đồng) *Một hợp đồng là một bộ $\langle I, Init, MSpec, Inv_p, Inv_r, Pro_p, Pro_r \rangle$, với:*

- *$I = (I_p, I_r)$ là một giao diện được định nghĩa trong định nghĩa 4.2. Gọi $Md = Md_r \cup Md_p, Fd = Fd_r \cup Fd_p$,*
- *$Init$ là một sự khởi đầu mà gắn với mỗi biến trong Fd và mỗi biến địa phương với một giá trị cùng kiểu,*
- *$MSpec$ là một hàm đặc tả phương thức mà gắn mỗi phương thức $op(in, out)$ trong $Md = Md_r \cup Md_p$ với một thiết kế $\langle \alpha, FP, FN \rangle$, với $(\alpha \setminus (in \cup out)) \subseteq Fd$,*
- *Inv_p and Inv_r là các mệnh đề trên các đặc trưng được cung cấp và đặc trưng được yêu cầu một cách tương ứng trong hợp đồng (được gọi là bất biến của hợp đồng). Inv_p biểu diễn một thuộc tính bất biến của các biến trong trong đặc tả đặc trưng Fd_p. Vì thế, Inv_p thỏa bởi Init. Inv_r biểu diễn các thuộc tính mà được yêu cầu cho giá trị của các biến trong Fd_r khi chúng được cung cấp. Inv_p biểu diễn giả thiết thành phần tạo ra về hành vi đối với môi trường, và*
- *Pro_p và Pro_r là các đặc tả giao thức, mà là các Vết thời gian.Pro_p là ngôn ngữ Vết thời gian được biểu diễn bởi một biểu thức chính quy trên (Md_p, D_p, J_p), và Pro_r là một ngôn ngữ Vết thời gian được biểu diễn bởi biểu*

thức chính quy trên (M_r, D_r, J_r). *Cả hai* (Md_p, D_p, J_p) *và* (Md_r, D_r, J_r) *là các bảng chữ cái khoảng,* D_p *là quan hệ phụ thuộc trên* Md_p *mà thành phần cung cấp (*D_P *càng lớn thì tính đồng thời càng ít), và* D_r *là quan hệ phụ thuộc trên* Md_r *mà thành phần mong đợi từ môi trường, và đối với bất kỳ phương thức* m *nào với* $Mspec(m) = \langle \alpha, FP, l \leq dur \leq u \rangle$, *lấy* $J_{[.]}(m) = [l, u]$, *với* $[.]$ *hoặc là* p *hoặc là* r.

Các biến trong F_d là chỉ đọc đối với các thành phần khác. Inv_p trong một hợp đồng biểu thị một thuộc tính của các biến của hợp đồng mà nó đưa ra cho môi trường, và vì vậy Inv_p nên được thỏa mãn bởi bất kỳ phương thức nào của hợp đồng. Để làm rõ hơn về giao thức trong hợp đồng, luận án đưa ra một sự phân lớp cho khái niệm này như sau. D_p là quan hệ phụ thuộc giữa các phương thức cung cấp mà thành phần đưa ra cho môi trường. Những phương thức a và b mà $(a, b) \in D_p$ phải được gọi một cách tuần tự, tức là một phương thức được yêu cầu chỉ khi phương thức khác đã kết thúc. Chỉ Vết thời gian T của lời gọi tới phương thức trong M_p từ môi trường là đúng với $ttr(Pro_p)$ (tức là $ttword(T) \subseteq ttword(ttr(Pro_r))$) là được phép. Pro_r là tập các chuỗi lời gọi mà thành phần sử dụng để tương tác với các thành phần khác từ các dịch vụ mà nó yêu cầu. Từ đây về sau, đối với một quan hệ R và một tập $A \subseteq dom(R)$ bởi $R|_A$ chúng ta ký hiệu là giới hạn của R trên A, với $dom(R)$ ký hiệu miền giá trị của quan hệ R. Bất kỳ hành động (phương thức) m mà có đặc tả $\langle a, FP, l \leq dur \leq u \rangle$), chúng ta luôn giả thiết rằng m được gán với khoảng thời gian $[l, u]$.

Định nghĩa 4.4 (Làm mịn hợp đồng) *Hợp đồng* $Ctr_1 = \langle (I_{p1}, I_{r1}), MSpec_1, Init_1, Invp_1, Invr_1, Prop_1, Pror_1 \rangle$ *được làm mịn bởi hợp đồng*

$Ctr_2 = \langle (I_{p2}, I_{r2}), MSpec_2, Init_2, Inv_{p2}, Inv_{r2}, Pro_{p2}, Pro_{r2} \rangle$ *(ký hiệu* $Ctr_1 \sqsubseteq Ctr_2$ *) nếu và chỉ nếu:*

- $Fd_{p1} \subseteq Fd_{p2}$, $Fd_{r2} \subseteq Fd_{r1}$, *và* $Init_2|_{Fd_{p1}} = Init_1|_{Fd_{p1}}$, *với hàm* f *và tập* A, $f|_A$ *biểu thị hạn chế của* f *trên* A,
- $Md_{p1} \subseteq Md_{p2}$ *và* $Md_{r2} \subseteq Md_{r1}$,
- *Đối với tất cả phương thức op được khai báo trong* Md_{p1}, $Mspec_1(op) \sqsubseteq Mspec_2(op)$, *và* $Inv_{p2} \Rightarrow Inv_{p1}$,
- *Với tất cả phương thức khai báo trong* Md_{r2}, $Mspec_2(op) \subseteq Mspec_1(op)$, *và* $Inv_{r1} \Rightarrow Inv_{r2}$, *và*
- $D_{p1} = D_{p2}|_{Md_{p1}}$, $Pro_{p1} \subseteq Pro_{p2}$, $D_{r2} = D_{r1}|_{Md_{r2}}$, $Pro_{r2} \subseteq Pro_{r1}$.

Chúng ta giải thích định nghĩa này như sau. Ctr_2 cung cấp tất cả các dịch vụ mà Ctr_1 có, thậm chí còn tốt hơn, và có thể cung cấp nhiều hơn. Ctr_2 cần ít và yếu hơn các dịch vụ của Ctr_1 có. Điều kiện $Inv_{p2} \Rightarrow Inv_{p1}$ nói rằng thuộc tính của biến được đảm bảo bởi Ctr_1 thì cũng được đảm bảo bởi Ctr_2, trong khi điều kiện $Inv_{r1} \Rightarrow Inv_{r2}$ nói rằng giả thiết về đặc trưng từ môi trường tạo bởi Ctr_2 là yếu hơn Ctr_1. Vì thế, chúng ta có thể sử dụng Ctr_2 để thay thế Ctr_1 trong bất kỳ ứng dụng nào mà các dịch vụ trong Ctr_1 vẫn được đảm bảo.

4.4.3 Ghép nối các hợp đồng

Các hợp đồng có thể được ghép nối theo nhiều cách để hình thành các hợp đồng mới. Một khó khăn cho tính toán ghép nối các hợp đồng là làm thế nào có thể tính toán được các giao thức cho phép ghép đó. Điều này được thực hiện trong lý thuyết về hợp đồng [74]. Cách đơn giản nhất để ghép hai hợp đồng là đặt chúng với nhau nếu chúng có tập các đặc trưng và phương thức không giao nhau, và chúng có thể được thực thi trên các thiết bị phần cứng độc lập mà có thể chạy song song một cách độc lập với nhau.

Định nghĩa 4.5 (Ghép nối các hợp đồng) *Gọi* $Ctr_i = \langle (I_{pi}, I_{ri}), Mspec_i, Init_i, Invp_i, Inv_{ri}, Pro_{pi}, Pro_{ri} \rangle$, $i = 1, 2$ *là các hợp đồng mà có tập các đặc trưng và phương thức (yêu cầu và cung cấp) là không giao nhau. Phép hợp của* Ctr_1 *và* Ctr_2 *là hợp đồng* $Ctr_1 \cup Ctr_2 = \langle (I_{p1} \cup I_{p2}, I_{r1} \cup I_{r2}), Mspec_1 \cup Mspec_2, Init_1 \cup Init_2, Inv_{p1} \wedge Inv_{p2}, Inv_{r1} \wedge Inv_{r2}, Pro_p, Pro_r \rangle$, *với* $D_p = D_{p1} \cup D_{p2}$, $Prop = Pro_{p1} \cup Pro_{p2} \cup (Pro_{p1}.Pro_{p2})$, $D_r = D_{r1} \cup D_{r2}$, $Pro_r = Pro_{r1} \cup Pro_{r2} \cup (Pro_{r1}.Pro_{r2}).Pro_{p1}.Pro_{p2}$.

Vấn đề cần giải thích duy nhất trong định nghĩa này là làm thế nào để định nghĩa được giao thức của phép hợp. Khi đặt các hợp đồng cạnh nhau, bởi vì chúng được giả thiết là độc lập, tất cả các phương thức và đặc trưng có thể được sử dụng một cách song song. Phép nối $Pro_{p1}.Pro_{p2}$ là phép ghép nối hai Vết thời gian[2] với quan hệ phụ thuộc Dp mà không bao gồm sự phụ thuộc giữa bất kỳ phương thức nào của một thành phần và bất kỳ phương thức nào từ thành phần khác, đặc tả rằng các phương thức trong hai thành phần có thể được goi song song, cung cấp các giao thức từ mỗi thành phần được cho phép (chú ý rằng $Pro_{p1}Pro_{p2}$ và $Pro_{p2}Pro_{p1}$ là giống nhau về quan hệ Dp). Định nghĩa cũng nói rằng hợp đồng được ghép nối cũng nên cho phép các phương thức trong mỗi thành phần được sử dụng như nguyên gốc của nó.

[2]Được định nghĩa trong chương 3

Một cách khác để kết nối các hợp đồng là kết nối các phương thức được yêu cầu của một hợp đồng tới phương thức được cung cấp của hợp đồng khác.

Lấy $Ctr_i = \langle (I_{pi}, I_{ri}),\ Mspec_i,\ Init_i, Invp_i,\ Inv_{ri}, Pro_{pi},\ Pro_{ri} \rangle$, $i = 1, 2$ là các hợp đồng mà có tập tương thích các đặc trưng và phương thức cung cấp, tập tương thích các đặc trưng và phương thức được yêu cầu, tức là $f \in (Fd_{p1} \cap Fd_{p2})$ kéo theo $Init_1(f) = Init_2(f)$ và $op \in (Md_{p1} \cap Md_{p2}) \cup (Md_{r1} \cap Md_{r2})$ kéo theo $MSpec_1(op) \Leftrightarrow MSpec_2(op)$. $D_{p1}|_{(Md_{p1} \cap Md_{p2})} = D_{p2}|_{(Md_{pl} \cap Md_{p2})}, D_{r1}|_{(Md_{rl} \cap Md_{r2})} = D_{r2}|_{(Md_{r1} \cap Md_{r2})}$, và $Pro_r r1 \subseteq Pro_r p2$. Giả thiết rằng $I_{r1} \subseteq I_{p2}$, và $Inv_{p2} \Rightarrow Inv_{r1}$ và $Mspec_1(op) \sqsubseteq Mspec_2(op)$ đối với tất cả $op \in Md_{r1}$. Với những điều kiện như thế, hợp đồng Ctr_2 cung cấp tất cả các dịch vụ được yêu cầu bởi hợp đồng Ctr_1, và thỏa tất cả các giả thiết tạo ra bởi Ctr_1 về môi trường.

Cắm Ctr_1 vào Ctr_2, ký hiệu là $Ctr1 >> Ctr2$ được định nghĩa như sau:

$Ctr_1 >> Ctr_2 = \langle (I_{p1} \cup I_{p2},\ I_{r2}), Mspec_1|_{Md_{p1}} \cup Mspec_2,\ Init_1|_{Fd_{r1}} \biguplus Init_2,\ Inv_{p1} \wedge Inv_{p2},\ Inv_{r2},\ Pro_p, Pro_r \rangle$, với $(Init_1 \biguplus Init_2)(x)$ được định nghĩa là:

- $Init_1(x) = Init_2(x)$ nếu $x \in dom(Init_1) \cap dom(Init_2)$,
- $Init_1(x)$ nếu $x \in dom(Init_1) \setminus dom(Init_2)$, và
- $Init_2(x)$ nếu $x \in dom(Init_2) \setminus dom(Init_1)$.

Chúng ta bây giờ chỉ ra làm thế nào giao thức Pro_p, Pro_r được tính toán từ Pro_{pi} và $Pro_{ri}, i = 1, 2$ đối với hợp đồng $Ctr_1 >> Ctr_2$.

- Đầu tiên, hợp đồng phức hợp $Ctr_1 >> Ctr_2$ cũng nên cho phép các phương thức trong mỗi thành phần riêng được sử dụng theo cách nguyên bản của nó. Vì thế, $Pro_{p1} \cup Pro_{p2}$ nên được bao gồm trong Pro_p.
- Một phương thức m được cung cấp bởi Ctr_2 mà không yêu cầu bởi Ctr_1 có thể được sử dụng một cách song song với phương thức trong Ctr_1, và một phương thức m được cung cấp bởi Ctr_2 mà được yêu cầu bởi Ctr_1 phải là phụ thuộc với các phương thức trong Md_{p1} (Ở đây, chúng ta giả sử rằng phần cứng cho phép thực hiện song song). Vì thế $D_p = D_{p1} \cup D_{p2} \cup (Md_{p1} \times (Md_{p2} \setminus Md_{r1}) \cup (Md_{p2} \setminus Md_{r1} \times Md_{p2})$, và $Pro_{p1} Pro_{p2} \cup Pro_{p2} Pro_{p1}$ nên được bao gồm trong Pro_p, ở đây, phép ghép nối Vết là nền tảng cho quan hệ phụ thuộc D_p.

Khi $Ctr_1 >> Ctr_2$ được định nghĩa, chúng ta nói rằng Ctr_1 là có khả năng gắn với Ctr_2. Chú ý rằng khi kết nối hai hợp đồng theo cách này, thành phần kết quả trả về có thể không là một sự làm mịn của Ctr_1. Lý do là nó có thể yêu cầu một số thứ từ môi trường mà Ctr_1 không có. Vì thế, nếu chúng ta giả thiết

rằng nếu không yêu cầu thêm bất kỳ thứ gì từ môi trường, nó có thể là một sự làm mịn của Ctr_1.

Định lý 4.2 *Cho Ctr_1 tương thích với Ctr_2, nếu Ctr_2 đóng (tức là $I_{r2} = \emptyset$) thì $Ctr_1 \sqsubseteq (Ctr_1 >> Ctr_2)$.*

Chứng minh: Từ giả thiết Ctr_1 tương thích với Ctr_2 nên ta có thể cắm Ctr_1 vào Ctr_2, tức là ta có một hợp đồng Ctr được thành lập bằng cách cắm Ctr_1 vào Ctr_2, $Ctr = Ctr_1 >> Ctr_2$. Như vậy Ctr sẽ có tất cả các phương thức cung cấp của Ctr_1 và Ctr_2. Mặt khác do Ctr_2 đóng (tức là $I_{r2} = \emptyset$) nên Ctr_2 không yêu cầu gì ngoài môi trường mà Ctr_1 không có. Do đó Ctr cung cấp tất cả những gì mà Ctr_1 có, không yêu cầu dịch vụ gì thêm từ môi trường so với Ctr_1. Theo định nghĩa phép làm mịn thì rõ ràng hợp đồng Ctr không những có đầy đủ các dịch vụ mà Ctr_1 có mà còn cung cấp nhiều hơn do được mở rộng với Ctr_2, tức là hợp đồng Ctr được làm mịn từ hợp đồng Ctr_1 hay $Ctr_1 \sqsubseteq (Ctr_1 >> Ctr_2)$. Từ đó ta có điều phải chứng minh. ■

Định nghĩa này có thể được tổng quát hóa cho trường hợp tổng quát $I_{r1} \nsubseteq I_{p2}$, và định lý vẫn đúng.

4.4.4 Thành phần

Bây giờ chúng ta muốn hình thức hóa khái niệm về thành phần. Tiếp theo, với thành phần thụ động, chúng ta muốn nói tới một thành phần mà cung cấp các dịch vụ theo yêu cầu, thành phần chủ động (active component) là thành phần mà tương tác với kích thích từ môi trường, tức là các dịch vụ của nó được kích hoạt bởi các sự kiện. Bằng trực giác, một thành phần thụ động là một thực thi của một hợp đồng sử dụng các dịch vụ từ các thành phần thụ động khác thông qua hợp đồng của chúng. Để đơn giản trong biểu diễn, chúng ta sử dụng một kiến trúc đơn giản với khởi đầu client/server và đồng bộ.

Định nghĩa 4.6 (Thành phần thụ động) *Một thành phần thụ động là một bộ $Comp = \langle Ctr, Mcode \rangle$, với $Comp$ được định danh là tên của thành phần bao gồm:*

- *Một hợp đồng: $Ctr = \langle (Ip, Ir),\ Mspec,\ Init, Invp,\ Invr, Pro_p,\ Pro_r \rangle$, và*
- *Một thực thi của các phương thức được cung cấp I_p: $Mcode$ gắn với mỗi phương thức op trong Mdp một thiết kế được xây dựng từ các toán tử cơ bản(như được hiểu hoặc định nghĩa trong [40]) và các phương thức trong*

I_r sao cho phép chiếu của các Vết thời gian của thiết kế trên (Md_r, D_r) được bao gồm trong Pro_r. Điều kiện sau cần được thỏa mãn bởi Mcode: $(Mspec(op) \sqsubseteq Mcode(op))$, và Inv_p được duy trì bởi bất kỳ hành động nào được sử dụng trong Mcode (consistency condition).

Khi đó hợp đồng Ctr được gọi là được thực thi bởi Comp.

Định nghĩa này yêu cầu một thành phần phải đúng theo thiết kế của nó, tức là thực thi của các phương thức của nó phải đúng theo đặc tả. Tính đúng của nó có thể được kiểm chứng riêng biệt từ môi trường.

Cho $Comp_i = \langle Ctr_i, Mcode_i \rangle$, $i = 1, 2$ là các thành phần sao cho $Ctr_1 >> Ctr_2$, $Mcode'_1$ được lấy từ $Mcode_1$ bằng cách thay thế mỗi hành động của $op \in (Md_{r1} \cap Md_{p2})$ bằng $Mcode_2(op)$ trong phạm vi của $Mcode_1$. Dễ dàng chỉ ra rằng toán tử cắm được định nghĩa trên hợp đồng được thực thi trên các thành phần.

Hệ quả 4.1 *$\langle Ctr_1 >> Ctr_2, Mcode'_1 \cup Mcode_2|_{Md_2 \setminus Md_l} \rangle$ là một thành phần.*

Thường khi chuyển qua phần hoạt động của hệ thống thành phần, chúng ta đưa ra một ví dụ để chỉ ra các giao thức tương tranh có yếu tố thời gian giúp chúng ta thế nào.

