PHƯƠNG PHÁP TIẾP CẬN CHÍ NH THỨC DỰA TRÊN NGÔN TỪ CHO AN TOÀN VÀ KIỂM TRA AN NINH CỦA KIỂM SOÁT CÔNG NGHIỆP HỆ THỐNG

qua

Ramesh Neupane

Một luận án

nộp một phần để hoàn thành

của các yêu cầu về mức độ

Thạc sĩ Khoa học Máy tính

Đại học Boise State

Tháng 8 năm 2022

TRƯỜNG CAO ĐẲNG ĐẠI HỌC BANG BOISE

ỦY BAN QUỐC PHÒNG VÀ PHÊ DUYỆT ĐỌC CUỐI CÙNG

của luận án được nộp bởi

Ramesh Neupane

Tiêu đề luận án: Phương pháp tiếp cận chính thức dựa trên Ontology để xác minh an toàn và bảo mật của Hệ thống điều khiển công nghiệp

Ngày thi vấn đáp cuối kỳ: 15 tháng 4 năm 2022

Những cá nhân sau đây đã đọc và thảo luận về luận án do sinh viên Ramesh Neupane nộp, và họ đã đánh giá bài trình bày và phản hồi của sinh viên đối với các câu hỏi trong kỳ thi vấn đáp cuối cùng. Họ thấy rằng sinh viên đã vượt qua kỳ thi vấn đáp cuối cùng.

Tiến sĩ Hoda Mehrpouyan	Chủ tịch, Ủy ban giám sát
Tiến sĩ Bogdan Dit	Thành viên, Ủy ban giám sát
Tiến sĩ Craig Rieger	Thành viên, Ủy ban giám sát

Sự chấp thuận đọc cuối cùng của luận án đã được Hoda Mehrpouyan, Tiến sĩ, Chủ tịch Ủy ban giám sát chấp thuận. Luận án đã được chấp thuận bởi Cao đẳng sau đại học.

CỐNG HIẾN

Luận văn này xin dành tặng cho cha mẹ và các chị em của tôi vì tình yêu thương, sự ủng hộ vô bờ bến của họ và sự khích lệ.

LỜI CẢM ƠN

Tôi muốn bày tỏ lòng biết ơn sâu sắc nhất của tôi đến người hướng dẫn của tôi Tiến sĩ Hoda Mehrpouyan vì hỗ trợ và động viên tôi trong suốt quá trình học sau đại học. Nếu không có sự giúp đỡ của cô ấy và hướng dẫn, luận văn này sẽ không thể thực hiện được.

Tôi cũng muốn cảm ơn các thành viên ủy ban của tôi, Tiến sĩ Bogdan Dit và Tiến sĩ Craig Rieger (Phòng thí nghiệm quốc gia Idaho), vì những bình luận và phản hồi mang tính xây dựng của họ đã giúp định hình phạm vi của luận án này.

Tôi mãi mãi biết ơn cha mẹ tôi, Sita Neupane và Ramkrishna Neupane, vì tình yêu thương và sự ủng hộ vô tận của họ trong suốt hành trình học tập của tôi. Tôi cũng rất biết ơn gửi đến hai chị gái xinh đẹp của tôi, Rajani và Rashila, vì đã dành cho tôi tình yêu thương vô bờ bến.

Công trình của luận án này dựa trên công trình được hỗ trợ bởi Quỹ Khoa học Quốc gia ngày Khoa Khoa học và Kỹ thuật Máy tính và Thông tin (CISE), giải thưởng số 1846493 của chương trình Không gian mạng an toàn và đáng tin cậy (SaTC): Chính thức Công cụ cho An toàn và Bảo mật của Hệ thống Kiểm soát Công nghiệp (FORENSICS), vì vậy tôi muốn cảm ơn sự ủng hộ của họ.

Cuối cùng nhưng không kém phần quan trọng, tôi muốn cảm ơn các thành viên trong phòng thí nghiệm của tôi, đặc biệt là Sean O'Toole, Chidi Agbo và Chibuzo Ukegbu đã giúp đỡ, hướng dẫn và hỗ trợ tôi trong suốt chương trình sau đại học của tôi.

TÓM TẮT

Logic điều khiển, như một phần của Hệ thống điều khiển công nghiệp (ICS), được sử dụng để điều khiển các quá trình vật lý của các cơ sở hạ tầng quan trọng như nhà máy điện, nước và khí đốt phân phối, v.v. Thông thường nhất, Bộ điều khiển logic lập trình (PLC) các quá trình này thông qua các bộ truyền động dựa trên thông tin nhận được từ cảm biến đọc. Bất kỳ vấn đề an toàn hoặc tấn công mạng nào vào các hệ thống này đều có thể gây ra thảm họa hậu quả đối với cuộc sống con người và môi trường. Trong nỗ lực cải thiện sự im lặng và bảo mật của logic điều khiển, luận án này cung cấp các thuật toán và công cụ để chính thức xác định các yêu cầu về an toàn và bảo mật liên quan đến các quy trình vật lý và lĩnh vực công nghiệp. Ngôn ngữ Ontology Web (OWL) được sử dụng để tạo ra ngữ nghĩa mối quan hệ giữa các yếu tố của quá trình công nghiệp và lập bản đồ kiến thức của đầu vào và đầu ra của logic điều khiển. Mô tả Cơ sở kiến thức Logic (DL) xuất phát từ OWL cho phép chúng ta lý luận về các khái niệm an ninh và an toàn ngữ nghĩa để đảm bảo tính nhất quán của chúng. Tiếp theo, các thông số kỹ thuật chính thức này được dịch sang Timed Truy vấn Logic cây tính toán (TCTL) để xác minh logic điều khiển được mô hình hóa trong UPPAAL như một mạng lưới các máy tự động định thời (TA). Trong phần thứ hai của luận án, các điều kiện biên được kiểm tra để thực hiện xác minh mô hình. Biên giới kiểm tra là điều cần thiết trong ICS vì các giá trị đọc của cảm biến và bộ truyền động cần nằm trong phạm vi an toàn để đảm bảo hoạt động ICS an toàn.

Để chứng minh khái niệm, chúng tôi đã nghiên cứu một phần của quy trình hóa học công nghiệp

để thực hiện cách tiếp cận được đề xuất của chúng tôi. Kết quả thử nghiệm trong công trình này đã chứng minh rằng phương pháp đề xuất phát hiện ra sự không nhất quán trong các yêu cầu về an toàn và bảo mật và đảm bảo rằng các biến đầu vào và đầu ra của logic điều khiển nằm trong một phạm vi an toàn và bảo mật. Nghiên cứu hiệu suất của các triển khai của chúng tôi cho thấy rằng thời gian tăng theo tuyến tính với số tiên đề trong bản thể học và số của các lần lặp lại trong mô phỏng mô hình TA. Do đó, cách tiếp cận có thể mở rộng để có thực hiện thực tế để giúp các kỹ thuật viên và kỹ sư tạo ra một môi trường an toàn hơn và logic điều khiển an toàn hơn cho các quy trình ICS.

MỤC LỤC

DANH SÁCH CÁC HÌNH ẢNH

DANH SÁCH CÁC BẢNG

DANH SÁCH CÁC TỪ VIẾT TẮT

Logic cây tính toán CTL

Mô tả DL Logic

Khối chức năng FB

Sơ đồ khối chức năng FBD

Logic bậc nhất FOL

Máy tự động trạng thái hữu hạn FSA

Hệ thống điều khiển công nghiệp ICS

Công nghệ thông tin CNTT

Logic thời gian tuyến tính LTL

Công nghệ vận hành OT

Ngôn ngữ Ontology Web OWL

Bộ điều khiển logic lập trình PLC

Xác minh mô hình biểu tượng SMV

Ngôn ngữ quy tắc web ngữ nghĩa SWRL

TA Tự động hẹn giờ

Logic cây tính toán thời gian TCTL

Ngôn ngữ mô hình hóa thống nhất UML

PHẦN I

CHƯƠNG 1:

GIỚI THIỆU

Nhiều ngành công nghiệp và tiện ích công cộng sử dụng Hệ thống điều khiển công nghiệp (ICS), chẳng hạn như bộ điều khiển logic lập trình (PLC), hệ thống điều khiển phân tán (DCS), giám sát hệ thống điều khiển và thu thập dữ liệu (SCADA) và hệ thống thiết bị an toàn (SIS) để giám sát và kiểm soát các quy trình tự động hóa và hoạt động an toàn của chúng. Với những tiến bộ trong công nghệ thông tin (IT), trí tuệ nhân tạo (AI) và robot, máy tính các nhà khoa học và kỹ sư tiếp tục phát triển các ứng dụng ICS mới để cải thiện chất lượng và hiệu quả, cũng dẫn đến nhiều mạng lưới thương mại hơn sử dụng Inter- mạng để mở rộng việc chia sẻ thông tin để có hiệu quả hơn. Thật không may, điều này nâng cao các mối đe dọa an ninh mạng và nhu cầu về các kỹ thuật và công cụ để các nhà thiết kế hệ thống có thể xác định và thực hiện các yêu cầu bảo mật và an toàn mạnh mẽ hơn cho những yêu cầu quan trọng này hệ thống.

Mục tiêu của nghiên cứu này là thiết kế và phát triển các thuật toán và công cụ để cải thiện sự an toàn và bảo mật của ICS, là phần mềm phức tạp và có tính kết nối cao và hệ thống phần cứng. Những hệ thống này được coi là quan trọng và cần thiết cho phúc lợi của xã hội. Do đó, bất kỳ loại vấn đề hoặc mối đe dọa an ninh mạng nào cũng có thể dẫn đến sự tàn phá đáng kể, ảnh hưởng đến hàng triệu người. Xem xét mức độ nghiêm trọng của hậu quả, điều cần thiết là phải phát triển sự hiểu biết về cách tích hợp an toàn và các yêu cầu bảo mật vào phần mềm điều khiển. Chúng tôi trả lời câu hỏi này bằng

phát triển một cơ sở tri thức (KB) có thể phục vụ như một thuật ngữ tham chiếu cho yêu cầu của phần mềm điều khiển công nghiệp. Ngôn ngữ chính thức dựa trên ontology vốn có hỗ trợ công cụ xác thực tự động để kiểm tra tính nhất quán và sự hoàn thiện của các thông số kỹ thuật. Điều này dẫn đến việc tìm ra các vấn đề trong giai đoạn đầu của phân tích yêu cầu, điều này rất quan trọng trong trường hợp ứng dụng công nghiệp vì việc phát hiện ra vấn đề ở giai đoạn phát triển sau có thể gây ra hậu quả nghiêm trọng.

Các yêu cầu mâu thuẫn giữa các thuộc tính an toàn/thời gian thực và bảo mật nhu cầu của hệ thống có thể gây ra một số lỗ hổng trong hệ thống. Phát hiện những xung đột trong giai đoạn đầu làm tăng cả tính an toàn và bảo mật của hệ thống. Đó là tại sao đã có nhiều nghiên cứu về việc xác định xung đột trong các yêu cầu khác nhau [1, 2, 3]. Cách tiếp cận chính của những nghiên cứu đó là sử dụng một biểu diễn chung cho tất cả các yêu cầu để chúng có thể được dịch sang ngôn ngữ chính thức để sử dụng trong quá trình xác thực. Sau khi kiểm tra sự không nhất quán trong đặc tả yêu cầu, những yêu cầu đó phải được xác minh với hệ thống đã triển khai. Trong bối cảnh của ICS, để kiểm tra xem hệ thống có đáp ứng các yêu cầu về an toàn và chức năng hay không, có nhiều phương pháp thử nghiệm khác nhau [4, 5, 6, 7] đang được áp dụng. Vì chúng ta đang giải quyết với các ứng dụng quan trọng trong ICS, các phương pháp xác minh chính thức được khám phá trong [8, 9, 10, 10, 11, 12, 13, 14] đáng tin cậy hơn vì nó thực hiện một cách đầy đủ khám phá có hệ thống mô hình toán học biểu diễn hệ thống.

Để chính thức hóa các thông số kỹ thuật và yêu cầu, chúng tôi sử dụng Mô tả Logic (DL) là công thức toán học, là một phần của logic bậc nhất (FOL) và là một phần của gia đình hình thức biểu diễn tri thức. Chúng tôi sẽ xây dựng trên nghiên cứu [15] đã sử dụng DL để xác minh thời gian chạy nhằm xác định mối quan hệ giữa các cuộc tấn công khác nhau [16], khám phá phạm vi của một cuộc tấn công cụ thể [17] và trong

phát hiện xâm nhập [18]. Đóng góp của chúng tôi là đề xuất hình thức DL để xây dựng
đặc điểm kỹ thuật của các thực thể và mối quan hệ của chúng và thiết kế và phát triển một thuật ngữ
chứa đựng thông tin cần thiết về các quy trình mà logic điều khiển là
phải quản lý, và các yêu cầu về an toàn và bảo mật của nó. Ontology được xây dựng hoạt động
như một cơ sở kiến thức để xác minh các yêu cầu. Hai thuật toán hoạt động
đã được phát triển theo cách tiếp cận này để xác định các khái niệm an toàn/bảo mật và
yêu cầu nhất quán để tạo ra một ontology ICS. Thứ hai, DL-reasoner được sử dụng để
đảm bảo rằng các yêu cầu và quy tắc về thông số kỹ thuật là nhất quán trước khi xác minh
quá trình kiểm tra logic điều khiển được bắt đầu. Đối với quá trình kiểm tra mô hình, logic điều khiển
được trích xuất từ chương trình PLC và được mô hình hóa như một Automaton theo thời gian (TA), cùng với
với các yêu cầu từ ontology, được dịch sang Timed Computational
Tree Logic (TCTL) và được thử nghiệm để đảm bảo tính an toàn và bảo mật.

Trong nghiên cứu này, một phần của quá trình hóa học công nghiệp đã được thực hiện để minh họa
cách tiếp cận. Chúng tôi đã xây dựng một thuật ngữ học có liên quan thông qua việc phân tích chi tiết các chất hóa học
quá trình sử dụng trình soạn thảo ontology phổ biến, Prot´eg´e [19]. Trình suy luận DL Pellet [20]
được hỗ trợ bởi trình biên tập đã được sử dụng để kiểm tra sự không nhất quán và tính đầy đủ
của ontology. Chúng tôi đã có thể chứng minh rằng OWL-DL [21] cùng với Semantic
Các quy tắc của Ngôn ngữ quy tắc web (SWRL) [22] có thể được sử dụng để chính thức hóa
yêu cầu và đặc điểm kỹ thuật của quá trình hóa học. Chúng tôi tự động chuyển đổi
các quy tắc được thể hiện trong SWRL vào các truy vấn TCTL bằng cách sử dụng các toán tử thời gian như
như cuối cùng và toàn cầu. Công thức TCTL sau đó được đưa vào UPPAAL [23]
kiểm tra mô hình. Trong UPPAAL, trước tiên chúng tôi mô hình hóa logic điều khiển được triển khai trong
Chương trình PLC với các thành phần bổ sung cần thiết cho logic như TA. UPPAAL
có khả năng kiểm tra các thuộc tính TCTL trên một mô hình TA nhất định và cung cấp khả năng thực thi

dấu vết giúp xác định các vấn đề trong logic hệ thống. Kết quả thử nghiệm trên
quá trình hóa học cho thấy phương pháp đề xuất phát hiện ra sự không nhất quán trong
yêu cầu đặc tả với ontology và DL-reasoner. Hơn nữa, chúng tôi đã có thể
để chứng minh rằng các yêu cầu nhất quán sau đó có thể được sử dụng để xác minh thêm
logic điều khiển với sự trợ giúp của trình kiểm tra mô hình UPPAAL và chuyển đổi TCTL
truy vấn vào công thức TCTL.

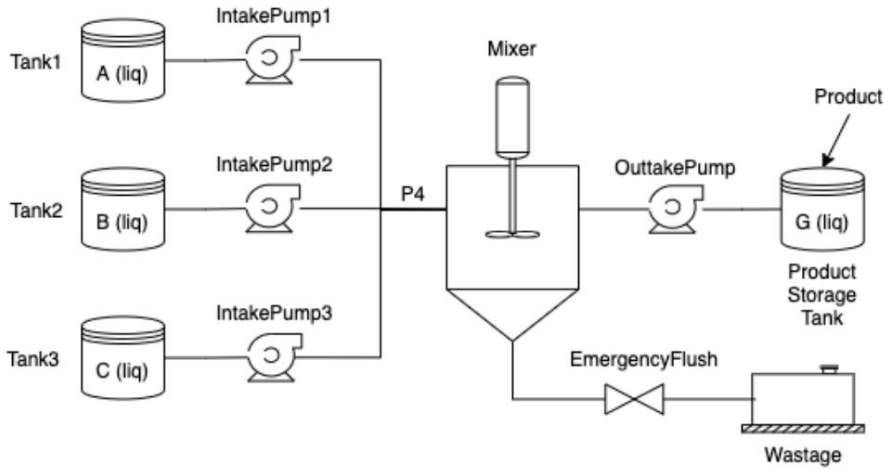
Tóm lại, mục đích của luận án này là chính thức hóa các quy định về an toàn và bảo mật.
thông số kỹ thuật yêu cầu, để kiểm tra tính nhất quán giữa chúng và khai thác điều đó
thông tin chính thức để xác minh logic điều khiển. Các câu hỏi nghiên cứu của luận án này
như sau:

1. Làm thế nào để chính thức hóa và tạo ra cơ sở tri thức để những điểm không nhất quán trong
   các yêu cầu về an toàn và bảo mật được tự động phát hiện?

