Sơ đồ trạng thái

Từ Wikipedia, bách khoa toàn thư miễn phí (State diagram)

Một sơ đồ trạng thái cho một cánh cửa mà chỉ có thể được mở và đóng

Biểu đồ trạng thái là một loại sơ đồ được sử dụng trong khoa học máy tính và các lĩnh vực có liên quan để mô tả hành vi của hệ thống. Các biểu đồ trạng thái yêu cầu hệ thống được miêu tả bao gồm một số hữu hạn các trạng thái; Đôi khi, điều này thực sự là trường hợp, trong khi những lần khác đây là một trừu tượng hợp lý. Nhiều hình thức sơ đồ trạng thái tồn tại, khác biệt một chút và có ngữ nghĩa khác nhau.

Mục lục [ẩn]

1. Khái quát chung

2 Đồ thị trực tiếp

2.1 Ví dụ: DFA, NFA, GNFA, hoặc máy Moore

2.2 Ví dụ: Máy tiện

3 biểu đồ Harel

4 Semantics thay thế

5 Sơ đồ nhà nước so với sơ đồ khối

6 Các phần mở rộng khác

7 Xem thêm

8 Tham khảo

9 Liên kết ngoài

Tổng quan [sửa]

Sơ đồ trạng thái được sử dụng để đưa ra một mô tả trừu tượng về hành vi của một hệ thống. Hành vi này được phân tích và đại diện như là một loạt các sự kiện có thể xảy ra ở một hoặc nhiều trạng thái có thể. Dưới đây "mỗi sơ đồ thường đại diện cho các đối tượng của một lớp đơn và theo dõi các trạng thái khác nhau của các đối tượng của nó thông qua hệ thống". [1]

Các biểu đồ trạng thái có thể được sử dụng để biểu thị bằng đồ hoạ các máy trạng thái hữu hạn. Điều này đã được giới thiệu bởi C.E. Shannon và W. Weaver trong cuốn sách năm 1949 của họ "Lý thuyết toán học truyền thông". Một nguồn khác là Taylor Booth trong cuốn sách "Sequential Machines and Automata Theory" năm 1967 của ông. Một đại diện có thể khác là bảng chuyển tiếp của Nhà nước.

Biểu đồ trực tiếp [sửa]

Một đồ thị trực tiếp

Một hình thức cổ điển của sơ đồ trạng thái cho một máy trạng thái hữu hạn hoặc tự động hữu hạn (FA) là một đồ thị trực tiếp với các phần tử sau (Q, Σ, Z, δ, q0, F): [2] [3]

Vertices Q: tập hợp các trạng thái hữu hạn, thường được biểu diễn bởi các vòng tròn và được gắn nhãn với các ký hiệu chỉ định duy nhất hoặc các từ được viết bên trong chúng

Ký hiệu đầu vào Σ: một tập hợp hữu hạn các ký hiệu đầu vào hoặc thiết kế

Các biểu tượng đầu ra Z: một tập hợp hữu hạn các biểu tượng đầu ra hoặc thiết kế

Chức năng đầu ra ω biểu diễn việc lập bản đồ các cặp có vị trí của các ký hiệu đầu vào và các trạng thái vào các biểu tượng đầu ra, được biểu thị toán học bằng ω: Σ × Q → Z.

Các cạnh δ: thể hiện sự chuyển tiếp từ trạng thái này sang trạng thái khác do đầu vào (được xác định bằng các ký hiệu được vẽ trên các cạnh). Mép thường được vẽ như mũi tên hướng từ trạng thái hiện tại đến trạng thái kế tiếp. Lập bản đồ này mô tả quá trình chuyển đổi trạng thái xảy ra khi nhập vào một biểu tượng cụ thể. Điều này được viết bằng toán học như δ: Q × Σ → Q, do đó δ (hàm chuyển tiếp) trong định nghĩa của FA được cho bởi cả hai đỉnh kết nối bởi một cạnh và biểu tượng trên một cạnh trong một biểu đồ mô tả FA . Δ (q, a) = p trong định nghĩa của FA nghĩa là từ trạng thái q dưới ký hiệu đầu vào a, sự chuyển tiếp sang trạng thái p xảy ra trong máy này. Trong biểu đồ đại diện cho FA, điều này được biểu diễn bởi một cạnh được gắn nhãn bởi một điểm từ đỉnh được dán nhãn bởi q đến đỉnh được gắn nhãn bởi p.

Bắt đầu trạng thái q0: (không được hiển thị trong các ví dụ dưới đây). Trạng thái khởi đầu q0 ∈ Q thường được biểu diễn bởi một mũi tên không có nguồn gốc chỉ tới trạng thái. Trong các văn bản cũ hơn, [2] [4] trạng thái bắt đầu không được hiển thị và phải được suy ra từ văn bản.

Chấp nhận nhà nước F: Nếu được sử dụng, ví dụ để chấp nhận automata, F ∈ Q là trạng thái chấp nhận. Nó thường được vẽ như một vòng tròn đôi. Đôi khi (các) trạng thái chấp nhận là các trạng thái "Cuối cùng" (dừng lại, bị mắc kẹt). [3]

Đối với một automaton hữu hạn xác định (DFA), automaton hữu hạn không xác định (NFA), automaton hữu hạn không xác định (GNFA) tổng quát, hoặc Moore machine, đầu vào được biểu hiện trên mỗi cạnh. Đối với máy Mealy, đầu vào và đầu ra được biểu thị trên mỗi cạnh, tách ra bằng dấu gạch chéo "/": "1/0" biểu thị sự thay đổi trạng thái khi gặp biểu tượng "1" gây ra biểu tượng "0". Đối với máy Moore, sản lượng của nhà nước thường được viết trong vòng tròn của tiểu bang, cũng cách nhau bởi dấu phân cách của tiểu bang với dấu gạch chéo "/". Cũng có những biến thể kết hợp hai ký hiệu này.

Ví dụ: nếu một tiểu bang có một số kết quả đầu ra (ví dụ: "a = xe máy theo chiều kim đồng hồ = 1, b = ánh sáng cẩn thận không hoạt động = 0") biểu đồ nên phản ánh điều này: "Q5 / 1,0" chỉ trạng thái q5 với đầu ra a = 1, b = 0. Thiết kế này sẽ được viết bên trong vòng tròn của tiểu bang.

Ví dụ: DFA, NFA, GNFA, hoặc máy Moore [sửa]

S1 và S2 là các trạng thái và S1 là một trạng thái chấp nhận hoặc trạng thái cuối cùng. Mỗi cạnh được gắn nhãn với đầu vào. Ví dụ này cho thấy một chấp nhận cho các chuỗi trên {0,1} có chứa một số chẵn.

DFAexample.svg

Ví dụ: Máy gia vị [sửa]

S0, S1, và S2 là các trạng thái. Mỗi cạnh được dán nhãn với "j / k" trong đó j là đầu vào và k là đầu ra.

Sơ đồ trạng thái của một máy gia vị đơn giản

Harel statechart [sửa]

Harel statecharts [5] đang sử dụng rộng rãi vì một biến thể đã trở thành một phần của Unified Modeling Language (UML). Loại sơ đồ cho phép mô hình hóa siêu cường, các vùng trực giao, và các hoạt động như một phần của một quốc gia.

Các sơ đồ trạng thái cổ điển yêu cầu tạo ra các nút khác biệt cho mỗi sự kết hợp hợp lệ các tham số xác định trạng thái. Điều này có thể dẫn đến một số lượng rất lớn các nút và chuyển tiếp giữa các nút cho tất cả, nhưng đơn giản nhất của hệ thống (nhà nước và vụ nổ chuyển tiếp). Sự phức tạp này làm giảm khả năng đọc của sơ đồ trạng thái. Với biểu đồ trạng thái Harel, có thể mô hình nhiều biểu đồ trạng thái chéo giữa các trạng thái trong sơ đồ trạng thái. Mỗi máy trạng thái chéo chức năng này có thể chuyển đổi nội bộ mà không ảnh hưởng đến các máy trạng thái khác trong sơ đồ trạng thái. Trạng thái hiện tại của mỗi máy trạng thái chéo trong bảng trạng thái xác định trạng thái của hệ thống. Biểu đồ trạng thái Harel tương đương với sơ đồ trạng thái nhưng nó cải thiện khả năng đọc của sơ đồ kết quả.

Ngữ nghĩa khác [sửa]

Có những bộ ngữ nghĩa khác có thể biểu diễn sơ đồ trạng thái. Ví dụ, có các công cụ cho việc lập mô hình và thiết kế logic cho bộ điều khiển nhúng [6]. Các sơ đồ này, như các máy trạng thái ban đầu của Harel, [7] hỗ trợ các trạng thái lồng nhau theo cấp bậc, các vùng trực giao, các hành động của nhà nước và các hành động chuyển tiếp [8].

