Shape, rectangle

Description automatically generated`

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**



**Giảng viên hướng dẫn:** LÊ HOÀNG NGHĨA

**Sinh viên thực hiện: nhóm 12**

Nguyễn Linh Anh Khoa 20520219

Trương Trọng Hiếu 20520185

Lê Hoàng Thịnh 20521959

**BÁO CÁO BÀI TẬP LỚN CUỐI KỲ**

**LỚP CE224.N11**

**TP. Hồ Chí Minh, 2023**

**TP. Hồ Chí Minh, tháng 4 năm 2022**

**TP. Hồ Chí Minh, tháng 4 năm 2022**

**TP. Hồ Chí Minh, tháng 4 năm 2022**

**TP. Hồ Chí Minh, tháng 4 năm 2022**

**TP. Hồ Chí Minh, tháng 4 năm 2022**

**TP. Hồ Chí Minh, tháng 4 năm 2022**

**TP. Hồ Chí Minh, tháng 4 năm 2022**

**TP. Hồ Chí Minh, tháng 4 năm 2022**

**TP. Hồ Chí Minh, tháng 4 năm 2022**

**TP. Hồ Chí Minh, tháng 4 năm 2022**

**TP. Hồ Chí Minh, tháng 4 năm 2022**

**TP. Hồ Chí Minh, tháng 4 năm 2022**

**TP. Hồ Chí Minh, tháng 4 năm 2022**

**TP. Hồ Chí Minh, tháng 4 năm 2022**

**TP. Hồ Chí Minh, tháng 4 năm 2022**

|  |  |  |  |
| --- | --- | --- | --- |
| **MSSV** | **Họ và Tên** | **Phần trăm thực hiện** | **Công việc** |
| 20520219 | Nguyễn Linh Anh Khoa | 35% | vẽ FSM, viết báo cáo, tổng hợp |
| 20520185 | Trương Trọng Hiếu | 35% | Vẽ FSM, tổng hợp |
| 20521959 | Lê Hoàng Thịnh | 30% | Vẽ FSM |

**MỤC LỤC**

[Phần 1. Tổng hợp 4](#_Toc123504695)

[Phần II: Mô hình FSM 12](#_Toc123504696)

# Phần 1. Tổng hợp

Từ file Finite State Machines and Modal Models in Ptolemy II

\*bộ bảo vệ-> phần kiểm tra(guards như các điều kiện để chuyển đổi state)

**2. Finite State Machines:** (p.2)

**Máy hữu hạn trạng thái** là hệ thống mà output sẽ phụ thuộc không chỉ vào trạng thái hiện tại mà còn là trạng thái hiện tại của cả hệ thống. Trạng thái của một hệ thống là tổng hợp của mọi thứ cần biết về trạng thái trước đó để xử lý output. Trạng thái của hệ thống có thể được đại diện bởi một biến trạng thái s ∈ Σ, Σ là tổng hợp toàn bộ trạng thái có thể xảy ra. **FSM** là một máy hữu hạn trạng thái khi Σ là hữu hạn.

**Ví dụ 1**: Giả thuyết một máy điều nhiệt điều khiển một máy sưởi là một máy trạng thái với Σ={*heating,cooling*}. Nếu trạng thái là *s=heating* thì máy sưởi được bật. Nếu *s=cooling* thì máy sưởi tắt. Giả định mục tiêu nhiệt độ là 20 độ C. Nếu máy sưởi đang bật, máy điều nhiệt sẽ cho nhiệt độ tăng đến qua mục tiêu, cho là 22 độ. Nếu máy sửi đang tắt thì máy điều nhiệt cho nhiệt xuống dưới mục tiêu, cho là 18 độ. Như vậy, hành vi dựa trên trạng thái, thứ sẽ kết luận trạng thái đã qua bằng việc ghi nhớ máy sưởi đã bật hay tắt. Chiến lược này tránh sự **biến động cao**, khi mà máy sưởi được bật và tắt liên tục một cách tức thời lúc nhiệt độ tiệm cận với nhiệt độ mục tiêu.