2. Làm thế nào để xác minh và xác nhận chính thức logic điều khiển để đảm bảo nó đáp ứng
   yêu cầu về an toàn, bảo mật và chức năng?

CHƯƠNG 2:

QUÁ TRÌNH HÓA HỌC ĐANG ĐƯỢC NGHIÊN CỨU



Hình 2.1: Quá trình hóa học

Để chứng minh khái niệm và trong toàn bộ bài báo, chúng tôi sẽ sử dụng hóa chất quá trình (CE) là một phiên bản được sửa đổi một phần của các quá trình công nghiệp trong nước tiện ích. Quy trình được chọn có bốn thành phần chính: bể chứa, đầu vào và máy bơm thoát nước, máy trộn và máy xả như thể hiện trong Hình 2.1. Quy trình hóa học chủ yếu liên quan đến phản ứng hóa học không thể đảo ngược của ba chất phản ứng lỏng tạo ra một sản phẩm dạng lỏng:

$$A(liq) + B(liq) + 2C(liq) = G(liq)$$

Trong quá trình này, lò phản ứng trộn nhận vật liệu từ ba thùng cung cấp

(Tank1, Tank2 và Tank3) thông qua các máy bơm (IntakePump1, IntakePump2, In-
takePump3). Mỗi bơm hút có tốc độ 2 lít mỗi giây và bơm xả
máy bơm có tốc độ 3 lít mỗi giây. Đầu tiên, IntakePump1 bơm 4 lít hóa chất
cal A từ Tank1 đến Mixer. Tiếp theo là việc chuyển 4 lít hóa chất
B từ Tank2 và 8 lít hóa chất C từ Tank3. Máy trộn bắt đầu chạy im-
trung gian với IntakePump2 và IntakePump3. Các thành phần sau đó được trộn lẫn cho
4 giây trước khi được chuyển đến một bể chứa sản phẩm nhàn rỗi với sự trợ giúp của
máy bơm OuttakePump. Đầu vào của người dùng RunProcess khởi động và tạm dừng hóa chất
quá trình, trong khi công tắc EmergencyStop đặt lại hệ thống và chạy chế độ khẩn cấp
xả van trong 5 giây.

Đối với quá trình này, việc cung cấp hóa chất cho phản ứng hóa học phải được thực hiện chính xác.
được thực hiện bằng cách đổ đầy các thùng chứa đầu vào. Hóa chất được cung cấp cho thùng chứa dựa trên
cách sử dụng, trong đó đặt ra một ngưỡng nhất định để đổ đầy bình, để bình không bị cạn.
Một số hạn chế về an toàn đối với các quy trình hóa học được thảo luận dưới đây cùng với
những hàm ý của nó.

1. Máy trộn không nên chạy nếu không có hóa chất
    - Chạy máy trộn rỗng sẽ làm tăng nhiệt độ trong máy trộn,
      cuối cùng có thể dẫn đến một vụ nổ
2. Cả ba máy bơm hút không nên chạy cùng lúc
    - Lưu lượng lớn hơn trên đường ống P4 có thể khiến đường ống bị vỡ
3. Trong lò phản ứng trộn hóa chất A không được quá 6 lít
    - Phản ứng tạo ra nhiều nhiệt hơn, gây ra sự cố cho máy trộn

4. Máy trộn và máy bơm OuttakePump không nên chạy cùng lúc

- Sản phẩm cuối cùng vô dụng

CHƯƠNG 3:

BỐI CẢNH VÀ CÁC CÔNG TRÌNH LIÊN QUAN

ICS đã bị cô lập khỏi các mạng bên ngoài trong nhiều năm, vì vậy việc xác minh và thử nghiệm tập trung nhiều hơn vào các thông số kỹ thuật chức năng và yêu cầu an toàn. Làm thế nào- bao giờ hết, với sự ra đời của Công nghiệp 3.0 và hiện nay là 4.0 [24], nhu cầu về dữ liệu thời gian thực từ Công nghệ vận hành (OT) đã khiến chúng ta không còn lựa chọn nào khác ngoài việc kết nối cơ sở hạ tầng quan trọng cho Internet. Với sự cởi mở và phức tạp ngày càng tăng của hệ thống, nó đã trở nên dễ bị tổn thương hơn trước các mối đe dọa từ cả bên trong và bên ngoài kẻ thù. Các lỗ hổng được báo cáo trong ICS đã tăng gấp mười lần từ năm 2010 đến 2017 [25]. Không có gì ngạc nhiên khi số lượng các cuộc tấn công cũng tăng lên đáng kể khi được thảo luận trong báo cáo [26]. Như đã đề cập trong bài báo, điểm yếu phổ biến nhất trong ngành là ranh giới bảo vệ yếu giữa mạng doanh nghiệp và ICS.

Sau khi truy cập vào hệ thống, kẻ tấn công cố ý sửa đổi các giá trị của PLC các biến để kiểm soát và phá hoại cơ sở hạ tầng. Do sự tích hợp nhanh chóng của CNTT và OT, có sự tách biệt rõ ràng về kiến thức trong hệ thống CNTT và OT [26].

Trong một số tài liệu, người ta cũng đề cập rằng ở một số khía cạnh nhất định của hệ thống, nhân viên từ các miền khác nhau xem các bộ phận khác của ICS như hộp đen. Vì vậy, khoảng cách này không chỉ tồn tại giữa CNTT và OT, mà còn bên trong các miền OT. Do đó, chúng tôi có thể nói rõ ràng rằng có một khoảng cách kiến thức giữa Công nghệ thông tin (CNTT) và OT mặc dù sự hợp tác của họ rất quan trọng đối với một ICS kiên cường.

Do khoảng cách này trong bảo mật CNTT và hoạt động OT, có thể xảy ra xung đột trong yêu cầu về an ninh và an toàn và các thông số kỹ thuật chức năng. Vì vậy, chúng ta cần phải có một cơ chế để kiểm tra tính nhất quán giữa các yêu cầu và để xác minh rằng logic điều khiển đáp ứng các yêu cầu này. Phần còn lại của chương này cung cấp một xem xét các cách tiếp cận khác nhau được áp dụng trong việc kiểm tra tính nhất quán của các yêu cầu và sử dụng chúng trong việc xác minh logic điều khiển.

3.1 Xác minh chính thức các thông số kỹ thuật

Trong khi xác minh và xác thực logic điều khiển được triển khai trong ICS, điều cần thiết là phải có yêu cầu nhất quán và đầy đủ. Yêu cầu nhất quán là những yêu cầu về an toàn và các yêu cầu về bảo mật không xung đột với nhau.

Kiểm tra tính nhất quán của các thông số kỹ thuật yêu cầu không phải là một ý tưởng mới. Heitmeyer et al. trong [1], đã mô tả một kỹ thuật kiểm tra tính nhất quán tự động của yêu cầu thông số kỹ thuật. Phương pháp giải thích trong bài báo được thiết kế đặc biệt để kiểm tra tính nhất quán của các yêu cầu được thể hiện trong bảng Giảm chi phí phần mềm (SCR) ký hiệu. Các chi tiết bảng được mô hình hóa trong Tự động trạng thái hữu hạn (FSA) cho xác minh thêm. Li et al. trong [2] lập luận rằng Ngôn ngữ mô hình hóa thống nhất (UML) các mô hình chủ yếu được sử dụng để kiểm tra tính nhất quán trong kỹ thuật phần mềm hơn là Mô hình SCR, do đó, họ đề xuất một cách logic chính thức để kiểm tra các yêu cầu được mô hình hóa trên UML. Mặc dù UML cung cấp hỗ trợ tốt cho việc chỉ định các yêu cầu, nhưng nó không hỗ trợ suy luận logic để kiểm tra các mô hình. Ngoài ra, một số tập hợp con của UML hỗ trợ kiểm tra mô hình gặp phải tình trạng không thể quyết định được trong suy luận [27]. Mặt khác, phương pháp tiếp cận dựa trên ontology sử dụng DL, hỗ trợ suy luận logic để phát hiện xung đột giữa các câu lệnh. Ngoài ra, DL là một phần có thể quyết định của Logic bậc nhất (FOL), do đó có thể quyết định được hầu hết thời gian.

Không chỉ trong kỹ thuật phần mềm, việc kiểm tra tính nhất quán của các thông số kỹ thuật đã được cũng được khám phá ở các khu vực khác. Kamsu-Foguem et al. [3] đã đề xuất một khái niệm khuôn khổ dựa trên đồ thị để kiểm tra chính thức sự tuân thủ của việc xây dựng để xác minh rằng chúng đáp ứng các tiêu chuẩn hoặc quy định nhất định. Trong bài báo này, tác giả đã mô hình hóa các yêu cầu xây dựng và các sự kiện về thông tin xây dựng trong một đồ thị khái niệm sử dụng công cụ trực quan dựa trên đồ thị có tên là CoGui [28]. Sử dụng công cụ, sử dụng các mối quan hệ bao hàm giữa đồ thị khái niệm để lý luận [29], đồ thị mô hình sau đó được lý giải và xác thực để kiểm tra tính tuân thủ.

Mặc dù cách tiếp cận này tự động kiểm tra tính nhất quán của các yêu cầu thông số kỹ thuật đã được sử dụng trong nhiều lĩnh vực, chúng tôi không tìm thấy nghiên cứu tương tự nào tập trung vào ICS. Do đó, chúng tôi đề xuất một phương pháp dựa trên ontology chính thức mới để kiểm tra sự nhất quán giữa các yêu cầu sử dụng lý luận dựa trên DL phổ biến. Đối với điều này cách tiếp cận này, chúng ta cần xây dựng một thuật ngữ ICS với các thành phần an toàn và bảo mật. Sau khi xây dựng một thuật ngữ, nó có thể được sử dụng không chỉ để kiểm tra xung đột mà còn để xác minh logic điều khiển.

3.2 Kiểm tra và thử nghiệm Logic điều khiển

Trong phần này, trước tiên chúng ta xem xét các phương pháp hiện có được sử dụng để kiểm tra logic điều khiển trong phần mềm PLC. Kiểm tra chức năng của các mô-đun logic điều khiển, với các trường hợp kiểm tra đầu vào được tạo ra thường xuyên, là kỹ thuật được sử dụng phổ biến nhất trong công nghiệp ứng dụng kiểm soát quy trình [4, 5]. Các trường hợp thử nghiệm được thiết kế dựa trên an toàn trước đó và kinh nghiệm, báo cáo về lỗi bảo mật và chuyên môn của người kiểm tra [6]. Trong khi các loại này của các phương pháp tiếp cận theo trường hợp thử nghiệm dễ triển khai, chúng thiếu thử nghiệm hoàn chỉnh Logic PLC và có khả năng cao là thiếu sót các lỗi chương trình quan trọng đã chưa được phát hiện trước đây [4, 30]. Để khắc phục vấn đề này và tự động

Để kiểm tra chính xác hành vi động của PLC, người ta sử dụng các công cụ dựa trên mô phỏng.

Lee et al. đã phát triển một môi trường thử nghiệm dựa trên mô phỏng [6] xem xét đặc điểm của hệ thống PLC quan trọng về an toàn như chu kỳ quét, kiến trúc CPU và bản đồ bộ nhớ của bộ vi xử lý PLC. Để mô phỏng chính xác phần mềm PLC hành vi và kiểm tra xem đầu ra có chính xác hay không dựa trên chương trình cụ thể đầu vào và trạng thái bên trong của chương trình, tác giả đã giới thiệu một nền tảng thử nghiệm phần mềm nắm bắt cả các điều kiện bên trong và bên ngoài của chu kỳ quét PLC. công cụ dựa trên mô phỏng khác, SIVAT [7] dịch Biểu đồ khối chức năng (FBD) vào C và thực hiện thử nghiệm chức năng. Mặc dù các công cụ dựa trên mô phỏng này và các trình mô phỏng [31, 32] cho phép thử nghiệm các tình huống cụ thể trong môi trường được kiểm soát chỉ định ngữ cảnh thời gian chạy, không nằm trong phạm vi kiểm tra các cuộc tấn công chưa biết và thất bại.

Để khắc phục những điểm yếu của thử nghiệm thủ công và dựa trên mô phỏng, các nhà nghiên cứu đã sử dụng phương pháp xác minh chính thức để tăng chất lượng an toàn quan trọng Chương trình PLC bằng cách phát hiện nhiều lỗi hơn [8, 9, 10, 10, 11]. Sau nỗ lực ban đầu của Palshikar và Nori [33] sử dụng logic thời gian để mô hình hóa chính thức chương trình PLC, nhiều nhà nghiên cứu đã dựa vào các phương pháp chính thức để xác nhận và xác minh Phần mềm PLC.

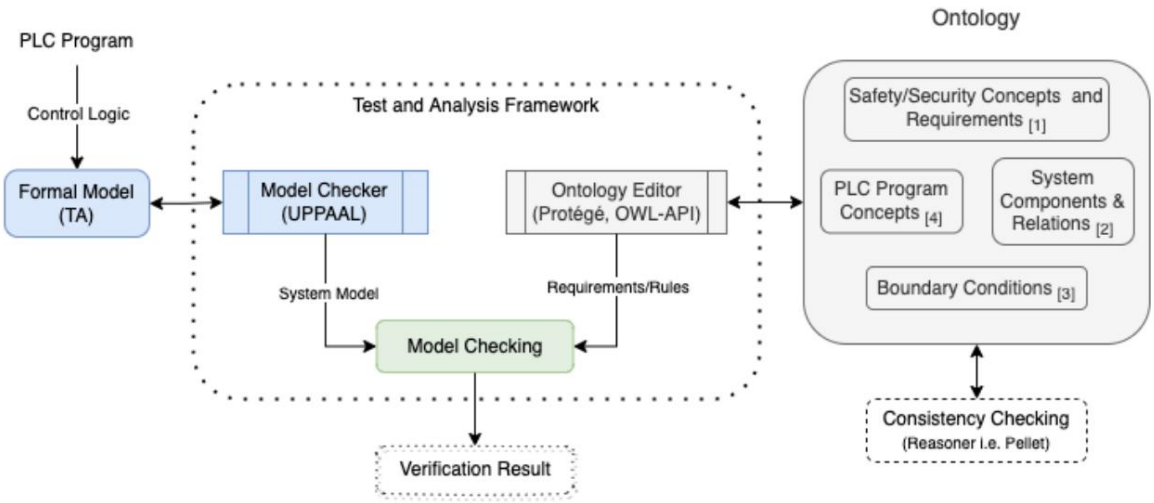
Rausch và Krogh [12] đã trình bày một phương pháp để dịch logic chương trình PLC vào các mô-đun và thông số kỹ thuật xác minh mô hình tượng trưng (SMV) vào Computa- Logic cây phân tử (CTL), được xác minh thêm bằng cách sử dụng công cụ SMV. Tương tự như vậy, Soliman et al. trong [13], đã dịch chương trình Khối chức năng PLC (FB) sang UP- PAAL [23] Mô hình TA để xác minh chính thức các yêu cầu an toàn được chính thức hóa trong thời gian logic ral. Cách tiếp cận này là trừu tượng hóa các mô hình của hệ thống và

kiểm tra các vấn đề an toàn là điều tuyệt vời nếu tất cả các yêu cầu và thông số kỹ thuật được đưa ra đều hoàn chỉnh và nhất quán. Hơn nữa, các mô hình được tạo ra cụ thể hơn cho behành vi của chương trình điều khiển, và do đó nó không giải quyết được mối quan hệ nhân quả của chương trình và các thành phần khác trong ICS. Ngược lại, nghiên cứu của chúng tôi sử dụng một cơ sở kiến thức cung cấp các thành phần liên quan đến ICS và mối quan hệ tàu giữa chúng. Do đó, nó đảm bảo rằng chương trình của bộ điều khiển đồng ý với các thành phần khác của hệ thống.

CHƯƠNG 4:

PHƯƠNG PHÁP

Phần này cung cấp tổng quan chung và động lực của phương pháp chúng tôi đề xuất-
để xác minh rằng logic chương trình PLC đáp ứng các yêu cầu về an toàn và bảo mật
tính chất. Phác thảo chung của phương pháp luận được thể hiện như trong Hình 4.1.



