LỜI CẢM ƠN

Trước hết em xin gửi lời cảm ơn đến thầy Đỗ Văn Chiểu, người đã hướng dẫn em rất nhiều trong suốt quá trình tìm hiểu nghiên cứu và hoàn thành khóa luận này từ lý thuyết đến ứng dụng. Sự hướng dẫn của thầy đã giúp em có thêm được những hiểu biết về một số vấn đề liên quan đến Otomat thời gian và hệ thời gian thực. Qua những phần lý thuyết này cũng lôi cuốn em và sẽ trở thành hướng nghiên cứu tiếp của em sau khi tốt nghiệp.

Đồng thời em cũng xin chân thành cảm ơn các thầy cô trong bộ môn cũng như các thầy cô trong trường đã trang bị cho em những kiến thức cơ bản cần thiết để em có thể hoàn thành tốt khóa luận này.

Em xin gửi lời cảm ơn đến các thành viên lớp CT1001, những người bạn đã luôn ở bên cạnh động viên, tạo điều kiện thuận lợi và cùng em tìm hiểu, hoàn thành tốt khóa luận.

Sau cùng, em xin gửi lời cảm ơn đến gia đình, bạn bè đã tạo mọi điều kiện để em xây dựng thành công khóa luận này.

Hải Phòng ngày 10, tháng 6 năm 2010

Sinh viên thực hiện

Đặng Thanh Tâm

MỤC LỤC

DANH MỤC TỪ VIẾT TẮT

Ký hiệu viết tắt	Giải thích
RTS	**Real Time System**
TA	**Timed Automata**

LỜI MỞ ĐẦU

Xã hội phát triển, công nghệ thông tin ngày càng giữ vị trí quan trọng trong đời sống và kỹ thuật. Nó giúp cho các nhà quản lý kinh doanh, thương mại ,quân đội,các hoạt động của con người trong nhiều lĩnh vực đem lại hiệu quả cao. Đặc biệt vấn đề về điều khiển đã hỗ trợ con người đắc lực trong việc điều khiển tự đông. Việc xây dựng các phần mềm như vậy đòi hỏi phần mềm thực thi phải có độ chính xác cao đặc biệt là luôn có các ràng buộc liên quan đến thời gian. Hiện nay có rất nhiều các nghiên cứu cũng như các phần mềm hỗ trợ cho việc thiết kế hệ thống thời gian thực được ứng dụng trong nhiều lĩnh vực khác nhau ví dụ như: hệ thống điều khiển không lưu, điều khiển tàu biển, các hệ thống ô tô, máy bay, tàu hỏa… Có nhiều các công cụ khác nhau từ lý thuyết đến thực nghiệm để làm việc này, tuy nhiên, đối với các hệ thống thời gian là một công cụ lý thuyết được rất nhiều người dùng với phần mềm trợ đặc tả Uppaal.

Đồ án trình bày trong ba chương với nội dung mỗi chương như sau:

Chương 1: Tìm hiểu về Model checking và lý thuyết hệ thời gian thực (Real timed system).

Chương 2 : Tìm hiểu về lý thuyết Otomat thời gian (Timed automata)

Chương 3 : Thử nghiệm đặc tả hệ thống với công cụ Uppaal.

CHƯƠNG 1:
TÌM HIỂU VỀ MODEL CHECKING VÀ LÝ THUYẾT VỀ HỆ THỜI GIAN THỰC

1.1 ĐÔI NÉT VỀ MÔ HÌNH KIỂM TRA (MODEL CHECKING) TRONG CÔNG NGHỆ PHẦN MỀM

Trong lĩnh vực khoa học máy tính logic, logic có thỏa mãn được cấu trúc nhất định.Model checking đề cập đến các vấn đề sau đây: Cho một mô hình của một hệ thống, kiểm tra tự động mô hình này xem có đáp ứng một số đặc điểm kỹ thuật đã cho hay không. Thông thường, một trong những hệ thống có trọng tâm là phần cứng hoặc các hệ thống phần mềm thì đặc điểm kỹ thuật yêu cầu về *safety* (an toàn) không thể vắng mặt của *deadlocks* và các trạng thái quan trọng tương tự mà có thể gây ra hệ thống sụp đổ. Để giải quyết như một vấn đề của thuật toán, cả hai mô hình của hệ thống và đặc điểm kỹ thuật được xây dựng ở một số ngôn ngữ toán học chính xác. Để kết thúc vấn đề này, nó được xây dựng như là một nhiệm vụ trong cụ thể là để kiểm tra xem có thỏa mãn công thức đạt kết quả yêu cầu không. Một vấn đề kiểm tra đơn giản mô hình là là đi kiểm tra xác minh xem liệu một công thức trong mệnh đề logic có thỏa mãn được cấu trúc nhất định.

Model checking là một kỹ thuật trong việc kiểm định các hệ thống hữu hạn trạng thái đồng thời như các thiết kế mạch tuần tự và các giao thức giao tiếp. Nó có một số lợi ích trên các tiếp cận truyền thống đó là dựa trên các mô phỏng, kiểm tra, suy diễn, lập luận. Một cách cụ thể, Model Checking là một kĩ thuật tự động hoàn toàn nhanh,nếu ở phần thiết kế có chứa lỗi Model Checking sẽ sản sinh ra một phản ví dụ mà có thể được sử dụng để xác định ra nguồn gốc của lỗi.

Thách thức chính trong Model checking là giải quyết bài toán bùng nổ các trạng thái bài toán này xảy ra trong hệ thống với nhiều thành phần tương tác với nhau hoặc các hệ thống với các cấu trúc dữ liệu có thể giả định rất nhiều dữ liệu khác nhau. Trong trường hợp như vậy số các trạng thái toàn cục có thể là rất lớn. Các nhà nghiên cứu đã có những bước tiến đáng kể trong việc giải các bài toán này trong những năm gần đây.

1.1.1 Các yêu cầu của mô hình hệ thống

- **Safety (an toàn):** Hệ thống sẽ không phát sinh lỗi (some thing bad will never happen).
- **Liveness**: Thuộc tính này đảm bảo rằng thuộc tính tốt nhất định sẽ xảy ra.(something "good" will happen but we don't know when)

1.2 GIỚI THIỆU VỀ HỆ THỜI GIAN THỰC

1.2.1 Khái niệm

Trong các tài liệu cũng như trong thực tế, khái niệm thời gian thực và hệ thống thời gian thực không phải lúc nào cũng được hiểu một cách thống nhất. Nhiều người cho rằng hệ thống thời gian thực là một hệ thống phải làm việc với yêu cầu thời gian rất nhanh. Ví dụ các hệ thống điều khiển tay máy, điều khiển động cơ,… Lại có người cho rằng thời gian thực là thời gian tuyệt đối, hệ thống thời gian thực là một hệ thống có khả năng làm việc với thời gian tuyệt đối theo giờ phút giây ngày tháng. Vậy chúng ta nên hiểu thế nào là hệ thống thời gian thực?

Hệ thống thời gian thực (RTS – Real Time System) là một hệ thống mà tính đúng đắn của nó không chỉ phụ thuộc vào các kết quả logic nó tạo ra mà còn phụ thuộc vào các thời điểm các kết quả được tạo ra.

Như vây, hệ thống thời gian thực là một hệ thống mà sự hoạt động tin cậy của nó chỉ phụ thuộc vào sự chính xác của kết quả mà còn phụ thuộc vào thời điểm đưa ra kết quả. Hệ thống có lỗi khi yêu cầu về thời gian không được thỏa mãn.

Hệ thống thời gian thực được thiết kế nhằm trả lời lại các yếu tố kích thích phát sinh từ các thiết bị phần cứng.

Trong thực tế, các yếu tố kích thích xảy ra trong thời gian rất ngắn vào khoảng vài mili giây, thời gian mà hệ thống trả lời lại yếu tố kích thích đó tốt nhất vào khoảng dưới một giây, khoảng thời gian để xử lý một sự kiện như vậy được gọi là dealtime, dealtime được xác định bởi thời gian bắt đầu và thời gian hoàn tất công việc.

Trong RTS cũng cần quan tâm xem những công việc thời gian thực có tuần hoàn hay không. Đối với real-time được hiểu công việc tuần hoàn *dealtime* ấn định theo từng chu kì xác định, đối với công việc không tuần hoàn *dealtime* xác định vào

lúc bắt đầu công việc. Các biến cố kích hoạt công việc không tuần hoàn thường dựa trên kỹ thuật xử lý ngắt của hệ thống phần cứng.

Một hệ thống *realtime* được hiểu là một hệ thống làm việc với các sự kiện tức thời (*realtime*). Tuy nhiên không phải mọi hệ thống đều có thể thực hiện được những qui định tức thời hay đáp trả lại sự kiện một cách tức thời như chúng ta mong muốn. Khi bắt tay xây dựng các ứng dụng phần mềm chúng ta luôn mong muốn thời gian trễ để đưa ra một lệnh hay một quyết định là nhỏ nhất, hay khi xây dựng các ứng dụng phần cứng chúng ta lại muốn thời gian đưa ra một tín hiệu đáp trả một sự kiện là phải gần như tức thời, nhưng sự thực không được như vậy vì các hệ thống đáp ứng sự kiện bao giờ cũng có một thời gian trễ nhất định. Khái niệm "hệ thống thời gian thực",ở đây hiểu ngầm như một hệ thống đáp ứng sự kiện với một thời gian trễ chấp nhận được hay nói cách khác, hệ thời gian thực là hệ thống có các ràng buộc về thời gian đối với các hành động của hệ thống.

1.2.2 Đặc điểm của hệ thống thời gian thực

Tính bị động: hệ thống phải phản ứng với các sự kiện xuất hiện vào các thời điểm thường không biết trước. Ví dụ: Sự vượt ngưỡng của một giá trị đo, sự thay đổi trạng thái của một thiết bị quá trình phải dẫn đến các phản ứng trong bộ điều khiển.

Tính nhanh nhạy: Hệ thống phải xử lý thông tin một cách nhanh chóng để có thể đưa ra kết quả phản ứng một cách kịp thời. Tuy tính nhanh nhạy là một đặc điểm tiêu biểu nhưng một hệ thống có tính năng thời gian thực không nhất thiết phải có đáp ứng thật nhanh mà quan trọng hơn là phải có phản ứng kịp thời đối với các yêu cầu, tác động bên ngoài.

Tính đồng thời: hệ thống phải có khả năng phản ứng và xử lý đồng thời nhiều sự kiện diễn ra. Có thể cùng một lúc bộ điều khiển được yêu cầu thực hiện nhiều vòng điều chỉnh, giám sát ngưỡng giá trị nhiều đầu vào, cảnh giới trạng thái làm việc của một số động cơ.

Tính tiền định: Dự đoán được thời gian phản ứng tiêu biểu, thời gian phản ứng chậm nhất cũng như trình tự đưa ra các phản ứng. Nếu một bộ điều khiển phải xử lý đồng thời nhiều nhiệm vụ ta phải tham gia quyết định được về trình tự thực hiện các

công việc và đánh giá được thời gian xử lý mỗi công việc. Như vậy người sử dụng mới có cơ sở để đánh giá về khả năng đáp ứng tính thời gian thực của hệ thống

1.2.3 Cấu tạo hệ thời gian thực

RTS có cấu tạo từ các thành tố chính sau:

Đồng hồ thời gian thực: cung cấp thông tin thời gian thực.

Bộ điều khiển ngắt: quản lý các biến cố không theo chu kỳ.

Bộ định biểu: quản lý các quy trình thực hiện.

Bộ quản lý tài nguyên: cung cấp các tài nguyên máy tính.

Bộ điều khiển thực hiện: khởi động các tiến trình.

Các thành tố trên có thể được phân định là thành tố cứng hay mềm tùy thuộc vào hệ thống và ý nghĩa sử dụng. Thông thường, các RTS được kết hợp vào phần cứng có khả năng tốt hơn so với hệ thống phần mềm có khả năng tương ứng và tránh được chi phí quá đắt cho việc tối ưu hóa phần mềm.

Hệ thống thời gian thực là hệ thống có ràng buộc về thời gian, tuy nhiên nếu thời gian ràng buộc này bị vi phạm (thời gian trả lời vượt quá giới hạn cho phép) hệ thống vẫn tiếp tục hoạt động bình thường, tác hại không đáng kể.

1.2.4 Hệ điều hành thời gian thực

Các hệ thống thông thường nhằm mục tiêu: tính chính xác,năng lực thực hiện trung bình ở mức có thể chấp nhận được, trong khi đó các hệ thống thời gian thực, một số tiêu chí được đề cao, tính chính xác, tính kịp thời, hoạt động phải có tính nhất quán. Hệ điều hành hệ thời gian thực cũng có các chức năng giống các hệ điều hành thông thường, điểm khác là hệ điều hành thời gian thực phải có các đặc điểm thỏa mãn các tiêu chí của hệ thống thời gian thực.