Sơ đồ trạng thái so với sơ đồ khối [sửa]

Những người mới đến với chế độ chính thức của máy nhà nước thường nhầm lẫn các sơ đồ trạng thái với sơ đồ luồng. Hình dưới đây cho thấy sự so sánh sơ đồ trạng thái với sơ đồ luồng. Một máy trạng thái (panel (a)) thực hiện các hành động để đối phó với các sự kiện rõ ràng. Ngược lại, sơ đồ (bảng điều khiển (b)) không cần sự kiện rõ ràng mà thay vào đó chuyển từ nút này sang biểu đồ của nó một cách tự động sau khi hoàn thành các hoạt động.

Sơ đồ trạng thái (a) và sơ đồ (b)

Các nút biểu đồ là các cạnh trong biểu đồ cảm ứng của các trạng thái. Lý do là mỗi nút trong một flowchart đại diện cho một lệnh của chương trình. Một lệnh chương trình là một hành động sẽ được thực hiện. Vì vậy, nó không phải là một nhà nước, nhưng khi áp dụng cho nhà nước của chương trình, nó dẫn đến việc chuyển đổi sang một bang khác.

Cụ thể hơn, danh sách mã nguồn biểu thị một biểu đồ chương trình. Thực hiện đồ thị chương trình (phân tích cú pháp và giải thích) kết quả trong một đồ thị trạng thái. Vì vậy, mỗi biểu đồ chương trình tạo ra một biểu đồ trạng thái. Việc chuyển đổi đồ thị chương trình sang biểu đồ trạng thái liên kết được gọi là "mở ra" của biểu đồ chương trình.

Biểu đồ chương trình là một chuỗi lệnh. Nếu không có biến nào tồn tại, thì tiểu bang chỉ bao gồm bộ đếm chương trình, theo dõi vị trí trong chương trình chúng ta đang thực thi (lệnh tiếp theo sẽ được áp dụng là gì).

Trong trường hợp này trước khi thực hiện một lệnh, chương trình truy cập là ở một số vị trí (nhà nước trước khi lệnh được thực hiện). Thực hiện lệnh di chuyển chương trình truy cập đến lệnh tiếp theo. Kể từ khi truy cập chương trình là toàn bộ nhà nước, nó sau đó thực hiện lệnh thay đổi nhà nước. Vì vậy bản thân lệnh tương ứng với sự chuyển tiếp giữa hai trạng thái.

Bây giờ xem xét trường hợp đầy đủ, khi các biến tồn tại và bị ảnh hưởng bởi các lệnh chương trình đang được thực hiện. Sau đó, giữa các địa điểm truy cập chương trình khác nhau, không chỉ làm thay đổi chương trình truy cập, nhưng các biến cũng có thể thay đổi các giá trị, do các lệnh thực hiện. Do đó ngay cả khi chúng ta xem lại một số lệnh của chương trình (ví dụ như trong một vòng lặp), điều này không ngụ ý chương trình nằm trong cùng một trạng thái.

Trong trường hợp trước, chương trình sẽ ở cùng trạng thái, bởi vì toàn bộ trạng thái chỉ là bộ đếm chương trình, vì vậy nếu chương trình counter trỏ đến cùng một vị trí (lệnh tiếp theo) nó chỉ cần xác định rằng chúng ta đang ở trong cùng một trạng thái. Tuy nhiên, nếu nhà nước bao gồm cả các biến, thì nếu những giá trị thay đổi đó, chúng ta có thể ở cùng vị trí của chương trình với các giá trị biến khác nhau, nghĩa là ở một trạng thái khác trong không gian trạng thái của chương trình. Thuật ngữ "mở ra" bắt nguồn từ sự nhân lên của các vị trí này khi tạo biểu đồ trạng thái từ biểu đồ chương trình.

Một ví dụ đại diện là một vòng lặp do tăng một số truy cập cho đến khi nó tràn và trở lại 0 lần nữa. Mặc dù vòng lặp Do thực thi cùng một lệnh increment lặp đi lặp lại, do đó biểu đồ chương trình thực hiện một chu kỳ, trong không gian trạng thái của nó không phải là một chu kỳ, mà là một dòng. Điều này kết quả từ trạng thái là vị trí của chương trình (ở đây là đi xe đạp) kết hợp với giá trị đếm, tăng lên một cách nghiêm ngặt (cho đến khi tràn), nên các trạng thái khác nhau được truy cập theo trình tự, cho đến khi tràn. Sau khi tràn bộ đếm trở lại 0, vì vậy trạng thái ban đầu được xem lại trong không gian trạng thái, đóng một chu kỳ trong không gian trạng thái (giả sử bộ đếm được khởi tạo thành 0).

Hình trên cho thấy sự đảo ngược của vai trò bằng cách sắp xếp các đường cung của sơ đồ trạng thái với các giai đoạn xử lý của sơ đồ.

Bạn có thể so sánh một sơ đồ khối với một dây chuyền lắp ráp trong quá trình sản xuất bởi vì sơ đồ luồng mô tả sự tiến triển của một số nhiệm vụ từ đầu đến cuối (ví dụ chuyển mã nguồn đầu vào thành đầu ra mã đối tượng bởi trình biên dịch). Một máy trạng thái thường không có khái niệm về sự tiến triển như vậy. Ví dụ, máy trạng thái cửa được hiển thị ở đầu bài viết này không ở giai đoạn nâng cao hơn khi nó ở trạng thái "đóng kín", so với trạng thái "mở"; Nó chỉ đơn giản phản ứng khác với các sự kiện mở / đóng. Một trạng thái trong một máy trạng thái là một cách hiệu quả để xác định một hành vi cụ thể chứ không phải là một giai đoạn chế biến.

Các phần mở rộng khác [sửa]

Một phần mở rộng thú vị là cho phép các vòng cung chảy từ bất kỳ trạng thái nào sang bất kỳ trạng thái nào. Điều này chỉ có ý nghĩa nếu hệ thống được phép ở trong nhiều trạng thái cùng một lúc, có nghĩa là một trạng thái cá nhân chỉ mô tả một điều kiện hoặc một khía cạnh khác của toàn thể trạng thái toàn cầu. Chủ nghĩa hình thức kết quả được biết đến như là một mạng Petri.

Một phần mở rộng khác cho phép tích hợp sơ đồ luồng trong các sơ đồ trạng thái của Harel. Phần mở rộng này hỗ trợ sự phát triển của phần mềm cả về sự kiện và định hướng công việc.

State Diagram UML

UML state machine, [1] còn được gọi là UML statechart, là một khái niệm toán học hóa một automaton hữu hạn trong các ứng dụng khoa học máy tính được biểu diễn trong ký hiệu Unified Modeling Language (UML).

Các khái niệm đằng sau nó là sắp xếp cách thiết bị, chương trình máy tính hoặc các quy trình khác (thường là kỹ thuật) sao cho một thực thể hoặc từng đơn vị con của nó luôn nằm trong chính xác một trong số các trạng thái có thể và nơi có tốt Xác định chuyển tiếp có điều kiện giữa các trạng thái này.

Máy trạng thái UML là một biến thể dựa trên đối tượng của biểu đồ trạng thái Harel, [2] được UML sửa đổi và mở rộng. [1] , [3] Mục tiêu của các máy UML là vượt qua những hạn chế chính của các máy trạng thái hữu hạn truyền thống trong khi vẫn giữ được những lợi ích chính. Các bảng trạng thái UML giới thiệu các khái niệm mới về trạng thái lồng nhau theo cấp bậc và các vùng trực giao, đồng thời mở rộng khái niệm hành động. Các máy trạng thái UML có các đặc tính của cả hai máy Món ăn và máy Moore. Chúng hỗ trợ các hành động phụ thuộc vào cả trạng thái của hệ thống và sự kiện khởi tạo, như trong Máy Món ăn, cũng như các hành động nhập cảnh và xuất cảnh, liên quan đến các trạng thái hơn là chuyển tiếp, như trong các máy của Moore.

Thuật ngữ "máy trạng thái UML" có thể tham khảo đến hai loại máy trạng thái: các máy trạng thái hành vi và các máy trạng thái giao thức. Các máy trạng thái hành vi có thể được sử dụng để mô hình hành vi của các thực thể riêng lẻ (ví dụ: trường hợp lớp). Các máy trạng thái giao thức được sử dụng để diễn tả các giao thức sử dụng và có thể được sử dụng để chỉ định kịch bản sử dụng hợp pháp của trình phân loại, giao diện và cổng.

