Nhập môn
Điều khiển thông minh

Chương 0

Một số khái niệm

Tự nhiên/nhân tạo

Ảo/thực

Khái niệm về điều khiển

Control (Điều khiển): là sức mạnh để ảnh hưởng hoặc chỉ đạo hành vi của mọi người hoặc quá trình của các thực tế.

Điều khiển trong Kinh tế và Thương mại

Điều khiển trong toán học và khoa học

Điều khiển trong kỹ thuật hệ thống, kỹ thuật máy tính và công nghệ

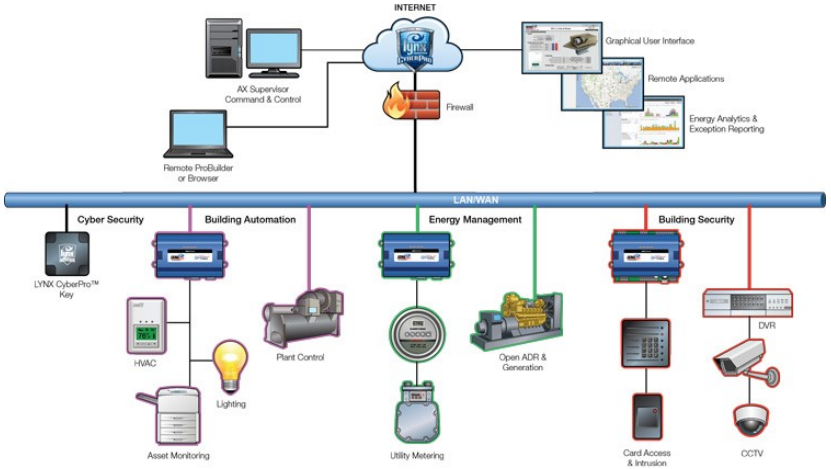
Điều khiển trong y học

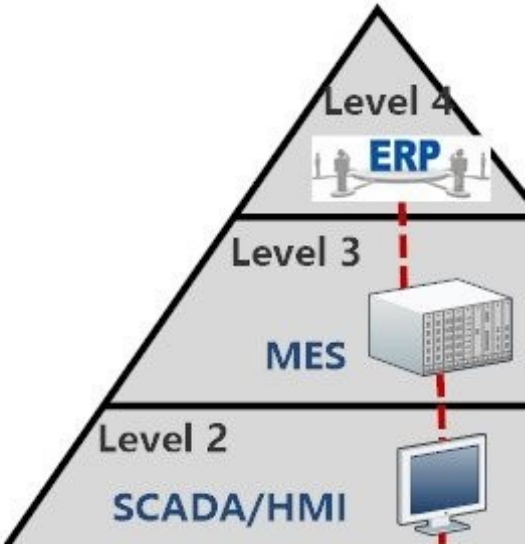
Điều khiển xã hội, tâm lý học và xã hội học

…

Khái niệm về điều khiển

Điều khiển tự động (Tự động hóa) là một loạt các công nghệ làm giảm sự can thiệp của con người vào các quy trình. Sự can thiệp của con người được giảm bớt bằng cách xác định trước các tiêu chí quyết định, mối quan hệ phụ và các hành động liên quan - và thể hiện những xác định trước đó trong máy móc.

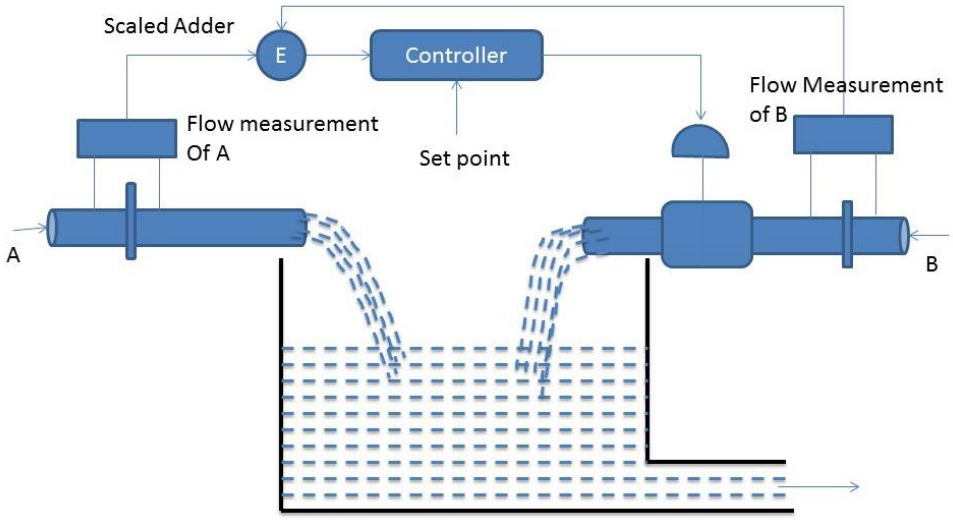




Khái niệm về điều khiển

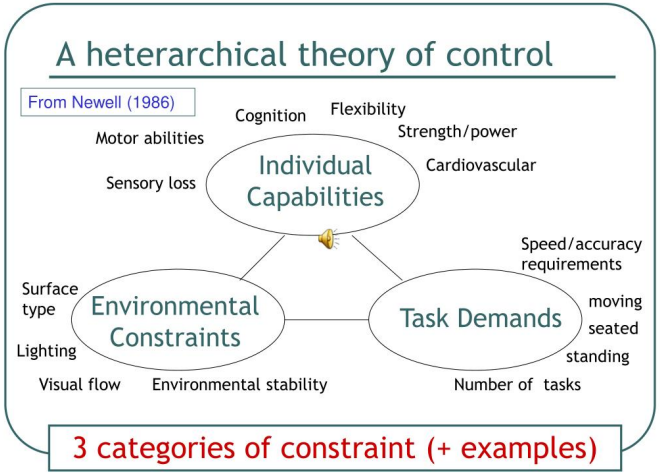
Kỹ thuật điều khiển (kỹ thuật hệ thống điều khiển) là một ngành kỹ thuật liên quan đến các hệ thống điều khiển, áp dụng lý thuyết điều khiển để thiết kế thiết bị và hệ thống với các hành vi mong muốn trong môi trường điều khiển.

Example of Ratio Control System



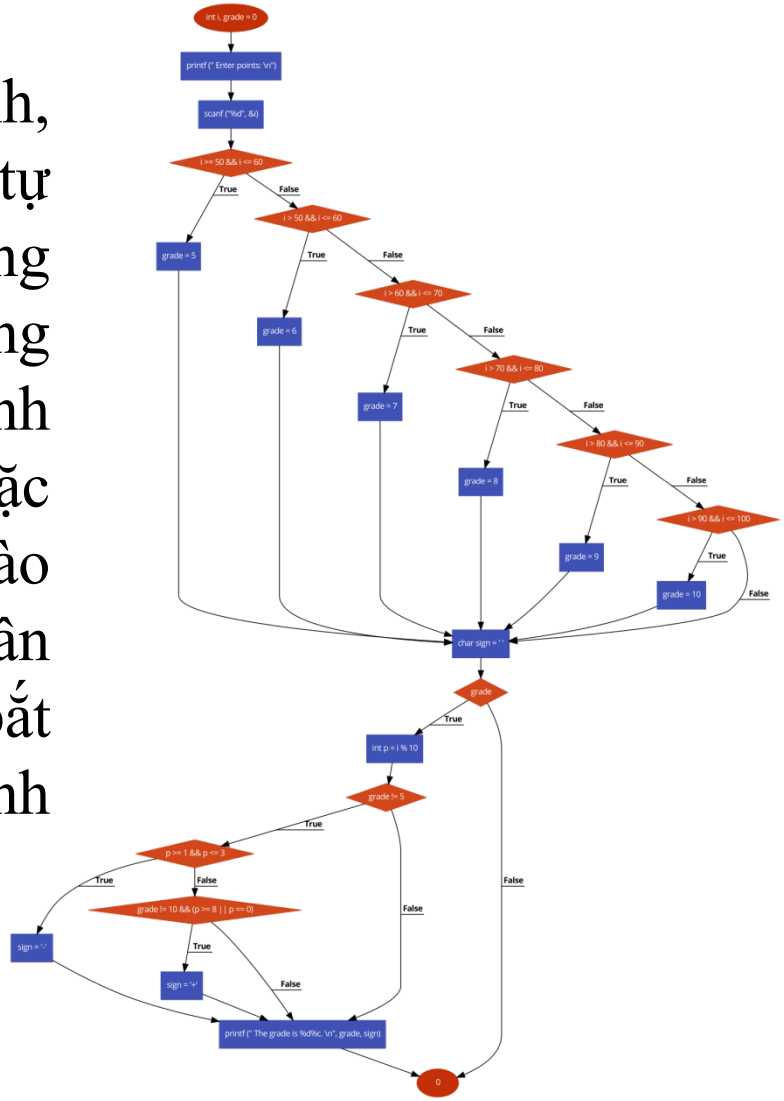
Khái niệm về điều khiển

Lý thuyết điều khiển liên quan đến việc kiểm soát các hệ thống động lực học trong các quy trình và máy móc được thiết kế. Mục tiêu là phát triển một mô hình hoặc thuật toán điều chỉnh việc áp dụng đầu vào hệ thống để đưa hệ thống đến trạng thái mong muốn ở đầu ra, đồng thời giảm thiểu bất kỳ sai lệch chậm trễ, vượt mức hoặc trạng thái ổn định nào và đảm bảo mức độ ổn định kiểm soát; Thường với mục đích đạt được mức độ tối ưu.



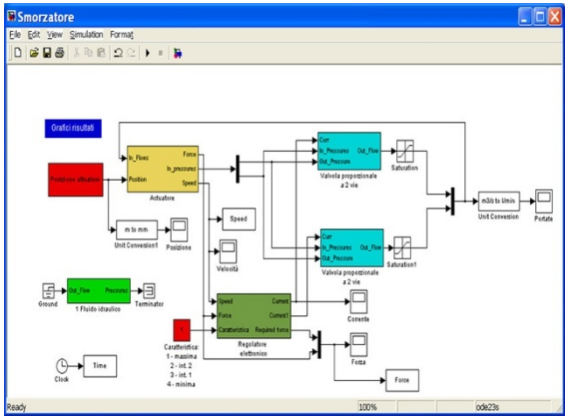
Khái niệm về điều khiển

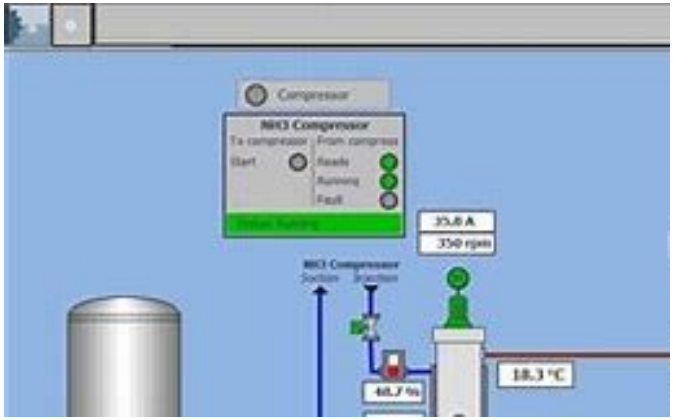
Trong khoa học máy tính, **luồng điều khiển** là thứ tự trong đó các câu lệnh, hướng dẫn hoặc cuộc gọi chức năng riêng lẻ của một chương trình bắt buộc được thực hiện hoặc đánh giá. Sự nhấn mạnh vào luồng kiểm soát rõ ràng phân biệt một ngôn ngữ lập trình bắt buộc với ngôn ngữ lập trình khai báo.



Khái niệm về điều khiển

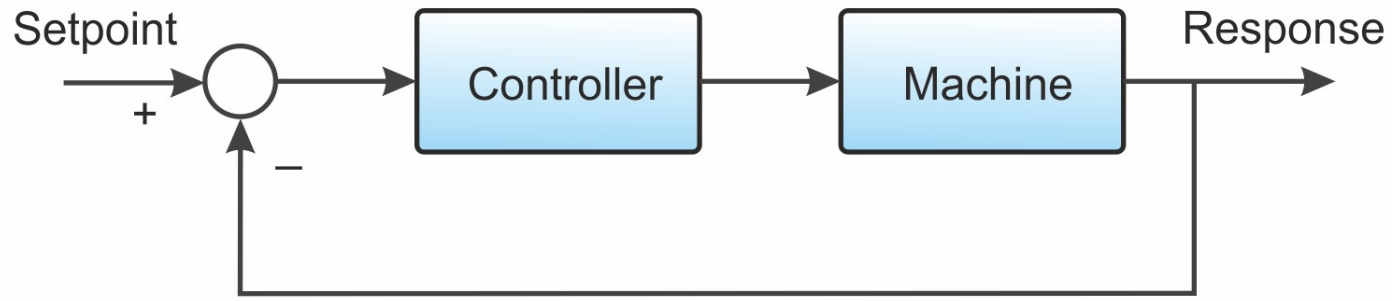
Điều khiển quá trình (công nghiệp) trong các quy trình sản xuất liên tục là một nguyên tắc sử dụng các hệ thống điều khiển công nghiệp để đạt được mức độ sản xuất nhất quán, kinh tế và an toàn mà không thể đạt được hoàn toàn bằng cách kiểm soát thủ công của con người. Nó được thực hiện rộng rãi trong các ngành công nghiệp như ô tô, khai thác mỏ, nạo vét, lọc dầu, sản xuất bột giấy và giấy, chế biến hóa chất và nhà máy phát điện, ...



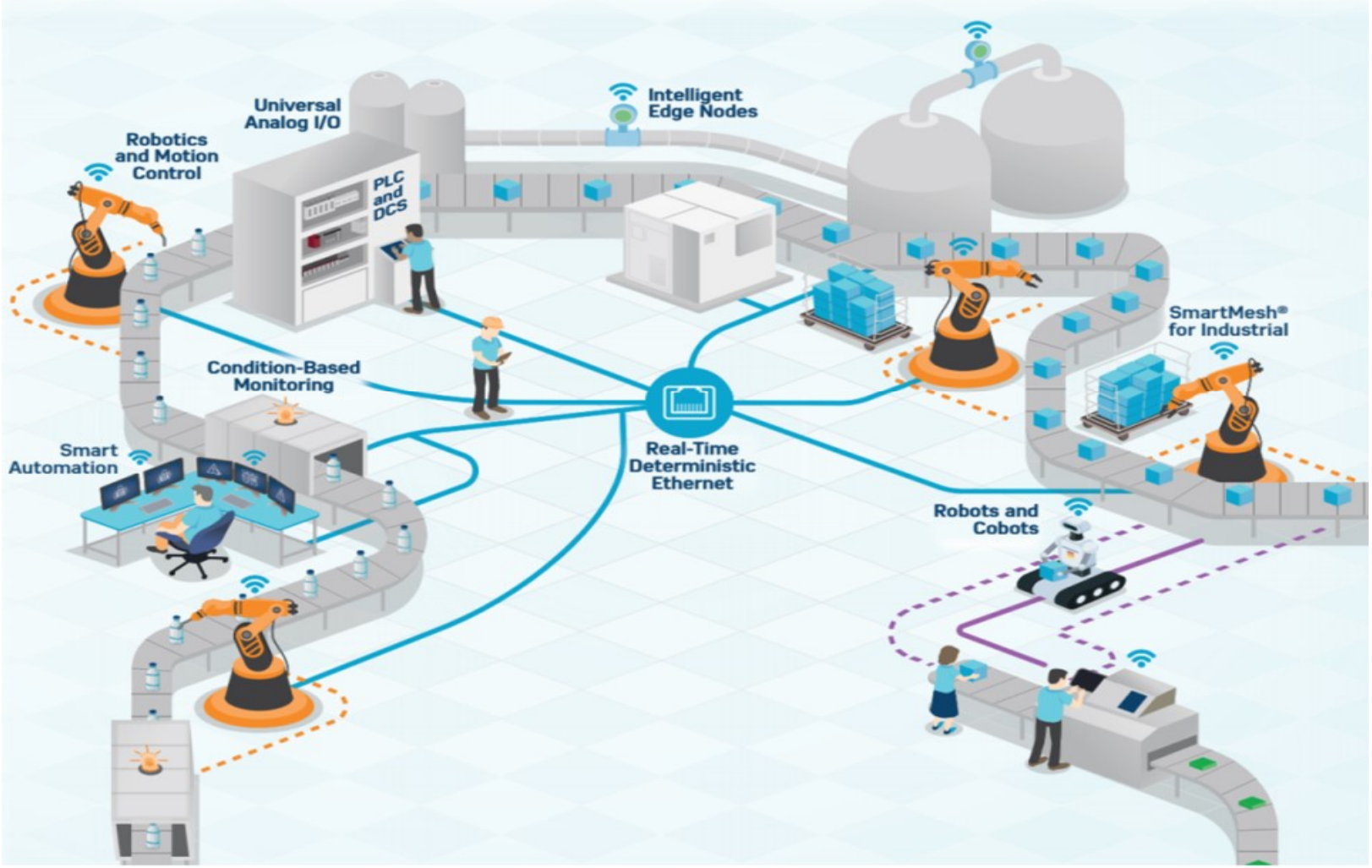


Hệ thống điều khiển

Một hệ thống điều khiển quản lý, ra lệnh, chỉ đạo hoặc điều chỉnh hành vi của các thiết bị hoặc hệ thống khác bằng cách sử dụng các vòng điều khiển. Nó có thể bao gồm từ một bộ điều khiển sưởi ấm căn hộ duy nhất bằng cách sử dụng bộ điều chỉnh nhiệt điều khiển nồi hơi trong nước đến các hệ thống điều khiển công nghiệp lớn được sử dụng để kiểm soát các quy trình hoặc máy móc.



Hệ thống điều khiển



Hệ thống điều khiển

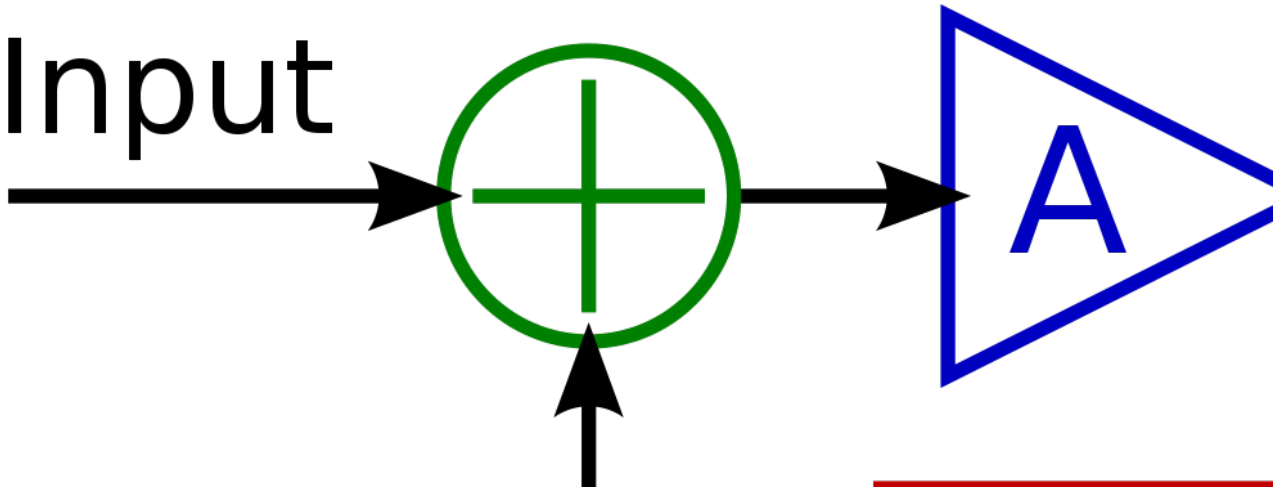
Vòng điều khiển: Có hai lớp hành động điều khiển phổ biến là vòng lặp mở và vòng lặp đóng.

Trong một hệ thống **điều khiển vòng mở**, hành động điều khiển từ bộ điều khiển là độc lập với biến quá trình.

Trong một hệ thống **điều khiển vòng kín**, hành động điều khiển từ bộ điều khiển phụ thuộc vào biến quy trình mong muốn và thực tế. Bộ điều khiển vòng kín có một vòng lặp phản hồi đảm bảo bộ điều khiển thực hiện hành động điều khiển để kiểm soát một biến quy trình có cùng giá trị với điểm đặt. Vì lý do này, bộ điều khiển vòng kín còn được gọi là bộ điều khiển phản hồi.

Hệ thống điều khiển

Điều khiển phản hồi: Trong trường hợp hệ thống phản hồi tuyến tính, một vòng điều khiển bao gồm cảm biến, thuật toán điều khiển và bộ truyền động được sắp xếp trong nỗ lực điều chỉnh một biến tại điểm đặt (SP).



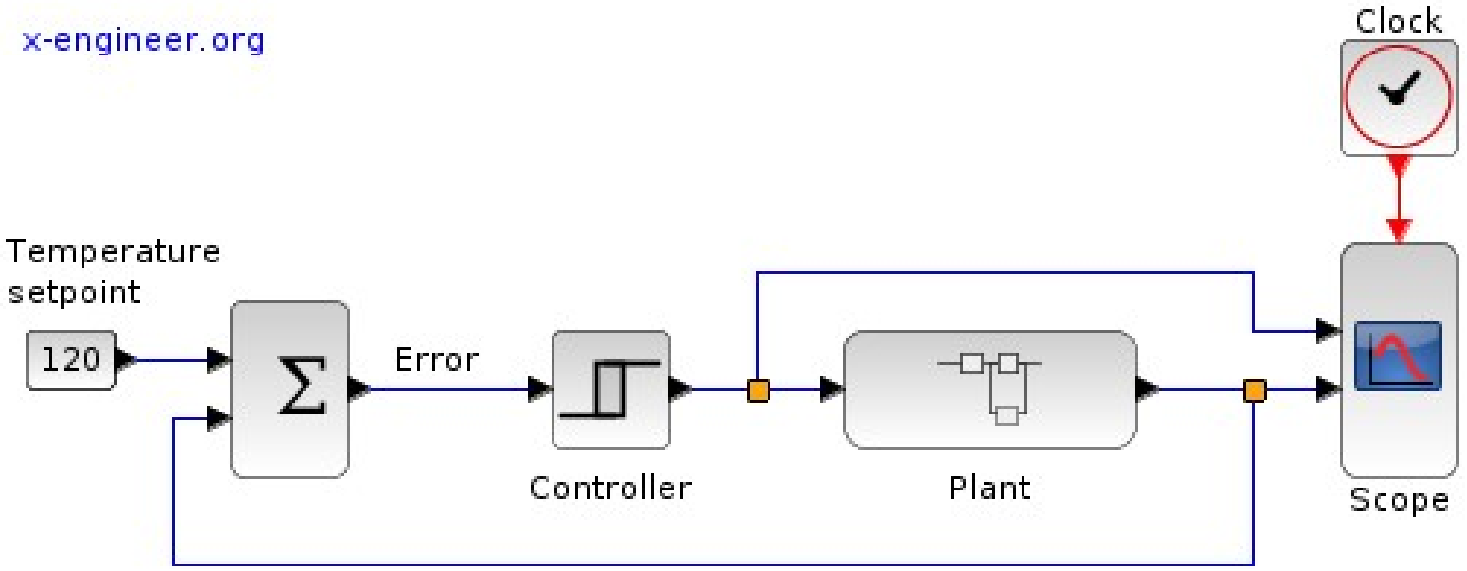
Hệ thống điều khiển

Điều khiển logic cho máy móc công nghiệp và thương mại được thực hiện qua các rơle điện liên kết với nhau và bộ hẹn giờ cam bằng cách sử dụng logic thang. Ngày nay, hầu hết các hệ thống như vậy được xây dựng với các vi điều khiển hoặc bộ điều khiển logic lập trình chuyên dụng hơn (PLCs).

Ngôn ngữ (ký hiệu) logic bậc thang vẫn được sử dụng như một phương pháp lập trình cho PLCs.

Hệ thống điều khiển

Điều khiển bật-tắt sử dụng bộ điều khiển phản hồi chuyển đổi đột ngột giữa hai trạng thái. Các hệ thống điều khiển bật tắt thường đơn giản về cấu trúc, có thể rẻ về giá thành mà vẫn hiệu quả.



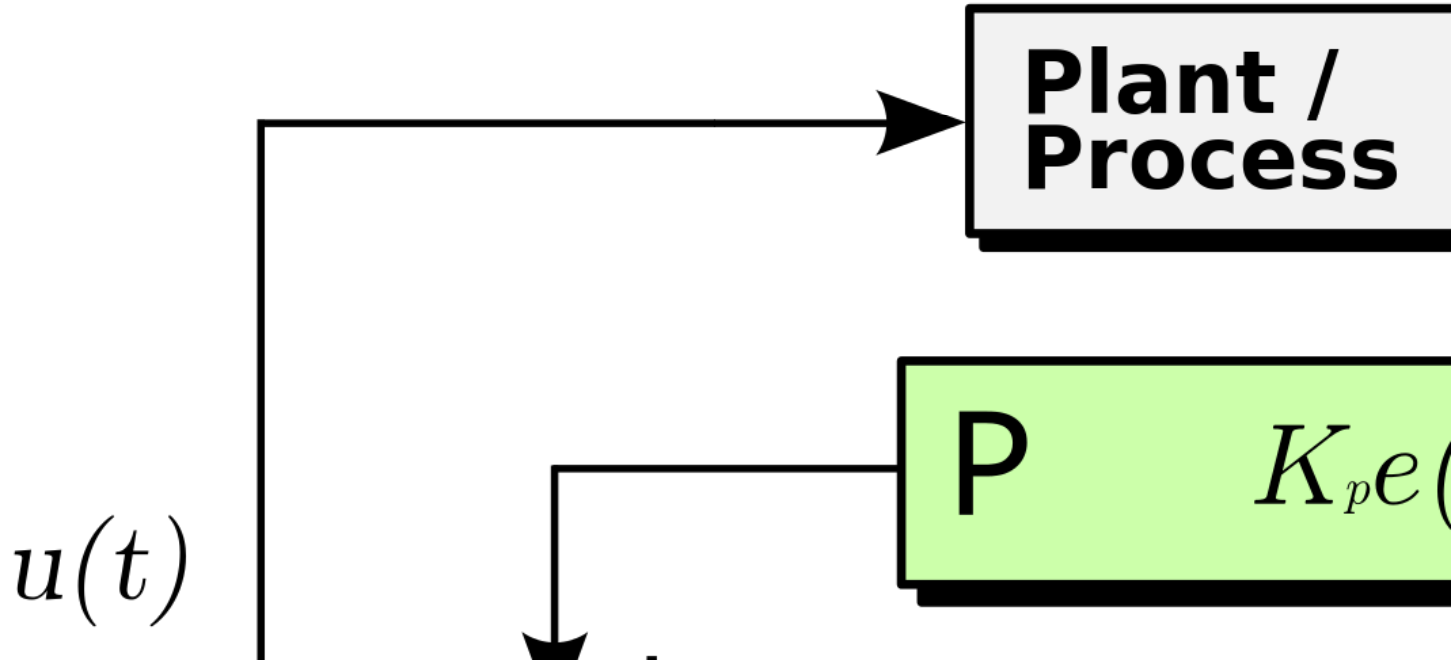
Hệ thống điều khiển

Hệ thống điều khiển tuyến tính sử dụng phản hồi âm để tạo ra tín hiệu điều khiển nhằm duy trì PV được kiểm soát tại SP mong muốn. Có một số loại hệ thống điều khiển tuyến tính với các khả năng khác nhau.

Điều khiển tỷ lệ-PI là một dạng điều khiển phản hồi tuyến tính, trong đó hiệu chỉnh được áp dụng cho biến được kiểm soát tỷ lệ thuận với sự khác biệt giữa giá trị mong muốn (SP) và giá trị đo được (PV).

Bộ điều khiển PI thuần túy phải hoạt động với lỗi còn lại trong hệ thống. Mặc dù bộ điều khiển PI loại bỏ lỗi này, chúng vẫn có thể chậm chạp hoặc tạo ra dao động. *Bộ điều khiển PID* giải quyết những thiếu sót cuối cùng này bằng cách giới thiệu một hành động phái sinh (D) để duy trì sự ổn định trong khi khả năng đáp ứng được cải thiện.

Hệ thống điều khiển



Hệ thống điều khiển

Logic mờ là một tiếp cận giúp thiết kế dễ dàng bộ điều khiển logic để điều khiển các hệ thống phức tạp liên tục khác nhau. Về cơ bản, một phép đo trong một hệ thống logic mờ có thể đúng một phần.

Các quy tắc của hệ thống được viết bằng ngôn ngữ tự nhiên và được dịch sang logic mờ.

Các phép đo từ thế giới thực bị mờ và logic được tính toán số học, trái ngược với logic Boolean và đầu ra được khử mờ để điều khiển thiết bị.

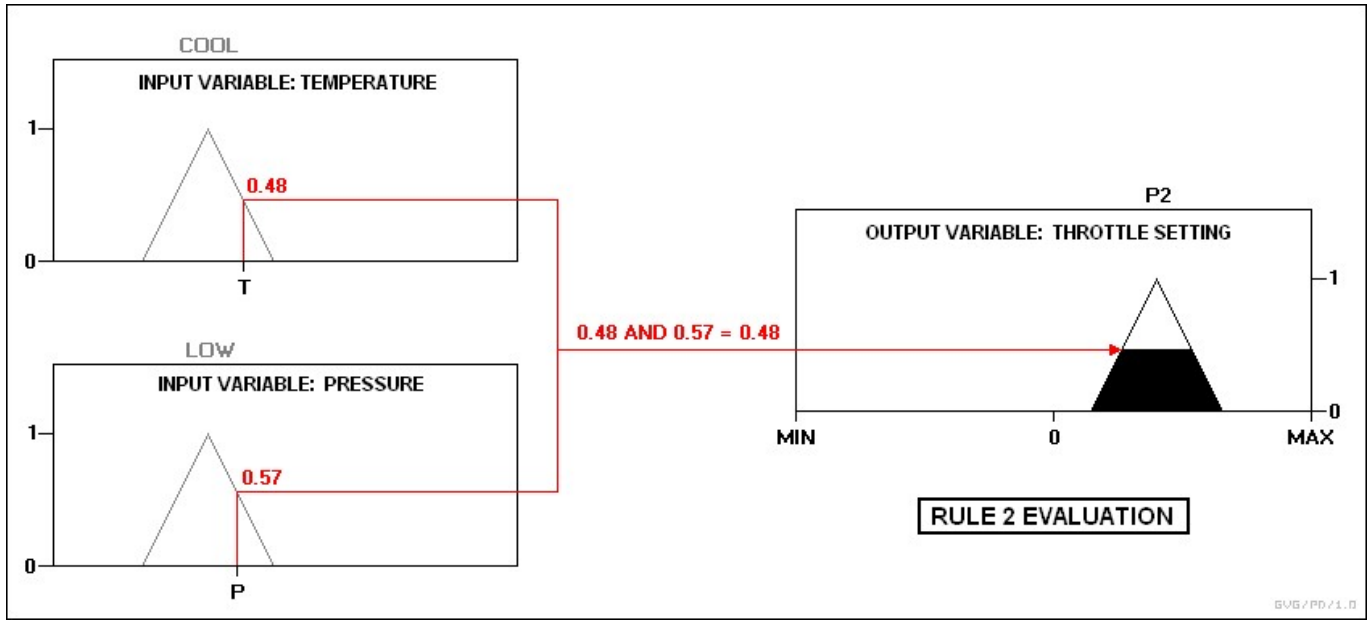
Khi một thiết kế mờ mạnh mẽ được giảm thành một tính toán nhanh, duy nhất, nó sẽ giống như một giải pháp vòng lặp phản hồi thông thường và có vẻ như thiết kế mờ là không cần thiết. Tuy nhiên, mô hình logic mờ có thể cung cấp khả năng mở rộng cho các hệ thống điều khiển lớn, nơi các phương pháp thông thường trở nên khó sử dụng hoặc tốn kém.

Điện tử mờ là một công nghệ điện tử sử dụng logic mờ thay vì logic hai giá trị thường được sử dụng trong các thiết bị điện tử kỹ thuật số.

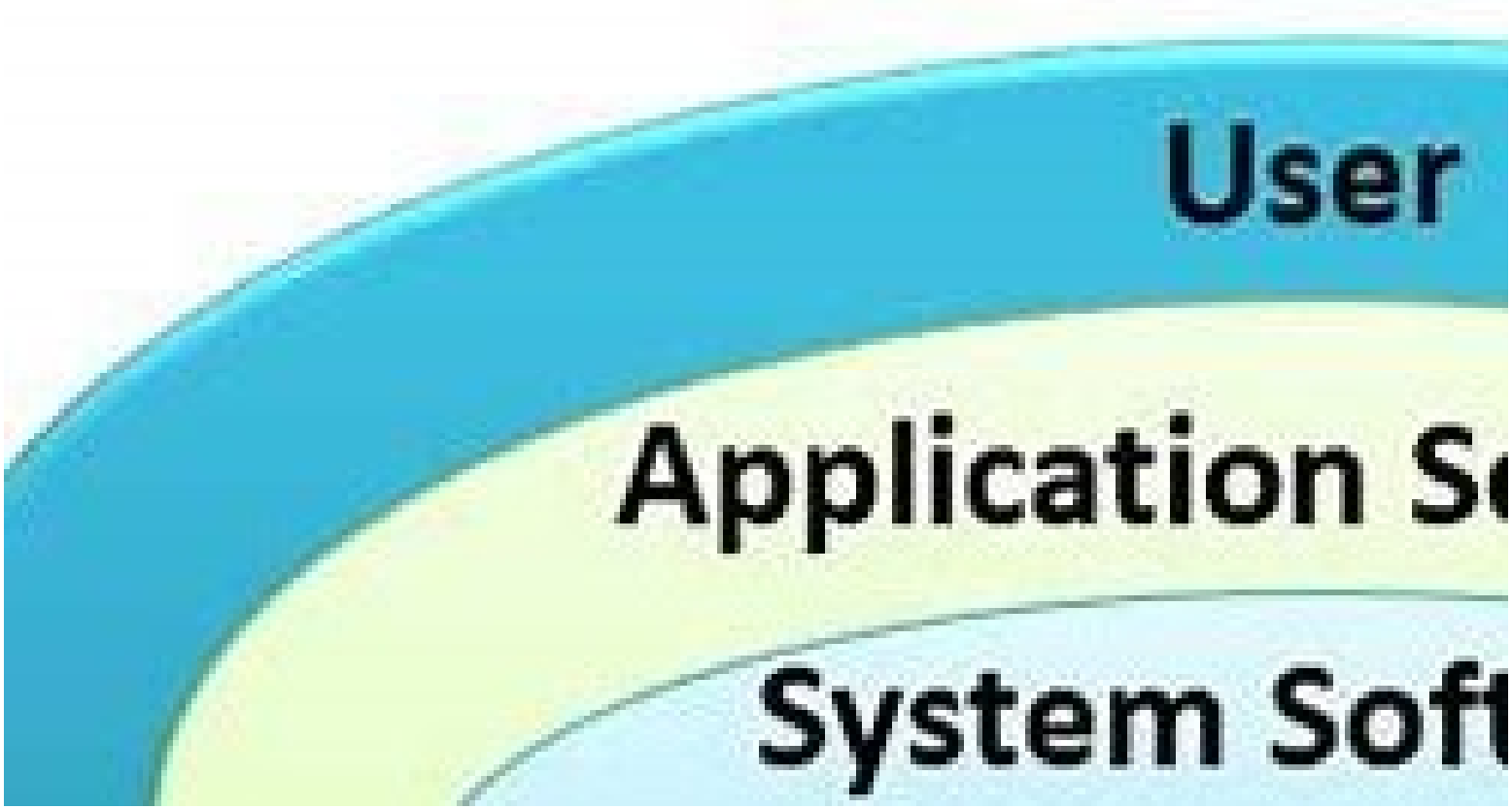
Hệ thống điều khiển

Phạm vi triển khai hệ thống điều khiển là từ bộ điều khiển nhỏ gọn thường với phần mềm chuyên dụng cho một máy hoặc thiết bị cụ thể, đến các hệ thống điều khiển phân tán để kiểm soát quy trình công nghiệp cho một nhà máy vật lý lớn.