Hình 4.1: Tổng quan về phương pháp tiếp cận được đề xuất

Trong phần đầu tiên của phương pháp luận, một thuật ngữ học dựa trên DL được thiết kế
và phát triển. Thuật ngữ được đề xuất được thiết kế để bao gồm 1- an toàn/bảo mật
các khái niệm và yêu cầu, 2- các thành phần hệ thống liên quan đến logic điều khiển, 3- một số
giá trị và điều kiện biên giới, và 4- khái niệm chương trình PLC. Với sự trợ giúp của
người lý luận tology, chúng tôi kiểm tra tính nhất quán và đầy đủ của các yêu cầu và

thông số kỹ thuật.

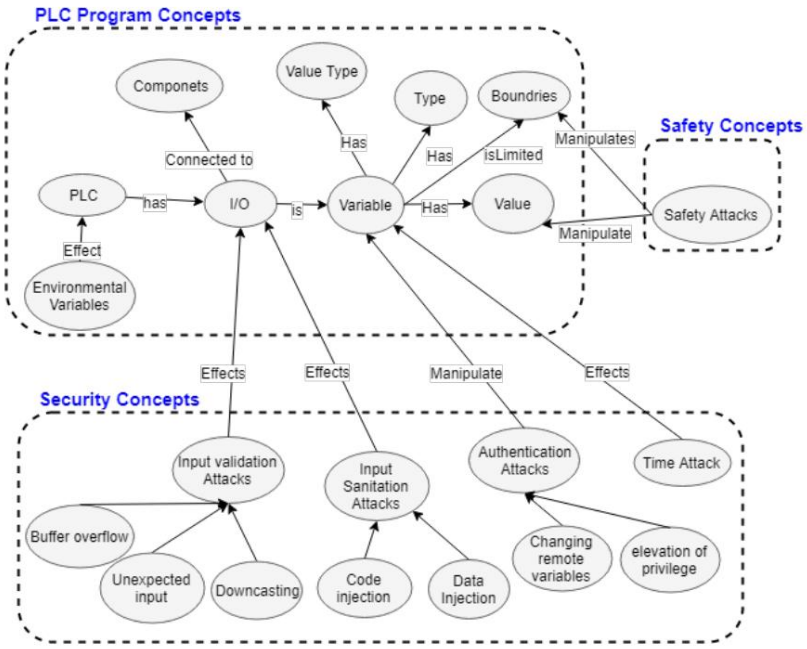
Trong phần thứ hai của phương pháp tiếp cận được đề xuất, logic điều khiển và mỗi các thành phần trong các quy trình được quản lý bởi bộ điều khiển được mô hình hóa bằng cách sử dụng một ngôn ngữ mô hình hóa chính thức. Mô hình đã dịch của logic điều khiển sau đó được sử dụng trong trình kiểm tra mô hình.

Cuối cùng, có mô hình chính thức và bản thể của các yêu cầu, việc xác minh tion được thực hiện. Các yêu cầu nhất quán và đầy đủ được đưa ra bởi ontology được dịch sang ngôn ngữ yêu cầu của trình kiểm tra mô hình, và sau đó mô hình được kiểm tra theo các yêu cầu đó thông qua kiểm tra mô hình. Các chi tiết của việc thực hiện sẽ được thảo luận thêm trong phần này.

Tại sao lại là Ontology? Trong nhiều tác phẩm trước đây, các yêu cầu được dịch thành máy tự động được kiểm tra tính nhất quán bằng cách sử dụng kiểm tra mô hình [34], hoặc các thông số kỹ thuật yêu cầu được mô hình hóa theo logic bậc nhất và sau đó được xác minh bằng SMT-Bộ giải quyết cho tính nhất quán [35]. Các cách tiếp cận này không mở rộng được đến các tập hợp yêu cầu lớn và nó cũng không xem xét mô hình hệ thống khi xem xét. Mặt khác, một thuật ngữ có khả năng mở rộng và xem xét mô hình hệ thống và các thông số kỹ thuật trong phân tích ysis. Nó đủ biểu cảm để đại diện cho hầu hết các yêu cầu của ICS. Đối với kiểm tra tính nhất quán, chúng ta không cần phải chuyển đổi nó sang các dạng logic khác vì các bộ suy luận như Hermit, Pellet, v.v. có sẵn cho ontology dựa trên DL.

Tại sao kiểm tra mô hình? Kiểm tra mô hình [36] là một phương pháp chính thức, trong đó thuộc tính hành vi mong muốn được xác minh so với mô hình hệ thống thông qua một biểu tượng khám phá không gian trạng thái của hệ thống. Ưu điểm chính của việc kiểm tra mô hình là không giống như các cách tiếp cận khác như logic và chứng minh định lý, nó không yêu cầu người dùng giám sát và chuyên môn trong các ngành toán học vì nó hoàn toàn tự động

[37]. Ngoài ra, nếu mô hình không đáp ứng được tính chất mong muốn, nó sẽ giúp gỡ lỗi bằng cách cung cấp một phản ví dụ, một thực thi của mô hình hệ thống vi phạm tài sản [38].



Hình 4.2: An toàn, bảo mật và thuật ngữ chương trình PLC

4.1 Bản thể học

Để xây dựng thuật ngữ trong Hình 4.2, thông tin chi tiết về ngành công nghiệp các quá trình đang được nghiên cứu là cần thiết. Kiến thức về miền đó trước tiên được chính thức hóa dựa trên ngôn ngữ chính thức, Ngôn ngữ Web Ontology (OWL)-DL [21] và sau đó nó là được thêm vào cơ sở kiến thức. OWL-DL có bộ công cụ phong phú có sẵn để tạo/chỉnh sửa một ontology trong khi có thể kiểm tra tính nhất quán của các thành phần của nó. Ngoài ra với điều đó, nó cũng có thể quyết định được với khả năng biểu đạt tối đa [39].

Trong OWL, lớp có ý nghĩa tương tự như khái niệm trong Logic mô tả (DL) và thuộc tính đại diện cho vai trò. Như được mô tả trong Hình 4.3, bước đầu tiên trong việc triển khai

Hình 4.3: Các bước xây dựng Ontology

	Kiến thức hệ thống (Định nghĩa lớp)	Định nghĩa chính thức (DL)
1	Bơm là một thành phần	Bơm Thành phần
2	Xả là một cái bơm	Xả Bơm
	IntakePump là một máy bơm	Bơm nạp Bơm
3 4	PLC là một bộ điều khiển	Bộ điều khiển PLC
5	Hóa chất là một vật liệu	Hóa học Vật liệu
6	IntakeTank và InputTank giống nhau	IntakeTank ≡ InputTank

Bảng 4.1: Định nghĩa lớp phân cấp của các thành phần ICS trong DL

thuật ngữ bắt đầu bằng việc xác định các lớp logic An toàn, bảo mật và kiểm soát và những ràng buộc và hạn chế của chúng (Hộp thuật ngữ (TBoxes)). Tiếp theo, chúng ta tạo các trường hợp của các lớp đó, cũng được gọi là Hộp khẳng định (ABox). Do đó, cơ sở kiến thức (KB) về cơ bản là một cặp (T, A), trong đó T là một Hộp T và A là một Hộp A. Phần còn lại của phần này giải thích các khối xây dựng của việc lập bản đồ logic điều khiển và các yêu cầu về an toàn và bảo mật của quá trình kiểm soát công nghiệp.

4.1.1 Thiết kế hệ thống và quy trình Ontology

Sử dụng DL, các thành phần được định nghĩa là một lớp phân cấp với sự trợ giúp của lớp con mối quan hệ tổng hợp. Ví dụ, Bảng 4.1 cho thấy mối quan hệ tổng hợp giữa các lớp của một nhà máy xử lý hóa chất. Bao hàm ( ) và Định nghĩa hoặc Tương đương lence (≡) là hai toán tử quan hệ chính được sử dụng để xác định các lớp.

Tương tự như vậy, sau khi xác định các thành phần cần thiết như một lớp, chúng ta cần xác định mối quan hệ và các thuộc tính dữ liệu của các lớp đó. Các mối quan hệ mô tả mối quan hệ giữa các thành phần và thuộc tính dữ liệu giải thích và gán một giá trị theo nghĩa đen

	Kiến thức hệ thống (Mối quan hệ và Thuộc tính)	Định nghĩa chính thức (DL)
	Bơm bơm vật liệu ra ngoài	pumpsOut(Bơm, Vật liệu)
1 2	Bể chứa vật liệu	containsMaterial(Bể chứa, Vật liệu)
3	Bình chứa có dung tích 20 lít	hasCapacity(Bình chứa, 20)

Bảng 4.2: Định nghĩa mối quan hệ và thuộc tính của các thành phần ICS trong DL

với một thực thể. Mối quan hệ có thể được chính thức hóa bằng cách thêm chúng vào như một vai trò liên quan cho hai cá nhân, cũng được gọi là thuộc tính đối tượng trong OWL. Các thuộc tính của các thành phần, tức là tốc độ, công suất, v.v., có thể được định nghĩa chính thức là các thuộc tính kiểu dữ liệu. Các thuộc tính dữ liệu có thể là các biến có kiểu dữ liệu int, float, v.v. Các biến này có thể có các giá trị là hằng số, có thể lựa chọn hoặc liên tục. Nếu giá trị là Constant(x) thì giá trị cố định của x phải được đưa ra. Nếu giá trị là Selectable(x) thì x là một tập hợp các giá trị chỉ định các tùy chọn lựa chọn có thể. Nếu giá trị là một Continuous(x) thì x phải chỉ định phạm vi có thể chấp nhận được. Như chúng ta có thể thấy trong Bảng 4.2, thuộc tính quan hệ pumpsOut mô tả mối quan hệ giữa Pump và Vật liệu. Tương tự như vậy, hasCapacity là một thuộc tính dữ liệu và nó liên quan giữa lớp tank và giá trị số nguyên theo nghĩa đen là 20.

4.1.2 Ontology về An toàn và Bảo mật

Để phát triển một hệ thống an toàn, trước tiên chúng ta cần phân tích các mối nguy hiểm và các biện pháp tương ứng. yêu cầu ở cấp độ hệ thống. Trong thuật ngữ của chúng tôi, các khái niệm an toàn được định nghĩa dựa trên về hai nhóm kỹ thuật phân tích mối nguy; dựa trên lỗi và dựa trên hệ thống [40]. Các phương pháp dựa trên lỗi tập trung vào tác động của lỗi thành phần đơn lẻ, tức là lỗi phân tích cây (FTA), trong khi các phương pháp dựa trên hệ thống dựa trên phân tích chi tiết thiết kế hệ thống và các thông số hệ thống. Dựa trên phân tích nguy cơ, an toàn yêu cầu được tạo ra.

Các yêu cầu về an toàn có thể được thêm vào ontology dưới dạng các ràng buộc. Các ràng buộc là được xây dựng như những điều kiện cần và đủ để một thành viên thuộc về một giai cấp. Trong DL, các điều kiện này là các tổ hợp Boolean của các thuộc tính cần thiết cho một lớp hoặc mối quan hệ với các thành viên khác và tài sản của họ. Ví dụ, sự an toàn Bể chứa nguyên liệu đầu vào chỉ nên chứa nguyên liệu đầu vào.

Đầu vàoT ank(t1) ≡ T ank(t1) chứaMaterial(t1, m1) Đầu vàoMaterial(m1)

sử dụng tên lớp Tank và InputMaterial và thuộc tính đối tượng con- tainsMaterial cũng như một liên từ lớp ( ). Chúng ta cũng có thể biểu thị số lượng những ràng buộc mà chúng ta có thể giới hạn số lượng thực thể thuộc về một lớp nhất định. Ví dụ, một bể chứa chỉ có thể chứa một loại vật liệu có thể được mô tả như sau:

Tank(t1) ≤ 1 chứaVật liệu(t1, m1) Vật liệu(m1)

Thuật ngữ bảo mật chứa các khái niệm bảo mật cụ thể cho ICS. Các nguồn để những khái niệm chung này là các chuyên gia bảo mật ICS, công trình đã xuất bản, cuộc tấn công trước đó báo cáo, v.v. Các khái niệm bảo mật chung có thể được định nghĩa chính thức trong DL như là quan hệ giả định. Ví dụ, các khái niệm như ”Tấn công xác thực đầu vào là một cuộc tấn công” và “Tràn bộ đệm là một cuộc tấn công xác thực đầu vào” có thể được chính thức hóa như theo sau.

InputV alidationAttack Tấn công

BufferOverflow InputV alidationAttack

Ngay cả những khái niệm phức tạp hơn như "Công tắc ghi đè có thể được điều khiển từ xa"
vì vậy nó là một vector tấn công" có thể được chính thức hóa như

RemoteControlSwitch(s1) ≡ Switch(s1) hasRemoteControl(s1)

RemoteControlSwitch AttackVector

OverrideSwitch RemoteControlSwitch

OverrideSwitch AttackV ector (suy ra)

trong đó mệnh đề logic thứ tư được suy ra từ ba mệnh đề trước đó.

4.1.3 Ngôn ngữ quy tắc web ngữ nghĩa

Các ràng buộc và yêu cầu trong ontology cũng có thể được biểu diễn bằng một tập hợp con khác
của logic vị từ với hệ thống chứng minh hiệu quả, được gọi là logic horn [41]. Các quy tắc logic horn
còn được gọi là các mệnh đề horn được sử dụng trong ontology để cung cấp thêm tính năng động cho DL
dựa trên ontology. SWRL [22] là ngôn ngữ quy tắc bao gồm cú pháp trừu tượng cho
quy tắc giống như cái sừng.

Ví dụ, yêu cầu bảo mật sau: "Công tắc dừng khẩn cấp có thể
chỉ được kích hoạt bởi người dùng được ủy quyền. Người dùng được ủy quyền nếu người dùng đã đăng nhập
và nằm trong nhóm người dùng được phép admin." có thể được diễn đạt như sau

isEnabled(EmergencyStop, đúng) isEnabledBy(EmergencyStop, user1)

isLoggedIn(user1, đúng) isAdmin(user1, đúng)

Tương tự như vậy, yêu cầu an toàn "Nếu bồn chứa hóa chất rỗng, không có bơm hút
nên chạy." có thể được diễn đạt như

cảm biếnReading(s1, 0) isEmpty(s1, true) isRunning(c1, f alse) IntakeP ump(c1)

4.1.4 Lý luận trong Ontology

Kiểm tra tính nhất quán

Một ontology là không nhất quán nếu tồn tại một thể hiện của một lớp hoặc một thuộc tính.
truyền lại một tiên đề của bản thể luận. Nếu có một mâu thuẫn logic trong một bản thể luận,
bản thể học trở nên vô nghĩa, đó là vì chúng ta có thể suy ra bất kỳ loại tuyên bố nào từ
một tập hợp các tiên đề logic mâu thuẫn với nhau [42]. Về mặt hình thức, một ontology được cho là
trở nên không nhất quán nếu một tiên đề trong bản thể học không thể thỏa mãn. Ví dụ: khẳng định
tions, {EmergencyFlush: Bơm, EmergencyFlush: Biến} gây ra sự không nhất quán trong
bản thể học, vì tên được coi là duy nhất trong bản thể học. Do đó, một bản thể học
mô hình, M (EmergencyFlush : Bơm EmergencyFlush : Biến). Ngoài ra, hãy
hãy xem xét các mệnh đề, ”OuttakePump đang chạy” tức là s1 = isRunning(OuttakePump,
1) và ”OuttakePump không chạy” tức là s2 = isRunning(OuttakePump, 0), do đó
M (s1 s2).

Tính đầy đủ của thông số kỹ thuật

Điều này có thể đạt được bằng cách kiểm tra tính đầy đủ của ontology.
là hoàn chỉnh nếu mỗi mệnh đề thuộc về một số câu lệnh và mọi trường hợp của
các thành phần từ vựng có liên quan đến một số câu lệnh. Ví dụ: Sau đây
yêu cầu bảo mật ”Để kích hoạt công tắc ghi đè, người dùng phải đăng nhập.” tức là
LoggedInUser(user1) switchAccess(user1, true) enableSwitch(true, user1). Điều này

không thể hoàn thành nếu có khẳng định User(user1) và tiên đề LoggedInUser
Người dùng không có trong thuật ngữ.

4.2 Kiểm tra mô hình

Trong kiểm tra mô hình, các hệ thống được mô hình hóa bằng các máy trạng thái hữu hạn và các thuộc tính được được viết bằng logic thời gian mệnh đề. Quy trình xác minh là một tìm kiếm không gian trạng thái của mô hình thiết kế. Khung kiểm tra mô hình chúng tôi đang khám phá là [43] trong đó các mô tả hệ thống được chỉ định trong TA, trong khi yêu cầu thông số kỹ thuật trong công thức TCTL. Vấn đề kiểm tra mô hình TCTL là P-SPACE hoàn chỉnh [44]. Trong phần sau, chúng tôi đưa ra các định nghĩa chính thức của TA và TCTL.