Ví dụ 4.1 *Cho A là một thành phần với hai phương thức trong giao diện của nó thỏa mãn các điều kiện sau:*

- *phương thức* $m1(x, y) : x \geq 0 \vdash y^2 = x, 2 \leq dur \leq 3$
- *phương thức* $m2(x, y) : x \geq 0 \vdash y^5 = x, 5 \leq dur \leq 7$.

$D = \{(ml, ml), (m2, m2)\}$ *là một quan hệ phụ thuộc, và giao thức là một ngôn ngữ Vết thời gian được sinh bởi* $\{m1m2)\}^*$ *theo D và J được định nghĩa như sau:* $J(m1) = [2, 3], J(m2) = [5, 7]$. *D cho phép m1 và m2 có thể được thực hiện song song với nhau. Cho B là một thành phần cung cấp một phương thức* $m(x, z) : x \geq 0 \vdash z = sqrt(x) + \sqrt{50} + \sqrt[5]{x}, 3 \leq dur \leq 10$.

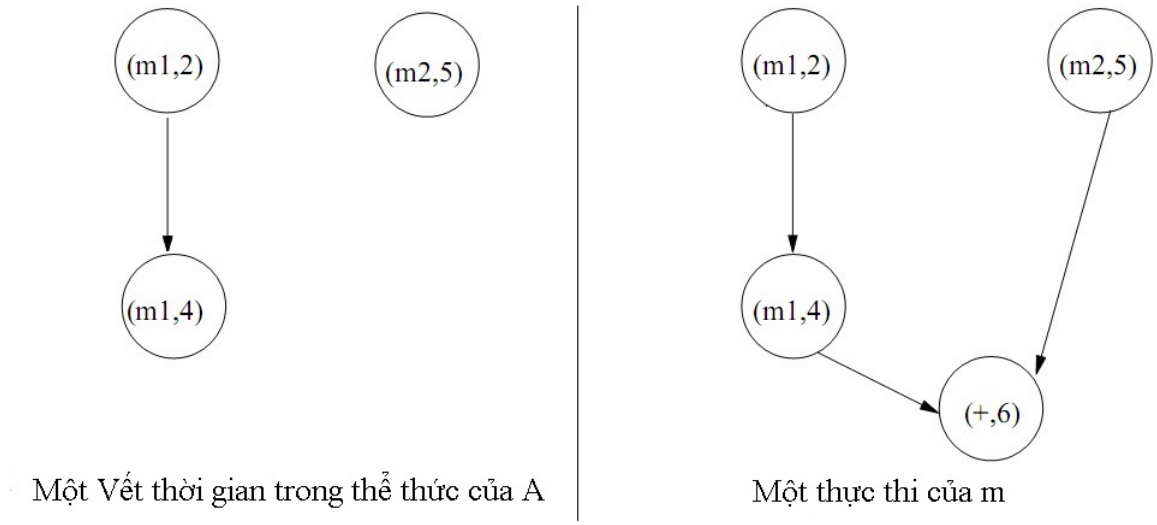
Liệu m có thể được thực thi sử dụng các dịch vụ từ A và giả sử rằng phép cộng không mất thời gian? Điều này hoàn toàn thực hiện được nhờ vào tính song song. Mã cho phương thức m là: $((A.m1(x, y1); A.m1(50, y3))\|A.m2(x, y2)); y1 + y3 + y2$. *Với cách thực hiện này, ràng buộc điều kiện* $5 \leq dur \leq 7$ *được đảm bảo! Sơ đồ thứ tự và thời gian cho thực thi này được chỉ ra trong Hình 4.2.*

Phần hoạt động của CBS theo kiến trúc trong phần hai bao gồm một số tiến trình và giao diện yêu cầu mà có thể cắm vào phần thụ động. Một tiến trình được mô tả bởi một chương trình mà sử dụng các dịch vụ từ phần thụ động để tương tác với các sự kiện từ môi trường của hệ thống. Các sự kiện không được điều khiển bởi hệ thống. Do đó, các sự kiện quan tâm từ môi trường và các dịch vụ hệ thống với các đặc tả của chúng và các giao thức giao diện hình thành hợp đồng giữa hệ thống với môi trường.

Định nghĩa 4.7 *Một giao diện hệ thống là một bộ* $SI = (E, Fd, SMd_p)$, *với* SMd_p *là một tập hữu hạn các phương thức,* Fd *là một tập các đặc trưng, và* E *là tập hữu hạn các sự kiện.*

Định nghĩa 4.8 *Một hợp đồng hệ thống là một bộ* $SysCtr = \langle SI, SMSpec, Inv, Behav\rangle$, *với*

- $SI = (E, Fd, SMd_p)$ *là một giao diện hệ thống,*
- Inv *là thuộc tính mô tả giá trị của các đặc trưng trong* Fd,
- $SMSpec$ *là hàm đặc tả phương thức mà gắn mỗi phương thức* $op(in, out)$ *trong* SMd_p *với một thiết kế* $\langle \alpha, FP\rangle$, *với* $(\alpha\ (in \cup out)) \subseteq Fd$, *và*
- $Behav$ *là mô tả hành vi mở rộng mà là tập con hữu hạn của* $\{(e, m) \mid e \in E$ *and* $m \in SMdp\}^*$. *Mỗi thành phần của Behav được gọi là một đặc tả tiến trình.*



Hình 4.2: Thời gian và thứ tự của $Mcode(m)$ và phép chiếu của nó trên các phương thức của B

Chúng ta muốn tránh giới thiệu của một lôgic trong phần này và sử dụng các thành phần của *Behav* như là một đặc tả hệ thống. Một cách phi hình thức cho một chuỗi trong *Behav* là nếu môi trường hệ thống đưa ra chuỗi các sự kiện xảy ra trong đó thì hệ thống cũng đưa ra chuỗi các dịch vụ (methods) được đặc tả bởi thứ tự đó. Các thành phần của *Behav* mô tả các luồng chương trình chạy song song.

Định nghĩa 4.9 (Thành phần chủ động) *Một thành phần chủ động* $ActComp = \langle Ctr, SysCtr, Mcode \rangle$*, bao gồm:*

- *Một hợp đồng* $Ctr = \langle (I_p, I_r), Mspec, Init, Inv_p, Inv_r, Pro_r \rangle$ *với giao diện cung cấp rống,* $I_p = (\emptyset, \emptyset)$*,*
- *Một hợp đồng hệ thống* $SysCtr = \langle SI,\ SMSpec, Inv,\ Behav \rangle$*, và*
- *Một tiến trình thực thi Mcode gắn mỗi phương thức op trong SMdp với một thiết kế được xây dựng từ một tập các phép toán cơ bản và phương thức trong* I_r*. Mcode thỏa mãn:*
  - *Cho Mcode được mở rộng thành một đồng cấu* $Mcode : Behav \Rightarrow (Commands \cup I_r)^*$ *trong cách thông thường. THì* $wtot(||_{\omega \in Bahav} Mcode(\omega) \mid _{I_r})$ *là một ngôn ngữ Vết có trong ngôn ngữ Vết Pro, với wtot tương ứng với bảng chữ cái phụ thuộc* (Md_r, D_r) *mà ngôn ngữ Vết thời gian là Pro, và* $||$ *là shuffle operator được định nghĩa thông thường trong ngôn ngữ hình thức, và*

– $(SMspec(op) \sqsubseteq Mcode(op))$ *for all* $op \in SMd_p$.

Một hệ thống trong mô hình thành phần của trong luận án là một thành phần chủ động được cắm vào một thành phần bị động đóng.

Định nghĩa 4.10 (Hệ thống) *Một hệ thống là một cặp của một thành phần chủ động* $ActComp = \langle Ctr, SysCtr, Mcode\rangle$ *và một thành phần bị động đóng* $Comp = \langle Ctr', Mcode'\rangle$ *sao cho* $Ctr >> Ctr'$ *được xác định theo định nghĩa phép cắm trên.*

Vì thế, một hệ thống là một hệ thống đóng tức là không yêu cầu bất kỳ các dịch vụ nào từ môi trường, và cung cấp các dịch vụ của nó cho môi trường như là các phản ứng của nó với các sự kiện kích hoạt từ môi trường. Đặc tả của hệ thống là hợp đồng hệ thống ký hiệu là $SysCtr$. Hai hệ thống là tương đương nếu chúng có cùng đặc tả, tức là chúng có hợp đồng hệ thống tương đương nhau. Định lý sau đây mô tả đặc trưng quan trọng nhất của lập trình theo thành phần.

Định lý 4.3 *Cho* $S = (ActComp, Comp')$ *là một hệ thống được hình thành bởi một thành phần chủ động* $ActComp = \langle Ctr,\ SysCtr, Mcode\rangle$ *và thành phần bị động* $Comp' = \langle Ctr',\ Mcode'\rangle$*, cho* $Comp'' = \langle Ctr'', Mcode''\rangle$ *là một thành phần bị động sao cho* $Ctr' \sqsubseteq Ctr''$*, thì* $(ActComp, Comp'')$ *cũng là một hệ thống tương đương với* S*.*

Chứng minh: Theo định nghĩa về làm mịn hợp đồng, nếu $Ctr' \sqsubseteq Ctr''$ thì hợp đồng Ctr'' yêu cầu ít dịch vụ hơn Ctr' và cung cấp nhiều dịch vụ hơn Ctr'. Do đó thành phần bị động $Comp''$ sẽ cung cấp nhiều dịch vụ ra ngoài hơn thành phần bị động $Comp'$. Theo định nghĩa về hệ thống thì hệ thống $S'' = (ActComp, Comp'')$ và hệ thống $S = (ActComp, Comp')$ sẽ có chung thành phần chủ động $ActComp$, tuy nhiên hệ thống S'' cung cấp nhiều dịch vụ hơn do thành phần bị động $Comp''$ cung cấp nhiều dịch vụ hơn thành phần bị động $Comp'$. Như vậy thì hệ thống S'' có đầy đủ các dịch vụ như hệ thống S thậm chí là tốt hơn do S'' có đầy đủ các đặc tả như S. Từ đó ta có S'' là tương đương với S. ■

4.5 Kết luận

Chúng ta đã thấy trong phần trước rằng các Vết thời gian Mazurkiewicz có sức mạnh và lại đơn giản về mặt hình thức cho đặc tả các giao thức tương tranh có yếu tố thời gian. Khi giới hạn trong ngôn ngữ Vết chính quy thời gian, từ

các lý thuyết ban đầu của ngôn ngữ Vết [75, 30], chúng ta đã gắn mỗi giao thức một ô-tô-mát bất đồng bộ mà có thể được sử dụng để kiểm tra tính nhất quán (consistency) của hệ thống.

Khi có một khai báo giao thức trong hợp đồng của một giao diện của thành phần, chúng ta cần kiểm tra xem môi trường của thành phần có phù hợp với đặc tả giao thức không. Điều này có thể được thực hiện bằng phân tích tĩnh hoặc kiểm chứng động lúc thực thi. Phân tích tĩnh là cần thiết để quyết định các quan hệ tương tranh của hệ thống. Khi kiểm tra lúc thực thi, cho trước một quan hệ tương tranh là nhất quán với đặc tả, hành vi của hệ thống được kiểm tra với các giao thức bằng mô phỏng nó với giao thức gắn với các ô-tô-mát bất đồng bộ. Một kỹ thuật tối ưu cho việc này là cần thiết và một thuật toán hiệu quả cho một trường hợp đặc biệt của hệ thống thành phần được đề xuất trong [41] nhưng chưa hỗ trợ đa luồng.

Trong thời gian gần đây, một số thành viên trong nhóm nghiên cứu đã thực hiện một số nghiên cứu chặt chẽ về các giao thức tương tác thành phần [74]. Tuy nhiên, công việc này tập trung vào các giao thức tuần tự, không thời gian để cho kết quả tốt hơn. So sánh với vấn đề này, nghiên cứu của luận án trong phần này tập trung vào vấn đề thời gian và tương tranh cho các giao thức, và có thể xem như là một sự mở rộng của công việc trước đó. Luận án đã đề xuất một mở rộng thời gian cho cho Vết Mazurkiewicz, và chứng minh được rằng sự mở rộng này là tiện lợi cho đặc tả và kiểm chứng các giao thức tương tranh có yếu tố thời gian mặc dù độ phức tạp tính toán vẫn là hàm mũ do ô-tô-mát đoán nhận ngôn ngữ vết thời gian là tương đương với một TA (chương 3). Từ kinh nghiệm của chúng tôi với một số thử nghiệm trong lĩnh vực này, chúng tôi có thể nói rằng, Vết thời gian Mazurkiewicz cần ít thay đổi nhất cho việc chuyển đổi mô hình từ không thời gian sang tương tranh có yếu tố thời gian. Các kết quả trong chương này được công bố trong công trình khoa học số [2] của danh mục công trình đã công bố.

Chương 5

Phương pháp đặc tả các thành phần trong hệ tương tranh có ràng buộc thời gian theo nguyên lý thiết kế dựa trên giao diện

Trong chương này, luận án đề xuất một phương pháp hỗ trợ cho việc đặc tả và kiểm chứng các hệ thống tương tranh có ràng buộc thời gian dựa trên thành phần. Ý tưởng cơ bản của phương pháp là áp dụng lý thuyết Vết thời gian vào mô hình thiết kế dựa trên giao diện để mở rộng mô hình này có thể đặc tả được các hệ thống dựa trên thành phần có tính tương tranh và ràng buộc thời gian. Để thực hiện nghiên cứu này, luận án giả sử ngôn ngữ của các thành phần hệ thống là các từ thời gian có thêm tính tương tranh. Như vậy, ngôn ngữ này sẽ được đặc tả một cách tiện lợi nhất bằng các Vết thời gian. Luận án đề xuất một ô-tô-mát đoán nhận các Vết thời gian này và gọi là ô-tô-mát giao diện tương tranh có ràng buộc thời gian. Các kết quả nghiên cứu này cũng đưa ra phương pháp ghép nối, làm mịn và chứng minh việc đảm bảo hai đặc tính cơ bản không thể thiếu của lý thuyết ô-tô-mát giao diện là thiết kế gia tăng và thực thi độc lập. Những kết quả này là điều kiện quan trọng để đặc tả chính xác hành vi của các hệ thống tương tranh có ràng buộc thời gian và chứng minh tính đúng đắn của các hệ thống này.

5.1 Giới thiệu

Trong hướng tiếp cận phát triển hệ thống dựa trên giao diện [3], giao diện đóng vai trò trung tâm trong việc thiết kế hệ thống dựa trên thành phần của hệ thống phần mềm và phần cứng. Ưu điểm của phương pháp thiết kế này là tính trực quan và dựa trên nền tảng lý thuyết toán học về ô-tô-mát nên độ tin cậy của hệ thống là cao. Theo phương pháp này, mỗi thành phần của hệ thống được đặc tả thông qua giao diện của chúng bao gồm đặc tả đầu vào, đầu ra và không quan tâm nhiều tới hoạt động bên trong của mỗi thành phần. Từ các đặc tả này, ta có thể xây dựng mô hình đặc tả chính xác các hành vi hệ thống cũng như các ràng buộc yêu cầu mà hệ thống cần đáp ứng. Do đó, ta có thể áp dụng các phương pháp kiểm chứng nhằm chứng minh tính đúng đắn của hệ thống thông qua các công cụ kiểm chứng tự động hoặc bán tự động. Chúng ta nói rằng hai hoặc nhiều thành phần là tương thích nếu chúng hoạt động cùng nhau đúng cách. Thiết kế giao diện tốt là dựa trên hai nguyên tắc. Đầu tiên, một giao diện nên đưa ra đầy đủ thông tin về một thành phần để làm cho nó có thể dự đoán nếu hai hoặc nhiều thành phần là tương thích bằng cách xem các đặc tả giao diện của chúng. Thứ hai, một giao diện không nên biểu diễn thêm thông tin về một thành phần so với yêu cầu của nguyên tắc đầu tiên.

Giao diện biểu diễn các thông tin về sự tương tác giữa các thành phần có thể được đoán nhận một cách tự nhiên trong một ngôn ngữ của một ô-tô-mát. Với ngôn ngữ này, có hai yêu cầu về cần phải đạt được. Thứ nhất, một ngôn ngữ giao diện nên hỗ trợ thiết kế gia tăng và thứ hai là hỗ trợ thực thi độc lập. Các phương pháp phát triển hệ thống dựa trên giao diện, việc quan trọng không thể thiếu là phải xác minh rằng cả hai yêu cầu trên phải được đáp ứng.

Hiện nay, đã có một số các phương pháp phát triển hệ thống dựa trên cách tiếp cận này bằng việc mở rộng mô hình kinh điển của Thomas A. Henzinger mà điển hình là nghiên cứu trong [46] đã đưa ra một số phương pháp cho việc đặc tả và kiểm chứng các hệ thống có yếu tố thời gian dựa trên giao diện. Các phương pháp này thường sử dụng lý thuyết ô-tô-mát thời gian [4] và các đặc tả tương tự để đặc tả các thành phần hệ thống. Theo đó, mỗi thành phần hệ thống được đặc tả bởi một ô-tô-mát thời gian. Tuy nhiên, việc đặc tả thêm tính tương tranh trên ô-tô-mát thời gian là rất khó khăn và phức tạp.

Với các ưu điểm của thiết kế dựa trên giao diện, bài toán cần giải quyết trong chương này là đưa ra phương pháp thiết kế dựa trên giao diện như giới thiệu phía trên cho các hệ thống tương tranh có ràng buộc thời gian, tức là xây dựng một lý thuyết về ô-tô-mát giao diện có khả năng đặc tả các ràng buộc thời gian

và tính tương tranh.

Một ví dụ dễ thấy là chúng ta có thể sử dụng mô hình thiết kế dựa trên giao diện cho bài toán về dịch vụ chuyển thông điệp (message-transmission service). Hệ thống gồm hai thành phần (biểu diễn bởi hai ô-tô-mát giao diện). Thành phần thứ nhất (User) biểu diễn hành động của người dùng và thành phần thứ hai (Comp) biểu diễn các hoạt động của việc gửi thông điệp. Hệ thống được mô hình bằng cách ghép nối hai thành phần này với nhau [28]. Vấn đề ở đây là khi ta cần tăng chất lượng dịch vụ (tăng tốc độ đáp ứng) bằng cách thực thi một số hành động một cách song song và thêm các ràng buộc thời gian cho các hành động để đảm bảo hệ thống đáp ứng đúng theo đặc tả người dùng. Như vậy thì mô hình này chưa thể hiện được và ta cần mở rộng để giải quyết được các bài toán tương tự.