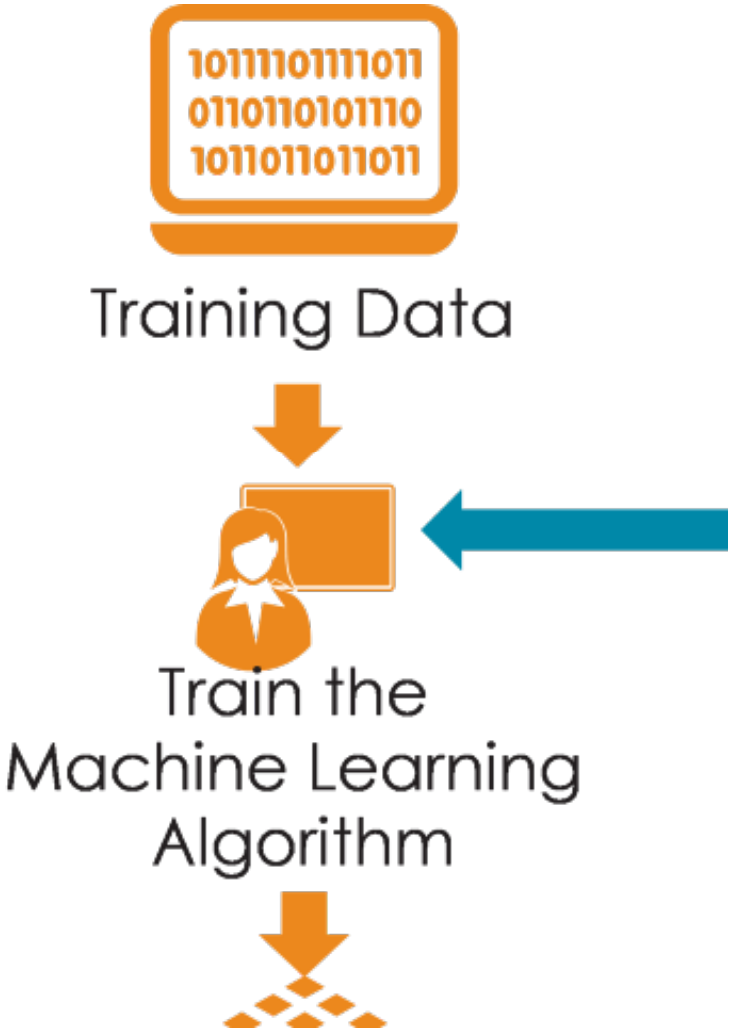
Hệ thống logic và bộ điều khiển phản hồi thường được thực hiện với bộ điều khiển logic có thể lập trình.



Phần cứng và phần mềm



Dữ liệu/mô hình/thuật toán/máy tính



Từ **Điều khiển** có hai ý nghĩa chính:

Đầu tiên, **Điều khiển** được hiểu là hoạt động kiểm tra xem một thiết bị vật lý hoặc toán học có hành vi thỏa đáng hay không.

Thứ hai, **Điều khiển** là hành động, thực hiện các quyết định đảm bảo rằng thiết bị hoạt động như mong muốn.

Chương 1

Mở đầu về điều khiển thông minh

Lịch sử phát triển của điều khiển học

Lý thuyết điều khiển có từ thế kỷ 19, khi cơ sở lý thuyết cho hoạt động của các bộ điều chỉnh đầu tiên được mô tả bởi James Clerk Maxwell. Lý thuyết điều khiển được Edward Routh phát triển hơn nữa vào năm 1874, Charles Sturm và năm 1895, Adolf Hurwitz, tất cả đều góp phần thiết lập các tiêu chí ổn định; và từ năm 1922 trở đi là thời kỳ phát triển lý thuyết điều khiển PID của Nicolas Minorsky.

Mặc dù ứng dụng chính của lý thuyết điều khiển toán học là trong kỹ thuật hệ thống điều khiển, liên quan đến việc thiết kế các hệ thống điều khiển quy trình cho ngành công nghiệp, các ứng dụng trong lĩnh vực khác cũng được phát triển phong phú.

Lý thuyết điều khiển rất hữu ích ở bất cứ hệ thống nào có phản hồi - do đó lý thuyết điều khiển cũng có các ứng dụng trong khoa học đời sống, kỹ thuật máy tính, xã hội học và nghiên cứu vận hành.

Lịch sử phát triển của điều khiển học

Mặc dù các hệ thống điều khiển của các loại khác nhau có từ thời cổ đại, một phân tích chính thức hơn về lĩnh vực này bắt đầu với một phân tích động lực của bộ điều chỉnh ly tâm, được thực hiện bởi nhà vật lý James Clerk Maxwell vào năm 1868, mang tên On Governors.

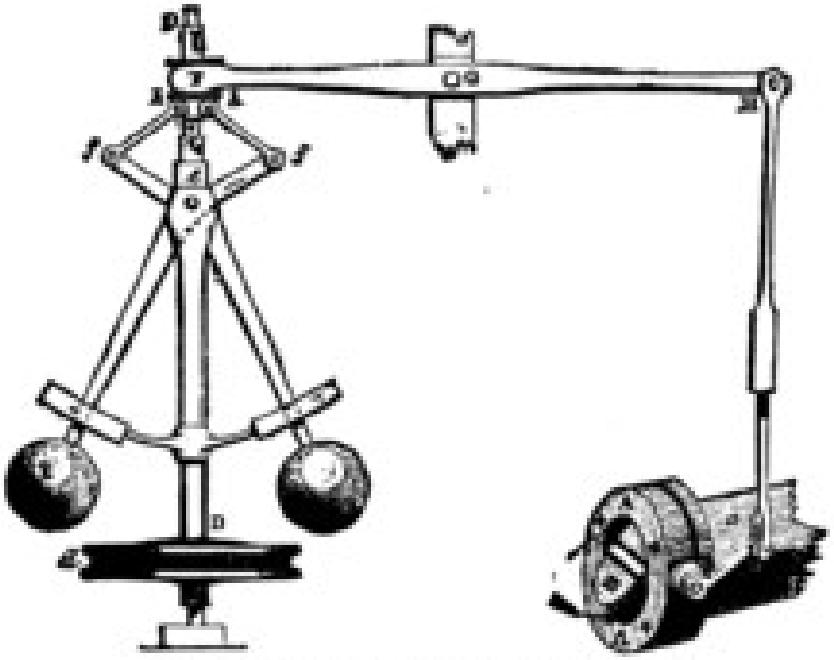
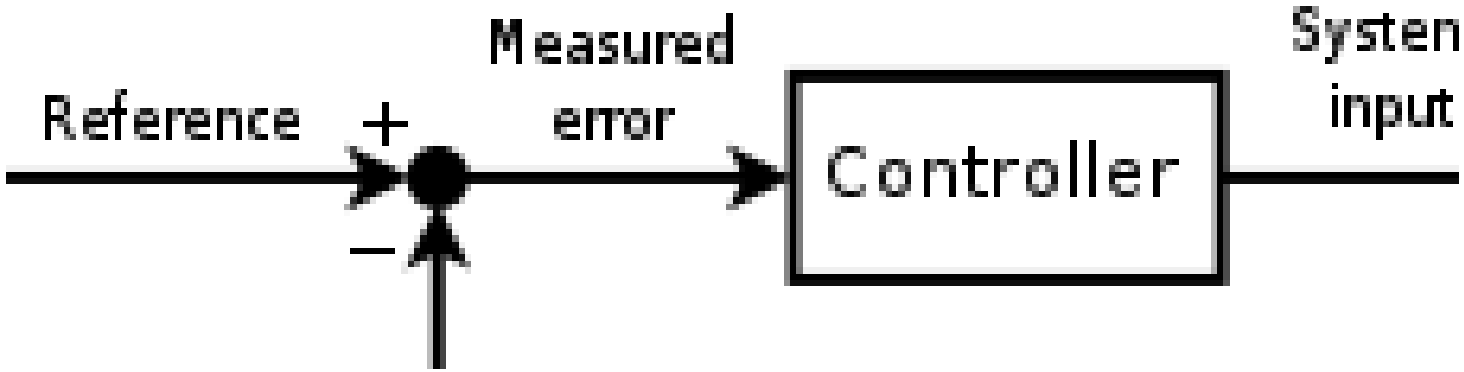


FIG. 4.—*Governor and Throttle-Valve.*

Lịch sử phát triển của điều khiển học

Ngày ngay, một bộ điều khiển giám sát biến quy trình được điều khiển (PV) và so sánh nó với điểm tham chiếu hoặc điểm đặt (SP). Sự khác biệt giữa giá trị thực tế và mong muốn của biến quy trình, được gọi là tín hiệu sai lệch SP-PV, được áp dụng làm phản hồi để tạo ra một hành động điều khiển để đưa biến quy trình được kiểm soát đến cùng giá trị với điểm đặt. Các khía cạnh khác cũng được nghiên cứu là khả năng kiểm soát và quan sát.



Cơ sở trong lý thuyết điều khiển

Mặc dù biểu diễn các vấn đề điều khiển dựa trên mô hình toán học của các hệ thống vật lý, về bản chất là phức tạp, những ý tưởng cơ bản trong lý thuyết điều khiển là đủ đơn giản rất trực quan. Những ý tưởng quan trọng này có thể được tìm thấy trong Tự nhiên, trong sự tiến hóa và hành vi của chúng sinh. Có ba khái niệm cơ bản trong lý thuyết điều khiển.

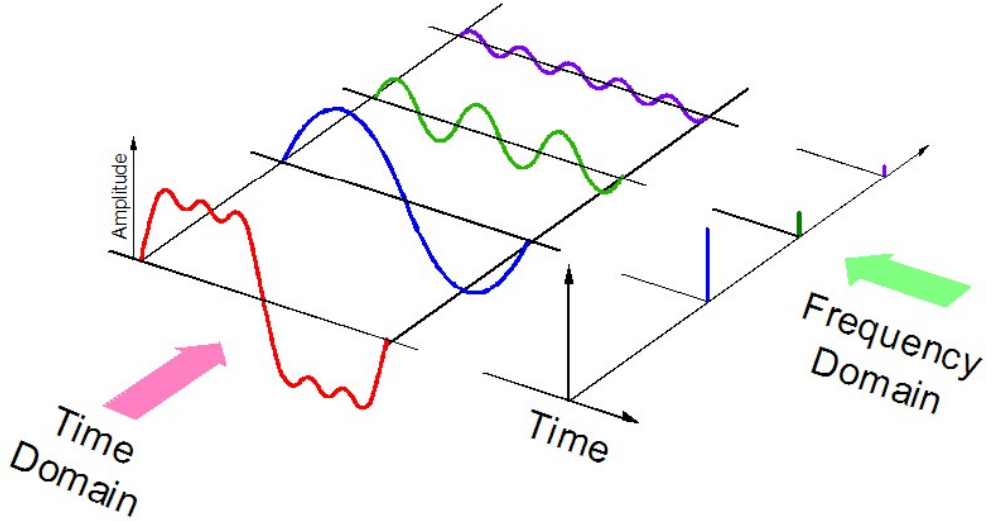
Đầu tiên là phản hồi: là một quá trình trong đó trạng thái của hệ thống, hoặc đầu ra của nó, xác định cách thức mà điều khiển phải được tính toán bất cứ lúc nào ngay lập tức.

Khái niệm quan trọng thứ hai trong lý thuyết điều khiển là **dao động** (biến động). Khái niệm này ngụ ý là chúng ta không nhất thiết phải buộc hệ thống và ép nó đến trạng thái mong muốn ngay lập tức hoặc trực tiếp.

Khái niệm thứ ba rất quan trọng trong lý thuyết điều khiển là **tối ưu hóa**. Đây là một nhánh của toán học nhằm tìm ra giá trị của các biến nhằm tối đa hóa lợi điểm và giảm thiểu nhược điểm với các ràng buộc có sẵn.

Cơ sở trong lý thuyết điều khiển

Trong vật lý, điện tử, kỹ thuật hệ thống điều khiển và thống kê, miền tần số đề cập đến việc phân tích các chức năng hoặc tín hiệu toán học liên quan đến tần số, thay vì thời gian. Nói một cách đơn giản, một biểu đồ miền thời gian cho thấy tín hiệu thay đổi theo thời gian như thế nào, trong khi biểu đồ miền tần số cho thấy có bao nhiêu tín hiệu nằm trong mỗi dải tần số nhất định trên một loạt các tần số.



Tên miền thời gian đề cập đến việc phân tích các chức năng toán học, tín hiệu vật lý hoặc chuỗi thời gian của dữ liệu kinh tế hoặc môi trường, liên quan đến thời gian. Trong miền thời gian, giá trị của tín hiệu hoặc hàm được biết đến với tất cả các số thực, cho trường hợp thời gian liên tục hoặc tại các khoảnh khắc riêng biệt khác nhau trong trường hợp thời gian rời rạc.

Kỹ thuật điều khiển

Điều khiển thích nghi là kỹ thuật xác định trực tuyến các tham số quy trình hoặc sửa đổi độ lợi bộ điều khiển, do đó có được các đặc tính mạnh mẽ. Điều khiển thích nghi đã được áp dụng lần đầu tiên trong ngành công nghiệp hàng không vũ trụ vào những năm 1950, và đã tìm thấy thành công đặc biệt trong lĩnh vực đó.

Điều khiển phân cấp là một loại hệ thống điều khiển trong đó một tập hợp các thiết bị và phần mềm quản lý được sắp xếp trong một cây phân cấp. Khi các liên kết trong cây được thực hiện bởi một mạng máy tính, thì hệ thống điều khiển phân cấp đó cũng là một hình thức của hệ thống điều khiển mạng.

Điều khiển thông minh sử dụng các phương pháp tính toán AI khác nhau như mạng thần kinh nhân tạo, xác suất Bayesian, logic mờ, học máy, tính toán tiến hóa và thuật toán di truyền hoặc kết hợp các phương pháp này, chẳng hạn như thuật toán mờ thần kinh, để điều khiển một hệ thống động.

Kỹ thuật điều khiển

Điều khiển tối ưu là một kỹ thuật điều khiển một "chỉ số tối ưu" cụ thể, trong đó tín hiệu điều khiển tối ưu hóa một "chỉ số tối ưu" nhất định: ví dụ, trong trường hợp của động cơ, mômen cần thiết để quay trục động cơ tới quỹ đạo mong muốn tiêu thụ ít năng lượng nhất.

Điều khiển bền vững là kỹ thuật điều khiển liên quan một cách tường minh đến sự không chắc chắn trong cách tiếp cận thiết kế bộ điều khiển. Bộ điều khiển được thiết kế bằng các phương pháp điều khiển bền vững có xu hướng có thể đối phó với sự khác biệt nhỏ giữa hệ thống thực sự và mô hình danh nghĩa được sử dụng để thiết kế.

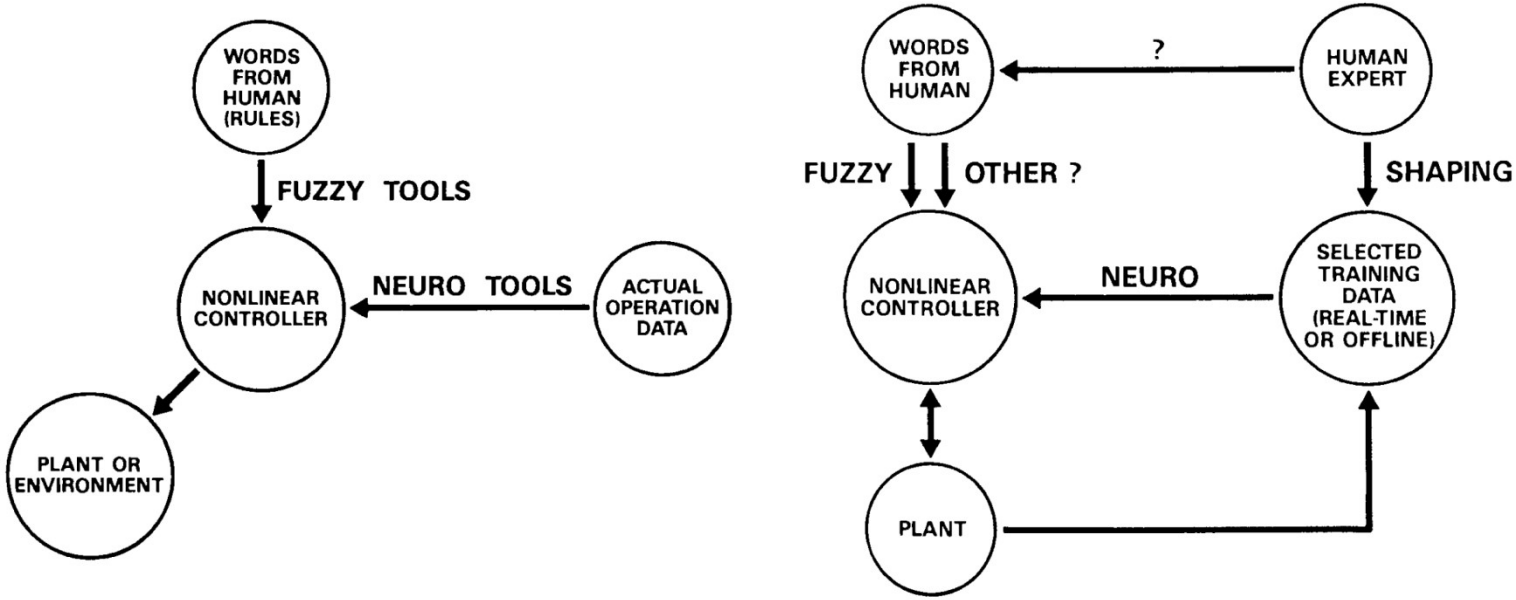
Điều khiển stochastic là kỹ thuật liên quan đến thiết kế điều khiển với sự không chắc chắn trong mô hình. Trong các vấn đề điều khiển stochastic điển hình, người ta cho rằng tồn tại tiếng ồn và nhiễu ngẫu nhiên trong mô hình và bộ điều khiển, và thiết kế điều khiển phải tính đến những sai lệch ngẫu nhiên này.

Điều khiển tự tổ chức là kỹ thuật điều khiển nỗ lực can thiệp vào các quá trình mà hệ thống tự tổ chức tiêu tán năng lượng.

Điều khiển thông minh

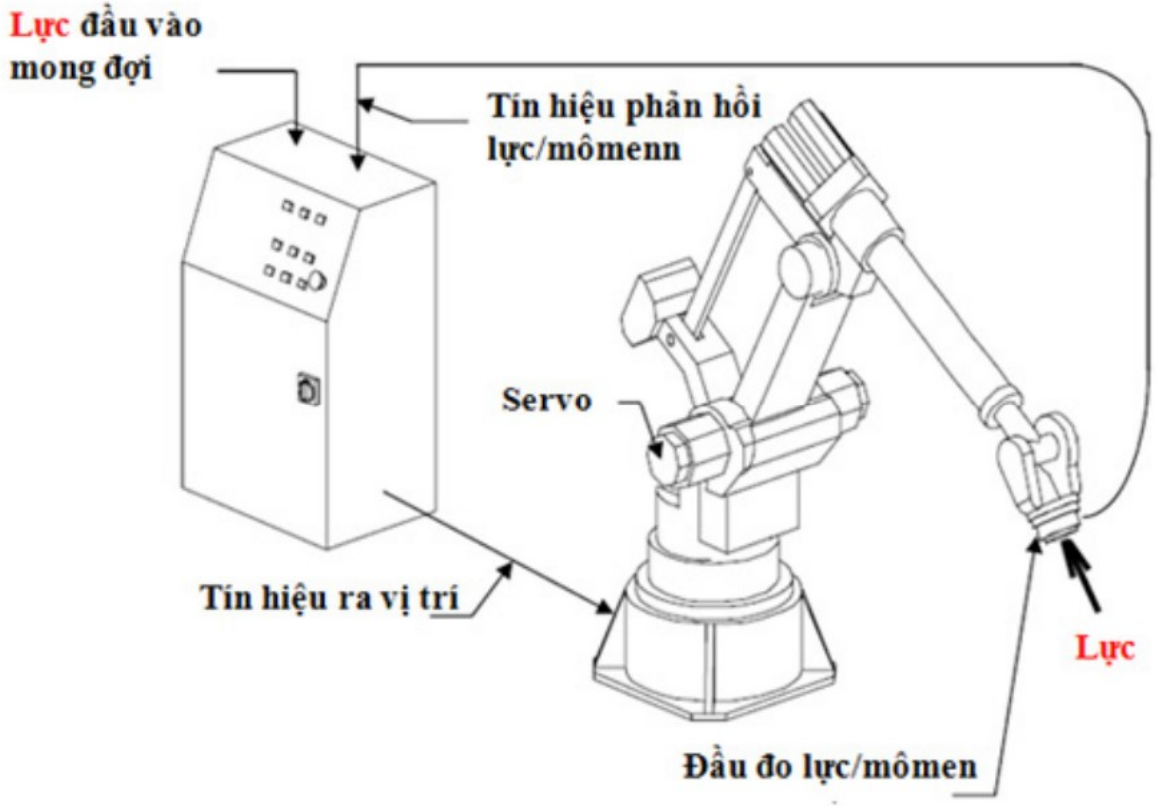
Điều khiển thông minh là việc sử dụng các hệ thống điều khiển đa năng, **học theo thời gian cách đạt được mục tiêu** (hoặc tối ưu hóa) trong môi trường phức tạp, ồn ào, phi tuyến tính mà động lực cuối cùng phải được học trong thời gian thực.

Loại điều khiển này không thể đạt được bằng những cải tiến đơn giản, gia tăng so với các phương pháp hiện có.

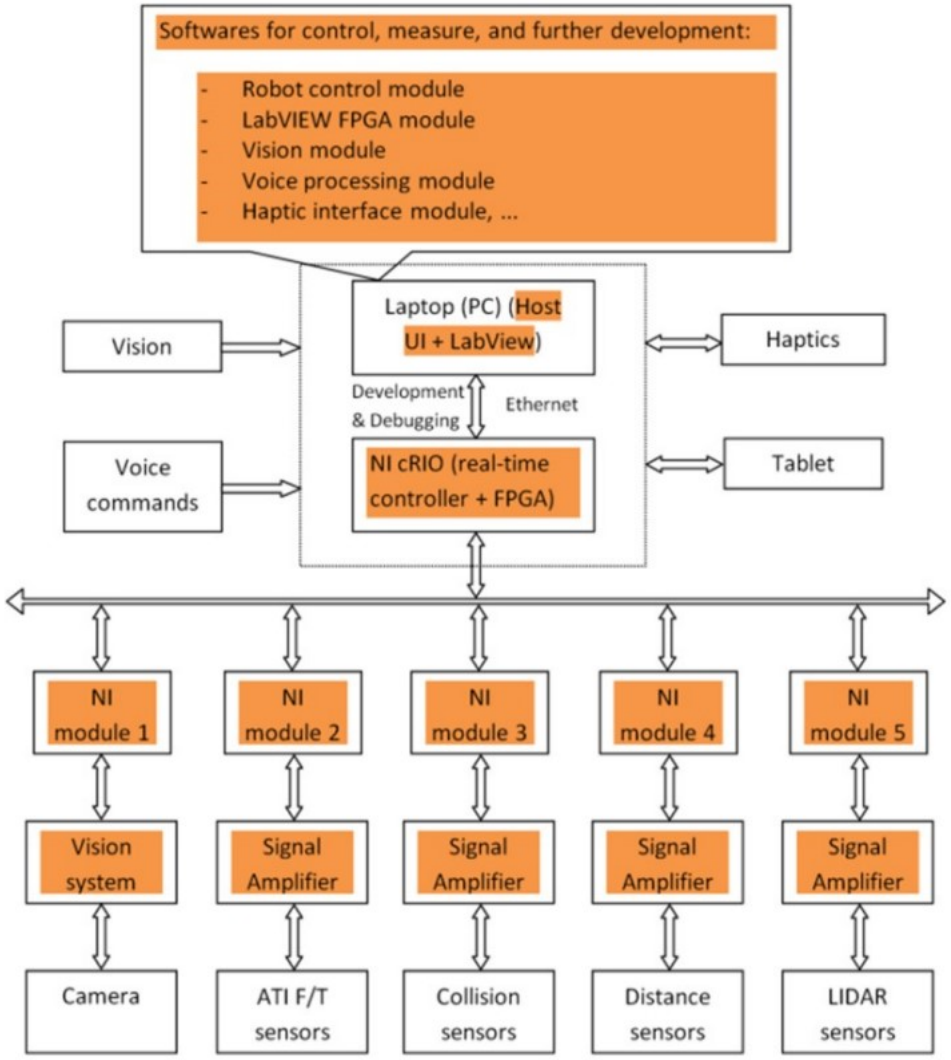


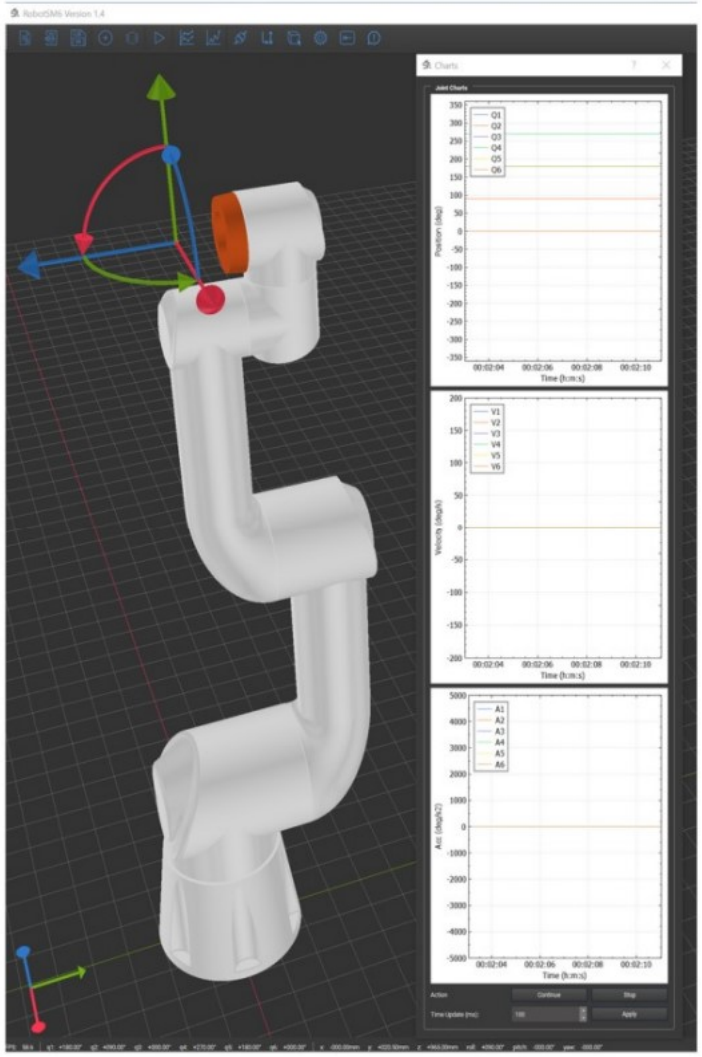
Hệ thống thông minh

Hệ thống thông minh

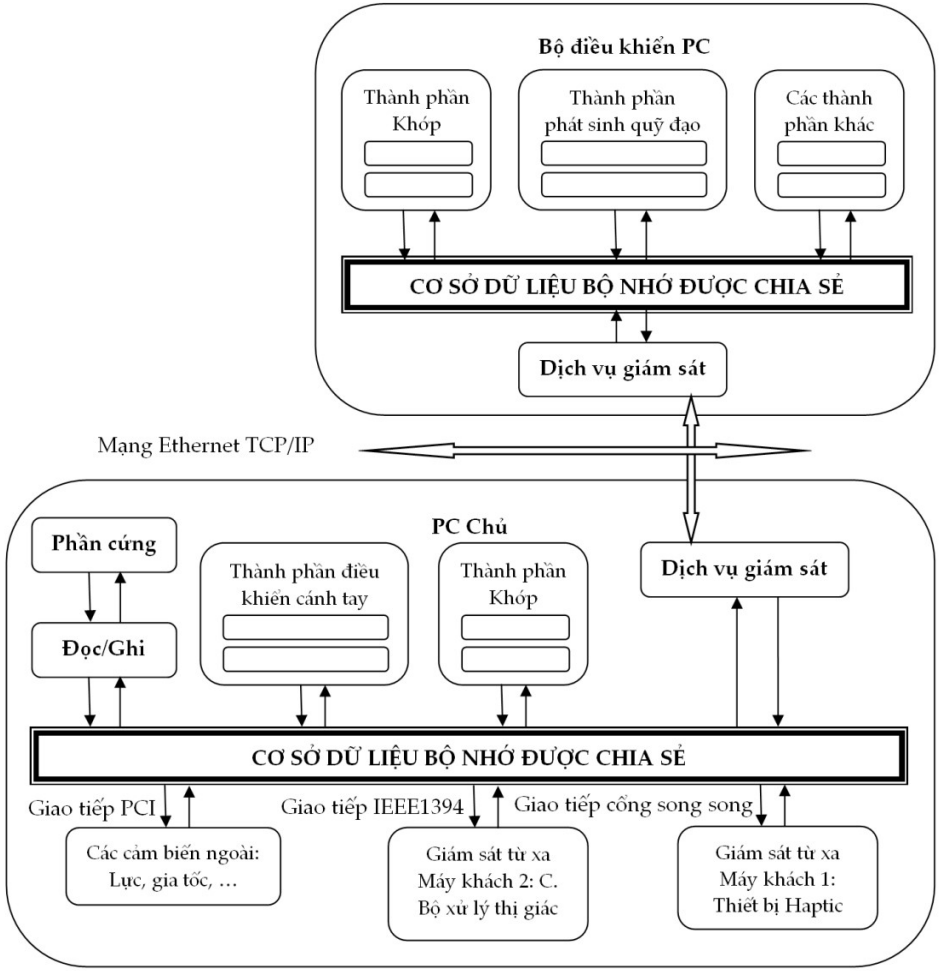


Hệ thống thông minh





Hệ thống thông minh



Đặc điểm của hệ thống điều khiển thông minh

Là hệ thống tích hợp phần cứng + phần mềm sử dụng máy tính số có thể thực hiện các thao tác số và biểu tượng tương tự như một người bình thường có thể, nhưng nhanh hơn và đáng tin cậy hơn -> máy tính (hoặc một chương trình máy tính) có thể suy nghĩ giống người.

Là cỗ máy có thể bắt chước hoặc vượt quá khả năng tinh thần của con người, bao gồm lý luận, hiểu biết, trí tưởng tượng, sự công nhận, sáng tạo và cảm xúc.

Một số thành công đã đạt được: đã có thể có thể chơi cờ vua ở cấp độ cao nhất, biên dịch ngôn ngữ và chẩn đoán y tế, …

Đặc điểm của hệ thống điều khiển thông minh

Hệ thống là sản phẩm liên ngành: khoa học máy tính, kỹ thuật tính toán hiện đại, trí tuệ nhân tạo, học máy, dữ liệu lớn, ...

Bao gồm: hệ thống dựa trên kiến thức (*hiểu biết*), trí thông minh tính toán (*suy luận/phân tích/tính toán*) và hệ thống *lai ghép*.

Đặc điểm của hệ thống điều khiển thông minh

Các hệ thống **dựa trên kiến thức** bao gồm các hệ thống **chuyên gia** và **dựa trên quy tắc** (luật/qui định/nguyên tắc), các hệ thống hướng đối tượng và dựa trên khung mẫu và các đại lý thông minh.

Trí thông minh tính toán bao gồm **mạng thần kinh**, **thuật toán di truyền** và các thuật toán tối ưu hóa khác.

Các kỹ thuật để xử lý sự không chắc chắn, chẳng hạn như logic mờ, phù hợp với cả hai loại.

Hệ thống điều khiển thông minh hiện tại cho phép giải quyết một loạt các vấn đề mà trước đây được coi là quá khó khăn theo cách hiệu quả hơn.

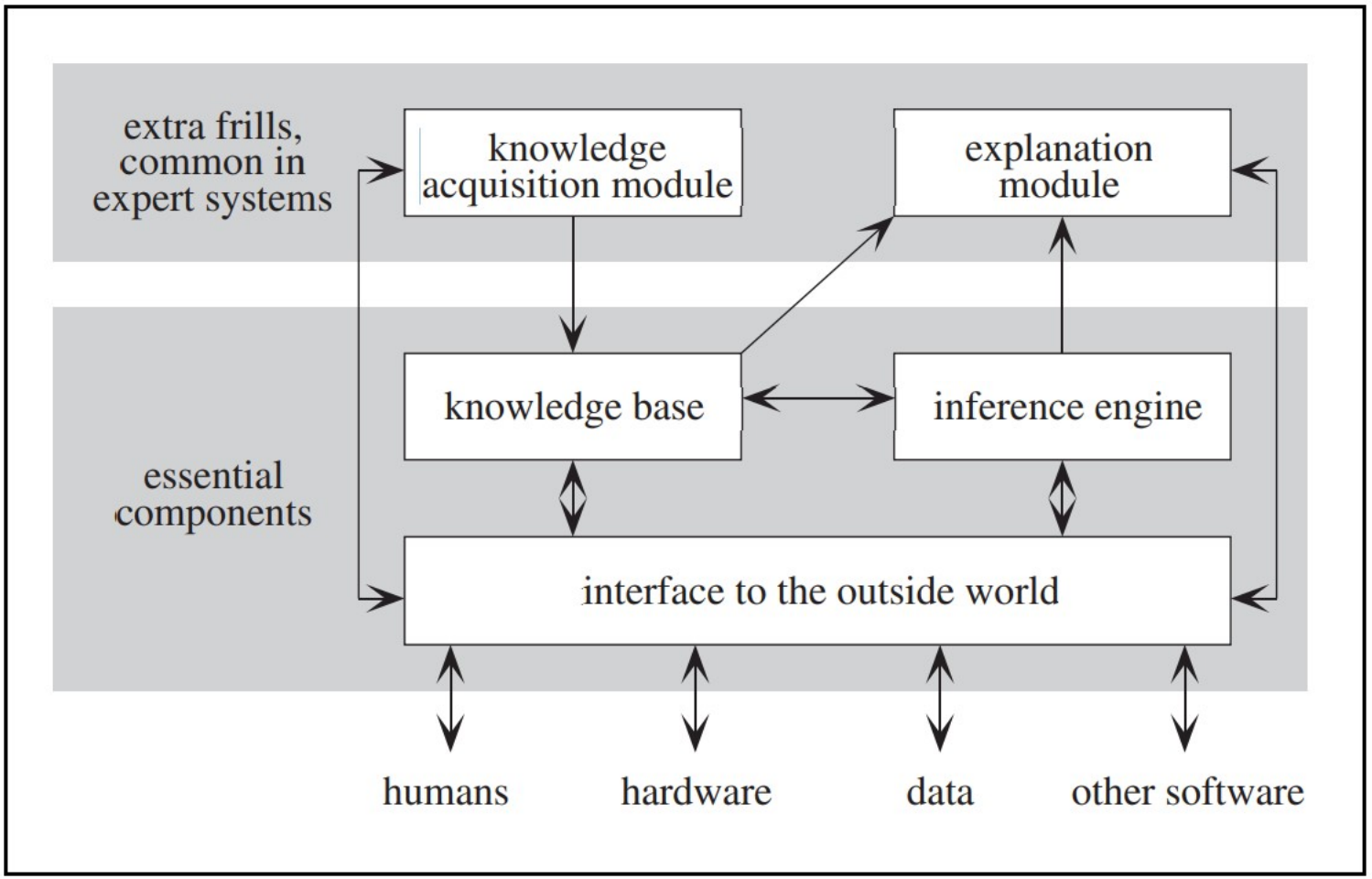
Hệ thống dựa trên hiểu biết

Trong một hệ thống dựa trên hiểu biết/kiến thức, có hai vai trò được tách biệt theo hai mô đun rõ ràng:

Mô đun cơ sở kiến thức: tích lũy kiến thức (bổ sung, cập nhật, …). Kiến thức được thể hiện rõ ràng trong cơ sở kiến thức, không ngầm trong cấu trúc của một chương trình (hệ thống truyền thống).

Mô đun điều khiển (công cụ/máy suy luận): thuật toán (chương trình) độc lập (không bị thay đổi) với mô đun cơ sở kiến thức được thay đổi liên tục bởi kiến thức và kinh nghiệm mới.

Hệ thống dựa trên hiểu biết



Hệ thống dựa trên hiểu biết

Cách tiếp cận hệ thống dựa trên kiến thức đơn giản hơn. Kiến thức được thể hiện rõ ràng trong **cơ sở kiến thức**, không ngầm trong **cấu trúc của một chương trình**.

Do đó, kiến thức có thể được thay đổi với **giao diện bộ suy luận** cơ sở kiến thức để giải thích mô-đun tiếp thu kiến thức thế giới bên ngoài mô-đun phần cứng phần mềm khác cần thiết các thành phần thêm phổ biến trong dữ liệu hệ thống chuyên gia. Các thành phần chính của một hệ thống dựa trên kiến thức tương đối dễ dàng. Công cụ suy luận sử dụng cơ sở kiến thức để giải quyết một nhiệm vụ cụ thể theo cách tương tự như một chương trình thông thường sử dụng tệp dữ liệu.

Cơ sở/nền tảng hiểu biết

Nền tảng kiến thức có thể phong phú với các hình thức kiến thức đa dạng. Để đơn giản coi cơ sở kiến thức chứa **quy tắc** và **thực tế**.

Tuy nhiên, các **quy tắc** có thể phức tạp và các **thực tế** có thể bao gồm các trình tự, thực thể có cấu trúc, thuộc tính của các thực thể đó và mối quan hệ giữa chúng.