4.2.1 Máy tự động hẹn giờ

TA [43] [45] là một phần mở rộng cho một máy tự động hữu hạn với một tập hợp hữu hạn các giá trị thực các biến được gọi là đồng hồ để chỉ định các ràng buộc về thời gian giữa hai sự kiện. Nếu X = {x1, x2, ... , xn} là một tập hợp hữu hạn các đồng hồ, khi đó định giá đồng hồ là một ánh xạ v: X R X và φ(X) biểu diễn tập hợp các công thức được gọi là ràng buộc đồng hồ. Ví dụ, Hình 4.4 cho thấy một máy tự động cơ bản có thời gian với các biến đồng hồ x và y. Một trong những khả năng chạy của máy tự động này là

(l0,(0, 0)) $\xrightarrow{4.1}$ (l0,(4.1, 4.1)) (l1,(4.1, 0)) $\xrightarrow{3.7}$ (l1,(5.3, 3.7)) (l2,(5.3, 3.7))

trong đó (l2, (5.3, 3.7)) có nghĩa là giá trị của biến đồng hồ x là 5.3 và y là 3.7 tại vị trí l2.

TA là một bộ A(L, L0, Σ, X, I, E), trong đó L là một tập hợp hữu hạn các vị trí, L0 L

Hình 4.4: Automaton thời gian

là một tập hợp các vị trí ban đầu, Σ là một tập hợp hữu hạn các nhãn, X là một tập hợp hữu hạn các đồng hồ, I là một ánh xạ gắn nhãn cho mỗi vị trí l với một số ràng buộc đồng hồ trong φ(X) và E là một tập hợp hữu hạn các cạnh có dạng e = (l, γ, α, x, l ′), với l, l ′ L nguồn và đích trạng thái, γ là sự kết hợp của các ràng buộc nguyên tử trên X, được gọi là bảo vệ, α là nhãn cho hành động rời rạc hoặc thời gian trễ và x X là một tập hợp các đồng hồ được thiết lập lại khi vượt qua cạnh.

Cái đồng hồ

Một tập hợp hữu hạn các biến đồng hồ thực được sử dụng để xác định thời gian định lượng những ràng buộc liên quan đến quá trình chuyển đổi. Đồng hồ là một biến có phạm vi trên R, tập hợp các số thực không âm. Tất cả các biến đồng hồ trong máy tự động được định thời tiến lên đồng thời.

Các tiểu bang

Trạng thái của A là một cặp (l, v) trong đó l L là trạng thái rời rạc tức là vị trí, và v là tập hợp tất cả các giải thích đồng hồ cho X thỏa mãn I(l). Trạng thái ban đầu của A là s0 = (10, 0).

Chuyển tiếp

Chúng ta hãy xem xét trạng thái s = (l, v). Sự chuyển đổi rời rạc của A, tức là (l, v) α (l ′ ,v′), xảy ra nếu cạnh e = (l, γ, α, x, l ′), sao cho v γ và v ′ = v[reset(e)], trong đó

reset(e) biểu diễn phép gán đồng hồ ánh xạ tất cả đồng hồ trong x thành 0. Thời gian
sự chuyển đổi của A tức là (l, v) δ (l, v + δ), xảy ra nếu (v + δ I(l)) và δ R. Thời gian
tiếp s và chuyển tiếp rời rạc cũng có thể được viết là s δ s + δ và s + δ α chuyển ′
tương ứng.

Chạy

Một chuỗi A từ trạng thái s là một chuỗi hữu hạn hoặc vô hạn r = s1 s1 +δ 1 δ 1 s2 s2 α1 δ 2
+δ 2 α2 ...,, trong đó si = s đối với mọi i = 1, 2, ..., si+δ i là quá trình chuyển đổi thời gian của si và si + 1
là sự chuyển tiếp rời rạc của si+δ i .

Các tiểu bang có thể đạt được

Trạng thái s có thể đạt được nếu tồn tại một chuỗi hữu hạn s0 δ 0 s0 +δ 0 ... sk δ s, trong đó s0=(l0
0) là trạng thái ban đầu và k N.

4.2.2 Logic cây tính toán theo thời gian

Các hệ thống trạng thái hữu hạn được mô hình hóa bằng đồ thị chuyển đổi trạng thái có nhãn gọi là Kripke
Cấu trúc. Cấu trúc Kripke là một bộ ba M = (S, R, L), trong đó, S là một tập hợp các trạng thái,
R S x S là một quan hệ chuyển tiếp, và L: S P(AP) là tập hợp các mệnh đề nguyên tử
(AP) đúng trong mỗi trạng thái. Cấu trúc có thể được tháo ra thành một cây vô hạn với
trạng thái ban đầu là gốc. Đường dẫn trong M là một chuỗi vô hạn các trạng thái, π = s0, s1,
... sao cho với i ≥ 0, (si , si+1) R.

Trong thiết kế công cụ kiểm tra mô hình, có những lựa chọn về ngôn ngữ thời gian
được sử dụng để chỉ định các thuộc tính. Hai loại chỉ định thuộc tính tạm thời
ngôn ngữ được phân biệt bởi mô hình thời gian cơ bản của chúng, Logic thời gian tuyến tính
(LTL) và CTL. Trong LTL, mỗi thời điểm trong thời gian có một tương lai có thể xảy ra duy nhất. tức là

các toán tử được cung cấp là để mô tả các sự kiện dọc theo một phép tính duy nhất
đường dẫn. Tuy nhiên, trong logic thời gian phân nhánh, chúng ta có thể có một khoảnh khắc thời gian tách biệt
vào nhiều tương lai có thể xảy ra. TCTL là phần mở rộng theo thời gian của logic CTL
logic phân nhánh thời gian, trong đó giới hạn của một toán tử thời gian được đưa ra dưới dạng một cặp
giới hạn dưới và giới hạn trên.

Công thức TCTL có thể được định nghĩa theo phương pháp quy nạp thông qua quy tắc sản xuất sau.

φ := đúng | p | Âp | φ | φ φ | E[φUIφ] | A[φUIφ]

trong đó, p là tập hợp các công thức nguyên tử, A và E là các công thức phổ quát và hiện sinh
lượng tử đường dẫn, tương ứng. U(Until) là toán tử thời gian và I biểu diễn bất kỳ
một trong các toán tử quan hệ (=, <, ≤). Chúng ta có thể định nghĩa nhiều thuộc tính sử dụng
Các toán tử thời gian "Luôn luôn" (hoặc G), hoặc "Cuối cùng" (hoặc F) dựa trên tập hợp
của các toán tử được trình bày bởi quy tắc sản xuất. Cho mô hình trạng thái hữu hạn M với
trạng thái ban đầu s0, công thức TCTL φ được thỏa mãn bởi mô hình có thể được chính thức
được biểu thị là (M, s0) φ. Ví dụ: Nếu p và q là công thức nguyên tử cục bộ, thì,

Bất biến: s0 A G[2,3] p, ngụ ý rằng p là đúng cho mọi đường đi có thể trong tương lai
giữa các trạng thái s0+2 và s0+3, được thể hiện trong Hình 4.5a.

Thời gian phản hồi bị giới hạn: s0 AG (p A F≤4 q), ngụ ý rằng đối với tất cả các đường dẫn, nó
luôn là trường hợp mà một khi p được giữ nguyên, q cuối cùng cũng được giữ nguyên trong vòng 4 đơn vị thời gian, được thể hiện trong
Hình 4.5b.

4.3 Chuyển đổi sang TCTL

Đã có một số công việc để dịch các thông số kỹ thuật trong ontology sang TCTL bằng cách sử dụng
hệ thống mẫu đặc tả (SPS), là một tập hợp các mẫu chức năng lặp lại

(a) s0 A G[2,3] p (b) AG (p A F≤4 q)

Hình 4.5: Thuộc tính TCTL

yêu cầu về thời gian và ngôn ngữ [46]. Trong công trình này, để đơn giản, chúng tôi chỉ xem xét chuyển đổi các yêu cầu được thể hiện trong các quy tắc SWRL. Lý do chính cho điều này phiên bản là chúng ta cần xác minh các yêu cầu bằng cách sử dụng trình kiểm tra mô hình hỗ trợ logic thời gian. Ưu điểm của chuyển đổi này là các yêu cầu/thuộc tính được tạo ra để kiểm tra xem có phù hợp với các thông số kỹ thuật chung và các thông số kỹ thuật khác không yêu cầu. Điều này quan trọng vì việc kiểm tra mô hình có thể tốn thời gian và tài nguyên tổng hợp, vì vậy, các thuộc tính chúng ta đang kiểm tra phải nhất quán và đầy đủ trong trường hợp này được đảm bảo bởi bản thể học.

Thuật toán 1 cho thấy việc tạo ra công thức thời gian trong TCTL. Đầu vào cho các thuật toán này là tập hợp các yêu cầu và ràng buộc từ ontology, tức là các mệnh đề horn của các khái niệm DL. Ví dụ: ”Một tràn tối đa của hóa chất, có thể gây vỡ đường ống. Vì vậy, chúng ta cần luôn đảm bảo rằng vật liệu chảy qua đường ống nhỏ hơn ngưỡng”, có thể được biểu diễn dưới dạng hình sừng các mệnh đề như

hasFlowrate(p1, fr) flowT ngưỡng(p1, th) fr < th

trong đó hasFlowrate và flowThreshold là các thuộc tính dữ liệu trong ontology liên quan

pipe1(p1) với giá trị nguyên cụ thể là fr và th tương ứng.

Đầu tiên, thuật toán chia mệnh đề thành tập hợp các nghĩa đen. Sau đó, mỗi lit-
eral được xem xét trong một tệp ánh xạ có chứa mối quan hệ giữa dữ liệu/đối tượng
các thuộc tính trong ontology và mẫu mô hình của trình kiểm tra mô hình (MapOntol-
ogyToModelChecker). Ví dụ: trong ví dụ trên, hasFlowRate(p1, fr)
sẽ là p1.flowrate=fr trong trình kiểm tra mô hình. Sau đó, nhiệm vụ cuối cùng là thêm tempo-
toán tử ral (A, E, ,). Trong trường hợp của chúng tôi, để đơn giản, chúng tôi đã giới hạn số lượng
của các toán tử thời gian để sử dụng, nhưng thuật toán có thể được mở rộng để hỗ trợ nhiều hơn
toán tử thời gian. Đầu ra của thuật toán này là một tập hợp các công thức TCTL. tức là E
(p q).

Thuật toán 1 Bản dịch các mệnh đề giống như sừng sang TCTL

```
1: Nhận tất cả các quy tắc
2: đối với quy tắc trong rules(r1, r2, ..., rn) thì làm
3:     literals     getLiterals(quy tắc);
4:     q     getHead(quy tắc);
5:     đối với l theo nghĩa đen làm
6:         pArr     MapOntologyToModelChecker(l);
7: kết thúc cho
8:     p     joinLiterals(pArr);
9:     tctl     addTemporalOperators(tctl);
       tctls[0, 1, ..i]     E   (p     q);
10:11: kết thúc cho
```

CHƯƠNG 5:

NGHIÊN CỨU TRƯỜNG HỢP

Để chứng minh khái niệm, chúng tôi thực hiện quy trình OT công nghiệp như chúng tôi đã thảo luận trong chương 2. Chúng tôi đã sử dụng Velocio PLC [47] để thực hiện logic điều khiển của hóa chất quá trình. Velocio sử dụng logic bậc thang và ngôn ngữ biểu đồ luồng để lập trình. Chúng tôi sử dụng Người được bảo vệ sẽ tạo ra một bản thể học với các lớp, cá nhân, thuộc tính và re- thiết yếu. tiên đề quan hệ. Prot´eg´e sử dụng OWL để tạo ra một ontology. Đối với các yêu cầu và các ràng buộc, các quy tắc SWRL được thêm vào thuật ngữ. Một lý luận Pellet[20] hoạt động với ontology được sử dụng để suy ra kiến thức dựa trên các dữ liệu hiện có hình thành trên ontology và kiểm tra tính nhất quán của ontology. Logic của chương trình PLC được mô hình hóa trong TA của trình kiểm tra mô hình UPPAAL [23]. Sau đó, các yêu cầu từ ontology được dịch thành các truy vấn UPPAAL để biết thêm xác minh mô hình.

Phần còn lại của phần này cung cấp một giới thiệu ngắn gọn về cách chúng tôi xây dựng thuật ngữ để phát hiện xung đột trong các yêu cầu, mô hình UPPAAL của chương trình PLC logic điều khiển và việc xác minh các thuộc tính được thực hiện bằng UPPAAL Verifier.

5.1 Ontology với Prot´eg´e và OWL-API

Do sự phổ biến của OWL trong web ngữ nghĩa, rất nhiều công cụ có sẵn cho biên tập và xây dựng một ontology. Prot´eg´e là một trong những công cụ mã nguồn mở và

	Thông tin và thông số kỹ thuật	DL tương đương (Ontology)
	OuttakePump là một máy bơm	OuttakePump Bơm
1 2	Bơm là một thành phần	Bơm Thành phần
3	Máy trộn sẽ chạy trong 11 giây	Máy trộn shouldRunFor.11
4	IntakePump1 phải chạy trước Mixer	IntakePump1 shouldRunBefore.Mixer
5	IntakePump2 đang chạy	IntakePump2 isRunning.True

Bảng 5.1: Định nghĩa lớp và thuộc tính

chứa một công cụ lý luận và suy luận hỗ trợ việc xác thực và xác minh của các truy vấn DL. Để xây dựng một ontology, chúng tôi đã sử dụng Prot´eg´e như một công cụ GUI và OWL-API [48] để tự động hóa việc truy cập ontology, tức là cập nhật, xóa. OWL-API là một Java API triển khai để tạo, thao tác và tuần tự hóa các thuật ngữ OWL. Để hướng dẫn quá trình xây dựng bản thể của chúng tôi, chúng tôi đã làm theo hướng dẫn [49] do Prot´eg´e cung cấp.

Trong Prot´eg´e, chúng ta có thể tạo các lớp phân cấp, xác định các thuộc tính đối tượng và mối quan hệ của chúng với các lớp riêng lẻ và thêm các thuộc tính dữ liệu gán một nghĩa đen giá trị cho các cá nhân lớp. Lúc đầu, chúng tôi xây dựng một thuật ngữ của các thành phần và các tính chất liên quan đến quá trình hóa học. Bảng 5.1 mô tả các ví dụ về thông tin được sử dụng để xây dựng ontology. Ngoài ra, các yêu cầu và ràng buộc là được thêm vào thông qua các truy vấn SWRL trong Prot´eg´e.

5.1.1 Ontology về An toàn và Bảo mật

Để tạo ra thuật ngữ an toàn và bảo mật, chúng tôi đã duyệt qua các bài báo nghiên cứu và tạp chí và lưu ý một số khái niệm bảo mật/an toàn quan trọng liên quan đến PLC và ICS trong chung. Phần còn lại của phần này cung cấp hình thức của một số ví dụ về các yêu cầu và khái niệm về an toàn/bảo mật mà chúng tôi đã sử dụng.

Yêu cầu bảo mật

Nếu có các đối tượng hoặc biến được khởi tạo trong chương trình PLC, kẻ tấn công có thể có thể truy cập vào hệ thống với nỗ lực tối thiểu bằng cách chèn [50]. Do đó, se-yêu cầu bảo mật cho trường hợp này sẽ là "Không để lại biến chưa sử dụng trong PLC chương trình.". Trong chương trình PLC, mỗi biến được gán cho đầu vào/đầu ra các thành phần hoặc được sử dụng trong chương trình để hoạt động tạm thời, nếu biến không không ảnh hưởng đến logic điều khiển, chúng là các biến không được sử dụng và dễ bị tấn công thông qua việc chèn ngay lập tức. Những loại suy luận này được thực hiện tự động bởi người lý luận DL dựa trên thông tin đã được thêm vào.

Các giá trị số được mã hóa cứng trong các chương trình PLC có thể dễ bị tấn công bởi al-giảm số lượng được thay đổi trực tiếp [50]. Yêu cầu bảo mật cho điều này trường hợp có thể là "Tránh sử dụng các giá trị số được mã hóa cứng trong các chương trình PLC". Những các yêu cầu được thêm vào thuật ngữ như sau:

hasV alue(Biến, Giá trị)

Giá trị được mã hóa cứng Giá trị

isV ulnerableT o(HardcodedV alue, InputV alidationAttack)

InputV alidationAttack SecurityAttack

Yêu cầu an toàn

Một trong những yêu cầu an toàn đối với quy trình hóa chất là “Cả ba bể chứa đầu vào không nên chạy cùng lúc.”. Lý do chính là P4 trong Hình 2.1 có ngưỡng lưu lượng là 6 lít mỗi giây, vì vậy chạy tất cả cùng nhau sẽ vượt quá ngưỡng, có thể gây ra tai nạn nghiêm trọng do vỡ đường ống. Một điều quan trọng khác

ràng buộc an toàn là nếu máy trộn chạy rỗng, nó sẽ làm tăng nhiệt độ của
máy trộn đột ngột, điều đó cuối cùng có thể dẫn đến một vụ nổ. Điều này có thể được viết
vì “Máy trộn không nên hoạt động nếu không có hóa chất”.