Để giải quyết bài toán này, luận án đề xuất mở rộng về thời gian của lý thuyết ô-tô-mát giao diện nhằm mục đích hỗ trợ đặc tả các hệ thống tương tranh có yếu tố thời gian. Theo đó, luận án giả sử mỗi hành động của hệ thống không những có đặc tả về chức năng mà còn đặc tả phi chức năng bao gồm đặc tả về có yếu tố thời gian thực thi yếu nhất. Như vậy, một giao thức thành phần hệ thống sẽ phải thỏa mãn ba ràng buộc[1]: Ràng buộc thứ tự, ràng buộc thời gian, và ràng buộc trên các lời gọi song song từ các luồng khác nhau, tức là ngôn ngữ giao diện của mỗi thành phần là các Vết thời gian. Do đó, ý tưởng của phương pháp đề xuất là sử dụng ô-tô-mát mà ngôn ngữ đoán nhận là các vết thời gian làm ô-tô-mát giao diện (hạn chế trên tập hành động đầu vào và đầu ra). Các nghiên cứu đề xuất trong chương này cũng đưa ra phương pháp ghép nối, làm mịn và chứng minh việc đảm bảo hai đặc tính cơ bản của lý thuyết ô-tô-mát giao diện là thiết kế gia tăng và thực thi độc lập. Các kết quả này là điều kiện quan trọng để đảm bảo và chứng minh tính đúng đắn của các hệ thống tương tranh có ràng buộc thời gian được thiết kế dựa trên thành phần.

5.2 Ô-tô-mát giao diện tương tranh có ràng buộc thời gian

Từ các kết quả có được trong chương 3, một Vết thời gian có ba đặc trưng của các giao thức tương tranh có yếu tố thời gian được đề cập trong phần giới thiệu. Hơn nữa, bảng chữ cái khoảng có thể biểu diễn ràng buộc trên sự thực thi của các hành động của hệ thống. Chúng ta có thể sử dụng ô-tô-mát hữu hạn bất đồng bộ $\mathcal{A}$ để biểu diễn một ngôn ngữ Vết L, và cùng với một hàm gán khoảng thời gian $J : \Sigma \longrightarrow intv$ để biểu diễn một ngôn ngữ Vết thời gian. Nhận thấy

[1]Đã giới thiệu trong phần đầu của luận án

rằng các Vết Mazurkiewicz là một biểu diễn hiệu quả cho ràng buộc đồng thời, luận án đã sử dụng một tập các Vết thời gian mà có biểu diễn hữu hạn như là một đặc tả cho các thể thức trong giao diện của thành phần. Phần này, luận án tiếp tục áp dụng Vết thời gian và cá kết quả để đưa ra phương pháp đặc tả các hệ tương tranh có yếu tố thời gian dựa trên giao diện.

Một hệ thống tương tranh có yếu tố thời gian được ghép nối từ các thành phần tương tranh có yếu tố thời gian. Như vậy, một thành phần tương tranh có yếu tố thời gian cũng là một hệ tương tranh có yếu tố thời gian nhưng phải đảm bảo một số chức năng cụ thể nào đó và thỏa mãn ba ràng buộc về giao thức được giới thiệu trong phần 5.1. Do đó, các ADA với ngôn ngữ đoán nhận là các Vết thời gian là phù hợp cho việc đặc tả các thành phần trong các hệ thống này. Trong phần này, luận án sẽ áp dụng các ô-tô-mát này với một số hạn chế về tập hành động vào đặc tả các giao diện thành phần. Luận án cũng đưa ra các khái niệm về điều kiện ghép nối, tính tương thích, làm mịn các thành phần và phương pháp ghép nối các thành phần.

5.2.1 Định nghĩa

Như đã trình bày trong phần 5.1, ngôn ngữ giao diện của một thành phần tương tranh có yếu tố thời gian là một Vết thời gian. Để biểu diễn hữu hạn ngôn ngữ này, luận án sử dụng ADA. Như vậy, một thể thức giao diện thành phần có thể được đặc tả như là một ADA. Để đơn giản nhưng không mất tính tổng quát, chỉ các ADA đơn định được sử dụng trong mô hình giao diện thành phần. Một ô-tô-mát giao diện tương tranh có yếu tố thời gian là một ADA trên các hành động đầu vào và đầu ra của hệ thống. Một cách hình thức ta đưa ra định nghĩa sau.

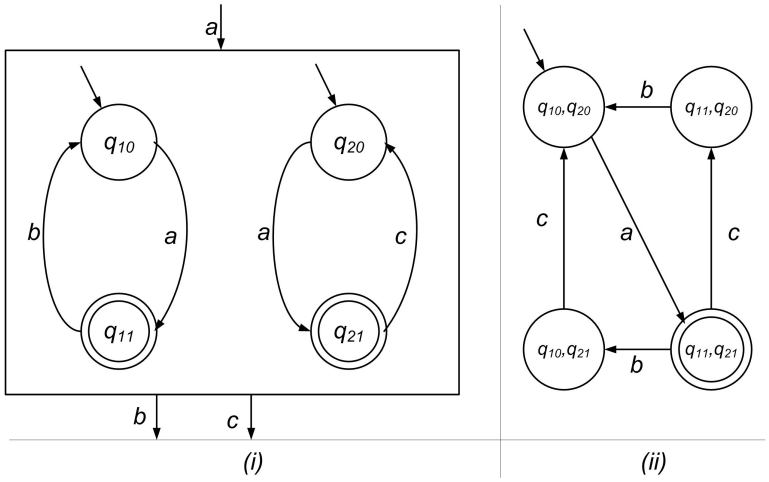
Định nghĩa 5.1 (Ô-tô-mát giao diện tương tranh có yếu tố thời gian) *Một ô-tô-mát giao diện tương tranh có yếu tố thời gian (Timed Concurrent Interface Automata - TCIA) là một bộ ba* $P = \langle I, O, (\mathcal{A}^D, J)\rangle$*, với* I *là tập các hành động đầu vào,* O *là tập các hành động đầu ra và* $(\mathcal{A}^D, J) = (\{S_i\}_{i\in Proc_\mathcal{A}}, \longrightarrow_\mathcal{A}, S_{in}, \{F_i\}_{i\in Proc_\mathcal{A}}, \{G_i\}_{i\in Proc_\mathcal{A}})$ *là một ô-tô-mát khoảng đơn định bất đồng bộ trên bảng chữ cái* $\Sigma = I \cup O$*.*

Để cho đơn giản, từ bây giờ tất cả các ô-tô-mát bất đồng bộ được xem là đơn định nếu không nói gì thêm và ta ký hiệu $\mathcal{A}$ thay cho $\mathcal{A}^D$. Cho một TCIA P, ngôn ngữ giao diện của P được định nghĩa như ngôn ngữ của ADA $(\mathcal{A}, J)$, tức là ngôn ngữ giao diện của P là ngôn ngữ Vết thời gian được đoán nhận bởi ADA $(\mathcal{A}, J)$.

Ký hiệu tập bước chuyển trạng thái của TCIA P là $tran(P) = tran(\mathcal{A}, J) = \longrightarrow_{\mathcal{A}}$. Nếu $a \in I$ $(a \in O)$ thì $(s, a, s') \in tran(\mathcal{A}, J)$ được gọi là bước chuyển đầu vào (ra) của TCIA P và được ký hiệu là $tran^I(P)$ $(tran^O(P))$.

Một hành động $a \in \Sigma$ được gọi là kích hoạt tại trạng thái $s \in S_{Proc}$ nếu $(s, a, s') \in tran(P)$. Ký hiệu $\Sigma(s) = I(s) \cup O(s)$ là tập tất cả các hành động được kích hoạt tại s. Các hành động đầu vào trong $I \setminus I(s)$ là tập các đầu vào không chấp nhận tại trạng thái s.

Ví dụ 5.1 *Cho một TCIA* $P = (I_P, O_P, (\mathcal{A}_P, J_P))$ *với* $I_P = \{a\}, O_P = \{b, c\}, Proc_P = \{1, 2\}, \widetilde{\Sigma} = \{\{a, b\}, \{a, c\}\}, S_P^{in} = \{q_{10}, q_{20}\}, F_P = G_P = \{q_{11}, q_{21}\}, S_P = \{q_{10}, q_{20}, q_{11}, q_{21}\}, J_P(a) = [1, 2], J_P(b) = [2, 3], J_P(c) = [1, 3]$*, và quan hệ độc lập* $I_P = \{(b, c), (c, b)\}$ *do b và c thuộc 2 tiến trình khác nhau. Ngôn ngữ Vết thời gian của P là tập các Vết thời gian thỏa mãn Vết khoảng* $T = \{(a(bca)^\omega, J_P)\}$*. Biểu diễn của P được chỉ ra trong Hình 5.1, các dịch chuyển trạng thái được chỉ ra trong Bảng 5.1.*



Hình 5.1: Một TCIA P với $J_P(a) = [1, 2], J_P(b) = [2, 3], J_P(c) = [1, 3]$ (i) và đồ thị chuyển trạng thái tương ứng (ii)

Bảng 5.1: Bảng chuyển của TCIA P trong ví dụ 5.1

$$
\begin{array}{lcl}
(q_{10}, q_{20}) & \xrightarrow{a} & (q_{11}, q_{21}) \\
(q_{10}, q_{21}) & \xrightarrow{c} & (q_{10}, q_{20}) \\
(q_{11}, q_{20}) & \xrightarrow{b} & (q_{10}, q_{20}) \\
(q_{11}, q_{21}) & \xrightarrow{b} & (q_{10}, q_{21}) \\
(q_{11}, q_{21}) & \xrightarrow{c} & (q_{11}, q_{20})
\end{array}
$$

5.2.2 Khả năng ghép nối và Tích song song các TCIA

Theo lý thuyết về ô-tô-mát giao diện, chúng ta cần tìm cách ghép nối các TCIA lại để được thành phần lớn hơn với nhiều chức năng hơn. Như vậy, cho hai TCIA P và Q, xây dựng cách ghép nối giữa chúng là chỉ ra ô-tô-mát đặc tả ô-tô-mát ghép nối của P và Q.

Hai TCIA P và Q là tương thích nếu phép ghép nối là khác rỗng. Trước tiên, chúng ta định nghĩa khả năng ghép nối của hai TCIA.

Định nghĩa 5.2 (Khả năng ghép nối của hai TCIA) *Hai TCIA P và Q là được gọi là có khả năng ghép nối nếu các điều kiện sau được thỏa mãn:*

1. *Tập các hành động đầu vào I_P và I_Q và tập các hành động đầu ra O_P và O_Q là không giao nhau,*
2. *Các hàm gán thời gian J_P và J_Q trên các hành động chung $shared(P,Q) = \Sigma_P \cap \Sigma_Q$ là không mâu thuẫn, tức là với mọi hành động $a \in shared(P,Q), J_P(a) \cap J_Q(a) \neq \emptyset$, và*
3. *Tập các tiến trình $Proc_P$ and $Proc_Q$ là không giao nhau.*

Tương tự lý thuyết về ô-tô-mát giao diện, luận án gọi một trạng thái là không hợp lệ nếu ở đó một ô-tô-mát có thể sản sinh đầu ra là một đầu vào không được chấp nhận của một ô-tô-mát khác.

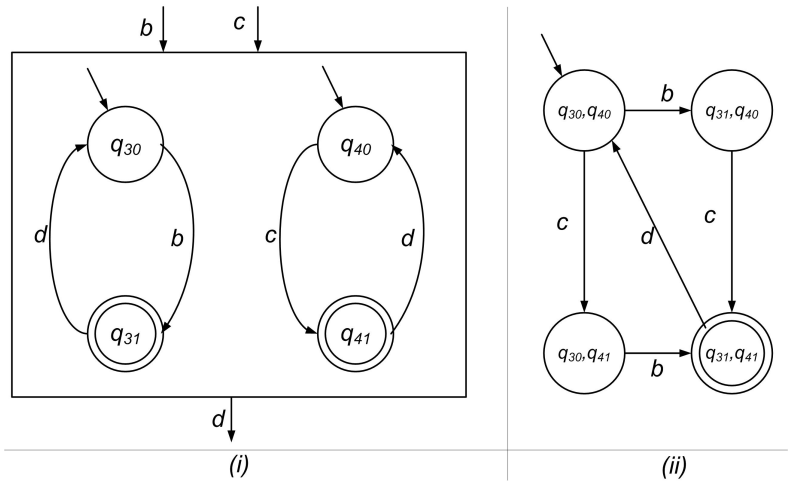
Định nghĩa 5.3 (Trạng thái không hợp lệ) *Cho trước hai TCIA có khả năng ghép nối P và Q, tập các trạng thái không hợp lệ của chúng ký hiệu là $Illegal(P,Q) \subseteq S^P \times S^Q$ với $S^P = \{S_i\}_{i \in Proc_P}$ và $S^Q = \{S_i\}_{i \in Proc_Q}$ là tập các trạng thái tương ứng của P và Q được định nghĩa như sau:*

$$Illegal(P,Q) = \{(s^P, s^Q) \in S^P \times S^Q \text{ sao cho tồn tại } a \in shared(P,Q) \text{ với } a \in O_P(s^P) \wedge a \notin I_Q(s^Q) \text{ hoặc } a \in O_Q(s^Q) \wedge a \notin I_P(s^P)\}.$$

Ví dụ 5.2 *Cho một TCIA $Q = (I_Q, O_Q, (\mathcal{A}_Q, J_Q))$ với $I_Q = \{b,c\}, O_P = \{d\}, Proc_Q = \{3,4\}, \widetilde{\Sigma}_Q = \{\{b,d\},\{c,d\}\}, S_Q^{in} = \{q_{30}, q_{40}\}, F_Q = G_Q = \{q_{31}, q_{41}\}, S_P = \{q_{30}, q_{40}, q_{31}, q_{41}\}, J_Q(d) = [2,4], J_Q(b) = [2,3], J_Q(c) = [1,3]$ và quan hệ độc lập $I_Q = \{(b,c),(c,b)\}$ do b và c thuộc 2 tiến trình khác nhau. Ngôn ngữ Vết thời gian của Q là tập các Vết thời gian thỏa mãn Vết khoảng $T = \{((bcd)^\omega, J_Q)\}$. Biểu diễn của Q được chỉ ra trong Hình 5.2.*

Bảng 5.2: Bảng chuyển của TCIA Q trong ví dụ 5.2

$$
\begin{array}{lcl}
(q_{30}, q_{40}) & \xrightarrow{c} & (q_{30}, q_{41}) \\
(q_{30}, q_{40}) & \xrightarrow{b} & (q_{31}, q_{40}) \\
(q_{30}, q_{41}) & \xrightarrow{b} & (q_{31}, q_{41}) \\
(q_{31}, q_{40}) & \xrightarrow{c} & (q_{31}, q_{41}) \\
(q_{31}, q_{41}) & \xrightarrow{d} & (q_{30}, q_{40})
\end{array}
$$



Hình 5.2: TCIA Q với $J_Q(b) = [2, 3], J_Q(c) = [1, 3], J_Q(d) = [2, 4]$ (i) và đồ thị chuyển trạng thái tương ứng (ii) là tương thích với TCIA P trong Ví dụ 5.1

Khi có đầy đủ các khái niệm cần thiết, chúng ta đưa ra định nghĩa về phép ghép song song giữa hai TCIA.

Định nghĩa 5.4 (Tích song song của hai TCIA) *Tích song song của hai TCIA có khả năng ghép nối với nhau* $P = (I_P, O_P, (\mathcal{A}_\mathcal{P}, J_P))$ *và* $Q = (I_Q, O_Q, (\mathcal{A}_\mathcal{Q}, J_Q))$ *ký hiệu là* $P \parallel Q$*, là một TCIA* $F = (I_F, O_F, (\mathcal{A}_\mathcal{F}, J_F))$ *với các thành phần được định nghĩa như sau:*

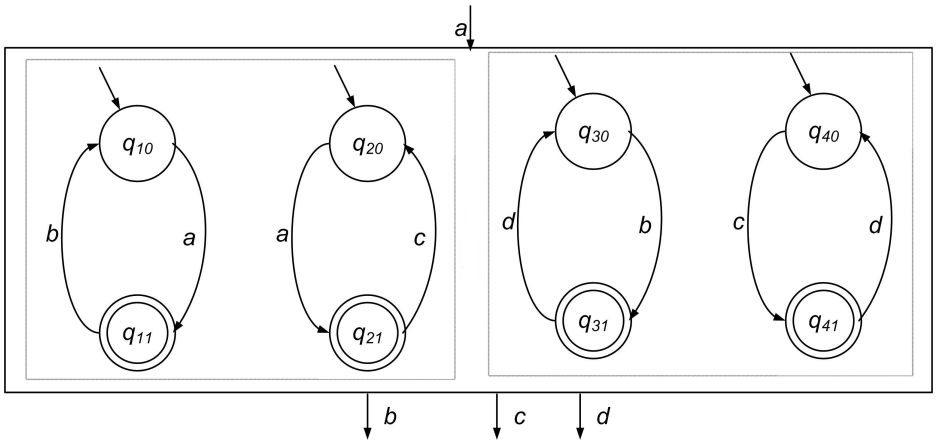
- $O_F = O_P \cup O_Q$ *và* $I_F = (I_P \cup I_Q) \setminus O_F$,
- $Proc_F = Proc_P \cup Proc_Q$, $\tilde{\Sigma}_F = \tilde{\Sigma}_P \cup \tilde{\Sigma}_Q$, $J_F = J_P \uplus J_Q = \{J_P(a) \cup J_P(b) | a \in \Sigma_P, b \in \Sigma_Q, a \neq b\} \cup \{J_P(a) \cap J_Q(a) | a \in \Sigma_P \cap \Sigma_Q\}$, *và*
- $\mathcal{A}_F = (\{S_i\}_{i \in Proc_P} \times \{S_i\}_{i \in Proc_Q}, \{\xrightarrow{a}_F\}_{a \in \Sigma_F}, \{(s^P, s^Q) | s^P \in S^P_{in} \wedge s^Q \in S^Q_{in}\}, \{F_i\}_{i \in Proc_P} \times \{F_i\}_{i \in Proc_Q}, \{G_i\}_{i \in Proc_P} \times \{G_i\}_{i \in Proc_Q})$, *với* $\xrightarrow{a}_F$ $= \{(s^P_a, s^Q_a) \xrightarrow{a}_F (s'^P_a, s'^Q_a) | s^P_a \xrightarrow{a}_P s'^P_a \wedge s^Q_a \xrightarrow{a}_Q s'^Q_a \wedge a \in (\Sigma_P \cap \Sigma_Q)\}$

$$\cup \{(s_a^P, s_a^Q) \xrightarrow{a}_F (s_a'^P, s_a^Q) | s_a^B \xrightarrow{a}_P s_a'^P \wedge a \in (\Sigma_P \setminus \Sigma_Q)\}$$
$$\cup \{(s_a^P, s_a^Q) \xrightarrow{a}_F (s_a^P, s_a'^Q) | s_a^Q \xrightarrow{a}_Q s_a'^Q \wedge a \in (\Sigma_Q \setminus \Sigma_P)\}.$$

Theo định nghĩa trên, vì phép ghép hai TCIA cũng là một TCIA nên bài toán kiểm tra tính rỗng của phép ghép là quyết định được (theo định lý 3.1).