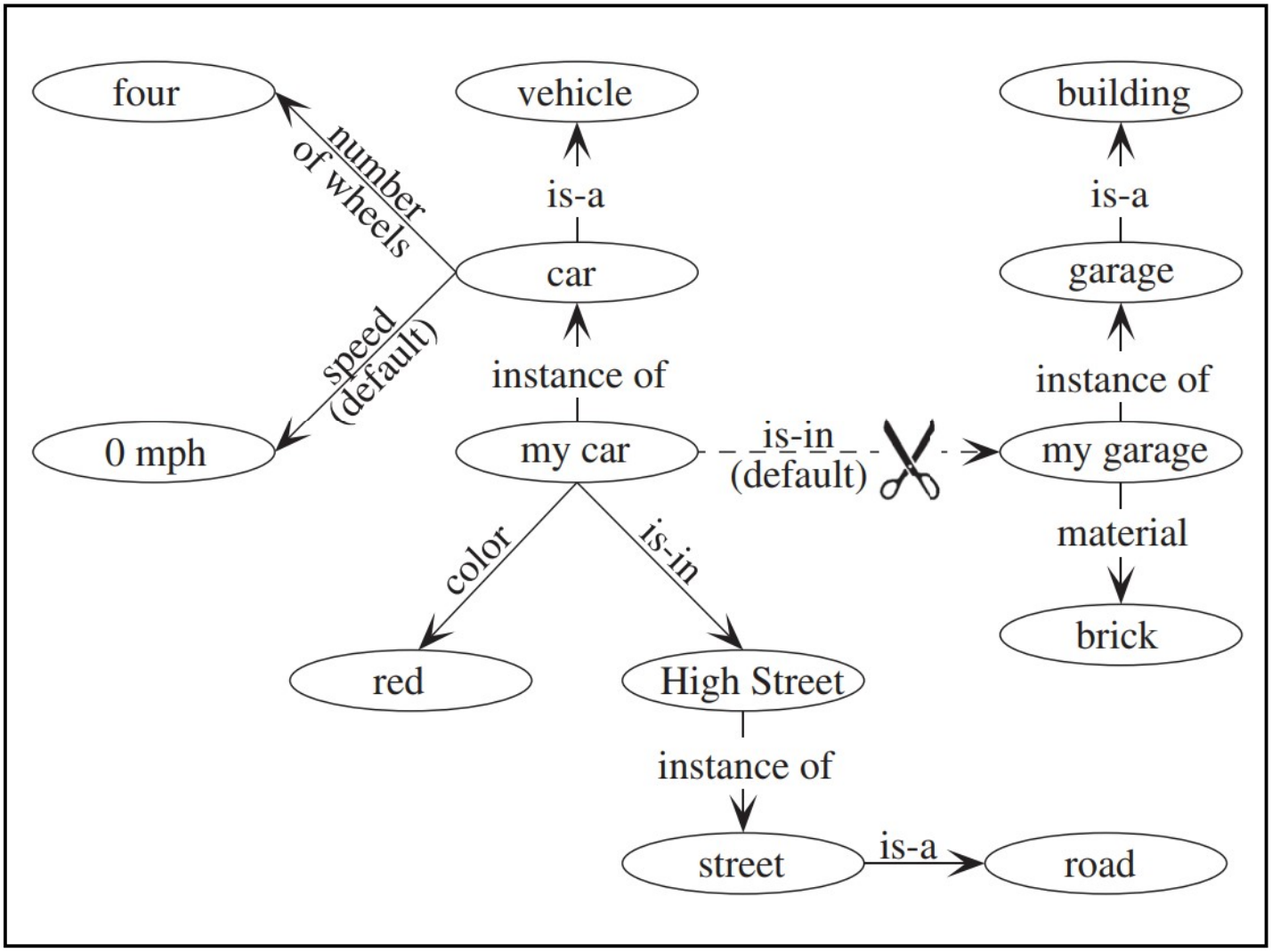
Vì nhiều hệ thống cần hàng trăm hoặc hàng ngàn các thực tế và quy tắc, đưa chúng vào một chương trình thông thường là một nhiệm vụ khó khăn. **Điều này có thể tương phản với một hệ thống dựa trên kiến thức, trong đó quy tắc và thực tế được thể hiện rõ ràng và có thể được thay đổi theo ý muốn**.

Thực tế có thể là tĩnh, trong đó trường hợp chúng có thể được ghi vào cơ sở kiến thức. Lưu ý rằng các thực tế tĩnh không cần phải là vĩnh viễn, nhưng chúng thay đổi đủ không thường xuyên để thay đổi có thể được đáp ứng bằng cách cập nhật cơ sở kiến thức khi cần thiết.

Thực tế có thể thay đổi nhanh với một phiên bản xác định hoặc cho một lần chạy của hệ thống.

Cơ sở kiến thức có thể chứa các mặc định, có thể được sử dụng làm thực tế trong trường hợp không có thực tế nhanh.

Cơ sở/nền tảng hiểu biết



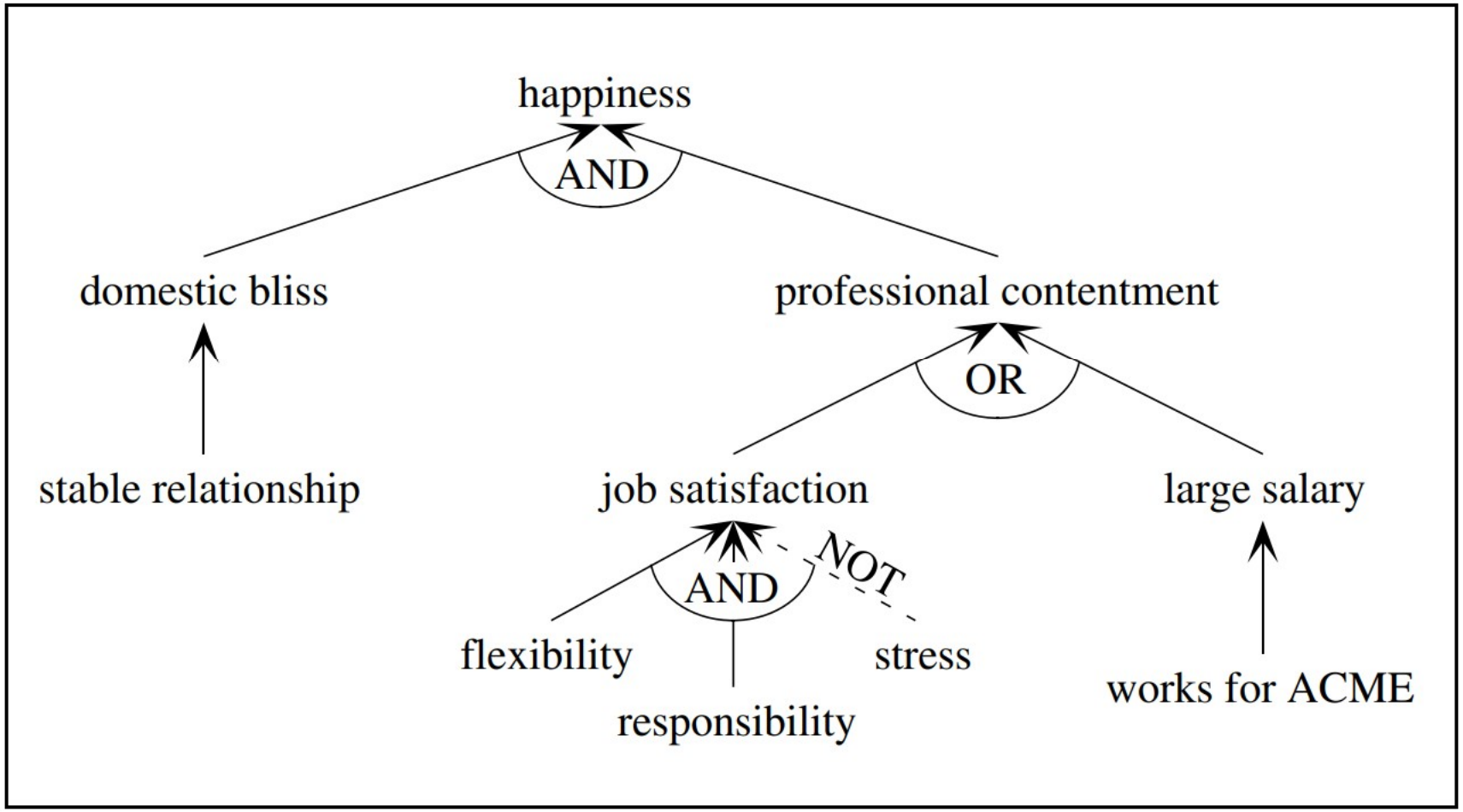
Cơ sở/nền tảng hiểu biết

Lưu ý phân biệt giữa thuộc tính và mối quan hệ. Thuộc tính là thuộc tính của các phiên bản đối tượng (chẳng hạn như xe của tôi) hoặc các lớp đối tượng (chẳng hạn như ô tô và xe). Mối quan hệ tồn tại giữa các phiên bản đối tượng và các lớp đối tượng.

Các thuộc tính và mối quan hệ có thể được biểu diễn như một mạng lưới, được gọi là mạng liên kết hoặc ngữ nghĩa. Trong đó, các thuộc tính được đối xử theo cách tương tự như các mối quan hệ.

Các thực tế có thể được cung cấp cho hệ thống dựa trên kiến thức ngay từ đầu (thực tế tĩnh) hoặc trong khi hệ thống đang chạy (các thực tế thoáng qua). Một hoặc nhiều thực tế nhất định có thể đáp ứng điều kiện của một quy tắc, dẫn đến việc tạo ra một thực tế mới, được gọi là một thực tế có nguồn gốc. Sự phụ thuộc lẫn nhau giữa các quy tắc xác định một mạng, được gọi là mạng suy luận.

Cơ sở/nền tảng hiểu biết



Loại trừ/Phân rã/Quy nạp

Các quy tắc tạo nên mạng lưới suy luận và toàn bộ mạng được thực hiện, được sử dụng để liên kết nguyên nhân và kết quả. **Đây là quá trình loại trừ.**

Nhiều vấn đề, chẳng hạn như chẩn đoán, liên quan đến lý luận theo hướng ngược lại, tức là, muốn xác định một nguyên nhân, khi đưa ra một kết quả. **Đây là quá trình phân rã**.

Chú ý: Mạng lưới suy luận đại diện cho một thế giới khép kín, nơi không có gì được biết đến ngoài ranh giới của nó. **Vì mỗi nút đại diện** cho một trạng thái có thể có của một số khía cạnh của thế giới, một mô hình của trạng thái tổng thể hiện tại của thế giới có thể được duy trì. Một mô hình như vậy phụ thuộc vào mức độ của các mối quan hệ giữa các nút trong mạng suy luận. Đặc biệt, nếu một sự thay đổi xảy ra ở một khía cạnh (nút) của thế giới, nhiều nút khác có thể bị ảnh hưởng. Xác định những gì khác được thay đổi trong mô hình thế giới do hậu quả của việc thay đổi một điều cụ thể được gọi là vấn đề khung mẫu.

Suy ra một quy tắc từ một tập hợp các trường hợp ví dụ về nguyên nhân và kết quả được gọi là **quy nạp**.

Loại trừ (suy diễn)/Phân rã/Quy nạp

Loại trừ: nguyên nhân + quy tắc → kết quả

Phân rã: kết quả + quy tắc → nguyên nhân

Quy nạp: nguyên nhân + kết quả → quy tắc

Máy/Công cụ suy luận

Có nhiều công cụ suy luận rất khác nhau tùy thuộc vào loại và độ phức tạp của kiến thức mà chúng đối phó. Hai loại động cơ suy luận quan trọng có thể được phân biệt: **truyền thuận** (hoạt động dựa vào dữ liệu) và **truyền ngược** (hoạt động dựa vào mục tiêu).

Hệ thống dựa trên kiến thức làm việc trong chế độ dựa trên dữ liệu lấy thông tin có sẵn (các thực tế "đã cho") và tạo ra càng nhiều thực tế có nguồn gốc càng tốt. Do đó, đầu ra không thể đoán trước. Điều này có thể có lợi thế dẫn đến các giải pháp mới hoặc sáng tạo cho một vấn đề hoặc bất lợi của việc lãng phí thời gian tạo ra thông tin không liên quan.

Cách tiếp cận dựa trên dữ liệu thường có thể được sử dụng cho các vấn đề giải thích, khi muốn biết bất cứ điều gì hệ thống có thể cho biết về một số dữ liệu.

Máy/Công cụ suy luận

Một chiến lược định hướng mục tiêu là thích hợp khi một giải pháp tập trung chặt chẽ hơn là chỉ ở mức cần thiết.

Ví dụ, một hệ thống lập kế hoạch có thể được yêu cầu để tạo ra một kế hoạch sản xuất một sản phẩm tiêu dùng. Bất kỳ kế hoạch nào khác đều không liên quan. Lúc này có thể sử dụng một hệ thống truyền ngược với đề xuất: một kế hoạch tồn tại để sản xuất một công cụ. Sau đó, là cố gắng xác định sự thật của mệnh đề này bằng cách **tạo ra kế hoạch**, hoặc nó có thể kết luận rằng đề xuất là sai và không có kế hoạch nào là có thể.

Lập trình thủ tục và khai báo

Lưu ý rằng một đặc điểm đặc biệt của một *hệ thống dựa trên kiến thức* là **kiến thức** *được tách ra* **khỏi lý luận**.

Trong cơ sở kiến thức, lập trình viên thể hiện thông tin về vấn đề cần giải quyết. Thông thường thông tin này là tuyên bố, tức là, lập trình viên nêu ra một số thực tế, quy tắc hoặc mối quan hệ mà không phải quan tâm đến chi tiết về cách thức và thời điểm thông tin đó được áp dụng. Mỗi ví dụ đại diện cho một phần kiến thức có thể tạo thành một phần của cơ sở kiến thức. Lập trình viên khai báo không nhất thiết phải nêu rõ làm thế nào, khi nào và nếu kiến thức nên được sử dụng. Những chi tiết này được tiềm ẩn trong công cụ suy luận.

Một công cụ suy luận thường được lập trình theo thủ tục - một tập hợp các lệnh tuần tự được tuân theo, liên quan đến việc trích xuất và sử dụng thông tin từ cơ sở kiến thức. Nhiệm vụ này có thể được thực hiện rõ ràng bằng cách sử dụng **siêu kiến thức** (kiến thức về kiến thức).

Lập trình thủ tục và khai báo

Hầu hết các chương trình thông thường là thủ tục chứa các hướng dẫn từng bước rõ ràng yêu cầu máy tính thực hiện các hành động cụ thể.

Mặt khác, tệp dữ liệu không chứa hướng dẫn cho máy tính, chỉ có thông tin dưới dạng một tập hợp các dữ liệu. Các hướng dẫn thủ tục để xác định những gì máy tính nên làm với dữ liệu trong chương trình.

Do đó, tệp dữ liệu là khai báo, trong khi chương trình là **thủ tục/hành động**. *Tệp dữ liệu tương tự như một cơ sở kiến thức* và *chương trình tương tự như công cụ suy luận tương ứng*.

Tất nhiên, một nền tảng kiến thức thích hợp sẽ phong phú hơn về nội dung, có lẽ chứa sự kết hợp của các quy tắc, thực tế, mối quan hệ và dữ liệu. Công cụ suy luận tương ứng sẽ giải thích kiến thức, kết hợp nó để tạo thành một cái nhìn tổng thể, áp dụng kiến thức vào dữ liệu và đưa ra quyết định.

Lập trình thủ tục và khai báo

Để đơn giản hóa có thể coi rằng cơ sở kiến thức được viết bằng các tuyên bố và tất cả các công cụ suy luận được viết theo thủ tục. Trong các hệ thống thực tế, một bộ sưu tập thông tin khai báo, chẳng hạn như một bộ quy tắc, thường cần phải được tô điểm bởi một số thông tin thủ tục.

Tương tự như vậy, có thể có một số công cụ suy luận đã được lập trình khai báo, đặc biệt là những công cụ được thực hiện trong Prolog. Tuy nhiên, các hướng dẫn khai báo cuối cùng phải được dịch sang các hướng dẫn thủ tục, vì máy tính chỉ có thể hiểu các hướng dẫn thủ tục ở cấp độ mã máy.

Hệ thống chuyên gia

Hệ thống chuyên gia là một loại hệ thống dựa trên kiến thức được thiết kế để hiện thực hóa chuyên môn trong một lĩnh vực chuyên ngành cụ thể. Ví dụ có thể là chuyên ngành về cấu hình mạng máy tính, chẩn đoán lỗi trong điện thoại hoặc tìm kiếm khoáng sản.

Một hệ thống chuyên gia được thiết kế để hoạt động như một người chuyên gia có thể tư vấn được về một loạt các vấn đề nằm trong phạm vi chuyên môn của hệ thống.

Thông thường, người sử dụng hệ thống chuyên gia sẽ tham gia vào một cuộc đối thoại trong đó họ mô tả vấn đề (chẳng hạn như các triệu chứng của lỗi) và hệ thống chuyên gia đưa ra lời khuyên, đề xuất hoặc khuyến nghị.

Cuộc đối thoại có thể được dẫn dắt bởi hệ thống chuyên gia, để người dùng trả lời một loạt các câu hỏi hoặc nhập thông tin vào bảng tính.

Hệ thống chuyên gia

Ngoài ra, hệ thống chuyên gia có thể cho phép người dùng chủ động trong việc tư vấn bằng cách cho phép họ cung cấp thông tin mà không nhất thiết phải được yêu cầu.

Vì một hệ thống chuyên gia là một hệ thống dựa trên kiến thức hoạt động như một chuyên gia tư vấn, do vậy một hệ thống chuyên gia phải cung cấp một số khả năng nhất định phản ánh những khả năng của một người tư vấn.

Đặc biệt, người ta thường tuyên bố rằng một hệ thống chuyên gia phải có khả năng biện minh cho dòng điều tra hiện tại của nó và giải thích lý do của nó khi đi đến kết luận.

Tuy nhiên, điều tốt nhất mà hầu hết các hệ thống chuyên gia có thể đạt được là tạo ra một dấu vết của các thực tế và quy tắc đã được sử dụng. Điều này tương đương với dấu vết của con đường thực hiện cho một chương trình thông thường, một khả năng thường được coi là tiêu chuẩn và không đặc biệt đáng chú ý.

Hệ thống chuyên gia

Một khung vỏ hệ thống chuyên gia là một hệ thống chuyên gia với một nền tảng kiến thức trống rỗng. Chúng được bán dưới dạng các gói phần mềm có độ phức tạp khác nhau. Về nguyên tắc, có thể mua một khung vỏ hệ thống chuyên gia, xây dựng một cơ sở kiến thức, và do đó tạo ra một hệ thống chuyên gia.

Tuy nhiên, tất cả các tên miền đều khác nhau và rất khó để một nhà cung cấp phần mềm xây dựng một khung vỏ xử lý đầy đủ tất cả.

Khung vỏ tốt nhất là linh hoạt trong khả năng của nó để biểu diễn và áp dụng kiến thức. Nếu không có sự linh hoạt này, có thể cần phải tạo ra các quy tắc và thực tế theo phong cách phức tạp để phù hợp với cú pháp hoặc buộc một loại hành vi nhất định từ hệ thống. Tình huống này hầu như không tốt hơn so với việc xây dựng một chương trình thông thường.

Tiếp thu/nhận kiến thức

Biểu diễn của kiến thức trong một nền tảng kiến thức chỉ có thể được giải quyết một khi kiến thức được biết đến. Dưới đây là ba cách tiếp cận riêng biệt để có được kiến thức liên quan cho một lĩnh vực cụ thể:

1. Kiến thức có được từ một chuyên gia trong lĩnh vực;
2. Kiến thức trong hệ thống dựa trên hiểu biết được xây dựng từ chuyên gia trong lĩnh vực;
3. Hệ thống tự động học hỏi từ các ví dụ.

Tiếp thu/nhận kiến thức

Cách tiếp cận đầu tiên thường được sử dụng, nhưng đầy khó khăn. Người trích xuất kiến thức từ chuyên gia và mã hóa nó trong cơ sở kiến thức được gọi là kỹ sư kiến thức.

Thông thường, kỹ sư kiến thức phỏng vấn một hoặc nhiều chuyên gia lĩnh vực và cố gắng làm cho họ nói rõ kiến thức chuyên sâu của họ theo cách mà kỹ sư kiến thức có thể hiểu được.

Những khó khăn giao tiếp không thể tránh khỏi có thể tránh được bằng cách tiếp cận thứ hai, trong đó chuyên gia lĩnh vực trở thành kỹ sư kiến thức hoặc kỹ sư kiến thức trở thành chuyên gia lĩnh vực.

Có nhiều trường hợp trong đó kiến thức hoặc không được biết đến hoặc không thể được thể hiện rõ ràng. Trong những trường hợp này, có thể tốt hơn là hệ thống tạo ra cơ sở kiến thức của riêng mình từ một tập hợp các ví dụ.

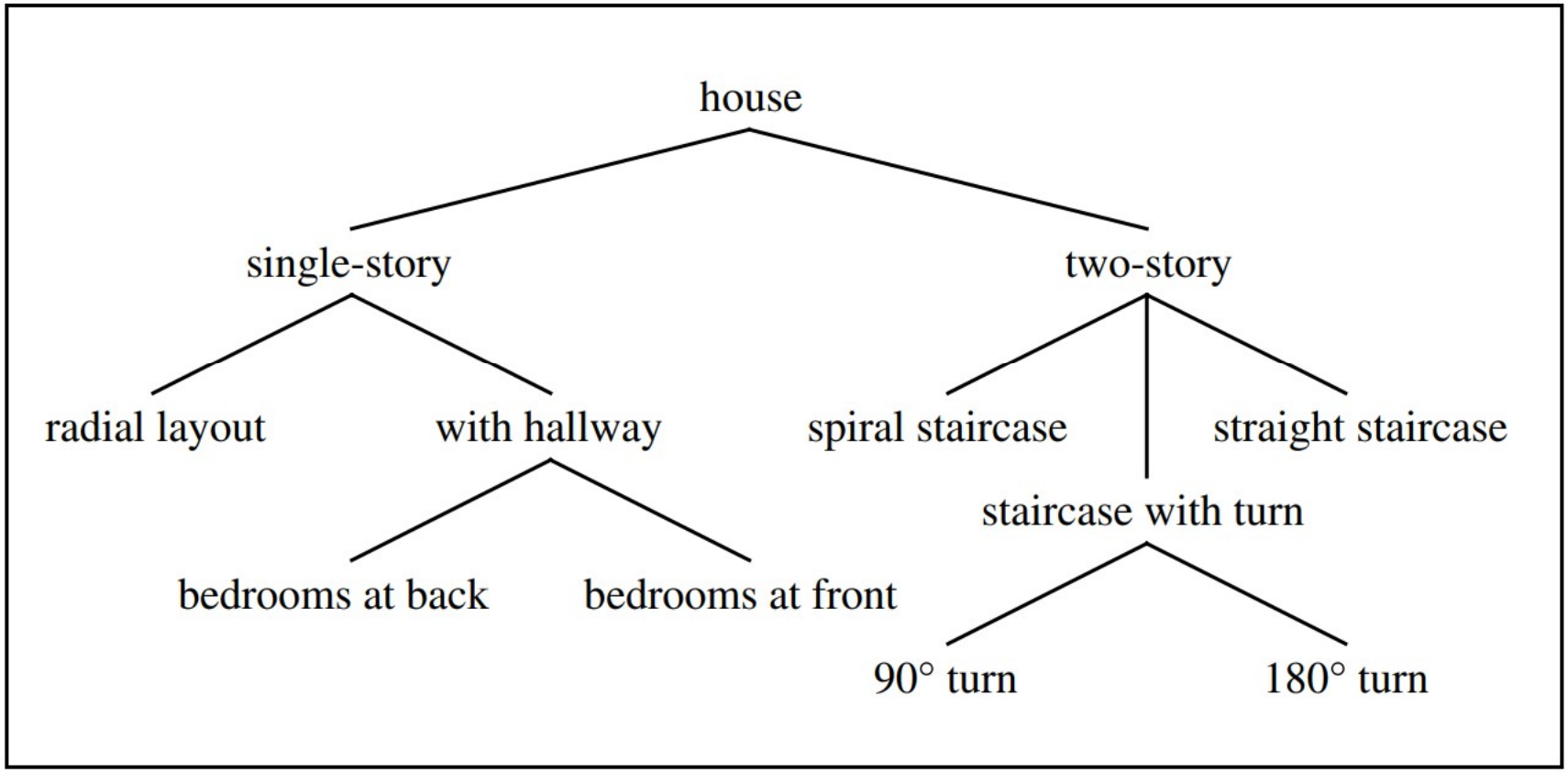
Tìm kiếm

Tìm kiếm là **chìa khóa** cho tất cả các nhiệm vụ giải quyết vấn đề trong thực tế, và đã là một trọng tâm chính của nghiên cứu trong các hệ thống thông minh.

Giải quyết vấn đề liên quan đến việc tìm kiếm một giải pháp. Các ứng dụng kỹ thuật tìm kiếm bao gồm tìm kiếm thiết kế, kế hoạch, hành động kiểm soát hoặc chẩn đoán lỗi. Tất cả các ứng dụng này liên quan đến việc tìm kiếm thông qua các giải pháp có thể (không gian tìm kiếm) để tìm một hoặc nhiều ứng dụng tối ưu hoặc thỏa đáng.

Tìm kiếm cũng là một vấn đề quan trọng đối với hoạt động nội bộ của một hệ thống dựa trên kiến thức.

Tìm kiếm



Tìm kiếm

Cơ sở kiến thức có thể chứa hàng trăm hoặc hàng ngàn quy tắc và thực tế. ***Vai trò chính của công cụ suy luận là tìm kiếm mục kiến thức phù hợp nhất để áp dụng tại bất kỳ thời điểm nào***.

Trong trường hợp tìm kiếm cơ sở kiến thức, có thể (mặc dù không hiệu quả lắm) để kiểm tra tất cả các lựa chọn thay thế trước khi chọn một. Điều này được gọi là tìm kiếm toàn diện.

Trong không gian tìm kiếm của một ứng dụng như thiết kế hoặc chẩn đoán, tìm kiếm toàn diện có thể không thực tế vì số lượng giải pháp ứng cử viên rất lớn.

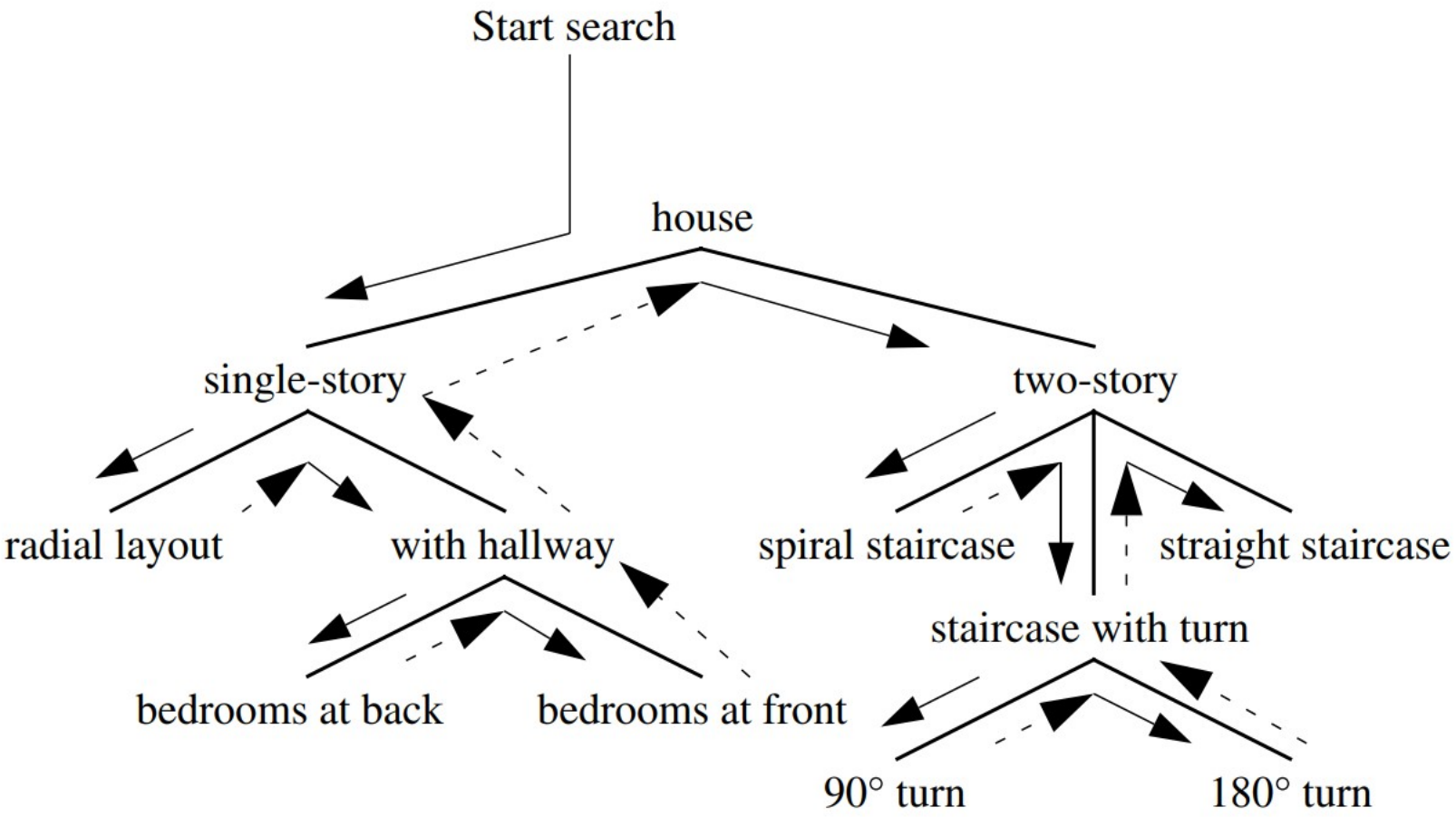
Do đó, một cuộc tìm kiếm thực tế phải có chọn lọc. Để đạt được điều này, các giải pháp ứng cử viên tạo nên không gian tìm kiếm có thể được tổ chức như một cây tìm kiếm.

Tìm kiếm

Cây tìm kiếm chỉ ra rằng các giải pháp có thể được phân loại theo một số cách, do đó các giải pháp tương tự được tập hợp lại với nhau trên cùng một nhánh của cây.

Không giống như một cây thật, cây được hiển thị ở đây có rễ ở trên cùng và lá của nó ở phía dưới. Khi chúng ta tiến về phía lá, sự khác biệt giữa các thiết kế trở nên ít đáng kể hơn. Mỗi thiết kế thay thế được tạo tự động hoặc tìm thấy trong cơ sở dữ liệu, sau đó được kiểm tra cho phù hợp. Điều này được mô tả như là một chiến lược tạo ra và thử nghiệm. Nếu một giải pháp vượt qua bài kiểm tra, việc tìm kiếm có thể tiếp tục để tìm thêm các giải pháp chấp nhận được hoặc nó có thể dừng lại.

Tìm kiếm



Tìm kiếm

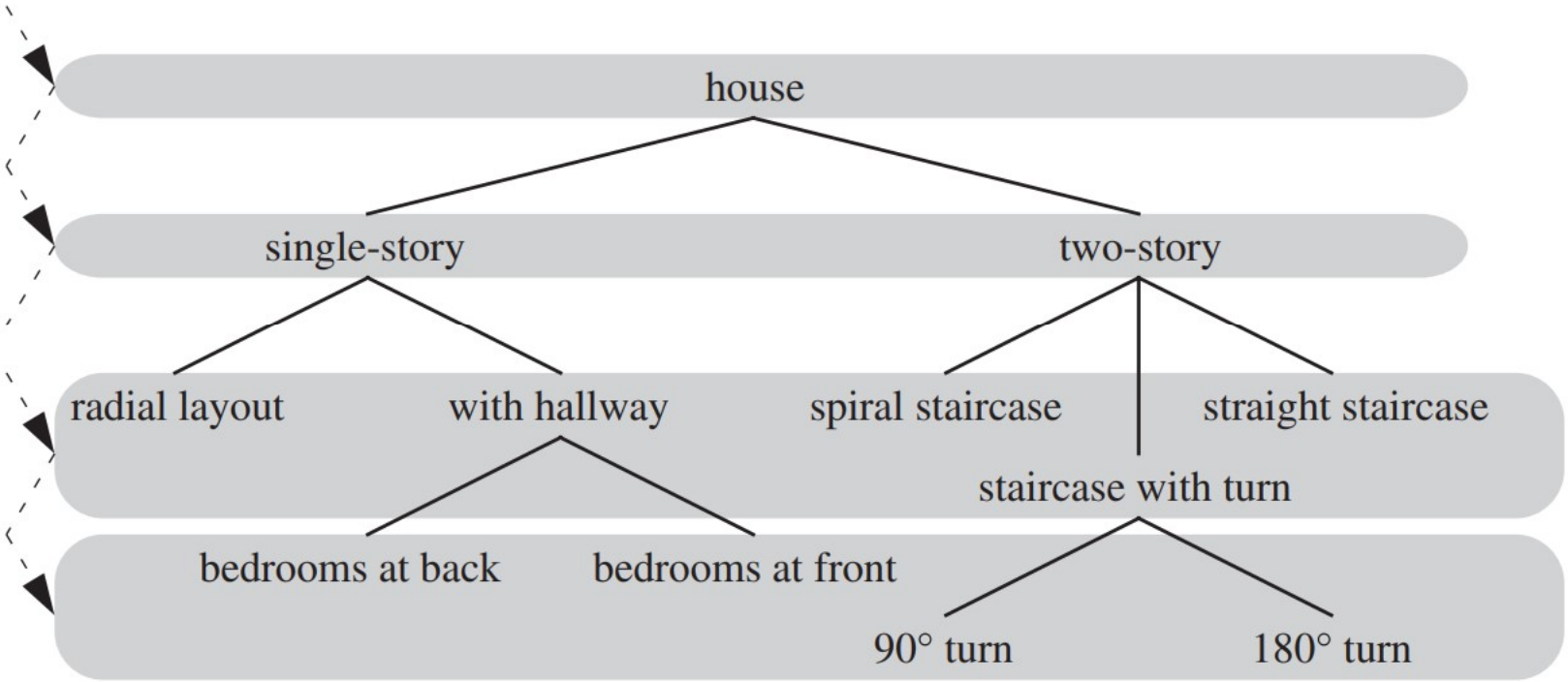
Hai chiến lược thay thế để tìm kiếm cây một cách có hệ thống là tìm kiếm theo chiều sâu trước và theo chiều rộng trước.

Ngược lại, cách tiếp cận theo chiều rộng trước liên quan đến việc kiểm tra tất cả các nút ở một mức độ nhất định trong cây trước khi tiến lên cấp độ tiếp theo.

Mỗi nút là một khái quát hóa của các nút trên các nhánh con của nó. Do đó, nếu một nút thất bại trong bài kiểm tra, tất cả các danh mục con được giả định là thất bại trong bài kiểm tra và bị loại bỏ khỏi tìm kiếm.

Các chiến lược tìm kiếm có hệ thống được mô tả ở trên là những ví dụ về tìm kiếm mù.

Tìm kiếm



Tìm kiếm

Việc tìm kiếm có thể được thực hiện hiệu quả hơn bằng cách loại bỏ các danh mục không khả thi ("cắt tỉa cây tìm kiếm") hoặc bằng cách đảm bảo rằng các lựa chọn thay thế có khả năng nhất được thử nghiệm trước những lựa chọn ít có khả năng nhất.

Để đạt được một trong hai điều này, chúng ta cần áp dụng suy nghiệm vào quá trình tìm kiếm. Do đó, tìm kiếm mù được sửa đổi thành một tìm kiếm suy nghiệm.

Barr và Feigenbaum đã khảo sát việc sử dụng từ suy nghiệm và đưa ra mô tả chung như sau:

Suy nghiệm là một quy tắc kinh nghiệm, chiến lược, thủ thuật, đơn giản hóa hoặc bất kỳ loại công cụ/phương thức nào khác hạn chế đáng kể việc tìm kiếm các giải pháp trong không gian tìm kiếm lớn. Suy nghiệm không đảm bảo các giải pháp tối ưu; trên thực tế, không đảm bảo bất kỳ giải pháp nào cả; tác dụng của suy nghiệm là cung cấp các giải pháp đủ tốt hầu hết thời gian.