5.1.2 Phát hiện sự không nhất quán

Một trong những lý do chính để chính thức hóa các thông số kỹ thuật và yêu cầu trong DL dựa trên
ontology là nó cung cấp phân loại tự động và kiểm tra sự không nhất quán. Đối với
Ví dụ, trong quá trình hóa học, chúng ta có tập hợp các yêu cầu sau: (1)
”Người dùng phải đăng nhập để bật/tắt công tắc.” (bảo mật) (2) ”Yêu cầu đăng nhập tại
ít nhất một phút để hoàn thành.” (đặc điểm kỹ thuật) (3) “Trong trường hợp khẩn cấp, EmergencyStop
công tắc phải được bật trong vòng 30 giây.” (an toàn). Ở đây, chúng ta có thể thấy rằng
yêu cầu rõ ràng là không nhất quán vì người dùng đã đăng nhập cần hơn 30
giây để vô hiệu hóa công tắc dừng khẩn cấp, cần phải vô hiệu hóa trong vòng 30
giây. Các yêu cầu trên được thêm vào thuật ngữ như sau:

LoggedInUser isLoggedIn giá trị 'true′

canEnableSwitch(Tên miền: LoggedInUser, Phạm vi: LRSwitch)

timeRequiredT oĐăng nhập(?u, ?t) t >= 60 Người dùng đã đăng nhập(?u)

LRSwitch(?s) enableSwitchW trong(?s, ?t1) Người dùng đã đăng nhập(?u)

timeRequiredT oĐăng nhập(?u, ?t2) t1 <= t2 V iolation(R1)

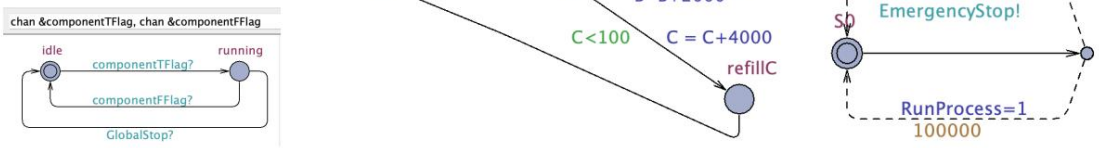
Sau khi thêm các quy tắc và thông số kỹ thuật này, bộ lý luận được đồng bộ hóa trong Protege
để xem có sự không nhất quán hoặc vi phạm không. Chúng ta có thể thấy (Hình 5.1) rằng có một
vi phạm gây ra bởi các quy tắc chúng tôi đã thêm vào. Người ta cũng có thể nhận thấy rằng lý do

cũng cung cấp lý do vi phạm.

Hình 5.1: Kiểm tra sự không nhất quán của yêu cầu (Protege)

5.2 Kiểm tra mô hình với UPPAAL

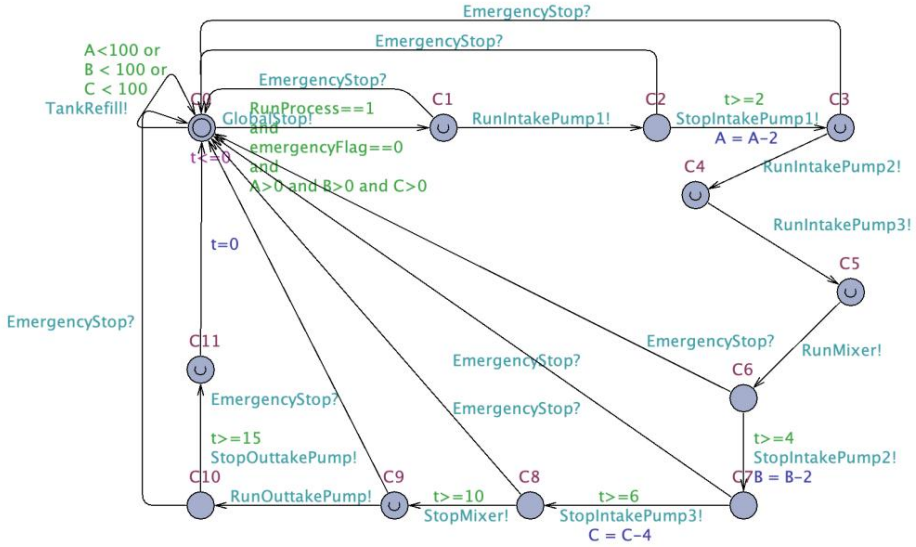
UPPAAL là một hộp công cụ để xác minh các hệ thống thời gian thực. Trong UPPAAL, hệ thống tem được mô hình hóa trong mạng lưới TA, trong khi các yêu cầu được chính thức hóa thành Ngôn ngữ truy vấn của UPPAAL là một tập hợp con của TCTL. UPPAAL mở rộng au- được định thời gian tomata với nhiều tính năng khác như mẫu, đồng bộ hóa, vị trí khẩn cấp và các biểu thức như bảo vệ, bất biến [51].



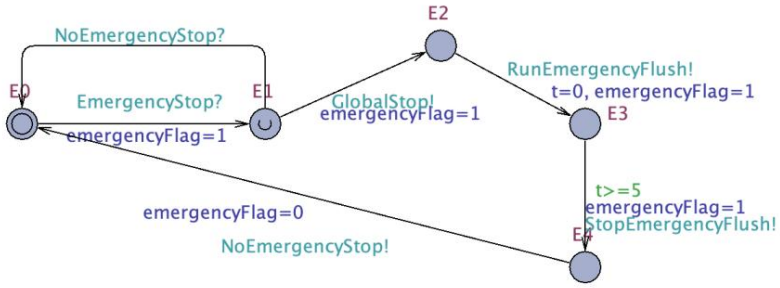
(a) Mô hình thành phần (b) Mô hình nạp lại bình (c) Đầu vào của người dùng

Hình 5.2: Mẫu mô hình UPPAAL

Trong nghiên cứu trường hợp của chúng tôi, tức là quy trình hóa học, bơm nạp, bơm xả, máy trộn, và xả hoạt động theo cùng một cách như một mô hình máy tự động được định thời gian. Vì vậy, chúng tôi đã tạo ra một mẫu được gọi là thành phần như thể hiện trong Hình 5.2a. Mẫu thành phần



Hình 5.3: Quá trình hóa học tổng thể



Hình 5.4: Trường hợp khẩn cấp

có hai trạng thái {idle, running}. Các trạng thái này có thể được thay đổi bằng cách sử dụng các biến

là tham số cho mẫu. Mẫu này có thể được khởi tạo để tạo ra từng

thành phần bằng cách tạo một đối tượng trong khai báo hệ thống của UPPAAL như sau:

    lượngPump1 = Thành phần(RunIntakePump1, StopIntakePump1),

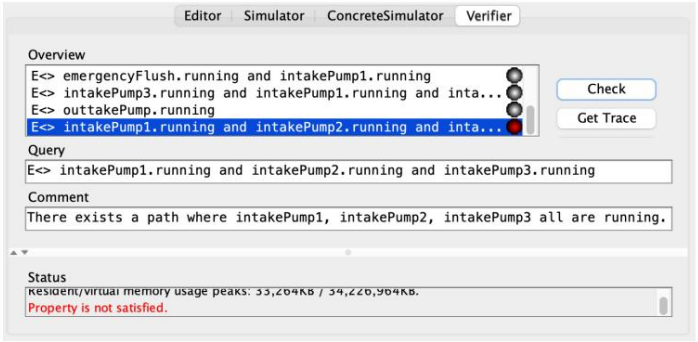
lượngPump2 = Thành phần(RunIntakePump2, StopIntakePump2),

intakePump3 = Thành phần(ChạyIntakePump3, DừngIntakePump3),

mixer = Thành phần(RunMixer, StopMixer),

outtakePump = Thành phần(RunOuttakePump, StopOuttakePump),

emergencyFlush = Thành phần (RunEmergencyFlush, StopEmergencyFlush).



Hình 5.5: Xác minh yêu cầu trên UPPAAL (R2)

Một trong những đặc điểm kỹ thuật của quá trình hóa học là nếu tất cả các hóa chất thùng chứa hóa chất có dung tích dưới 100 gallon, thì cần phải nạp lại sớm.

được mô hình hóa trong UPPAAL như thể hiện trong Hình 5.2b.

Như chúng tôi đã đề cập ở phần trước, yêu cầu "Cả ba máy bơm đều phải

không chạy cùng lúc.” được chính thức hóa trong các quy tắc SWRL như

isRunning(intakeP ump1, đúng) isRunning(intakeP ump2, đúng)

isRunning(intakeP ump3, true) V iolation(R2)

Nếu có trường hợp ba máy bơm này chạy đồng thời thì

vi phạm quy tắc. Quy tắc này được dịch sang các truy vấn TCTL với thuật toán (1)

được triển khai trong Java. Bản dịch cung cấp cho chúng ta công thức TCTL cho UPPAAL

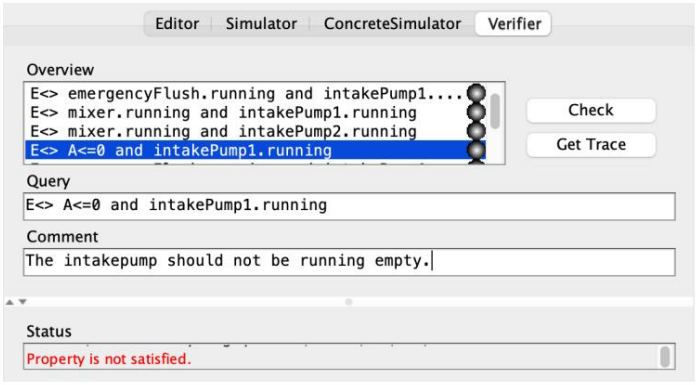
kiểm tra mô hình. Công thức được dịch là

E <> intakeP ump1. chạy và intakeP ump2. chạy và intakeP ump3. chạy

Công thức này hỏi trình kiểm tra mô hình xem có tồn tại đường dẫn nào mà cả ba máy bơm đều đang chạy. Chúng tôi đã chạy truy vấn này trên mô hình của chúng tôi trong UPPAAL Verifier và thuộc tính này không được thỏa mãn (Hình 5.5). Tương tự, theo quy tắc SWRL,

containsAmount(Hóa chất A, ?am)– am < 0 isRunning(intakepump1, true)

V iôlat hóa(R3)



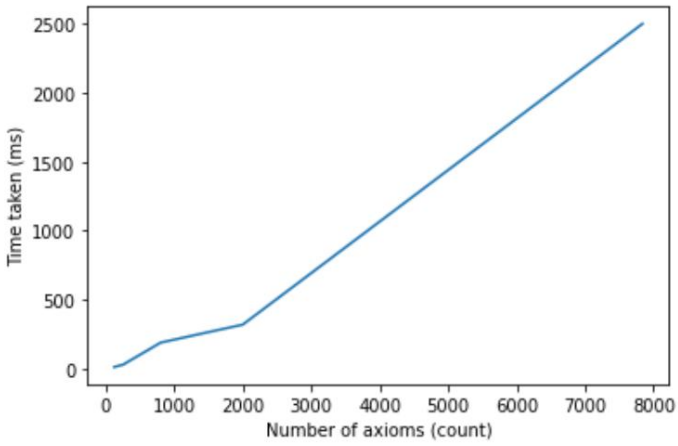
Hình 5.6: Xác minh yêu cầu trên UPPAAL (R3)

Quy tắc này nêu rõ rằng máy bơm nạp1 không được hoạt động khi không có nhiên liệu. Nghĩa là, phải có một số hóa chất có trong bể chứa đầu vào tương ứng. Điều này yêu cầu được dịch sang truy vấn TCTL như

E <> A <= 0 và intakeP ump1.running

Chúng tôi đã kiểm tra thuộc tính TCTL này cho mô hình trong UPPAAL (Hình 5.6).
kết quả cho thấy thuộc tính không được thỏa mãn, nghĩa là không có đường dẫn nào trong
thực hiện mô hình, trong đó intakepump1 sẽ chạy rỗng. Theo cách này,
chúng tôi có thể đảm bảo rằng các chính sách an toàn và bảo mật không bị vi phạm bởi PLC
chương trình kiểm soát. Các chính sách đang được kiểm tra được đảm bảo là nhất quán
và được người lý luận hoàn thiện.

Đối với thí nghiệm này, một ontology với tổng cộng 313 tiên đề logic đã được xây dựng.
tổng thời gian để kiểm tra tính nhất quán trung bình là 80 ms. Sự biến đổi của
năm quy tắc SWRL mất trung bình 780 ms, và sau đó kiểm tra mô hình mất gần 3
giây cho mỗi yêu cầu cụ thể của chúng tôi. Thí nghiệm này được thực hiện trên 2.4
GHz Quad-Core Intel Core i5 MacBook Pro với 16 GB RAM. Tổng thời gian thực hiện
vì quá trình này tăng lên khi chúng ta tăng kích thước của thuật ngữ và các biến chứng
trong mô hình TA. Nếu chúng ta tăng kích thước của ontology với số lượng logic lớn hơn
tiên đề, người lý luận sẽ mất nhiều thời gian hơn để phân loại và phát hiện tự động
sự không nhất quán. Thời gian thực hiện tăng tuyến tính với số lượng tiên đề như được hiển thị trong
Hình 5.7. Chúng ta có thể tìm thấy một phân tích thực nghiệm chi tiết của Dalwadi et al. trên [52].
Tương tự như vậy, kiểm tra mô hình cũng có thể chạy vào các trạng thái không xác định do không gian trạng thái của nó
vấn đề nổ khi mô hình trở nên phức tạp.



Hình 5.7: Số lượng tiên đề so với thời gian thực hiện

CHƯƠNG 6:

KẾT LUẬN VÀ CÔNG VIỆC TƯƠNG LAI

Phần đầu tiên của luận án này trình bày một cách tiếp cận để xác minh logic điều khiển PLC sử dụng một ontology. Cơ sở kiến thức ontology được xây dựng bằng OWL với sự trợ giúp của OWL-API. Một lý luận dựa trên DL được sử dụng để kiểm tra tính nhất quán giữa yêu cầu được thêm vào ontology. Ontology được tạo ra được sử dụng để xác minh xem logic điều khiển vi phạm các yêu cầu được thêm vào trong thuật ngữ.

Tuy nhiên, nghiên cứu này có một số hạn chế. Đầu tiên là chúng ta cần để trích xuất logic điều khiển từ chương trình PLC và sau đó logic đó được dịch sang ngôn ngữ khác. Trích xuất logic thủ công và dịch có thể giúp xác định một số vấn đề trước đây không thấy rõ trong chương trình PLC gốc. Tuy nhiên, nó là một quá trình mất mát, vì vậy, công việc trong tương lai có thể là tự động hóa điều này. Hạn chế thứ hai là rằng thuật toán được trình bày trong bài báo này để dịch các yêu cầu trong DL hoặc SWRL quy tắc cho các truy vấn TCTL chỉ hoạt động cho các trường hợp cụ thể. Và cuối cùng, hạn chế thứ ba là rằng trong khi tạo ra bản thể học và chính thức hóa các yêu cầu, kiến thức chuyên môn về lĩnh vực là bắt buộc.

Trong công việc tương lai, chúng tôi sẽ mở rộng yếu tố khái niệm trong DL để thể hiện các tình huống nguy hiểm và lỗ hổng bảo mật có thể dẫn đến mất mát, từ chối và thao túng. Các trường hợp là các yếu tố thuộc về một khái niệm và các vai trò là các mối quan hệ nhị phân giữa hai khái niệm. Dựa trên hình thức DL, chúng ta sẽ xây dựng một nền tảng an toàn và bảo mật

tology, nghĩa là, một biểu diễn kiến thức về một tập hợp các khái niệm trong một miền và mối quan hệ giữa chúng. Để xây dựng bản thể luận, chúng tôi sẽ sử dụng STPA-safesec [53], là một kỹ thuật phân tích an toàn và bảo mật là phần mở rộng của Mô hình và quy trình tai nạn lý thuyết hệ thống của Leveson (STAMP) và hệ thống Phân tích quy trình lý thuyết (STPA) [54, 55] để lập bản đồ các khái niệm an toàn và bảo mật và vai trò. Chúng tôi sẽ thực hiện phân tích STPA-Safesec bằng cách: (1) xác định kiểm soát khái niệm lớp, (2) xác định các hành động kiểm soát nguy hiểm (khái niệm an toàn), (3) lập bản đồ lớp điều khiển và thành phần (học thuyết), (4) sửa đổi tính an toàn và ràng buộc an ninh, (5) tạo ra các kịch bản nguy hiểm. Ngoài ra để có nhiều en-ontology phong phú, chúng tôi đang làm việc trên việc sử dụng thông tin cơ sở kiến thức ontology giống như giá trị biên, mối quan hệ trong xác minh mô hình tượng trưng của UPPAAL.