Các khái niệm máy cơ bản cơ bản [sửa]

Nhiều hệ thống phần mềm được điều khiển theo sự kiện, có nghĩa là họ liên tục chờ đợi sự xuất hiện của một số sự kiện bên ngoài hoặc nội bộ chẳng hạn như nhấp chuột, nhấn nút, đánh dấu thời gian hoặc sắp tới của một gói dữ liệu. Sau khi nhận ra sự kiện, các hệ thống như vậy phản ứng bằng cách thực hiện tính toán thích hợp có thể bao gồm thao tác phần cứng hoặc tạo ra sự kiện "mềm" kích hoạt các thành phần phần mềm nội bộ khác. (Đó là lý do tại sao các hệ thống điều khiển sự kiện được gọi là các hệ phản động) Một khi xử lý sự kiện đã hoàn tất, hệ thống sẽ quay trở lại để chờ sự kiện tiếp theo.

Phản ứng đối với một sự kiện thường phụ thuộc vào cả loại sự kiện và về trạng thái nội bộ của hệ thống và có thể bao gồm sự thay đổi trạng thái dẫn đến sự chuyển đổi trạng thái. Mô hình các sự kiện, trạng thái và trạng thái chuyển tiếp giữa các trạng thái này có thể được trừu tượng và biểu diễn dưới dạng một máy trạng thái hữu hạn (FSM).

Khái niệm FSM rất quan trọng trong việc lập trình dựa trên sự kiện bởi vì nó làm cho việc xử lý sự kiện phụ thuộc một cách rõ ràng vào cả kiểu sự kiện và trạng thái của hệ thống. Khi sử dụng đúng, một máy trạng thái có thể cắt giảm đáng kể số lượng đường thực hiện thông qua mã, đơn giản hóa các điều kiện được kiểm tra tại mỗi điểm phân nhánh và đơn giản hóa việc chuyển đổi giữa các phương thức thực hiện khác nhau. Ngược lại, sử dụng lập trình hướng sự kiện mà không có mô hình FSM cơ bản có thể dẫn các lập trình viên tạo ra mã ứng dụng phức tạp dễ bị lỗi, khó mở rộng và quá phức tạp.

Sơ đồ trạng thái UML cơ bản [sửa]

UML giữ nguyên hình thức tổng quát của sơ đồ trạng thái truyền thống. Các sơ đồ trạng thái UML là đồ thị trực tiếp, trong đó các nút cho biết trạng thái và các kết nối biểu thị sự chuyển tiếp trạng thái. Ví dụ, Hình 1 cho thấy một biểu đồ trạng thái UML tương ứng với máy tính của máy tính trạng thái. Trong UML, các trạng thái được biểu diễn dưới dạng hình chữ nhật tròn được gắn nhãn với tên trạng thái. Các chuyển tiếp, được biểu diễn dưới dạng các mũi tên, được gắn nhãn với các sự kiện kích hoạt được thực hiện theo tùy chọn bởi danh sách các hành động được thực hiện. Quá trình chuyển tiếp ban đầu bắt nguồn từ vòng tròn rắn và chỉ định trạng thái mặc định khi hệ thống bắt đầu. Mỗi sơ đồ trạng thái phải có quá trình chuyển đổi như vậy, không được gắn nhãn vì nó không được kích hoạt bởi một sự kiện. Quá trình chuyển đổi ban đầu có thể có các hành động liên quan.

Sự kiện [sửa]

Sự kiện là một sự việc xảy ra mà ảnh hưởng đến hệ thống. Nói một cách nghiêm túc, trong đặc tả UML, [1], thuật ngữ sự kiện liên quan đến loại xuất hiện chứ không phải là bất kỳ trường hợp cụ thể nào xảy ra. Ví dụ: Kéo phím là sự kiện cho bàn phím, nhưng mỗi lần nhấn phím không phải là một sự kiện mà là một ví dụ cụ thể của sự kiện Keystroke. Một sự kiện quan tâm khác cho bàn phím có thể là Bật nguồn, nhưng bật nguồn vào ngày mai lúc 10:05:36 sẽ chỉ là một ví dụ của sự kiện Bật nguồn điện.

Một sự kiện có thể có các tham số liên quan, cho phép ví dụ sự kiện truyền đạt không chỉ sự xuất hiện của một số sự cố thú vị mà còn thông tin định lượng về sự xuất hiện đó. Ví dụ, sự kiện Keystroke được tạo ra bằng cách nhấn một phím trên bàn phím máy tính đã liên kết các thông số truyền mã quét ký tự cũng như trạng thái của các phím Shift, Ctrl và Alt.

Một thể hiện sự kiện tồn tại lâu hơn cho sự xuất hiện tức thời đã tạo ra nó và có thể truyền đạt sự xuất hiện này cho một hoặc nhiều máy trạng thái. Sau khi được tạo, ví dụ sự kiện đi qua một chu trình chế biến có thể bao gồm tối đa ba giai đoạn. Thứ nhất, trường hợp sự kiện được nhận khi nó được chấp nhận và đang chờ xử lý (ví dụ, nó được đặt trên hàng đợi sự kiện). Sau đó, trường hợp sự kiện được gửi đến máy trạng thái, tại thời điểm đó nó trở thành sự kiện hiện tại. Cuối cùng, nó được tiêu thụ khi máy trạng thái kết thúc xử lý sự kiện sự kiện. Trường hợp sự kiện đã qua sử dụng không còn để xử lý.

Các quốc gia [sửa]

Một nhà nước nắm bắt các khía cạnh có liên quan của lịch sử của hệ thống rất hiệu quả. Ví dụ: khi bạn nhấn phím trên bàn phím, mã ký tự được tạo ra sẽ là chữ hoa hoặc chữ thường, tùy thuộc vào việc Caps Lock đang hoạt động. Do đó, hành vi của bàn phím có thể được chia thành hai trạng thái: trạng thái "mặc định" và trạng thái "caps\_locked". (Hầu hết các bàn phím bao gồm một đèn LED chỉ ra rằng bàn phím nằm ở trạng thái "caps\_locked"). Hành vi của bàn phím chỉ phụ thuộc vào các khía cạnh nhất định của lịch sử, cụ thể là liệu phím Caps Lock đã được nhấn hay không, Về bao nhiêu và chính xác những phím khác đã được ép trước đó. Một tiểu bang có thể tóm tắt tất cả các chuỗi sự kiện có thể (nhưng không liên quan) và chỉ nắm bắt được những cái có liên quan.

Để liên kết khái niệm này với lập trình, điều này có nghĩa là thay vì ghi lại lịch sử sự kiện trong vô số các biến, cờ, và logic phức tạp, bạn chỉ dựa vào một biến trạng thái chỉ có thể giả định một số giới hạn các giá trị xác định trước (ví dụ: , Hai giá trị trong trường hợp của bàn phím). Giá trị của biến nhà nước rất rõ ràng xác định trạng thái hiện tại của hệ thống tại bất kỳ thời điểm nào. Khái niệm nhà nước làm giảm vấn đề xác định bối cảnh thực hiện trong đoạn mã để thử nghiệm chỉ là biến nhà nước thay vì nhiều biến, do đó loại bỏ rất nhiều logic điều kiện.

Trạng thái mở rộng [sửa]

Một giải thích có thể của nhà nước cho các hệ thống phần mềm là mỗi tiểu bang đại diện cho một bộ khác nhau của các giá trị hợp lệ của toàn bộ bộ nhớ chương trình. Ngay cả đối với các chương trình đơn giản chỉ với một vài biến số cơ bản, sự giải thích này dẫn đến một số lượng thiên văn của các trạng thái. Ví dụ: một số nguyên 32-bit có thể đóng góp cho hơn 4 tỷ tiểu bang khác nhau. Rõ ràng, cách giải thích này không thực tế, vì vậy các biến chương trình thường bị tách rời khỏi các trạng thái. Thay vào đó, điều kiện hoàn chỉnh của hệ thống (gọi là trạng thái mở rộng) là sự kết hợp của khía cạnh định tính (trạng thái) và các khía cạnh định lượng (các biến trạng thái mở rộng). Trong cách diễn giải này, sự thay đổi của biến không phải lúc nào cũng hàm ý sự thay đổi các khía cạnh định lượng của hành vi hệ thống và do đó không dẫn đến sự thay đổi trạng thái.

Các máy trạng thái được bổ sung bởi các biến được gọi là các máy trạng thái mở rộng và các máy trạng thái UML thuộc thể loại này. Các máy trạng thái mở rộng có thể áp dụng chủ nghĩa hình thức cơ bản cho những vấn đề phức tạp hơn là thực tế mà không bao gồm các biến trạng thái mở rộng. Ví dụ, giả sử hành vi của bàn phím phụ thuộc vào số ký tự gõ vào nó cho đến nay và sau, nói rằng, 1.000 phím bấm, bàn phím chia nhỏ và đi vào trạng thái cuối cùng. Để mô hình hóa hành vi này trong một máy trạng thái không có bộ nhớ, bạn cần phải giới thiệu 1000 trạng thái (ví dụ, nhấn phím trong trạng thái đột qu12123 sẽ dẫn đến đột qu state bang124, v.v.), điều này rõ ràng là một đề xuất không khả thi. Ngoài ra, bạn có thể xây dựng một máy nhà nước mở rộng với một biến số truy cập xuống key\_count. Các truy cập sẽ được khởi tạo đến 1.000 và decremented của mỗi keystroke mà không thay đổi trạng thái. Khi bộ đếm đạt đến mức không, máy trạng thái sẽ nhập trạng thái cuối cùng.