Yêu cầu được đặt ra với một hệ điều hành thời gian thực là khả năng dự đoán trước. Yêu cầu thứ hai là có thể nhận biết và điều khiển tất cả các thành phần hệ thống trong các hệ điều hành truyền thống phần lớn phần cứng và hệ thống là ẩn hoặc trừu tượng đối với người sử dụng hoặc các nhà thiết kế ứng dụng. Tuy nhiên người sử dụng các hệ điều hành thời gian thực phải truy cập và điều khiển được các hành vi của các

thành phần hệ thống để đảm bảo tính tiên đoán được. Yêu cầu thứ ba là hệ điều hành thời gian thực nên là hệ thống mở, nghĩa là hệ thống định nghĩa một tập các cơ chế mềm dẻo, phù hợp chứ không tập trung vào chiến lược cụ thể.

1.2.5 Vì sao chọn hệ thời gian thực

Có thể nói tất cả các hệ thống điều khiển là hệ thời gian thực. Ngược lại một số lớn các hệ thống thời gian thực là các hệ thống điều khiển, không hệ thống điều khiển nào có thể hoạt động bình thường nếu như nó không đáp ứng được các yêu cầu về thời gian.

Hệ thống thời gian thực được ứng dụng phổ biến trong rất nhiều lĩnh vực như thương mại, quân đội, y tế, giá o dục, cơ sở hạ tầng… và ngày nay đang phát triển mạnh mẽ mà một số lĩnh vực tiêu biểu:

- Các hệ thống phương tiện như : ô tô, xe điện ngầm, máy bay, tàu hỏa…..
- Các hệ thống điều khiển giao thông: điều khiển không lưu, điều khiển tàu biển…

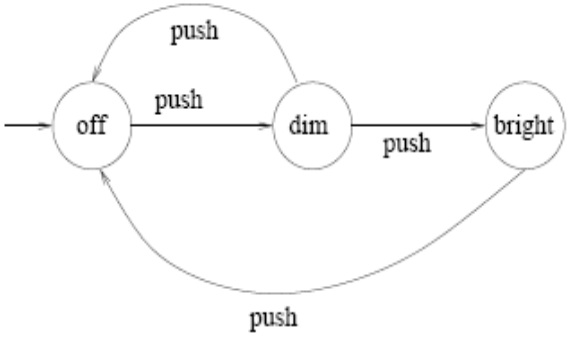
Dưới đây là một số ví dụ**:**

Hoạt động của một đèn điện có một nút bật tắt (được mô tả trong hình 1). Nguyên tắc hoạt động.

+ Ban đầu đèn tắt người dùng ấn vào nút một ấn thì đèn sáng mờ (dim)

+ Sau đó: nếu người dùng ấn vào nút thêm một ấn nữa thì đèn sáng hẳn lên (bright), nếu người dùng ấn vào nút 2 ấn liên tiếp thì đèn tắt (off).

+ Khi đèn đang ở chế độ sáng (bright) người dùng chỉ cần ấn vào nút một lầnthì đèn sẽ tắt.



Hình 1. Mô hình đèn đơn giản

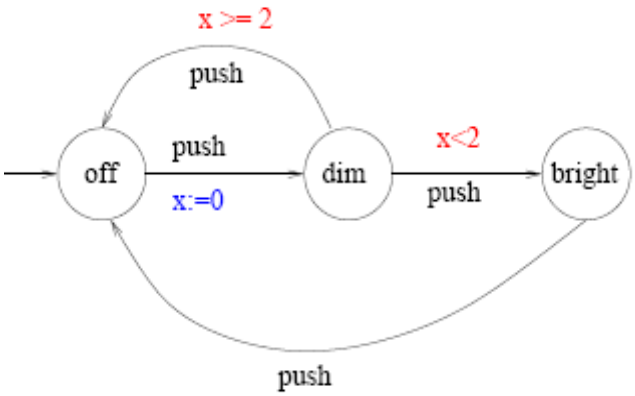
Nhưng một điều đáng lưu ý ở đây là đèn hoạt động vẫn còn nhập nhằng không rõ ràng không đưa ra một qui định cụ thể bật nút trong khoảng thời gian như thế nào thì đèn sáng mờ, đèn sáng hẳn và đèn tắt.

đồng hồ thời gian thực hoạt động của nó là:

+ Ban đầu đèn tắt người dùng ấn vào nút một cái đèn chuyển sang sáng mờ

+ Sau đó người dùng ấn nút giữ trong khoảng thời gian nhỏ hơn 2 giây thì đèn sáng nếu mà ấn nút giữ trong khoảng thời gian lớn hơn hoặc bằng 2 giây thì đèn tắt.

+ Sau khi đèn sáng lên(bright) mà người dùng ấn vào nút một cái thì đèn cũng tắt trở về trạng thái ban đầu.



Hình 2. Mô hình đèn khi có thêm ràng buộc

CHƯƠNG 2:
LÝ THUYẾT VỀ OTOMAT THỜI GIAN

2.1 ĐỊNH NGHĨA OTOMAT THOI GIAN

Một otomat thời gian (timed automata-TA) là một tập gồm 6 thành phần (L, l_0, C, A, E, I) , trong đó:

- L là tập các trạng thái (locations)
- $l_0 \in L$ là trạng thái ban đầu
- C là tập các đồng hồ
- A là tập các hành động (actions)
- $E \in L \times A \times B(C) \times 2^C \times L$ là một tập các cạnh giữa các trạng thái với một hành động, một ràng buộc là bộ đồng hồ phải được Reset về 0.
- $I : L \rightarrow B(C)$ chỉ định bất biến cho các trạng thái.

2.2 KHÁI NIỆM VỀ OTOMAT THỜI GIAN

Otomat thời gian thực là một máy hữu hạn trạng thái với một tập các đồng hồ. Mỗi đồng hồ là một hàm số ánh xạ vào một tập số thực không âm, nó ghi lại thời gian trôi qua giữa các sự kiện. Các đồng hồ được đồng bộ hóa về mặt thời gian.

Chúng ta xét một ví dụ của bộ điều khiển tự động trong việc mở và đóng cổng tại chỗ giao nhau của đường sắt. Hệ thống bao gồm 3 thành phần: tàu (TRAIN), cổng (GATE), và bộ điều khiển (CONTROLLER).

2.2.1 Mô tả cách dịch chuyển các trạng thái từng thành phần của TRAIN, GATE, CONTROLLER như sau:

❖ Mô tả mô hình TRAIN (tàu):

+ Tập các hành động là A = {approach, exit, in, out, id_T} trạng thái bắt đầu là s_0.

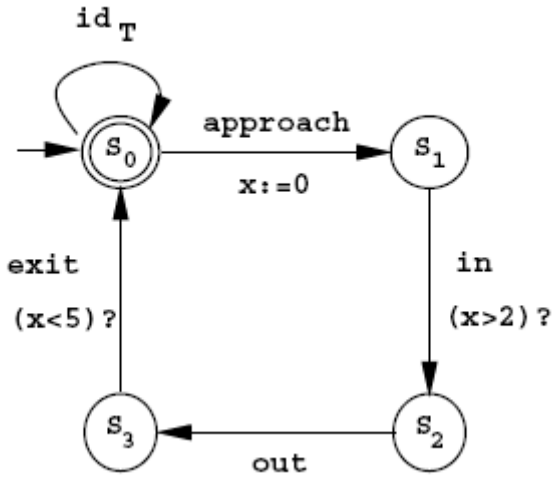
+ Với L là tập các vị trí $s_0 \in L$ là vị trí ban đầu

+ C là tập đồng hồ C = x

+ E là một ràng buộc đồng hồ phải reset về 0

- + I : L → B(C) chỉ định bất biến cho các trạng thái.

Tại vị trí ban đầu s_0 với hành động id_T để thời gian trôi qua trong lúc tàu chưa đi vào tới cổng. Tàu truyền đến bộ điều khiển (controller) với 2 hành động là tiếp cận (approach) và thoát (exit) khi tàu đi vào vị trí s_1 bộ điều khiển với hành động tiếp cận (approach) đồng hồ thời gian được reset = 0 trong khoảng thời gian mà x>2 tàu sẽ đi tiếp vào đến vị trí s_2 sau đó tàu ra khỏi vị trí s_2 ngay sang luôn vị trí s_3 lúc này nếu đồng hồ thời gian mà chạy với x<5 thì bộ điều khiển nhận được lệnh cho tàu thoát và đồng hồ được reset về 0.



Hình 3. Mô hình TRAIN

❖ Mô tả mô hình GATE (cổng)

+ Tập các hành động là A = {lower, down, raise, up}.

+ Với L là tập các vị trí $s_0 \in L$ là vị trí ban đầu

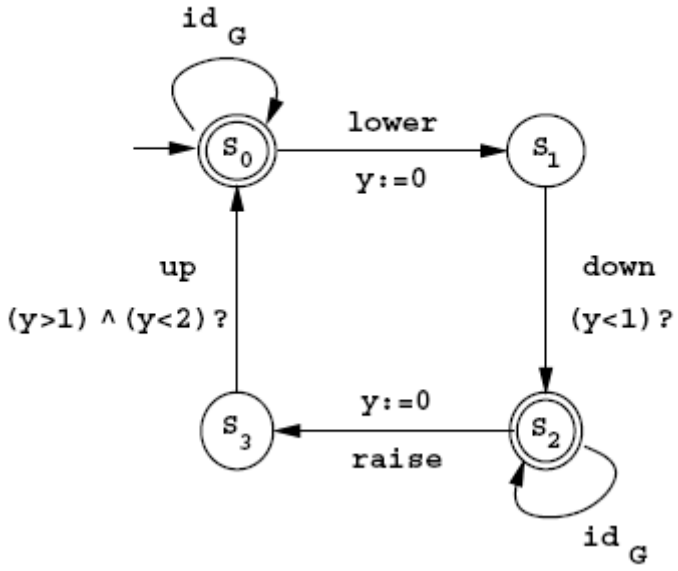
+ C là tập đồng hồ C = y

+ E là một ràng buộc đồng hồ phải reset về 0

- + I : L → B(C) chỉ định bất biến cho các trạng thái.

Tại vị trí s_0 là trạng thái ban đầu cổng (GATE) chưa nhận tín hiệu từ bộ điều khiển với hành động id_G khi đó một đồng hồ thời gian y reset y =0 lúc này tàu chuyển sang trạng thái s_1 cổng nhận được tín hiệu từ bộ điều khiển với hành động là hạ xuống (lower) ,tại s_1 này mà đồng hồ trong khoảng thời gian y < 1 bộ điều khiển sẽ phát tín

hiệu cho tàu đi xuống (down) đồng thời khi này cổng sẽ chuyển sang vị trí s_2 .Tại vị trí s_2 này cổng ở trạng thái id_G nên bộ điều khiển sẽ reset ngay đồng hồ thời gian y = 0 lại, để cho cổng tiếp tục nhận tín hiệu với hành động kéo lên (raise) và chuyển sang vị trí s_3, trong khoảng thời gian 1<y<2 thì tàu được đi lên và cổng lại được quay về trạng thái ban đầu s_0 với hành động id_G.



Hình 4. Mô hình GATE

❖ Mô hình mô tả bộ điều khiển(CONTROLLER).

+ Tập các hành động là A = {approach, exit, raise, lower, id_C}

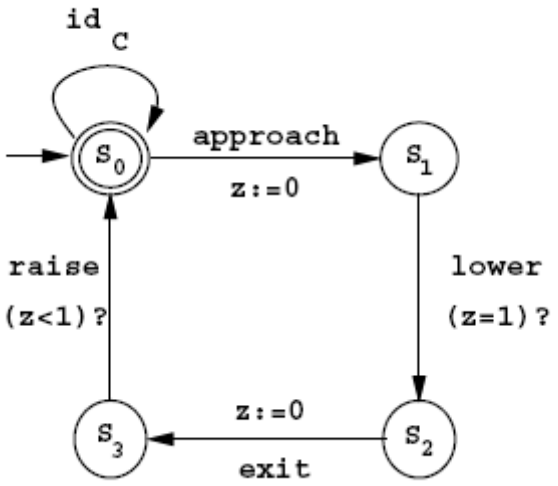
+ Với L là tập các vị trí $s_0 \in L$ là vị trí ban đầu

+ C là tập đồng hồ C = z

+ E là một ràng buộc đồng hồ phải reset về 0

- + I : L → B(C) chỉ định bất biến cho các trạng thái.