Ví dụ 5.3 *Phép ghép song song của hai TCIA P và Q được chỉ ra trong Hình 5.3. Ngôn ngữ Vết thời gian của phép ghép này là tập các Vết thời gian thỏa mãn Vết khoảng $T = \{(a(bcad)^\omega, J\}$ với $J = J_P \cup J_Q$ và quan hệ độc lập $I = \{(a, d), (b, c)\}$.*

Vấn đề là trong ô-tô-mát tích ghép sẽ có thể tồn tại các trạng thái không được phép (do việc cung cấp các dịch vụ cho nhau của phép ghép bị thiếu hoặc mâu thuẫn) hoặc các trạng thái không đến được[2]. Chúng ta sẽ cần đưa ra giải pháp để loại bỏ các trạng thái này. Giải pháp chung cho vấn đề này là cung cấp một môi trường cho thành phần hoạt động. Như vậy, một môi trường của một thành phần cũng là một TCIA mà cung cấp đầy đủ cho các điều kiện cho các thành phần khác ghép nối.



Hình 5.3: Kết quả phép tích song song giữa P và Q trong Hình 5.1 và 5.2

.

Định nghĩa 5.5 (Môi trường) *Một môi trường cho một TCIA P là một TCIA E thỏa mãn các điều kiện sau:*

- *E và P là có khả năng ghép nối,*
- *E khác rỗng,*

[2]Là các trạng thái mà ở đó không thể đi tới được một trong các trạng thái khác (không dẫn được tới trạng thái kết thúc)

- $I_E = O_P$, *và*
- $Illegal(P, E) = \emptyset$.

Ví dụ 5.4 *TCIA Q trong Hình 5.2 là môi trường của TCIA P trong Hình 5.1 vì Q và P thỏa mãn các điều kiện về khả năng ghép nối, Q không rỗng, $I_Q = O_P$ và rõ ràng theo định nghĩa trạng thái bất hợp pháp thì $Illegal(P, Q) = \emptyset$.*

Tương tự như trường hợp không có thời gian, luận án đưa ra định nghĩa về tính tương thích của hai TCIA là có khả năng ghép nối và ghép được (tức là không tạo thành ô-tô-mát rỗng).

Định nghĩa 5.6 (Tương thích) *Hai TCIA P và Q là tương thích nếu và chỉ nếu chúng có khả năng ghép nối và phép ghép nối song song của chúng là không rỗng.*

Từ định nghĩa về phép ghép song song, cho trước các TCIA P, Q và R có khả năng ghép nối, tính chất kết hợp của phép ghép vẫn được thỏa mãn. Tính chất này thể hiện qua định lý dưới đây.

Định lý 5.1 $(P \parallel Q) \parallel R = P \parallel (Q \parallel R)$.

Chứng minh: Giả sử $P = (I_P, O_P, (\mathcal{A}_\mathcal{P}, J_P))$, $Q = (I_Q, O_Q, (\mathcal{A}_\mathcal{Q}, J_Q))$ và $QR = (I_R, O_R, (\mathcal{A}_\mathcal{R}, J_R))$, ta phải chứng minh $(P \parallel Q) \parallel R = P \parallel (Q \parallel R)$. Thật vậy, từ định nghĩa về phép tích ghép, ta có hai TCIA ghép nối với nhau cho ra một TCIA, tổng quát cho trường hợp ghép nối ba TCIA, ta có $(P \parallel Q) \parallel R$, là một TCIA $F = (I_F, O_F, (\mathcal{A}_\mathcal{F}, J_F))$ với các thành phần được định nghĩa như sau:

- $O_F = (O_P \cup O_Q) \cup O_R$ và $I_F = ((I_P \cup I_Q) \cup I_R) \setminus O_F$,
- $Proc_F = (Proc_P \cup Proc_Q) \cup Proc_R$, $\tilde{\Sigma}_F = (\tilde{\Sigma}_P \cup \tilde{\Sigma}_Q) \cup \tilde{\Sigma}_R$, $J_F = (J_P \uplus J_Q) \uplus J_R = \{(J_P(a) \cup J_P(b)) \cup J_R(c) | a \in \Sigma_P, b \in \Sigma_Q, c \in \Sigma_R, a \neq b \neq c\} \cup \{(J_P(a) \cap J_Q(a)) \cap J_R(a) | a \in (\Sigma_P \cap \Sigma_Q) \cap \Sigma_R\}$, và
- $\mathcal{A}_F = (\{S_i\}_{i \in Proc_P} \times \{S_i\}_{i \in Proc_Q} \times \{S_i\}_{i \in Proc_R}, \{\xrightarrow{a}_F\}_{a \in \Sigma_F}, \{(s^P, s^Q, , s^R) | s^P \in S^P_{in} \wedge s^Q \in S^Q_{in} \wedge s^R \in S^R_{in}\}, \{F_i\}_{i \in Proc_P} \times \{F_i\}_{i \in Proc_Q} \times \{F_i\}_{i \in Proc_R}, \{G_i\}_{i \in Proc_P} \times \{G_i\}_{i \in Proc_Q} \times \{G_i\}_{i \in Proc_R})$, với $\xrightarrow{a}_F$
  $= \{(s^P_a, s^Q_a, s^R_a) \xrightarrow{a}_F (s'^P_a, s'^Q_a, s'^R_a) | s^P_a \xrightarrow{a}_P s'^P_a \wedge s^Q_a \xrightarrow{a}_Q s'^Q_a \wedge s^R_a \xrightarrow{a}_R s'^R_a \wedge a \in (\Sigma_P \cap \Sigma_Q \cap \Sigma_R)\}$
  $\cup \{(s^P_a, s^Q_a, s^R_a) \xrightarrow{a}_F (s'^P_a, s^Q_a, s^R_a) | s^P_a \xrightarrow{a}_P s'^P_a \wedge a \in (\Sigma_P \setminus \Sigma_Q \setminus \Sigma_R)\}$
  $\cup \{(s^P_a, s^Q_a, s^R_a) \xrightarrow{a}_F (s^P_a, s'^Q_a, s^R_a) | s^Q_a \xrightarrow{a}_Q s'^Q_a \wedge a \in (\Sigma_Q \setminus \Sigma_P \setminus \Sigma_R)\}$
  $\cup \{(s^P_a, s^Q_a, s^R_a) \xrightarrow{a}_F (s^P_a, s^Q_a, s'^R_a) | s^R_a \xrightarrow{a}_R s'^R_a \wedge a \in (\Sigma_R \setminus \Sigma_P \setminus \Sigma_Q)\}$.

Mặt khác $P \parallel (Q \parallel R)$ là một TCIA $H = (I_H, O_H, (\mathcal{A}_\mathcal{H}, J_H))$ với

- $O_H = O_P \cup (O_Q \cup O_R)$ và $I_H = (I_P \cup (I_Q \cup I_R) \setminus O_H$,
- $Proc_F = Proc_P \cup (Proc_Q \cup Proc_R)$, $\tilde{\Sigma}_H = \tilde{\Sigma}_P \cup (\tilde{\Sigma}_Q \cup \tilde{\Sigma}_R)$, $J_H = J_P \uplus (J_Q \uplus J_R) = \{J_P(a) \cup (J_P(b) \cup J_R(c)) | a \in \Sigma_P, b \in \Sigma_Q, c \in \Sigma_R, a \neq b \neq c\} \cup \{J_P(a) \cap (J_Q(a) \cap J_R(a)) | a \in (\Sigma_P \cap \Sigma_Q) \cap \Sigma_R\}$, và
- $\mathcal{A}_H = (\{S_i\}_{i \in Proc_P} \times \{S_i\}_{i \in Proc_Q} \times \{S_i\}_{i \in Proc_R}, \{\xrightarrow{a}_H\}_{a \in \Sigma_F}, \{(s^P, s^Q, , s^R) | s^P \in S^P_{in} \wedge s^Q \in S^Q_{in} \wedge s^R \in S^R_{in}\}, \{F_i\}_{i \in Proc_P} \times \{F_i\}_{i \in Proc_Q} \times \{F_i\}_{i \in Proc_R}, \{G_i\}_{i \in Proc_P} \times \{G_i\}_{i \in Proc_Q} \times \{G_i\}_{i \in Proc_R})$, với $\xrightarrow{a}_H$
  $= \{(s^P_a, s^Q_a, s^R_a) \xrightarrow{a}_H (s'^P_a, s'^Q_a, s'^R_a) | s^P_a \xrightarrow{a}_P s'^P_a \wedge s^Q_a \xrightarrow{a}_Q s'^Q_a \wedge s^R_a \xrightarrow{a}_R s'^R_a \wedge a \in (\Sigma_P \cap \Sigma_Q \cap \Sigma_R)\}$
  $\cup \{(s^P_a, s^Q_a, s^R_a) \xrightarrow{a}_H (s'^P_a, s^Q_a, s^R_a) | s^P_a \xrightarrow{a}_P s'^P_a \wedge a \in (\Sigma_P \setminus \Sigma_Q \setminus \Sigma_R)\}$
  $\cup \{(s^P_a, s^Q_a, s^R_a) \xrightarrow{a}_H (s'^P_a, s^Q_a, s^R_a) | s^Q_a \xrightarrow{a}_Q s'^Q_a \wedge a \in (\Sigma_Q \setminus \Sigma_P \setminus \Sigma_R)\}$
  $\cup \{(s^P_a, s^Q_a, s^R_a) \xrightarrow{a}_H (s'^P_a, s^Q_a, s^R_a) | s^R_a \xrightarrow{a}_R s'^R_a \wedge a \in (\Sigma_R \setminus \Sigma_P \setminus \Sigma_Q)\}$.

Do các phép toán hợp $\cup$ là có tính giao hoán, kết hợp nên ta có TCIA $H = F$. Từ đó ta có điều phải chứng minh. ■

Một TCIA biểu diễn giả thiết về môi trường và các đảm bảo của thành phần. Các dịch chuyển đầu ra cho ta giả thiết về các hành động đầu ra phải được chấp nhận. Các hành động đầu vào không chấp nhận của một trạng thái ngăn cấm môi trường cung cấp các đầu ra đó. Các đảm bảo về chuỗi các hành động và lựa chọn các hành động. Phép ghép nối song song kết nối các giả thiết môi trường với các đảm bảo thành phần. Có một môi trường đặc biệt nhưng đơn giản cho tất cả các phép ghép nối song song của P và Q đó là nó chấp nhận tất cả các đầu ra của $P \parallel Q$ và không có hành động đầu vào.

5.2.3 Làm mịn các thành phần

Khi phát triển các thành phần, chúng ta quan tâm tới việc sửa đổi một thành phần có sẵn nào đó để được thành phần mới tốt hơn theo khía cạnh cung cấp nhiều dịch vụ hơn và yêu cầu ít dịch vụ được cung cấp từ môi trường. Do đó, tại mỗi trạng thái của ô-tô-mát được dùng để làm mịn, ta có khái niệm làm mịn trạng thái với ý nghĩa trạng thái mới sẽ cung cấp đầu ra nhiều hơn và yêu cầu ít hơn đầu vào. Ta có khái niệm hình thức như sau.

Định nghĩa 5.7 (Làm mịn trạng thái) *Cho hai TCIA P và Q, một quan hệ làm mịn trạng thái từ P vào Q là một quan hệ hai ngôi $\preceq \in S^P \times S^Q$ sao cho với mọi $(u, v) \in S^P \times S^Q, v \preceq u$ các điều kiện sau là thỏa mãn:*

- $I_P(u) \subseteq I_Q(v)$,
- $O_Q(v) \subseteq O_P(u)$,
- *hàm ánh xạ thời gian J là thống nhất trên các hành động chung của u và v, tức là* $\forall a \in (I_P(u) \cap I_Q(v)) \cup (O_P(u) \cap O_Q(v)), J_Q(a) \subseteq J_P(a)$, *và*
- $\forall a \in O_Q(v) \cup I_P(u)$ *và với mọi trạng thái* $u'|(u, a, u') \in tran(P)$ *có tồn tại một trạng thái* $v'|(v, a, v') \in tran(Q)$ *sao cho* $v' \succeq u'$.

Từ khái niệm trên, luận án đưa ra định nghĩa về làm mịn giao diện. Định nghĩa này tương tự như trường hợp không có thời gian.

Định nghĩa 5.8 (Làm mịn) *Một TCIA Q được gọi là được làm mịn từ TCIA P và được ký hiệu là $Q \succeq P$ nếu:*

- $I_Q \subseteq I_P$,
- $O_P \subseteq O_Q$, *và*
- *tồn tại một quan hệ làm mịn trạng thái từ P vào Q mà $u \in S^P_{in}, v \in S^Q_{in}$ ta có $v \succeq u$.*

Như vậy, một TCIA luôn làm mịn chính nó, tức là cho một TCIA P, ta luôn có $P \succeq P$. Hiển nhiên ta có kết quả sau được suy luận trực tiếp từ định nghĩa phép làm mịn.

Định lý 5.2 *Cho ba TCIA P, Q, R, nếu $P \succeq Q$ và $Q \succeq R$ thì $P \succeq R$.*

Cuối cùng, ta chỉ ra một kết quả quan trọng trong việc đảm bảo khả năng thực thi độc lập trong lý thuyết về ngôn ngữ giao diện cho mô hình của chúng ta. Kết quả này chỉ ra mối quan hệ giữa phép ghép song song và phép làm mịn các TCIA. Tức là nếu hai TCIA là làm mịn nhau thì phép ghép song song của chúng với cùng một TCIA khác cũng là làm mịn nhau. Điều này đảm bảo rằng một TCIA được ghép nối vào hệ thống thì phép làm mịn của nó cũng ghép được vào hệ thống.

Định lý 5.3 *Cho ba TCIA P, Q và R sao cho Q và R là có khả năng ghép nối và $I_Q \cap O_R \subseteq I_P \cap O_R$, nếu P và R là tương thích và $Q \succeq P$ thì Q và R là tương thích và $Q \parallel R \succeq P \parallel R$.*

Chứng minh: Ta chỉ ra ô-tô-mát ghép $Q \parallel R$ và $\succeq P \parallel R$ thỏa mãn ba điều kiện của phép làm mịn theo định nghĩa 5.8.

1. Ta chỉ ra $I_{Q\parallel R} \subseteq I_{P\parallel R}$. Thật vậy, do $Q \succeq P$ nên ta có $I_Q \subseteq I_P$, do đó $I_{Q\parallel R} \subseteq I_{P\parallel R}$.

2. Ta chỉ ra $O_{Q\|R} \subseteq O_{P\|R}$. Do $Q \succeq P$ nên ta có $O_Q \subseteq O_P$, do đó $O_{Q\|R} \subseteq O_{P\|R}$.

3. Tồn tại một quan hệ làm mịn trạng thái từ $P \parallel R$ vào $Q \parallel R$ mà $u \in S_{in}^{P\|R}, v \in S_{in}^{Q\|R}$ ta có $v \succeq u$. Thật vậy, do $Q \succeq P$ nên tồn tại một quan hệ làm mịn từ trạng thái đầu vào của P vào trạng thái đầu vào của Q. Mặt khác trạng thái đầu vào của $P \parallel R$ gồm trạng thái đầu vào của P và R, trạng thái đầu vào của $Q \parallel R$ gồm trạng thái đầu vào của Q và R và $R \succeq R$. Do vậy theo định nghĩa về làm mịn trạng thái thì ta có $v \succeq u$ với $u \in S_{in}^{P\|R}, v \in S_{in}^{Q\|R}$.

■

Như vậy, trong thiết kế dựa trên giao diện, một hệ thống sẽ được đặc tả thông qua tích các giao diện của các thành phần của nó. Đối với mô hình này, hệ tương tranh có yếu tố thời gian sẽ được đặc tả bởi một giao diện là một ô-tô-mát giao diện tương tranh có yếu tố thời gian. Ô-tô-mát này là kết quả của tập các ghép nối song song của các giao diện thành phần tạo nên hệ thống.

5.3 Các nghiên cứu liên quan

Hiện nay đã có nhiều ngôn ngữ hỗ trợ đặc tả giao diện cho các thành phần phần mềm. Tuy nhiên các ngôn ngữ này hoặc không hình thức hoặc quá nặng nề cho việc mô hình hệ thống. Do đó, các nghiên cứu gần đây đã tập trung đề xuất các phương pháp hình thức để thiết kế hệ thống dựa trên giao diện nhưng phải đơn giản và tiện dụng. Trong phần này, luận án sẽ đưa ra một số nghiên cứu liên quan tới đặc tả hệ thống tương tranh có yếu tố thời gian.

Luca de Alfaro và Thomas A. Henzinger [28] là những người đặt nền tảng và đưa ra các nghiên cứu về lý thuyết ô-tô-mát giao diện dùng để đặc tả các giao diện thành phần. Trong phương pháp này, mỗi thành phần được coi như là một "hộp đen", người sử dụng chỉ biết các thông tin về điều kiện đầu vào và đầu ra của thành phần (và được gọi là "giao diện"). Như vậy, mỗi thành phần được đặc trưng bởi giao diện của nó và được đặc tả bởi một ô-tô-mát giao diện và hệ thống sẽ được ghép nối bởi các thành phần thông qua các phép ghép được định nghĩa. Các nghiên cứu trong này quan tâm tới các thể thức tương tác giữa các thành phần, cách ghép nối, điều kiện ghép nối các thành phần. Các kết quả của phương pháp đã được ứng dụng và phát triển trong các nghiên cứu mở rộng và ứng dụng thực tế. Tuy nhiên, phương pháp này không hỗ trợ đặc tả thành phần có yếu tố thời gian do các ô-tô-mát giao diện được đề xuất không hỗ trợ đặc tả thuộc tính thời gian.

Một số phương pháp khác được đề xuất như trong [33], Laurent Doyen đã bổ sung vào lý thuyết giao diện [3] để các thành phần có thể được sử dụng lại. Stavros Tripakis [69] đề xuất lý thuyết giao diện và tập trung vào quan hệ đầu vào, đầu ra của các biến trong giao diện đó. Tuy nhiên, các nghiên cứu này vẫn không hỗ trợ đặc tả có yếu tố thời gian và tính tương tranh.