Biểu đồ trạng thái từ hình 2 là một ví dụ về một máy trạng thái mở rộng, trong đó trạng thái hoàn chỉnh của hệ thống (gọi là trạng thái mở rộng) là sự kết hợp của khía cạnh định tính - "trạng thái" - và khía cạnh định lượng - trạng thái mở rộng Biến (chẳng hạn như key\_count xuống bàn phím).

Lợi thế rõ ràng của các máy nhà nước mở rộng là tính linh hoạt. Ví dụ, kéo dài tuổi thọ của "bàn phím rẻ" từ 1.000 đến 10.000 lần bấm phím sẽ không làm phức tạp thêm máy trạng thái mở rộng. Yêu cầu sửa đổi duy nhất sẽ làm thay đổi giá trị khởi tạo của key\_count-down-counter trong bước chuyển tiếp ban đầu.

Tính linh hoạt của các máy gia tốc mở rộng này đi kèm với một mức giá, bởi vì sự kết hợp phức tạp giữa các khía cạnh "định tính" và "định lượng" của nhà nước mở rộng. Sự ghép nối xảy ra thông qua các điều kiện bảo vệ gắn liền với quá trình chuyển đổi, như thể hiện trong hình 2.

Điều kiện bảo vệ [sửa]

Các điều kiện bảo vệ (hoặc đơn giản chỉ là bảo vệ) là các biểu thức Boolean được đánh giá dựa trên giá trị của các biến trạng thái mở rộng và các tham số sự kiện. Các điều kiện bảo vệ ảnh hưởng đến hoạt động của máy trạng thái bằng cách cho phép hành động hoặc quá trình chuyển tiếp chỉ khi họ đánh giá thành TRUE và vô hiệu hóa chúng khi họ đánh giá FALSE. Trong ký pháp UML, điều kiện bảo vệ được hiển thị bằng dấu ngoặc vuông (ví dụ [key\_count == 0] trong Hình 2).

Sự cần thiết cho bảo vệ là hậu quả trực tiếp của việc thêm các biến trạng thái mở rộng bộ nhớ vào chế độ máy chủ của bang. Được sử dụng ít ỏi, các biến trạng thái mở rộng và bảo vệ tạo nên một cơ chế mạnh mẽ có thể đơn giản hóa thiết kế. Nhưng đừng để tên ưa thích ("guard") và ký hiệu UML súc tích đánh lừa bạn. Khi bạn thực sự mã một máy nhà nước mở rộng, các vệ sĩ trở thành cùng IFs và ELSEs mà bạn muốn loại bỏ bằng cách sử dụng máy nhà nước ở nơi đầu tiên. Quá nhiều người trong số họ, và bạn sẽ thấy mình trở lại trong một hình vuông ("spaghetti mã"), nơi mà các vệ sĩ có hiệu quả vượt qua xử lý của tất cả các điều kiện có liên quan trong hệ thống.

Thật vậy, lạm dụng các biến nhà nước mở rộng và bảo vệ là cơ chế chính của sự phân rã kiến ​​trúc trong thiết kế dựa trên các máy nhà nước. Thông thường, trong cuộc chiến hàng ngày, có vẻ như rất hấp dẫn, đặc biệt đối với các lập trình viên mới về chế độ máy chủ của nhà nước, thêm một biến thể mở rộng khác và một điều kiện bảo vệ khác (IF khác hoặc ELSE) chứ không phải là yếu tố Liên quan đến hành vi vào một khía cạnh định tính mới của hệ thống - nhà nước. Từ kinh nghiệm trong các chiến hào, khả năng phân rã kiến ​​trúc như vậy là tỷ lệ thuận với việc tạo ra hoặc xóa các trạng thái (liên quan đến chiến lược thực tế được sử dụng để thực hiện các máy trạng thái UML).

Một trong những thách thức chính trong việc trở thành một nhà thiết kế máy nhà nước hiệu quả là phát triển một ý thức mà các phần của hành vi nên được nắm bắt là các khía cạnh "định tính" ("nhà nước") và các yếu tố nào được để lại tốt hơn là khía cạnh "định lượng" (Các biến trạng thái mở rộng). Nói chung, bạn nên chủ động tìm kiếm các cơ hội để nắm bắt lịch sử sự kiện (cái đã xảy ra) như là "trạng thái" của hệ thống, thay vì lưu trữ thông tin này trong các biến trạng thái mở rộng. Ví dụ, một máy trạng thái thể hiện hành vi của một máy tính bỏ túi có thể sử dụng một biến trạng thái mở rộng DecimalFlag để nhớ rằng người dùng nhập vào dấu thập phân để tránh nhập nhiều số thập phân trong cùng một số. Tuy nhiên, một giải pháp tốt hơn là quan sát rằng nhập một dấu thập phân thực sự dẫn đến một trạng thái khác biệt "enter\_the\_fractional\_part\_of\_a\_number", trong đó máy tính bỏ qua các điểm thập phân. Giải pháp này là cấp trên vì một số lý do. Lý do nhỏ hơn là nó loại bỏ một biến trạng thái mở rộng và sự cần thiết để khởi tạo và kiểm tra nó. Lý do quan trọng hơn là giải pháp dựa trên nhà nước mạnh mẽ hơn bởi vì thông tin ngữ cảnh được sử dụng rất cục bộ (chỉ ở trạng thái đặc biệt này) và bị loại bỏ ngay khi nó trở nên không liên quan. Một khi số được nhập chính xác, nó không thực sự quan trọng cho các hoạt động tiếp theo của máy tính cho dù số đó có một điểm thập phân. Máy trạng thái chuyển sang trạng thái khác và tự động "quên" bối cảnh trước. Biến nhà nước mở rộng DecimalFlag, mặt khác, "đặt xung quanh" quá thời gian thông tin trở nên không liên quan (và có lẽ đã lỗi thời!). Tệ hơn nữa, bạn không được quên đặt lại DecimalFlag trước khi nhập một số khác hoặc cờ sẽ không chính xác cho thấy thực tế người dùng đã nhập điểm thập phân, nhưng có lẽ điều này xảy ra trong ngữ cảnh của số trước đó.

Giữ hành vi như là "trạng thái" có định lượng cũng có những bất lợi và hạn chế của nó. Thứ nhất, trạng thái và tô pô chuyển đổi trong một máy trạng thái phải tĩnh và cố định ở thời gian biên dịch, có thể quá hạn chế và không linh hoạt. Chắc chắn, bạn có thể dễ dàng tạo ra "các máy trạng thái" có thể sửa đổi chúng trong thời gian chạy (điều này thường xảy ra khi bạn cố gắng "mã spaghetti" như một máy trạng thái). Tuy nhiên, điều này giống như viết mã tự sửa đổi, thực sự đã được thực hiện trong những ngày đầu của chương trình nhưng đã nhanh chóng bị bác bỏ như một ý tưởng tồi tệ. Do đó, "nhà nước" chỉ có thể nắm bắt các khía cạnh tĩnh của hành vi được biết đến một cách ưu tiên và không có khả năng thay đổi trong tương lai.

Ví dụ, tốt để nắm bắt mục nhập của một điểm thập phân trong máy tính như là một trạng thái riêng biệt "enter\_the\_fractional\_part\_of\_a\_number", bởi vì một số có thể chỉ có một phần phân đoạn, mà cả hai đều được biết đến một priori và không có khả năng thay đổi trong tương lai. Tuy nhiên, việc thực hiện "bàn phím rẻ tiền" mà không có các biến trạng thái mở rộng và các điều kiện bảo vệ thì thực tế là không thể. Ví dụ này chỉ ra điểm yếu chính của "trạng thái" định lượng mà chỉ đơn giản không thể lưu trữ quá nhiều thông tin (chẳng hạn như phạm vi rộng các thao tác gõ phím). Các biến trạng thái mở rộng và các bảo vệ là một cơ chế để thêm tính linh hoạt thời gian chạy thêm vào các máy trạng thái. [7]

Hành động và chuyển tiếp [sửa]

Khi một thể hiện sự kiện được gửi, máy trạng thái đáp ứng bằng cách thực hiện các hành động, chẳng hạn như thay đổi một biến, thực hiện I / O, gọi một hàm, tạo ra một sự kiện khác, hoặc chuyển sang trạng thái khác. Bất kỳ giá trị tham số nào liên quan đến sự kiện hiện tại đều có sẵn cho tất cả các hành động trực tiếp gây ra bởi sự kiện đó.