Tại vị trí s_0 là trạng thái ban đầu bộ điều khiển (CONTROLLER) với hành động id_C khi đó nó thiết lập một đồng hồ z = 0 bộ điều khiển chuyển sang trạng thái s_1, ở vị trí s_1 này mà trong khoảng thời gian z = 1gặp hành động hạ xuống từ cổng GATE (lower) thì bộ điều khiển chuyển sang trạng thái s_2 tại đây đồng hồ được thiết lập lại z = 0 và gặp hành động exit là cho tàu được thoát và bộ điều khiển chuyển sang trạng thái s_3 khi mà đồng hồ thời gian chạy trong khoảng giá trị z <1bộ điều khiển gặp hành động kéo lên của cổng GATE sau lúc đó bộ điều khiển sẽ quay trở về vị trí ban đầu s_0 với hành động id_C.



Hình 5. Mô hình CONTROLLER

2.2.2 Định nghĩa 2 (ngữ nghĩa của TA)

Cho (L, l_0, C, A, E, I) là một Otomat thời gian. Ngữ nghĩa của TA được định nghĩa như một hệ dịch chuyển được gán nhãn $< S, s_0, \rightarrow >$, trong đó $S \subseteq L \times R$ là tập các trạng thái, $s_0 = (l_0, u_0)$ là trạng thái ban đầu, $\rightarrow S \times \{IR_{\geq 0} \wedge A\} \times S$ là sự truyền có quan hệ như sau :

- $(l, u + d)$ nếu $,0 \leq d' \leq d = u + d' \in I(l)$

- (l', u') nếu có $e = (l, a, g, r, l') \in E$ sao cho $u \in g$, $u' = [r \rightarrow 0]\, u$, và $u' \in I(l)$ mà $d \in R_{\geq 0}$, $u + d$ ánh xạ mỗi đồng hồ x trong C đến giá trị $u(x) + d$, và $[r \rightarrow 0]\, u$ biểu thị giá trị đồng hồ ,mỗi giá trị trên đồng hồ là $r = 0$ và u nếu u thuộc $C \setminus r$

Một mạng Otomat là một Otomat bao gồm nhiều các Otomat thành phần được biểu diễn như sau: Cho $\mathcal{A}_i = (L_i, l_i^0, C, A, E_i, l_i)$

Trong đó $1 \leq i \leq n$, một vector trạng thái $\bar{l} = (l_1, \ldots, l_n)$

Các trạng thái bất biến $l(\bar{l}) = l(l_1)$,và ….,$l(l_n)$

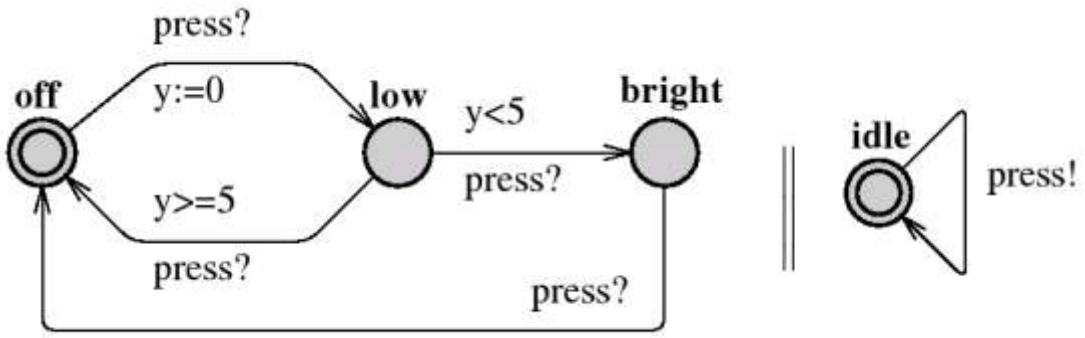
$\bar{l}[l_i / l'_i]$ kí hiệu vector trạng thái này là trạng thái l'_i thay thế cho trạng thái l_i

2.2.3 Định nghĩa của một mạng otomat thời gian (Semantics of a Network of Timed Automata)

$\mathcal{A}_i = (L_i , l^0_i, C, A, E_i, l_i)$ là một Otomat thời gian. Trong đó $\overline{lO} = (l_1^0,\ldots,l_n^0)$ là các vector trạng thái. Ngữ nghĩa được định nghĩa như một hệ chuyển dịch $< S, s_0, \rightarrow >$, trong đó $S = (L_1 \text{ x } \ldots \text{x } L_n) \text{ x } R^C$ là tập các trạng thái, $s_0 = (l_0, u_0)$ là trạng thái ban đầu, và $\rightarrow$ S x S có quan hệ truyền như sau :

- $(\bar{l},u) \rightarrow (\bar{l},u + d)$ với $0 \le d' \le d$,$u + d' \in I(\bar{l})$
- $(\bar{l}, u) \rightarrow (\bar{l}\ [l'_i / l_i], u')$ với $l_i \xrightarrow{g,r} l'_i$ s,t,u $\in$g, u'= [r $\rightarrow$ 0]u và u' $\in I(\bar{l})$
- $(\bar{l}, u) \rightarrow (l\ [l'_j / l_j, l'_i / l_i], u')$ với $l_i \rightarrow l'_i$ và $l_j \xrightarrow{gj,rj} l'_j$ s,t,u $\in$(g_i và g_j), u'=[$r_i \cup r_j \rightarrow 0$] và u' $\in I(\bar{l})$

Một ví dụ cho mạng otomat thời gian:



(a)Đèn điện (b) Người dùng

Hình 6. Ví dụ về đèn điện đơn giản

Hình 6. Cho thấy mô hình một otomat thời gian là một đèn đơn giản.

(Lamp.off,y=0) $\rightarrow$(Lamp.off, y=3) $\rightarrow$(Lamp.low, y=0) $\rightarrow$(Lamp.low, y=3) $\rightarrow$(Lamp.low, y=0) $\rightarrow$(Lamp.low, y=0.5) $\rightarrow$(Lamp.bright, y=0.5) $\rightarrow$(Lamp.bright, y=1000)…..

CHƯƠNG 3:
THỬ NGHIỆM ĐẶC TẢ HỆ THỐNG VỚI CÔNG CỤ UPPAAL

3.1 GIỚI THIỆU

UPPAAL là một môi trường tích hợp cho các mô hình, mô phỏng và kiểm tra hệ thống thời gian thực cùng phát triển bởi cơ bản nghiên cứu khoa học máy tính tại đại học Aalborg ở Đan Mạch và cục công nghệ thông tin tại đại học Uppsala ở Thụy Điển. Đó là thích hợp cho các hệ thống có thể được mô hình như một tập hợp các qui trình phi xác định với cơ cấu kiểm soát hữu hạn và có giá trị thực tế đồng hồ, giao tiếp thông qua các kênh hoặc chia sẻ các biến. Lĩnh vực ứng dụng điển hình bao gồm điều khiển thời gian thực và giao thức truyền thông nói riêng những nơi mà các khía cạnh của thời gian rất quan trọng.

UPPAAL là một môi trường tích hợp cho các mô hình, xác nhận và xác minh của hệ thống thời gian thực mô hình như mạng lưới của automata hẹn giờ, mở rộng với các loại dữ liệu (số nguyên, mảng, vv…).

3.1.1 Phát hành

Việc phát hành chính thức hiện nay UPPAAL 4.0.1 (ngày 11 tháng 2 năm 2010) việc phát hành 4.0 là kết quả của phát triển thêm và nhiều tính năng mới và cải tiến được giới thiệu. Để hỗ trợ các mô hình tạo ra trong phiên bản trước của UPPAAL, phiên bản 4.0 có thể chuyển đổi các mô hình cũ nhất trực tiếp từ các GUI cách khác nó có thể chạy trong chế độ tương thích bằng cách xác định UPPAAL – OLD – SYNTAX biến môi trường từ ngày 26/02/2008. Phân phối một bản chụp phát triển sắp tới UPPAAL 4.2 sự phát triển phiên bản hiện tại là 4.1.2 ảnh chụp được phát hành ngày 11/09/2009.

3.1.2 Cài đặt

Để cài đặt, giải nén zip-file. Điều này sẽ tạo thư mục uppaal-4.0.7 có chứa ít nhất các tập tin uppaal, uppaal.jar, vàthư mục lib, bin-Linux, Win32-bin, lib, và demo. Các bin-thư mục tất cả cần có các verifyta hai tập tin (exe). Và máy chủ (exe). cộng

với một số tập bổ sung, tùy thuộc vào nền tảng. Các thư mục demo-nên chứa một số demo file với xml và hậu tố.. q.

Lưu ý rằng UPPAAL sẽ không chạy mà không có Java 2 cài đặt trên máy chủ hệ thống. Java 2 cho SunOS, Windows95/98/Me/NT/2000/XP, và Linux có thể được download từ http://java.sun.com.

Các phiên bản hiện tại của UPPAAL hiện không có phiên bản hỗ trợ, nó chạy trên môi trường Java (JRE). Bạn cần phải sử dụng phiên bản gần đây nhất sẵn sàng cho nền tảng của bạn. Khả năng tương thích các vấn đề với Windows Vista đã được báo cáo, tuy nhiên nó đang tin rằng những vấn đề là do JRE này. Xin vui lòng kiểm tra xem lại hạn sử dụng JRE mới nhất cho Windows Vista trước khi báo cáo bất kỳ vấn đề với Uppaal chạy trên Windows Vista.

Để chạy trên các hệ thống Linux UPPAAL chạy kịch bản có tên là 'uppaal'. Để chạy trên các hệ thống Windows 95/98/ME/NT/2000/XP nhấn đúp chuột vào tập tin uppaal.jar.

3.1.3 UPPAAL trợ giúp

Công cụ UPPAAL bao gồm ba phần chính:

- Một giao diện người dùng đồ họa (GUI)
- Một máy chủ xác định
- Một công cụ dòng lệnh

Các giao diện được sử dụng cho mô hình, mô phỏng và xác minh. Đối với cả hai mô phỏng và xác minh giao diện sử dụng máy chủ xác minh. Trong mô phỏng này máy chủ được sử dụng để tính toán các trạng thái kế tiếp nhau.

3.1.4 Mô hình hệ thống EDITOR (Biên tập)

Các biên tập viên hệ thống được sử dụng để tạo và chỉnh sửa hệ thống để được phân tích. Một mô tả hệ thống được xác định bởi một tập các qui trình mẫu (có thể với các tờ khai cục bộ) khai báo toàn cục, hệ thống được định nghĩa trong phần này giúp ta mô tả , và nơi để đặt khai báo. làm thế nào để sử dụng khung bên trái của trình biên tập, được gọi là cây chuyển hướng, làm thế nào để vẽ automata với biên tập viên.

Ngôn ngữ mô tả hệ thống được sử dụng trong UPPAAL được mô tả trong các ngôn ngữ tham khảo.

3.1.5 SIMULATOR (Mô phỏng)

Mô phỏng là một công cụ xác nhận cho phép kiểm tra việc hành quyết có thể năng động của một hệ thống trong thiết kế ban đầu (hoặc mô hình) giai đoạn. Trong ý nghĩa này, nó cung cấp có nghĩa là không tốn kém phát hiện lỗi trước khi kiểm tra xác minh của mô hình, giả lập này cũng được sử dụng để hình dung, hành quyết (tức là dấu vết biểu tượng).