**2.1. lkFSMActor.**

FSMActor là một Actor tổng hợp khi sàng lọc bao gồm cả trạng thái và sự biến đổi(chuyển tiếp), khác biệt với Actor khác. Ptolemy II cung cấp ký hiệu trục quan cho những trạng thái và biến đổi này(được thể hiện ở Figure 1). Ở figure này, FSMActor có 2 input và 2 ouput, được tạo ra bằng công cụ model builder. Nói chung, FSMActor có thể tùy chỉnh số lượng cổng input và ouput. Actor giao tiếp với input rồi tạo ra output được xác định bằng một FSM, được quan sát trực quan ở dưới cùng của figure. FSM chứa một số lượng hữu hạn trạng thái(trong figure là 3 trạng thái). Một trong những trạng thái này là **trạng thái đầu**(được gắn nhãn initialState), là state bắt đầu thực hiện của actor trong model. Trạng thái đầu được biểu thị bằng một đường viền đậm. Một vài trạng thái có thể là **trạng thái kết thúc,** biểu hiện trực quan bằng một đường viền đôi(thông tin nhiều hơn về trạng thái kết thúc sẽ xuất hiện sau). Các trạng thái được kết nối bằng các chuyển tiếp, là chú thích thứ gì sẽ quyết định sẽ xảy ra khi actor thực thi. Trước khi đi vào chi tiết thì ví dụ về máy điều nhiệt có thể giúp cho ký hiệu được trực giác hóa.

**SỰ RA ĐỜI CỦA STATE MACHINES (p.3)**

Máy trạng thái thường được mô tả trong tài liệu bằng bộ 5(Σ,*I*,*O*,*T*,σ). Σ là số trạng thái và σ là trạng thái đầu tiên. Máy trạng thái bất định có thể có nhiều hơn một trạng thái khởi đầu, trong trường hợp σ ⊂ Σ là chính nó thể hiện cả bộ trạng thái, trường hợp đặc biệt này không được hỗ trợ bởi Ptolemy II FSM. *I* là một bộ các giá trị được định cho input. Trong Ptolemy II FSM, *I* là một hàm số dưới dạng   
*i*: *Pi* → *D* ∪ ε, *Pi* là tập hợp các cổng input(hoặc tên các cổng input), *D* là bộ giá trị thể hiện cho các cổng input ở một giai đoạn thực thi cụ thể, ε đại diện cho các input “vắng mặt”(*i*(*p*) = ε khi *p*\_isPresent được xác định là False). *O* giống với bộ định tất cả các giá trị khả thi cho các cổng output ở một giai đoạn thực thi.

Với một máy trạng thái khả định, *T* là một hàm/chức năng dưới dạng T : Σ × *I* → Σ × *O*, thể hiện cho các quan hệ chuyển tiếp trong FSM. Trên thực tế, bộ bảo vệ và đầu ra chỉ là mã hóa của chức năng này. Đối với một máy trạng thái bất định(được Ptolemy II hỗ trợ), miền của *T* là bộ nguồn của Σ × *O*, cho phép có nhiều hơn một điểm trạng thái đến và giá trị của output.

Giả thiết cổ điển của của máy trạng thái phân biệt giữa **máy Mealy** và **máy Moore**. Máy Mealy kết hợp output và bộ chuyển đổi. Máy Moore kết hợp giữa output và trạng thái. Ptolemy II hỗ trợ cả hai loại, sử dụng output action cho Mealy và cải chế trạng thái trong Modal Model cho Moore.

Máy trạng thái của Ptolemy II thực chất là máy trạng thái mở rộng, đòi hỏi một model giàu tài nguyên hơn cái ở trên. FSM mở rộng thêm vào model ở trên bộ *V* thể hiện các giá trị của biến số, là thể hiện hàm dưới dạng *v* : *N* → *D*, khi đó *N* là bộ tên của biến số và *D* là bộ giá trị mà biến số đang mang. Một máy FSM mở rộng là bộ 6 (Σ,*I*,*O*,*T*,σ,*V*) khi đó sự chuyển tiếp được hàm hóa dưới dạng *T* : Σ × *I* × *V* → Σ × *O* × *V*(đối với máy trạng thái khả định). Chức năng này được mã hóa bởi sự biến chuyển, bộ bảo vệ, output, và các bộ thực thi của FSM.