Chuyển đổi từ trạng thái này sang trạng thái khác được gọi là sự chuyển đổi trạng thái, và sự kiện gây ra nó được gọi là sự kiện kích hoạt, hoặc chỉ đơn giản kích hoạt. Trong ví dụ về bàn phím, nếu bàn phím ở trạng thái "mặc định" khi phím CapsLock được nhấn, bàn phím sẽ nhập trạng thái "caps\_locked". Tuy nhiên, nếu bàn phím đã ở trạng thái "caps\_locked", nhấn CapsLock sẽ gây ra một quá trình chuyển đổi khác - từ "caps\_locked" sang trạng thái "mặc định". Trong cả hai trường hợp, nhấn CapsLock là sự kiện kích hoạt.

Trong các máy trạng thái mở rộng, quá trình chuyển đổi có thể có một bảo vệ, nghĩa là quá trình chuyển đổi chỉ có thể "cháy" nếu người bảo vệ đánh giá TRUE. Một nhà nước có thể có nhiều chuyển tiếp để đáp ứng với cùng một kích hoạt, miễn là họ có bảo vệ không trật tự; Tuy nhiên, tình trạng này có thể tạo ra các vấn đề trong chuỗi đánh giá của các vệ sĩ khi kích hoạt chung xảy ra. Đặc tả UML [1] cố ý không quy định bất kỳ thứ tự cụ thể nào; Thay vào đó, UML đặt gánh nặng cho người thiết kế để tạo ra những người bảo vệ theo cách mà trình tự đánh giá của họ không quan trọng. Thực tế, điều này có nghĩa là các biểu hiện bảo vệ không nên có tác dụng phụ, ít nhất là không có gì có thể làm thay đổi đánh giá của các vệ sĩ khác có cùng kích hoạt.

Mô hình thực hiện chạy hoàn thành [sửa]

Tất cả các máy tính của nhà nước, kể cả máy UML, đều giả định rằng một máy của nhà nước hoàn thành quá trình xử lý của mỗi sự kiện trước khi nó có thể bắt đầu xử lý sự kiện tiếp theo. Mô hình thực hiện này được gọi là chạy để hoàn thành, hoặc RTC.

Trong mô hình RTC, hệ thống xử lý sự kiện trong các bước RTC rời rạc, không thể phân chia. Các sự kiện mới đến không thể gián đoạn quá trình xử lý sự kiện hiện tại và phải được lưu trữ (thường trong hàng đợi sự kiện) cho đến khi máy trạng thái trở lại không hoạt động. Những ngữ nghĩa này hoàn toàn tránh bất kỳ vấn đề đồng thời nội bộ nào trong một máy trạng thái. Mô hình RTC cũng xung quanh vấn đề khái niệm về các hành động xử lý liên quan đến quá trình chuyển đổi, nơi mà máy trạng thái không ở trong trạng thái được xác định rõ (giữa hai trạng thái) trong suốt thời gian hành động. Trong quá trình xử lý sự kiện, hệ thống không phản hồi (không được theo dõi), do đó trạng thái không được xác định trong thời gian đó không có ý nghĩa thiết thực.

Tuy nhiên, lưu ý rằng RTC không có nghĩa là một máy trạng thái phải độc quyền CPU cho đến khi bước RTC hoàn thành. [1] Giới hạn quyền ưu tiên chỉ áp dụng cho bối cảnh nhiệm vụ của máy trạng thái đã bận xử lý các sự kiện. Trong một môi trường đa nhiệm, các tác vụ khác (không liên quan đến ngữ cảnh công việc của máy trạng thái bận) có thể được chạy, có thể là preempting máy đang thực hiện hiện tại. Miễn là các máy nhà nước khác không chia sẻ các biến hoặc các tài nguyên khác với nhau, không có nguy cơ đồng thời.

Ưu điểm chính của chế biến RTC là tính đơn giản. Bất lợi lớn nhất của nó là đáp ứng của một máy nhà nước được xác định bởi bước RTC dài nhất của nó. Đạt được các bước RTC ngắn có thể làm phức tạp nhiều thiết kế thời gian thực.

UML mở rộng cho các hình thức FSM truyền thống [sửa]

Mặc dù các FSM truyền thống là một công cụ tuyệt vời để giải quyết các vấn đề nhỏ hơn, nhưng cũng thường được biết rằng chúng có xu hướng trở nên không thể quản lý được, thậm chí đối với các hệ thống có liên quan đến vừa phải. Do hiện tượng được biết đến như là sự bùng nổ của nhà nước và quá trình chuyển đổi, sự phức tạp của FSM truyền thống có xu hướng phát triển nhanh hơn so với sự phức tạp của hệ thống mà nó mô tả. Điều này xảy ra bởi vì chế độ chính thức của máy nhà nước truyền thống gây ra các lần lặp lại. Ví dụ: nếu bạn cố gắng thể hiện hành vi của một máy tính bỏ túi đơn giản bằng FSM truyền thống, bạn sẽ nhận thấy ngay rằng nhiều sự kiện (ví dụ: Xóa hoặc Tắt) đã được xử lý giống nhau ở nhiều tiểu bang. Một FSM thông thường thể hiện trong hình dưới đây, không có cách nào để nắm bắt được tính phổ biến như vậy và yêu cầu lặp lại các hành động và sự chuyển tiếp tương tự ở nhiều tiểu bang. Những gì còn thiếu trong các máy nhà nước truyền thống là cơ chế để phân loại hành vi chung để chia sẻ nó trên nhiều tiểu bang.

Các máy trạng thái UML xác định chính xác sự thiếu sót này của FSM thông thường. Chúng cung cấp một số tính năng để loại bỏ các lần lặp lại để sự phức tạp của một máy trạng thái UML không còn phát nổ nhưng có xu hướng trung thành đại diện cho sự phức tạp của hệ thống phản ứng nó mô tả. Rõ ràng, các tính năng này là rất thú vị cho các nhà phát triển phần mềm, bởi vì chỉ có họ làm cho toàn bộ máy nhà nước tiếp cận thực sự áp dụng cho các vấn đề cuộc sống thực.

Các trạng thái lồng nhau theo bậc (Hierarchically nested states) [sửa]

Sự đổi mới quan trọng nhất của các máy trạng thái UML trên FSM truyền thống là việc giới thiệu các trạng thái lồng nhau theo cấp bậc (đó là lý do tại sao các biểu đồ trạng thái được gọi là các máy trạng thái phân cấp hoặc HSM). Các ngữ nghĩa kết hợp với việc làm tổ nhà nước như sau (xem hình 3): Nếu một hệ thống ở trạng thái lồng nhau, ví dụ như "result" (gọi là substate), nó cũng (mặc nhiên) là ở trạng thái "on" xung quanh (được gọi là Superstate). Máy trạng thái này sẽ cố gắng để xử lý bất kỳ sự kiện trong bối cảnh của substate, mà khái niệm là ở cấp thấp hơn của hệ thống phân cấp. Tuy nhiên, nếu "kết quả" thay thế không quy định cách xử lý sự kiện, sự kiện này không bị loại bỏ một cách lặng lẽ như trong một máy trạng thái "phẳng" truyền thống; Thay vào đó, nó được tự động xử lý ở ngữ cảnh cấp cao hơn của tiểu bang superstate "on". Đây là những gì có nghĩa là bởi hệ thống đang ở trong trạng thái "kết quả" cũng như "trên". Tất nhiên, việc làm tổ nhà nước không giới hạn ở một mức độ, và quy tắc xử lý sự kiện đơn giản áp dụng đệ quy cho bất kỳ mức độ làm tổ nào.

Hình 3: Máy tính bỏ túi (trái) và máy trạng thái UML với trạng thái nesting (bên phải)

Các quốc gia có các quốc gia khác được gọi là các quốc gia hỗn hợp; Ngược lại, các trạng thái không có cấu trúc bên trong được gọi là trạng thái đơn giản. Trạng thái lồng nhau được gọi là substate trực tiếp khi nó không được chứa bởi bất kỳ trạng thái nào khác; Nếu không, nó được gọi là một substate lồng nhau quá cảnh.

Bởi vì cấu trúc bên trong của một trạng thái phức tạp có thể phức tạp một cách tùy tiện, bất kỳ máy trạng thái phân cấp nào cũng có thể được xem như là một cấu trúc nội bộ của một số trạng thái phức hợp (cao cấp). Nó là khái niệm thuận tiện để xác định một nhà nước tổng hợp như là gốc rễ cuối cùng của hệ thống máy móc của nhà nước. Trong đặc tả UML, [1] mọi máy trạng thái có một trạng thái hàng đầu (gốc trừu tượng của mọi hệ thống phân cấp của máy nhà nước), chứa tất cả các phần tử khác của toàn bộ máy trạng thái. Biểu đồ đồ hoạ của trạng thái đầu trang bao quanh này là tùy chọn.