Trí thông minh tính toán

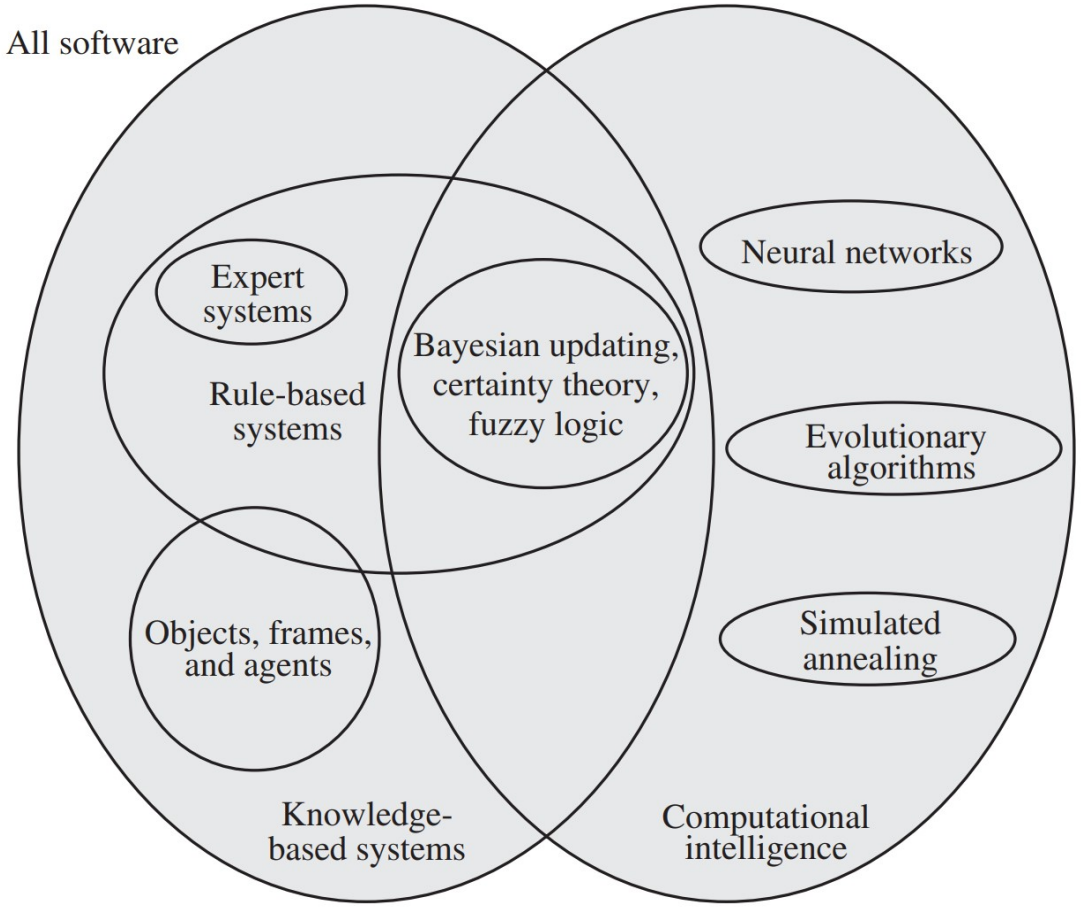
Các kiểu hệ thống dựa trên kiến thức đều là những đại diện mang tính biểu tượng, trong đó kiến thức được thể hiện rõ ràng bằng các từ và biểu tượng được kết hợp để tạo thành các quy tắc, thực tế, mối quan hệ hoặc các hình thức đại diện kiến thức khác.

Khi kiến thức được viết rõ ràng, nó có thể được đọc và hiểu bởi một con người. Những kỹ thuật biểu tượng này tương phản với các kỹ thuật số như thuật toán di truyền và mạng lưới thần kinh.

Trong trường hợp kiến thức không được nêu rõ ràng nhưng được thể hiện bằng các con số được điều chỉnh khi hệ thống cải thiện độ chính xác của nó. Những kỹ thuật này được gọi chung là trí thông minh tính toán (CI) hoặc tính toán mềm.

Có 03 kỹ thuật để xử lý sự không chắc chắn: cập nhật Bayesian, các thành tố chắc chắn và logic mờ. Tất cả các kỹ thuật này đều sử dụng hỗn hợp các quy tắc và các giá trị số liên quan, và do đó chúng có thể được coi vừa là công cụ của trí thông minh tính toán và vừa là công cụ của hệ thống dựa trên kiến thức.

Trí thông minh tính toán



Trí thông minh tính toán

Hiện tại, trí thông minh tính toán bao gồm:

- Mạng lưới thần kinh;
- Thuật toán di truyền hoặc, tổng quát, thuật toán tiến hóa;
- Các phương pháp xác suất như cập nhật Bayesian và các thành tố chắc chắn;
- Logic mờ;
- Kết hợp các kỹ thuật này với nhau và với Hệ thống dựa trên hiểu biết.

Tích hợp với các phần mềm khác

Những kỹ thuật này không nhất thiết đại diện cho các lựa chọn thay thế duy nhất, và thường có thể được sử dụng kết hợp với nhau.

Chương trình sẽ mô tả một số cách mà các kỹ thuật hệ thống thông minh có thể được sử dụng phối hợp trong các hệ thống lai ghép/kết hợp.

Chương trình sẽ mô tả một số ứng dụng thực tế, hầu hết trong số đó là lai ghép của một số dạng thức.

Nếu một vấn đề có thể được chia nhỏ thành các nhiệm vụ con, một hệ thống **bảng đen** có thể cung cấp một cách thích hợp để giải quyết nó.

Tích hợp với các phần mềm khác

Hệ thống **bảng đen** cho phép mỗi nhiệm vụ phụ được xử lý bằng một kỹ thuật thích hợp, từ đó đóng góp hiệu quả nhất cho giải pháp tổng thể.

Sử dụng KBS và CI không phải là giải pháp phù hợp với tất cả các loại vấn đề. Mỗi vấn đề đòi hỏi phải có công cụ thích hợp nhất, nhưng KBS và CI có thể được sử dụng cho nhiều vấn đề không thể thực hiện được bằng các công cụ khác.

Chương 2

Hệ thống dựa trên quy tắc/luật

Quy tắc và thực tế

Hệ thống dựa trên quy tắc là hệ thống dựa trên tri thức trong đó cơ sở tri thức được biểu diễn dưới dạng một tập hợp hoặc các tập hợp quy tắc.

Quy tắc là một phương tiện biểu đạt kiến thức. Loại quy tắc đơn giản nhất được gọi là *quy tắc dẫn xuất* và có dạng

```
IF <condition> THEN <conclusion>
```

Các quy tắc dẫn xuất thường có thể được viết ở dạng gần giống với ngôn ngữ tự nhiên, trái ngược với ngôn ngữ máy tính. Một quy tắc đơn giản như trên có thể hiểu được đối với bất kỳ ai hiểu tiếng Anh. Mặc dù các quy tắc có thể phức tạp hơn nhiều, nhưng bản chất rõ ràng của chúng vẫn khiến chúng dễ hiểu hơn mã máy tính thông thường.

Quy tắc và thực tế

Để sử dụng được các quy tắc trong một hệ thống dựa trên quy tắc, hệ thống sẽ cần có quyền truy cập vào các thực tế.

Thực tế là những tuyên bố vô điều kiện được giả định là đúng tại thời điểm chúng được sử dụng.

Thực tế có thể có từ:

- được tra cứu từ cơ sở dữ liệu;
- đã được lưu trữ trong bộ nhớ máy tính;
- xác định từ các cảm biến kết nối với máy tính;
- thu được bằng cách nhắc người dùng cung cấp thông tin;
- rút ra bằng cách áp dụng các quy tắc cho các sự kiện khác.

Quy tắc và thực tế

Thực tế có thể được coi là các quy tắc đặc biệt, trong đó phần điều kiện luôn đúng.

Thực tế mới được lưu trữ trong bộ nhớ máy tính và có thể được sử dụng để đáp ứng các điều kiện của quy tắc, do đó dẫn xuất đến các thực tế có nguồn gốc khác. Bộ sưu tập các thực tế đã được biết tại bất kỳ thời điểm nào trong hệ thống được gọi là cơ sở thực tế/dữ liệu.

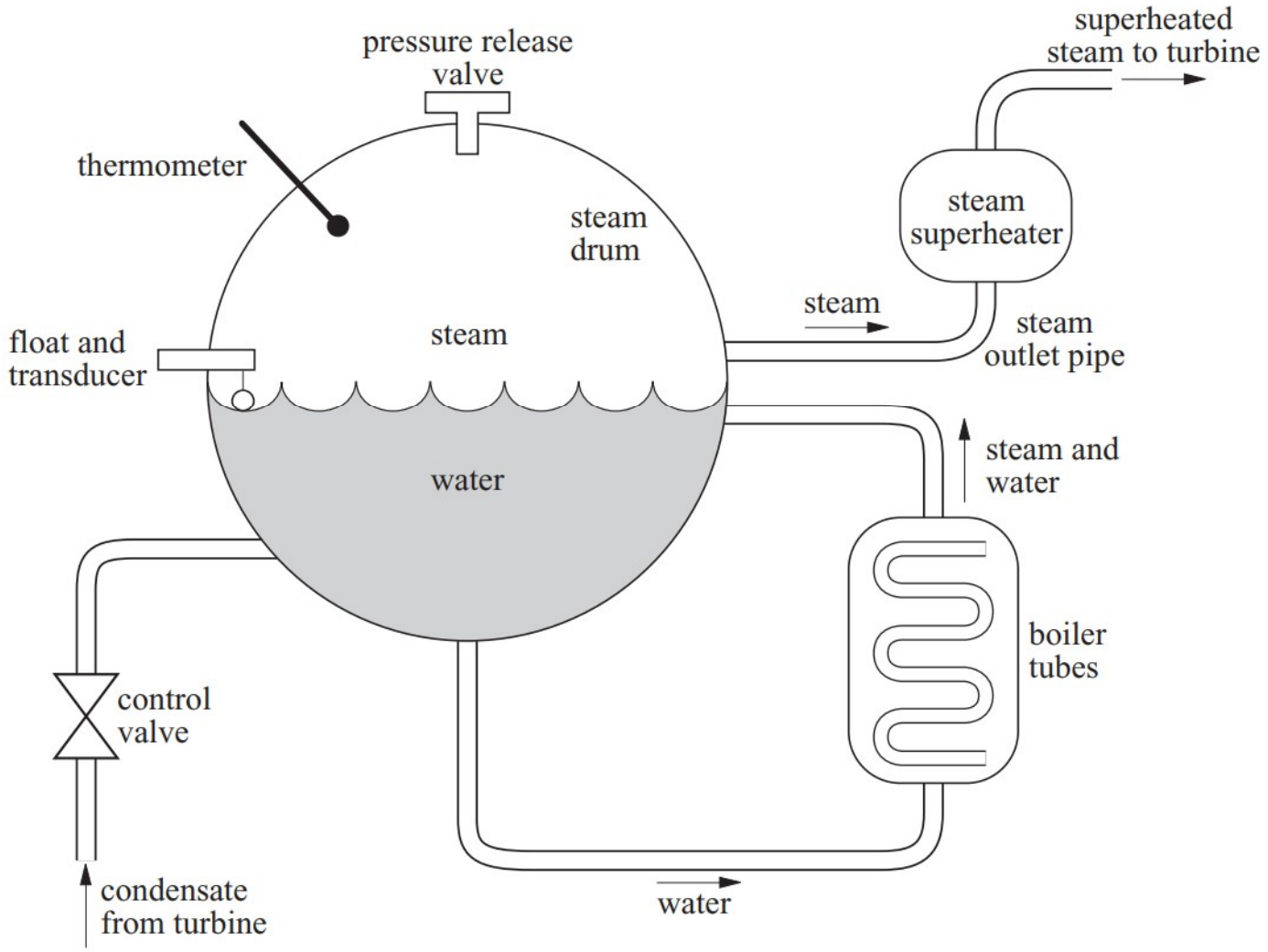
Viết quy tắc là một kiểu lập trình khai báo, vì các quy tắc đại diện cho kiến thức có thể được sử dụng bởi máy tính mà không cần chỉ rõ cách thức và thời điểm áp dụng kiến thức đó.

Quy tắc và thực tế

Thứ tự của các quy tắc trong một chương trình không quan trọng và có thể thêm mới các quy tắc hoặc sửa đổi những quy tắc hiện có mà không sợ tác dụng phụ.

Để các quy tắc và thực tế đã khai báo trở nên hữu ích, cần phải có một công cụ suy luận để giải thích và áp dụng chúng. Các công cụ suy luận được tích hợp vào một loạt các công cụ phần mềm, bao gồm các phần mềm hệ thống chuyên nghiệp, bộ công cụ trí tuệ nhân tạo, thư viện phần mềm và ngôn ngữ Prolog.

Ví dụ về một hệ thống dựa trên quy tắc



Ví dụ về một hệ thống dựa trên quy tắc

```
/* Rule 2.1 */
IF water level low THEN open control valve

/* Rule 2.2 */
IF temperature high AND water level low
THEN open control valve AND shut down boiler tubes

/* Rule 2.3 */
IF steam outlet blocked THEN replace outlet pipe

/* Rule 2.4 */
IF release valve stuck THEN steam outlet blocked

/* Rule 2.5 */
IF pressure high AND release valve closed
THEN release valve stuck

/* Rule 2.6 */
IF steam escaping THEN steam outlet blocked
```

Ví dụ về một hệ thống dựa trên quy tắc

```
/* Rule 2.7 */
IF temperature high AND NOT(water level low)
THEN pressure high

/* Rule 2.8 */
IF transducer output low THEN water level low

/* Rule 2.9 */
IF release valve open AND flow rate high
THEN steam escaping

/* Rule 2.10 */
IF flow rate low THEN control valve closed
```

Quy tắc và thực tế

Thực tế cấp thấp và thực tế cấp cao.

Quy tắc rộng (nông) và quy tắc sâu.

Quy tắc cấp thấp và quy tắc cấp cao.

Quy tắc cấp thấp thường phụ thuộc vào thực tế cấp thấp.

Quy tắc cấp cao tiến gần tới cung cấp một giải pháp cho vấn đề.

Kiểm tra quy tắc và kích hoạt quy tắc

Các công cụ suy luận giúp thông dịch và áp dụng các quy tắc. Quy trình áp dụng các quy tắc có thể được chia nhỏ như sau:

i) lựa chọn các quy tắc để kiểm tra - đây là những quy tắc có sẵn;

ii) xác định cái nào trong số này có thể áp dụng — đây là những quy tắc được kích hoạt; chúng tạo nên tập hợp xung đột;

iii) chọn một quy tắc để kích hoạt/chạy.

Duy trì tính nhất quán

Một ưu điểm chính của các hệ thống dựa trên quy tắc là tính linh hoạt của chúng. Các quy tắc mới có thể được được thêm tùy ý, nhưng chỉ khi mỗi quy tắc được viết cẩn thận và không giả định hành vi của các quy tắc khác vì khi quy tắc hoạt động, nó không đủ bền vững và khó bổ sung thêm kiến thức.

Nói chung, các quy tắc có thể phức tạp hơn đáng kể trong thực tế. Ví dụ: các quy tắc có thể chứa kết hợp các điều kiện và kết luận.

Giả định về thế giới đóng

Nếu không khẳng định được một mệnh đề đã cho là đúng, thì trong nhiều hệ thống dựa trên quy tắc, mệnh đề được cho là sai. Giả định này, được gọi là *giả định thế giới đóng*, đơn giản hóa logic cần thiết vì tất cả các mệnh đề chỉ có thể là ĐÚNG hoặc SAI.

Có hai biện pháp có thể được thực hiện để tránh sự mơ hồ này, đó là sửa đổi các quy tắc hoặc sửa đổi công cụ suy luận.

Nếu không sử dụng tiếp cận giả định về thế giới đóng, thì cần tiếp cận một loại thứ ba, cụ thể là UNKNOWN.

Sử dụng các biến trong quy tắc

Trong các hệ thống trong thực tế, thường sử dụng các biến để làm cho các quy tắc phổ quát hơn, do đó giảm số lượng các quy tắc cần thiết và giữ bộ quy tắc có thể quản lý được. Loại quy tắc thường được yêu cầu có dạng:

For all x, IF <condition about x> THEN <conclusion about x>

Trong trường hợp không thể dự đoán trước các giá trị có thể có của một biến, việc sử dụng biến trở nên cần thiết.
Việc sử dụng tên biến trong các quy tắc là không thể thiếu đối với ngôn ngữ Prolog. Trong Prolog, thay vì sử dụng dấu chấm hỏi, các biến được phân biệt với các hằng bằng cách có dấu gạch dưới hoặc chữ hoa làm ký tự đầu tiên của chúng. Các ngôn ngữ khác sẽ yêu cầu người dùng lập trình cơ sở này hoặc mua phần mềm phù hợp.

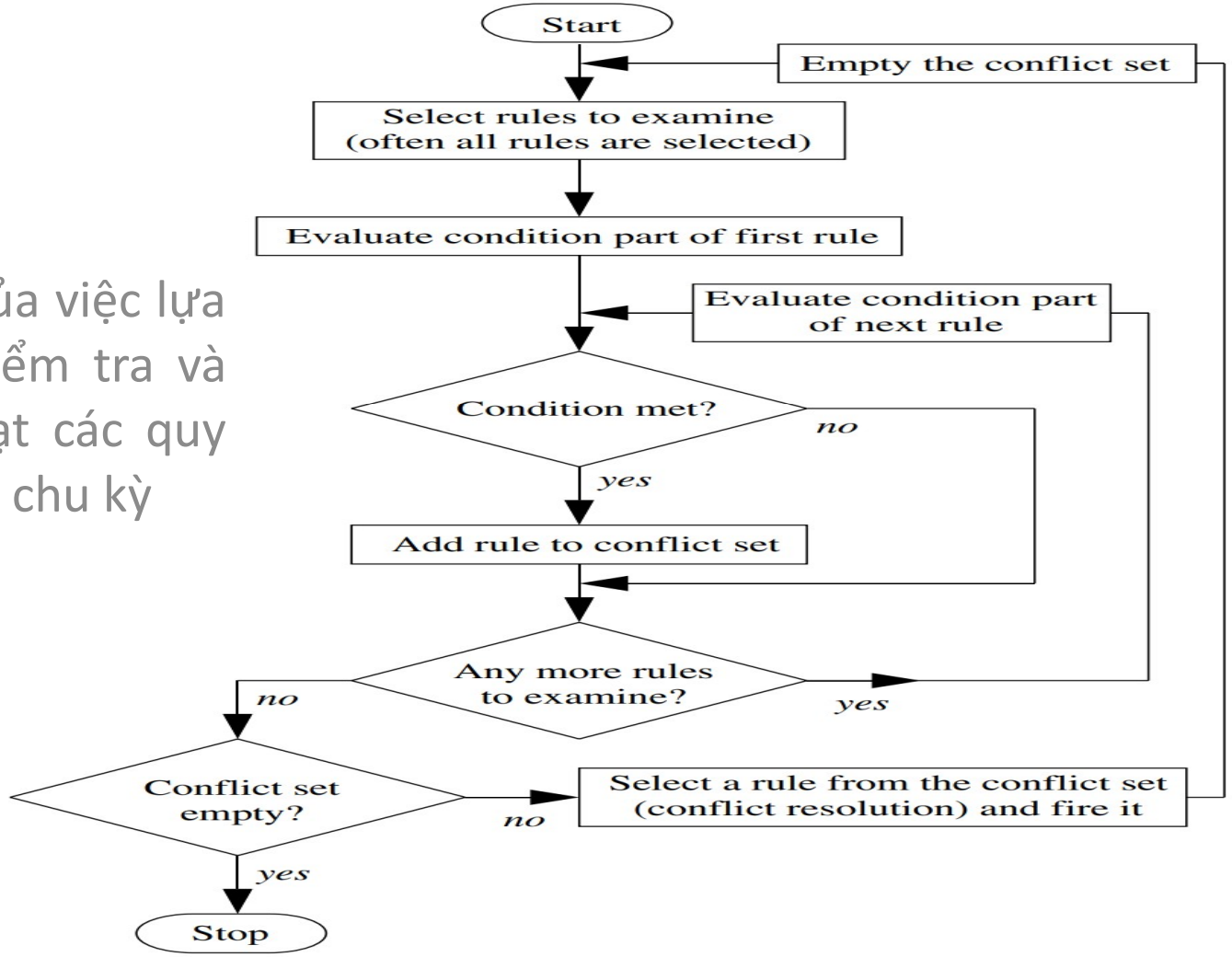
Truyền thuận (tiếp cận theo hướng dữ liệu)

Công cụ suy luận áp dụng một chiến lược để quyết định sử dụng các quy tắc nào và khi nào sử dụng chúng.

Truyền thuận là tên được đặt cho chiến lược hướng dữ liệu, tức là các quy tắc được chọn và áp dụng theo cơ sở thực tế hiện tại. Cơ sở dữ kiện bao gồm tất cả các thực tế mà hệ thống đã biết, cho dù được rút ra bởi các quy tắc hay được cung cấp trực tiếp.

Truyền thuận (tiếp cận theo hướng dữ liệu)

Sơ đồ của việc lựa chọn, kiểm tra và kích hoạt các quy tắc theo chu kỳ



Truyền thuận (tiếp cận theo hướng dữ liệu)

Các điểm chính của sơ đồ được thể hiện như sau:

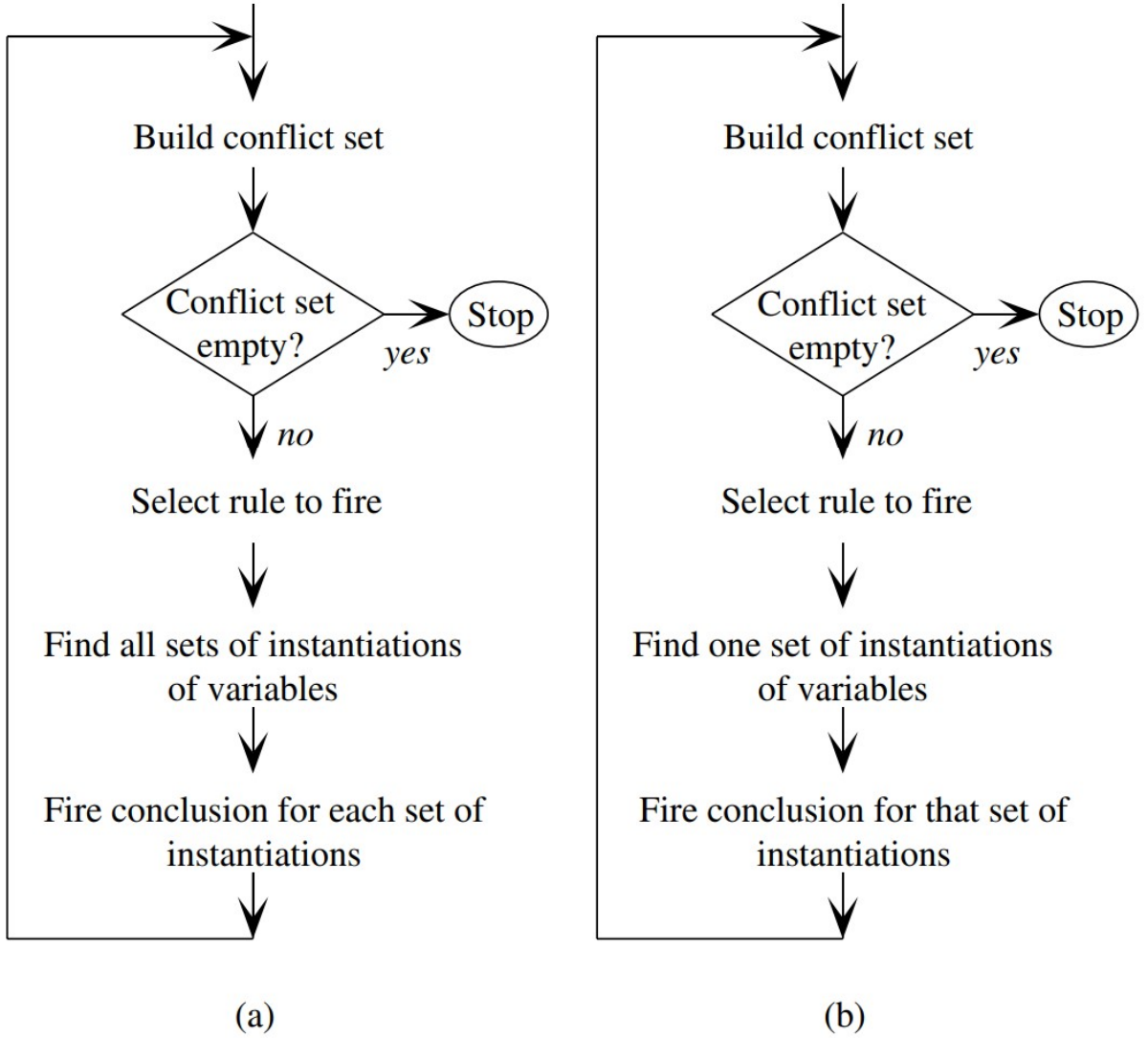
- Các quy tắc được kiểm tra và được kích hoạt dựa trên cơ sở thực tế hiện tại, độc lập với bất kỳ mục tiêu định trước nào;
- Bộ quy tắc có sẵn để kiểm tra có thể bao gồm tất cả các quy tắc hoặc một tập hợp con của quy tắc;
- Trong số các quy tắc có sẵn, những quy tắc có điều kiện được thỏa mãn được cho là đã được kích hoạt. Các quy tắc này tạo nên tập xung đột/mâu thuẫn và phương pháp chọn quy tắc từ tập xung đột là giải quyết xung đột/mâu thuẫn;
- Mặc dù tập xung đột có thể chứa nhiều quy tắc, nhưng chỉ một quy tắc được kích hoạt trong một chu kỳ nhất định. Điều này là do khi một quy tắc đã kích hoạt, kho lưu trữ có khả năng thay đổi, và do đó, không thể đảm bảo rằng các quy tắc khác trong tập hợp xung đột vẫn thỏa mãn các phần điều kiện của chúng.

Khởi tạo một biến và nhiều biến

Có nhiều lược đồ biến thể cho truyền thuận. Khi các biến được sử dụng trong các quy tắc, các kết luận có thể được thực hiện chỉ bằng cách sử dụng tập hợp đầu tiên của các khởi tạo được tìm thấy - đây là khởi tạo đơn lẻ.

Ngoài ra, các kết luận có thể được thực hiện lặp đi lặp lại bằng cách sử dụng tất cả các khởi tạo có thể có - đây là nhiều khởi tạo. Đa khởi tạo là cách tiếp cận theo chiều rộng để giải quyết vấn đề và đơn khởi tạo là cách tiếp cận theo chiều sâu.

Truyền thuận (tiếp cận theo hướng dữ liệu)



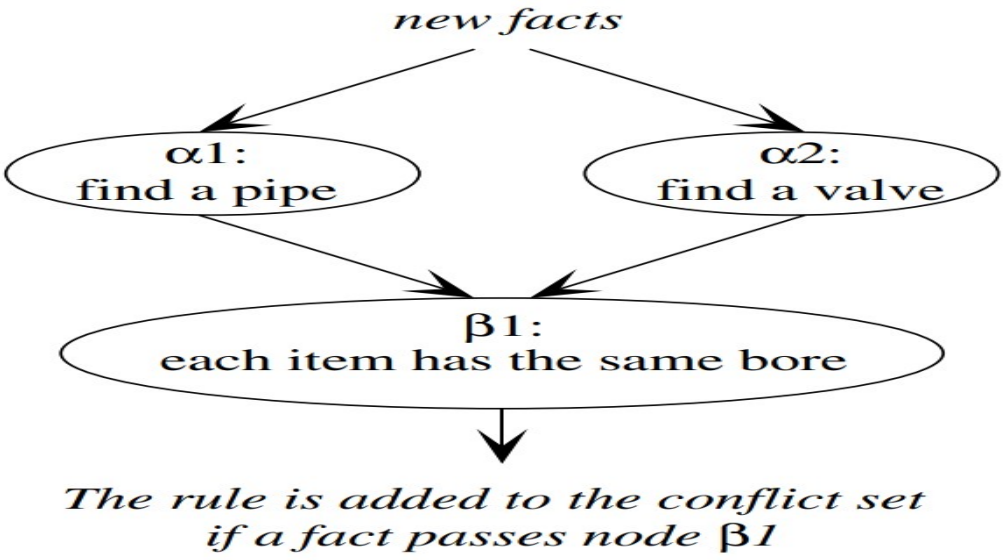
Thuật toán Rete

Khi một quy tắc đã được chọn từ tập hợp xung đột và được kích hoạt, tập hợp xung đột bị loại bỏ và quá trình bắt đầu lại từ đầu. Cái này là do việc kích hoạt một quy tắc làm thay đổi cơ sở dữ kiện, do đó, một bộ quy tắc khác có thể đủ điều kiện cho tập hợp xung đột. Dẫn đến, một tập hợp xung đột mới được hình thành bằng cách kiểm tra lại các phần điều kiện của tất cả các quy tắc có sẵn. Trong hầu hết các ứng dụng, việc kích hoạt một quy tắc chỉ tạo ra những thay đổi nhỏ đối với cơ sở thực tế và tới thành viên của tập hợp xung đột.

Một cách tiếp cận hiệu quả hơn sẽ chỉ kiểm tra những quy tắc mà điều kiện của nó bị ảnh hưởng bởi những thay đổi được thực hiện đối với cơ sở dữ kiện ở chu kỳ trước. Thuật toán Rete (phát âm là “ree-tee”) là một cách để đạt được điều này.

Thuật toán Rete

Trước khi chạy hệ thống, các thành phần điều kiện của tất cả các quy tắc được hợp vào một mạng Rete, trong đó mỗi nút đại diện cho một điều kiện nguyên tử, tức là một điều kiện chứa một thử nghiệm đơn giản. Có hai loại nút - nút alpha có thể được thỏa mãn bởi một thực tế duy nhất, trong khi các nút beta chỉ có thể được thỏa mãn bởi một cặp thực tế.



Thuật toán Rete

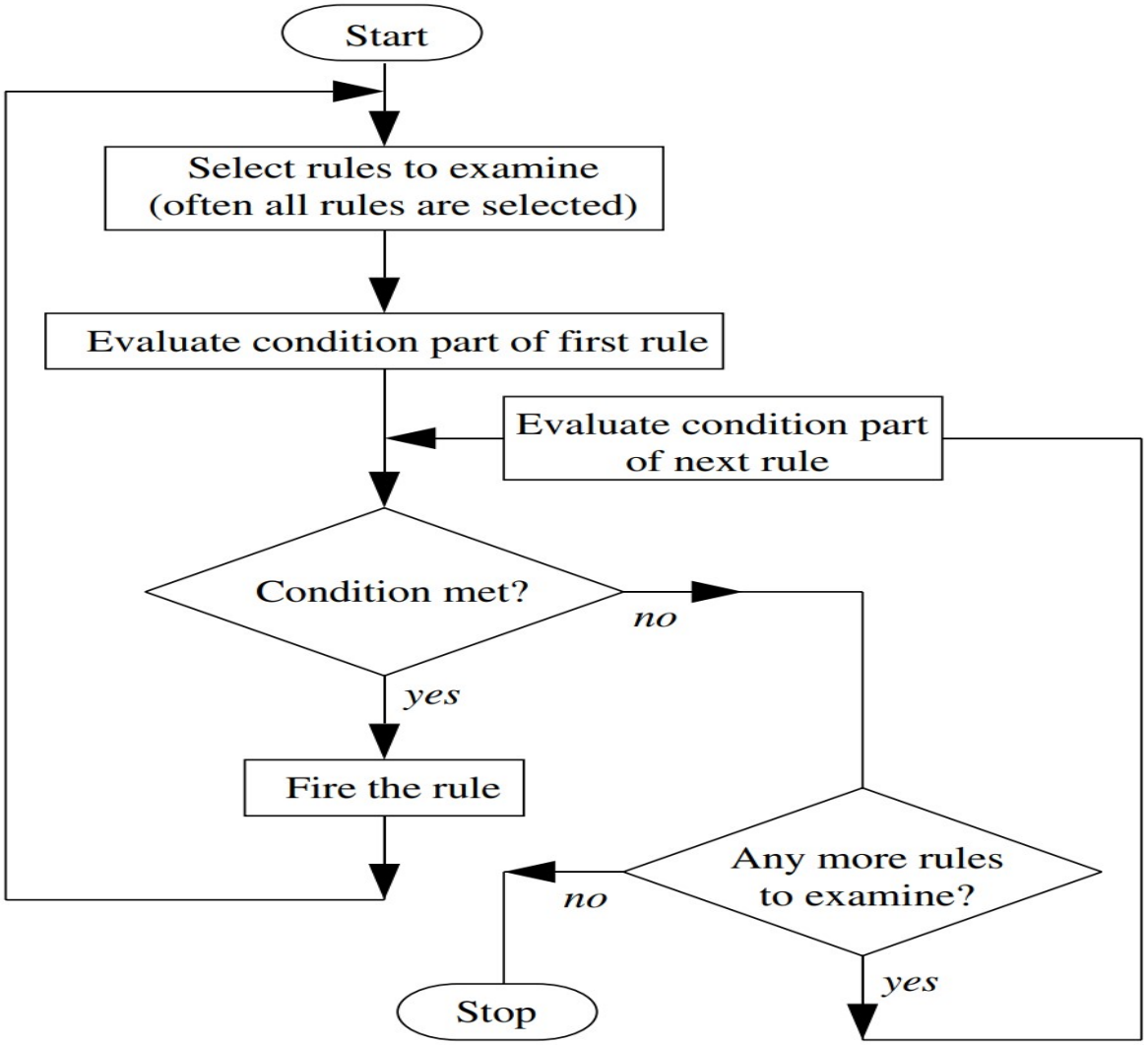
Một mạng Rete đầy đủ sẽ chứa các nút đại diện cho các điều kiện con của tất cả các quy tắc trong cơ sở quy tắc. Mỗi khi một quy tắc được kích hoạt, các thực tế đã thay đổi sẽ được đưa vào mạng và các thay đổi đối với tập hợp xung đột được tạo ra. Khi các quy tắc chứa các điều kiện con giống hệt nhau, các nút có thể được chia sẻ, do đó tránh việc kiểm tra trùng lặp các điều kiện. Trong một đánh giá của một số các bộ công cụ trí tuệ nhân tạo có sẵn trên thị trường sử dụng truyền thuận, những bộ tích hợp thuật toán Rete được phát hiện là cung cấp những cải tiến đáng kể về hiệu suất.

Giải quyết xung đột

Đến trước, phục vụ trước:

Giải quyết xung đột là phương pháp chọn một quy tắc để kích hoạt từ những quy tắc có thể kích hoạt, tức là từ tập hợp các quy tắc được kích hoạt, được gọi là tập hợp xung đột. Tập hợp xung đột hoàn chỉnh được tìm thấy trước khi chọn một quy tắc để kích hoạt. Vì chỉ một quy tắc từ tập hợp xung đột có thể thực sự kích hoạt trong một chu kỳ nhất định, nên thời gian dành để đánh giá các phần điều kiện của quy tắc khác quy tắc bị lãng phí trừ khi kết quả được lưu bằng cách sử dụng thuật toán Rete hoặc kỹ thuật tương tự. Một chiến lược khắc phục sự kém hiệu quả này là sa thải ngay lập tức quy tắc đầu tiên được tìm thấy đủ điều kiện cho tập hợp xung đột.

Giải quyết xung đột



Giải quyết xung đột

Giá trị ưu tiên:

Thay vì dựa vào thứ tự quy tắc như một phương tiện xác định mức độ ưu tiên của quy tắc, các quy tắc có thể được viết để mỗi quy tắc có một giá trị ưu tiên được nêu rõ ràng. Khi có nhiều hơn một quy tắc có thể kích hoạt, quy tắc được chọn là quy tắc có mức độ ưu tiên cao nhất.

Vì việc kiểm tra các quy tắc không được kích hoạt thể hiện sự lãng phí, việc sử dụng hiệu quả các ưu tiên sẽ là lựa chọn các quy tắc kiểm tra theo thứ tự ưu tiên. Khi một quy tắc được tìm thấy có thể kích hoạt, nó có thể được kích hoạt ngay lập tức.

Quy tắc của quy tắc/siêu quy tắc

Siêu quy tắc là các quy tắc không liên quan cụ thể đến kiến thức về ứng dụng hiện tại, mà là kiến thức về cách kiến thức đó nên được áp dụng. Do đó, Metarules là “quy tắc về quy tắc” (hoặc hơn thế nữa nói chung, "các quy tắc về kiến thức").

Truyền ngược (tiếp cận dựa vào mục tiêu)

Cơ chế truyền ngược:

Truyền ngược là một chiến lược suy luận giả định sự tồn tại của một mục tiêu cần được thiết lập hoặc bác bỏ.