**Ví dụ 2**: Một model máy biến nhiệt và máy sưởi được thể hiện trong Figure 2. Người đọc nên hiểu được figure dễ dàng. Trong figure đó, FSMActor có input *temperature* và output *heat*. Hành vi của nó được định bởi FSM trong ô màu xám. Có 2 trạng thái, Σ ={*heating,cooling*}. Có 4 bộ chuyển tiếp. Bộ bảo vệ của mỗi chuyển tiếp sẽ cho ra điều kiện nhận biết bộ chuyển tiếp đang thực thi. Bộ xử lý đầu ra trên từng bộ chuyển tiếp đưa ra giá trị ở output của bộ chuyển tiếp đó khi nó đang được dùng. Đọc qua diagram, ta thấy khi đang ở trạng thái *heating*, nếu input *temperature* thấp hơn *heatOffThreshold*(22.0) thì giá trị output là *heatingRate*(0.1). Khi input *temperature* cao hơn hoặc bằng *heatOffThreshold* thì FSM chuyển thành trạng thái *cooling* và xử lý sao cho output thành *coolingRate*(-0.05). Một ví dụ thực thi nữa được cho trong Figure 4.

**2.2. Luật thực thi của FSMActor.**

Một FSMActor chứa một bộ những trạng thái và bộ chuyển tiếp trạng thái. Một trong những trạng thái là trạng thái ban đầu, bất kể trạng thái nào cũng có thể là trạng thái kết thúc. Mỗi bộ chuyển tiếp sẽ có một “biểu thức/điều kiện” bảo vệ, các cổng output và các bộ đặt giá trị(**set actions**). Khi bắt đầu thực thi, trạng thái của actor được đặt là trạng thái bắt đầu. Sau đó, mỗi lần hành động của actor là một chuỗi liên tục các bước tiến hành. Ở phương pháp fire(), actor:

1. Đọc input đầu vào;-

2. đánh giá các bộ bảo vệ các chuyển tiếp của trạng thái hiện tại;-

3. chọn một bộ chuyển tiếp có bộ bảo vệ được bật **True** và;-

4. thực thi xử lý đầu ra trên bộ chuyển tiếp được chọn nếu có. Trong phương pháp postfire(), actor;-

5. thực thi bộ đặt (set action) của bộ chuyển tiếp hiện tại, và;-

6. chuyển đổi trạng thái hiện tại đến trạng thái đích theo bộ chuyển đổi được chọn.

(p.6)

Đây là 2 cách riêng biệt để hỗ trợ cho actor trong miền dùng cho lặp điểm cố định(fixed-point iteration) như SR và Continous đã giải thích ở Section 2.9. Ở miền không thuộc những thứ ở trên(như là PN, SDF, DDF), bước 1 đến bước 6 có thể thực hiện liên tục trong mỗi lần lặp lại, và điểm khác nhau giữa fire() và postfire() không quan trọng nữa.

**Kiến thức thêm: Hiện tượng trễ**

Máy biến nhiệt ở ví dụ 2 đã trình bày được một hình thái cụ thể của hành vi lệ thuộc trạng thái được gọi là **hiện tượng trễ**. Một hệ thống có hiện tượng trễ mang tính chất là thang thời gian tuyệt đối là không liên hệ nhau. Giả sử input là một “hàm số” gì đó về thời gian, x:R → R (với máy biến nhiệt, x(t) là nhiệt độ ở thời điểm t). Cho là input x làm cho output y:R → R, cũng là một hàm số của thời gian. Tương tự, trong Figure 4, tín hiệu x là cao còn y là thấp. Trong hệ thống này, thay vì x là input thì x’ được tính bởi:

x’(t) = x(α·t)

đối với hằng số không âm α thì output y’ được tính bởi:

y’(t) = y(α·t)

Chia tỷ lệ trục thời gian input dẫn đến tỷ lệ trục thời gian ở output, cho nên thang thời gian tuyệt đối là không liên quan nhau.

Một cách thực hiện khác cho máy biến nhiệt là sử dụng một ngõ nhiệt độ đơn, nhưng phải bắt buộc rằng máy sưởi được giữ tắt hoặc bật trong một khoảng thời gian nhất định, bất kể nhiệt độ là bao nhiêu. Thiết kế này sẽ không có được tính chất của hiện tượng trễ.