Như bạn thấy, ngữ nghĩa của phân rã trạng thái phân cấp được thiết kế để tạo điều kiện tái sử dụng hành vi. Các substates (các trạng thái lồng nhau) chỉ cần xác định sự khác biệt từ superstates (chứa các trạng thái). Một substate có thể dễ dàng kế thừa hành vi thông thường từ superstate (s) của nó bằng cách đơn giản bỏ qua các sự kiện xử lý thông thường, sau đó sẽ tự động được xử lý bởi các trạng thái cấp cao hơn. Nói cách khác, việc phân tổ trạng thái phân cấp cho phép lập trình bằng sự khác biệt [8].

Các khía cạnh của hệ thống phân cấp nhà nước nhấn mạnh thường nhất là trừu tượng - một kỹ thuật cũ và mạnh mẽ để đối phó với sự phức tạp. Thay vì giải quyết tất cả các khía cạnh của một hệ thống phức tạp cùng một lúc, thường có thể bỏ qua (tóm tắt đi) một số phần của hệ thống. Các trạng thái phân cấp là một cơ chế lý tưởng để ẩn các chi tiết nội bộ bởi vì người thiết kế có thể dễ dàng thu nhỏ hoặc phóng to để ẩn hoặc hiện các trạng thái lồng nhau.

Tuy nhiên, các trạng thái hỗn hợp không chỉ đơn giản là che dấu sự phức tạp; Họ cũng tích cực làm giảm nó thông qua cơ chế mạnh mẽ của xử lý sự kiện theo cấp bậc. Nếu không có tái sử dụng như vậy, thậm chí sự gia tăng vừa phải về tính phức tạp của hệ thống cũng có thể dẫn đến sự tăng lên của số lượng các trạng thái và sự chuyển tiếp. Ví dụ, máy trạng thái phân cấp đại diện cho máy tính bỏ túi (hình 3) tránh lặp lại quá trình chuyển đổi Clear and Off ở hầu như mọi trạng thái. Tránh lặp lại cho phép sự phát triển của HSMs để duy trì tỷ lệ thuận với tăng trưởng phức tạp của hệ thống. Khi hệ thống mô hình phát triển, cơ hội để tái sử dụng cũng tăng lên và do đó có khả năng chống lại sự gia tăng không cân xứng về số lượng các trạng thái và sự chuyển tiếp điển hình của FSM truyền thống.

Các vùng trực giao [sửa]

Phân tích theo phân loại trạng thái phân rã có thể bao gồm việc áp dụng các hoạt động 'độc quyền-hoặc' cho bất kỳ nhà nước nhất định. Ví dụ, nếu một hệ thống nằm trong superstate "on" (hình 3), nó có thể là trường hợp nó nằm trong "substart\_1" substate HO ORC toán tử "operand2" HO ORC "substEntered" substate HO ORC kết quả " Substate. Điều này sẽ dẫn đến mô tả của "on" siêu phân vị như là một 'OR nhà nước'.

Biểu đồ trạng thái UML cũng giới thiệu sự phân rã và phân rã bổ sung. Sự phân rã như vậy có nghĩa là một nhà nước hỗn hợp có thể chứa hai hoặc nhiều vùng trực giao (các phương tiện trực giao tương thích và độc lập trong bối cảnh này) và rằng ở trong một trạng thái hỗn hợp đó phải ở trong tất cả các vùng trực giao của nó cùng một lúc [

Các khu vực trực giao giải quyết vấn đề thường xuyên của sự gia tăng tổ hợp số lượng tiểu bang khi hành vi của một hệ thống bị phân mảnh thành các phần hoạt động độc lập, đồng thời hoạt động. Ví dụ, ngoài bàn phím chính, bàn phím máy tính có bàn phím số độc lập. Từ cuộc thảo luận trước, nhớ lại hai trạng thái của bàn phím chính đã được xác định: "default" và "caps\_locked" (xem hình 1). Bàn phím số cũng có thể ở hai trạng thái - "số" và "mũi tên" - phụ thuộc vào việc Num Lock có hoạt động hay không. Toàn bộ không gian trạng thái của bàn phím trong sự phân hủy tiêu chuẩn là sản phẩm Cartesian của hai thành phần (bàn phím chính và bàn phím số) và bao gồm bốn trạng thái: "default-numbers", "default- arrows", "caps\_locked-numbers, "Và" caps\_locked-arrows. " Tuy nhiên, đây sẽ là một đại diện tự nhiên vì hành vi của bàn phím số không phụ thuộc vào trạng thái của bàn phím chính và ngược lại. Việc sử dụng các vùng trực giao cho phép loại bỏ các hành vi độc lập như là một sản phẩm Cartesian để tránh và, thay vào đó, để chúng tách rời, như thể hiện trong hình 4.

Hình 4: Hai vùng trực giao (bàn phím chính và bàn phím số) của bàn phím máy tính

Lưu ý rằng nếu các vùng trực giao hoàn toàn độc lập với nhau, phức tạp phức tạp của chúng chỉ đơn giản là phụ gia, có nghĩa là số lượng các trạng thái độc lập cần thiết để mô hình hệ thống chỉ đơn giản là tổng k + l + m + ..., trong đó k, L, m, ... biểu thị các số OR-các trạng thái trong mỗi vùng trực giao. Tuy nhiên, trường hợp chung của sự phụ thuộc lẫn nhau, mặt khác, kết quả trong phức tạp nhân, do đó, nói chung, số lượng các tiểu bang cần thiết là sản phẩm × k × × × x ....

Trong hầu hết các tình huống thực tế, các đơn vị trực giao sẽ chỉ xấp xỉ trực giao (tức là không thực sự độc lập). Do đó, biểu đồ trạng thái UML cung cấp một số cách để các vùng trực giao truyền thông và đồng bộ hóa các hành vi của chúng. Trong số các bộ cơ cấu (đôi khi phức tạp) phong phú, có lẽ tính năng quan trọng nhất là các trực giao có thể phối hợp các hành vi của chúng bằng cách gửi các sự kiện sự kiện với nhau.

Mặc dù các vùng trực giao dẫn đến tính độc lập của thực hiện (cho phép đồng thời nhiều hoặc ít hơn), đặc tả UML không yêu cầu phải thực hiện một luồng riêng biệt cho mỗi vùng trực giao (mặc dù điều này có thể được thực hiện nếu cần). Trong thực tế, thông thường, các vùng trực giao thực hiện trong cùng một sợi. [10] Đặc tả UML chỉ yêu cầu người thiết kế không dựa vào bất kỳ thứ tự cụ thể nào đối với trường hợp sự kiện được gửi đến các vùng trực giao có liên quan.

Hành động nhập cảnh và xuất cảnh [sửa]

Mỗi tiểu bang trong một biểu mẫu trạng thái UML có thể có hành động nhập tùy chọn, được thực hiện khi nhập cảnh vào một tiểu bang, cũng như các hành động xuất cảnh tùy chọn, được thực hiện khi xuất cảnh từ một tiểu bang. Các hành động nhập cảnh và xuất cảnh được kết hợp với các trạng thái chứ không phải sự chuyển tiếp. Bất kể trạng thái được nhập hay thoát ra như thế nào, tất cả các hành động nhập cảnh và xuất cảnh sẽ được thực hiện. Do đặc tính này, biểu đồ trạng thái hoạt động giống như máy Moore. Ký hiệu UML cho các hành động nhập cảnh và xuất cảnh của tiểu bang là đặt từ "nhập cảnh" (hoặc "thoát") ở trong tiểu bang ngay bên dưới ô tên, tiếp theo là dấu gạch chéo và danh sách các hành động tùy ý (xem Hình 5).

Giá trị của các hành động nhập cảnh và xuất cảnh là chúng cung cấp phương tiện để khởi tạo và dọn dẹp được bảo đảm, giống như các nhà xây dựng lớp và destructors trong lập trình hướng đối tượng. Ví dụ, hãy xem xét trạng thái "door\_open" từ Hình 5, tương ứng với hành vi lò nướng bánh mỳ trong khi cánh cửa mở. Trạng thái này có một yêu cầu rất quan trọng về an toàn: Luôn tắt lò sưởi khi cửa mở. Ngoài ra, trong khi cửa mở, đèn bên trong chiếu sáng lò sẽ sáng lên.