Các nghiên cứu trong các công trình liên quan đến luận án đã đưa ra lý thuyết Vết thời gian như là một sự mở rộng về thời gian của lý thuyết Vết. Các tác giả đã chỉ ra lợi ích của việc biểu diễn này cho việc đặc tả các hệ thống tương tranh có yếu tố thời gian, trong đó đề cập nhiều vào khả năng biểu diễn linh hoạt, ngắn gọn và đặc biệt là phương pháp này có biểu diễn hữu hạn bằng ô-tô-mát và lôgic thời gian tuyến tính trên Vết. Từ đó, các tác giả chỉ rõ Vết thời gian là một biểu diễn hữu ích hơn các dạng biểu diễn dựa trên ngôn ngữ thời gian và ô-tô-mát thời gian. Sau đó, các tác giả cũng ứng dụng lý thuyết Vết thời gian vào trong lý thuyết rCOS[74] cho việc đặc tả các hệ thống dựa trên thành phần để có thể đặc tả thêm các tính tương tranh và ràng buộc thời gian. Tuy nhiên, các kết quả trên không quan tâm tới các ràng buộc đầu vào/đầu ra, một kiểu thiết kế mới cho các hệ thống mà người thiết kế chỉ quan tâm tới các ràng buộc bên ngoài thành phần mà bên trong coi như là "hộp đen" của hệ thống. Một ứng dụng khác của lý thuyết Vết thời gian cho đặc tả các hệ thống phân tán có yếu tố thời gian cũng được đề xuất là rất hiệu quả nhưng cũng chưa đề cập tới khái niệm về ô-tô-mát giao diện và do đó cũng không thể ứng dụng trực tiếp cho đặc tả hệ thống theo hướng tiếp cận dựa trên giao diện thành phần.

Gần đây, có nhiều nghiên cứu và phương pháp được đề xuất sử dụng các ô-tô-mát giao diện cho việc đặc tả và kiểm chứng các hệ thống dựa trên thành phần. Trong [5], Angelov đã giới thiệu một phương pháp đặc tả hệ thống nhúng sử dụng ô-tô-mát giao diện. Chouali và các cộng sự [21] đề xuất một phương pháp hình thức kiểm chứng về ghép nối các thành phần. Cao và nhóm nghiên cứu trong [19] đã mở rộng ô-tô-mát giao diện với các ký hiệu z với mục đích đưa ra một tiếp cận đặc tả cho việc ghép nối các giao diện với ngôn ngữ Z (Z language) [2]. Các ứng dụng khác sử dụng ô-tô-mát giao diện được đề xuất bởi Li và cộng sự trong [52]. Trong nghiên cứu này, các tác giả biểu diễn một một hình ghép nối dịch vụ web mới và thuật toán kiểm chứng dựa trên ô-tô-mát giao diện và mở rộng để hỗ trợ mô tả các dịch vụ web. Aarts [1] đã nghiên cứu và đưa ra một nền tảng cho phương pháp học phụ thuộc lịch sử để loại bỏ phương pháp cũ sử dụng lý thuyết ô-tô-mát giao diện. Lüttgen trong [54] giới thiệu một phương pháp sửa đổi mô hình ô-tô-mát giao diện để xử lý các tính toán bên trong và nghiên cứu một mô hình để tạo ra ô-tô-mát giao diện với sự tương

thích tối ưu. Ô-tô-mát này gọi là ô-tô-mát giao diện làm giàu (richer interface automata).

Gần đây, các kết quả trong [46] đã đưa ra phương pháp đặc tả hệ thống có yếu tố thời gian dựa trên giao diện. Nghiên cứu đã đưa ra khái niệm ô-tô-mát giao diện có yếu tố thời gian dựa trên lý thuyết về ô-tô-mát thời gian để tiếp cận đặc tả các hệ có yếu tố thời gian dựa trên giao diện. Phương pháp này cũng cung cấp các cách thức làm mịn, kiểm tra tính đúng đắn và phương pháp ghép nối. Tuy nhiên, phương pháp này vẫn còn hạn chế là các nghiên cứu tập trung đặc tả và kiểm chứng ràng buộc thời gian mà chưa thể hiện tính tương tranh trên từng thành phần. D.V.Hung và Truong Hoang trong nghiên cứu [27] đã đề xuất một phương pháp là một mở rộng của lý thuyết giao diện quan hệ của Stavros Tripakis và nhóm nghiên cứu. Nghiên cứu này đưa ra khái niệm về giao diện thời gian thực, đây là giao diện với các ràng buộc thời gian trên các hành động đầu vào và đầu ra. Tuy nhiên, mô hình này không hỗ trợ đặc tả các tính chất tương tranh của hệ thống.

Như vậy, mỗi phương pháp đề xuất chỉ giải quyết được một lớp nhỏ các bài toán hoặc có độ phức tạp cao, chưa phù hợp và khó áp dụng cho đặc tả và kiểm chứng các hệ thống tương tranh có ràng buộc thời gian. Cách tiếp cận dựa trên ô-tô-mát giao diện là hợp lý bởi tính trực quan, dễ sử dụng, độ tin cậy cao cho các hệ thống là đối tượng nghiên cứu của luận án.

5.4 Kết luận

Chương này, luận án đã đề xuất một phương pháp đặc tả các hệ thống tương tranh có yếu tố thời gian. Ý tưởng của phương pháp là dựa trên lý thuyết về thiết kế dựa trên giao diện kết hợp với lý thuyết Vết thời gian. Khi đó mỗi thành phần hệ thống được đặc tả bởi một ô-tô-mát giao diện tương tranh có yếu tố thời gian. Ô-tô-mát này là ô-tô-mát khoảng bất đồng bộ mà có tập hành động được phân chia thành hai tập khác rỗng rời nhau là tập hành động đầu vào và tập hành động đầu ra. Trong nghiên cứu này, luận án định nghĩa và chứng minh các vấn đề về khả năng tương thích, phép ghép nối, môi trường và làm mịn thành phần. Phương pháp cũng đảm bảo được hai đặc tính cơ bản của lý thuyết thiết kế dựa trên giao diện là thiết kế gia tăng và thực thi độc lập. Các yếu tố trên đảm bảo hệ thống luôn có khả năng mở rộng bằng việc ghép nối các thành phần tương thích với nhau mà không ảnh hưởng bởi thứ tự kết hợp ghép nối (tính kết hợp) và các thành phần được làm mịn có thể ghép vào hệ thống

cũ để tạo thành hệ thống mới với nhiều dịch vụ đầu ra và ít yêu cầu ràng buộc đầu vào.

Tuy vậy, phương pháp vẫn còn hạn chế là chưa hỗ trợ đặc tả thuộc tính hệ thống bằng lôgic, chưa có công cụ để kiểm chứng tự động và về mặt lý thuyết, độ phức tạp tính toán là tương đương với độ phức tạp tính toán của TA. Trong thời gian tới, chúng tôi sẽ nghiên cứu hoàn thiện phương pháp bằng việc đề xuất phương pháp đặc tả thuộc tính hệ thống dựa trên lôgic bằng việc mở rộng về thời gian của lôgic thời gian tuyến tính trên Vết. Như vậy, chúng ta để có thể kiểm chứng mô hình hệ thống ngay tại bước thiết kế bằng việc kết hợp với các công cụ đặc tả có sẵn để áp dụng phương pháp đặc tả này nhằm đánh giá tính hiệu quả về ứng dụng của phương pháp. Các kết quả nghiên cứu trong chương này được công bố trong công trình nghiên cứu số [4] của danh mục công trình khoa học đã công bố của luận án.

Chương 6

Mô hình đặc tả và kiểm chứng các hệ phân tán có ràng buộc thời gian dựa trên hệ dịch chuyển phân tán

Giới thiệu

Với sự phát triển của khoa học công nghệ, các hệ thống phần mềm ngày càng phức tạp hơn. Hiện nay, các hệ tính toán song song hay các hệ thống phân tán được phát triển rất mạnh do các ứng dụng thực tế của chúng trong nhiều lĩnh vực khác nhau như các hệ điều khiển máy bay, tàu hỏa, các hệ thống phần mềm trong ngân hàng hay các hệ thống nhúng,v.v. Do đó, vấn đề đảm bảo độ tin cậy hệ thống phân tán cũng đang được quan tâm nghiên cứu một cách rộng rãi. Các kết quả trong [47, 50, 55] đã đưa ra một số phương pháp hình thức để đặc tả và kiểm chứng các hệ thống này. Tuy nhiên, các phương pháp này tập trung nghiên cứu các hệ thống không có ràng buộc thời gian trong khi các ràng buộc có yếu tố thời gian là một thuộc tính không thể thiếu của các hệ thống hiện nay. Một số các nghiên cứu được tổng hợp trong [66] đã cho một cái nhìn bao quát về các nghiên cứu hệ thống phân tán, trong đó các nghiên cứu về bài toán kiểm chứng hệ phân tán có sử dụng Vết Mazurkiewicz để giảm không gian trạng thái. Các nghiên cứu đã chỉ ra rằng, đối với các hệ phân tán (hay các hệ tương tranh), Vết Mazurkiewicz (gọi tắt là Vết) [30, 75, 14] là một nền tảng cơ bản phù hợp cho đặc tả các hệ thống này. Tuy nhiên, lý thuyết Vết chưa hỗ trợ ràng buộc thời gian trong đặc tả.

Bài toán cần giải quyết trong chương này là đưa ra kỹ thuật đặc tả và kiểm chứng các hệ phân tán có yếu tố thời gian. Đây là những hệ thống không chỉ có các ràng buộc về tương tranh mà còn có ràng buộc về thời gian, về lời gọi giữa các phương thức. Phương pháp đưa ra phải đơn giản, và có thể kiểm chứng được hệ thống nhằm đảm bảo độ tin cậy của các hệ thống này.

Xuất phát từ những phân tích trên, để giải quyết bài toán đề xuất trong chương này, luận án giới thiệu một phương pháp được nghiên cứu và đề xuất dựa trên việc mở rộng hệ dịch chuyển phân tán của Martin Leucker [50]. Martin Leucker đã sử dụng mô hình này để mô hình hóa (đặc tả và kiểm chứng) các mạch phần cứng đồng bộ. Thực tế, các hành động của một hệ thống cần một khoảng thời gian nhất định để thực hiện và cung cấp dịch vụ. Do đó chúng không nên được gọi quá nhiều trên một đơn vị thời gian. Để đảm bảo về chất lượng dịch vụ, các hệ thống này cần có đặc tả ràng buộc về thời gian thực thi cho các hành động là thời gian tối thiểu và tối đa một hành động cần để thực hiện. Thời gian này dễ dàng được biểu diễn thông qua việc sử dụng lý thuyết Vết thời gian. Như vậy, với việc mở rộng về thời gian của hệ dịch chuyển phân tán, luận án đã đưa ra một phương pháp hiệu quả trong đặc tả hệ thống phân tán có yếu tố thời gian như là minh chứng cho ứng dụng của Vết thời gian.

Với kỹ thuật này, hành vi hệ thống được đặc tả bằng một hệ dịch chuyển phân tán có yếu tố thời gian, các thuộc tính hệ thống được đặc tả bằng một lôgic thời gian trên các cấu hình Foata. Các hệ thống được mô hình bởi hệ dịch chuyển phân tán sẽ dễ dàng được mô hình bằng hệ dịch chuyển phân tán thời gian khi các hệ thống này có thêm các ràng buộc thời gian cho các hành động của nó. Một ví dụ cho trường hợp này là các mạch phần cứng đồng bộ có ràng buộc thời gian [35] được D'Souza và các cộng sự đề xuất sử dụng mạng các ô-tô-mát khoảng để mô hình hóa. Phương pháp này khá phức tạp, khó sử dụng và đặc biệt khó thể hiện đặc tính tương tranh. Ta có thể sử dụng phương pháp đề xuất để mô hình hệ thống này đơn giản hơn và ít thao tác chuyển đổi hơn. Các kết quả nghiên cứu cũng chỉ ra rằng, bài toán kiểm chứng hệ thống là quyết định được mặc dù độ phức tạp tính toán không phải là đa thức.

6.1 Hệ phân tán có ràng buộc thời gian

Trong chương này, luận án đưa ra một mô hình hình thức cho đặc tả các hệ phân tán có yếu tố thời gian. Mô hình này là mở rộng về thời gian của hệ dịch chuyển phân tán của Martin Leucker được đề cập trong nghiên cứu [50]. Gọi $Proc = \{1, 2, ..., m\}$ là tập các chỉ số các tiến trình, và bảng chữ cái phụ thuộc là

(Σ, D) sao cho $\Sigma = \bigcup_{j \in Proc} \Sigma_j$ và $D = \{(a,b) | \exists j.\{a,b\} \subseteq \Sigma_j\}$. Điều này có nghĩa là hai hành động trong Σ là độc lập nếu chúng thuộc hai tiến trình khác nhau trong hệ thống. Ta kí hiệu $\widetilde{\Sigma}\widehat{=}(\Sigma_i)_{i \in Proc}$ là bảng chữ cái phân tán. Cho $\omega \in \Sigma^\omega, proj_i(\omega)$ ký hiệu phép chiếu của từ ω trên Σ_i; và $pref(\omega)$ ký hiệu tập tất cả các tiền tố của ω. Cho tập $\{S_j\}_j \in Proc$, và $a \in \Sigma$ với $loc(a) = \{j_i, j_2, ..., j_k\}$.

Định nghĩa 6.1 (Hệ dịch chuyển phân tán) *Một hệ dịch chuyển phân tán (DTS - Distributed Transition Systems) trên bảng chữ cái phân tán $\widetilde{\Sigma}$ là một cấu trúc $\mathcal{A} = (\{S_i\}_{i \in Proc}, \longrightarrow, S_{in})$ trong đó:*

- *Mỗi S_i là một tập trạng thái địa phương thứ i ký hiệu là i-local,*
- *Đặt $\overline{S} = \prod_{i \in Proc} S_i$ là tập các trạng thái toàn cục và $S_{in} \subseteq \overline{S}$ là tập các trạng thái khởi đầu, và*
- *Đặt $States = \prod_{i \in Proc}(S_i \cup \{*\})$. Quan hệ dịch chuyển $\longrightarrow \subseteq States \times \Sigma \times States$ thỏa mãn điều kiện sau:*

$$Nếu\ (\overline{q}, a, \overline{q'}) \in \longrightarrow\ thì\ \overline{q}[i] = \overline{q'}[i] = *,\ với\ i \in Proc \setminus loc(a).$$

Ký hiệu * dùng để diễn tả các thành phần của một trạng thái toàn cục không ảnh hưởng bởi quan hệ dịch chuyển. Hành vi của $\mathcal{A}$ được xác định thông qua thực thi của chúng. Có hai định nghĩa về thực thi là thực thi đồng bộ và thực thi bất đồng bộ tùy theo mục đích sử dụng hệ thống.

Định nghĩa 6.2 (Thực thi đồng bộ) *Một thực thi đồng bộ ρ của một DTS là một chuỗi vô hạn $\overline{q_1}A_1\overline{q_2}...$ của các trạng thái và tập các hành động độc lập thỏa mãn các điều kiện sau:*

- $\overline{q_1} \in S_{in}$,
- *đối với mỗi $j \geq 1$ và với mỗi $a \in A_j, (\overline{q_j}|_{loc(a)}, a, \overline{q_{j+1}}|_{loc(a)}) \in \longrightarrow$ và $\overline{q_j}|_P) = \overline{q_{j+1}}|_P)$ với $P = Proc \setminus \bigcup_{a \in A_j}(loc(a))$. Vì thế dịch chuyển là sự thực thi song song của các hành động đồng thời theo quan hệ dịch chuyển, và*
- *Hơn thế, A_j là tập lớn nhất theo khía cạnh không có tập lớn hơn thỏa mãn quan hệ chuyển. Điều này đảm bảo tất cả các thành phần có thể thực hiện một dịch chuyển trong các bước thực thi.*

Định nghĩa 6.3 (Thực thi bất đồng bộ) *Một thực thi bất đồng bộ ρ của một DTS là một chuỗi vô hạn $\overline{q_1}a_1\overline{q_2}...$ của các trạng thái $\overline{q_j}$ và hành động a_j thỏa mãn các điều kiện sau:*

- $\overline{q_1} \in S_{in}$, *và*
- *Đối với mỗi* $j \geq 1$ *ta có* $(\overline{q_j}|_{loc(a_j)}, a_j, \overline{q_{j+1}}|_{loc(a_j)}) \in \longrightarrow$ *và* $\overline{q_j}|_{Proc \setminus loc(a_j)}) = \overline{q_{j+1}}|_{Proc \setminus loc(a_j)})$.

Khi hệ DTS được xác định hành vi là thực thi đồng bộ, chúng ta gọi là DTS đồng bộ và tương ứng khi thực thi là bất đồng bộ thì chúng ta gọi DTS là DTS bất đồng bộ. Bây giờ chúng ta sẽ mở rộng DTS với ràng buộc thời gian để đặc tả hệ có yếu tố thời gian của chúng ta. Chúng ra định nghĩa một DTS khoảng (DDTS - Duration Distributed Transition Systems) như là một DTS được trang bị thêm một hàm gán ràng buộc thời gian $J : J(a) = (J_i(a))_{i \in loc(a)}$ với mỗi dịch chuyển a (a-transition).

Định nghĩa 6.4 *Một DTS khoảng (DDTS) là một bộ* $(\mathcal{A}, J)$ *với* $\mathcal{A}$ *là một DTS và* J *là hàm gán ràng buộc khoảng thời gian được xác định như trên.*

Như vậy, một DDTS có thể coi là một DTS trên bảng chữ cái khoảng. Bây giờ chúng ta định nghĩa một thực thi thời gian của DDTS $(\mathcal{A}, J)$ để định nghĩa hành vi của một DDTS. Tương tự trường hợp không có thời gian, chúng ta định nghĩa hai thực thi là đồng bộ và bất đồng bộ.

Định nghĩa 6.5 (Thực thi đồng bộ thời gian) *Một thực thi đồng bộ thời gian của một DDTS là bộ* (ρ, θ) *sao cho:*

- ρ *là một thực thi đồng bộ trên DTS* $\mathcal{A}$, *và*
- *Đối với mỗi* $j \geq 1$ *và với mỗi* $a \in A_{j+1}, b \in A_j$ *và* $(a, b) \in D$ *ta có* $\theta(a) - \theta(b) \in J(a)$.