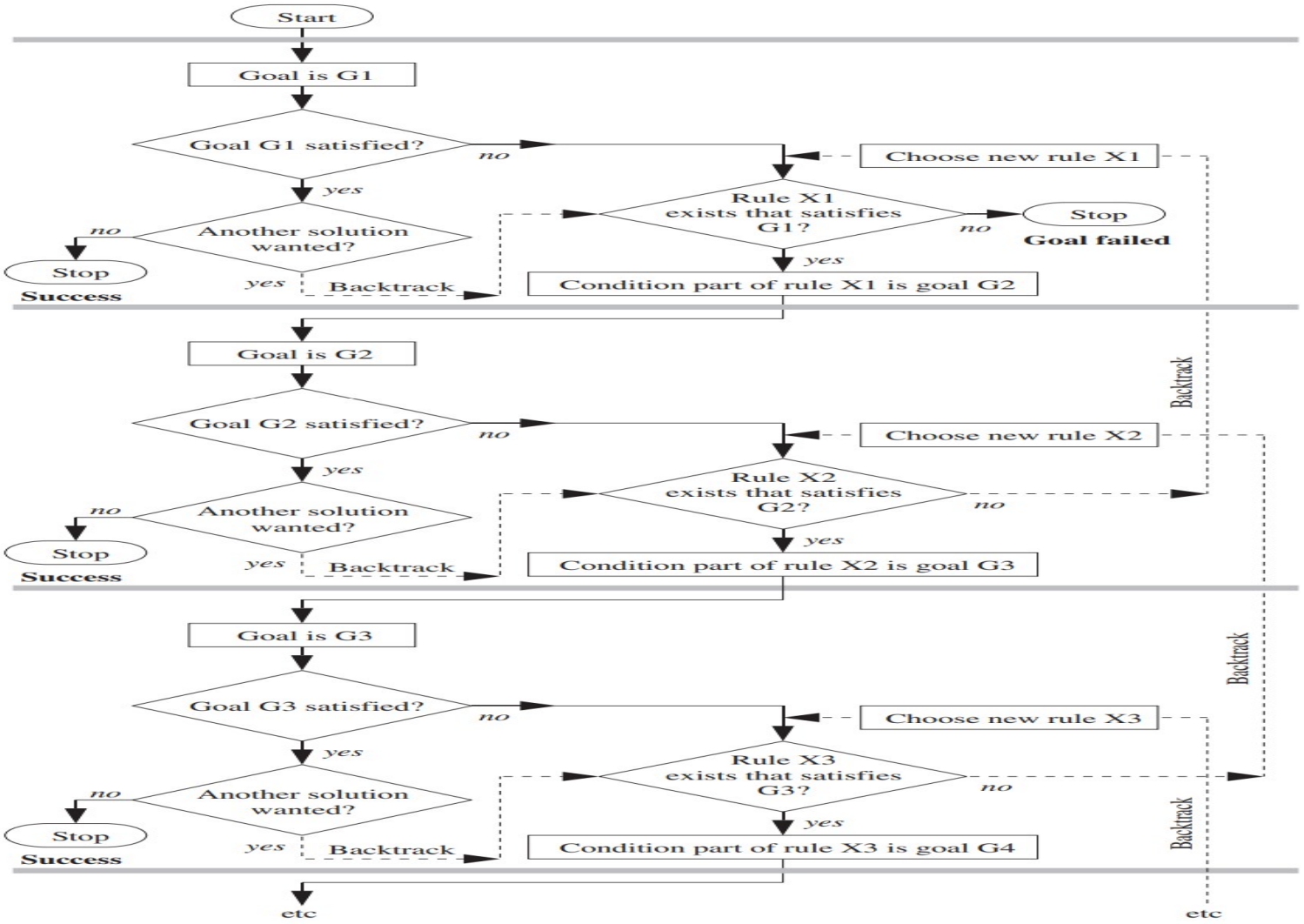
Cơ chế truyền ngược giả định là tìm kiếm quy tắc theo chiều sâu. Điều này có nghĩa là bất cứ khi nào có sự lựa chọn giữa các quy tắc, chỉ một cái được chọn, những cái khác chỉ được kiểm tra nếu xảy ra truyền ngược.

Cơ chế suy luận được tích hợp trong ngôn ngữ Prolog là một truyền ngược theo chiều sâu-trước tiên. Có hai bộ về các trường hợp truyền ngược diễn ra:

i) khi một mục tiêu không thể được thỏa mãn bởi bộ quy tắc hiện đang được xem xét; hoặc

ii) khi một mục tiêu đã được thỏa mãn và muốn điều tra các cách khác để đạt được mục tiêu (tức là tìm các giải pháp khác).

Truyền ngược (tiếp cận dựa vào mục tiêu)



Một số lưu ý khi thực hiện truyền ngược

Để đơn giản hóa có giả định rằng mỗi quy tắc chỉ có một điều kiện, do đó việc thỏa mãn một điều kiện có thể được biểu diễn dưới dạng một mục tiêu duy nhất.

Trong thực tế, các quy tắc có nhiều hơn một điều kiện.

Truyền ngược là một quá trình lặp đệ quy. Do vậy, mục tiêu khó đạt được, khi độ dài của chuỗi quy tắc không thể được xác định trước.

Định nghĩa đệ quy của một hàm là một định nghĩa bao gồm chính hàm đó. Đệ quy là một khía cạnh quan trọng của ngôn ngữ trí tuệ nhân tạo Lisp và Prolog cũng như nhiều ngôn ngữ máy tính khác.

Các biến thể của truyền ngược

i. Tìm kiếm quy tắc theo chiều sâu hoặc theo chiều rộng;
ii. Có hay không theo đuổi các giải pháp khác (tức là, các phương tiện khác để đạt được mục tiêu) sau khi một giải pháp đã được tìm thấy;
iii. Các cách khác nhau để lựa chọn giữa các nhánh để khám phá (chỉ tìm kiếm theo chiều sâu);
iv. Quyết định thứ tự kiểm tra các quy tắc (chỉ tìm kiếm theo chiều rộng-ưu tiên);
v. Đã tìm thấy chuỗi liên tiếp các quy tắc cho phép các điều kiện cấp thấp nhất được thỏa mãn, có nên kích hoạt trình tự các quy tắc hay chỉ đơn giản là để kết luận rằng mục tiêu đã được minh chứng.

Các biến thể của truyền ngược

Truyền ngược theo chiều rộng giống hệt với theo chiều sâu, ngoại trừ cơ chế quyết định giữa các quy tắc thay thế để kiểm tra.

Giải pháp đầu tiên được tìm thấy sẽ là giải pháp có chuỗi quy tắc ngắn nhất (hoặc kết hợp ngắn nhất). Một bất lợi với phương pháp truyền ưu tiên theo chiều rộng là cần một lượng lớn bộ nhớ máy tính, vì hệ thống cần lưu giữ một bản ghi tiến trình dọc theo tất cả các nhánh của cây tìm kiếm, thay vì chỉ một nhánh.

Mục (v) trong danh sách các biến thể của truyền ngược là quan trọng nhưng không kém phần tinh tế. Nếu một mục tiêu có khả năng được thỏa mãn bởi một loạt các quy tắc, quy tắc đầu tiên có thể thực hiện được, thì mục tiêu được giả định là đúng và không có quy tắc nào thực sự bị sa thải. Nếu tất cả những gì cần thiết là xác nhận mục tiêu, thì cách tiếp cận này là phù hợp.

Các biến thể của truyền ngược

Trong một số triển khai truyền ngược, khi đường dẫn đến mục tiêu đã được thiết lập, các quy tắc sẽ được kích hoạt cho đến khi mục tiêu ban đầu được hoàn thành.

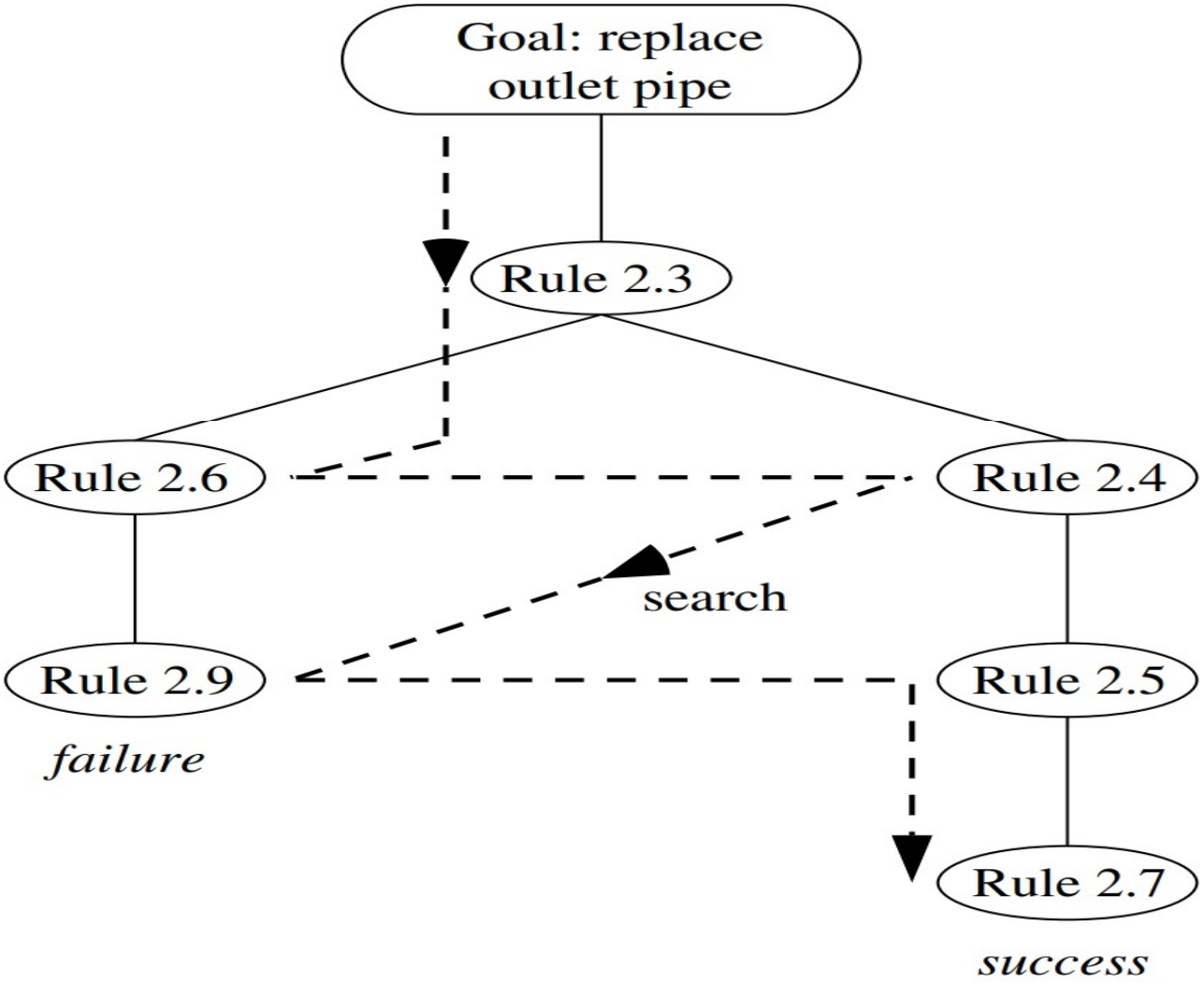
Trong các hệ thống dựa trên quy tắc phức tạp hơn, các quy tắc có thể gọi và chạy một mô-đun được mã hóa theo thủ tục như một phần trong kết luận của chúng. Trong các hệ thống như vậy, việc kích hoạt các quy tắc là điều cần thiết để vai trò của mã thủ tục trở thành đã nhận ra.

Cần lưu ý rằng trong một số hệ thống truyền ngược, cú pháp quy tắc bị đảo ngược, so với các ví dụ mà chúng ta đã thảo luận cho đến nay

```
DEDUCE <conclusion> IF <condition>
```

Việc đặt kết luận trước điều kiện phản ánh thực tế trong các hệ thống truyền ngược, là phần kết luận của quy tắc được đánh giá đầu tiên và chỉ khi kết luận có liên quan thì điều kiện mới được kiểm tra.

Các biến thể của truyền ngược



Một chiến lược lai ghép/kết hợp

Một hệ thống được gọi là ARBS (Hệ thống bảng đen dựa trên quy tắc và thuật toán), sử dụng công cụ suy luận có thể được coi là cách kết hợp một phần truyền thuận và một phần truyền ngược.

Truyền ngược thông thường liên quan đến việc kiểm tra ban đầu một quy tắc đạt được mục tiêu. Nếu điều đó không thể kích hoạt, các quy tắc tiền trước của nó là đệ quy được kiểm tra, cho đến khi các quy tắc có thể được áp dụng và đạt được tiến bộ đối với mục tiêu. Trong tất cả các các vấn đề liên quan đến diễn giải dữ liệu, các quy tắc cấp cao liên quan đến bản thân mục tiêu không bao giờ có thể kích hoạt cho đến khi các quy tắc cấp thấp hơn để thao tác dữ liệu đã được kích hoạt. Do đó, các cơ chế tiêu chuẩn cho truyền thuận hoặc truyền ngược liên quan đến rất nhiều việc kiểm tra quy tắc dự phòng. Chiến lược kết hợp là một phương tiện để loại bỏ nguồn kém hiệu quả này.

Một chiến lược lai ghép/kết hợp

Theo chiến lược lai ghép/kết hợp, một mạng lưới phụ thuộc quy tắc được xây dựng trước khi chạy hệ thống.

Đối với mỗi quy tắc, mạng hiển thị những quy tắc nào khác có thể kích hoạt nó, tức là tiền thân của nó và những quy tắc nào nó có thể bật, tức là những quy tắc phụ thuộc của nó.

Trong chế độ hướng dữ liệu, được gọi là truyền thuận có hướng, chiến lược lai ghép đạt được cải thiện bằng cách sử dụng mạng phụ thuộc để chọn các quy tắc cho kiểm tra. Quy tắc cấp thấp liên quan đến dữ liệu cảm biến được chọn ban đầu để kiểm tra. Sau đó, các quy tắc cấp cao hơn, dẫn đến một giải pháp, được chọn tùy thuộc vào quy tắc nào đã thực sự kích hoạt.

Kỹ thuật này là một cách hiệu quả để thực hiện chỉ dẫn được đánh dấu "chọn các quy tắc để kiểm tra" trong sơ đồ truyền thuận.

Một chiến lược lai ghép/kết hợp

Cơ chế tương tự có thể dễ dàng được điều chỉnh để cung cấp một chiến lược hướng tới mục tiêu hiệu quả. Với một mục tiêu cụ thể, cơ chế kiểm soát có thể chọn nhánh của mạng lưới phụ thuộc dẫn đến mục tiêu đó và sau đó truyền ngược thông qua các quy tắc đã chọn.

Đối với một cơ sở quy tắc nhất định, mạng phụ thuộc chỉ cần được tạo một lần và sau đó có sẵn cho hệ thống tại thời điểm chạy. Thứ tự của các quy tắc trong cơ sở quy tắc không ảnh hưởng đến hệ thống, bởi vì việc áp dụng các quy tắc được quyết định bởi vị trí của chúng trong mạng lưới phụ thuộc chứ không phải trong tập quy tắc.

Sử dụng mạng phụ thuộc để chỉ đạo việc kiểm tra và kích hoạt quy tắc là một cách để đảm bảo thứ tự các sự kiện này.

Việc sử dụng mạng phụ thuộc phức tạp hơn khi các biến được sử dụng trong các quy tắc vì sự phụ thuộc giữa các quy tắc ít chắc chắn hơn.

Các phương tiện giải thích

Một trong những tuyên bố thường được đưa ra để hỗ trợ các hệ thống chuyên gia là chúng có thể giải thích lý do của chúng và điều này mang lại cho người sử dụng các hệ thống như vậy sự tin tưởng vào tính chính xác hoặc sự khôn ngoan trong các quyết định của hệ thống. Tuy nhiên, các giải thích được đưa ra bởi nhiều hệ thống không chỉ là dấu vết của việc áp dụng các quy tắc. Mặc dù đây là một cơ sở quan trọng, nhưng việc truy tìm luồng của một chương trình máy tính là hoạt động tiêu chuẩn chứ không phải là một khả năng đặc biệt.

Phương tiện giải thích có thể được chia thành hai loại:

- Kết luận đã được rút ra như thế nào;
- Tại sao một dòng lập luận cụ thể đang được tuân theo.

Các phương tiện giải thích

Loại giải thích đầu tiên thường được áp dụng khi hệ thống đã hoàn thành suy luận của nó, trong khi loại thứ hai có thể áp dụng trong khi hệ thống đang thực hiện quá trình lập luận của nó.

Loại giải thích thứ hai đặc biệt thích hợp trong một hệ thống chuyên gia tương tác, liên quan đến đối thoại giữa người dùng và máy tính. Trong một cuộc đối thoại như vậy, người dùng thường sẽ muốn xác định lý do tại sao các câu hỏi cụ thể được đặt ra. Nếu một trong hai loại giải thích là không chính xác hoặc không thể xem xét, người dùng có thể mất lòng tin hoặc bỏ qua các phát hiện của hệ thống.

Sử dụng các phương tiện giải thích để tăng độ tin cậy của người dùng vào hệ thống, như một công cụ hỗ trợ giảng dạy và như một sự trợ giúp để gỡ lỗi. Chất lượng giải thích có thể được cải thiện bằng cách đặt ra nghĩa vụ cho người viết quy tắc cung cấp ghi chú giải thích cho mỗi quy tắc. Sau đó, những ghi chú này có thể được đưa vào truy vết quy tắc hoặc được sao chép tại thời điểm chạy để giải thích dòng suy luận hiện tại. Các cơ sở giải thích cũng có thể phù hợp hơn bằng cách cung cấp cho người dùng mức độ chi tiết phù hợp với nhu cầu của họ.

Nhận xét

Các quy tắc là một cách hiệu quả để biểu diễn tri thức trong nhiều lĩnh vực ứng dụng. Chúng linh hoạt nhất khi các biến được sử dụng trong các quy tắc và chúng có thể đặc biệt hữu ích khi hợp tác với các thuật toán thủ tục hoặc hệ thống hướng đối tượng. ***Vai trò diễn giải, lựa chọn và áp dụng các quy tắc được thực hiện bởi công cụ suy luận***. Việc viết quy tắc lý tưởng nên độc lập với các chi tiết của công cụ suy luận, ngoài việc đáp ứng các yêu cầu về cú pháp của nó. Trong thực tế, người viết quy tắc cần phải biết chiến lược áp dụng các quy tắc và bất kỳ giả định nào do công cụ suy luận đưa ra. Ví dụ, trong giả định thế giới đóng, bất kỳ sự kiện nào không được cung cấp hoặc suy ra đều được cho là sai. Truyền thuận và truyền ngược là hai chiến lược riêng biệt để áp dụng các quy tắc, nhưng cũng có thể có nhiều biến thể của hai chiến lược này.

Chương 3

Ứng phó với sự không chắc chắn

Nguồn gốc của sự không chắc chắn

Trong phần trước khi thảo luận về các hệ thống dựa trên quy tắc đã giả định rằng chúng ta đang sống trong một thế giới rõ ràng, nơi mọi giả thuyết đều đúng, sai hoặc chưa biết. Hơn nữa, nhiều hệ thống có thể sử dụng giả định thế giới đóng, theo đó bất kỳ giả thuyết nào chưa biết đều được giả định là sai. Sau cùng, có thể một hệ thống là nhị phân, nơi mọi thứ đều đúng hay sai. Mặc dù mô hình thực tế này hữu ích trong nhiều ứng dụng, nhưng thực tế các quy trình lập luận hiếm khi rõ ràng như vậy.

Nguồn gốc của sự không chắc chắn

Có ba dạng khác biệt của sự không chắc chắn có thể liên quan đến các quy tắc:

1. Sự không chắc chắn trong chính các quy tắc.
2. Bằng chứng không chắc chắn là bằng chứng mà quy tắc dựa trên đó có thể không chắc chắn. Có hai lý do giải thích cho sự không chắc chắn này là: bằng chứng có thể đến từ một nguồn không hoàn toàn đáng tin cậy, bản thân bằng chứng có thể đã được rút ra bởi một quy tắc mà kết luận của nó có thể xảy ra hơn là chắc chắn.
3. Sử dụng ngôn ngữ mơ hồ không rõ ràng.

Nguồn gốc của sự không chắc chắn

Điều quan trọng là phải phân biệt giữa các nguồn không chắc chắn này, vì chúng cần được xử lý theo cách khác nhau. Có một số tình huống trong tự nhiên thực sự ngẫu nhiên và kết quả của nó, mặc dù không chắc chắn, có thể được dự đoán trên cơ sở thống kê.

Một số kỹ thuật mà chúng ta sẽ thảo luận dựa trên lý thuyết xác suất. Những điều này giả định rằng một cách tiếp cận thống kê có thể được áp dụng, mặc dù giả định này sẽ chỉ là một ước tính gần đúng với các trường hợp thực tế trừ khi vấn đề thực sự là ngẫu nhiên.

Nguồn gốc của sự không chắc chắn

Chương này sẽ xem xét một số kỹ thuật thường được sử dụng để lập luận với sự không chắc chắn. Cập nhật Bayesian có nguồn gốc nghiêm ngặt dựa trên lý thuyết xác suất, nhưng các giả định cơ bản của nó, ví dụ: tính độc lập thống kê của nhiều phần bằng chứng, có thể không đúng trong thực tế các tình huống. Lý thuyết độ chắc chắn không có cơ sở toán học chặt chẽ, nhưng đã được đưa ra như một cách thực tế để khắc phục một số hạn chế của cập nhật Bayes. Lý thuyết khả năng, hay logic mờ, cho phép sử dụng dạng thứ ba của sự không chắc chắn, tức là ngôn ngữ mơ hồ, được sử dụng một cách chính xác. Các giả định và sự tùy tiện của các kỹ thuật khác nhau có nghĩa là lý luận về sự không chắc chắn vẫn còn là một vấn đề gây tranh cãi.

Cập nhật Bayesian

Biểu diễn sự không chắc chắn bằng xác suất

Cập nhật Bayesian giả định rằng có thể gán một xác suất cho mọi giả thuyết hoặc khẳng định, và xác suất có thể được cập nhật dựa trên bằng chứng cho hoặc chống lại một giả thuyết hoặc khẳng định. Việc cập nhật này có thể sử dụng trực tiếp định lý Bayes hoặc có thể được đơn giản hóa một chút bằng cách tính toán tỷ lệ khả năng xảy ra. Một trong những ứng dụng thành công sớm nhất của việc cập nhật Bayesian cho các hệ thống chuyên gia là PROSPECTOR, một hệ thống hỗ trợ khảo sát khoáng sản bằng cách giải thích dữ liệu địa chất.

Cập nhật Bayesian

Ứng dụng trực tiếp định lý Bayes

Kỹ thuật cập nhật Bayes cung cấp một cơ chế cập nhật xác suất của giả thuyết P(H) khi có bằng chứng E. Thường thì bằng chứng là một triệu chứng và giả thuyết là một chẩn đoán. Kỹ thuật này dựa trên ứng dụng của định lý Bayes (đôi khi được gọi là quy tắc Bayes). Định lý Bayes cung cấp một biểu thức cho xác suất có điều kiện P(H|E) của giả thuyết H đưa ra một số bằng chứng E, theo P(E|H), tức là xác suất có điều kiện của E cho trước H:

$$P(H|E) = \frac{P(H) \times P(E|H)}{P(E)}$$

Cập nhật Bayesian

Ứng dụng trực tiếp định lý Bayes

Định lý Bayes sau đó có thể được mở rộng như sau:

$$P(H|E) = \frac{P(H) \times P(E|H)}{P(H) \times P(E|H) + P(\sim H) \times P(E|\sim H)}$$

$$P(\sim H) = 1 - P(H)$$

Cập nhật Bayesian

Ứng dụng trực tiếp định lý Bayes

Phương trình trên cung cấp một cơ chế để cập nhật xác suất của giả thuyết H dựa trên bằng chứng mới E.

Điều này được thực hiện bằng cách cập nhật giá trị hiện có của P(H) thành giá trị của P(H | E) được sinh ra bởi Phương trình trên.

Việc áp dụng phương trình cần có kiến thức về các giá trị sau:

- P(H), xác suất hiện tại của giả thuyết. Nếu đây là lần cập nhật đầu tiên cho giả thuyết này, thì P(H) là xác suất trước.
- P(E | H), xác suất có điều kiện mà bằng chứng có mặt, cho rằng giả thuyết là đúng.
- P(E | ~ H), xác suất có điều kiện mà bằng chứng có mặt, cho rằng giả thuyết là sai.

Cập nhật Bayesian

Ứng dụng trực tiếp định lý Bayes

Do đó, để xây dựng một hệ thống sử dụng trực tiếp định lý Bayes theo cách này, cần có trước các giá trị P(H), P(E | H) và P(E | ~ H) cho tất cả các giả thuyết khác nhau và bằng chứng được bao phủ bởi các quy tắc. Việc lấy các giá trị này thoạt nhìn có vẻ phức tạp hơn so với biểu thức có thể tính toán được, cụ thể là P(H | E).

Tuy nhiên, trong trường hợp các vấn đề về chẩn đoán, xác suất có điều kiện của bằng chứng, được đưa ra một giả thuyết, thường sẵn có hơn xác suất có điều kiện của một giả thuyết, được đưa ra bằng chứng. Ngay cả khi P(E | H) và P(E | ~ H) không có sẵn dưới dạng quan sát thống kê chính thức, thì ít nhất chúng cũng có thể có sẵn dưới dạng ước tính không chính thức.

Cập nhật Bayesian

Ứng dụng trực tiếp định lý Bayes

Vì vậy, ở ví dụ trong sách, một chuyên gia có thể có một số ý tưởng về tần suất hơi nước được quan sát thấy thoát ra khi có tắc nghẽn đường thoát nước, nhưng ít có khả năng biết mức độ thường xuyên thoát hơi nước do tắc nghẽn đường ống thoát nước. Chương 1 đã giới thiệu các ý tưởng về Loại trừ, Phân rã và Quy nạp. Trên thực tế, định lý Bayes thực hiện Phân rã (tức là xác định nguyên nhân) bằng cách sử dụng thông tin Loại trừ (tức là khả năng xảy ra các triệu chứng, ảnh hưởng hoặc bằng chứng). Tiền đề cho rằng thông tin Loại trừ sẵn có hơn thông tin Phân rã là một trong những lý do giải thích cho việc sử dụng cập nhật Bayes.

Cập nhật Bayesian

Tỷ lệ khả năng xẩy ra

Tỷ lệ khả năng xảy ra cung cấp một phương tiện thay thế để đại diện cho cập nhật Bayes.

Tỷ lệ cược O(H) của một giả thuyết H nhất định có liên quan đến xác suất P (H) của nó theo các quan hệ:

$$O(H) = \frac{P(H)}{P(\sim H)} = \frac{P(H)}{1 - P(H)}$$

$$P(H) = \frac{O(H)}{O(H) + 1}$$

Cập nhật Bayesian

Tỷ lệ khả năng xẩy ra

Do đó, một giả thuyết với xác suất 0.2 có tỷ lệ cược 0.25 (hoặc "4 ăn 1 chống lại"). Tương tự, một giả thuyết với xác suất là 0.8 có tỷ lệ cược là 4 (hoặc "4 ăn 1"). Một khẳng định hoàn toàn chắc chắn, tức là có xác suất bằng 1, có tỷ lệ cược vô hạn. Trong thực tế, các giới hạn thường được thiết lập trên các giá trị tỷ lệ cược sao cho, ví dụ, nếu O(H)> 106 thì H là đúng, và nếu O(H) <106 thì H là sai. Giới hạn như vậy là tùy ý.

Để suy ra các phương trình cập nhật, hãy bắt đầu bằng cách xem xét giả thuyết "không phải H" hoặc ~ H:

$$P(\sim H|E) = \frac{P(\sim H) \times P(E|\sim H)}{P(E)} \qquad O(H|E) = \frac{P(H|E)}{P(\sim H|E)}$$

Cập nhật Bayesian

Tỷ lệ khả năng xẩy ra

O(H | E) là tỷ lệ cược cập nhật của H, với sự hiện diện của bằng chứng E, và A là trọng số khẳng định của bằng chứng E. Đây là một trong hai tỷ lệ khả năng xảy ra. Loại còn lại là trọng lượng từ chối D của bằng chứng E. Trọng lượng từ chối có thể thu được bằng cách xem xét sự vắng mặt của bằng chứng, tức là, ~ E:

$$O(H|{\sim}E) = D \times O(H) \quad D = \frac{P({\sim}E|H)}{P({\sim}E|{\sim}H)} = \frac{1-P(E|H)}{1-P(E|{\sim}H)}$$

Thay vì hiển thị các giá trị tỷ lệ cược, có phạm vi vô hạn, các xác suất tương ứng đã được hiển thị. Trọng lượng (A hoặc D) đã được hiển thị trên thang logarit trong phạm vi từ 0.01 đến 100.

Cập nhật Bayesian

Sử dụng tỷ lệ khả năng xẩy ra

Trong nhiều trường hợp, việc không có một phần bằng chứng hỗ trợ có thể làm giảm khả năng xảy ra một giả thuyết nhất định. Nói cách khác, sự vắng mặt của bằng chứng hỗ trợ tương đương với sự có mặt của bằng chứng đối lập. Sự vắng mặt đã biết của bằng chứng khác với việc không biết liệu bằng chứng có hiện diện hay không và có thể được sử dụng để giảm xác suất (hoặc tỷ lệ cược) của giả thuyết bằng cách áp dụng trọng số phủ nhận, D.

Nếu một vật chứng E đã cho có trọng lượng khẳng định A lớn hơn 1, thì trọng lượng phủ nhận của nó phải nhỏ hơn 1 và ngược lại:

A> 1 ngụ ý D <1,

A <1 ngụ ý D> 1. Không có bằng chứng ủng hộ giả thuyết.

Cập nhật Bayesian

Sử dụng tỷ lệ khả năng xẩy ra

Giống như việc áp dụng trực tiếp quy tắc Bayes, tỷ lệ khả năng có lợi thế là các định nghĩa của A và D được hiểu theo xác suất có điều kiện của bằng chứng, được đưa ra một giả thuyết, thay vì ngược lại. Như đã chỉ ra ở trên, người ta thường giả định rằng thông tin này sẵn có hơn xác suất có điều kiện của một giả thuyết, với bằng chứng, ít nhất là theo cách không chính thức. Ngay cả khi không có xác suất có điều kiện chính xác, việc cập nhật Bayesian bằng cách sử dụng tỷ lệ khả năng vẫn là một kỹ thuật hữu ích nếu các giá trị tìm kiếm có thể được gắn với A và D.

Cập nhật Bayesian

Xử lý bằng chứng không chắc chắn

Trong trường hợp giả định rằng bằng chứng chắc chắn có mặt (tức là có xác suất bằng 1) hoặc chắc chắn không có (tức là có xác suất bằng 0). Nếu xác suất của bằng chứng nằm giữa các điểm cực trị này, thì độ tin cậy trong kết luận phải được chia tỷ lệ một cách thích hợp. Có hai lý do tại sao bằng chứng có thể không chắc chắn:

• bằng chứng có thể là một khẳng định được tạo ra bởi một quy tắc không chắc chắn khác và do đó có một xác suất đi kèm với nó;

• bằng chứng có thể ở dạng dữ liệu không hoàn toàn đáng tin cậy, chẳng hạn như đầu ra từ cảm biến.

Cập nhật Bayesian

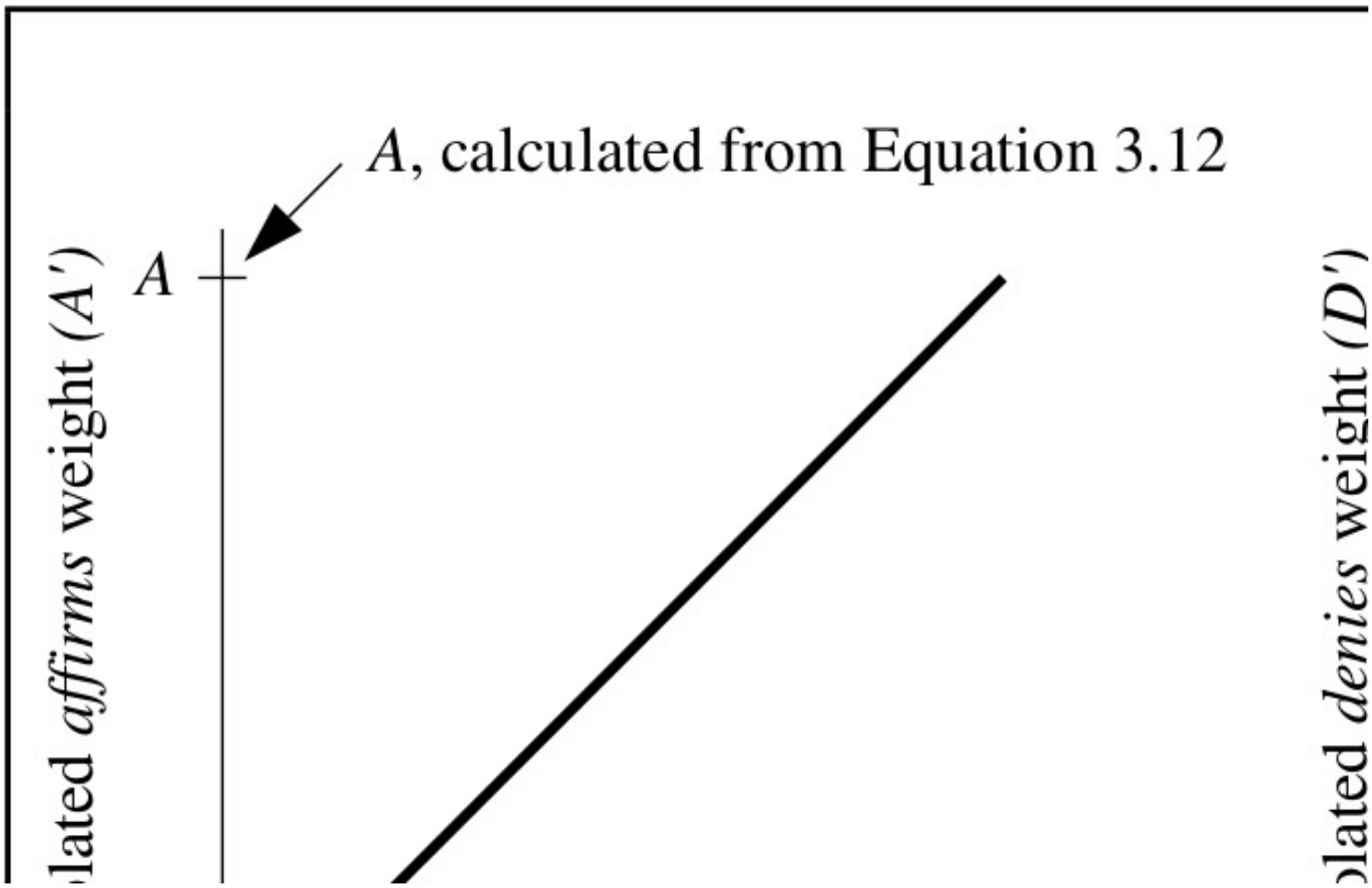
Xử lý bằng chứng không chắc chắn

Về xác suất, chúng ta muốn tính P(H | E), trong đó E là bất định. Chúng ta có thể xử lý vấn đề này bằng cách giả định rằng E được khẳng định bởi một quy tắc khác có bằng chứng là B, trong đó B là chắc chắn (có xác suất 1). Với bằng chứng B, xác suất của E là P(E | B). Bài toán của chúng ta sau đó trở thành một trong những phép tính P(H | B). Một biểu thức cho điều này đã được đưa ra bởi Duda và các cộng sự:

$$P(H|B) = P(H|E) \times P(E|B) + P$$

Cập nhật Bayesian

Xử lý bằng chứng không chắc chắn



Cập nhật Bayesian

Xử lý bằng chứng không chắc chắn

Biểu thức này có thể hữu ích nếu định lý Bayes đang được sử dụng trực tiếp (Phần 3.2.2), nhưng cần có một giải pháp thay thế khi sử dụng tỷ lệ khả năng xảy ra. Một kỹ thuật là sửa đổi các trọng số khẳng định và phủ nhận để phản ánh độ không đảm bảo đo trong E. Một phương tiện để đạt được điều này là nội suy tuyến tính các trọng số vì xác suất của E thay đổi trong khoảng từ 1 đến 0. Hình trang trước minh họa quá trình chia tỷ lệ này, trong đó khẳng định nội suy và từ chối trọng lượng lần lượt được ký hiệu A‘ và D'. Trong khi P(E) lớn hơn 0.5, trọng lượng khẳng định được sử dụng và khi P(E) nhỏ hơn 0.5, trọng lượng từ chối được sử dụng. Trong phạm vi giá trị của P(E), A‘ và D' thay đổi giữa 1 (trọng số trung tính) và A và D, tương ứng. Quá trình nội suy đạt được loại kết quả phù hợp, nhưng không có cơ sở chặt chẽ. Các biểu thức được sử dụng để tính toán các giá trị nội suy là:

$$A' = [2(A - 1) \times \mathrm{P}$$

Cập nhật Bayesian

Kết hợp bằng chứng

Phần lớn tranh cãi liên quan đến việc sử dụng cập nhật Bayes tập trung vào vấn đề làm thế nào để kết hợp một số phần bằng chứng hỗ trợ cho cùng một giả thuyết. Nếu n mẩu bằng chứng được tìm thấy ủng hộ giả thuyết H, thì phát biểu lại chính thức của phương trình cập nhật là đơn giản:

$$O(H|E_1 \& E_2 \& E_3$$

$$A = \frac{P(E_1 \& E_2 \&}{}$$

Cập nhật Bayesian

Kết hợp bằng chứng

Tuy nhiên, mức độ hữu ích của cặp phương trình này là đáng nghi ngờ, vì chúng ta không biết trước phần bằng chứng nào sẽ có sẵn để hỗ trợ giả thuyết H. Chúng ta sẽ phải viết biểu thức cho A bao gồm tất cả các phần bằng chứng có thể có Ei, như cũng như tất cả các kết hợp của cặp Ei & Ej, của bộ ba Ei & Ej & Ek, của bộ tứ Ei & Ej & Ek & Em, v.v. Vì đây rõ ràng là một yêu cầu không thực tế, đặc biệt là khi số lượng các bằng chứng có thể có (hoặc các triệu chứng trong một vấn đề chẩn đoán) lớn, nên một cách đơn giản hóa thường được tìm kiếm. Vấn đề trở nên dễ quản lý hơn nhiều nếu giả định rằng tất cả các phần bằng chứng là độc lập về mặt thống kê. Giả định này là một trong những khía cạnh gây tranh cãi nhất của việc sử dụng cập nhật Bayes trong các hệ thống dựa trên tri thức, vì giả định này hiếm khi chính xác. Tính độc lập thống kê của hai phần bằng chứng (E1 và E2) có nghĩa là xác suất quan sát E1 cho rằng E2 đã được quan sát là trùng với xác suất quan sát E1 không có thông tin về E2. Nói rõ hơn điều này, tính độc lập thống kê của E1 và E2 được định nghĩa là:

Cập nhật Bayesian

Kết hợp bằng chứng

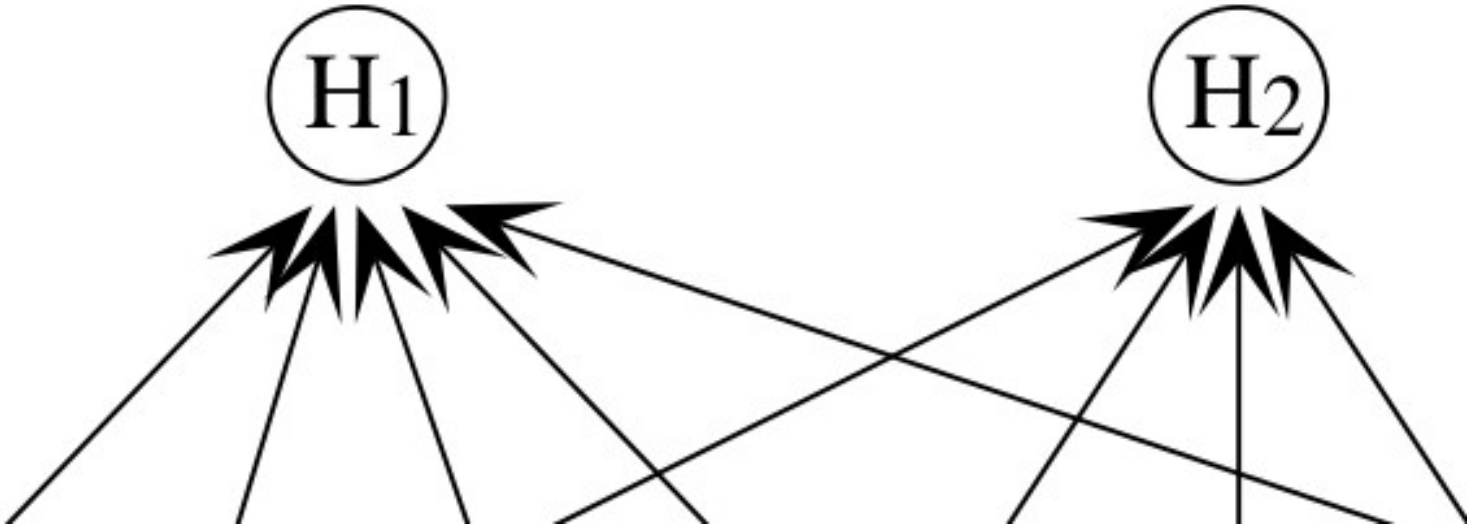
$$P(E_1|E_2) =$$

Nếu giả định về tính độc lập được đưa ra, thì người viết quy tắc chỉ cần lo lắng về việc cung cấp các trọng số của biểu mẫu:

$$A_i = \frac{P(E}{P(E_i}$$

Cập nhật Bayesian

Kết hợp bằng chứng



Cập nhật Bayesian

Kết hợp bằng chứng

Đối với mỗi phần bằng chứng Ei có khả năng cập nhật H. Nếu, trong một lần chạy hệ thống nhất định, n phần bằng chứng được tìm thấy ủng hộ hoặc phản đối H, thì phương trình cập nhật đơn giản là:

$$O(H|E_1\&E_2\&E_3....E_n) = A_1 \times A_2 \times A$$

Cập nhật Bayesian

Kết hợp bằng chứng

Các vấn đề phát sinh từ sự phụ thuộc lẫn nhau của các mảnh bằng chứng có thể tránh được nếu cơ sở quy tắc được cấu trúc đúng. Các mảnh bằng chứng ở đâu được biết là phụ thuộc vào nhau, chúng không nên được kết hợp trong một quy tắc duy nhất. Thay vào đó, các xác nhận - và các quy tắc tạo ra chúng - nên được sắp xếp theo thứ bậc từ dữ liệu đầu vào cấp thấp đến kết luận cấp cao, với nhiều mức độ giả thuyết giữa. Điều này không giới hạn số lượng bằng chứng được xem xét để đưa ra kết luận, nhưng kiểm soát sự tương tác giữa các phần bằng chứng.