(p.9)

**2.3. Tham chiếu giá trị input của FSMActor.**

Nếu một cổng input tên *portName* được dùng làm điều kiện bảo vệ, nó sẽ nhắm vào input hiện tại ở port đó trên channel zero. Như ví dụ trong Figure 2, điều kiện bảo vệ “temperature < heaOffThreshold” thì biến *temperature* sẽ nhắm đến giá trị hiện tại của cổng input tên *temperature*, và *heatOffThreshold* sẽ dựa trên tham số tên *heatOffThreshold*.

Trong nhiều model để tính toán, một input có thể vắng mặt. nếu một port *p* vắng, thì một điều kiện như “p < 10” sẽ định là False. Tuy nhiên, nếu điều kiện có dạng “p < 10 || true” sẽ định là True. Một kỹ thuật rõ ràng, minh bạch hơn làm cho model máy trạng thái dễ đọc hơn là sử dụng symbol portName\_isPresent trong điều kiện bảo vệ. Đây là một thức nhị phân sẽ trả về True nếu input trên port *portName* có mặt.

**2.4. Output Actions.**

Khi một bộ chuyển tiếp được chọn, **bộ xử lý output(Output Actions)** của nó sẽ được thực thi. Khâu xử lý output được định bởi tham số *outputActions* của bộ chuyển tiếp. Mẫu của một output action thường là *portName* = *expression*, còn điều kiện có thể dựa trên giá trị input như trên hoặc bất kì tham số nào trong phạm vi. Như trong ví dụ Figure 2, dòng:

output: heat = coolingRate

xác định rằng cổng output *heat* nên trình bày ra giá trị được cho từ tham số *coolingRate*. Đây là hành vi của **máy Mealy**, là loại mà output được tạo ra bởi bộ chuyển đổi hơn là trạng thái. Hành vi của **máy Moore** cũng có thể thực hiện được bằng cách dùng “state refinement” tạo ra output, như giải thích ở Section 3.

Nhiều output action có thể được cho bằng cách chia chúng ra với dấu chấm phẩy, như *port1 = expression1; port1 = expression1*

**2.5. Bộ hành động và Máy trạng thái mở rộng.**

Bộ hành động/thực thi(**set actions**) trên một bộ chuyển đổi có thể được dùng để đặt giá trị của tham số trong máy trạng thái. Một công dụng thực tế của nó là tạo máy trạng thái mở rộng(**extended FSM**), thực chất là một máy hữu hạn trạng thái mở rộng với một biến số là trạng thái dạng số. Gọi là “mở rộng” vì số trạng thái dựa trên con số rõ ràng mà biến số có thể mang. Nó có thể là vô hạn.

**Ví dụ 3**: Một ví dụ đơn giản của máy FSM mở rộng được thể hiện ở Figure 7. Ở ví dụ này, FSMActor có tham số *count*, được thể hiện bằng dấu chấm nhỏ màu xanh cạnh nó. Bộ chuyển tiếp từ trạng thái đầu tiên *init* đến trạng thái *counting* khởi tạo *count* thành 0 trong bộ đặt giá trị. Trạng thái *counting* có hai bộ chuyển tiếp: một là tự chuyển về chính mình, hai là chuyển đến *final*. Chuyển tiếp về chính trạng thái hiện tại sẽ xảy ra khi *count* bé hơn 5. Chuyển tiếp đó sẽ làm tăng giá trị của *count* 1 đơn vị. Khi giá trị của *count* đến 5, chuyển tiếp đến *final* sẽ xảy ra. Trước đó, output được đặt ngang bằng với input. Khi thực hiện quá trình chuyển tiếp đó, output sẽ giữ nguyên không đổi. Vì vậy, output của model này là dãy 0, 1, 2, 3, 4, 5, 5, 5,…

**2.6. Trạng thái đích (Final States).**

Một FSM có thể có trạng thái đích/cuối, là trạng thái mà khi được thực thi sẽ thể hiện kết thúc của quá trình thực thi của cả máy trạng thái.