PHẦN II

CHƯƠNG 7:

THAM KHẢO KIẾN THỨC VỀ LOGIC XÁC MINH

Trong phần trước, chúng ta đã thảo luận về cách thức ontology dựa trên DL, cùng với lý do soner có thể được sử dụng để phát hiện xung đột trong các yêu cầu và thông số kỹ thuật của ICS. Sau khi kiểm tra tính nhất quán, các yêu cầu đó được sử dụng trong kiểm tra mô hình bằng dịch các yêu cầu từ các quy tắc SWRL thành các truy vấn TCTL. Phần này thảo luận về cách ontology được xây dựng có thể được sử dụng như một cơ sở kiến thức tham khảo trong phần mềm xác minh.

7.1 Giới thiệu

ICS hiện nay dễ bị tấn công và đe dọa từ bên ngoài hơn [25] do sự thâm nhập nhanh chóng của nó mối quan hệ với Internet. Một số lỗ hổng phổ biến trong các ngành công nghiệp là Ghi ngoài giới hạn, Đọc ngoài giới hạn và Xác thực đầu vào không hợp lệ [10]. Những lỗ hổng này có thể dẫn đến mất tính khả dụng, tính toàn vẹn và tính bảo mật trong các hệ thống quan trọng mà độ chính xác và khả năng kiểm soát là rất quan trọng đối với hoạt động an toàn của các quy trình. Những lỗ hổng này có thể được khai thác như một phần của logic điều khiển tấn công sửa đổi trong đó kẻ tấn công cố gắng thao túng các phép đo cảm biến và các giá trị biến chương trình với một số giá trị không mong muốn/ngoài giới hạn gây ra bộ điều khiển để gửi ra các lệnh điều khiển không an toàn. Thao tác từ xa có thể

bao gồm những thay đổi trong các biến vượt ra ngoài giới hạn khiến hệ thống ở trong một trạng thái không an toàn. Những ví dụ này cho thấy rằng nếu các kỹ sư, kỹ thuật viên và quản lý rủi ro các nhóm kỹ sư hiểu cách logic điều khiển và các hệ thống con, hệ thống và các thành phần liên quan đến nhau và cách dữ liệu được duy trì và trao đổi giữa nhiều lĩnh vực khác nhau trong các quy trình công nghiệp, họ sẽ có thể thực hiện kiểm soát an ninh và an toàn thích hợp và chuẩn bị đánh giá/giảm thiểu rủi ro chính xác kế hoạch.

Để cung cấp sự hiểu biết và mô hình hóa mối quan hệ tồn tại giữa dữ liệu trên các miền phần mềm khác nhau, trong thử nghiệm phần mềm, kiểm tra ranh giới ing [56, 57] là phương pháp phổ biến để phát hiện xem một biến có nằm trong phạm vi an toàn hay không và các giới hạn an toàn cho hoạt động chính xác của hệ thống. Kiểm tra phạm vi [58, 59] là một trong những phương pháp kiểm tra giới hạn để xác minh xem biến có nằm trong phạm vi an toàn hay không phạm vi. Các kỹ thuật này thường được sử dụng trong thời gian biên dịch để tạo ra kết quả an toàn mã. Tuy nhiên, không có nhiều công cụ và thuật toán có sẵn cho việc thử nghiệm như vậy trong các hệ thống quan trọng về an toàn dễ bị tấn công toàn vẹn, tức là cảm biến các cuộc tấn công nipulation, các cuộc tấn công Giao diện người máy (HMI) sẽ dẫn đến tình trạng thiếu nhận thức tình huống. Các quy trình kiểm soát theo bản chất được yêu cầu phải có khả năng phục hồi cao với khả năng bảo vệ an toàn và an ninh mạnh mẽ.

Trong phần này của luận án, chúng tôi sử dụng thuật ngữ dựa trên DL [60] để mô tả kiến thức liên quan đến ranh giới thành phần, sự phụ thuộc của quy trình, khái niệm về an toàn và an toàn. Chúng tôi đề xuất một phương pháp xác minh chính thức để kiểm tra ranh giới trong quá trình xác minh logic điều khiển. Đầu tiên, chúng tôi mô phỏng mô hình chính thức của chương trình Bộ điều khiển logic lập trình (PLC) sử dụng UPPAAL [61] và UPPAAL-API. Sau đó, một thuật toán xác minh ranh giới được thiết kế và phát triển để

nhập thông tin cần thiết từ thuật ngữ.

Phần còn lại của chương này được tổ chức như sau. Mục 7.2 cung cấp một tổng quan về một số bối cảnh và các tác phẩm liên quan. Phần 7.3 tóm tắt việc thực hiện của chúng tôi việc xác minh ranh giới và định nghĩa chính thức các điều kiện ranh giới. Phần 7.4 giải thích chi tiết thuật toán và các khái niệm sơ bộ. Phần 7.5 thảo luận về kết quả thực hiện. Cuối cùng, Phần 7.6 kết thúc chương với công việc tương lai.

7.2 Các tác phẩm liên quan

Các kỹ thuật phân tích mã tĩnh và động được coi là một trong những kỹ thuật quan trọng phương pháp phát hiện lỗ hổng bảo mật trong thiết kế và phát triển phần mềm chu kỳ [62, 63]. Phân tích tĩnh kiểm tra mã nguồn hoặc nhị phân mà không chạy mã, trong khi phân tích động liên quan đến việc thực hiện thực tế của phần mềm. Nhiều các công cụ và kỹ thuật có sẵn cho cả hai phân tích, tuy nhiên mỗi loại đều có ưu điểm và nhược điểm. Trong khi kiểm tra mã tĩnh không tính đến việc xem xét mã động bản chất của dữ liệu ICS, nó đòi hỏi ít tài nguyên hơn và có thể được sử dụng ở giai đoạn đầu của thiết kế và phát triển mã. Alvares et al. [64] đề xuất tính toán các kỹ thuật thông minh như thuật toán di truyền (GA) để kiểm tra tĩnh nguồn mã. Trong công trình của mình, các tác giả đã thực hiện kiểm tra ranh giới của chương trình các biến dựa trên loại và kích thước của biến.

Cụ thể hơn, các loại phân tích này đã được sử dụng để phát hiện xâm nhập trong Hệ thống mạng vật lý (CPS) [65]. Trong công trình này, các tác giả đã sử dụng thời gian tĩnh phân tích để kiểm tra ranh giới nhằm phát hiện các hướng dẫn trái phép trong CPS thời gian thực môi trường. Sau thành công của các phương pháp chính thức bao gồm (phân tích tĩnh, chứng minh định lý và kiểm tra mô hình) trong việc xác minh và xác nhận CPS, Zheng

et. al [66] đã tạo ra một báo cáo về tình trạng hiện tại của các hoạt động xác minh nghệ thuật trong CPS. Các tác giả kết luận rằng ”các kỹ thuật phương pháp chính thức hiện có và mô phỏng vẫn chưa đủ để hỗ trợ cho sự phát triển của toàn bộ mục đích chung “CPS”.

Một trong những phần khó khăn nhất của việc xác minh và xác nhận kết quả CPS từ các chiều của khả năng tương tác dữ liệu hoặc hợp nhất dữ liệu. Chúng tôi dựa vào hợp nhất dữ liệu để hiểu, giám sát và phân tích nhiều cảm biến, hệ thống điều khiển tinh vi và các quy trình công nghiệp trong robot, nhà máy điện hạt nhân, sản xuất năng lượng, phân phối và vận chuyển. Năm 2013, một tiêu chuẩn quốc tế Khuyến nghị ITU-T X.1255 có tiêu đề “Khung để khám phá thông tin quản lý danh tính” là được chấp thuận áp dụng mô hình dữ liệu để cung cấp khái niệm thống nhất nhằm thể hiện siêu dữ liệu hồ sơ cho mục đích tương tác giữa các hệ thống không đồng nhất [67]. Sau đây khuyến nghị này, PLC-PROV [14] đề xuất sử dụng nguồn gốc của hệ thống tem để phát hiện các vi phạm về an ninh và an toàn. Nguồn gốc dữ liệu là một loại siêu dữ liệu mô tả sự phụ thuộc giữa các tập dữ liệu và quy trình. Để tạo một dữ liệu biểu đồ nguồn gốc, các biến đầu vào và đầu ra của các cảm biến và bộ truyền động với các dấu thời gian được thu thập vào các dấu vết thực thi hệ thống. Các dấu vết thu được được đưa vào Curator, một công cụ quản lý nguồn gốc dữ liệu tạo ra một biểu đồ để Hiển thị luồng dữ liệu từ cảm biến đến bộ truyền động thông qua bộ điều khiển PLC. Biểu đồ được tạo trong Curator được sử dụng để phát hiện các vi phạm về an toàn và bảo mật. Biểu đồ nguồn gốc bị giới hạn ở các thành phần an toàn tạo nên sự phụ thuộc nhân quả đồ thị mật độ giữa đầu vào cảm biến và đầu ra bộ truyền động thông qua chương trình PLC dựa trên các dấu vết thực hiện của chương trình. Theo hiểu biết của chúng tôi, đã có đã có công việc hạn chế về phương pháp tiếp cận cơ sở kiến thức để kiểm tra ranh giới tự động

và xác minh chính thức PLC có chứa bản đồ về an toàn và bảo mật

đến các thành phần và quy trình.

7.3 Phương pháp luận

Hình 7.1 mô tả tổng quan về phương pháp được đề xuất, bao gồm hai giai đoạn:

1-Một ontology dựa trên DL được phát triển để bao gồm các chính sách an toàn/bảo mật, ràng buộc an toàn-

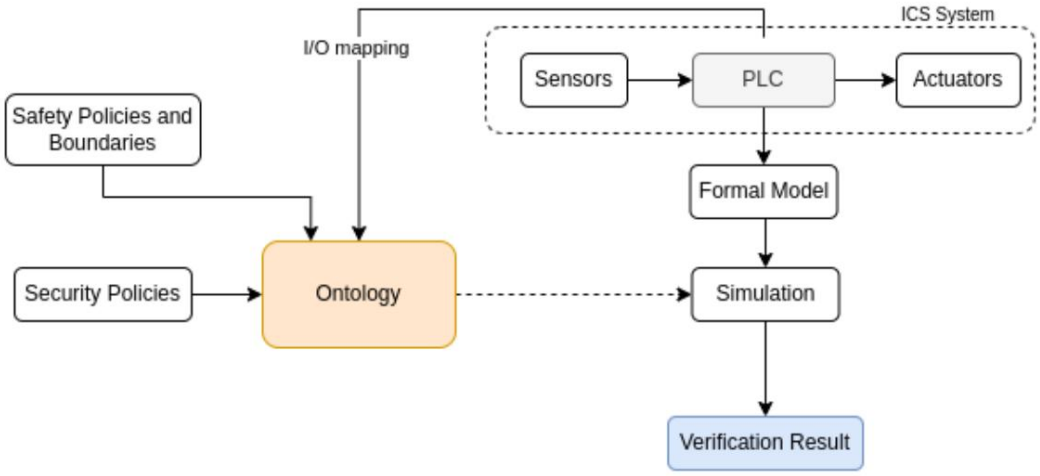
và lập bản đồ đầu vào và đầu ra của hệ thống ICS bao gồm chương trình PLC.

Với sự trợ giúp của lý luận dựa trên DL, tính nhất quán và đầy đủ của yêu cầu-

các thông số kỹ thuật và đặc điểm trong ontology được xác minh. 2- PLC được mô hình hóa theo hình thức

ngôn ngữ mô hình hóa trong đó các biến và thành phần được xác minh để đảm bảo an toàn

và các ranh giới bảo mật do thuật ngữ cung cấp được thỏa mãn.



Hình 7.1: Phương pháp tổng thể

7.3.1 Xác minh và xác thực dựa trên Ontology

Một cách tiếp cận cơ sở tri thức ontology được sử dụng trong lĩnh vực kỹ thuật phần mềm như một

công cụ quản lý kiến thức để cải thiện việc chia sẻ kiến thức và thực hành học tập

trong thử nghiệm phần mềm [68]. Thuật ngữ được trình bày trong công trình của họ bao gồm kiến thức

của miền kiểm thử phần mềm chứa các khái niệm, thuộc tính và mối quan hệ của chúng
tàu. Cổng thông tin cơ sở kiến thức được giới thiệu trong bài báo này bao gồm một kinh nghiệm-
cổng thông tin chia sẻ, nơi kinh nghiệm của người thử nghiệm được chia sẻ và thêm vào thuật ngữ.
lớp lý luận lý giải về các khái niệm và tiên đề để cung cấp sự xác thực. Sự kiện
Đồng thời, nó được lưu trữ vào lớp lưu trữ để giữ lại cơ sở kiến thức. Kiến thức
lớp truy xuất sử dụng các truy vấn SPARQL để nhập thông tin có liên quan từ
cơ sở kiến thức. Tương tự như vậy, ROoST: Ontology tham chiếu cho thử nghiệm phần mềm
[69] cung cấp các hướng dẫn để tạo ra một thuật ngữ tham chiếu cho Kiểm thử phần mềm và cho
đánh giá chi tiết về kỹ thuật thử nghiệm được đề xuất. Mặc dù các hướng dẫn này
rất tuyệt vời để xây dựng một thuật ngữ về các khái niệm và thuộc tính thử nghiệm phần mềm như một
tham khảo cho người kiểm thử phần mềm, chúng không được triển khai và sử dụng trong thực tế
công cụ kiểm thử phần mềm.

Thuật toán 2 Kiểm tra ranh giới trên mô phỏng mô hình

```
1: khởi tạo ranh giới <danh sách>
2: Lấy tất cả ValueRange từ ontology
3: đối với bV ar trong V alueRange(vr1, vr2, ..., vrn) do
4:     ranh giới     [bVar.from, bVar.to]
5: kết thúc cho
6: cho Mô hình mô phỏng làm
7:     cho tất cả các biến làm
8:         nếu biến không nằm trong ranh giới thì
9:             vi phạm     chi tiết trạng thái (số lần lặp, chuyển đổi, giá trị)
           kết thúc nếu
10:11: kết thúc cho
12: kết thúc cho
13: trả lại vi phạm
```

Trong công trình này, chúng tôi thực hiện xác thực logic điều khiển bằng cách sử dụng bộ suy luận dựa trên ontology.
Quá trình này đảm bảo tính chính xác của các điều kiện biên của các thành phần.
Người lý luận truy vấn các câu lệnh logic trong thuật ngữ và tạo ra các cờ nếu

suy diễn và suy luận dẫn đến các tuyên bố tạo thành một sự trùng lặp. Ngoài ra, một quá trình xác minh được thực hiện để đảm bảo rằng hành vi logic điều khiển đáp ứng các yêu cầu về an toàn và bảo mật. Trong trường hợp này, hành vi của chương trình PLC phải tuân thủ các điều kiện biên và sự phụ thuộc trong thuật ngữ.

Thuật toán 2 mô tả thuật toán được đề xuất để kiểm tra ranh giới của điều khiển các biến logic và các thành phần. Đầu tiên, các điều kiện cần được kiểm tra được nhập từ ontology. Đặc biệt, phạm vi giá trị của biến PLC và các thành phần hệ thống được nhập khẩu. Sau đó, mô hình chính thức của chương trình PLC và hệ thống được mô phỏng để kiểm tra các điều kiện được nhập từ bản thể học.

7.3.2 Xác minh ranh giới

Để nắm bắt các yêu cầu về an toàn và bảo mật, chúng ta cần bắt đầu ở cấp độ thành phần chẳng hạn như bộ điều khiển công nghiệp, cảm biến và bộ truyền động. Người dùng xác định số lượng của đầu vào và đầu ra đến/từ thuật toán điều khiển. Các đầu vào được kết nối với cảm biến, công tắc hoặc đầu ra của bộ điều khiển khác. Các đầu ra được kết nối với bộ truyền động đầu vào của các bộ điều khiển hoặc các đầu vào của các bộ điều khiển khác. Mỗi đầu vào và đầu ra có thể tương tác với toàn bộ hoặc một phần của một thành phần. Do đó, mỗi thành phần được biểu diễn bằng nhiều các biến được kết nối với đầu vào và đầu ra của bộ điều khiển. Mỗi thành phần được xác định được định nghĩa là một tập hợp các biến mô hình hóa chức năng của thành phần đó, tức là máy bơm. Mỗi biến, tức là tốc độ, công suất, v.v., có một kiểu như float, boolean, v.v., và một loại giá trị như liên tục, có thể lựa chọn, v.v. Mỗi thành phần sẽ được chính thức hóa như một khái niệm DL. Các biến có thể có các giá trị là hằng số, có thể lựa chọn, hoặc liên tục. Nếu giá trị là Hằng số(x) thì phải đưa ra giá trị cố định của x. Nếu giá trị là Selectable(x) thì x là một tập hợp các giá trị chỉ định khả năng lựa chọn

tùy chọn. Nếu giá trị là Liên tục(x) thì x sẽ chọn phạm vi chấp nhận được.