Mạng suy luận là một phương tiện thuận tiện để biểu diễn các cấp độ xác nhận từ dữ liệu đầu vào, thông qua các suy luận trung gian để đưa ra kết luận cuối cùng. Hình 3.3 và 3.4 cho thấy hai mạng suy luận có thể có. Mỗi nút đại diện cho một giả thuyết hoặc một phần bằng chứng và có một xác suất liên quan (không được hiển thị). Trong Hình 3.3, người viết quy tắc đã cố gắng rút ra tất cả các bằng chứng liên quan đến kết luận cụ thể với nhau trong một quy tắc duy nhất cho mỗi kết luận. Điều này tạo ra một mạng nông, không có cấp độ trung gian giữa dữ liệu đầu vào và kết luận. Một hệ thống như vậy sẽ chỉ đáng tin cậy nếu có rất ít hoặc không có sự phụ thuộc giữa dữ liệu đầu vào.

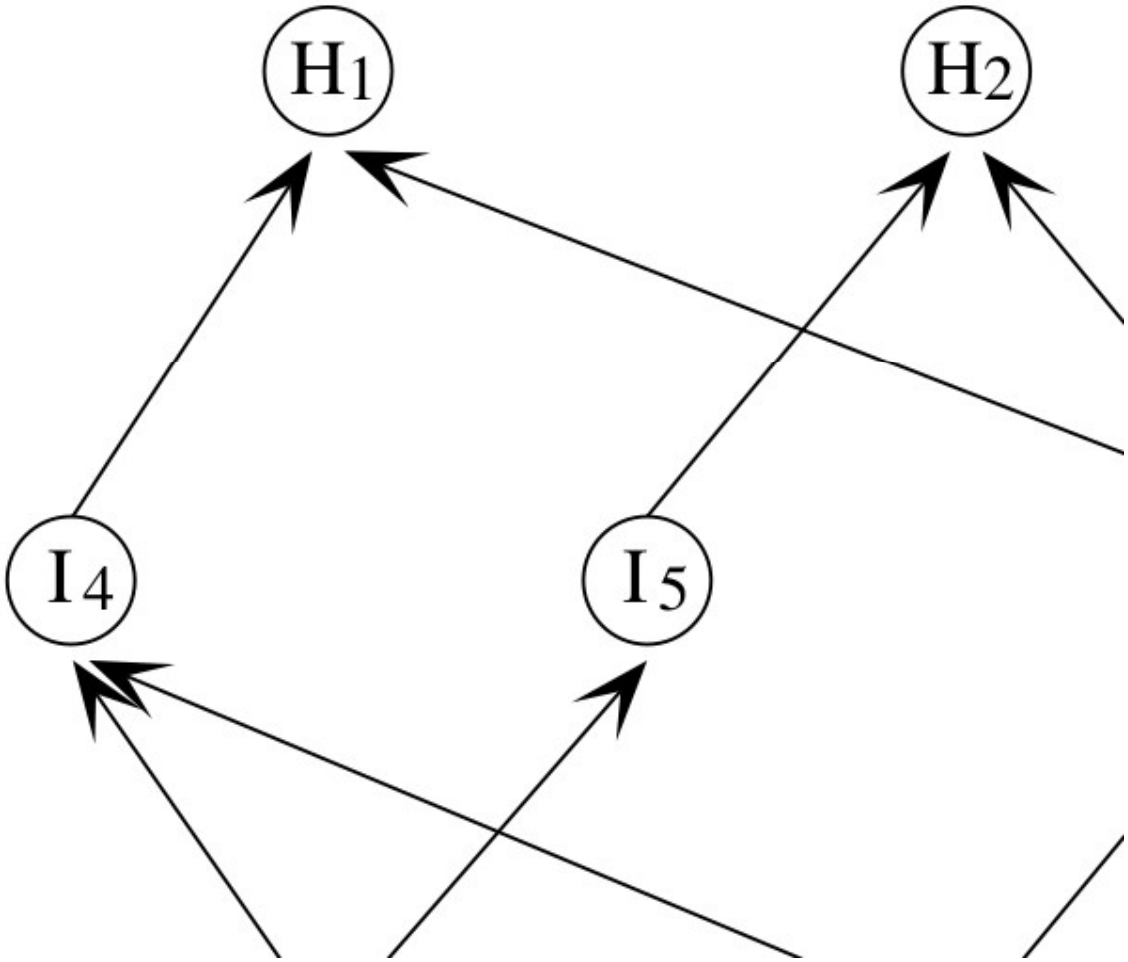
Cập nhật Bayesian

Kết hợp bằng chứng

Ngược lại, mạng suy luận trong Hình 3.4 bao gồm một số bước trung gian. Các xác suất tại mỗi nút được sửa đổi khi quá trình suy luận tiếp tục, cho đến khi chúng đạt đến giá trị cuối cùng. Lưu ý rằng các quy tắc trong ví dụ điều khiển lò hơi sử dụng một số nút trung gian, giúp làm cho các quy tắc dễ hiểu hơn và tránh trùng lặp các thử nghiệm đối với các phần bằng chứng cụ thể.

Cập nhật Bayesian

Kết hợp bằng chứng



Cập nhật Bayesian

Kết hợp các quy tắc Bayes với các quy tắc dẫn xuất

Trong một hệ thống dựa trên quy tắc, thực tế có thể cần kết hợp các quy tắc không chắc chắn với các quy tắc dẫn xuất.

Điều này tránh được vấn đề cung cấp xác suất trước cho giả thuyết hoặc trọng số cho bằng chứng.

Nếu một quy tắc xuất chứa nhiều phần bằng chứng độc lập với nhau, thì xác suất kết hợp của chúng có thể được rút ra từ lý thuyết xác suất tiêu chuẩn. Ví dụ: hãy xem xét một quy tắc trong đó hai phần bằng chứng độc lập dính liền với nhau (tức là chúng được nối với nhau bằng VÀ):

NẾU bằng chứng E1 VÀ bằng chứng E2 THÌ giả thuyết H3.

Xác suất của giả thuyết H3 được cho bởi:

P(H3) = P(E1) x P(E2)

Các quy tắc dẫn xuất chứa bằng chứng độc lập tách rời (tức là được kết hợp bởi OR) có thể được xử lý theo cách tương tự. Vì vậy, đã đưa ra quy tắc:

NẾU bằng chứng E1 HOẶC bằng chứng E2 THÌ giả thuyết H3

Cập nhật Bayesian

Kết hợp các quy tắc Bayes với các quy tắc dẫn xuất

Xác suất của giả thuyết H3 được cho bởi:

P(H3) = P(E1) + P(E2) – (P(E1) x P(E2))

Ví dụ xem trong tài liệu

Cập nhật Bayesian

Ưu điểm và nhược điểm của việc cập nhật Bayesian

Cập nhật Bayes là một phương tiện xử lý sự không chắc chắn bằng cách cập nhật xác suất của một khẳng định khi bằng chứng cho hoặc chống lại khẳng định được cung cấp.

Cập nhật Bayesian

Những ưu điểm chính của việc cập nhật Bayesian là:

(i) Kỹ thuật này dựa trên một định lý thống kê đã được chứng minh.

(ii) Khả năng xảy ra được biểu thị dưới dạng xác suất (hoặc tỷ lệ cược), có ý nghĩa được xác định rõ ràng và quen thuộc.

(iii) Kỹ thuật này yêu cầu xác suất loại trừ, thường dễ ước tính hơn xác suất loại trừ. Người dùng cung cấp các giá trị cho khả năng xảy ra của bằng chứng (các triệu chứng) đưa ra một giả thuyết (nguyên nhân) thay vì ngược lại.

(iv) Các tỷ lệ khả năng xảy ra và xác suất trước có thể được thay thế bằng các phỏng đoán hợp lý. Điều này có lợi cho lợi thế (i), vì các xác suất được tính toán sau đó không thể được hiểu theo nghĩa đen, mà là một thước đo không chính xác về khả năng xảy ra.

(v) Bằng chứng cho và chống lại một giả thuyết (hoặc sự hiện diện và không có bằng chứng) có thể được kết hợp trong một quy tắc duy nhất bằng cách sử dụng các trọng số khẳng định và phủ nhận.

(vi) Nội suy tuyến tính của các tỷ lệ khả năng xảy ra có thể được sử dụng để tính đến bất kỳ sự không chắc chắn nào trong bằng chứng (tức là sự không chắc chắn về việc liệu phần điều kiện của quy tắc có được thỏa mãn hay không), mặc dù đây là một giải pháp đặc biệt.

(vii) Xác suất của một giả thuyết có thể được cập nhật khi có nhiều hơn một bằng chứng.

Cập nhật Bayesian

Những bất lợi chính của cập nhật Bayesian là:

(i) Xác suất trước của một khẳng định phải được biết hoặc đoán tại.

(ii) Các xác suất có điều kiện phải được đo lường hoặc ước tính hoặc, nếu không đạt được những xác suất đó, các phỏng đoán phải được thực hiện ở các tỷ lệ khả năng xảy ra phù hợp. Mặc dù các xác suất có điều kiện thường dễ đánh giá hơn xác suất trước đó, tuy nhiên chúng vẫn là một nguồn sai số đáng kể. Các ước tính về khả năng xảy ra thường bị che khuất bởi một cái nhìn chủ quan về tầm quan trọng hoặc tiện ích của một phần thông tin.

(iii) Giá trị xác suất duy nhất cho sự thật của một khẳng định không cho chúng ta biết gì về độ chính xác của nó.

(iv) Bởi vì bằng chứng cho và chống lại một khẳng định được gộp lại với nhau, không có hồ sơ nào được lưu giữ về số lượng của mỗi khẳng định.

(v) Việc bổ sung một quy tắc mới để khẳng định một giả thuyết mới thường đòi hỏi

thay đổi các xác suất trước đó và trọng số của một số quy tắc khác. Điều này trái ngược với một trong những ưu điểm chính của hệ thống dựa trên tri thức.

(vi) Giả định rằng các mảnh bằng chứng là độc lập thường là không có cơ sở. Các lựa chọn thay thế duy nhất là tính toán trọng số khẳng định và phủ nhận cho tất cả các kết hợp có thể có của bằng chứng phụ thuộc hoặc để cấu trúc lại cơ sở quy tắc để giảm thiểu những tương tác này.

(vii) Kỹ thuật nội suy tuyến tính để xử lý bằng chứng không chắc chắn không được chứng minh về mặt toán học.

(viii) Các biểu diễn dựa trên tỷ lệ cược, theo yêu cầu để sử dụng tỷ lệ khả năng xảy ra, không thể xử lý sự thật tuyệt đối, tức là tỷ lệ cược = vô cùng.

Lý thuyết độ chắc chắn

Giới thiệu:

Lý thuyết độ chắc chắn là sự điều chỉnh của cập nhật Bayes được kết hợp vào lớp vỏ hệ thống chuyên gia EMYCIN. EMYCIN dựa trên MYCIN, một hệ thống chuyên gia hỗ trợ chẩn đoán các bệnh truyền nhiễm. Tên EMYCIN có nguồn gốc từ "MYCIN thiết yếu", phản ánh thực tế là nó không cụ thể cho chẩn đoán y tế và việc xử lý tình trạng không chắc chắn được đơn giản hóa. Lý thuyết độ chắc chắn đại diện cho một nỗ lực để khắc phục một số thiếu sót của cập nhật Bayes, mặc dù tính chặt chẽ toán học của cập nhật Bayes đã bị mất. Vì sự nghiêm ngặt này hiếm khi được chứng minh bằng chất lượng của dữ liệu, nên đây không thực sự là một vấn đề.

Lý thuyết độ chắc chắn

Đưa ra các giả thuyết không chắc chắn:

Thay vì sử dụng xác suất, mỗi khẳng định trong EMYCIN có một giá trị chắc chắn đi kèm với nó. Giá trị độ chắc chắn có thể nằm trong khoảng từ 1 đến –1.

Đối với giả thuyết H đã cho, giá trị chắc chắn C (H) của nó được cho bởi:

C(H) = 1.0 nếu H được biết là đúng;

C(H) = 0.0 nếu H chưa biết;

C(H) = –1.0 nếu H được biết là sai.

Có sự giống nhau giữa giá trị chắc chắn và xác suất, như vậy:

C(H) = 1.0 tương ứng với P(H) = 1.0;

C(H) = 0.0 tương ứng với P(H) ở giá trị tiên nghiệm của nó;

C(H) = –1.0 tương ứng với P(H) = 0.0.

Lý thuyết độ chắc chắn

Đưa ra các giả thuyết không chắc chắn:

Mỗi quy tắc cũng có một độ chắc chắn đi kèm với nó, được gọi là hệ số chắc chắn CF. Yếu tố độ chắc chắn đóng một vai trò tương tự như khẳng định và phủ nhận trọng số trong hệ thống Bayes:

NẾU <sự tin cậy> THÌ <giả thuyết> VỚI hệ số chắc chắn CF

Một phần tính đơn giản của lý thuyết chắc chắn bắt nguồn từ thực tế là các thước đo độ chắc chắn giống hệt nhau được gắn với các quy tắc và giả thuyết. Hệ số chắc chắn của một quy tắc được sửa đổi để phản ánh mức độ chắc chắn của bằng chứng, sao cho hệ số chắc chắn được sửa đổi CF’ được đưa ra bởi:

CF’ = CF x C(E)

Nếu bằng chứng được biết là có mặt, tức là C(E) = 1, thì cho ra CF’ = CF.

Lý thuyết độ chắc chắn

Đưa ra các giả thuyết không chắc chắn:

Kỹ thuật cập nhật độ chắc chắn của giả thuyết H, dựa trên bằng chứng E, liên quan đến việc áp dụng hàm tổng hợp sau:

Nếu C(H) >= 0 và CF' >= 0:

$$C(H \mid E) = C(H) + [CF' \times (1 - C(H))]$$

Nếu C(H) <= 0 và CF' <= 0:

$$C(H \mid E) = C(H) + [CF' \times (1 + C(H))]$$

Nếu C(H) và CF' ngược dấu:

$$C(H \mid E) = (C(H) + CF') / (1 - \min|C(H)|, |CF'|)$$

ở đây:

C(H | E) là độ chắc chắn của H được cập nhật theo bằng chứng E;

C(H) là độ chắc chắn ban đầu của H, tức là, 0 trừ khi nó đã được cập nhật bởi ứng dụng trước đó của một quy tắc;

|x| = độ lớn của x, bỏ qua dấu của nó.

Lý thuyết độ chắc chắn

Đưa ra các giả thuyết không chắc chắn:

Từ các phương trình trên có thể thấy rằng thủ tục cập nhật bao gồm việc thêm một giá trị dương hoặc âm vào độ chắc chắn hiện tại của một giả thuyết. Điều này trái ngược với cập nhật Bayesian, trong đó tỷ lệ xác suất của một giả thuyết được nhân với tỷ lệ khả năng xảy ra thích hợp. Hàm tổng hợp được biểu diễn bởi các Công thức 3.28 đến 3.30 được vẽ trong Hình 3.5, và có thể thấy nó có hình dạng khá giống với phương trình cập nhật Bayes (vẽ trong Hình 3.1).

Lý thuyết độ chắc chắn

Đưa ra các giả thuyết không chắc chắn:

Trong phiên bản tiêu chuẩn của lý thuyết độ chắc chắn, một quy tắc chỉ có thể được áp dụng nếu độ chắc chắn của bằng chứng C(E) lớn hơn 0, tức là nếu bằng chứng có nhiều khả năng xuất hiện hơn không. EMYCIN hạn chế kích hoạt quy tắc hơn nữa bằng cách yêu cầu C(E) > 0.2 để quy tắc được coi là áp dụng. Sự biện minh cho phương pháp tìm kiếm này là nó tiết kiệm sức mạnh tính toán và làm cho các giải thích rõ ràng hơn, vì các quy tắc hiệu quả biên bị loại bỏ. Trên thực tế, có thể cho phép các quy tắc kích hoạt bất kể giá trị của C(E). Khi đó, việc không có bằng chứng hỗ trợ, được chỉ ra bởi C(E) < 0, sẽ được tính đến vì CF' sẽ có dấu hiệu ngược lại với CF.

Lý thuyết độ chắc chắn

Đưa ra các giả thuyết không chắc chắn:

Mặc dù không có lời giải thích lý thuyết nào cho hàm để cập nhật các giá trị chắc chắn, nhưng nó có một số thuộc tính mong muốn:

chức năng là liên tục và không có điểm kỳ dị hoặc bước;

độ chắc chắn cập nhật C(H | E) luôn nằm trong giới hạn –1 và +1;

nếu C(H) hoặc CF' là +1 (tức là chắc chắn đúng) thì C(H | E) cũng là +1;

nếu C(H) hoặc CF' là –1 (tức là chắc chắn sai) thì C(H | E) cũng là –1;

khi các kết luận trái ngược nhau được kết hợp với nhau, chúng có xu hướng triệt tiêu lẫn nhau, tức là nếu C(H) = - CF' thì C(H | E) = 0;

một số phần bằng chứng độc lập có thể được kết hợp bằng cách áp dụng lặp lại chức năng, và kết quả không phụ thuộc vào thứ tự áp dụng các phần bằng chứng;

nếu C(H) = 0, tức là độ chắc chắn của H ở giá trị tiên nghiệm của nó, thì C(H | E) = CF';

nếu bằng chứng chắc chắn (tức là C(E) = 1) thì CF' = CF.

mặc dù không phải là một phần của việc thực hiện tiêu chuẩn, nhưng có thể tính đến việc không có bằng chứng bằng cách cho phép các quy tắc kích hoạt khi C (E) <0.

Lý thuyết độ chắc chắn

Sự kết hợp logic của các bằng chứng:

Trong hệ thống cập nhật Bayes, mỗi phần bằng chứng đóng góp vào một giả thuyết được giả định là độc lập và được đưa ra khẳng định và phủ nhận trọng số của chính nó. Trong các hệ thống dựa trên lý thuyết chắc chắn, yếu tố chắc chắn được liên kết với quy tắc nói chung, thay vì với các phần bằng chứng riêng lẻ. Vì lý do này, lý thuyết độ chắc chắn cung cấp một thuật toán đơn giản để xác định giá trị của hệ số chắc chắn sẽ được áp dụng khi có nhiều hơn một mục bằng chứng trong một quy tắc. Mối quan hệ giữa các phần bằng chứng được thể hiện rõ ràng bằng cách sử dụng AND và OR. Nếu các phần bằng chứng riêng biệt nhằm đóng góp vào một giả thuyết duy nhất độc lập với nhau, thì chúng phải được đặt trong các quy tắc riêng biệt. Thuật toán để kết hợp các mục bằng chứng trong một quy tắc duy nhất được vay mượn từ lý thuyết khả năng của Zadeh (Phần 3.4). Thuật toán bao gồm các trường hợp bằng chứng dính liền (tức là kết hợp bởi AND), rời rạc (tức là kết hợp bởi OR) và phủ định (sử dụng NOT).

Lý thuyết độ chắc chắn

Sự kết hợp logic của các bằng chứng:

Kết hợp

Hãy xem xét một quy tắc của biểu mẫu:

NẾU <bằng chứng E1> VÀ <bằng chứng E2> THÌ <giả thuyết> VỚI yếu tố chắc chắn CF

Độ chắc chắn của bằng chứng tổng hợp được đưa ra bởi C(E1 VÀ E2), trong đó:

C (E1 VÀ E2) = min [C (E1), C (E2)]

Lý thuyết độ chắc chắn

Sự kết hợp logic của các bằng chứng:

Phân ly

Hãy xem xét một quy tắc của biểu mẫu:

NẾU <bằng chứng E1> HOẶC <bằng chứng E2> THÌ <giả thuyết> VỚI yếu tố chắc chắn CF

Độ chắc chắn của bằng chứng tổng hợp được đưa ra bởi C(E1 HOẶC E2), trong đó:

C (E1 HOẶC E2) = max [C (E1), C (E2)]

Lý thuyết độ chắc chắn

Sự kết hợp logic của các bằng chứng:

Phủ định

Hãy xem xét một quy tắc của biểu mẫu:

NẾU KHÔNG <bằng chứng E> THÌ <giả thuyết> VỚI yếu tố chắc chắn CF

Độ chắc chắn của bằng chứng phủ định được đưa ra bởi C(~E), trong đó:

C(~E) = - C(E)

Lý thuyết độ chắc chắn

Sự liên hệ các yếu tố chắc chắn với xác suất:

Lưu ý rằng có sự tương đồng giữa các yếu tố chắc chắn gắn liền với các giả thuyết và xác suất của các giả thuyết đó, như vậy:

C(H) = 1.0 tương ứng với P(H) = 1.0;

C(H) = 0.0 tương ứng với P(H) ở giá trị tiên nghiệm của nó;

C(H) = –1.0 ứng với P(H) = 0.0.

Ngoài ra, tồn tại một mối quan hệ chính thức giữa yếu tố chắc chắn kết hợp với một quy tắc và xác suất có điều kiện P(H | E) của giả thuyết H đưa ra một số bằng chứng E. nhưng thay vào đó chỉ được ước lượng hoặc chọn để đưa ra loại kết quả phù hợp. Các mối quan hệ chính thức như sau.

Lý thuyết độ chắc chắn

Sự liên hệ các yếu tố chắc chắn với xác suất:

Nếu bằng chứng E ủng hộ giả thuyết H, tức là P(H | E) lớn hơn P(H), thì:

$$CF = \frac{P(H|E) - P(H)}{1 - P(H)}$$

Nếu bằng chứng E phản đối giả thuyết H, tức là P(H | E) nhỏ hơn P (H), thì:

$$CF = \frac{P(H|E) - P(H)}{P(H)}$$

Lý thuyết khả năng: tập mờ và logic mờ

Cập nhật Bayes và lý thuyết chắc chắn là các kỹ thuật để xử lý độ không đảm bảo phát sinh, hoặc được giả định là phát sinh, từ các biến thể thống kê hoặc ngẫu nhiên. Lý thuyết khả năng giải quyết một nguồn khác của sự không chắc chắn, đó là sự mơ hồ trong việc sử dụng ngôn ngữ. Lý thuyết khả năng, hay logic mờ, được Zadeh phát triển và xây dựng dựa trên lý thuyết của ông về các tập mờ. Zadeh khẳng định rằng mặc dù lý thuyết xác suất có thể thích hợp để đo lường khả năng xảy ra của một giả thuyết, nhưng nó không nói gì về ý nghĩa của giả thuyết.

Lý thuyết khả năng: tập mờ và logic mờ

Bộ sắc nét và bộ mờ

Tập hợp mờ là một phương tiện để làm mịn các ranh giới. Lý thuyết về tập mờ thể hiện về mặt định lượng không chính xác bằng cách giới thiệu các hàm liên thuộc đặc trưng có thể giả định các giá trị từ 0 đến 1 tương ứng với các cấp độ thành viên từ "không phải là thành viên" đến "thành viên đầy đủ". Nếu F là một tập mờ, thì hàm liên thuộc μF(x) đo mức độ mà một giá trị tuyệt đối x thuộc về F. Mức độ liên thuộc này đôi khi được gọi là khả năng mà x được mô tả bởi F. Quá trình suy ra các các giá trị khả dĩ đối với một giá trị cho trước của x được gọi là quá trình mờ.

Lý thuyết khả năng: tập mờ và logic mờ

Sự khác biệt chính giữa tập hợp mờ và tập hợp rõ nét là:

- một phần tử có bậc thuộc (0 - 1) của một tập mờ;
- thành viên của một tập mờ không loại trừ thành viên của tập khác.

Lý thuyết khả năng: tập mờ và logic mờ

Luật mờ:

Nếu một biến được đặt thành giá trị theo quy tắc rõ ràng, giá trị của nó sẽ thay đổi theo các bước khi các quy tắc khác nhau kích hoạt. Cách duy nhất để làm trơn tru các bước đó là có một số lượng lớn các quy tắc. Tuy nhiên, chỉ cần một số lượng nhỏ các quy tắc mờ để tạo ra các thay đổi trơn tru trong các đầu ra khi các giá trị đầu vào thay đổi. Số lượng luật mờ cần thiết phụ thuộc vào số lượng biến, số lượng tập mờ và cách kết hợp các biến trong điều kiện luật mờ.

Lý thuyết khả năng: tập mờ và logic mờ

Khử mờ:

Khử mờ đặc biệt quan trọng khi biến mờ là một hành động điều khiển chẳng hạn như "dòng điện đặt", trong đó một cài đặt cụ thể được yêu cầu. Việc sử dụng logic mờ trong hệ thống điều khiển sẽ được thảo luận thêm trong Chương 14.

Quá trình khử mờ có hai giai đoạn:

Giai đoạn 1: mở rộng các chức năng thành viên,

Giai đoạn 2: tìm kiếm trung tâm.

Lý thuyết khả năng: tập mờ và logic mờ

Khử mờ: (Giai đoạn 1)

Bước đầu tiên trong quá trình giải mờ là điều chỉnh các tập mờ phù hợp với các khả năng đã tính toán. Một phương pháp thường được sử dụng là quy tắc hoạt động sản phẩm của Larsen [12, 13], trong đó các hàm liên thuộc được nhân với các giá trị khả năng tương ứng của chúng. Tác dụng là nén các tập mờ sao cho các đỉnh bằng với các giá trị khả năng đã tính được, như trong Hình 3.10. Một số tác giả [14] áp dụng một cách tiếp cận thay thế trong đó các tập mờ được cắt bớt, như trong Hình 3.11. Đối với hầu hết các hình dạng của tập mờ, sự khác biệt giữa hai cách tiếp cận là nhỏ, nhưng quy tắc hoạt động sản phẩm của Larsen có ưu điểm là đơn giản hóa các tính toán và cho phép quá trình làm mờ sau đó là quá trình giải mờ để trả về giá trị ban đầu, ngoại trừ được mô tả trong Một bất thường về giải mờ dưới đây.

Lý thuyết khả năng: tập mờ và logic mờ

Khử mờ: (Giai đoạn 2)

Phương pháp giải mờ được sử dụng phổ biến nhất là phương pháp trọng tâm, đôi khi được gọi là trọng tâm, khối tâm, hoặc phương pháp tâm diện tích. Giá trị được giải mờ được coi là điểm dọc theo trục biến mờ là tâm hay điểm cân bằng của tất cả các hàm liên thuộc được chia tỷ lệ được lấy cùng nhau cho biến đó (Hình 3.12). Một cách để hình dung điều này là tưởng tượng các chức năng thành viên được cắt ra từ thẻ cứng và được dán lại với nhau ở nơi (và nếu) chúng chồng lên nhau. Giá trị được giải mờ là điểm cân bằng dọc theo trục biến mờ của hình dạng tổng hợp này. Khi hai hàm liên thuộc chồng lên nhau, cả hai vùng chồng chéo đều đóng góp vào khối lượng của hình dạng tổng hợp. Hình 3.12 cho thấy một trường hợp đơn giản, không liên quan đến các tập mờ thấp và cao. Ví dụ mà chúng tôi đang theo dõi liên quan đến áp suất lò hơi phức tạp hơn và được mô tả trong Khử mờ ở các cực dưới đây.

Tổng kết chương

Một số phương án khác nhau tồn tại để gán các giá trị số cho các xác nhận nhằm thể hiện mức độ tin cậy vào chúng và để cập nhật các mức độ tin cậy dựa trên bằng chứng hỗ trợ hoặc phản đối. Khó khăn lớn nhất nằm ở việc thu được các giá trị chính xác của khả năng xảy ra, dù được đo lường dưới dạng xác suất hay bằng một số phương pháp khác. Các yếu tố chắc chắn được liên kết với các quy tắc trong lý thuyết chắc chắn, và các trọng số khẳng định và phủ nhận trong cập nhật Bayes, có thể được rút ra từ các ước lượng xác suất. Tuy nhiên, một cách tiếp cận thực dụng hơn thường được áp dụng, đó là chọn các giá trị tạo ra loại kết quả phù hợp, mặc dù các giá trị không thể được chứng minh về mặt lý thuyết. Vì các kỹ thuật phức tạp hơn (ví dụ, lý thuyết bằng chứng Dempster – Shafer và Inferno) cũng phụ thuộc vào các ước tính xác suất thường không rõ ràng, nên việc sử dụng chúng hiếm khi được chứng minh.

Tổng kết chương

Cập nhật Bayes hoàn toàn dựa trên lý thuyết xác suất, trong khi nhiều kỹ thuật thay thế là đặc biệt. Trên thực tế, cập nhật Bayesian cũng là một kỹ thuật đặc biệt vì:

- nội suy tuyến tính của trọng số khẳng định và phủ nhận thường được sử dụng như một phương tiện thuận tiện để bù đắp cho sự không chắc chắn trong bằng chứng;
- tỷ lệ khả năng xảy ra (hoặc xác suất mà chúng được suy ra) và các xác suất trước thường dựa trên các ước tính hơn là phân tích thống kê;
- các mục bằng chứng riêng biệt hỗ trợ một khẳng định duy nhất được giả định là độc lập về mặt thống kê, mặc dù điều này có thể không đúng trong thực tế.

Tổng kết chương

Mạng nơron (xem Chương 8) đại diện cho một cách tiếp cận thay thế tránh được những khó khăn trong việc thu được các ước tính xác suất đáng tin cậy. Mạng nơron có thể được sử dụng để đào tạo một hệ thống máy tính bằng cách sử dụng nhiều ví dụ để nó có thể đưa ra kết luận có trọng số dựa trên bằng chứng được cung cấp. Tất nhiên, với một tập hợp các ví dụ đủ lớn, cũng có thể tính toán chính xác các xác suất và trọng số trước đó cần thiết để cập nhật Bayesian hoặc một trong các dẫn xuất của nó hoạt động hiệu quả.

Logic mờ cũng liên kết chặt chẽ với mạng nơ-ron, như sẽ được thảo luận trong Chương 9. Logic mờ cung cấp một cách chính xác để xử lý các thuật ngữ mơ hồ như thấp và cao. Kết quả là, một bộ quy tắc nhỏ có thể tạo ra các giá trị đầu ra thay đổi trơn tru khi các giá trị đầu vào thay đổi.

Chương 4

Hệ thống hướng đối tượng

learncpp.com

Udemy.com

Đối tượng và khung

Thiết kế hệ thống thường liên quan đến việc chia nhỏ các vấn đề phức tạp thành các cấu thành đơn giản hơn. Đối tượng và khung là hai cách có liên quan chặt chẽ với nhau để đạt được điều này trong khi vẫn duy trì tính toàn vẹn tổng thể của hệ thống.

Lập trình dựa trên khung thường gắn liền với việc xây dựng và tổ chức các hệ thống dựa trên tri thức.

Lập trình hướng đối tượng (OOP) được sử dụng trong nhiều hệ thống phần mềm bao gồm nhưng không giới hạn ở các hệ thống thông minh.

OOP đã được phát triển như một “cách tốt hơn để lập trình”, trong khi khung được hình thành như một cách biểu diễn và tổ chức thông tin linh hoạt và biểu cảm.

Đối tượng và khung

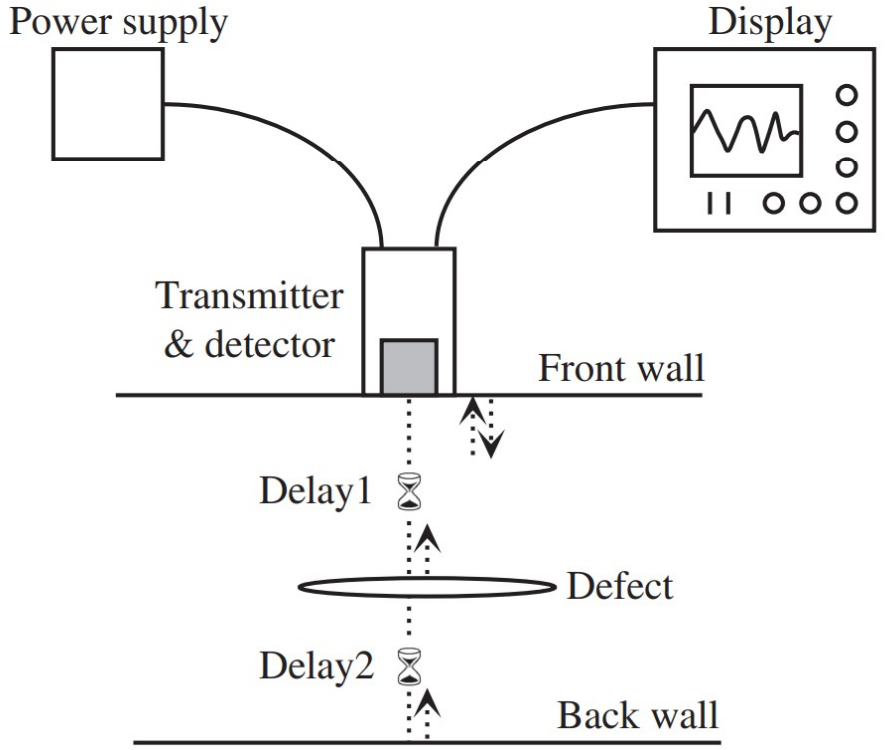
Cả hai kỹ thuật đều hỗ trợ thiết kế phần mềm và làm cho phần mềm kết quả dễ bảo trì hơn, dễ thích nghi hơn và có thể tái sử dụng. Chúng cung cấp một cấu trúc để chia nhỏ đại diện một thế giới thành các thành phần có thể quản lý được chẳng hạn như ngôi nhà, chủ sở hữu và con chó.