**Ví dụ 4:** Một biến thể của ví dụ 3 được thể hiện ở Figure 8, Biến thể này có tham số *isFinalState* của trạng thái *final* được đặt là *true*. Điều này được thể hiện bằng đường vẽ kép bao quanh trạng thái. Khi tiến vào trạng thái đó, FSMActor chỉ ra cho bộ điều khiển rằng nó không muốn thực thi nữa (FSMActor thực hiện việc này bằng cách trả về *false* từ postfire()). Kết quả là output gửi đến bộ phận *Display* là dãy số hữu hạn 0, 1, 2, 3, 4, 5, 5.

Trong vòng thực hiện, FSMActor tiếng vào trạng thái có đánh dấu cuối cùng thì postfire() của FSMActor trả về False. Việc này chỉ định cho bộ điều khiển rằng FSMActor không muốn thực hiện lần nữa. Đa số bộ điều khiển sẽ tránh việc chạy actor đó nữa, nhưng vẫn tiếp tục thực thi model. Bộ điều khiển SDF thì khác vậy. Bởi vì nó giả định mức tiêu thụ thông thường cũng những tỉ lệ đầu ra cho toàn bộ actor, và vì nó xây dựng thời gian biểu một cách tĩnh nên không thể cho phép actor từ chối thực thi. Vì thế, bộ điều khiển SDF sẽ dừng việc chạy model ngay khi có bất kì actor nào trả về False từ postfire().

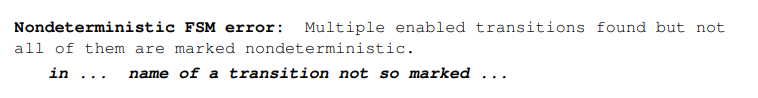
**2.7. Bộ chuyển tiếp mặc định. (Default Transitions)**

Máy FSM có thể có những chuyển tiếp mặc định, là những chuyển tiếp có tham số mặc định được đặt là True(xem Figure 6). Những chuyển tiếp này được dùng nếu không có bất kì chuyển tiếp ra ngoài nào của trạng thái hiện tại được kích hoạt. Chuyển tiếp mặc định được thể hiện dưới dạng vòng cung nét đứt thay vì nét liền.

**Ví dụ 5**: FSM trong Figure 2 có thể được thực hiện bằng cách sử dụng chuyển tiếp mặc định được thể hiện ở Figure 9. Ở đây, các chuyển tiếp mặc định chỉ đơn giản chỉ ra là bộ chuyển tiếp đi đến trạng thái khác không được bật, nên FSM giữ nguyên trạng thái và cho ra output.

Nếu một chuyển tiếp mặc định có một điều kiện bảo vệ thì bộ chuyển tiếp đó chỉ được bật khi bộ bảo vệ trả về True ***và*** không có bất kì chuyển tiếp không mặc định nào thực thi. Ghi chú rằng sử dụng chuyển tiếp mặc định với các model tính toán theo thời gian có thể hơi phức tạp.

**2.8. Máy trạng thái bất định. (Nondeterministic State Machines)**

Nếu có nhiều hơn một bộ bảo vệ trả về True ở mọi thời điểm thì FSM đó là một máy trạng thái bất định. Bộ chuyển tiếp có bộ bảo vệ trả về True một cách đồng thời được gọi là **Bộ chuyển tiếp bất định**. Mặc định thì, bộ chuyển tiếp không được bất định, nên nếu nhiều hơn một bộ bảo vệ trả về True, bạn sẽ thấy một ngoại lệ như này:

Để cho phép chúng được mang tính bất định, đặt tham số *nondeterministic* thành True ở mỗi bộ chuyển tiếp có thể được kích hoạt trong lúc một chuyển tiếp khác đang kích hoạt.

**Ví dụ 6**: Một model máy biến nhiệt bị lỗi được thể hiện trong Figure 10. Khi FSM đang trong trạng thái *heating*, cả hai chuyển tiếp đều được bật (các bộ bảo vệ đều trả về True) nên đều có thể được cho phép thực thi. Cả hai chuyển tiếp đều được đánh dấu bất định, được biểu thị trực quan bằng đường chuyển tiếp màu đỏ. Đồ thị biểu diễn được thể hiện ở Figure 11. Lưu ý rằng máy sưởi lúc này sẽ được bật chỉ một khoảng ngắn, làm cho nhiệt độ loanh quanh 18 độ ở ngưỡng mà máy sưởi được bật.