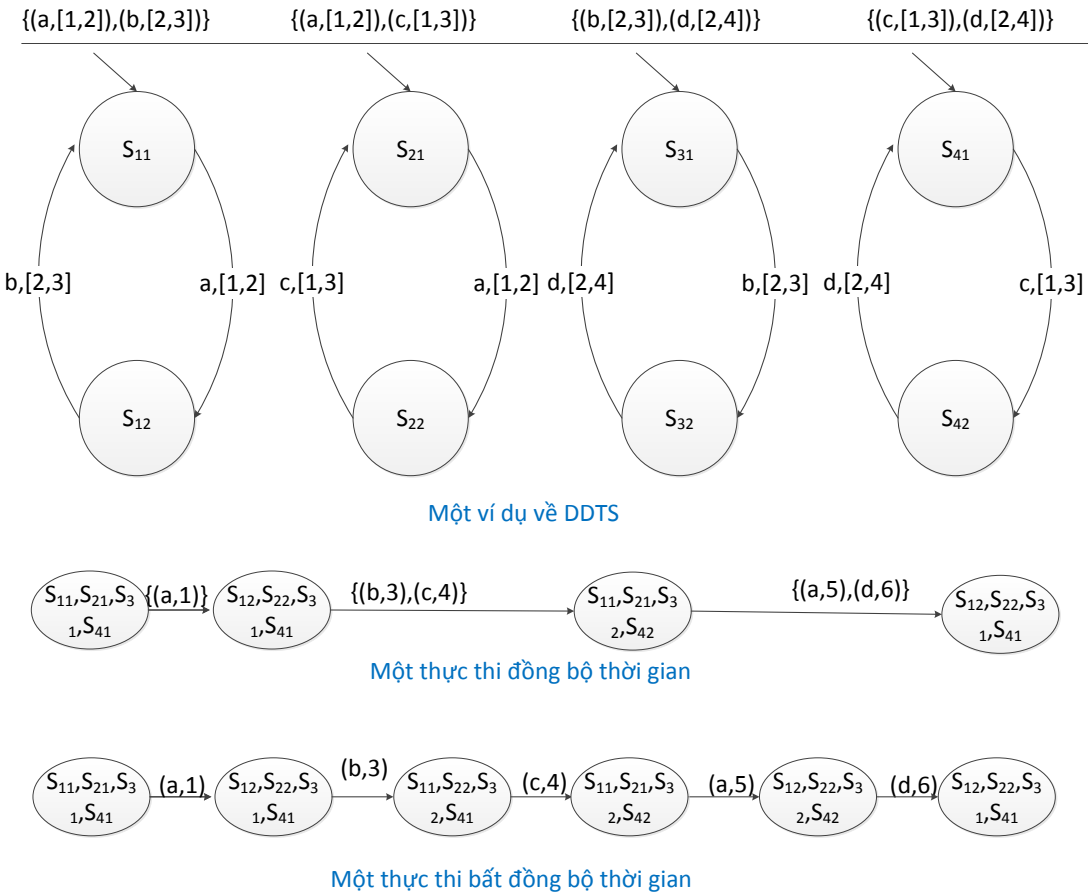
Định nghĩa 6.6 (Thực thi bất đồng bộ thời gian) *Một thực thi bất đồng bộ thời gian của một DDTS là bộ* (ρ, θ) *sao cho:*

- ρ *là một thực thi bất đồng bộ trên DTS* $\mathcal{A}$, *và*
- *Đối với mỗi* $j \geq 1$ *ta có nếu* $(a_{j+1}, a_j) \in D$ *thì* $\theta(a_{j+1}) - \theta(a_j) \in J(a_{j+1})$.

Một ví dụ về hệ phân tán có yếu tố thời gian với bốn tiến trình ($n = 4$), mỗi tiến trình có hai trạng thái địa phương và hai hành động. Mô hình hệ thống và các thực thi đồng bộ và bất đồng bộ của nó được biểu diễn bởi Hình 6.1.

Một thực thi đồng bộ thời gian có liên quan tới Vết thời gian và cấu hình Foata thời gian như sau: Xét chuỗi thực thi bất đồng bộ $(\overline{q_1} A_1 \overline{q_2} A_2 ...)$, mọi hành động $a \in A_i$ có thể được tương ứng với một đỉnh duy nhất v_{ia}, tức là $\lambda(v_{ia}) = a$. Quan hệ $\leq$ được cho như sau: nếu $i \leq j$ thì $v_{ia} \leq v_{ja}$. Quan hệ độc lập và phụ

thuộc được cho bởi nếu hai hành động thuộc hai tiến trình (thành phần) khác nhau thì độc lập nhau. Như vậy, với mỗi thực thi bất đồng bộ thời gian (ρ, θ), ta có tương ứng một Vết thời gian (T, θ) với $T = \langle \{v_{ia} | i \geq 1\}, a \in A_i, \leq, \lambda \rangle$. Đây cũng chính là Vết thời gian với các bước là $A_1, A_2, ...$ Cấu hình Foata thời gian cho Vết này là $C_{A_0} \longrightarrow C_{A_1} \longrightarrow ...$ với $C_{A_0} = \emptyset$. Vì vậy, ta có thể coi thực thi đồng bộ thời gian như là một đồ thị cấu hình Foata (trên Vết thời gian). Điều này cho phép ta phân tích DDTS bằng việc thông qua cấu hình Foata thời gian tương ứng với các thực thi bất đồng bộ thời gian.



Hình 6.1: Hệ phân tán có yếu tố thời gian và các thực thi đồng bộ và bất đồng bộ của nó

Định nghĩa 6.7 (Ngôn ngữ của DDTS) *Cho $(\mathcal{A}, J)$ là một DDTS, ngôn ngữ thời gian của $(\mathcal{A}, J)$ được viết tắt là $L(\mathcal{A}, J)$ là tập tất cả các từ thời gian được xác định bởi tất cả các thực thi đồng bộ thời gian của $(\mathcal{A}, J)$.$L(\mathcal{A}, J) = \{\bigcup_{(\rho,\theta)} tword(T, \theta) | (T, \theta)$ là một Vết thời gian tương ứng với mỗi thực thi đồng bộ thời gian (ρ, θ) của $(\mathcal{A}, J)\}$.*

Kết quả dưới đây chỉ ra mối quan hệ giữa DDTS và ô-tô-mát thời gian khoảng được định nghĩa trong [42] hoặc tương tự như D'Souza và các đồng nghiệp định nghĩa trong [35] theo khía cạnh ngôn ngữ. Nó thể hiện việc chuyển đổi một DDTS tương ứng với một ô-tô-mát. Như vậy, bài toán kiểm chứng có thể dễ dàng giải quyết thông qua lý thuyết về ô-tô-mát.

Mệnh đề 6.1 *Cho một DDTS* $(\mathcal{A}, J)$ *luôn tồn tại một ô-tô-mát thời gian khoảng* $B_{\mathcal{A}}$ *đoán nhận tất cả các từ được xác định bởi thực thi đồng bộ trên* $\mathcal{A}$*.*

Chứng minh: Cho trước một DTS $\mathcal{A}$, từ kết quả trong [50] chúng ta có một ô-tô-mát Büchi $B_{\mathcal{A}}$ mà mọi trạng thái của nó cũng là là trạng thái kết thúc và đoán nhận tất cả các từ được xác định bởi tất cả các thực thi đồng bộ của $\mathcal{A}$. Lấy $B_{\mathcal{A}}$ kết hợp với hàm gán thời gian J chúng ta có ô-tô-mát Büchi khoảng $(B_{\mathcal{A}}, J)$ đoán nhận tất cả các từ thời gian của $\mathcal{A}$ mà thỏa các ràng buộc thời gian do ánh xạ J yêu cầu. Vì vậy, ô-tô-mát này đoán nhận ngôn ngữ $L(\mathcal{A}, J)$. ■

6.2 Lôgic thời gian trên cấu hình Foata

Trong hệ phân tán, chúng ta quan tâm tới các hành động mà có thể được thực thi một cách độc lập với nhau như là một bước dịch chuyển của hệ thống. Tương tự trường hợp không có thời gian, chúng ta định nghĩa cấu hình Foata của một Vết thời gian với mục đích phân tích hành vi của hệ thống. Phần này, luận án giới thiệu một lôgic thời gian được thể hiện trực tiếp trên Vết thời gian dùng để đặc tả các thuộc tính của một hệ phân tán có yếu tố thời gian.

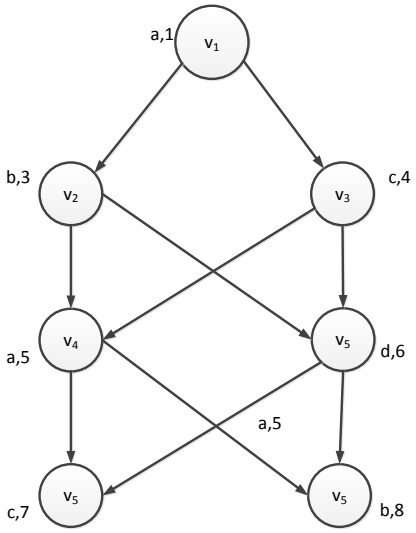
Định nghĩa 6.8 (Cấu hình Foata trên Vết thời gian) *Cho* $(T, \theta) = ((V, \leq, \lambda), \theta)$ *là một Vết thời gian trên bảng chữ cái phụ thuộc* (Σ, D)*.* $\min(V)$ *là tập nhỏ nhất các đỉnh theo quan hệ* $\leq$ *trong* V*. Gọi* FC *là tập con nhỏ nhất của* V *sao cho:*

- $\emptyset \in FC$*, và*
- *với mọi* $C \in FC, (\downarrow \min(V \setminus C)) \in FC$*.*

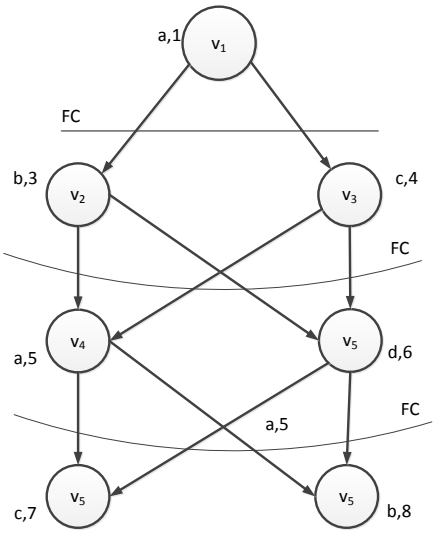
FC *được gọi là tập các cấu hình Foata của Vết* (T, θ)*.*

Cho một Vết thời gian (T, θ), rõ ràng là mỗi cấu hình Foata của Vết thời gian (T, θ), với hàm gán nhãn θ là một đẳng cấu với các tiền tố T. Vì thế, đồ thị của các cấu hình Foata của Vết thời gian (T, θ) là đồ thị con của (T, θ), ký hiệu là $FCG(T)$. Theo trường hợp không có thời gian trong [29], chúng ta có tập các

cấu hình Foata FC có thứ tự tuyến tính và được ký hiệu là $fconf(T)$. Một cấu hình Foata và đồ thị cấu hình Foata[1] của một Vết thời gian được chỉ ra trong Hình 6.3 và Hình 6.4.



Hình 6.2: Một Vết thời gian

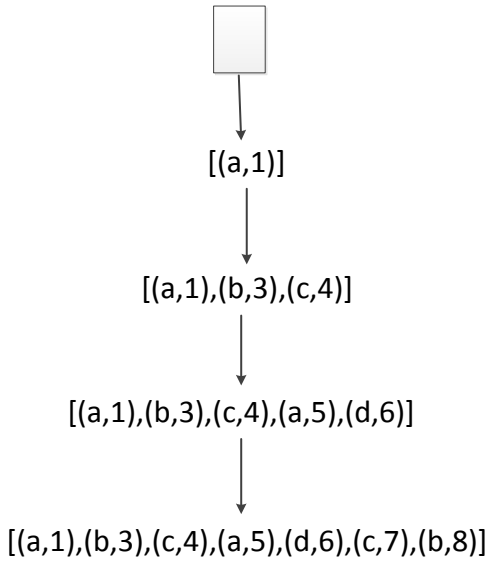


Hình 6.3: Cấu hình Foata của một Vết thời gian trong Hình 6.2

Tiếp theo, chúng ta đưa ra định nghĩa về tuyến tính hóa và chuẩn Foata như là một phân tích của một Vết thời gian nhằm đưa ra một quan hệ với ô-tô-mát

[1]Thay vì chỉ ra các sự kiện (hành động) của các cấu hình của Vết thời gian, chúng tôi biểu diễn bởi các nhãn của chúng (dùng ánh xạ λ và θ)

thời gian và các từ thời gian.



Hình 6.4: Đồ thị cấu hình Foata của Vết thời gian được chỉ ra trong Hình 6.2

Định nghĩa 6.9 (Tuyến tính hóa Foata) *Một tuyến tính hóa Foata của một Vết thời gian* $T = (\langle V, \leq, \lambda\rangle, \theta)$ *là một tuyến tính hóa* $T' = ((\langle V, \leq', \lambda\rangle, \theta)$ *biểu diễn như là một tích hữu hạn các Vết thời gian được định nghĩa như công thức sau:*

$$((V, \leq', \lambda), \theta) = \prod_{i=1,\ldots,\infty}((V_i, \leq'_i, \lambda_i), \theta_i)$$

sao cho với mỗi $i \geq 1$*:*

- V_i *là tập các hành động độc lập từng đôi một,*
- *với mỗi* $v \in V_{i+1}$ *tồn tại một* $v' \in V_i$ *sao cho* $(v, v') \in D$*, và*
- *với mỗi* $v \in V_i, \theta_i(v) = \theta(v)$*.*

Định nghĩa 6.10 (Chuẩn Foata) *Một từ thời gian* $\omega \in (\Sigma \times \mathcal{R}^{\geq 0})^\omega$ *là một dạng chuẩn Foata nếu và chỉ nếu các điều kiện sau được thỏa mãn:*

1. $\omega = u_1 u_2 \ldots$ *với* $u_i \in (\Sigma \times \mathcal{R}^{\geq 0})^*$*,*
2. *với mỗi* $i \geq 1$ *từ thời gian* u_i *là một tích (ghép)các hành động độc lập, và*

3. *với mỗi* $i \geq 1$ *và với mọi* a *trong* u_{i+1} *tồn tại một ký hiệu hành động* b *trong* u_i *mà phụ thuộc với* a.

Các từ u_i *được gọi là các bước của hệ thống.*

Vì thế, trong mỗi bước, có một tập các hành động được thực thi. Những hành động này là độc lập nhau trong bảng chữ cái Σ, tức là trong $I(\Sigma)$.

Kết quả sau dễ dàng suy ra từ các khái niệm cơ bản của Vết thời gian, tuyến tính hóa Foata và dạng chuẩn Foata. Cho một từ thời gian $\omega \in (\Sigma \times \mathbb{R}^{\geq 0})^*, \omega = a_1 t_1 a_2 t_2 ...$, chúng ta ký hiệu $\omega = (a_1 a_2 ..., \theta)$ nếu và chỉ nếu $\theta(a_i) = t_i$ với mọi $i \geq 1$.

Hệ quả 6.1 *Cho một Vết thời gian* (T, θ), $tword(T, \theta) = \{\omega \in (ttow(T), \theta)\}$ *và* ω *là một dạng chuẩn Foata.*

Ví dụ 6.1 *Cho* $\omega = (a, 1)(b, 2)(c, 3)(a, 4)(d, 5)(b, 6)(c, 7)$ *là một từ thời gian trên bảng chữ cái* $\Sigma = \{a, b, c, d\}$ *và quan hệ phụ thuộc* $D = \{(a, b), (a, c), (b, d), (c, d)\}$. *Rõ ràng là* ω *ở dạng chuẩn Foata và* $\omega = u_1 u_2 u_3 u_4$ *với* $u_1 = a, u_2 = bc, u_3 = ad, u_4 = bc$.

Phần tiếp theo, luận án đề xuất một lôgic thời gian tuyến tính được giải nghĩa trực tiếp trên cấu hình Foata.

Cú pháp

Công thức của hệ lôgic được cho dưới dạng như sau:

Định nghĩa 6.11 *Cho* (Σ, I) *là bảng chữ cái độc lập và* $I(\Sigma)$ *là tập các tập con là các phần tử độc lập của* Σ. *Lôgic thời gian tuyến tính là tập các công thức được định nghĩa theo cách như sau:*

$$\varphi :: \top | \neg\varphi | \varphi_1 \vee \varphi_2 | \langle A \rangle \varphi | EX_{dur} \varphi | \varphi \cup \psi$$

với $A \in I(\Sigma), dur \in intv$.