3.1.6 VERIFIER (Xác minh)

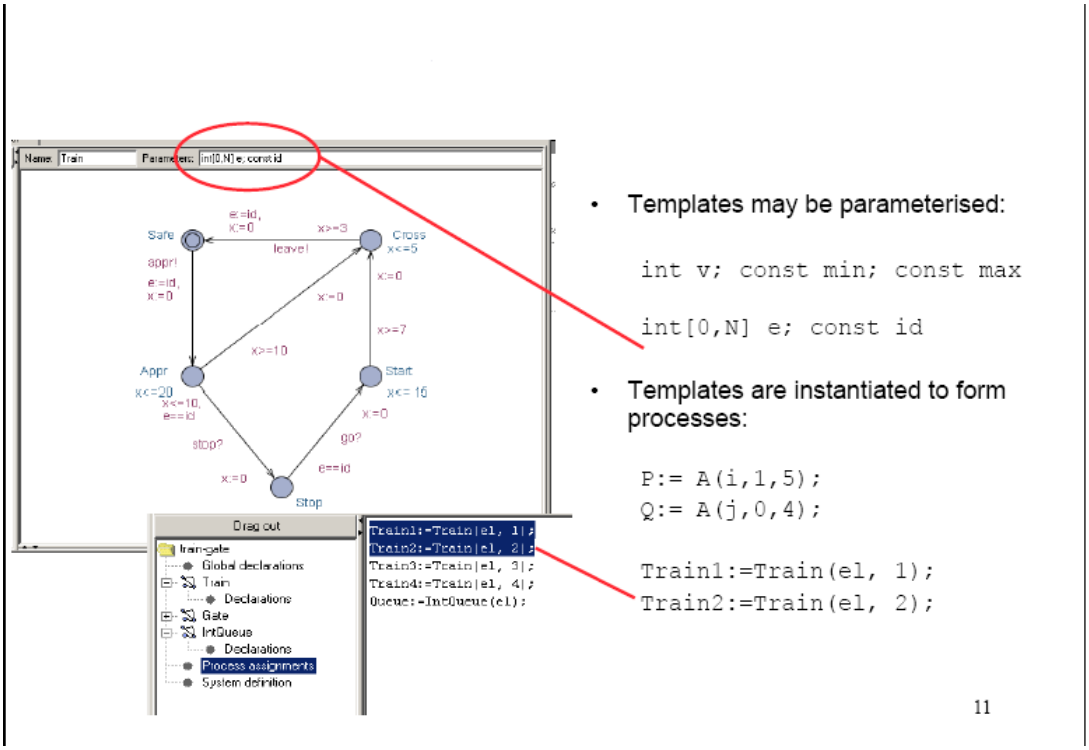
Xác minh là để kiểm tra xem hệ thống có an toàn hay không. Trong phần này giúp chúng ta mô tả làm thế nào để sử dụng các công cụ, làm thế nào để chỉ định và xác minh yêu cầu. Các yêu cầu đặc điểm kỹ thuật ngôn ngữ và xác minh lựa chọn được mô tả trong các phần riêng biệt.

3.1.7 Mô tả hệ thống

Một mô hình hệ thống trong UPPAAL bao gồm một mạng lưới các qui trình mô tả như là mở rộng theo thời gian ***automata*** , các mô tả của mô hình bao gồm ba phần: khai báo cục bộ và toàn cục, các mẫu otomat và định nghĩa hệ thống .

3.1.8 TEMPLATES (Các mẫu)

UPPAAL cung cấp một ngôn ngữ phong phú để xác định các mẫu trong các hình thức mở rộng theo thời gian. Trái ngược với otomat cổ điển, otomat trong UPPAAL có thể sử dụng một ngôn ngữ biểu thức để kiểm tra và cập nhập đồng hồ, các biến, các loại hồ sơ, gọi người dùng định nghĩa chức năng năng v.v….Các mẫu otomat bao gồm: địa điểm và các cạnh. Một mẫu cũng có thể có khai báo các biến cục bộ và biến toàn cục.



Hình 6. Ví dụ về mẫu

3.1.9 Tham số

Các mẫu và chức năng được ***parameterised***. Cú pháp cho các tham số được xác định bởi ngữ pháp cho tham số:

Tham số :: = [tham số (',' Các thông số) *]

Tham số :: = loại [' & '] ID ArrayDecl*

Danh sách tham số không được kết thúc bởi dấu ";"

Chú ý: thông số mảng phải được bắt đầu = 1 dấu "&" được thông qua tham chiếu.

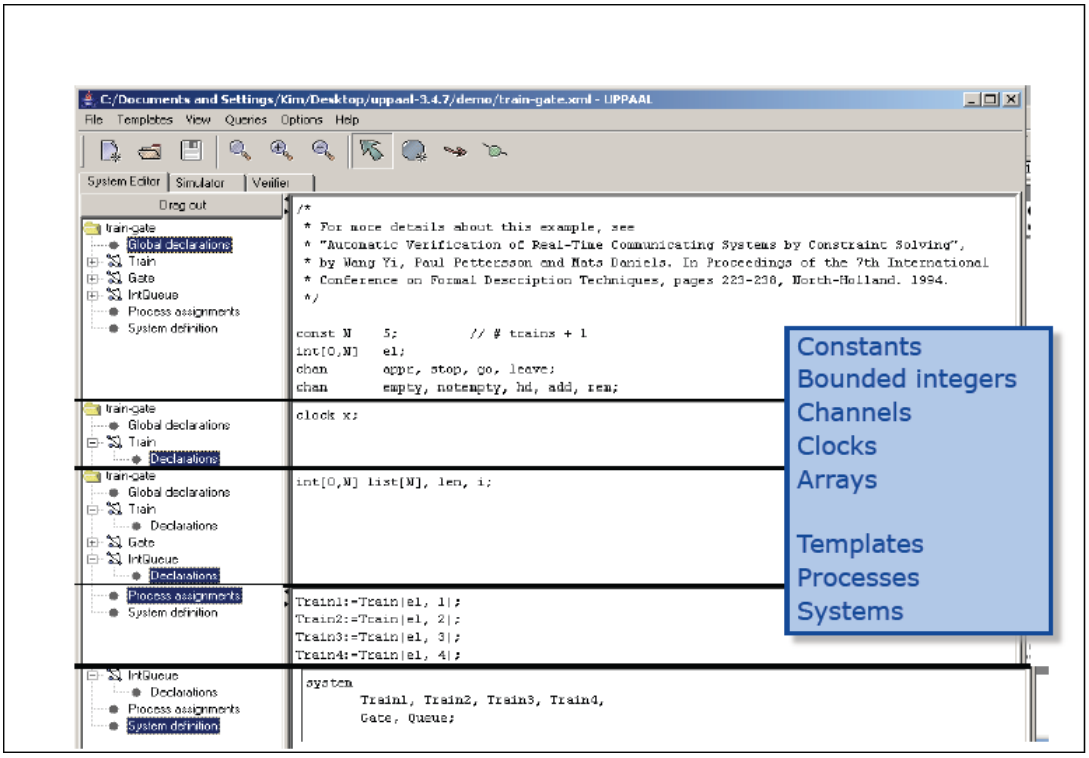
Các ví dụ:

P (đồng hồ &x, bool bit)

Q (đồng hồ & x, đồng hồ &y, int i1$i2, chan chan $a, $b)

Qui trình mẫu Q có 6 tham số: 2 đồng hồ, 2 biến nguyên (với phạm vi mặc định) và 2 kênh

- Khai báo



Hình 7. Ví dụ về khai báo

Khai báo (cho một mẫu) và có thể chứa các khai báo của đồng hồ, số nguyên, các kênh, mảng, hồ sơ, và các loại. Cú pháp là mô tả bằng ngữ pháp cho các khai báo:

khai báo:: = (VariableDecl | TypeDecl | Chức năng | ChanPriority) *

VariableDecl:: = Loại VariableID (',' VariableID) * ';'

VariableID:: = ID ArrayDecl * ['=' Initialiser]

Initialiser:: = Expression

| "('Initialiser (', 'Initialiser) *') '

TypeDecls:: = 'typedef' ID * Loại ArrayDecl (',' ID ArrayDecl *) * ';'

Các ví dụ

const int a = 1;
bool b [8], c [4];

- int a [2] [3] = ((1, 2, 3), (4, 5, 6));
- chan d; // một kênh d

- chan e; // một kênh e

- struct (int a; bool b;) s1 = (2, true);
- meta int

;

int a;

int b;

- Khai báo kiểu

Các từ khoá typedef được sử dụng để đặt tên các kiểu

Ví dụ

Khai báo S ghi có chứa một số nguyên a, b *boolean* một và đồng hồ một c:

typedef struct

(

int a;

bool b;

đồng hồ c;

) S;

3.1.10 Các dòng lệnh (Command Line)

UPPAAL có thể được thực hiện từ dòng lệnh bằng cách sử dụng các lệnh sau đây trên unix :

uppaal [OPTION] ... [FILENAME]

Trên cửa sổ, các lệnh sau đây có thể được sử dụng (ví dụ, bằng cách sử dụng "Run "trong Start Menu) :

java- jar \ đường dẫn \ uppaal.jar [OPTION] ... [FILENAME]

đây là đường dẫn đầy đủ đến uppaal.jar file (nó cũng cần thiết để xác định đường dẫn đầy đủ đến java thực thi).

Các tùy chọn tên tập tin đề cập đến một mô hình được nạp lúc khởi động.

Các tùy chọn dòng lệnh có sẵn là:

- antialias ngày | off

(Mặc định trên)

engineName <filename>

Tên của máy chủ xác minh (mặc định là máy chủ trên Unix và server.exe trên Windows) sẽ được sử dụng bởi các GUI.

enginePath <path>

Đường dẫn đến máy chủ xác minh (ví dụ như bin - Win32) để được sử dụng bởi các GUI.

3.1.11 Help

Hiển thị một bản tóm tắt các tùy chọn.

serverHost <name>

serverPort <no>

splashScreen ngày | off

Vô hiệu hóa hoặc cho phép màn hình giật.

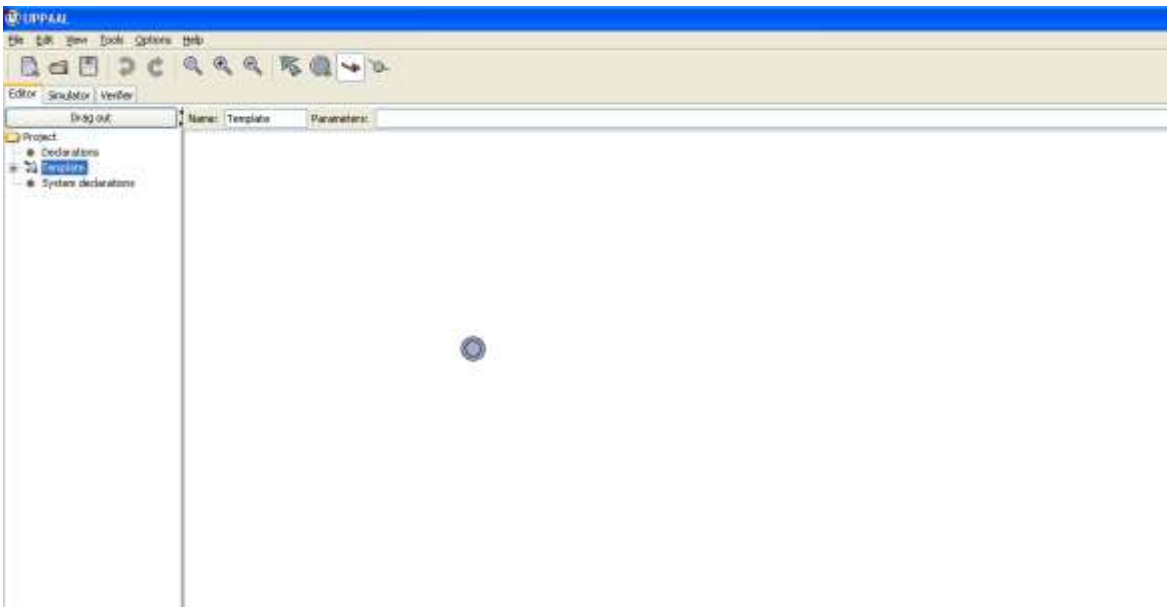
exportToEPS templateName

Đưa ra mẫu để đặt tên EPS .

psColors ngày | off

3.1.12 Vẽ

Bên phải của hệ thống trình soạn thảo được sử dụng để vẽ ***automata***. Hiện tại có bốn công cụ vẽ có tên Chọn, Nơi, Biên, và Đỉnh đại diện bởi các nút trong thanh công cụ .



Hình 8. Các công cụ để vẽ automata

Chọn công cụ được sử dụng để chọn , di chuyển, chỉnh sửa và xóa. Các yếu tố có thể được chọn bằng cách nhấp vào chúng, các yếu tố có thể được thêm hoặc loại bỏ khỏi vùng lựa chọn bằng cách giữ phím điều khiển, việc lựa chọn có thể được di chuyển bằng cách kéo chuột. Nó có thể thay đổi nguồn và đích của một cạnh bằng cách di chuyển con chuột để bắt đầu hoặc kết thúc của một cạnh cho đến khi một vòng tròn nhỏ xuất hiện. Kéo vòng tròn này đến một vị trí mới để thay đổi mã nguồn hoặc mục tiêu của các mép .

- Add location: được sử dụng để thêm vị trí mới. Đơn giản chỉ cần bấm vào với nút chuột trái để thêm một vị trí mới.
- Add edge: công cụ vẽ cạnh được sử dụng để thêm cạnh mới giữa các vị trí. Bắt đầu cạnh bằng cách nhấp vào vị trí đầu.
- Add nail : Công cụ được sử dụng để gắn nhãn mới cho một cạnh. Đơn giản chỉ cần click và kéo bất cứ nơi nào trên một cạnh để thêm vào và đặt một nhãn mới.

Đối với người dùng với ba nút chuột , nút chuột giữa có thể được dùng để tạo ra mới. trình soạn thảo sẽ tự động chọn công cụ chính xác: Nhấp chuột vào một chỗ trống tạo ra một vị trí mới , cách bấm vào một vị trí tạo ra một cạnh mới.