Trong một FSM bất định, nếu có nhiều hơn một bộ chuyển tiếp được bật thì toàn bộ chúng đều được đánh dấu là bất định, một cái sẽ được chọn ngẫu nhiên ở thức fire(). Nếu fire() được gọi nhiều hơn một lần trong chu trình lặp lại thì các lần gọi tiếp theo trong cùng chu trình sẽ luôn chọn cùng một bộ chuyển tiếp.

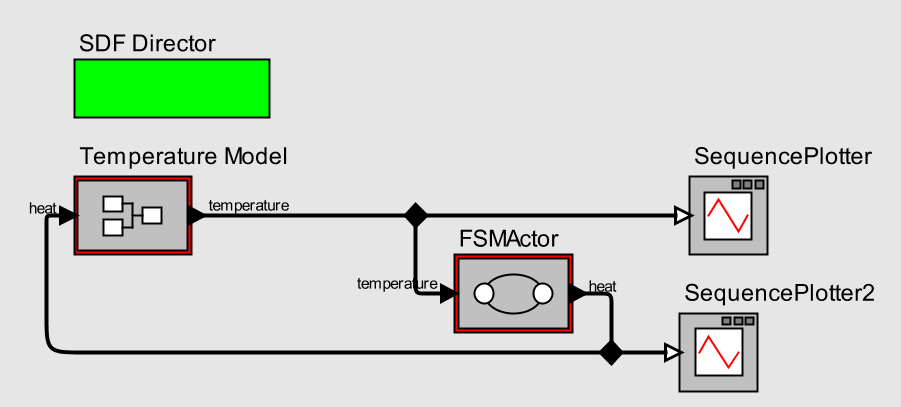
Nếu có nhiều hơn một bộ chuyển tiếp mặc định hướng ra một trạng thái thì những bộ chuyển tiếp này phải được đánh dấu là bất định hoặc trường hợp ngoại lệ sẽ xảy ra. Những chuyển tiếp mặc định bất định được thể hiện dưới dạng vòng cung nét đứt màu đỏ.

**2.9. Lặp điểm cố định. (Fixed-Point Iterations)**

Trong mục 2.2 trên đã giải thích được về sự thực thi của FSMActor được chia thành hai phần, bước 1 đến bước 4 được biểu diễn ở phần thức fire(), và bước 5-6 được biểu diễn ở phần postfire(). Sự tách biệt này là cần thiết trong miền để biểu diễn được lặp điểm cố định, như SR và Continous. Trong lĩnh vực với lặp điểm cố định, thức fire() có thể được gọi nhiều lần trong một chu kỳ lặp trong khi bộ điều khiển tìm hướng đi.