Trong một hệ thống dựa trên khung, mỗi thành phần này có thể được biểu diễn bằng một khung, chứa thông tin về chính nó. Các khung có nghĩa là bị động, giống như các mục nhập trong cơ sở dữ liệu, chúng không tự thực hiện bất kỳ tác vụ nào.

Các đối tượng cũng tương tự như vậy, nhưng cũng như việc lưu trữ thông tin về bản thân chúng, chúng cũng có khả năng thực hiện một số nhiệm vụ nhất định. Khi nhận được chỉ dẫn, chúng sẽ thực hiện một hành động theo hướng dẫn, miễn là trong khả năng của chúng. Vì vậy, chúng giống như những người hầu vâng lời.

Đối tượng và khung

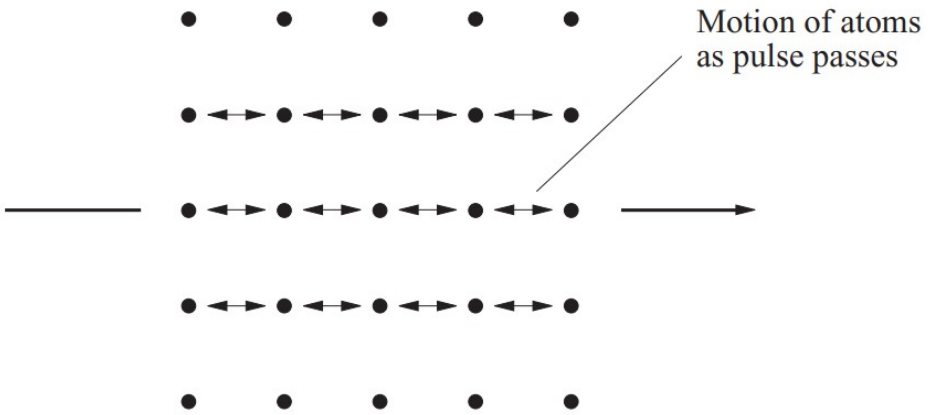
Các đối tượng và khung có nhiều điểm tương đồng. Trong chương này, OOP được mô tả một số chi tiết, sau đó việc xem xét các khung tập trung vào sự khác biệt của chúng với các đối tượng.



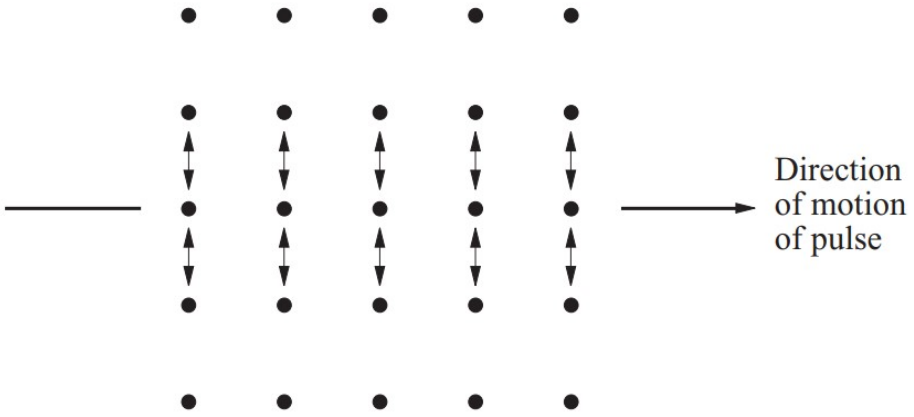
Đối tượng và khung

Ví dụ mô phỏng nhỏ của hình ảnh siêu âm. Sự sắp xếp vật lý được mô phỏng được thể hiện trong Hình. Nó bao gồm một đầu dò siêu âm, chứa một máy dò và một máy phát, nằm trên bề mặt của một bộ phận được thử nghiệm. Đầu dò phát ra một xung siêu âm, tức là âm thanh ở tần số cao hơn phạm vi có thể nghe được của con người. Xung được phản xạ mạnh bởi các thành trước và sau của thành phần và bị phản xạ yếu bởi các khuyết tật bên trong thành phần. Cường độ và thời gian xuất hiện của các xung phản xạ tới máy dò được vẽ trên máy hiện sóng. Có thể có hai loại xung siêu âm - theo phương dọc và xung cắt - truyền qua chất rắn khác nhau. Các nguyên tử của vật liệu di chuyển tới lui, song song với hướng của xung dọc, nhưng chúng chuyển động vuông góc với hướng của xung cắt. Hai loại xung truyền với tốc độ khác nhau trong một vật liệu nhất định và hoạt động khác nhau tại các vật cản hoặc các cạnh. Xung phát ra luôn luôn theo chiều dọc, nhưng phản xạ có thể tạo ra cả xung dọc và xung cắt.

Đối tượng và khung



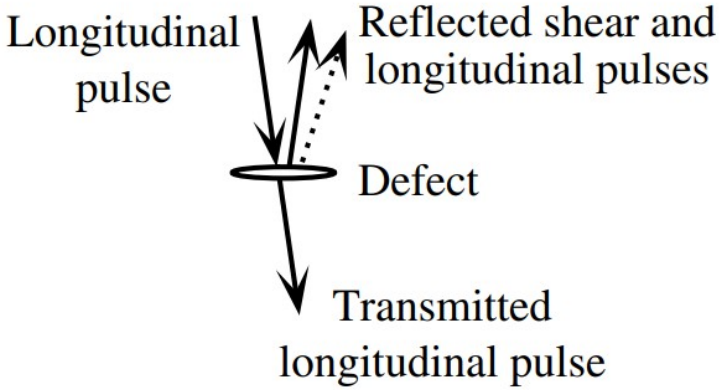
(a) Longitudinal (or compressive) pulse

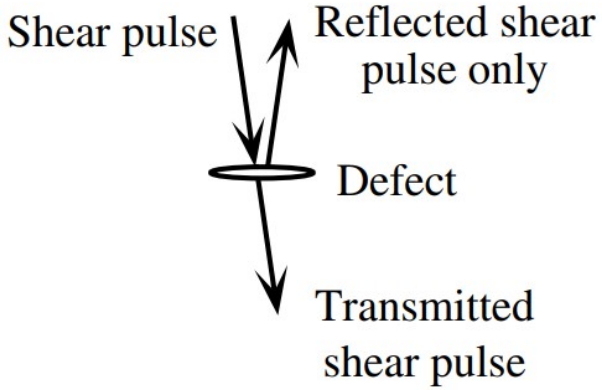


(b) Shear (or transverse) pulse

Đối tượng và khung

Hệ thống này, giống như nhiều hệ thống khác, tự tạo mô hình với các đối tượng. Mỗi thành phần của hệ thống có thể được coi như một đối tượng độc lập, chứa dữ liệu và mã xác định hành vi cụ thể của riêng nó. Như vậy, sẽ có các đối tượng đại diện cho từng xung, máy phát, máy dò, thành phần, thành trước, thành sau, các khuyết tật, và máy hiện sóng. Các đối tượng này đứng trong các mối quan hệ được xác định rõ ràng với nhau, ví dụ: một đối tượng máy phát có thể tạo ra một đối tượng xung.





Giới thiệu về OOP

Ngoài những lợi thế thiết thực về khả năng bảo trì, khả năng thích ứng và khả năng tái sử dụng, OOP cũng là một cách tự nhiên để biểu diễn nhiều vấn đề trong thế giới thực bằng máy tính. Các chương trình được các kỹ sư và nhà khoa học quan tâm thường thực hiện tính toán hoặc đưa ra quyết định về các thực thể vật chất. Các chương trình như vậy phải chứa một mô hình của các thực thể đó và OOP là một cách thuận tiện để thực hiện mô hình này. Mọi thực thể có thể được đại diện bởi một "đối tượng" độc lập, trong đó một đối tượng chứa dữ liệu và mã có thể được thực hiện trên những dữ liệu đó. Do đó, một hệ thống để mô phỏng hình ảnh siêu âm có thể bao gồm một xung siêu âm được phản xạ giữa các đặc điểm trong thành phần thép. Xung, các tính năng và bản thân thành phần đều có thể được biểu diễn bằng các đối tượng.

Giới thiệu về OOP

Lý thuyết hệ thống dẫn trực tiếp đến cách tiếp cận như vậy, vì một hệ thống có thể được định nghĩa là "một tập hợp các thành phần, ngăn cách với môi trường của chúng bằng một ranh giới, nhưng có liên quan với nhau và được tổ chức để đạt được một mục đích rõ ràng." Do đó, OOP có thể được coi như là một máy tính triển khai lý thuyết hệ thống, trong đó mỗi thành phần tự nó là một hệ thống con có thể được đại diện bởi một đối tượng.

Mô hình hóa với các đối tượng không bị giới hạn ở những thứ vật lý, nhưng mở rộng để bao gồm các thực thể trong môi trường lập trình như cửa sổ, biểu tượng, menu, đồng hồ và bất kỳ thứ gì khác mà bạn có thể nghĩ đến. Trên thực tế, sự phát triển của giao diện người dùng đồ họa là một trong những động lực chính cho OOP. Điều này chứng tỏ rằng OOP là một kỹ thuật mạnh mẽ theo đúng nghĩa của nó và hoàn toàn không bị hạn chế đối với các hệ thống dựa trên tri thức.

Giới thiệu về OOP

Tuy nhiên, OOP thực sự có một vai trò quan trọng trong nhiều hệ thống dựa trên tri thức, vì nó cung cấp một cách thể hiện những thứ đang được lý luận, các thuộc tính và hành vi của chúng cũng như các mối quan hệ giữa chúng.

Có nhiều ngôn ngữ và môi trường lập trình cung cấp hướng đối tượng, một số trong số đó được cung cấp dưới dạng phần mở rộng cho các ngôn ngữ hiện có. Bốn ngôn ngữ OOP được sử dụng rộng rãi là C ++, Smalltalk, Java và CLOS (Hệ thống đối tượng Lisp chung). Smalltalk là một môi trường lập trình chứ không chỉ là một ngôn ngữ. Nó bao gồm một số menu, cửa sổ, trình duyệt (xem Phần 4.5.4) và các lớp cài sẵn (xem Phần 4.4.1). C ++ và CLOS lần lượt là các phiên bản mở rộng của ngôn ngữ C và Lisp, nhưng nhiều triển khai hiện đại cũng bao gồm các công cụ hỗ trợ tương tự như của Smalltalk.

Giới thiệu về OOP

Theo Pascoe [1], ngôn ngữ OOP cung cấp ít nhất các tiện ích sau:
- Biểu diễn dữ liệu
- Kế thừa
- Đóng gói (hoặc ẩn thông tin)
- Liên kết động

Biểu diễn dữ liệu

Lớp:
Ví dụ siêu âm liên quan đến một số loại vật thể khác nhau. Xung cắt là các đối tượng cùng loại với nhau, nhưng khác loại với máy dò. Sự trừu tượng hóa dữ liệu cho phép chúng ta xác định các kiểu này và các chức năng đi kèm với chúng. Các kiểu đối tượng này được gọi là các lớp và chúng tạo thành các khuôn mẫu cho chính các đối tượng. Các đối tượng đại diện cho cùng một ý tưởng hoặc khái niệm được nhóm lại với nhau thành một lớp. Các từ loại và lớp nói chung là tương đương và được sử dụng thay thế cho nhau ở đây.

Biểu diễn dữ liệu

Lớp:

Hầu hết các ngôn ngữ máy tính bao gồm một số kiểu dữ liệu tích hợp đơn giản, chẳng hạn như số nguyên hoặc thực. Một ngôn ngữ hướng đối tượng cho phép chúng ta xác định các kiểu dữ liệu bổ sung (tức là các lớp) được xử lý giống nhau hoặc gần giống nhau đối với các kiểu tích hợp sẵn. Các lớp do người dùng định nghĩa đôi khi được gọi là kiểu dữ liệu trừu tượng. Đây có thể là các lớp có vai trò cụ thể trong một miền cụ thể, chẳng hạn như Shear_pulse, Circle hoặc Polymer, hoặc chúng có thể mô tả phiên bản chuyên biệt của kiểu dữ liệu chuẩn, chẳng hạn như Big_integer.

Định nghĩa của một lớp bao gồm tên lớp, các thuộc tính của nó (Phần 4.4.3), các hoạt động của nó (Phần 4.4.4) và các mối quan hệ của nó với các lớp khác (Phần 4.5 và 4.7). C ++ và một số ngôn ngữ OOP khác yêu cầu các kiểu thuộc tính và khả năng hiển thị từ các lớp khác cũng phải được chỉ định.

Biểu diễn dữ liệu

Phiên bản/thể hiện:

Nó đã được nêu trong Phần 4.4.1 rằng các lớp tạo thành khuôn mẫu cho “chính các đối tượng”, nghĩa là các thể hiện của đối tượng. Khi một lớp đã được định nghĩa, các thể hiện của lớp có thể được tạo ra có các thuộc tính được định nghĩa cho lớp. Ví dụ: xung # 259 có thể là một xung siêu âm có vị trí tại một thời điểm nhất định là (x = 112mm, y = 294mm, z = 3,5mm). Chúng tôi sẽ biểu diễn xung # 259 như một thể hiện của lớp Longitudinal_pulse. Như một ví dụ rõ ràng hơn, chiếc xe của tôi là một ví dụ của loại xe hơi. Lớp xác định các đặc điểm và hành vi của các thể hiện của nó. Do đó, một lớp có thể được coi là bản thiết kế mà từ đó các thể hiện được xây dựng.

Thuật ngữ đối tượng và thể hiện thường được sử dụng thay thế cho nhau, mặc dù sẽ hữu ích khi sử dụng đối tượng sau để nhấn mạnh sự phân biệt với một lớp.

Biểu diễn dữ liệu

Phiên bản/thể hiện:
Đây chỉ là một phần mở rộng của các khái niệm về kiểu dữ liệu và biến tồn tại trong các ngôn ngữ lập trình khác. Ví dụ, hầu hết các ngôn ngữ đều bao gồm định nghĩa kiểu số nguyên có sẵn. Tuy nhiên, chúng ta chỉ có thể dựa trên các thuộc tính của một số nguyên bằng cách tạo một thể hiện, tức là một biến số nguyên. Trong C, điều này sẽ giống như sau:

```
int x;      /* create an instance, x, of type int */
x = 3;      /* manipulate x */
```

Biểu diễn dữ liệu

Phiên bản/thể hiện:
Tương tự như vậy, khi chúng ta đã định nghĩa một lớp trong ngôn ngữ hướng đối tượng, các thuộc tính của nó chỉ có thể được sử dụng bằng cách tạo một hoặc nhiều trường hợp. Xem xét lớp Longitudinal_pulse. Một thể hiện của lớp này được tạo ra bởi đối tượng truyền (bản thân nó là một thể hiện của Bộ truyền lớp). Ví dụ này đại diện cho một xung cụ thể, có vị trí, biên độ, pha, tốc độ và hướng. Khi xung này tương tác với một khuyết tật (trường hợp khác), một thể hiện mới của Longitudinal_pulse phải được tạo, vì sẽ có cả xung được truyền và xung phản xạ (xem Hình 4.3). Xung mới sẽ có cùng thuộc tính với xung cũ vì cả hai đều bắt nguồn từ cùng một lớp, nhưng một số thuộc tính này sẽ có các giá trị khác nhau được liên kết với chúng (ví dụ: biên độ và hướng sẽ khác nhau).

Biểu diễn dữ liệu

Thuộc tính (hoặc thành phần/thành viên dữ liệu):
Một định nghĩa lớp bao gồm các thuộc tính và hoạt động của nó. Các thuộc tính là các đại lượng cụ thể mô tả các thể hiện của lớp đó. Chúng đôi khi được mô tả như là các vị trí mà các giá trị được chèn vào. Do đó, lớp Shear_pulse có thể có các thuộc tính như biên độ, vị trí, tốc độ và hướng. Chỉ tên và, tùy thuộc vào ngôn ngữ, các loại thuộc tính cần được khai báo trong lớp, mặc dù các giá trị cũng có thể được cung cấp tùy chọn. Lớp hoạt động như một khuôn mẫu, để khi một xung cắt mới được tạo ra, nó sẽ chứa tên của các thuộc tính này. Các giá trị thuộc tính có thể được cung cấp hoặc tính toán cho mỗi trường hợp hoặc một giá trị được cung cấp trong lớp có thể được sử dụng làm giá trị mặc định. Các thuộc tính có thể thuộc bất kỳ kiểu nào, bao gồm các kiểu dữ liệu trừu tượng, tức là các lớp. Một số ngôn ngữ, chẳng hạn như C ++, yêu cầu định nghĩa lớp xác định trước các kiểu thuộc tính. Biên độ và tốc độ có thể thuộc loại float, trong khi vị trí và hướng sẽ là loại vector.

Biểu diễn dữ liệu

Thuộc tính (hoặc thành phần/thành viên dữ liệu):
Nếu một giá trị cho một thuộc tính được cung cấp trong định nghĩa lớp, thì mọi phiên bản mới sẽ mang những giá trị đó theo mặc định trừ khi được ghi đè cụ thể. Hầu hết các ngôn ngữ OOP phân biệt giữa thuộc tính lớp (hoặc biến lớp) và thuộc tính cá thể (hoặc biến cá thể). Giá trị của một thuộc tính lớp không đổi cho tất cả các trường hợp của lớp đó. Ngược lại, các cá thể chứa các bản sao riêng của mỗi thuộc tính cá thể. Nếu một mặc định đã được chỉ định, mỗi cá thể của lớp đã cho có cùng giá trị ban đầu cho một thuộc tính cá thể. Tuy nhiên, các giá trị của thuộc tính instance sau đó có thể thay đổi từ phiên bản này sang trường hợp khác. Do đó, trong ví dụ trên, tốc độ có thể là một thuộc tính lớp cho Shear_pulse, vì tốc độ sẽ giống nhau đối với tất cả các xung cắt trong một vật liệu nhất định. Mặt khác, biên độ và vị trí là các thuộc tính của từng xung riêng lẻ và được biểu diễn dưới dạng các thuộc tính thể hiện.

Biểu diễn dữ liệu

Thuộc tính (hoặc thành phần/thành viên dữ liệu):
Trong C ++, các thuộc tính được gọi là thành viên dữ liệu. Tất cả các thành viên dữ liệu được giả định là thuộc tính cá thể trừ khi chúng được khai báo là tĩnh. Các thành viên dữ liệu tĩnh được lưu trữ tại cùng một vị trí bộ nhớ cho mọi thể hiện của một lớp, vì vậy chỉ có một bản sao, bất kể có thể có bao nhiêu thể hiện. Do đó, các thành viên dữ liệu tĩnh tương đương với các thuộc tính của lớp.

Biểu diễn dữ liệu

Hoạt động/tính toán (hoặc phương pháp/phương thức của các hàm thành viên):
Mỗi đối tượng có một tập hợp các hoạt động hoặc chức năng mà nó có thể thực hiện. Ví dụ, một đối tượng Shear_pulse có thể chứa hàm di chuyển. Hàm này có thể coi các tham số của nó là số lượng và hướng chuyển động, đồng thời trả về một giá trị mới cho thuộc tính vị trí của nó. Trong một số ngôn ngữ OOP, các phép toán thuộc về các đối tượng được gọi là các phương thức. Trong C ++, chúng được gọi là các hàm thành viên. Các phép toán được định nghĩa cho một lớp và sau đó có thể được sử dụng bởi tất cả các phiên bản của lớp đó. Các phiên bản của các lớp khác cũng có thể truy cập vào các hoạt động này (xem Phần 4.5 và 4.6).

Biểu diễn dữ liệu

Tạo và xóa các phiên bản:
Tạo một cá thể mới yêu cầu máy tính dành một số bộ nhớ để lưu trữ cá thể đó. Với định nghĩa lớp, Myclass, việc tạo các phiên bản mới tương tự như trong Smalltalk và trong C ++. Trong Smalltalk, chúng ta có thể viết:

```
Myinstance := Myclass new.      "Myinstance is a global variable"
```

Các từ trong dấu ngoặc kép là một nhận xét/chú thích và được trình biên dịch bỏ qua. Tương đương trong C ++ sẽ là:

```
Myclass* myinstance;
        // declare a pointer to objects of type Myclass
myinstance = new Myclass;
        // pointer now points to a new instance
```

Biểu diễn dữ liệu

Tạo và xóa các phiên bản:
Ở đây, biểu tượng // chỉ ra một nhận xét. Phiên bản C ++ có thể được viết tắt thành:

```
Myclass* myinstance = new Myclass;
        // new pointer points to a new instance.
```

Một số lượng lớn các phiên bản có thể được tạo ra trong khi chạy một hệ thống hướng đối tượng điển hình. Do đó, một cân nhắc quan trọng là việc giải phóng bộ nhớ không còn cần thiết. Trong ví dụ về mô phỏng siêu âm, các trường hợp xung mới được tạo ra thông qua phản xạ, nhưng chúng phải được loại bỏ khi chúng đến đầu dò. Bộ nhớ bị chiếm bởi các trường hợp không mong muốn này phải được giải phóng lại nếu chúng ta muốn tránh xây dựng một hệ thống với sự thèm muốn vô độ đối với bộ nhớ máy tính. Trong ví dụ Smalltalk ở trên, Myinstance là một biến toàn cục, được biểu thị bằng cách viết hoa chữ cái đầu tiên của tên.

Biểu diễn dữ liệu

Tạo và xóa các phiên bản:
Các biến toàn cục có thể truy cập được từ bất kỳ phần nào của chương trình và được giữ lại trong môi trường lập trình cho đến khi bị xóa rõ ràng. Phổ biến hơn là đính kèm các phiên bản vào các biến tạm thời, được hiển thị ở đây trong Smalltalk:

```
|myinstance|     "myinstance is declared as a temporary variable"
myinstance := Myclass new.
```

Các biến tạm thời chỉ tồn tại trong phương thức mà chúng được khai báo và thời gian tồn tại của chúng là thời gian tồn tại của phương thức kích hoạt. Khi quá trình thực thi phương thức kết thúc, các biến này (và các biến toàn cục đã bị xóa) để lại một vùng bộ nhớ “chưa biết”. Hệ thống Smalltalk tự động lấy lại bộ nhớ này - một quá trình được gọi là thu gom rác. Tùy thuộc vào việc triển khai cụ thể, việc thu gom rác có thể khiến chương trình bị "đóng băng" trong giây lát trong khi hệ thống thực hiện quản lý bộ nhớ của nó. Thu gom rác là một tính năng của Smalltalk, CLOS và các ngôn ngữ OOP khác. Một số triển khai cho phép lập trình viên ảnh hưởng đến thời gian thu gom rác, trong khi trong các triển khai khác, thu gom rác là một quá trình nền không được chú ý.

Biểu diễn dữ liệu

Tạo và xóa các phiên bản:
Trong C ++, trách nhiệm quản lý bộ nhớ thuộc về lập trình viên. Các đối tượng được tạo như mô tả ở trên phải được phá hủy rõ ràng khi chúng không còn cần thiết nữa:

delete myinstance;

Thao tác này giải phóng bộ nhớ tương ứng. C ++ cũng cho phép một phương pháp tạo và xóa đối tượng thay thế dựa trên phạm vi của đối tượng. Một phiên bản mới, myinstance, có thể được tạo trong một khối mã theo cách sau:

Myclass myinstance;

Phiên bản chỉ tồn tại trong khối mã cụ thể đó, đó là phạm vi của nó. Khi luồng thực thi đi vào khối mã, đối tượng sẽ tự động được tạo. Tương tự, nó bị xóa ngay khi luồng thực thi rời khỏi khối.

Biểu diễn dữ liệu

Tạo và xóa các phiên bản:

C ++ cho phép lập trình viên xác định các chức năng đặc biệt sẽ được thực hiện khi một thể hiện mới được tạo hoặc xóa. Chúng được gọi tương ứng là hàm tạo và hàm hủy. Hàm tạo là một hàm thành viên có tên giống với tên của lớp và thường được sử dụng để đặt các giá trị ban đầu của một số thuộc tính. Hàm hủy được định nghĩa theo cách tương tự với hàm tạo và tên của nó là của lớp trước dấu ngã (~). Nó chủ yếu được sử dụng để giải phóng bộ nhớ khi một cá thể bị xóa. Ví dụ, hãy xem xét định nghĩa cho Sonic_pulse trong C ++:

```
// class definition:
class Sonic_pulse
{
   protected:
      float amplitude;
   public:
      Sonic_pulse(float initial_amplitude);              // constructor
      ~Sonic_pulse();                                     // destructor
};

// constructor definition:
Sonic_pulse::Sonic_pulse(float initial_amplitude)
{
   amplitude=initial_amplitude;                  // set up initial value
}

// destructor definition:
Sonic_pulse::~Sonic_pulse()
{
        // perform any tidying up that may be necessary before
        // deleting the object
}
```

Biểu diễn dữ liệu

Tạo và xóa các phiên bản:
Lớp có một thuộc tính duy nhất, biên độ và điều này được đặt thành giá trị ban đầu bởi phương thức khởi tạo. Hàm tạo được tự động gọi ngay sau khi một phiên bản Sonic_pulse mới được tạo. Để tạo một xung âm mới có biên độ ban đầu là 131,4 đơn vị, chúng ta sẽ viết bằng C ++:

```
Sonic_pulse* myinstance = new Sonic_pulse(131.4);   //technique 1
```

or:

```
Sonic_pulse myinstance(131.4);                      //technique 2
```

Biểu diễn dữ liệu

Tạo và xóa các phiên bản:
Nếu một hàm hủy đã được xác định cho một lớp, nó sẽ tự động được gọi bất cứ khi nào một cá thể bị xóa. Nếu thể hiện được tạo bằng kỹ thuật 1, thì lập trình viên phải xóa nó một cách rõ ràng. Nếu nó được tạo bằng kỹ thuật 2, nó sẽ tự động bị xóa khi vượt ra khỏi phạm vi kiểm soát của chuỗi chương trình.
Lưu ý rằng các thành viên dữ liệu trong C ++ thường được cung cấp tiền tố m_, do đó biên độ, ở trên, sẽ trở thành m_amp Biên độ. Tuy nhiên, quy ước này đã không được thông qua ở đây vì nó sẽ gây nhầm lẫn so sánh giữa các phân đoạn của mã C ++ và mã Smalltalk.

Kế thừa

Kế thừa đơn:

Quay trở lại với ví dụ của chúng ta về mô phỏng siêu âm, lưu ý rằng có hai lớp xung âm, đó là Longitudinal_pulse và Shear_pulse. Mặc dù có một số khác biệt giữa hai lớp, nhưng cũng có nhiều điểm tương đồng. Sẽ là khó sử dụng nhất nếu tất cả thông tin phổ biến này phải được chỉ định hai lần. Vấn đề này có thể tránh được bằng cách sử dụng kế thừa, đôi khi được gọi là dẫn xuất. Một lớp Sonic_pulse được định nghĩa bao gồm cả hai loại xung. Tất cả các thuộc tính và phương thức phổ biến có thể được xác định tại đây. Các lớp Longitudinal_pulse và Shear_pulse sau đó được chỉ định là các lớp con hoặc chuyên môn của Sonic_pulse. Ngược lại, Sonic_pulse được cho là lớp cha hoặc tổng quát của hai lớp còn lại. Mối quan hệ phụ / siêu lớp có thể được coi như là một loại, tức là xung cắt là một loại xung âm. Các biểu thức sau, thường được sử dụng để mô tả mối quan hệ is-a-kind-of, là tương đương.

Kế thừa

Kế thừa đơn:

Các biểu thức sau, thường được sử dụng để mô tả mối quan hệ is-a-kind-of, là tương đương:

• một lớp con là-một-loại-của lớp cha;
• con cái là-một-loại-cha mẹ;
• một lớp dẫn xuất là-một-loại-lớp cơ sở;
• một lớp chuyên biệt là-một-loại-một-loại-chung của lớp học;
• một lớp chuyên biệt kế thừa từ một lớp tổng quát.

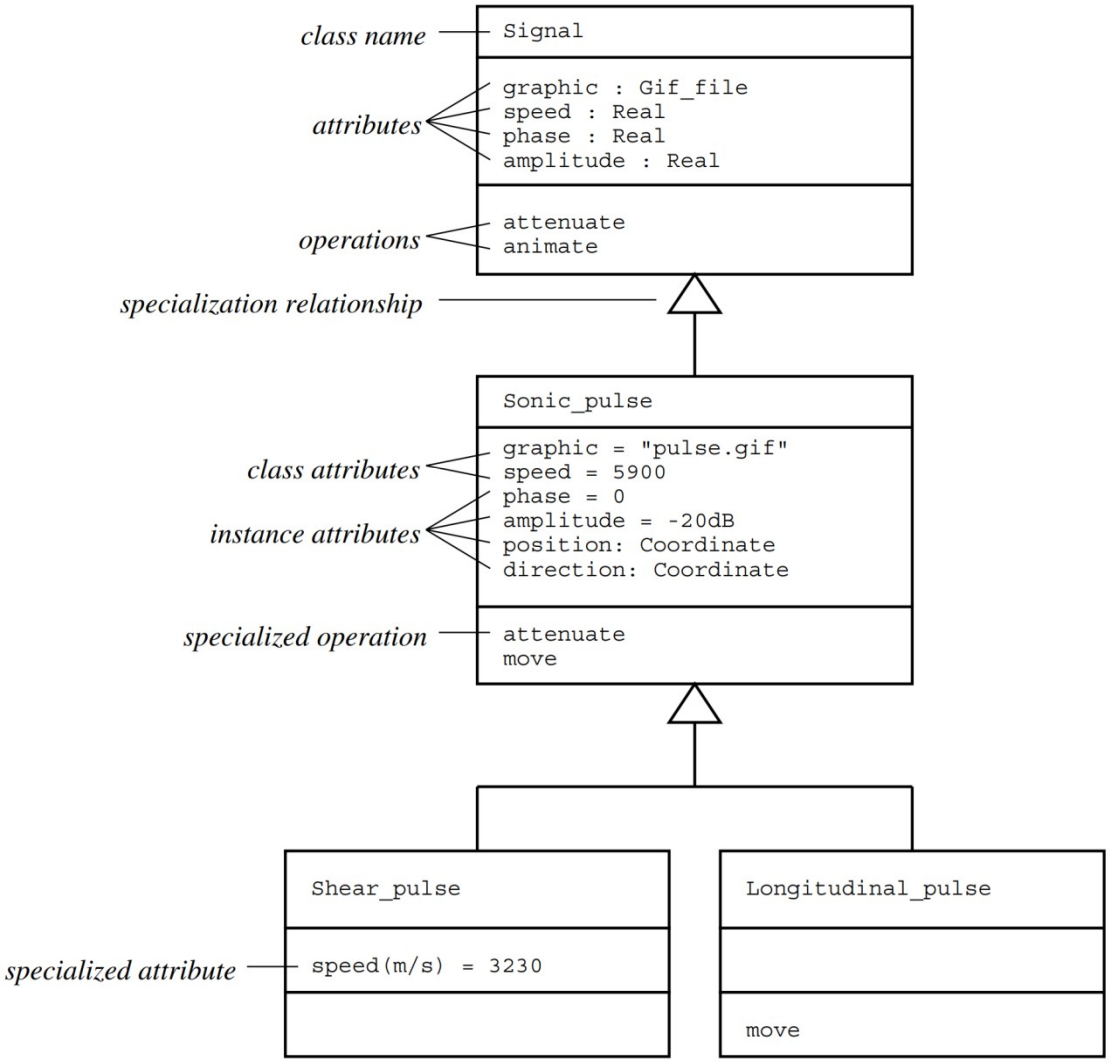
Khi định nghĩa một lớp dẫn xuất, chỉ cần xác định tên của lớp cơ sở và những điểm khác biệt so với lớp cơ sở. Các thuộc tính và giá trị ban đầu của chúng (nếu được cung cấp) và các phương thức được kế thừa bởi lớp dẫn xuất. (Trong C ++, có một số ngoại lệ không được kế thừa, đặc biệt là hàm tạo, hàm hủy và toán tử nạp chồng =, được mô tả trong Phần 4.11.3.) Bất kỳ thuộc tính hoặc phương thức nào được định nghĩa lại trong lớp dẫn xuất được cho là chuyên biệt, cũng giống như lớp dẫn xuất được cho là một chuyên biệt hóa của lớp cơ sở.

Kế thừa

Kế thừa đơn:

Một dạng chuyên môn hóa khác là giới thiệu các thuộc tính và phương thức bổ sung trong lớp dẫn xuất. C ++ cung cấp cho người dùng một số quyền kiểm soát đối với các phương thức và thuộc tính của lớp cơ sở có thể truy cập được từ lớp dẫn xuất và phương thức nào không (Phần 4.6). Nhiều ngôn ngữ OOP khác giả định rằng tất cả các thuộc tính và phương thức của lớp cơ sở đều có thể truy cập được từ lớp dẫn xuất, ngoại trừ những thuộc tính và phương thức được chuyên biệt hóa rõ ràng.

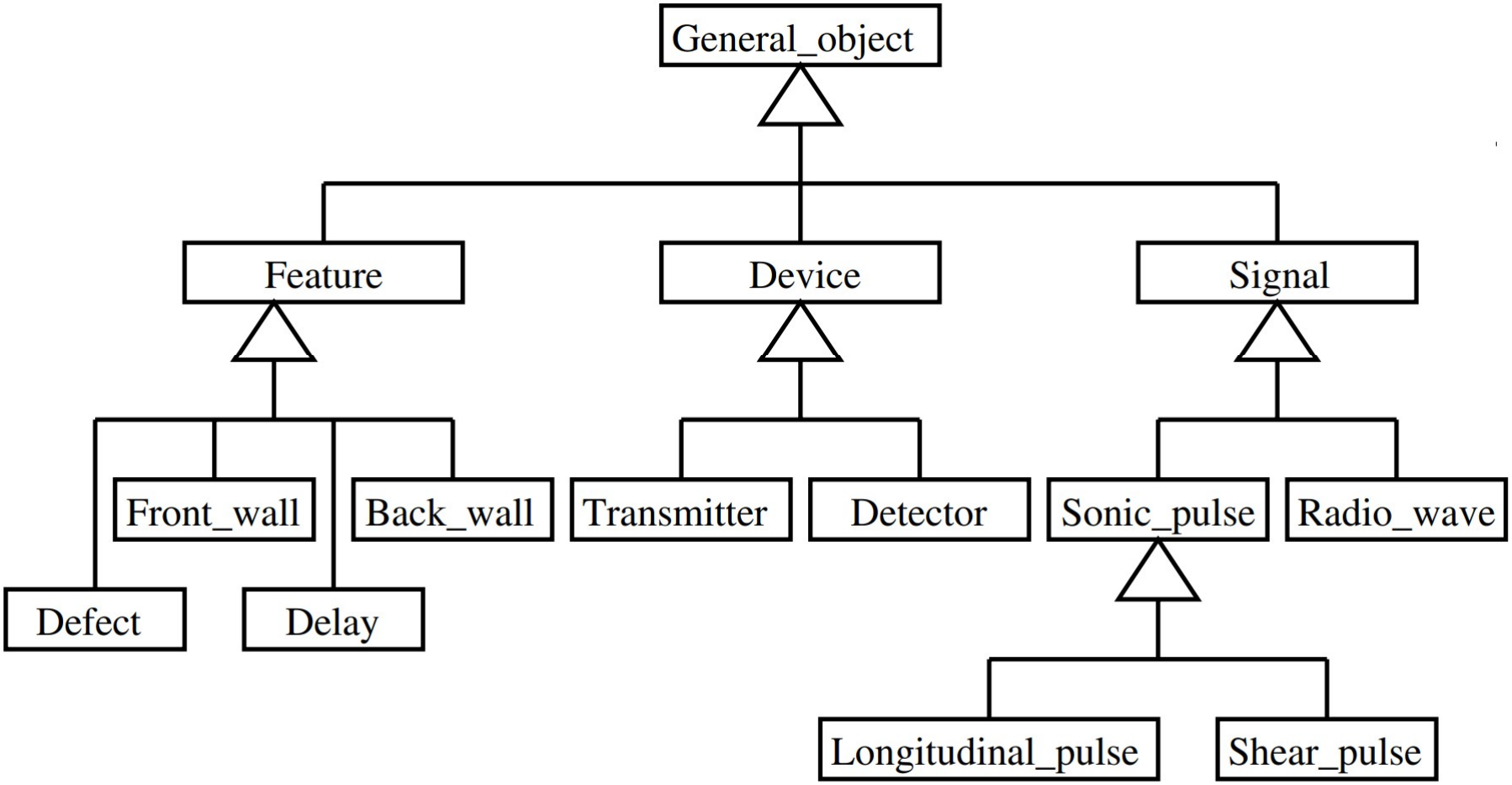
Kế thừa



Kế thừa

Kế thừa đơn:
Hình 4.4 cho thấy các định nghĩa lớp cho Sonic_pulse, Superclasss Signal của nó và các lớp con Shear_pulse và Longitudinal_pulse. Các thuộc tính đồ họa, tốc độ, pha và biên độ đều được khai báo ở lớp cấp cao nhất và được kế thừa bởi các lớp khác. Đồ họa biến lớp, xác định cách trình bày trên màn hình của một cá thể, được gán một giá trị ở mức Sonic_pulse. Giá trị này, cũng như phần khai báo của thuộc tính, được kế thừa bởi các lớp con. Các giá trị cho biên độ và pha của biến cá thể được kế thừa theo cùng một cách. Các giá trị kế thừa có thể bị ghi đè khi một phiên bản được tạo hoặc sau đó. Tốc độ biến lớp cho Shear_pulse được gán một giá trị khác với giá trị mà nếu không sẽ được kế thừa.