Các yêu cầu về an toàn và bảo mật trong ICS được định nghĩa là những ràng buộc đối với đầu vào và các giá trị đầu ra của logic bộ điều khiển. Những ràng buộc này có thể là giới hạn trên các biến hoặc sự phụ thuộc giữa các biến. Những ràng buộc này cũng có thể được xác định tại các cấp độ thành phần và loại. Đối với các ràng buộc ở cấp độ thành phần, tất cả các trường hợp của thành phần đó thừa hưởng ràng buộc đó. Đối với các ràng buộc ở cấp độ loại, tất cả các biến thuộc loại đó sẽ có ràng buộc đó.

ranh giới

Ranh giới b_v^L được định nghĩa là các giá trị an toàn có thể có mà một biến v có thể có ở mức L = {Thành phần, Kiểu, Thể hiện}. Kiểu của b được xác định dựa trên biến ranh giới được xác định dựa trên. Kiểu giá trị của biến v xác định xem ranh giới b là một tập hợp hoặc một phạm vi. Đối với các loại giá trị có thể lựa chọn và hằng số, b được định nghĩa là một tập hợp các giá trị và đối với các kiểu giá trị liên tục, b được định nghĩa là một phạm vi [b1, b2].

$$b_v^{\text{Thành phần}} := \text{trường hợp(I)} \quad \text{Thành phần(C), Iv} \models b$$

$$b_v^{\text{Kiểu}} := \text{biến(v), trong đó biến.type = Kiểu, v} \models b$$

$$b_v^{\text{Trường hợp}} := \text{trường hợp(I), Iv} \models b$$

Ví dụ, nếu bơm nạp 1 là một trường hợp của bơm nạp thành phần. Điều này có thể được biểu diễn trong biến chương trình là RunIntakePump1. Nếu biến có thể là một kiểu boolean, thì ranh giới (b) cho trường hợp này sẽ là {0, 1}, tức là ChạyIntakePump1 $\models$ b.

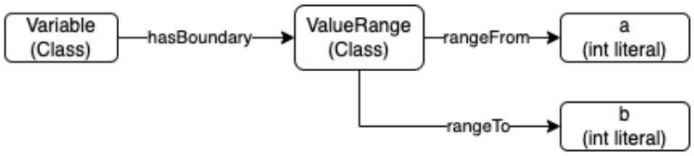
Để biểu diễn điều kiện biên này trong một ontology, trước tiên chúng ta cần định nghĩa một lớp được gọi là ValueRange. Phạm vi giá trị sẽ có giá trị bắt đầu và kết thúc, vì vậy chúng ta có hai thuộc tính dữ liệu rangeFrom và rangeTo ánh xạ thể hiện ValueRange tới giá trị theo nghĩa đen. Điều này được chính thức hóa trong DL như sau:

V alueRange(BooleanRange), Biến(IntakePump1)

hasBoundary(IntakePump1, BooleanRange)

rangeF rom(BooleanRange, 0), rangeT o(BooleanRange, 1)

Hình 7.2 cho thấy chế độ xem đồ họa của biểu diễn ranh giới. Tương tự như vậy, chúng tôi có thể thêm nhiều ranh giới và ánh xạ phụ thuộc hơn vào thuật ngữ.



Hình 7.2: Biểu diễn ranh giới trong ontology

Phụ thuộc

Ngoài các ranh giới mô tả hành vi an toàn của các thành phần, nó cũng có thể phụ thuộc lẫn nhau. Ví dụ, nếu mực nước trong bể là tối đa, thì máy bơm phải tắt. Điều này có thể được chính thức hóa trong ontology như

waterLevel(T ank1, l1) l1 > ngưỡng levelT maxW aterLevel(T ank1, 'true')

maxW aterLevel(T ank1, 'true') isRunning(P ump1, 'f alse')

trong đó waterLevel là một thuộc tính ánh xạ lớp Tank với các giá trị số nguyên, levelThreshold là mức nước tối đa cho phép của một bể chứa, maxWaterLevel ánh xạ trạng thái của Tank và isRunning ánh xạ trạng thái đang chạy của Pump.

7.4 Nghiên cứu tình huống

7.4.1 Chuẩn bị

OWL-API

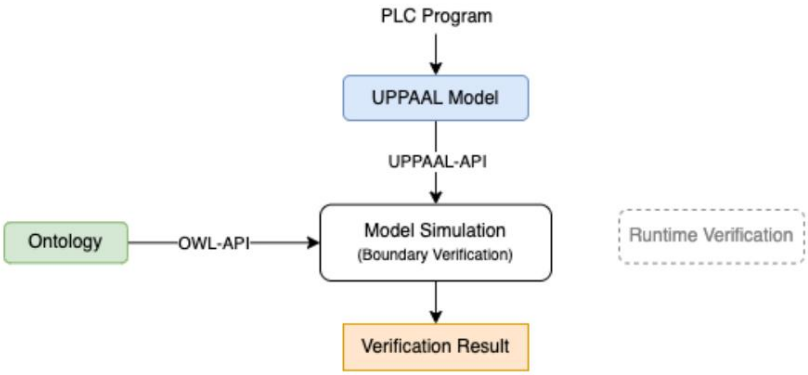
OWL là một ngôn ngữ dựa trên logic tính toán được thiết kế để biểu diễn các dữ liệu phong phú và phức tạp kiến thức về mọi thứ và mối quan hệ giữa chúng. OWL-API [48] là ban đầu được phát triển vào năm 2003 để hỗ trợ việc tạo và thao tác OWL ontology. OWL-API được thiết kế để hỗ trợ việc thao tác T-Box, nghĩa là, ontology cấp lược đồ. Nó được triển khai bằng Java và có sẵn mã nguồn mở. Một một trong những lợi thế của OWL-API là chúng ta có thể sử dụng ontology trong bộ nhớ cùng nhau với thuật ngữ được sử dụng trong cơ sở dữ liệu.

UPPAAL-API

UPPAAL-API [70] là một API cho trình kiểm tra mô hình UPPAAL được triển khai trong Java để tạo, thao tác và mô phỏng các mô hình UPPAAL. API đọc mô hình được viết trong các tệp XML và tạo ra một đối tượng của lớp UppaalSystem được cung cấp bởi UPPAAL. Nó cung cấp hai gói thiết yếu: (1) động cơ chứa chính công cụ kiểm tra mô hình cũng như công cụ mô phỏng mô hình và (2) mô hình cung cấp các lớp, phương pháp và cấu trúc dữ liệu cần thiết để biểu diễn mô hình của một hệ thống.

7.4.2 Tích hợp mô phỏng mô hình UPPAAL và Ontology

Đối với thử nghiệm logic chương trình, chúng tôi mô phỏng hành vi của chương trình PLC trong một mô hình kiểm tra. Đối với mô phỏng chương trình động, chúng tôi sử dụng trình mô phỏng UPPAAL có sẵn với trình kiểm tra mô hình UPPAAL. Để có tính linh hoạt hơn trong phân tích và để có triển khai xác minh tự động, chúng tôi đã sử dụng API UPPAAL cung cấp.



Hình 7.3: Triển khai tổng thể

Hình 7.3 cho thấy việc triển khai tổng thể phương pháp tiếp cận của chúng tôi. OWL-API được sử dụng để truy cập thông tin như giá trị ranh giới từ ontology. UPPAAL-API là được sử dụng để truy cập và mô phỏng mô hình UPPAAL TA. Xác minh ranh giới là được thực hiện trong khi mô phỏng mô hình.

Xác minh ranh giới

Thuật toán 2 cho thấy thông tin ranh giới được trích xuất từ thuật ngữ học và được sử dụng để kiểm tra logic trong mô phỏng mô hình. Đầu tiên, tất cả các biến và thành phần có ranh giới cần được kiểm tra được khởi tạo trong một danh sách. Sau đó, ValueRange lớp từ ontology được truy cập để gán phạm vi cho biến trong mô phỏng.

Sau đó, một mô phỏng mô hình được thực hiện trong đó trong mỗi vòng lặp mô phỏng các biến được kiểm tra xem nó có nằm trong ranh giới không. Nếu nó nằm ngoài phạm vi, mô phỏng sẽ in đưa ra cảnh báo về vi phạm. Thuật toán này được triển khai trong Java và có thể được tìm thấy trên Github [71].

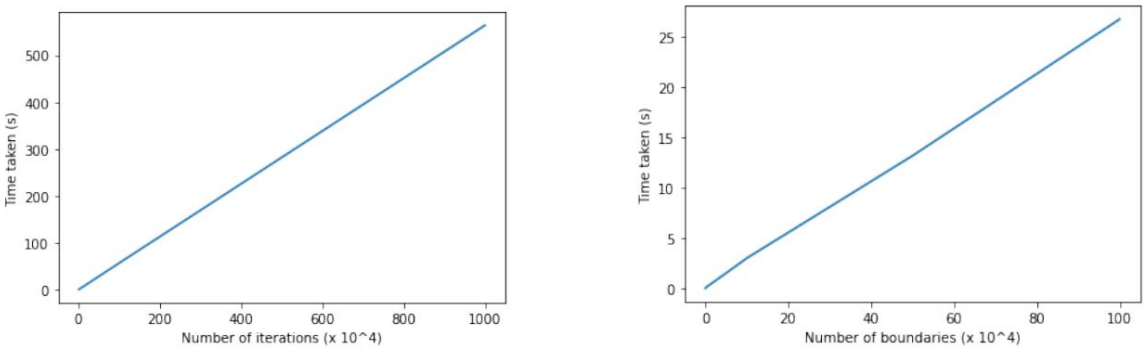
7.5 Kết quả và thảo luận

```
BoundaryA = [50, 1000]
BoundaryB = [50, 1000]
BoundaryC = [70, 1000]
13151, t ≥ 6, chemicalProcess: C8 → C9, mixer: running → idle, Warnings: C = 68,
13152, t ≥ 10, chemicalProcess: C9 → C10, outtakePump: idle → running, Warnings: C = 68,
13153, t ≥ 10, chemicalProcess: C10 → C11, outtakePump: running → idle, Warnings: C = 68,
13154, t ≥ 15, chemicalProcess: C11 → C0, Warnings: C = 68,
13155, t = 0, chemicalProcess: C0 → C0, fillTank: idle → RT, Warnings: C = 68,
13156, t = 0, fillTank: RT → refillC, Warnings: C = 68,
21044, t ≥ 6, chemicalProcess: C8 → C9, mixer: running → idle, Warnings: C = 68,
21045, t ≥ 10, chemicalProcess: C9 → C10, outtakePump: idle → running, Warnings: C = 68,
21046, t ≥ 10, chemicalProcess: C10 → C11, outtakePump: running → idle, Warnings: C = 68,
21047, t ≥ 15, chemicalProcess: C11 → C0, Warnings: C = 68,
21048, t = 0, chemicalProcess: C0 → C0, fillTank: idle → RT, Warnings: C = 68,
21049, t = 0, fillTank: RT → refillC, Warnings: C = 68,
35015, t ≥ 6, chemicalProcess: C8 → C9, mixer: running → idle, Warnings: C = 68,
35016, t ≥ 10, chemicalProcess: C9 → C10, outtakePump: idle → running, Warnings: C = 68,
35017, t ≥ 10, chemicalProcess: C10 → C11, outtakePump: running → idle, Warnings: C = 68,
35018, t ≥ 15, chemicalProcess: C11 → C0, Warnings: C = 68,
35019, t = 0, chemicalProcess: C0 → C0, fillTank: idle → RT, Warnings: C = 68,
35020, t = 0, chemicalProcess: C0 → C1, Warnings: C = 68,
35021, t = 0, fillTank: RT → refillC, Warnings: C = 68,
--------------------------------------------------
```

Hình 7.4: Thực hiện mô phỏng

Hình 7.4 hiển thị kết quả của một trong những lần thực hiện mô phỏng mà chúng tôi đã thực hiện. Chúng tôi đã sử dụng cùng một mô hình UPPAAL đã được thảo luận trong phần nghiên cứu tình huống của Phần I. Chúng ta có thể thấy rằng sự vi phạm ranh giới đối với C bắt đầu tại lần lặp chương trình 13151. Vào thời điểm này, quá trình chuyển đổi mô hình của mô hình quá trình hóa học là từ C8 sang C9 và trạng thái của thành phần bộ trộn thay đổi từ trạng thái chạy sang trạng thái nhàn rỗi trạng thái. Giá trị của C là 68, nằm ngoài ranh giới của C, tức là [70, 1000].
Chúng ta cũng có thể nói rằng sự vi phạm tiếp tục trong năm lần lặp lại nữa. Sau đó, sự vi phạm lại xảy ra ở các lần lặp 21044 và 35015 của chương trình.

Từ ví dụ thực hiện này, chúng ta có thể thấy rằng mô hình đi vào một điều không mong đợi



(a) so với số lần lặp lại mô phỏng (b) so với số lượng giá trị biên giới

Hình 7.5: Nghiên cứu thời gian đánh giá hiệu suất

trạng thái; nghĩa là, hệ thống có thể có một lượng hóa chất nằm ngoài giới hạn an toàn-
ary. Theo cách này, logic điều khiển được mô hình hóa trong UPPAAL TA được xác minh so với
ranh giới hiện diện trong bản thể học.

Trong thí nghiệm này, chúng tôi giới hạn số lần lặp lại ở mức 50.000 cho mô phỏng
của mô hình UPPAAL. Chúng tôi đã thực hiện nghiên cứu thời gian bằng cách thay đổi số lượng
số lần lặp lại và số lượng điều kiện biên được hiển thị trong Hình 7.5a và Hình
7.5b. Như được thấy trong biểu đồ, độ phức tạp về thời gian là tuyến tính, điều này làm cho nó khá
có thể mở rộng.

7.6 Kết luận và các công trình tương lai

Bài báo này đề xuất một khuôn khổ dựa trên ontology, bao gồm an toàn và bảo mật
kiến thức về các thành phần và biến số để xác minh ranh giới tĩnh của PLC.
Những loại công cụ và thuật toán xác minh chính thức nhanh, nhẹ này sẽ giúp ích
các kỹ thuật viên và kỹ sư kiểm tra logic điều khiển của họ ở giai đoạn đầu của thiết kế
và phát triển để đảm bảo các đặc tính an toàn và an ninh được đáp ứng trong an toàn-
các quá trình quan trọng.

Tại thời điểm này, chỉ có các giá trị biên của các biến được sử dụng trong thử nghiệm để minh họa phương pháp của chúng tôi. Trong công trình này, chúng tôi cung cấp một công cụ nền tảng cho việc tích hợp của trình kiểm tra mô hình UPPAAL chính thức và thuật ngữ OWL. Các công cụ này có thể là được phát triển thêm để làm phong phú thêm các công cụ kiểm tra và xác minh có sẵn.

TÀI LIỆU THAM KHẢO

[1] CL Heitmeyer, RD Jeffords và BG Labaw, “Tính nhất quán tự động
kiểm tra các thông số kỹ thuật yêu cầu,” ACM Trans. Softw. Eng. Methodol.,
tập 5, trang 231–261, tháng 7 năm 1996.

[2] Xiaoshan Li, Zhiming Liu và Jifeng He, “Kiểm tra tính nhất quán của UML lại
“quirements,” trong Hội nghị quốc tế lần thứ 10 của IEEE về Kỹ thuật phức hợp
Hệ thống máy tính (ICECCS'05), trang 411–420, tháng 6 năm 2005.

[3] B. Kamsu-Foguem, FH Abanda, MB Doumbouya và JF Tchouanguem,
“Lý luận về ontology dựa trên đồ thị để xác minh chính thức các quy tắc BREEAM,”
Cogn. Syst. Res., tập 55, trang 14–33, tháng 6 năm 2019.

[4] BF Adiego, EB Vi˜nuela, và A. Merezhin, “Kiểm tra và xác minh PLC
mã để kiểm soát quy trình,” tháng 10 năm 2013.

[5] J. Song, E. Jee, và D.-H. Bae, “FBDTester 2.0: Trình tự thử nghiệm tự động
thế hệ cho các chương trình FBD với các trạng thái bộ nhớ trong,” Khoa học máy tính
Lập trình, tập 163, trang 115–137, tháng 10 năm 2018.