# Phần II: Mô hình FSM

**Diagram

Description automatically generated with medium confidenceDiagram

Description automatically generatedBài 1**

**Diagram

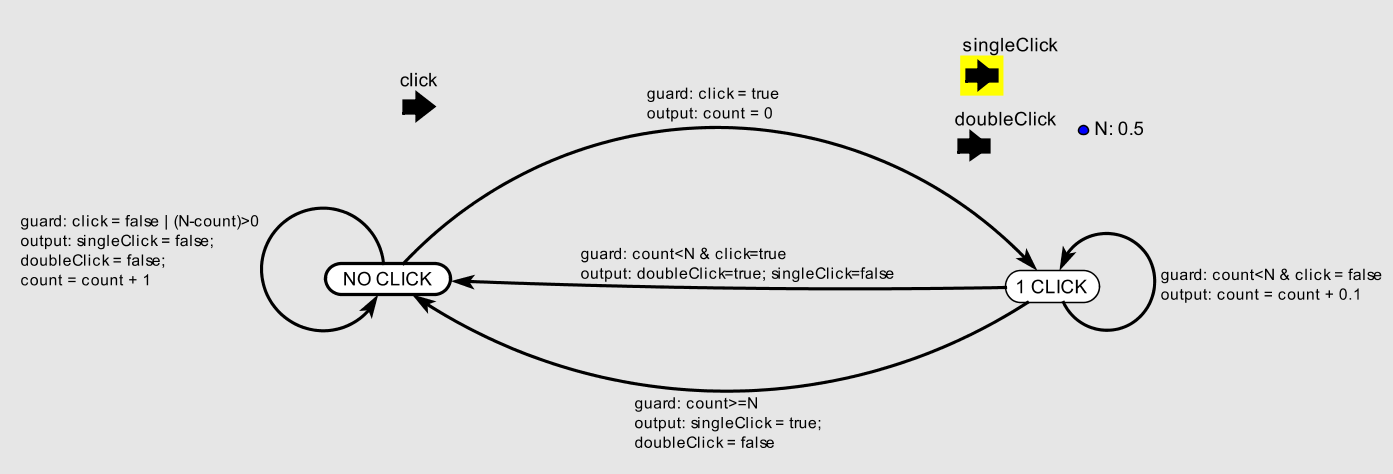
Description automatically generated**

**Graphical user interface, text, application

Description automatically generatedBài 2**

**Diagram

Description automatically generated**

****

**Bài 3**

**Diagram

Description automatically generatedDiagram

Description automatically generatedDiagram

Description automatically generatedGraphical user interface, text, application

Description automatically generateda)**

**Shape

Description automatically generatedDiagram

Description automatically generatedGraphical user interface, text

Description automatically generatedb)**

**Bài 4**

**Shape

Description automatically generated with medium confidenceGraphical user interface, text, table

Description automatically generated with medium confidencea)**

**Diagram

Description automatically generated**

**Shape

Description automatically generatedTable

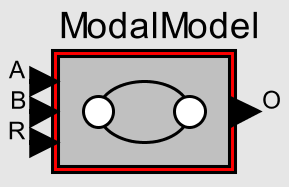
Description automatically generated with medium confidenceb)**

**Diagram

Description automatically generated**

**Graphical user interface, text, application

Description automatically generatedGraphical user interface, application, chat or text message

Description automatically generatedc)**

**A picture containing text

Description automatically generatedDiagram

Description automatically generatedGraphical user interface, diagram

Description automatically generated with medium confidence**

**Bài 5**

**Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generatedShape

Description automatically generatedA picture containing graphical user interface

Description automatically generateda)**

**Text, application

Description automatically generatedText

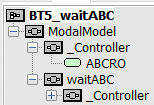
Description automatically generated**

**Graphical user interface, text, application, chat or text message

Description automatically generated**

**Diagram

Description automatically generatedGraphical user interface, application

Description automatically generatedShape

Description automatically generatedb)**

**Graphical user interface, text

Description automatically generatedc)**

**A picture containing text, clipart

Description automatically generatedDiagram

Description automatically generatedDiagram

Description automatically generated with low confidence**

**Diagram

Description automatically generated with low confidenceShape

Description automatically generatedGraphical user interface

Description automatically generated with medium confidenced)**

**Diagram

Description automatically generatedA picture containing text, clipart, screenshot

Description automatically generated**

**Graphical user interface, text, application, chat or text message

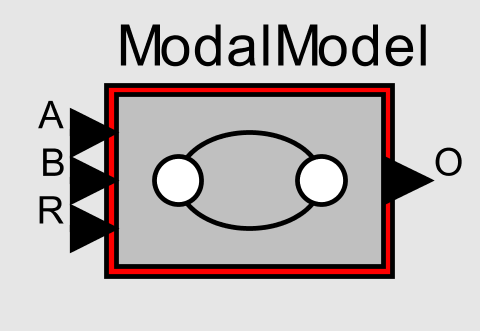
Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**Bài 6**

**Graphical user interface, text

Description automatically generatedDiagram

Description automatically generated**

**Diagram

Description automatically generated with medium confidenceBài 7**

**Diagram

Description automatically generateda)**

**b)**

**Diagram

Description automatically generated**

**Diagram

Description automatically generatedc)**

**Bài 8**

1. Có thể xây dựng mô hình
2. Không thể đạt được một biến thể nhỏ gọn tương tự của ABCRO nếu không giới hạn

**Bài 9**

1. Output: A

2. Output: B

3. Output: A

4. Output: A

5. Output: A

6. Output: C

1. Output: A

2. Output: B

3. Output: A

4. Output: A

5. Output: A

6. Output. C