3.1.13 Thanh công cụ (Tool Bar)

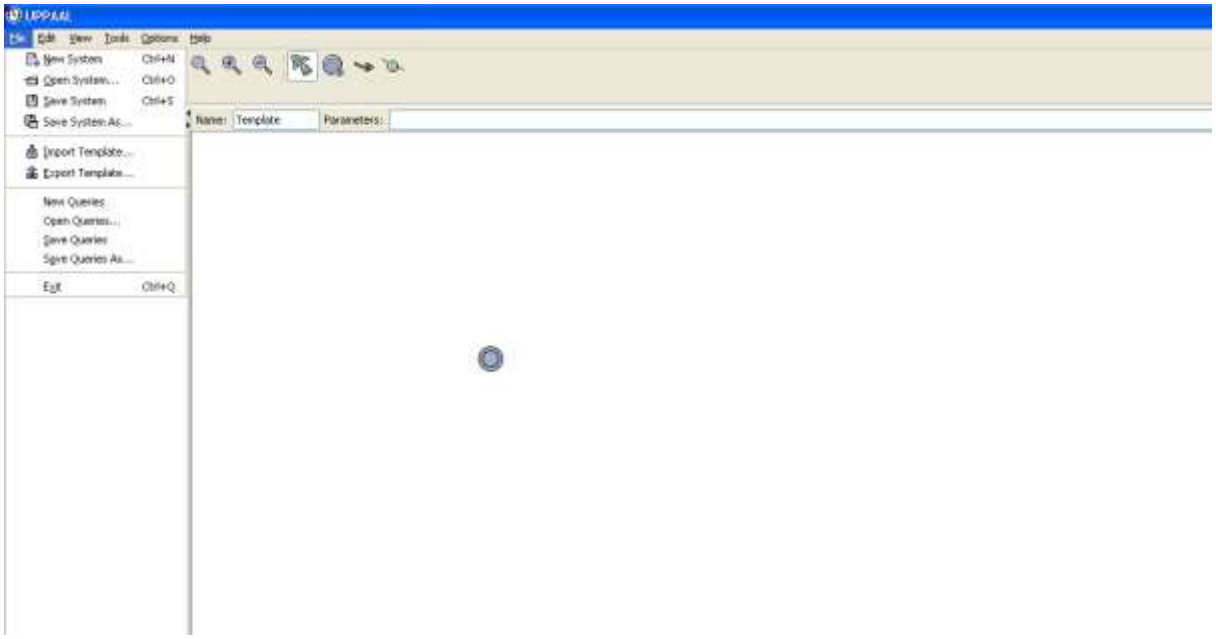
Thanh công cụ là bình thường nằm ngay dưới thanh trình đơn nhưng có thể được di chuyển (bằng cách sử dụng con chuột) đến vị trí khác hoặc thậm chí đến một cửa sổ riêng biệt. Thanh công cụ được chia thành ba nhóm. Hai nhóm tận cùng bên trái cung cấp truy cập nhanh vào một số các trình đơn được sử dụng các mục thường xuyên nhất. Nhóm bên phải chứa các công cụ chỉnh sửa.

Nhóm đầu tiên chứa các nút sau đây: *New*, *Open Project*, và *Save*. Những tính năng này được mô tả trong menu File phần.

Nhóm thứ hai chứa các nút sau đây: *Phóng to để Fit, Zoom In,* và *Zoom Out.* Tính năng này được mô tả trong phần menu View .

Nhóm thứ ba bao gồm các công cụ được sử dụng trong trình soạn thảo để chọn và di chuyển các yếu tố của một ***automaton***, và để thêm địa điểm, các cạnh và nhãn. Tính năng này được mô tả trong phần vẽ trên .

3.1.13.1 Menu File

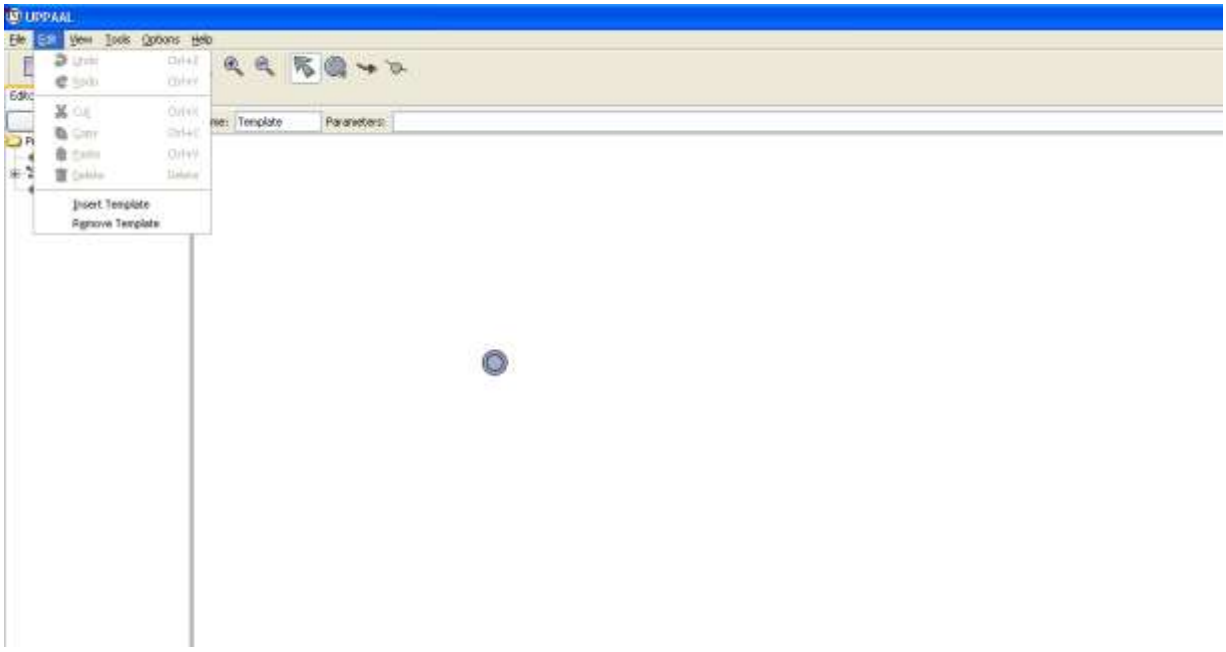


Hình 9. Các thành phần menu file

Trình đơn tận cùng bên trái của thanh menu là menu file. Nó chủ yếu được sử dụng để mở và lưu lại (một phần của) mô tả hệ thống hay chi tiết kỹ thuật yêu cầu được tạo ra trong UPPAAL. Các mục có sẵn là:

- New System (hệ thống mới): tạo một hệ thống mới
- **Open System (mở hệ thống):** tải một hệ thống đã có từ tập tin. Các yêu cầu đặc điểm kỹ thuật tương ứng (nghĩa là cùng một tên tập tin, nhưng với các hậu tố. Q) được nạp vào xác minh, nếu nó tồn tại.
- **Save System (lưu hệ thống):** hệ thống lưu trong trình soạn thảo cho tập tin.
- **Save System As (lưu hệ thống như):** lưu hệ thống trong trình soạn thảo một tập tin chỉ định.
- **Import Template:**Một cửa sổ hộp thoại sẽ được hiển thị cho phép một tập hợp các sẵn các mẫu để được nhập khẩu.
- **Export Template:** Hiện tại mẫu ở định dạng tập tin Encapsulated Postscript.
- **New Queries:** đặc tả yêu cầu biên tập với một tập tin trống.
- **Open Queries (mở câu hỏi):** tải một bộ hiện có của các chi tiết kỹ thuật yêu cầu từ tập tin.
- **Save Queries(lưu câu hỏi):** để lưu thông số kỹ thuật yêu cầu trong biên tập các tập tin.
- **Save Queries As(lưu truy vấn theo):** chi tiết kỹ thuật yêu cầu trong trình soạn thảo để một tên file được chỉ định.
- **Exit (thoát**): UPPAAL.

3.1.13.2 Menu Edit

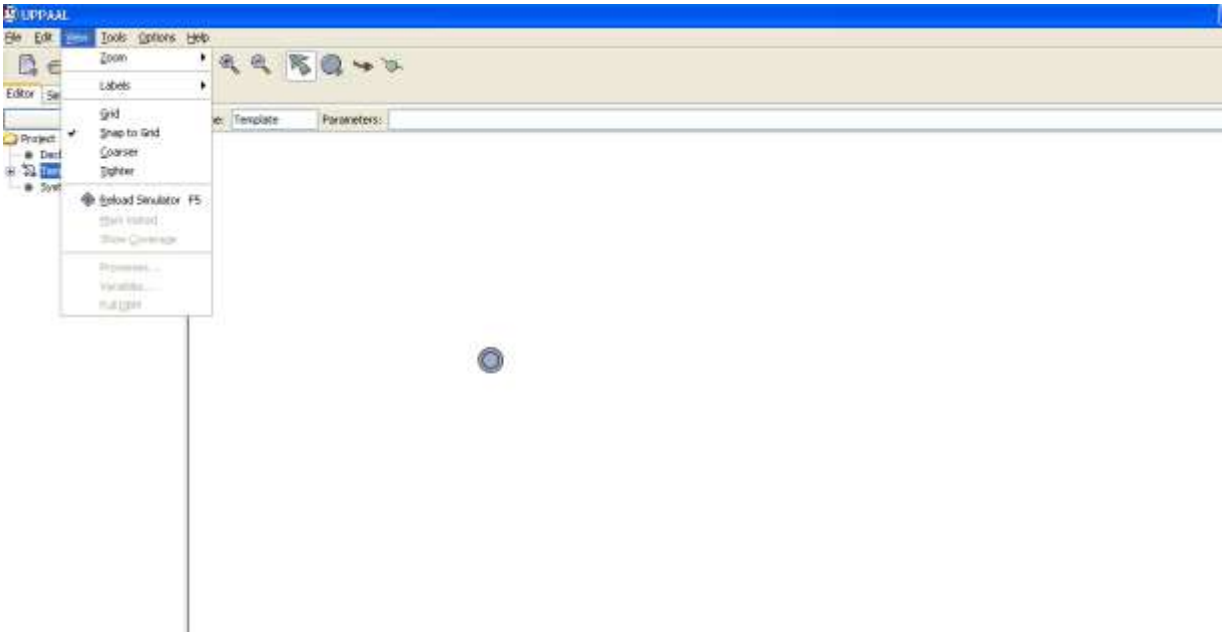


Hình 10. Các thành phần menu edit

Các menu Edit cung cấp một tập hợp các lệnh được hỗ trợ trong trình soạn thảo hệ thống. Các mặt hàng là:

- Undo
- Redo
- Cut
- Copy
- Paste
- Delete
- Insert Template
- Remove Template

3.1.13.3 Menu View

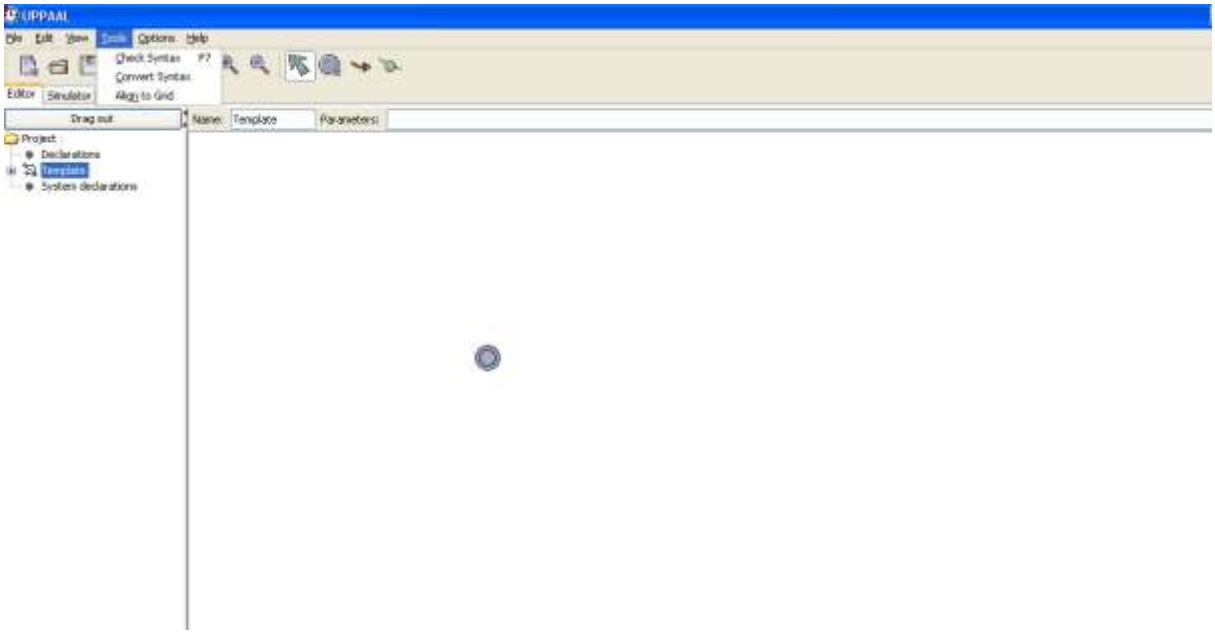


Hình 11. Các thành phần menu view

- Trình đơn xem được sử dụng để sửa đổi sự xuất hiện của hệ thống hiện đang được hiển thị trong hệ thống biên tập.
- **Zoom:** hiển thị một menu phụ với zoom giá trị cố định, zoom để phù hợp với phóng to trong, trên, hoặc để kích thước bình thường. Một sự thay đổi trong giá trị zoom ảnh hưởng đến các mẫu trình biên tập hoặc các quá trình trong giả lập (nếu một trong những công cụ được kích hoạt).
- **Labels:**hiển thị một menu phụ mà từ đó ta có thể chọn những loại nhãn sẽ được hiển thị trong vùng vẽ. Ngay cả khi ẩn, tất cả các nhãn có thể được nhìn thấy trong ***tooltip*** của địa điểm và góc cạnh.
- **Show Grid:** lưới vẽ được hiển thị khi mặt hàng này được kiểm tra.
- **Snap to Grid:** làm cho các đối tượng bản vẽ mới (như địa điểm, nhãn) sắp xếp cho lưới điện ***snap***. Kích thước của lưới ***snap*** có liên quan đến kích thước của lưới vẽ.
- **Lưu ý:** các tùy chọn ***Grid Snap*** *để* có thể sử dụng ngay cả khi lưới bản vẽ không được hiển thị.
- **Reload Simulator:** Tác phẩm của hệ thống hiện đang được nạp vào trình soạn thảo, để giả lập và xác minh các.

- **Processes:** hiển thị một cửa sổ hộp thoại cho ẩn và hiện các quy trình trong bảng điều khiển quá trình của mô phỏng này.
- **Variables:** hiển thị một cửa sổ hộp thoại cho ẩn và hiển thị các biến trong bảng các biến của mô phỏng này.
- **Full DBM:** hiển thị tất cả các hạn chế trên đồng hồ trong bảng các biến của mô phỏng này. Nếu không được chọn một số lượng tối thiểu các khó khăn sẽ được hiển thị.

3.1.13.4Menu Tool (công cụ)

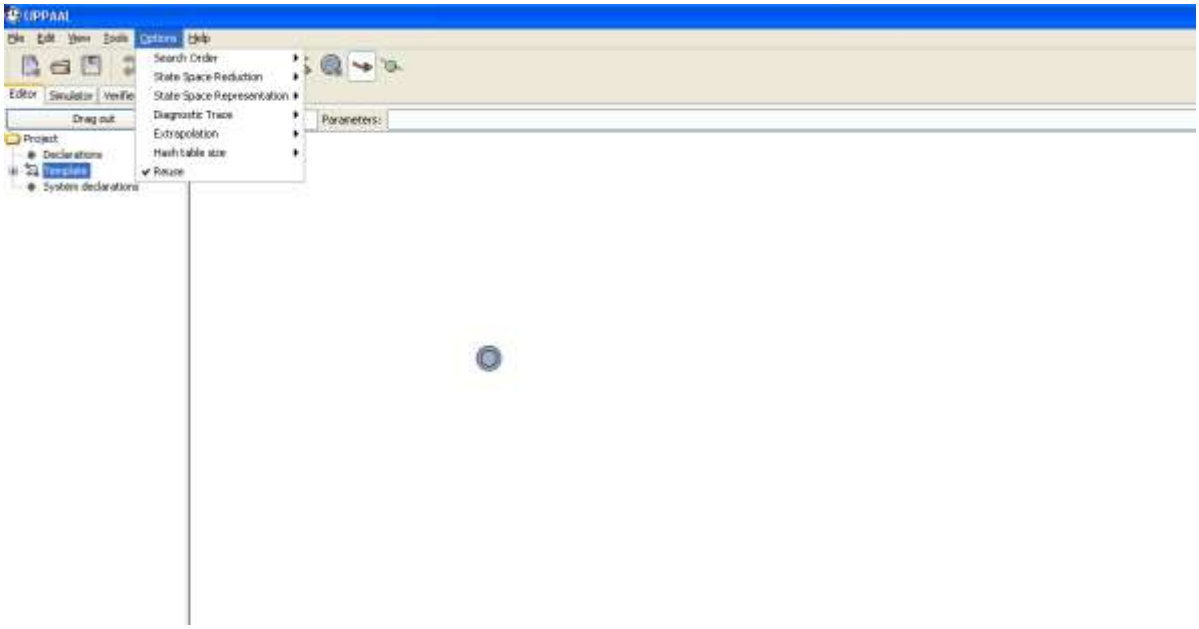


Hình 12. Các thành phần menu tool

Trình đơn các công cụ chứa một bộ công cụ hữu ích trong hệ thống edtior . Các mặt hàng là:

- Check syntax (Kiểm tra cú pháp): kiểm tra nếu tìm thấy bất kỳ lỗi nào đều được cảnh báo và liệt kê ở phần dưới của khu vực vẽ của hệ thống biên tập.
- Convert syntax (Chuyển đổi cú pháp): hỗ trợ trong một hệ thống phù hợp với các cú pháp được sử dụng trong UPPAAL 3.4 đến cú pháp hiện hành.
- Align to Grid: làm cho tất cả các đối tượng hiện có của mẫu hiện tại hiển thị trên lưới.

3.1.13.5 Menu option

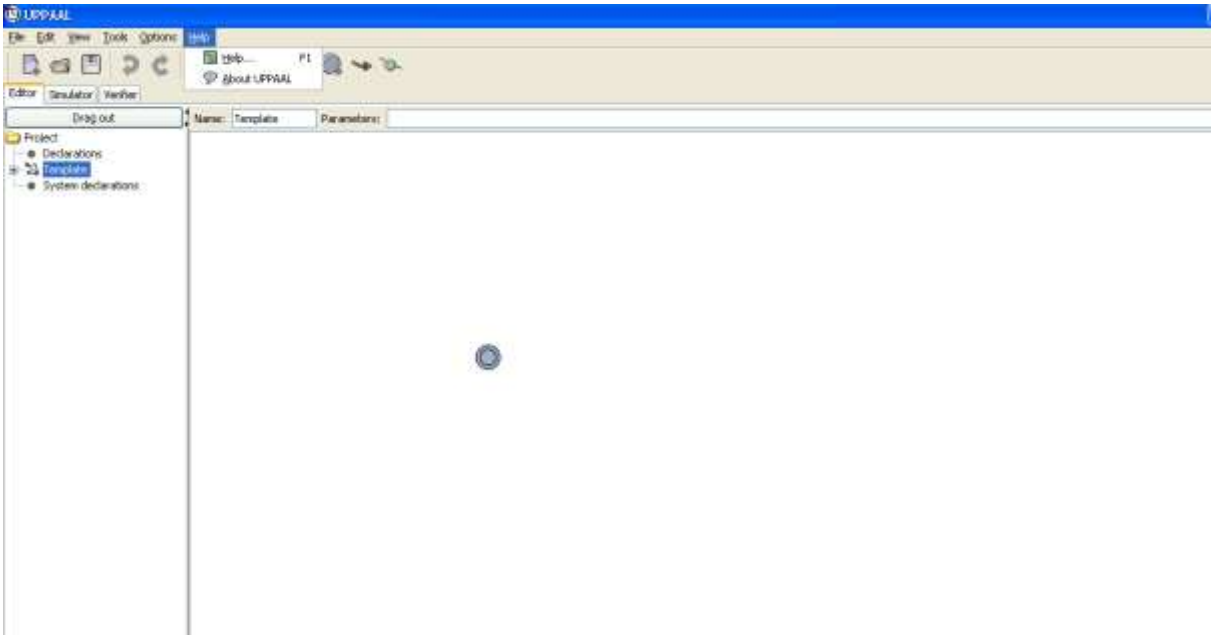


Hình 13. Các thành phần menu option

Các menu tùy chọn các thiết lập để kiểm soát kiểm tra mô hình**.** Bao gồm:

- Search Order
- State Space Reduction
- State Space Representation
- Diagnostic Trace
- Extrapolation
- Hash table size
- Reuse

3.1.13.6 Menu help



Hình 14. Các thành phần menu help

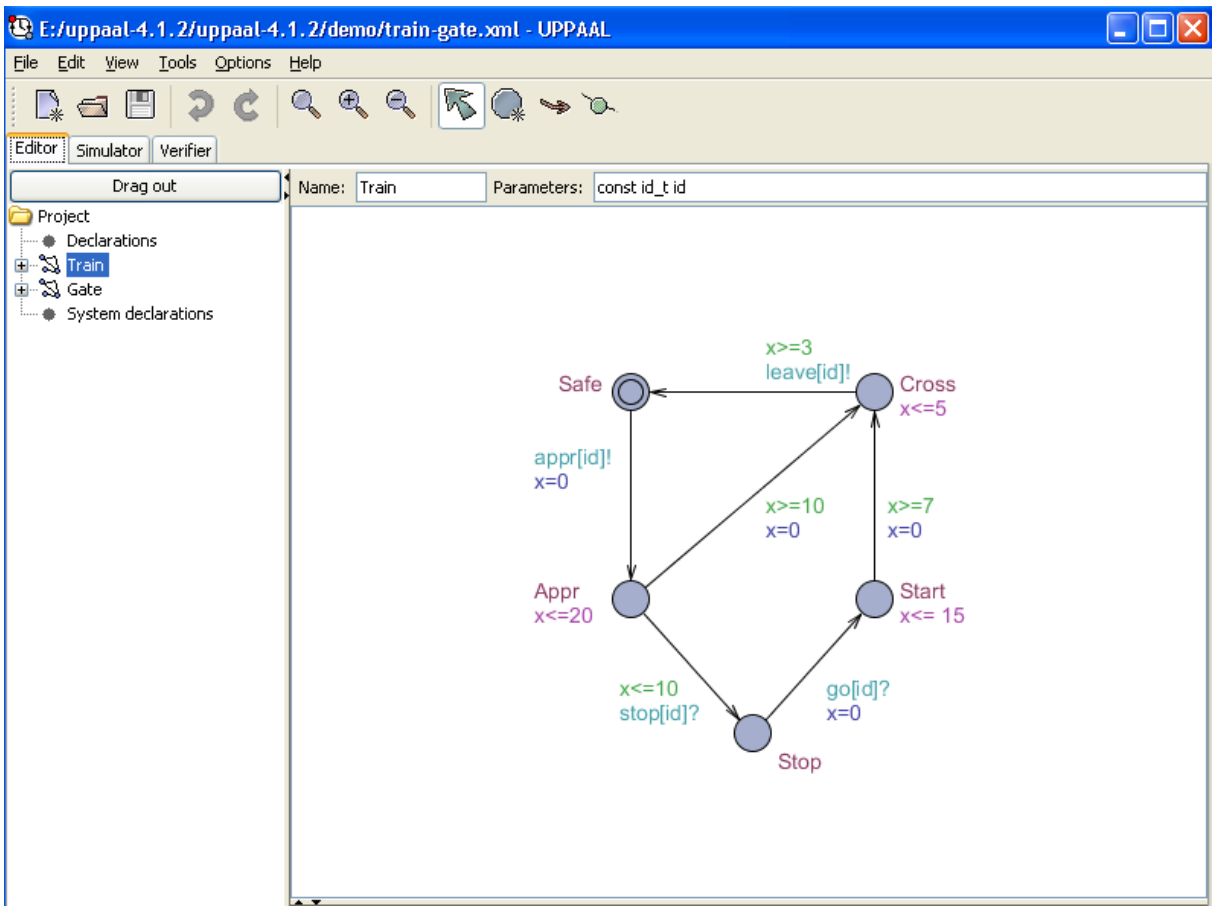
Thực đơn giúp có hai mục: trợ giúp mở ra một cửa sổ riêng biệt hiển thị các trang trợ giúp, và về mà mở ra một cửa sổ hiển thị số phiên bản và thông tin về bản quyền của UPPAAL. Bao gồm:

Help … F1

About Uppaal

3.2 VÍ DỤ ĐẶC TẢ MÔ HÌNH TRAIN – GATE BẰNG UPPAAL

Ý tưởng đằng sau công cụ này là để làm mô hình một hệ thống với timed automata sử dụng biên tập đồ họa, dựa theo đó để làm cho nó hoạt động như định trước, và cuối cùng để xác minh rằng nó đúng với một tập các đặc tính. Giao diện đồ họa của Java đa uuClient phản ánh ý tưởng này và được chia thành 3 phần chính: biên tập, mô phỏng và sự xác minh, có thể truy cập thông qua 3 tab.