[6] SH Lee, SJ Lee, J. Park, E.-C. Lee, và HG Kang, “Sự phát triển của
môi trường thử nghiệm dựa trên mô phỏng cho phần mềm quan trọng đối với an toàn,” 2018.

[7] S. Richter và JU Witiig, “Quy trình xác minh và xác nhận cho I&C an toàn
hệ thống,” Nucl. Plant J., tập 21, số 3, trang 36–+, 2003.

[8] D. Darvas, I. Majzik, và EB Vi˜nuela, “Bản dịch chương trình PLC để xác minh
mục đích cation,” Periodica Polytechnica Kỹ thuật Điện và Máy tính
Khoa học, tập 61, trang 151–165, tháng 5 năm 2017.

[9] S. Kottler, M. Khayamy, SR Hasan và O. Elkeelany, “Xác minh chính thức
chương trình logic bậc thang sử dụng NuSMV,” 2017.

[10] R. Sun, A. Mera, L. Lu, và D. Choffnes, “SoK: Các cuộc tấn công vào kiểm soát công nghiệp
logic và các biện pháp phòng thủ dựa trên xác minh chính thức,” tháng 6 năm 2020.

[11] O. Ljungkrantz, K. ˚Akesson, M. Fabian và C. Yuan, “Một thông số kỹ thuật chính thức-
ngôn ngữ lập trình cho logic điều khiển dựa trên PLC,” trong Hội nghị quốc tế IEEE lần thứ 8 năm 2010
Hội nghị về tin học công nghiệp, trang 1067–1072, tháng 7 năm 2010.

[12] M. Rausch và BH Krogh, “Xác minh chính thức các chương trình PLC,” 1998.

[13] D. Soliman và G. Frey, “Xác minh và xác nhận các ứng dụng an toàn dựa trên
về các khối chức năng an toàn PLCopen sử dụng máy tự động hẹn giờ trong uppaal,” 2009.

[14] A. Al Farooq, E. Al-Shaer, T. Moyer và K. Kant, “IoTC2: Một phương pháp chính thức
phương pháp tiếp cận để phát hiện xung đột trong các hệ thống IoT quy mô lớn,” năm 2019 IFIP/IEEE
Hội thảo về Quản lý dịch vụ và mạng tích hợp (IM), trang 442–447,
Tháng 4 năm 2019.

[15] F. Baader, A. Bauer và M. Lippmann, “Xác minh thời gian chạy bằng cách sử dụng một
logic mô tả,” trong Frontiers of Combining Systems, trang 149–164, Springer
Berlin Heidelberg, 2009.

[16] J.-b. Gao, B.-w. Zhang, X.-h. Chen, và Z. Luo, “Mô hình dựa trên bản thể học của
tấn công mạng và máy tính để đánh giá an ninh,” Tạp chí Thượng Hải
Đại học Giao thông (Khoa học), tập 18, số 5, trang 554–562, 2013.

[17] Y. Wu, RA Gandhi và H. Siy, “Sử dụng các mẫu ngữ nghĩa để nghiên cứu các lỗ hổng
các khả năng được ghi lại trong các kho phần mềm lớn,” trong Biên bản báo cáo của ICSE năm 2010
Hội thảo về Kỹ thuật phần mềm cho hệ thống an toàn, trang 22–28, 2010.

[18] J. Undercoffer, A. Joshi và J. Pinkston, “Mô hình hóa các cuộc tấn công máy tính: Một
ontology để phát hiện xâm nhập,” trong Hội thảo quốc tế về những tiến bộ gần đây
trong Phát hiện xâm nhập, trang 113–135, Springer, 2003.

[19] H. Knublauch, M. Horridge, MA Musen, AL Hiệu trưởng, R. Stevens, N. Drum-
mond, PW Lord, NF Noy, J. Seidenberg, và H. Wang, “Người được bảo trợ OWL
“trải nghiệm”, trong OWLED, 2005.

[20] B. Parsia và E. Sirin, “Viên thuốc: Một lý luận viên cú,” trong Third international
áp phích hội nghị web ngữ nghĩa, tập 18, trang 13, Citeseer, 2004.

[21] DL McGuinness, F. Van Harmelen và những người khác, “Ngôn ngữ ontology web OWL
“Tổng quan,” khuyến nghị của W3C, tập 10, số 10, trang 2004, 2004.

[22] MJ O'Connor và A. Das, “SWRLTab: Một môi trường có thể mở rộng cho
làm việc với các quy tắc SWRL trong Prot´eg´e-OWL,” tháng 1 năm 2006.

[23] J. Bengtsson, K. Larsen, F. Larsson, P. Pettersson và W. Yi, “UPPAAL—a
bộ công cụ để xác minh tự động các hệ thống thời gian thực,” trong International hybrid
hội thảo hệ thống, trang 232–243, 1995.

[24] Y. Yin, KE Stecke, và D. Li, “Sự tiến hóa của hệ thống sản xuất từ
công nghiệp 2.0 đến công nghiệp 4.0,” Int. J. Prod. Res., tập 56, trang 848–861, tháng 1.
2018.

[25] KE Hemsley và DRE Fisher, “Lịch sử của hệ thống điều khiển công nghiệp mạng
sự cố”, đại diện kỹ thuật, Phòng thí nghiệm quốc gia Idaho, tháng 12 năm 2018.

[26] GM Makrakis, C. Kolias, G. Kambourakis, C. Rieger và J. Benjamin, “Công nghiệp
thử nghiệm và bảo mật cơ sở hạ tầng quan trọng: Phân tích kỹ thuật về bảo mật thực tế
sự cố,” IEEE Access, tập 9, trang 165295–165325, 2021.

[27] M. Yu, Z. Wang, và X. Niu, “Xác minh mô hình biên đạo dịch vụ dựa trên
logic mô tả,” Toán. Lý thuyết. Kỹ thuật, tập 2016, tháng 1 năm 2016.

[28] JF Baget, M. Chein, M. Croitoru và những người khác, “Kiến thức logic, dựa trên đồ thị
đại diện với CoGui,” GAOC: Graphes et, 2010.

[29] M. Croitoru, Biểu diễn kiến thức và lý luận dựa trên đồ thị: AI thực tế
ứng dụng. Luận án tiến sĩ, Đại học Montpellier 2, tháng 11 năm 2014.

[30] S. Guo, M. Wu, và C. Wang, “Thực hiện biểu tượng của logic lập trình được
mã troller,” trong Biên bản cuộc họp chung lần thứ 11 năm 2017 về Nền tảng của
Kỹ thuật phần mềm, ESEC/FSE 2017, (New York, NY, Hoa Kỳ), trang 326–336,
Hiệp hội máy móc điện toán, tháng 8 năm 2017.

[31] SC Park, CM Park, G. Wang, J. Kwak và S. Yeo, “PLCStudio: Mô phỏng
dựa trên xác minh mã PLC,” trong Hội nghị mô phỏng mùa đông năm 2008, trang 222–
228, tháng 12 năm 2008.

[32] L.-J. Koo, CM Park, CH Lee, S. Park, và G.-N. Wang, “Khung mô phỏng-
công việc xác minh các chương trình PLC trong ngành công nghiệp ô tô,” Int. J. Prod.
Res., tập 49, trang 4925–4943, tháng 8 năm 2011.

[33] G. Palshikar và KV Nori, “Đặc tả chính thức các chương trình PLC sử dụng tem-
logic thông thường,” tháng 12 năm 1994.

[34] A. Fuxman, M. Pistore, J. Mylopoulos và P. Traverso, “Kiểm tra mô hình sớm
yêu cầu thông số kỹ thuật trong tropos,” trong Biên bản báo cáo IEEE quốc tế lần thứ năm
Hội thảo về Kỹ thuật Yêu cầu, trang 174–181, tháng 8 năm 2001.

[35] P. Filipovikj, M. Nyberg, và G. Rodriguez-Navas, “Đánh giá lại mô hình-
phương pháp tiếp cận dựa trên việc chính thức hóa các yêu cầu trong lĩnh vực ô tô,” vào năm 2014
Hội nghị Kỹ thuật Yêu cầu Quốc tế lần thứ 22 của IEEE (RE), trang 444–
450, tháng 8 năm 2014.

[36] EM Clarke, “Kiểm tra mô hình,” trong Nền tảng của Công nghệ phần mềm và
Khoa học máy tính lý thuyết, trang 54–56, Springer Berlin Heidelberg, 1997.

[37] N. Nourollahi, “Xác minh và kiểm tra mô hình TCTL của hệ thống thời gian thực trên
automata định thời và cấu trúc kripke định thời, và các vấn đề chuyển đổi cảm ứng trong
thời gian dày đặc,”

[38] H. Bel Mokadem, B. B'erard, V. Gourcuff, O. De Smet, và J. Roussel, “Xác minh-
của một hệ thống đa nhiệm được định thời gian với uppaal,” IEEE Trans. Autom. Sci. Eng.,
tập 7, trang 921–932, tháng 10 năm 2010.

[39] X. Lin, H. Zhang và M. Gu, “OntCheck: Một tính chính xác tĩnh được thúc đẩy bởi Ontology

công cụ kiểm tra cho các mô hình dựa trên thành phần," J. Appl. Math., tập. 2013, tháng 4. 2013.

[40] I. Friedberg, K. McLaughlin, P. Smith, D. Laverty và S. Sezer, "STPA-SafeSec: Phân tích an toàn và bảo mật cho các hệ thống mạng vật lý," Tạp chí thông tin Bảo mật và Ứng dụng, tập 34, trang 183–196, tháng 6 năm 2017.

[41] S.-˚A. T¨arnlund, "Khả năng tính toán mệnh đề Horn," BIT, tập 17, trang 215–226, tháng 6 1977.

[42] P. Haase và J. V¨olker, "Học tập và lý luận về bản thể học – giải quyết sự không chắc chắn sự ô uế và không nhất quán," trong Ghi chú bài giảng về Khoa học máy tính, Ghi chú bài giảng trong khoa học máy tính, trang 366–384, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[43] R. Alur, C. Courcoubetis và D. Dill, "Kiểm tra mô hình trong thời gian thực dày đặc," Inform. and Comput., tập 104, trang 2–34, tháng 5 năm 1993.

[44] M. Kanovich, TB Kirigin, V. Nigam, A. Scedrov và C. Talcott, "Rời rạc so với. thời gian dày đặc trong việc phân tích các giao thức bảo mật mạng-vật lý," trong Nguyên tắc về An ninh và Niềm tin, trang 259–279, Springer Berlin Heidelberg, 2015.

[45] R. Alur và DL Dill, "Một lý thuyết về máy tự động định thời*," Máy tính lý thuyết Khoa học, tập 126, trang 183–235, 1994.

[46] N. Mahmud, C. Seceleanu, và O. Ljungkrantz, "Đặc điểm kỹ thuật và ngữ nghĩa Phân tích các yêu cầu của hệ thống nhúng: Từ logic mô tả đến nhịp độ logic thực tế," trong Kỹ thuật phần mềm và phương pháp chính thức, trang 332–348, Springer Xuất bản quốc tế, 2017.

[47] “Velocio.net.” https://velocio.net/. Truy cập: 2022-5-9.

[48] M. Horridge và S. Bechhofer, “API OWL: Một API java cho các thuật ngữ OWL,” Ngữ nghĩa. Web, tập. 2, không. 1, trang 11–21, 2011.

[49] “Bản thể luận là gì và tại sao chúng ta cần nó.” https://protege.stanford.edu/ấn phẩm/phát triển bản thể học/ontology101-noy-mcguinness.html. Truy cập: 2022-3-1.

[50] SE Valentine, Lỗ hổng mã PLC thông qua hệ thống SCADA. Luận án tiến sĩ, Đại học Nam Carolina, 2013.

[51] A. David, KG Larsen, A. Legay, M. Mikuˇcionis, và DB Poulsen, “Uppaal Hướng dẫn SMC,” Int. J. Softw. Tools Technol. Trans., tập 17, trang 397–415, tháng 8. 2015.

[52] N. Dalwadi, B. Nagar, và A. Makwana, “Đánh giá hiệu suất của ngữ nghĩa những người lý luận,” trong Biên bản Hội nghị quốc tế lần thứ 19 về quản lý của Dữ liệu, trang 109–112, 2013.

[53] I. Friedberg, K. McLaughlin, P. Smith, D. Laverty và S. Sezer, “Stpa-safesec: Phân tích an toàn và bảo mật cho các hệ thống mạng vật lý,” Tạp chí thông tin an ninh và ứng dụng, tập 34, trang 183–196, 2017.

[54] N. Leveson, “Mô hình tai nạn mới để thiết kế hệ thống an toàn hơn,” Khoa học an toàn, tập 42, số 4, trang 237–270, 2004.

[55] NG Leveson, Kỹ thuật xây dựng một thế giới an toàn hơn: Tư duy hệ thống áp dụng vào an toàn. Nhà xuất bản MIT, 2016.

[56] TVN Nguyen và F. Irigoin, “Kiểm tra ràng buộc mảng hiệu quả và hiệu suất,”
Giao dịch ACM về Ngôn ngữ lập trình và Hệ thống (TOPLAS), tập 27,
số 3, trang 527–570, 2005.

[57] N. Hasabnis, A. Misra và R. Sekar, “Kiểm tra giới hạn trọng lượng nhẹ,” trong
Biên bản Hội nghị quốc tế lần thứ mười về thế hệ mã và
Tối ưu hóa, trang 135–144, 2012.

[58] V. Markstein, J. Cocke và P. Markstein, “Tối ưu hóa kiểm tra phạm vi,”
trong Biên bản báo cáo của hội nghị chuyên đề SIGPLAN năm 1982 về Xây dựng trình biên dịch,
trang 114–119, 1982.

[59] P. SPEET và E. Goroii, “Hỗ trợ phần cứng và tạo mã cho động
kiểm tra phạm vi trong c,”

[60] A.-Y. Turhan, “Giới thiệu về logic mô tả – một chuyến tham quan có hướng dẫn,” 2013.

[61] KG Larsen, P. Pettersson, và W. Yi, “Uppaal tóm lại,” Quốc tế
Tạp chí về công cụ phần mềm chuyển giao công nghệ, tập 1, số 1, trang 134–152, 1997.

[62] A. Aggarwal và P. Jalote, “Tích hợp phân tích tĩnh và động để giải thích
“kiểm tra lỗ hổng bảo mật”, trong Hội nghị phần mềm máy tính quốc tế lần thứ 30 và
Hội nghị ứng dụng (COMPSAC'06), tập 1, trang 343–350, IEEE, 2006.

[63] Z. Tzermias, G. Sykiotakis, M. Polychronakis và EP Markatos, “Kết hợp
phân tích tĩnh và động để phát hiện các tài liệu độc hại,” trong
Biên bản Hội thảo Châu Âu lần thứ tư về An ninh Hệ thống, trang 1–6,
2011.

[64] M. Alvares, T. Marwala và FB de Lima Neto, “Ứng dụng của máy tính
trí thông minh cho phần mềm tĩnh kiểm tra chống lại lỗ hổng hỏng bộ nhớ
“tie,” trong Hội nghị chuyên đề IEEE năm 2013 về Trí tuệ tính toán trong An ninh mạng
(CICS), trang 59–66, tháng 4 năm 2013.

[65] C. Zimmer, B. Bhat, F. Mueller và S. Mohan, “Phát hiện xâm nhập dựa trên thời gian
trong các hệ thống mạng vật lý,” trong Biên bản báo cáo của Hội nghị quốc tế ACM/IEEE lần thứ nhất
Hội nghị về Hệ thống mạng vật lý, trang 109–118, 2010.

[66] X. Zheng, C. Julien, M. Kim, và S. Khurshid, “Nhận thức về trạng thái của
nghệ thuật trong việc xác minh và xác nhận trong các hệ thống mạng vật lý,” IEEE Systems
Tạp chí, tập 11, số 4, trang 2614–2627, 2015.

[67] ER Griffor, C. Greer, DA Wollman, MJ Burns, et al., “Khung cho
hệ thống mạng vật lý: Tập 1, tổng quan,” 2017.

[68] S. Vasanthapriyan, J. Tian, D. Zhao, S. Xiong, và J. Xiang, “Một bản thể học dựa trên
hệ thống quản lý kiến thức để kiểm thử phần mềm,” trong SEKE, trang 230–235,
2017.

[69] EF d. Souza, R.d. A. Falbo và NL Vijaykumar, “ROoST: Tài liệu tham khảo
ontology về thử nghiệm phần mềm,” Appl. Ontol., tập 12, trang 59–90, tháng 3 năm 2017.

[70] “Java API :: Tài liệu UPPAAL.” https://docs.uppaal.org/
toolsandapi/javaapi/. Truy cập: 2022-3-31.

[71] R. Neupane, “testmerge: Triển khai nghiên cứu.” https://github.com/
codeezer/testMerge.