Tất nhiên, hành vi đó có thể được mô phỏng bằng cách thêm các hành động thích hợp (vô hiệu hóa nóng và bật ánh sáng) tới mọi đường dẫn chuyển tiếp đến trạng thái "door\_open" (người dùng có thể mở cửa bất cứ lúc nào trong quá trình "nướng" hoặc "nướng "Hoặc khi lò nướng không được sử dụng ở tất cả). Không được quên để dập tắt đèn nội bộ với mỗi quá trình chuyển đổi để lại trạng thái "door\_open". Tuy nhiên, một giải pháp như vậy sẽ gây ra sự lặp lại của hành động trong nhiều quá trình chuyển đổi. Quan trọng hơn, cách tiếp cận như vậy sẽ làm cho thiết kế dễ bị lỗi khi sửa đổi hành vi tiếp theo (ví dụ: lập trình viên tiếp theo làm việc trên một tính năng mới, chẳng hạn như top-browning, đơn giản là quên tắt bộ phận làm nóng khi chuyển sang "door\_open").

Các hành động nhập cảnh và xuất cảnh cho phép thực hiện hành vi mong muốn một cách an toàn, đơn giản và trực quan hơn. Như thể hiện trong hình 5, có thể xác định rằng hành động xuất cảnh từ "sưởi ấm" tắt lò sưởi, hành động nhập cảnh để "door\_open" sáng lên đèn lò, và hành động thoát khỏi "door\_open" dập tắt đèn. Việc sử dụng các hành động nhập cảnh và xuất cảnh là tốt hơn để đặt một hành động chuyển đổi vì nó tránh được mã hóa lặp đi lặp lại và cải thiện chức năng bằng cách loại bỏ một mối nguy an toàn; (Lò sưởi khi cửa mở). Các ngữ nghĩa của hành động thoát ra đảm bảo rằng, bất kể đường chuyển tiếp, lò sưởi sẽ bị tắt khi máy nướng không ở trong trạng thái "sưởi ấm".

Bởi vì các hành động nhập cảnh được thực hiện tự động bất cứ khi nào một nhà nước liên kết được nhập, họ thường xác định các điều kiện hoạt động hoặc bản sắc của nhà nước, rất nhiều như là một nhà xây dựng lớp xác định danh tính của đối tượng đang được xây dựng. Ví dụ, nhận dạng của trạng thái "sưởi ấm" được xác định bởi thực tế là lò sưởi được bật lên. Điều kiện này phải được thiết lập trước khi đi vào bất kỳ điểm thay thế nào của "sưởi ấm" bởi vì các hành động nhập cảnh đến một điểm thay thế của "sưởi ấm", như "nướng", dựa vào việc khởi tạo thích hợp của siêu cao su "nóng" và chỉ thực hiện những khác biệt từ khởi tạo này. Do đó, thứ tự thực hiện các hành động nhập cảnh phải luôn luôn tiến hành từ trạng thái ngoài cùng nhất sang trạng thái bên trong (từ trên xuống dưới).

Không đáng ngạc nhiên, thứ tự này là tương tự như thứ tự trong đó nhà xây dựng lớp được gọi. Việc xây dựng một lớp học luôn bắt đầu ở gốc của hệ thống phân lớp và đi qua tất cả các cấp thừa kế xuống lớp đang được khởi tạo. Việc thực hiện các hành động xuất cảnh, tương ứng với việc triệu hồi destructor, tiến hành theo thứ tự ngược lại (từ dưới lên).

Chuyển tiếp nội bộ [sửa]

Thông thường, một sự kiện chỉ gây ra một số hành động nội bộ để thực hiện nhưng không dẫn đến sự thay đổi trạng thái (chuyển đổi trạng thái). Trong trường hợp này, tất cả các hành động được thực hiện bao gồm chuyển đổi nội bộ. Ví dụ: khi gõ trên bàn phím, nó sẽ phản hồi bằng cách tạo các mã ký tự khác. Tuy nhiên, trừ khi phím Caps Lock được nhấn, trạng thái của bàn phím không thay đổi (không có trạng thái chuyển tiếp xảy ra). Trong UML, tình huống này nên được mô hình hoá bằng các chuyển tiếp nội bộ, như trong Hình 6. Ký hiệu UML cho quá trình chuyển đổi nội bộ tuân theo cú pháp chung được sử dụng cho hành động xuất cảnh (hoặc nhập cảnh), ngoại trừ thay vì nhập từ (hoặc thoát) Được gắn nhãn với sự kiện kích hoạt (ví dụ, xem quá trình chuyển đổi nội bộ được kích hoạt bởi sự kiện ANY\_KEY trong Hình 6).

Trong trường hợp không có hành động nhập cảnh và xuất cảnh, chuyển tiếp nội bộ sẽ giống hệt với chuyển tiếp tự (chuyển tiếp, trong đó trạng thái đích là giống như trạng thái nguồn). Trên thực tế, trong một máy Cổ điển, các hành động liên quan trực tiếp với quá trình chuyển đổi trạng thái, do đó, cách duy nhất để thực hiện các hành động mà không thay đổi trạng thái là thông qua quá trình tự chuyển đổi (mô tả như một vòng lặp trực tiếp trong Hình 1 từ đầu bài viết này). Tuy nhiên, với sự có mặt của các hành động nhập cảnh và xuất cảnh, như trong biểu đồ trạng thái UML, quá trình tự chuyển đổi liên quan đến việc thực hiện các hành động xuất cảnh và nhập cảnh và do đó khác biệt rõ rệt so với quá trình chuyển đổi nội bộ.

Ngược lại với quá trình tự chuyển đổi, không có hành động nhập cảnh hay xuất cảnh nào được thực hiện như là kết quả của quá trình chuyển đổi nội bộ, ngay cả khi quá trình chuyển đổi nội bộ được kế thừa từ cấp cao hơn của hệ thống phân cấp so với trạng thái đang hoạt động. Quá trình chuyển đổi nội bộ được thừa hưởng từ superstates ở bất kỳ mức nào của hành động làm tổ như thể chúng được xác định trực tiếp trong trạng thái hiện đang hoạt động.

Trình tự thực thi chuyển tiếp [sửa]

Việc làm tổ nhà nước kết hợp với các hành động nhập cảnh và xuất cảnh đã làm giảm đáng kể ngữ nghĩa chuyển đổi trạng thái trong HSMs so với FSM truyền thống. Khi giao dịch với các trạng thái lồng nhau được phân cấp và các vùng trực giao, trạng thái hiện tại của thuật ngữ đơn giản có thể khá khó hiểu. Trong HSM, nhiều trạng thái có thể hoạt động cùng một lúc. Nếu máy trạng thái nằm trong trạng thái lá chứa trong trạng thái hỗn hợp (có thể chứa trong trạng thái phức hợp bậc cao, v.v ...), tất cả các trạng thái hỗn hợp trực tiếp hay gián tiếp chứa trạng thái lá cũng hoạt động . Hơn nữa, bởi vì một số trạng thái hỗn hợp trong hệ thống phân cấp này có thể có các vùng trực giao, trạng thái hoạt động hiện tại thực sự được biểu diễn bởi một cây các trạng thái bắt đầu với trạng thái đỉnh duy nhất ở gốc tới các trạng thái đơn giản ở các lá. Đặc tả UML đề cập đến một cây trạng thái như là cấu hình của nhà nước. [1]

Trong UML, sự chuyển đổi trạng thái có thể kết nối trực tiếp hai trạng thái. Hai trạng thái này, có thể được tổng hợp, được chỉ định làm nguồn chính và mục tiêu chính của quá trình chuyển đổi. Hình 7 cho thấy một ví dụ chuyển đổi đơn giản và giải thích vai trò của nhà nước trong quá trình chuyển đổi đó. Đặc tả UML quy định rằng thực hiện chuyển đổi trạng thái liên quan đến việc thực hiện các hành động sau theo trình tự sau (xem Phần 15.3.14 trong Mô hình hoá thống nhất OMG (OMG UML), Phiên bản Cơ sở hạ tầng 2.2 [1]):

Đánh giá tình trạng bảo vệ liên quan đến quá trình chuyển đổi và chỉ thực hiện các bước sau nếu người bảo vệ đánh giá TRUE.

Thoát khỏi cấu hình nguồn gốc.

Thực hiện các hành động liên quan đến quá trình chuyển đổi.

Nhập cấu hình trạng thái đích.

Trình tự chuyển tiếp dễ dàng giải thích trong trường hợp đơn giản của cả nguồn chính và mục tiêu chính làm tổ ở cùng một cấp độ. Ví dụ, chuyển tiếp T1 thể hiện trong hình 7 gây ra việc đánh giá của guard g (); Tiếp theo là chuỗi các hành động: a (); B (); T (); C (); D (); Và e (); Giả sử rằng các guard g () đánh giá là TRUE.