Hình 15.Ví dụ về Train-Automata của Train-Gate.

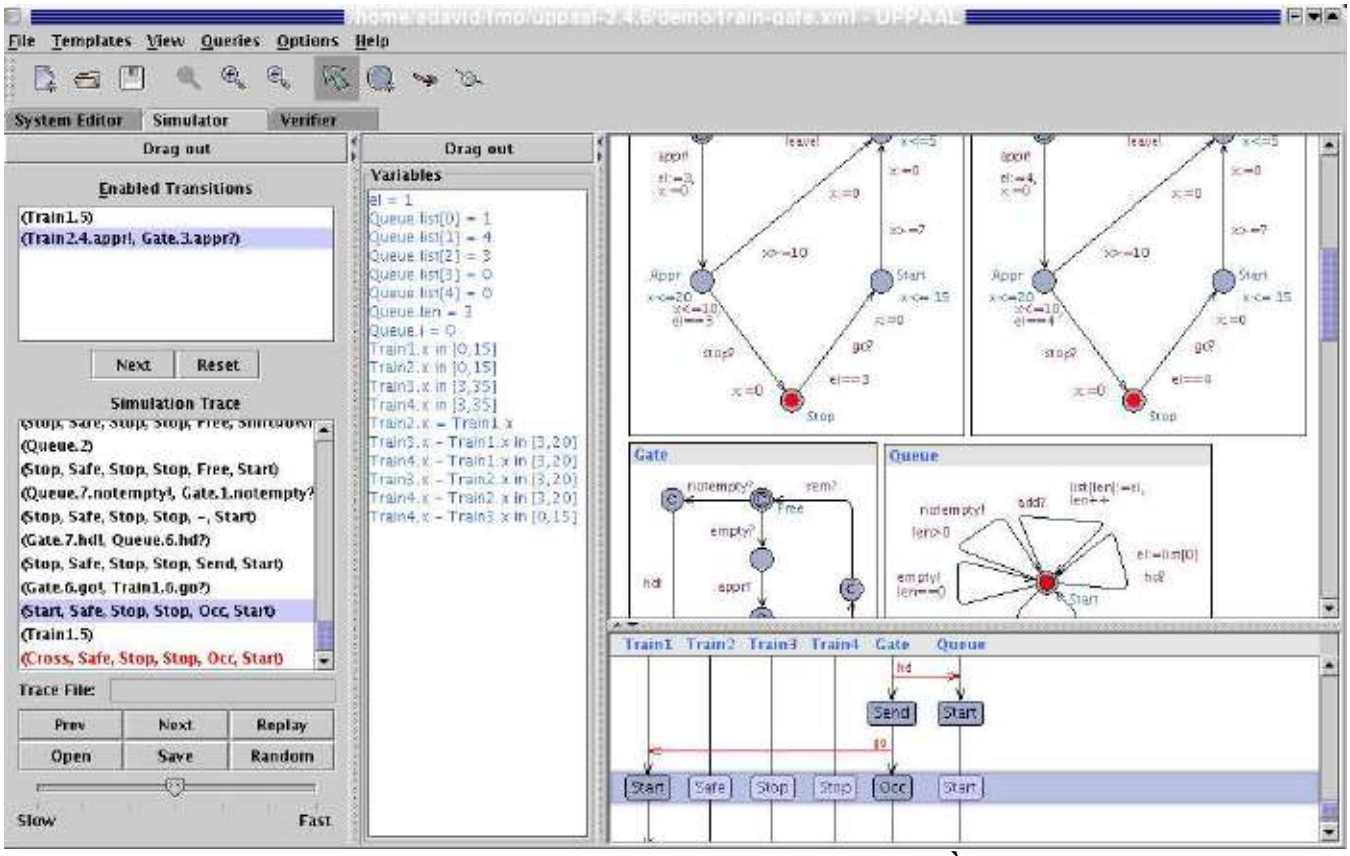
Nút Select đã được kích hoạt trên thanh công cụ. Trong chế độ này người ta có thể di chuyển địa điểm và các cạnh, hoặc chỉnh sửa các nhãn (label). Các chế độ khác để thêm vào các địa điểm, các cạnh, và đỉnh của các cạnh (được gọi là nail). Vị trí mới sẽ không có tên mặc định. 2 trường văn bản cho phép người dùng đặt tên mẫu và các tham số của nó. Mẹo nhỏ: nút chuột giữa là một cách tắt để thêm các yếu tố mới, tức là nhấn nó trên các miền, một vị trí hoặc cạnh cho biết thêm một vị trí mới, cạnh, hoặc nail tương ứng.

Trình biên tập (The Editor): Một hệ thống được định nghĩa như là một mạng của timed-automata , được gọi là tiến trình của công cụ, được xếp song song. Một tiến trình được hiển thị trong thực tế từ một tham số mẫu. Trình biên tập được chia thành 2 phần: một khung cây để truy cập vào các mẫu khác và sự công bố, và một miền vẽ / soạn thảo văn bản. Hình 15 cho thấy trình biên tập cùng với ví dụ về Train-Gate ở trong phần 4. Các vị trí đã được đánh tên và không thay đổi, còn các cạnh thì được đánh tên với một điều kiện an toàn, đồng bộ, và sự phân công.

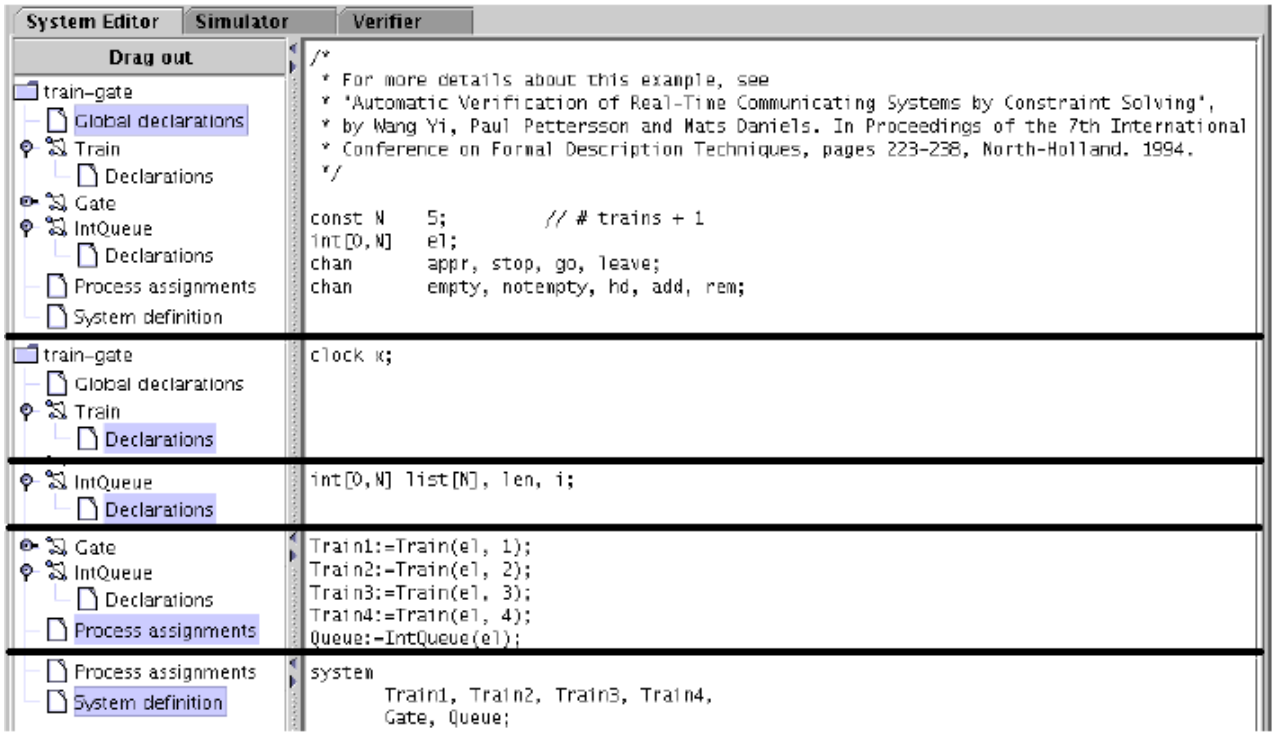
Cây ở phía bên trái cho phép truy cập đến các bộ phận hệ khác của hệ thống mô tả như sau:

Global declaration (sự khai báo toàn cục): chứa các biến toàn cục kiểu interger (số nguyên), đồng hồ, các kênh đồng bộ, và các hằng số.

Templates (Các mẫu): Train, Gate và IntQueue là các tham số khác nhau của timed-automata. Một mẫu có thể có các biến cục bộ, các kênh và các hằng số.



Hình 16. Khung nhìn của tab mô phỏng cho ví dụ về Train-Gate.



Hình 17. Sự khác nhau giữa khai báo toàn cục và khai báo cục bộ trong ví dụ về Train-Gate.

Chúng ta đưa một vài ảnh màn hình (screenshoot) của công cụ để thể hiện sự khai báo một cách xúc tích.

Process Assignments (Các tiến trình được phân công): Mẫu được hiển thị vào các quá trình. Phần phân công tiến trình chứa sự khai báo trong những trường hợp này.

System definition (Định nghĩ hệ thống): Một danh sách các tiến trình của hệ thống.

Cú pháp được sử dụng trong các nhãn và các khai báo là trong phần trợ giúp hệ thống của công cụ. Sự khai báo cục bộ và khai báo toàn cục được thể hiện trong Hình 17 Cú pháp đồ họa được trực tiếp đưa vào từ những mô tả của timed-automata trong phần 2.

The Simulator (Sự mô phỏng): Sự mô phỏng có thể được dùng theo 3 cách: Người dùng có thể chạy hệ thống bằng tay (tức là không tự động) và chọn kiểu chuyển đổi để thi hành. Chế độ ngẫu nhiên có thể được bật để cho hệ thống chạy riêng, hoặc người dùng có thể làm theo sự chỉ dẫn (lưu lại hoặc nhập từ các xác minh) để biết làm

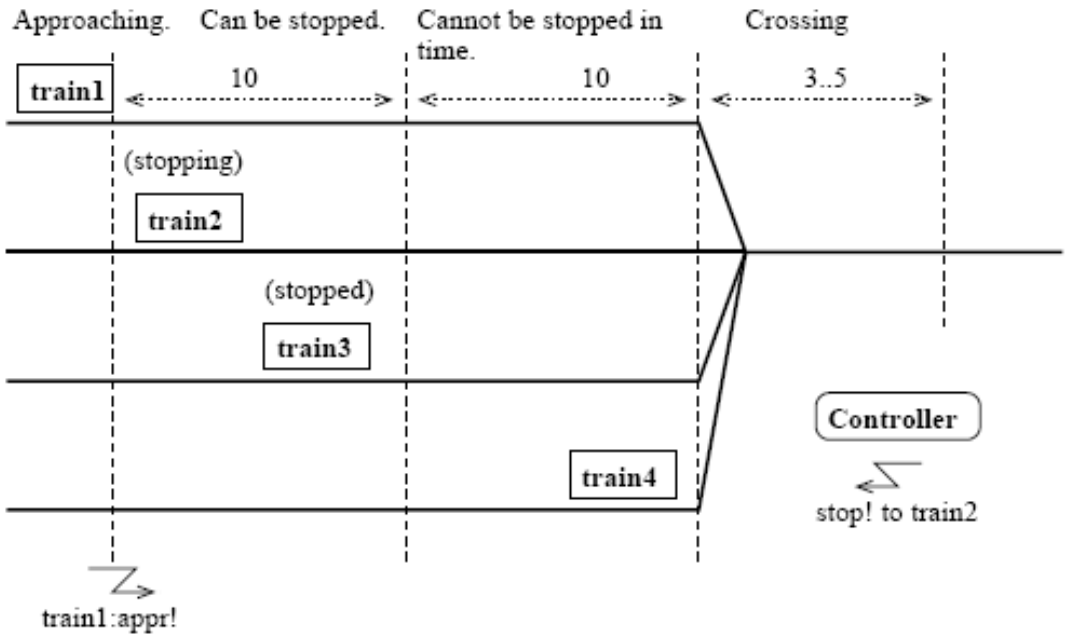
thế nào để một chế độ nào đó có thể truy cập được. Hình 10 thể hiện sự mô phỏng. Nó được chia thành 4 phần:

The control part (Phần điều khiển): được dùng để chọn và kích hoạt sự chuyển đổi, làm theo chỉ dẫn, và bật chế độ mô phỏng ngẫn nhiên.