Ngữ nghĩa

Định nghĩa 6.12 (Ngữ nghĩa của logic) *Cho* (T, θ) *là một Vết thời gian trên* (Σ, I), *ngữ nghĩa của công thức* $\varphi \in TLTL_f(\Sigma, I)$ *được định nghĩa trên cấu hình Foata thời gian* C *như sau:*

- $T, C \models \top$
- $T, C \models \neg\varphi \Leftrightarrow T, C \not\models \varphi$

- $T, C \models \varphi \vee \psi \Leftrightarrow T, C \models \varphi$ *hoặc* $T, C \models \psi$
- $T, C \models \langle A \rangle \varphi$ *khi và chỉ khi tồn tại một* $A' \in I(\Sigma), A' \supseteq A$ *và* $C' \in fconf(T, \theta)$ *sao cho* $C \xrightarrow{A'} C'$ *và* $T, C' \models \varphi$ *với* $\lambda(C' \setminus C) = A'$
- $T, C \models EX_{dur}\varphi$ *nếu tồn tại một* $A \in I(\Sigma)$ *và* $C' \in fconf(T, \theta)$ *sao cho* $C \xrightarrow{A} C'$ *và* $T, C' \models \varphi$ *và với mọi* $b \in C' \setminus C, \forall a \in \lessdot_b$ *ta có* $\theta(b) - \theta(a) \in dur$
- $T, C \models \varphi \cup \psi$ *khi và chỉ khi tồn tại* $C' \in fconf(T, \theta), C' \supseteq C$ *sao cho* $T, C' \models \psi$ *và mọi* $C" \in fconf(T, \theta), C \subseteq C" \subset C'$ *kéo theo* $T, C" \models \varphi$

Ta nói, T mô hình φ, T là một mô hình của φ hoặc T thỏa φ khi và chỉ khi $T, \emptyset \models \varphi$, trong trường hợp này, để cho gọn, ta có thể viết $T \models \varphi$. Tất cả các mô hình của công thức $\varphi \in TLTL_f(\Sigma, I)$ tạo thành tập con của các Vết trên $((\Sigma, I))$ và do đó còn được gọi là ngôn ngữ được định nghĩa bởi φ và ký hiệu là $L(\varphi)$. Bài toán đặt ra là cho một công thức $\varphi \in TLTL_f(\Sigma, I)$, liệu có tồn tại một Vết thời gian T sao cho T là một mô hình của φ. Tương tự như trường hợp không có ràng buộc thời gian, ta có định lý sau:

Định lý 6.1 *Cho* $\varphi \in TLTL_f(\Sigma, I)$*, luôn tồn tại một thuật toán kiểm tra xem liệu có tồn tại Vết thời gian* T *sao cho* T *thỏa công thức* φ*.*

Chứng minh: Trước tiên, ta nhận thấy rằng trong công thức $TLTL_f(\Sigma, I)$, nếu ta gán mỗi ký hiệu a xuất hiện trong công thức ψ trong $EX_{dur}\psi$ với một hàm gán thời gian khoảng J với *J(a)=dur* thì công thức $\varphi \in TLTL_f(\Sigma, I)$ tương ứng với công thức $\varphi' \in LTL_f(\Sigma', I)$ với $\Sigma' = (\Sigma, J)$. Khi đó, tương tự như trường hợp không có thời gian, ta có thể xây dựng được một ô-tô-mát Büchi $\mathcal{A}_{\varphi'}$ có tính chất đoán nhận tất cả các tuyến tính Foata ω của φ' khi và chỉ khi $wtot(\omega) \models \varphi'$. Như vậy $\mathcal{A}_{\varphi'}$ sẽ đoán nhận các mô hình của φ' (và cả các từ khác nữa, tức là $L(\varphi) \subseteq L(\mathcal{A}_{\varphi'})$). Tiếp theo, ta xây dựng một ô-tô-mát Büchi $\mathcal{A}_{\mathcal{F}}$ đoán nhận tất cả các từ ω có dạng chuẩn Foata. Ô-tô-mát này được xây dựng trong tài liệu [50]. Cuối cùng, chúng ta xây dựng ô-tô-mát giao $\mathcal{A}$ sao cho $L(\mathcal{A}) = \mathcal{L}(\mathcal{A}_{\varphi'}) \cap \mathcal{A}_{\mathcal{F}}$. Kiểm tra ngôn ngữ rỗng của *(*$\mathcal{A}$*,J)* cho ta kết quả của bài toán. Nếu *L(*$\mathcal{A}$*,J)* $=\emptyset$ thì khẳng định không tồn tại bất kỳ Vết thời gian T nào thỏa công thức φ. Ngược lại, khi đó sẽ tồn tại ít nhất một từ $\omega \in L(\mathcal{A}, J)$, đặt $T = wtot(\omega)$, rõ ràng rằng T chính là Vết thỏa công thức φ. ■

Mệnh đề 6.2 *Cho* $\varphi \in TLTL_f(\Sigma, I)$*, tồn tại một ô-tô-mát Büchi thời gian khoảng* $\mathcal{A}_{\neg\varphi}$ *đoán nhận ngôn ngữ* $L(\neg\varphi)$*.*

Chứng minh: Như lập luận trong chứng minh định lý 6.1, một công thức $TLTL_f(\Sigma, I)$ có thể coi là công thức $LTL_f(\Sigma_{dur}, I)$ với Σ_{dur} là bảng chữ cái khoảng, tức là bảng chữ cái (Σ, J). Theo [50] ta có với một công thức $\psi \in LTL_f(\Sigma, I)$ luôn tồn tại một ô-tô-mát Büchi $\mathcal{A}_{\neg\psi}$ sao cho $L(\neg\psi) = L(\mathcal{A}_{\neg\psi})$. Do đó, nếu cho công thức $\varphi \in TLTL_f(\Sigma, I)$ sẽ có tương ứng một công thức $\varphi' \in LTL_f(\Sigma_{dur}, I)$ sao cho $L(\neg\varphi) = L(\neg\varphi')$ trên cùng bảng chữ cái khoảng. Và do vậy sẽ tồn tại một ô-tô-mát Büchi $\mathcal{A}_{\neg\varphi'}$ sao cho $L(\neg\varphi') = L(\mathcal{A}_{\neg\varphi'}) = \mathcal{L}(\neg\varphi)$ trên bảng chữ cái khoảng, đây cũng chính là ô-tô-mát Büchi khoảng. ■

Phần tiếp theo luận án sẽ đưa ra thuật toán cho bài toán kiểm chứng hệ thống.

6.3 Bài toán kiểm chứng

Bài toán kiểm chứng mô hình được phát biểu như sau. Cho một hệ DDTS $(\mathcal{A}, J)$ đặc tả hành vi của một hệ thống phân tán có yếu tố thời gian và L là tập các từ thời gian (chuẩn Foata) tương ứng với các thực thi của hệ dịch chuyển, một công thức $\varphi \in TLTL_f(\Sigma, I)$ đặc tả tính chất của hệ thống, liệu hệ thống có thỏa tính chất được đặc tả bởi công thức lôgic φ đó hay không, tức là $L \subseteq L(\varphi)$ hay không.

Ý tưởng cơ bản giải quyết bài toán này dựa trên phương pháp lý thuyết ô-tô-mát truyền thống như sau: Trước tiên, chúng ta xây dựng một ô-tô-mát $B_{\mathcal{A}}$ đoán nhận tất cả các thực thi đồng bộ của DTS $\mathcal{A}$. Tiếp theo, chúng ta xây dựng ô-tô-mát Buchi $B_{\neg\varphi}$ đoán nhận ngôn ngữ phần bù của $L(\varphi)$. Cuối cùng, chúng ta lấy giao của hai ô-tô-mát để được một ô-tô-mát giao $\mathcal{C} = \mathcal{B}_{\mathcal{A}} \cap \mathcal{B}_{\neg\varphi}$, kết hợp nó với ánh xạ J ta được ô-tô-mát khoảng $(\mathcal{C}, J)$. Việc kiểm tra tính rỗng của ô-tô-mát khoảng là quyết định được, từ đó ta sẽ có lời giải cho bài toán.

Tính quyết định của bài toán được giải quyết bằng định lý sau:

Định lý 6.2 *Cho DDTS $(\mathcal{A}, J)$ với L là tập các từ thời gian (chuẩn Foata) tương ứng với các thực thi của hệ dịch chuyển và một công thức $\varphi \in TLTL_f(\Sigma, I)$, luôn tồn tại một thuật toán để quyết định liệu $L \subseteq L(\varphi)$.*

Chứng minh: Từ mệnh đề 6.2 ta dễ dàng chỉ ra một ô-tô-mát Büchi $B_{\neg\varphi}$ đoán nhận ngôn ngữ phần bù của $L(\varphi)$ là $L(\neg\varphi)$. Từ mệnh đề 6.1 ta có thể xây dựng một ô-tô-mát Büchi $\mathcal{B}_{\mathcal{A}}$ đoán nhận ngôn ngữ là các thực thi đồng bộ của DTS $\mathcal{A}$. Kết hợp với ánh xạ thời gian J, ta có một ô-tô-mát Büchi khoảng $(\mathcal{B}_{\mathcal{A}}, \mathcal{J})$.

Dựa vào hai ô-tô-mát này với việc kiểm tra tính rỗng của ngôn ngữ giao của chúng, ta có thể quyết định được liệu $L \subseteq L(\varphi)$. ■

Dựa trên cách chứng minh định lý 6.2, luận án đưa ra thuật toán kiểm chứng hệ thống có bốn bước cơ bản như sau.

- Bước 1. Xây dựng một ô-tô-mát Büchi $\mathcal{B}_{\mathcal{A}}$ đoán nhận ngôn ngữ là các thực thi đồng bộ của DTS $\mathcal{A}$. Kết hợp với ánh xạ thời gian J, ta có một ô-tô-mát Büchi khoảng $(\mathcal{B}_{\mathcal{A}}, J)$.
- Bước 2. Chúng ta xây dựng ô-tô-mát Büchi $B_{\neg\varphi}$ đoán nhận ngôn ngữ phần bù của $L(\varphi)$ là $L(\neg\varphi)$ theo như chứng minh mệnh đề 6.2. Và ta có một ô-tô-mát Büchi khoảng $(\mathcal{B}_{\neg\varphi}, \mathcal{J})$ đoán nhận ngôn ngữ được xác định bởi phần bù công thức $\neg\varphi$.
- Bước 3. Xây dựng ô-tô-mát Büchi khoảng $(\mathcal{B}, \mathcal{J})$ sao cho $L(\mathcal{B}, J) = L(\mathcal{B}_{\mathcal{A}}, J) \cap L(\mathcal{B}_{\neg\varphi}, J)$.
- Bước 4. Kiểm tra tính rỗng của $L(\mathcal{B}, J)$. Nếu $L(\mathcal{B}, J) = \emptyset$ thì khẳng định là hệ thống thỏa mãn các thuộc tính kiểm tra và ngược lại.

6.4 Các nghiên cứu liên quan

Các hệ phân tán (có yếu tố thời gian) được xây dựng và phát triển trong nhiều lĩnh vực khác nhau. Do đó, nghiên cứu các phương pháp để đảm bảo độ tin cậy của hệ thống này là cần thiết. Các kết quả trong [47] đã đưa ra một số phương pháp hình thức về tính toán, lập trình song song, về xử lý các vấn đề tương tranh [50, 55] để đặc tả và kiểm chứng các hệ thống này. Tuy nhiên, các phương pháp tập trung nghiên cứu các hệ thống không có ràng buộc thời gian. Trước vấn đề đó, một số các nghiên cứu về hệ có yếu tố thời gian mà điển hình là các ô-tô-mát thời gian[4] đã đặt nền tảng cho sự phát triển mạnh mẽ của các hệ thống có ràng buộc thời gian, tuy nhiên lớp các ngôn ngữ thời gian được đoán nhận bởi ô-tô-mát thời gian lại không đóng đối với phép lấy phần bù. Vì vậy, một số nghiên cứu trong [67, 35, 48] đã dựa trên lý thuyết ô-tô-mát thời gian và bổ sung một số ràng buộc để đảm bảo tính đóng của ngôn ngữ với phép lấy phần bù. Dù vậy, những phương pháp này vẫn còn một số hạn chế về đặc tả và biểu diễn các thuộc tính bằng lôgic do các phương pháp này có ngữ nghĩa của hệ thống được thể hiện dựa trên các từ trong khi đối với các hệ phân tán, hoạt động của hệ thống tương đương với một Vết (lớp tương đương các từ có thứ tự

bộ phận), mặt khác việc thể hiện tính tương tranh trong hệ phân tán đối với các phương pháp này cũng có nhiều hạn chế và khó sử dụng.

Các nghiên cứu khác đã chỉ ra rằng, đối với các hệ phân tán (hay các hệ tương tranh), Vết Mazurkiewicz là phù hợp cho đặc tả các hệ thống phân tán [30, 75, 14]. Martin Leucker [50] đã đề xuất mô hình hệ phân tán bằng hệ dịch chuyển phân tán, trong đó ngôn ngữ của hệ dịch chuyển là các Vết. Tác giả cũng chỉ ra ưu điểm của việc sử dụng hệ dịch chuyển cũng như các Vết trong việc xử lý các kỹ thuật của mô hình hệ phân tán. Tuy nhiên, lý thuyết Vết chưa hỗ trợ đặc tả ràng buộc thời gian. Một số các nghiên cứu được tổng hợp trong [66] đã cho một cái nhìn bao quát về các nghiên cứu hệ thống phân tán, trong đó các nghiên cứu về bài toán kiểm chứng hệ phân tán có sử dụng Vết Mazurkiewicz để giảm không gian trạng thái. Các nghiên cứu còn đề cập tới kiểm chứng các hệ phân tán có thời gian. Trong đó, các tác giả sử dụng các lý thuyết về ô-tô-mát thời gian [4] và mạng Petri để biểu diễn mô hình hệ thống nên việc đặc tả các tính chất tương tranh còn gặp nhiều khó khăn như đã phân tích trên. Để giải quyết vấn đề này, Vết thời gian đã được đề xuất[2] là một sự mở rộng Vết Mazurkiewicz với ràng buộc thời gian trên các hành động của hệ thống. Trong đó, các nghiên cứu đã chỉ ra rằng, Vết thời gian là phù hợp cho đặc tả hệ thống tương tranh có yếu tố thời gian bởi tính đơn giản nhưng vẫn đủ mạnh để đặc tả một lớp lớn các ứng dụng thực tế cho các hệ tương tranh có yếu tố thời gian, đặc biệt là có khả năng hỗ trợ kiểm chứng tự động.

6.5 Kết luận

Chương này đã giới thiệu một phương pháp hình thức để đặc tả và kiểm chứng các hệ phân tán có yếu tố thời gian. Ý tưởng cơ bản của phương pháp là mở rộng hệ dịch chuyển phân tán của Martin Leucker với ràng buộc thời gian trên các quan hệ dịch chuyển để đặc tả hành vi hệ thống. Vết thời gian đã được ứng dụng để đặc tả hành vi hệ thống. Bên cạnh đó nghiên cứu cũng đề xuất một lôgic thời gian trên các cấu hình Foata của Vết thời gian để đặc tả các thuộc tính lôgic. Luận án đã chỉ ra rằng bài toán kiểm chứng cho phương pháp này là quyết định được tuy độ phức tạp tính toán vẫn là hàm mũ (tương đương với độ phức tạp tính toán trên ô-tô-mát thời gian). Các hệ phân tán thường được coi là hệ đa tiến trình và được biểu diễn bằng tích các tiến trình hệ thống, điều này rất phù hợp cho việc áp dụng bộ mô hình kiểm chứng SPIN[3] để kiểm thử

[2]Trong danh mục công trình khoa học liên quan tới luận án

[3]http://spinroot.com

hệ thống. Mặc dù SPIN đã được nghiên cứu để mở rộng về thời gian (Timed SPIN) nhưng kết quả còn nhiều hạn chế và công cụ vẫn còn nhiều lỗi và chưa được sử dụng rộng rãi (đối với phần mở rộng về thời gian). Nhận thấy khả năng mở rộng của SPIN là khả quan, chúng tôi cũng đang nghiên cứu để có thể biểu diễn ràng buộc thời gian vào ngôn ngữ Promela mà không cần sửa đổi mã nguồn SPIN. Các kết quả trong chương này được công bố trong công trình khoa học số [3] của danh mục công trình khoa học đã công bố của luận án.

Chương 7

Kết luận

Trong chương này, luận án tổng hợp lại các kết quả nghiên cứu, các đóng góp chính của luận án và các vấn đề chưa giải quyết cũng như định hướng các nghiên cứu tiếp theo.

7.1 Các kết quả đạt được

Công nghệ phần mềm dựa trên thành phần với nhiều ưu điểm đã trở thành một trong những hướng tiếp cận phát triển phần mềm thông dụng nhất hiện nay. Nhằm đảm bảo chất lượng dịch vụ của các phần mềm phát triển theo hướng tiếp cận này, luận án đề xuất một phương pháp hình thức hỗ trợ đặc tả các hệ thống tương tranh có ràng buộc thời gian, được phát triển dựa trên thành phần. Với phương pháp nghiên cứu của luận án là tìm hiểu các phương pháp đang có, phân tích các ưu điểm, hạn chế để tổng hợp đưa ra các đề xuất giải pháp hiệu quả nhất, luận án đã tìm hiểu các phương pháp của các tác giả đã nghiên cứu liên quan tới công nghệ phần mềm dựa trên thành phần, bài toán đặc tả và kiểm chứng các hệ thống phần mềm dựa trên thành phần có tính tương tranh và (hoặc) ràng buộc thời gian [18, 25, 72, 37, 26, 24, 30, 55, 75, 4, 35, 48]. Các nghiên cứu cơ sở nền tảng cho thấy hạn chế của các phương pháp hiện tại là chưa hỗ trợ một cách hiệu quả việc đặc tả và kiểm chứng các hệ tương tranh có ràng buộc thời gian. Từ đó, luận án đề xuất lựa chọn việc mở rộng lý thuyết vết Mazurkiewicz về thời gian và được gọi là Vết thời gian. Với mở rộng này, các kết quả nghiên cứu đã cho thấy Vết thời gian là một công cụ hiệu quả, có ý nghĩa cho việc hỗ trợ đặc tả các ràng buộc về thời gian trên các hệ thống tương tranh. Các phát triển trong lý thuyết Vết thời gian hỗ trợ đặc tả hành vi hệ

thống (là các Vết thời gian), mô hình hệ thống (là các giao diện thành phần có thể được biểu diễn thông qua các vết khoảng hoặc ô-tô-mát khoảng bất đồng bộ), các thuộc tính logic của hệ thống (là các công thức TLTL). Vết thời gian có thể dễ dàng đặc tả các hành vi, tính chất của hệ thống tương tranh có ràng buộc thời gian mà hiện nay hiếm thấy phương pháp nào có thể đặc tả được.

Tiếp sau đó, luận án đã nghiên cứu việc áp dụng Vết thời gian vào ba mô hình đặc tả hệ thống dựa trên thành phần nhằm mở rộng các mô hình này để có thể đặc tả được các hệ thống tương tranh có ràng buộc thời gian, cụ thể.

i) Luận án đề xuất một mô hình dựa trên lý thuyết rCOS [74]. Các Vết thời gian được sử dụng để đặc tả các giao thức trong các giao diện thành phần nhằm hỗ trợ đặc tả các truy cập đồng thời có yếu tố thời gian tới các dịch vụ của một thành phần. Mô hình này cũng định nghĩa một hợp đồng bao gồm đặc tả phương thức và định nghĩa một thành phần như là một sự thực thi của một hợp đồng. Hệ thống trong mô hình này sẽ là một thành phần chủ động được cắm vào một thành phần bị động. Bài toán khó trong mô hình này là chỉ ra thuật toán tìm các giao thức cho các thành phần kết hợp đã được giải quyết.

ii) Luận án phát triển một mô hình thiết kế hệ thống dựa trên giao diện [28, 3, 46], trong đó đề xuất khái niệm ô-tô-mát giao diện tương tranh có yếu tố thời gian, là một ô-tô-mát khoảng bất đồng bộ khi chỉ quan tâm các hành động đầu vào và đầu ra của hệ thống. Khi đó mỗi thành phần của hệ thống sẽ được đặc tả bởi một ô-tô-mát này và hệ thống sẽ là một tích song song các ô-tô-mát. Các vấn đề về ghép nối, khả năng ghép nối, môi trường và làm mịn cũng được đề xuất và chứng minh.

iii) Luận án đã chỉ ra phương pháp đề xuất có khả năng mở rộng cho các hệ thống phân tán có yếu tố thời gian bằng việc đưa ra mô hình hệ thống phân tán tương tranh có yếu tố thời gian. Nghiên cứu này là sự mở rộng về thời gian của hệ dịch chuyển phân tán [50] mà ở đó ngôn ngữ đoán nhận là các Vết thời gian.