Tuy nhiên, trong trường hợp chung của các trạng thái nguồn và trạng thái đích được lồng nhau ở các cấp độ khác nhau của hệ thống phân cấp của bang, có thể không rõ ràng ngay là cần bao nhiêu cấp độ làm tổ. Các đặc tả UML [1] quy định rằng một quá trình chuyển đổi liên quan đến việc thoát khỏi tất cả các trạng thái lồng nhau từ trạng thái đang hoạt động hiện tại (có thể là một substate trực tiếp hoặc chuyển tiếp của trạng thái nguồn chính) đến, nhưng không bao gồm, tổ tiên của tổ tiên ít nhất (LCA) Của nguồn chính và các quốc gia mục tiêu chính. Như tên cho thấy, LCA là trạng thái hỗn hợp thấp nhất đồng thời là siêu phân vị (tổ tiên) của cả nguồn và đích. Như được mô tả trước đây, thứ tự thực hiện các hành động xuất cảnh luôn từ trạng thái lồng nhau sâu nhất (trạng thái đang hoạt động) lên bậc lên tới LCA nhưng không thoát khỏi LCA. Ví dụ, LCA (s1, s2) của các trạng thái "s1" và "s2" thể hiện trong hình 7 là trạng thái "s".

Nhập cấu hình trạng thái đích bắt đầu từ cấp độ mà các hành động thoát ra trái (tức là từ bên trong LCA). Như đã mô tả trước đây, các hành động nhập cảnh phải được thực hiện bắt đầu từ trạng thái cấp cao nhất xuống hệ thống phân cấp của tiểu bang tới trạng thái đích chính. Nếu tiểu bang mục tiêu chính là hợp thành, ngữ nghĩa UML quy định để "khoan" vào tiểu sổ của nó đệ quy bằng cách sử dụng chuyển tiếp ban đầu địa phương. Cấu hình trạng thái đích được nhập hoàn toàn chỉ sau khi gặp trạng thái lá mà không có sự chuyển tiếp ban đầu.

Sự chuyển đổi nội bộ so với bên ngoài [sửa]

Trước khi UML 2, [1] chỉ sử dụng ngữ nghĩa chuyển tiếp duy nhất là quá trình chuyển đổi bên ngoài, trong đó nguồn chính của quá trình chuyển đổi luôn luôn xuất phát và mục tiêu chính của quá trình chuyển đổi luôn được nhập. UML 2 vẫn giữ nguyên ngữ nghĩa "chuyển tiếp bên ngoài" để tương thích ngược, nhưng cũng giới thiệu một kiểu chuyển tiếp mới được gọi là chuyển đổi cục bộ (xem Phần 15.3.15 trong Ngôn ngữ Mô hình Hợp nhất (UML), Phiên bản Cơ sở hạ tầng 2.2. Đối với nhiều cấu trúc liên kết chuyển đổi, chuyển tiếp bên ngoài và địa phương thực sự giống hệt nhau. Tuy nhiên, quá trình chuyển đổi cục bộ không gây ra sự thoát ra và quay trở lại trạng thái nguồn chính nếu trạng thái đích chính là nguồn thay thế của nguồn chính. Ngoài ra, sự chuyển đổi trạng thái của địa phương không gây ra và thoát khỏi trạng thái đích chính nếu mục tiêu chính là siêu cường của tiểu bang nguồn chính.

Hình 8 so sánh các chuyển tiếp địa phương (a) và bên ngoài (b). Trong hàng trên cùng, bạn sẽ thấy trường hợp của nguồn chính chứa mục tiêu chính. Quá trình chuyển đổi cục bộ không gây ra sự thoát khỏi nguồn, trong khi quá trình chuyển đổi bên ngoài làm cho thoát ra và quay trở lại nguồn. Trong hàng dưới cùng của hình 8, bạn sẽ thấy trường hợp mục tiêu chính có chứa nguồn chính. Quá trình chuyển đổi cục bộ không gây ra mục tiêu nhập cảnh, trong khi quá trình chuyển đổi bên ngoài gây ra và quay trở lại mục tiêu.

Sự trì hoãn sự kiện [sửa]

Đôi khi một sự kiện đến một thời điểm bất tiện đặc biệt, khi một máy trạng thái đang ở trong trạng thái không thể xử lý sự kiện. Trong nhiều trường hợp, bản chất của sự kiện có thể được trì hoãn (trong giới hạn) cho đến khi hệ thống đi vào một tiểu bang khác, trong đó nó được chuẩn bị tốt hơn để xử lý các sự kiện ban đầu.

Các máy trạng thái UML cung cấp một cơ chế đặc biệt để trì hoãn các sự kiện ở các bang. Ở mỗi tiểu bang, bạn có thể bao gồm một mệnh đề [danh sách sự kiện] / trì hoãn. Nếu sự kiện trong danh sách sự kiện hoãn lại của tiểu bang hiện tại xảy ra, sự kiện sẽ được lưu lại (deferred) để xử lý trong tương lai cho đến khi một tiểu bang được nhập mà không liệt kê sự kiện trong danh sách sự kiện hoãn lại của nó. Khi nhập cảnh vào trạng thái như vậy, máy trạng thái UML sẽ tự động nhớ lại các sự kiện đã lưu mà không còn hoãn lại và sau đó sẽ tiêu thụ hoặc hủy các sự kiện này. Có thể cho một superstate để có một sự chuyển tiếp được định nghĩa trên một sự kiện được trì hoãn bởi một substate. Phù hợp với các khu vực khác trong đặc tả của các máy trạng thái UML, các substate được ưu tiên hơn superstate, sự kiện sẽ được hoãn lại và quá trình chuyển đổi cho superstate sẽ không được thực hiện. Trong trường hợp các vùng trực giao nơi một vùng trực giao đảo ngược sự kiện và một địa điểm khác tiêu thụ sự kiện, người tiêu dùng sẽ được ưu tiên và sự kiện được tiêu thụ và không bị trì hoãn.

Các hạn chế của các máy trạng thái UML [sửa]

Các bảng trạng thái Harel, vốn là tiền thân của các máy UML, đã được phát minh là "một hình thức thị giác cho các hệ thống phức tạp", [2] nên từ khi thành lập, chúng đã được liên kết không thể tách rời với biểu diễn đồ họa dưới dạng sơ đồ trạng thái. Tuy nhiên, điều quan trọng là phải hiểu rằng khái niệm về máy trạng thái UML vượt qua bất kỳ ký hiệu đặc biệt nào, đồ hoạ hoặc văn bản. Các đặc điểm kỹ thuật UML [1] làm cho sự phân biệt này rõ ràng bằng cách tách rõ ngữ nghĩa máy nhà nước từ các ký hiệu.

Tuy nhiên, các ký hiệu của UML statecharts không phải là hoàn toàn thị giác. Bất kỳ máy trạng thái không cần thiết nào yêu cầu một lượng lớn thông tin văn bản (ví dụ, đặc tả hành động và bảo vệ). Cú pháp chính xác của hành động và biểu thức bảo vệ không được định nghĩa trong đặc tả UML, vì vậy nhiều người sử dụng cấu trúc tiếng Anh hoặc, chính thức hơn, các biểu thức trong một ngôn ngữ thực hiện như C, C ++ hoặc Java. [11] Trong thực tế, điều này có nghĩa là ký hiệu ký hiệu trạng thái UML phụ thuộc rất nhiều vào ngôn ngữ lập trình cụ thể.

Tuy nhiên, hầu hết các ngữ nghĩa statecharts đều có xu hướng nặng nề về ký hiệu đồ hoạ. Ví dụ, sơ đồ trạng thái kém trình bày các trình tự xử lý, có thể là thứ tự đánh giá của bảo vệ hoặc thứ tự của các sự kiện dispatching đến các vùng trực giao. Các đặc điểm kỹ thuật UML sidesteps những vấn đề này bằng cách đặt gánh nặng cho các nhà thiết kế không dựa vào bất kỳ trình tự cụ thể. Tuy nhiên, trong trường hợp máy UML thực sự được thực hiện, thì chắc chắn kiểm soát hoàn toàn trật tự thực hiện, dẫn đến sự chỉ trích rằng ngữ nghĩa UML có thể không cần thiết. Tương tự như vậy, sơ đồ trạng thái yêu cầu rất nhiều thiết bị đường ống dẫn nước (pseudostates, như nối, nĩa, nút nối, các điểm lựa chọn, vv) để đại diện cho dòng chảy của kiểm soát đồ họa. Nói cách khác, các phần tử của ký hiệu đồ hoạ không thêm nhiều giá trị trong đại diện cho dòng chảy kiểm soát so với mã có cấu trúc đơn giản.

Ký hiệu UML và ngữ nghĩa thực sự hướng tới các công cụ UML được máy tính hoá. Một máy trạng thái UML, như được biểu diễn trong một công cụ, không chỉ là sơ đồ trạng thái, mà là một sự kết hợp của biểu diễn đồ họa và văn bản, nó chính xác nắm bắt cả topo và các hành động của nhà nước. Những người dùng của công cụ có thể có được một số khung nhìn bổ sung của cùng một máy trạng thái, cả trực quan và văn bản, trong khi mã được tạo ra chỉ là một trong số nhiều khung nhìn khả dụng.