The variable view: thể hiện giá trị của các biến kiểu integer (số nguyên) và đồng hồ. UPPAAL không thể hiện chế độ cụ thể với các giá trị thực tế cho đồng hồ. Bởi có vô vàn những chế độ như thế nên thay vào đó, UAPPAAL đã hiển thị một tập các chế độ cụ thể được biết đến như là các chế độ tượng trưng. Tất cả các chế độ cụ thể ở trong chế độ tượng trưng này chia sẻ cùng một vị trí vector và cùng một giá trị cho các biến riêng rẽ. Các giá trị khả thi của đồng hồ được mô tả bởi một tập các rằng buộc. Đồng hồ được công nhận trong chế độ tượng trưng chính xác là thỏa mãn tất cả các rằng buộc (điều kiện).

3.2.1 Mô tả Train Gate

Ví dụ về Train Gate được mô tả trong Uppaal: hệ thống điều khiển tàu hỏa trong việc truy cập và giao cắt giữa các tàu. Hệ thống định nghĩa như một số tàu , trong ví dụ có 4 tàu và 1 bộ điều khiển. Một tàu không thể dừng tức thời và khởi động lại trong cùng một thời gian. Do đó có một số ràng buộc về thời gian trên tàu trước khi vào cầu, 1 tàu sẽ gửi một tín hiệu appr! Sau đó nó cứ có 10 đơn vị thời gian để nhận 1 tín hiệu dừng điều này cho phép nó dừng một cách an toàn trước khi vào cầu. Sau 10 đơn vị thời gian nó cần nhiều hơn 10 đơn vị thời gian để đi đến cầu nếu nó không muốn dừng. Nếu 1 tàu bị dừng lại nó sẽ lấy lại tiến trình của nó khi bộ điều khiển gửi 1 tín hiệu go cho nó sau khi tàu đằng trước rời khỏi cầu và đã gửi 1 tín hiệu leave.

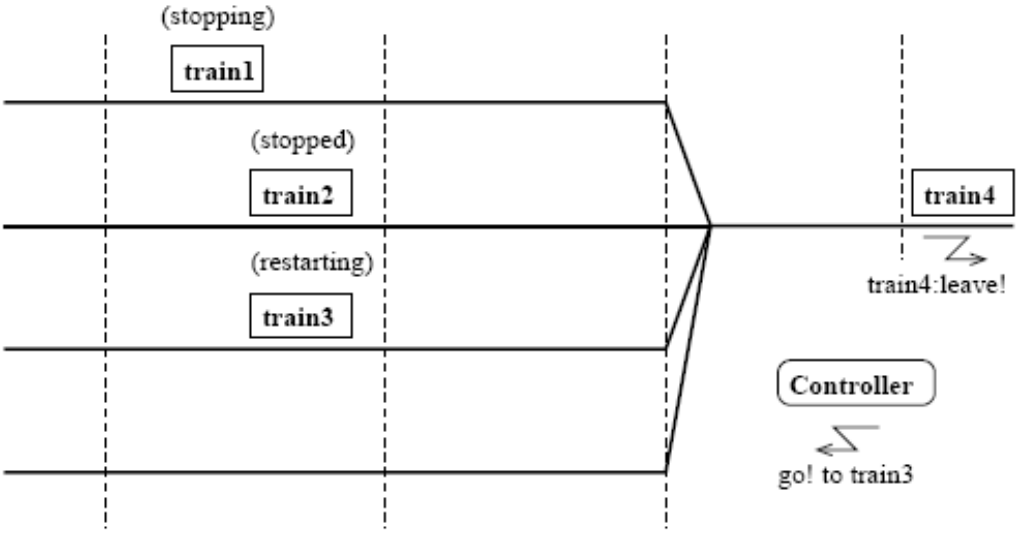


Hình 18. Ví dụ Train - Gate

3.2.2 Mô hình trong Upaal

Mô hình Train Gate có 3 mẫu: Train, Gate, Int Queue.

- Mẫu Train: có 5 trạng thái Safe, appr, stop, start, cross. Trạng thái khởi tạo là safe nó thể hiện là tàu vẫn chưa tới cầu. Trạng thái này không có các ràng buộc bất biến nào nên tàu có thể dừng ở trạng thái này bất kì vào lúc thời gian nào. Khi 1 tàu đến phải đồng bộ với bộ điều khiển. Công việc này được thực hiện bởi kênh đồng bộ appr. Bộ điều khiển có một tín hiệu appr biến đồng hồ x được reset và các biến tham số e được thiết lập định danh cho tàu, biến này được sử dụng bởi 1 hàng đợi queue và bộ điều khiển sẽ biết tàu nào được tiếp tục, tàu nào sẽ dừng…vị trí ràng buộc bất biến x<=20 có hiệu quả là trạng thái này phải được dừng trong vòng 20 đơn vị thời gian, 2 dịch chuyển ra ngoài được chặn bởi ràng buộc x<=10 hoặc x>=10.



Hình 19. Mẫu Train.

Hai ràng buộc trả về 2 vùng trước khi vào cầu có thể bị dừng hoặc có thể không bị dừng. Tại thời điểm chính xác là 10 cả dịch chuyển này đều được bật nó cho phép chúng ta chọn các điều khiển bất kì tình huống nào nếu như chỉ có 1 tàu. Nếu như tàu không được dừng x<10 thì dịch chuyển tới trạng thái stop được thực hiện ngược lại thì tàu sẽ đi đến trạng thái cross. Khi dịch chuyển đến trạng thái stop cũng được chặn bởi điều kiện e==id và được đồng bộ với kênh stop? Khi bộ điều khiển quyết định dừng 1 tàu nó sẽ quyết định tàu nào và đồng bộ với stop!

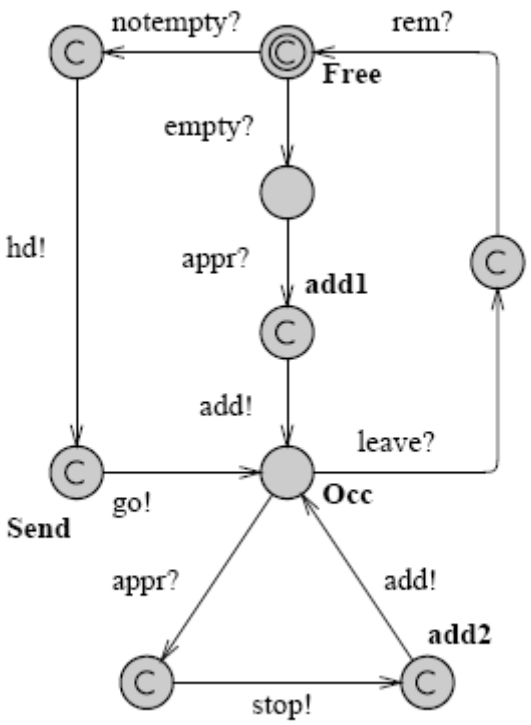
Ở trạng thái stop không có ràng buộc bất biến: 1tàu có thể dừng vô thời hạn đồng bộ go? chặn e==id đảm bảo rằng đúng tàu được khởi động lại. Các mô hình hóa ở đây, chúng ta giả sử rằng tàu có thể nhận go? đồng bộ thậm chí khi nó không dừng hoàn toàn điều này sẽ đưa ra một sự khởi động không đơn định.

Trạng thái start có ràng buộc bất biến x<=15 và dịch chuyển đi ra có ràng buộc x>=7. Điều này tàu được khởi động lại và tiến vào vùng giao cắt trong khoảng thời gian 7<x<15 đơn vị thời gian.

Trạng thái cross tương tự như start theo cách thức nó sẽ mất 3 đến 5 đơn vị thời gian sau khi vào.

- Mẫu Gate: Bộ điều khiển Gate trong hình đồng bộ với quêu và các tàu. Một số trạng thái của nó không có tên điển hình là các trạng thái trung chuyển an toàn được đánh giá là c. Bộ điều khiển được bắt đầu ở Freelaf cầu được rỗi ở đây kiểm tra hàng

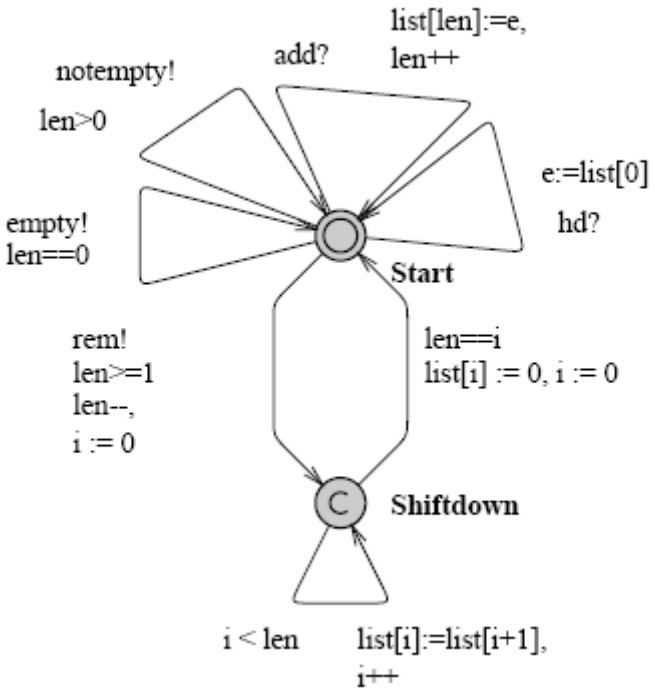
đợi xem rỗng hay không. Nếu hàng đợi rỗng bộ điều khiển sẽ đợi 1 tàu tiến tới (trạng thái tiếp theo) với một đồng bộ appr? Khi một tàu tiến tới nó được thêm vào trong hàng đơi với đông bộ add. Nếu hàng đợi không rỗng thì tàu đầu tiên trên hàng đợi (được đọc bởi hd!) được khởi động lại với đồng bộ go!



Hình 20. Mẫu Gate

Trạng thái Occ bộ điều khiển đợi tàu đang chạy rời khỏi cầu (leave) nếu như có tàu khác đang tới (appr?) chúng sẽ bị dừng (stop?) và được thêm vào hàng đợi (add!) khi mà một tàu rời khỏi tàu bộ điều khiển sẽ bỏ nó khỏi hàng đợi với đồng bộ rem?

- Mẫu hàng đợi Queue: Queue trong hình 15 có 1 trạng thái start khi mà nó đang đợi kênh từ bộ điều khiển. Trạng thái shiftdow được sử dụng để tính toán 1 dịch chuyển hàng đợi (nó cần thiết khi phần tử đầu bỏ đi) cái mẫu này sử dụng một mảng số nguyên như 1 hang đợi FIFO.



Hình 21. Mẫu hàng đợi Queue

3.2.3 Xác minh

Chúng ta kiểm tra các thuộc tính đơn giản như reachability,safety,liveness và để tránh deadlock. Các thuộc tính đến được reachability đơn giản kiểm tra xem 1 trạng thái được đưa ra là đến được:

KẾT LUẬN

Đồ án đã đi tìm hiểu lý thuyết về Model Checking, hệ thống thời gian thực. Để đặc tả các hệ thống thời gian thực, các nhà thiết kế phần mềm thường sử dụng otomat thời gian với công cụ hỗ trợ Uppaal. Trên cơ sở đó,đồ án đã tìm hiểu về lý thuyết otomat thời gian: các khái niệm, các ví dụ hệ thời gian thực có thêm ràng buộc về thời gian. Đồng thời đã tìm hiểu về công cụ Uppaal trong việc đặc tả otomat thời gian và trình bày chương trình thử nghiệm của Uppaal (Mô tả bài toán Train - Gate).

Đồ án bước đầu làm quen với việc đặc tả và kiểm chứng hệ thời gian thực. Trong thời gian tiếp theo em sẽ đi tìm hiểu sâu hơn về các kĩ thuật khác liên quan. Đồng thời em sẽ tìm hiểu kĩ hơn về công cụ Model Checking rất hiệu quả là Uppaal.

TÀI LIỆU THAM KHẢO

AD-ATheoryOfTimedAutomata[1]

Model checking timed Automata – Sergio – Yovine – Verimag – Centre – Equation – 2.Av.De.Vignate- 38610 Gières - France

new-tutorial - org UnivGerd – Behrmann Alexandre David, and Kim G.Lasen – department of computer sciense, Aabersity, Denmark.