Như vậy, các kết quả trong luận án đã đạt được mục tiêu của luận án, đề xuất phương pháp hiệu quả cho đặc tả các giao diện thành phần hệ thống đảm bảo chất lượng dịch vụ (các ràng buộc về thời gian) và tính tương tranh (các hoạt động đồng thời hoặc tuần tự của các tiến trình hệ thống). Các kết quả này là đáng tin cậy, có ý nghĩa và có thể mở rộng phát triển ứng dụng thực tiễn.

7.2 Hướng phát triển tiếp theo

Luận án đã nghiên cứu và đưa ra giải pháp hiệu quả đáp ứng mục tiêu, mục đích và các yêu cầu đề ra. Tuy nhiên luận án vẫn còn một số vấn đề cần được nghiên cứu để hoàn thiện phương pháp, cụ thể như sau:

i) Phương pháp đề xuất trong luận án chưa được áp dụng mô hình trên một công cụ kiểm chứng mô hình cụ thể. Các kết quả trong các nghiên cứu đã chứng minh ý nghĩa khoa học về mặt lý thuyết tính toán nhưng chưa có được một công cụ kiểm chứng mô hình cụ thể nào được áp dụng nên khả năng ứng dụng của phương pháp đề xuất vẫn còn bỏ ngỏ. Do đó cần triển khai áp dụng lý thuyết Vết thời gian vào một mô hình kiểm chứng cụ thể để có thể nâng cao hiệu quả cũng như ý nghĩa khoa học của nghiên cứu vào ứng dụng thực tế.

ii) Lôgic thời gian đặc tả Vết thời gian còn đơn giản và chưa chứng minh được mối liên hệ với các ô-tô-mát và ngôn ngữ Vết khi có ràng buộc thời gian trên toán tử U(Until). Mặc dù có thể có nhiều giải pháp để có thể biểu diễn thay thế cho toán tử U khi có ràng buộc thời gian nhưng việc nghiên cứu tiếp lôgic thời gian trên Vết thời gian một cách chi tiết và sâu hơn cũng là vấn đề rất cần thiết. Cần nghiên cứu và đưa ra giải pháp cụ thể cho trường hợp này.

iii) Các ứng dụng cho các bài toán thực tế chưa rõ ràng. Cần hoàn thiện lý thuyết về phương diện thực hành nhiều hơn, nghiên cứu và áp dụng phương pháp vào các ví dụ điển hình trong thực tế để làm nổi bật ý nghĩa của lý thuyết nghiên cứu.

iv) Thuật toán kiểm chứng cần phát triển chi tiết và cụ thể hơn để có thể triển khai xây dựng các công cụ kiểm chứng mô hình. Khó có thể áp dụng lý thuyết mới đề xuất trong luận án vào một bộ kiểm chứng mô hình có sẵn nào đó. Do vậy, cần phải phát triển tiếp để có thể tiến tới xây dựng một công cụ hỗ trợ cho phương pháp đề xuất và áp dụng công cụ này vào các hệ thống để minh chứng cho tính hiệu quả của phương pháp đề xuất.

Các nghiên cứu gần đây đã đưa ra thêm khái niệm, lý thuyết về giao diện liên hệ có ràng buộc thời gian (timed relation interface) thể hiện không chỉ khái niệm giao diện thông thường mà còn biểu diễn được mối quan hệ giữa các hành động đầu vào và đầu ra của thành phần. Đây là nghiên cứu mới, khá hay để có thể đặc tả một cách toàn diện hơn nữa các đặc tính hệ thống hiện nay, tuy

nhiên mô hình lý thuyết này chưa hỗ trợ đặc tả tính tương tranh trong hệ thống. Luận án đã nghiên cứu và có thấy rằng có thể áp dụng lý thuyết Vết thời gian vào để mở rộng các đặc tả giao diện theo lý thuyết đó. Các kết quả nghiên cứu về vấn đề này sẽ sớm được công bố trong thời gian gần nhất.

DANH MỤC CÁC CÔNG TRÌNH KHOA HỌC CỦA TÁC GIẢ LIÊN QUAN ĐẾN LUẬN ÁN

[1] Do Van Chieu and Dang Van Hung (2010), *"An extension of Mazurkiewicz traces and their applications in specication of real-time systems"*, In Proceedings of the second international Conference on knowledge and systems engineering 2010. IEEE Computer Society, pp. 167-172.

[2] Do Van Chieu and Dang Van Hung (2011), *"A formal model for concurrent real-time component-based systems"*, Tạp chí Khoa học và Công nghệ - Viện Khoa học và Công nghệ Việt Nam, Vol. 49, No.4A, pp. 313-226 (ISSN: 0866 708X).

[3] Do Van Chieu and Dang Van Hung (2012), *"Timed traces and their applications in specication and veriction of distributed real-time systems"*, In Proceedings of the Third International Symposium on Information and Communication Tecnology 2012, IEEE Computer Society, pp. 31-41.

[4] Do Van Chieu (2015), *"A Formal Method for Specifying Interface of Component in Real-time Concurrent Systems"* , Tạp chí Bưu chính Viễn thông, Vol. E-3, No. 8(12), pp. 48-57 (ISSN: 1859 – 3534), 2015.

Tài liệu tham khảo

[1] F. Aarts, F. Heidarian, and F. Vaandrager. A theory of history dependent abstractions for learning interface automata. In *Proceedings of the 23rd International Conference on Concurrency Theory*, CONCUR'12, pages 240–255, Berlin, Heidelberg, 2012. Springer-Verlag.

[2] J. Abrial. Data semantics. In *IFIP Working Conference Data Base Management*, pages 1–60, 1974.

[3] L. D. Alfaro and T. A. Henzinger. Interface-based design. In *In Engineering Theories of Software Intensive Systems, proceedings of the Marktoberdorf Summer School*. Kluwer, 2004.

[4] R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[5] C. Angelov, F. Zhou, and K. Sierszecki. Specification of embedded control systems behaviour using actor interface automata. In *Proceedings of the 8th IFIP WG 10.2 International Conference on Software Technologies for Embedded and Ubiquitous Systems*, SEUS'10, pages 167–178, Berlin, Heidelberg, 2010. Springer-Verlag.

[6] M. S. Anne Remke, editor. *Stochastic Model Checking. Rigorous Dependability Analysis Using Model Checking Techniques for Stochastic Systems*. Springer, 2012.

[7] E. Asarin, P. Caspi, and O. Maler. A kleene theorem for timed automata. In *12th Annual IEEE Symposium on Logic in Computer Science (LICS'97). IEEE Computer*, pages 160–171. IEEE Computer Society Press, 1997.

[8] C. Baier and J.-P. Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[9] A. Basu, M. Bozga, and J. Sifakis. Modeling heterogeneous real-time components in bip. In *Proceedings of the Fourth IEEE International Conference on Software Engineering and Formal Methods*, pages 3–12, Washington, DC, USA, 2006. IEEE Computer Society.

[10] G. Behrmann, R. David, and K. G. Larsen. A tutorial on uppaal. pages 200–236. Springer, 2004.

[11] J. Bengtsson and W. Yi. Timed automata: Semantics, algorithms and tools. In *Lectures on Concurrency and Petri Nets*, pages 87–124, 2003.

[12] T. Y. Bin Zhou and C. J. Myers. Framework of timed trace theoretic verification revisited. In *10th Asian Test Symposium (ATS 2001), 19-21 November 2001, Kyoto, Japan*, pages 437–442, 2001.

[13] B. Bollig. *Formal Models of Communicating Systems: Languages, Automata, and Monadic Second-Order Logic (Texts in Theoretical Computer Science. An Eatcs Series).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[14] B. Bollig and M. Leucker. Deciding ltl over mazurkiewicz traces. *Data Knowl. Eng.*, 44(2):219–238, 2003.

[15] A. W. Brown and K. C. Wallnau. The current state of cbse. *IEEE Softw.*, 15:37–46, September 1998.

[16] J. Brzozowski and E. Leiss. On equations for regular languages, finite automata, and sequential networks. *Theoretical Computer Science*, 1980.

[17] L. Burdy, Y. Cheon, D. Cok, M. D. Ernst, J. Kiniry, G. T. Leavens, K. Rustan, M. Leino, and E. Poll. An overview of jml tools and applications, 2003.

[18] X. Cai, M. R. Lyu, and K. fai Wong. Component-based software engineering: Technologies, development frameworks, and quality assurance schemes. In *Lecture Notes*, pages 372–379. IEEE Computer Society, 2000.

[19] Z. Cao and H. Wang. Extending interface automata with z notation. In *Proceedings of the 4th IPM International Conference on Fundamentals of Software Engineering*, FSEN'11, pages 359–367, Berlin, Heidelberg, 2012. Springer-Verlag.

[20] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *J. ACM*, 28:114–133, January 1981.

[21] S. Chouali and A. Hammad. Formal verification of components assembly based on sysml and interface automata. *Innov. Syst. Softw. Eng.*, 7(4):265–274, Dec. 2011.

[22] J.-P. K. Christel Baier. *Principles of Model Checking*. The MIT Press, 2008.

[23] M. M. Chupilko and A. S. Kamkin. Runtime verification based on executable models: On-the-fly matching of timed traces. In *Proceedings Eighth Workshop on Model-Based Testing, MBT 2013, Rome, Italy, 17th March 2013.*, pages 67–81, 2013.

[24] E. M. Clarke, O. Grumberg, and D. Peled. *Model checking*. MIT Press, 1999.

[25] I. Crnkovic. Component-based software engineering - new challenges in software development. In *in Software Development. Software Focus*, pages 127–133. John Wiley, Sons, 2001.

[26] I. Crnkovic and M. Larsson. A case study: Demands on component-based development. In *Proceedings, 22th International Conference of Software Engineering*. ACM, IEEE, May 2000.

[27] H. Dang Van and H. Truong. Modeling and specification of real-time interfaces with utp. In Z. Liu, J. Woodcock, and H. Zhu, editors, *Theories of Programming and Formal Methods*, volume 8051 of *Lecture Notes in Computer Science*, pages 136–150. Springer Berlin Heidelberg, 2013.

[28] L. de Alfaro and T. A. Henzinger. Interface automata. In *Proceedings of the Ninth Annual Symposium on Foundations of Software Engineering (FSE), ACM*, pages 109–120. Press, 2001.

[29] V. Diekert. *Combinatorics on traces*. Springer-Verlag New York, Inc., New York, NY, USA, 1990.

[30] V. Diekert. *The Book of Traces*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1995.

[31] V. Diekert and P. Gastin. From local to global temporal logics over mazurkiewicz traces. *Theor. Comput. Sci.*, 356(1):126–135, 2006.

[32] V. Diekert and Y. Métivier. Partial commutation and traces, 1997.

[33] L. Doyen, T. A. Henzinger, B. Jobstmann, and T. Petrov. Interface theories with component reuse. In *Proceedings of the 8th ACM international conference on Embedded software*, EMSOFT '08, pages 79–88, New York, NY, USA, 2008. ACM.

[34] D. D'Souza. A logical study of timed distributed automata. 2000.

[35] D. D'Souza and P. S. Thiagarajan. Product interval automata: A subclass of timed automata. In *Proceedings of the 19th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 60–71, London, UK, 1999. Springer-Verlag.

[36] M. Griss, G. Pour, and J. Favaro. Making the transition to component-based enterprise software development overcoming the obstacles - patterns for success. In *Proceedings of the Technology of Object-Oriented Languages and Systems*, TOOLS '99, pages 527–, Washington, DC, USA, 1999. IEEE Computer Society.

[37] M. L. Griss. Software reuse: Architecture, process, and organization for business success. *Israeli Conference on Computer-Based Systems and Software Engineering*, 0:86, 1997.

[38] D. P. Guelev and D. V. Hung. Reasoning about qos contracts in the probabilistic duration calculus. *Electr. Notes Theor. Comput. Sci.*, 238(6):41–62, 2010.

[39] C. Hoare and H. Jifeng. *Unifying Theories of Programming*. Prentice Hall, 1998.

[40] D. V. Hung. Toward a formal model for component interfaces for real-time systems. In *FMICS '05: Proceedings of the 10th international workshop on Formal methods for industrial critical systems*, pages 106–114, New York, NY, USA, 2005. ACM.

[41] D. V. Hung and B. V. Anh. Model checking real-time component based systems with blackbox testing. In *Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, RTCSA '05, pages 76–79, Washington, DC, USA, 2005. IEEE Computer Society.

[42] D. V. Hung and B. V. Anh. Model checking real-time component based systems with blackbox testing. In *RTCSA '05: Proceedings of the 11th IEEE*

International Conference on Embedded and Real-Time Computing Systems and Applications, pages 76–79, Washington, DC, USA, 2005. IEEE Computer Society.

[43] D. V. Hung and P. H. Thai. Towards a template language for component-based programming, 2007.

[44] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning About Systems*. Cambridge University Press, New York, NY, USA, 2004.

[45] H. Jifeng, Z. Liu, and L. Xiaoshan. Contract-oriented component software development. Technical report, UNU/IIST, P.O. Box 3058, Macao SAR, 2003.

[46] D. K. Kaynar, N. Lynch, R. Segala, and F. Vaandrager. *The Theory of Timed I/O Automata (Synthesis Lectures in Computer Science)*. Morgan & Claypool Publishers, 2006.

[47] R. M. Keller. Parallel program schemata and maximal parallelism i. fundamental results. *J. ACM*, 20(3):514–537, 1973.

[48] R. Kemmerer. Formally specifying and verifying real-time systems. In *ICFEM '97: Proceedings of the 1st International Conference on Formal Engineering Methods*, page 112, Washington, DC, USA, 1997. IEEE Computer Society.

[49] K. G. Larsen, P. Pettersson, and W. Yi. Model-checking for real-time systems. pages 62–88, 1995.

[50] M. Leucker. On model checking synchronised hardware circuits. In J. He and M. Sato, editors, *Proceedings of the 6th Asian Computing Science Conference (ASIAN'00)*, volume 1961 of *Lecture Notes in Computer Science*, page 182–198, Penang, Malaysia, 2000. Springer, Springer.

[51] M. Leucker. Logics for mazurkiewicz traces. Technical report, PHD THESIS, LEHRSTUHL FUR INFORMATIK II, RWTH AACHEN, 2002.

[52] J. Li, S. Chen, L. Jian, and H. Zhang. A web services composition model and its verification algorithm based on interface automata. In *Proceedings of the 2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, TRUSTCOM '11, pages 1556–1563, Washington, DC, USA, 2011. IEEE Computer Society.

[53] Z. Liu, H. Jifeng, and X. Li. rcos: A refinement calculus for object systems. Technical report, Theoretical Computer Science, 2006.

[54] G. Lüttgen, W. Vogler, and S. Fendrich. Richer interface automata with optimistic and pessimistic compatibility. *Acta Inf.*, 52(4-5):305–336, June 2015.

[55] A. Mazurkiewicz. Trace theory. In *Advances in Petri nets 1986, part II on Petri nets: applications and relationships to other models of concurrency*, pages 279–324, New York, NY, USA, 1987. Springer-Verlag New York, Inc.

[56] Y. min Wang, O. P. Damani, and W. jyh Lee. Reliability and availability issues in distributed component object model (dcom), 1997.

[57] F. Mogavero. *Logics in Computer Science: A Study on Extensions of Temporal and Strategic Logics*. Atlantis Press, 2013.

[58] M. Mukund and P. S. Thiagarajan. Linear time temporal logics over mazurkiewicz traces. In *Proceedings of the 21st International Symposium on Mathematical Foundations of Computer Science*, MFCS '96, pages 62–92, London, UK, 1996. Springer-Verlag.

[59] T. A. Ngoc Hung PHAM, Viet-Ha NGUYEN and T. KATAYAMA. A minimized assumption generation method for component-based software verification. *IEICE TRANSACTIONS on Information and Systems*, E93-D(8):2172–2181, Aug. 2010.

[60] D. Peled and W. Penczek. Using asynchronous büchi automata for efficient automatic verification of concurrent systems. In *PROC. OF PSTV'95*, pages 115–130, 1996.

[61] J. L. Peterson. Petri nets. *ACM Comput. Surv.*, 9:223–252, September 1977.

[62] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS '77, pages 46–57, Washington, DC, USA, 1977. IEEE Computer Society.

[63] G. Pour. Software component technologies: Javabeans and activex. In *Proceedings of the Technology of Object-Oriented Languages and Systems*, TOOLS '98, pages 375–, Washington, DC, USA, 1998. IEEE Computer Society.

[64] G. Pour. Enterprise javabeans, javabeans & xml expanding the possibilities for web-based enterprise application development. In *Proceedings of the*

31st International Conference on Technology of Object-Oriented Language and Systems, TOOLS '99, pages 282–, Washington, DC, USA, 1999. IEEE Computer Society.

[65] J. Rumbaugh, I. Jacobson, and G. Booch, editors. *The Unified Modeling Language reference manual*. Addison-Wesley Longman Ltd., Essex, UK, UK, 1999.

[66] L. P. Serge Haddad, Fabrice Kordon and L. Petrucci, editors. *Models and Analysis for Distributed Systems*. Wiley-ISTE, 2011.

[67] J. Sifakis. Modeling real-time systems - challenges and work directions. In *In Proceedings of the 1st International Workshop on Embedded Software (EMSOFT), Lecture Notes in Computer Science*, pages 373–389. Springer Verlag, 2001.

[68] P. S. Thiagarajan. A trace based extension of linear time temporal logic. In *LICS*, pages 438–447. IEEE Computer Society, 1994.

[69] S. Tripakis, B. Lickly, T. A. Henzinger, and E. A. Lee. On relational interfaces. Technical Report UCB/EECS-2009-60, EECS Department, University of California, Berkeley, May 2009.

[70] P. Vanbekbergen, G. Goossens, and B. Lin. Modeling and synthesis of timed asynchronous circuits. In *Proceedings of the Conference on European Design Automation*, EURO-DAC '94, pages 460–465, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[71] K. Werner, editor. *ICALP '92: Proceedings of the 19th International Colloquium on Automata, Languages and Programming*, volume 623, London, UK, 1992. Springer-Verlag.

[72] S. S. Yau and B. Xia. Object oriented distributed component software development based on corba. In *Proceedings of the 22nd International Computer Software and Applications Conference*, COMPSAC '98, pages 246–251, Washington, DC, USA, 1998. IEEE Computer Society.

[73] T. Yoneda and H. Ryu. Timed trace theoretic verification using partial order reduction. In *5th International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC '99), 19-22 April 1999, Barcelona, Spain*, page 108, 1999.

[74] N. Zhan, E. Y. Kang, and Z. Liu. Component publications and compositions. In *Proceedings of the 2nd international conference on Unifying theories of programming*, UTP'08, pages 238–257, Berlin, Heidelberg, 2010. Springer-Verlag.

[75] W. Zielonka. Notes on finite asynchronous automata. *ITA*, 21(2):99–135, 1987.