Kế thừa



Kế thừa

Kế thừa đơn:
Các hoạt động cũng như các thuộc tính được kế thừa. Do đó, các trường hợp của Shear_pulse và Longitudinal_pulse có quyền truy cập vào các hoạt động hoạt hình và giảm dần. Cái trước được kế thừa qua hai thế hệ, trong khi cái sau được chuyên biệt hóa ở cấp lớp Sonic_pulse. Di chuyển hoạt động được xác định ở cấp lớp Sonic_pulse và được Shear_pulse kế thừa, nhưng được định nghĩa lại cho Longitudinal_pulse.
Hình 4.5 cho thấy sự kế thừa giữa các đối tượng khác trong mô phỏng siêu âm. Lưu ý rằng mối quan hệ có tính chất bắc cầu, tức là:

If x is-a-kind-of y and y is-a-kind-of z, then x is-a-kind-of z.

Do đó, lớp Defect, ví dụ, kế thừa thông tin được xác định ở cấp Tính năng và ở cấp Đối tượng chung.

Kế thừa

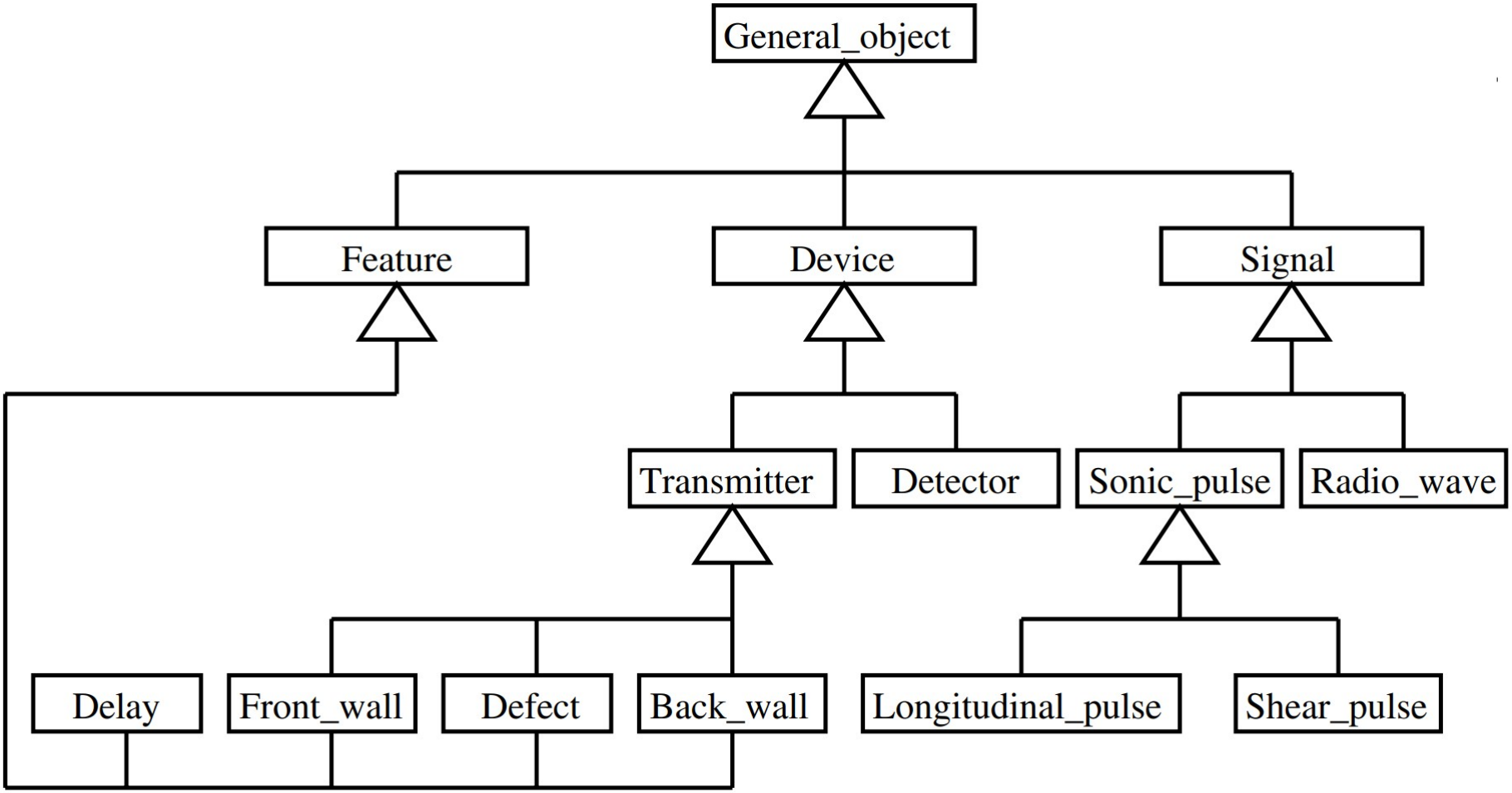
Đa kế thừa (kế thừa nhiều lần) và lặp lại:

Trong ví dụ minh họa trong Hình 4.5, mỗi con cái chỉ có một cha mẹ. Các mối quan hệ chuyên môn hóa do đó tạo thành một hệ thống phân cấp. Một số ngôn ngữ OOP nhấn mạnh vào kế thừa phân cấp. Tuy nhiên, những người khác cho phép con cái thừa kế từ nhiều hơn một cha mẹ. Đây được gọi là đa kế thừa.

Khi đa kế thừa xảy ra, các mối quan hệ chuyên môn hóa giữa các lớp xác định một mạng chứ không phải là một hệ thống phân cấp.

Hình 4.6 cho thấy cách đa kế thừa có thể được áp dụng cho mô phỏng siêu âm. Các phần của thành phần tương tác với xung sonic đều kế thừa từ Class Feature. Ba trong số các lớp bắt nguồn từ Tính năng được yêu cầu để mô phỏng phản xạ từng phần của các xung. Điều này được thực hiện bằng cách tạo ra một hoặc nhiều xung phản xạ, trong khi xung hiện có được truyền theo hướng thuận với biên độ giảm dần. Mã để tạo xung được chứa trong định nghĩa lớp cho Máy phát. Đa kế thừa cho phép những lớp cần tạo xung (Front_wall, Back_wall và Defect) kế thừa khả năng này từ Transmitter, đồng thời kế thừa các chức năng và thuộc tính khác từ Feature.

Kế thừa



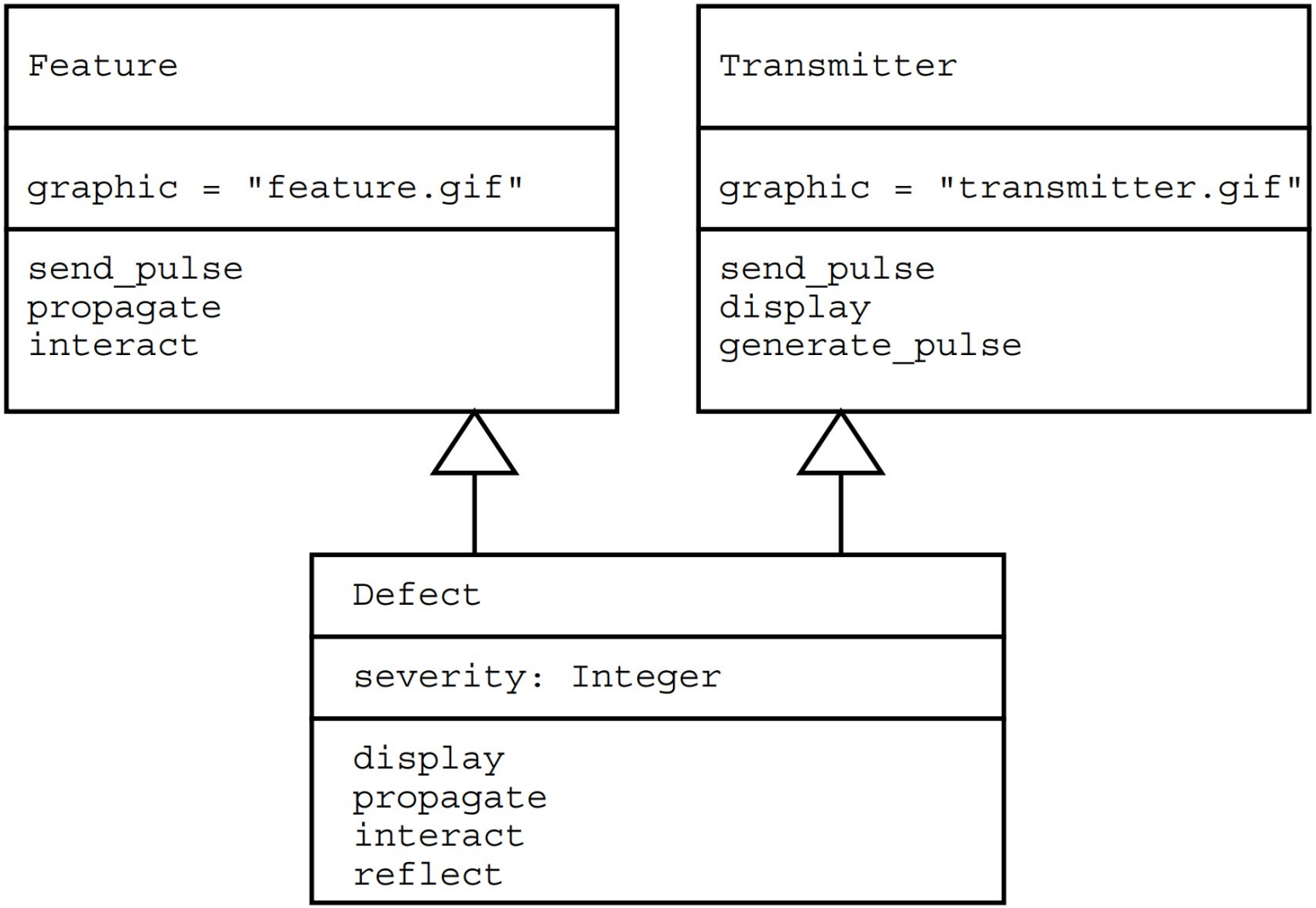
Kế thừa

Đa kế thừa (kế thừa nhiều lần) và lặp lại:

Mặc dù đa kế thừa hữu ích, nhưng nó có thể gây ra sự mơ hồ. Điều này được minh họa trong Hình 4.7, trong đó Defect kế thừa graphic feature.gif từ Feature và đồng thời kế thừa graphic receiver.gif từ Transmitter. Điều này đặt ra hai câu hỏi. Hai thuộc tính trùng tên có tham chiếu đến cùng một thuộc tính không? Nếu vậy, lớp Defect có thể chỉ có một giá trị tương ứng với đồ họa thuộc tính, vậy giá trị nào nên được chọn? Tương tự, Defect kế thừa các định nghĩa xung đột cho hoạt động send_pulse.

Cách đáng tin cậy nhất để giải quyết những xung đột như vậy là để trình biên dịch ngôn ngữ phát hiện ra chúng, để sau đó lập trình viên có thể trình bày rõ ràng ý nghĩa dự định. Một số môi trường OOP cho phép người dùng đặt chiến lược mặc định để giải quyết xung đột. Một ví dụ có thể là ưu tiên cho lớp cha gần nhất hoặc xa nhất với gốc của cây kế thừa. Điều này sẽ không hữu ích với ví dụ trong Hình 4.7 vì hai cha mẹ cách đều nhau từ gốc. Ngoài ra, tên lớp có thể được coi là quan trọng theo một cách nào đó, hoặc ưu tiên có thể được dành cho chuyên môn được xác định gần đây nhất.

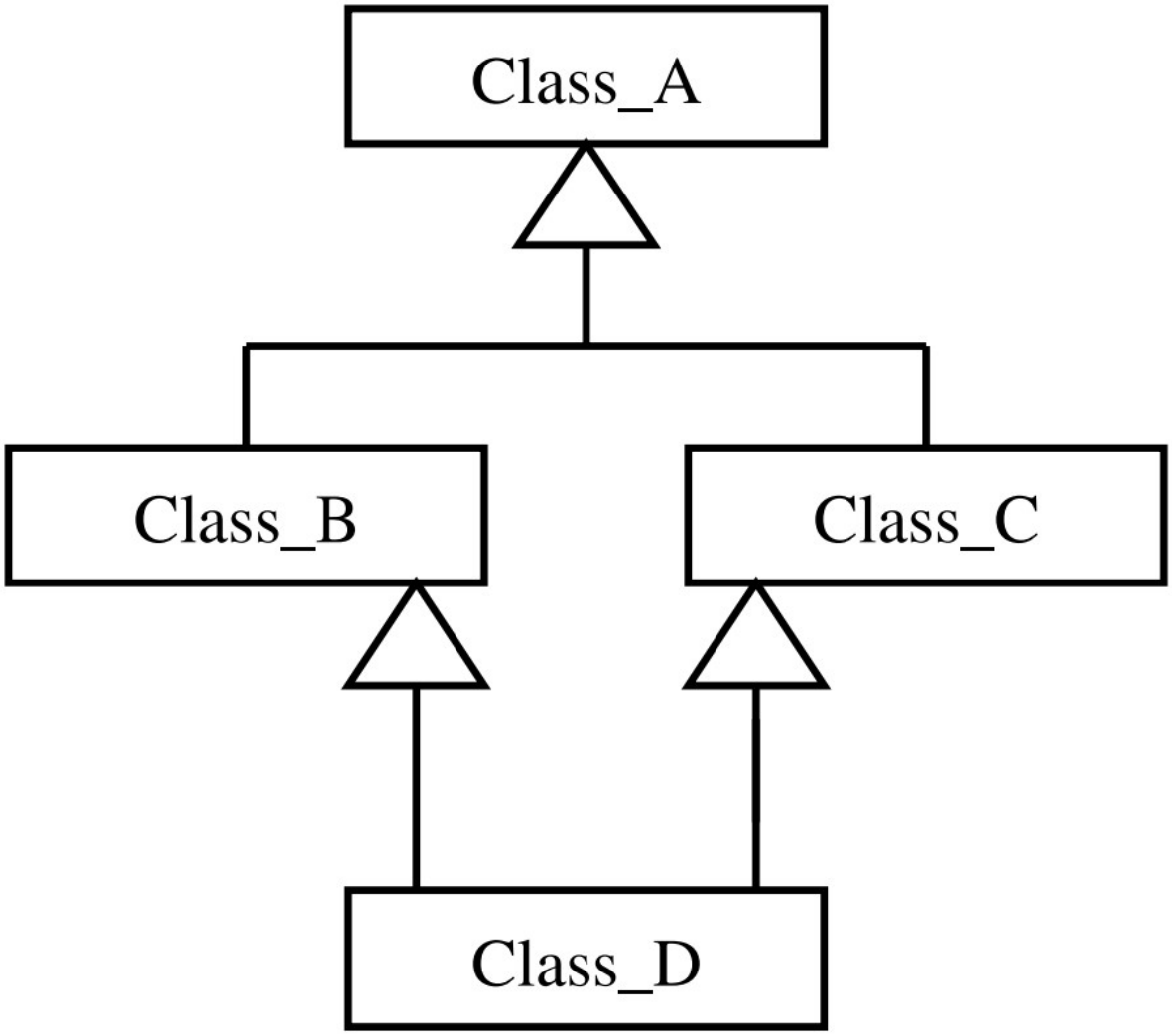
Kế thừa



Kế thừa

Đa kế thừa (kế thừa nhiều lần) và lặp lại:
Một vấn đề nữa là một lớp có thể được kế thừa gián tiếp từ lớp khác thông qua nhiều hơn một tuyến, như trong Hình 4.8. Đây được gọi là kế thừa lặp lại. Mặc dù ý nghĩa của lập trình viên có thể rõ ràng, nhưng ngôn ngữ OOP phải có một số chiến lược để nhận biết và xử lý sự kế thừa lặp đi lặp lại, nếu nó cho phép. C ++ cung cấp cho người lập trình một sự lựa chọn trong số hai chiến lược. Lớp D có thể có hai bản sao của Lớp A, một bản cho mỗi tuyến kế thừa. Ngoài ra, nó có thể chỉ có một bản sao duy nhất, nếu cả Lớp B và Lớp C đều được khai báo có Lớp A là lớp cơ sở ảo.
Đa kế thừa làm nảy sinh ý tưởng về mixin, là các lớp được thiết kế chỉ để giúp tổ chức cấu trúc kế thừa. Không thể tạo phiên bản mixin. Ví dụ, hãy xem xét một hệ thống phân cấp lớp cho các vật liệu kỹ thuật. Các vật liệu polyethylene, Bakelite, vàng, thép và silicon nitride có thể được phân loại là polyme, kim loại hoặc gốm sứ. Kế thừa đơn sẽ cho phép chúng ta xây dựng các mối quan hệ phân cấp này. Theo đa kế thừa, chúng ta có thể phân loại các tài liệu theo nhiều cách khác nhau cùng một lúc. Hình 4.9 cho thấy việc sử dụng hỗn hợp Rẻ và Dòn cho mục đích này.

Kế thừa



Kế thừa

Chuyên môn hóa các phương thức/phương pháp:
Chúng ta đã thấy rằng chuyên môn hóa có thể liên quan đến việc giới thiệu các phương thức hoặc thuộc tính mới, ghi đè các phép gán mặc định cho các thuộc tính hoặc định nghĩa lại các hoạt động kế thừa. Nếu một hoạt động cần được xác định lại, tức là chuyên biệt, không phải lúc nào cũng cần phải viết lại nó từ đầu. Trong ví dụ của chúng tôi, một định nghĩa về phương thức truyền được kế thừa bởi lớp Defect từ lớp Feature. Phiên bản chuyên biệt của sự lan truyền cũng giống như vậy, ngoại trừ xung âm thanh phải được làm suy giảm. Điều này có thể đạt được bằng cách gọi định nghĩa kế thừa từ bên trong định nghĩa chuyên biệt và sau đó thêm lệnh suy giảm, được hiển thị ở đây trong C ++:

```
void Defect::propagate(Sonic_pulse* inst1)
{
  Feature::propagate(inst1);
    // propagate a pulse, inst1, using inherited version of
    // 'propagate'
  inst1->attenuate(0.5);                  // Attenuate the pulse.
    // The member function 'attenuate' must be defined for the
    // class 'Sonic_pulse'.
}
```

Kế thừa

Chuyên môn hóa các phương thức/phương pháp:
Chúng ta có thể muốn gọi phương pháp chuyên biệt truyền từ bên trong định nghĩa của tương tác, một hàm xử lý tương tác tổng thể của một xung có khiếm khuyết:

```
void Defect::interact(Sonic_pulse* inst1)
{
  propagate(inst1);
    // propagate a pulse, inst1, using the locally defined
    // member function
  reflect(inst1);
    // generate a new pulse and send it in the opposite
    // direction
}
```

Kế thừa

Trình duyệt:

Nhiều hệ thống OOP không chỉ cung cấp một ngôn ngữ mà còn cung cấp một môi trường để lập trình. Môi trường có thể bao gồm các công cụ giúp OOP dễ dàng hơn và một trong những công cụ quan trọng nhất trong số những công cụ này là trình duyệt lớp. Một trình duyệt lớp hiển thị các mối quan hệ kế thừa, thường ở dạng cây như trong Hình 4.5 và 4.6. Nếu người lập trình muốn thay đổi một lớp hoặc chuyên môn hóa nó để tạo một lớp mới, họ chỉ cần chọn lớp từ trình duyệt và sau đó chọn kiểu thay đổi từ menu. Ngẫu nhiên, bản thân các trình duyệt luôn được xây dựng bằng OOP. Do đó, class-browser có thể là một thể hiện của một lớp được gọi là Class_browser, có thể là một chuyên môn của Trình duyệt, bản thân nó là một chuyên môn của Window. Lớp Class_browser có thể được chuyên môn hóa hơn nữa để cung cấp các phương tiện hiển thị hoặc chỉnh sửa khác nhau.

Đóng gói

Đóng gói, hay ẩn thông tin, là một thuật ngữ được sử dụng để diễn đạt khái niệm rằng các thuộc tính cá thể và các phương thức xác định một đối tượng thuộc về đối tượng đó chứ không thuộc về đối tượng nào khác. Do đó, các phương thức và thuộc tính là riêng tư và được cho là được đóng gói trong đối tượng. Giao diện của mỗi đối tượng tiết lộ càng ít càng tốt các hoạt động bên trong của đối tượng. Đối tượng có quyền kiểm soát dữ liệu của chính nó và những dữ liệu đó không thể bị thay đổi trực tiếp bởi các đối tượng khác. Các thuộc tính lớp là một ngoại lệ, vì chúng không được đóng gói trong bất kỳ cá thể nào mà được chia sẻ giữa tất cả các cá thể của cùng một lớp.

Nhìn chung, Smalltalk tuân thủ nguyên tắc đóng gói, trong khi C ++ áp dụng cách tiếp cận tự do hơn. Trong Smalltalk, đối tượng A chỉ có thể ảnh hưởng đến đối tượng B bằng cách gửi cho B một thông báo yêu cầu đối tượng gọi một trong các phương thức của nó (xem Phần 4.9). Ngoài cơ chế này, bí mật được duy trì giữa các đối tượng. Phương thức trên B có thể truy cập và thay đổi dữ liệu của chính nó, nhưng nó không thể truy cập dữ liệu của bất kỳ đối tượng nào khác.

Đóng gói

C ++ mang lại sự linh hoạt trong việc thực thi đóng gói thông qua các điều khiển truy cập. Tất cả các thành viên dữ liệu và các chức năng thành viên (gọi chung là thành viên) được cấp phát một trong bốn cấp độ truy cập:

• private: chỉ truy cập từ các hàm thành viên của lớp này (mặc định);

• được bảo vệ: quyền truy cập từ các hàm thành viên của lớp này và của các lớp dẫn xuất;

• công khai: truy cập từ bất kỳ phần nào của chương trình;

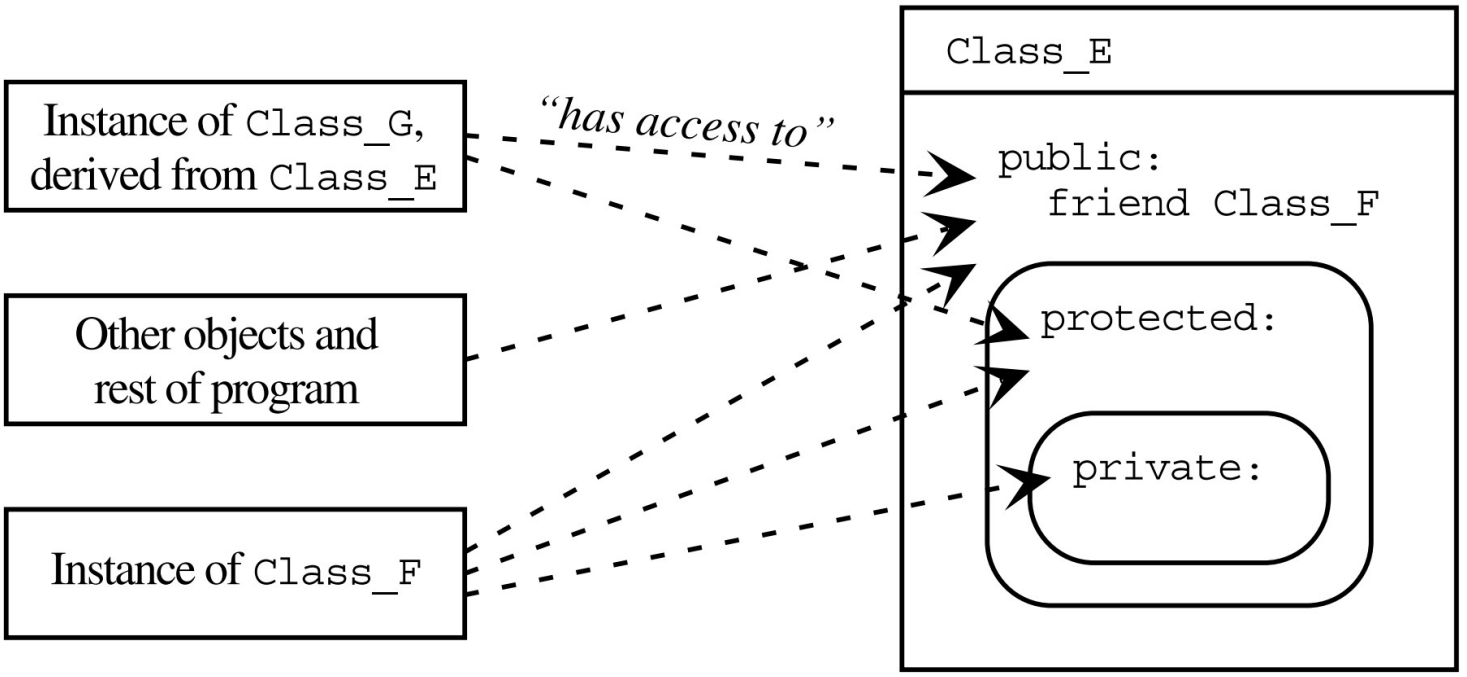
• bạn bè: truy cập từ các chức năng thành viên của các lớp được chỉ định.

Các điều khiển truy cập được minh họa trong Hình 4.10. C ++ cho phép hai kiểu dẫn xuất (tức là kế thừa), công khai và riêng tư. Trong dẫn xuất riêng, các thành viên được bảo vệ và công khai trong lớp cơ sở trở thành thành viên riêng của lớp dẫn xuất. Trong dẫn xuất công khai, mức độ truy cập của các thành viên trong lớp dẫn xuất là không thay đổi so với lớp cơ sở. Trong cả hai trường hợp, lớp dẫn xuất không có quyền truy cập vào các thành viên riêng của lớp cơ sở. Hai loại dẫn xuất được thể hiện trong Hình 4.11.

Ngôn ngữ mô hình hợp nhất (UML)

Ngôn ngữ mô hình hóa hợp nhất (UML) [2] cung cấp một phương tiện xác định mối quan hệ giữa các lớp và cá thể và biểu diễn chúng theo sơ đồ. Chúng tôi đã bắt gặp hai mối quan hệ như vậy:

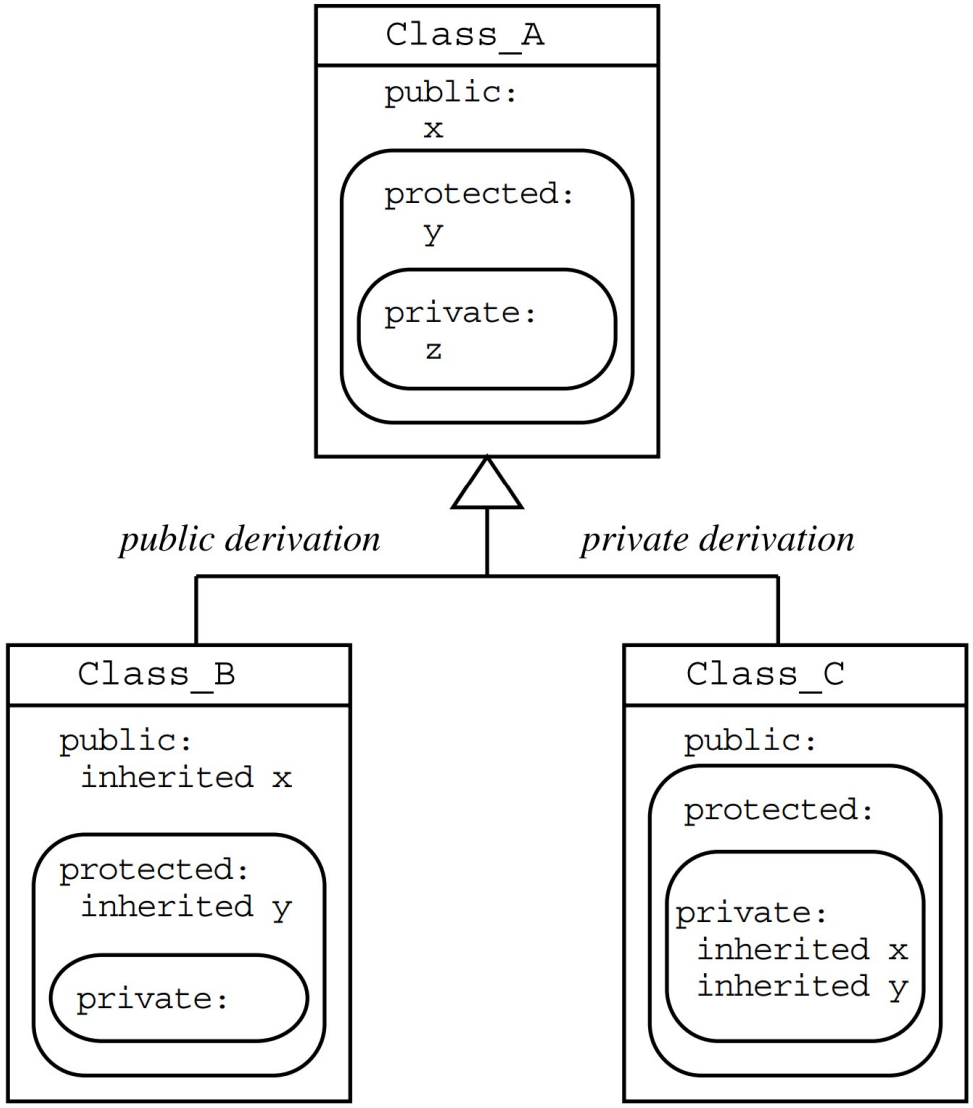
- chuyên môn hóa / tổng quát hóa;
- Phiên bản.



Ngôn ngữ mô hình hợp nhất (UML)

Sự đặc biệt hóa mô tả mối quan hệ là một loại giữa một lớp con và lớp cha, và liên quan đến việc kế thừa thông tin chung. Thuyết minh mô tả mối quan hệ giữa một lớp và một thể hiện của lớp đó. Một thể hiện của một lớp và một lớp con của một lớp đều được cho là khách hàng của lớp đó, vì cả hai đều lấy thông tin từ nó, nhưng theo những cách khác nhau.

Ngôn ngữ mô hình hợp nhất (UML)



Ngôn ngữ mô hình hợp nhất (UML)

Bây giờ chúng ta sẽ xem xét thêm ba loại mối quan hệ:

- tổng hợp;
- thành phần;
- sự kết hợp.

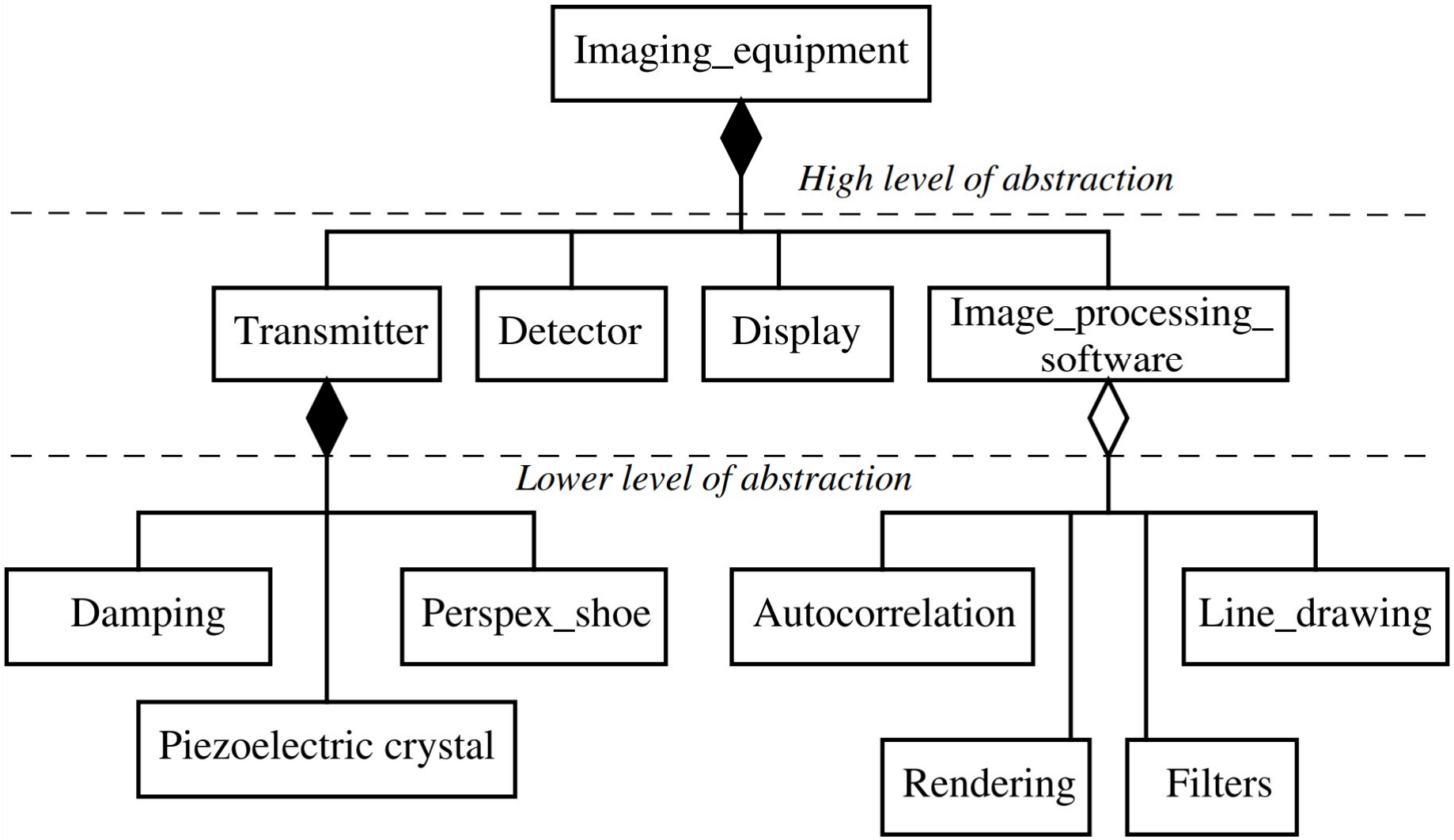
Ngôn ngữ mô hình hợp nhất (UML)

Mối quan hệ ***tổng hợp*** tồn tại khi một đối tượng có thể được xem như bao gồm một số đối tượng subobject. Bản thân thế giới phần mềm cung cấp một ví dụ điển hình, vì một thư viện phần mềm có thể được xem như một tập hợp của nhiều mô-đun thành phần. Mức độ sở hữu các mô-đun của thư viện phần mềm là khá yếu, theo nghĩa là cùng một mô-đun cũng có thể thuộc về một thư viện phần mềm khác. Mối quan hệ cấu thành là một trường hợp đặc biệt trong đó có mức độ sở hữu mạnh mẽ các bộ phận cấu thành. Ví dụ: một chiếc ô tô bao gồm động cơ, khung, cửa ra vào, ghế ngồi, bánh xe, v.v. Đây có thể được coi là quan hệ thành phần vì việc sao chép hoặc xóa một chiếc ô tô sẽ yêu cầu sao chép hoặc xóa các đối tượng thành phần này. Quay trở lại ví dụ về siêu âm của chúng ta, chúng ta có thể coi thiết bị hình ảnh như một thành phần của máy phát, máy dò, màn hình và phần mềm xử lý hình ảnh. Phần mềm xử lý ảnh là tập hợp của nhiều mô-đun khác nhau. Hình 4.12 cho thấy rằng các mối quan hệ lắp ráp và thành phần cho phép các vấn đề được nhìn nhận ở các mức độ trừu tượng khác nhau.

Ngôn ngữ mô hình hợp nhất (UML)

Liên hệ là những mối quan hệ lỏng lẻo giữa các đối tượng. Ví dụ, chúng có thể được sử dụng để đại diện cho bố cục không gian của các đối tượng, có thể bằng cách đặt tên cho đối tượng có liên quan dưới dạng thuộc tính đối tượng. Một dạng liên kết khác phát sinh khi một đối tượng sử dụng đối tượng khác bằng cách gửi thông điệp đến nó (hoặc, trong trường hợp C ++, gọi các hàm thành viên của nó hoặc truy cập các thành viên dữ liệu của nó) - xem Phần 4.9. Ví dụ, các xung có thể gửi thông báo đến đối tượng hiển thị để chúng có thể được hiển thị lại. Người gửi tin nhắn được gọi là đại lý và người nhận là máy chủ. Các đối tượng vừa gửi và nhận tin nhắn đôi khi được gọi là đại lý, mặc dù đây là cách sử dụng từ này gây nhầm lẫn, vì đại lý có một ý nghĩa khác được mô tả trong Chương 5.

Ngôn ngữ mô hình hợp nhất (UML)



Ngôn ngữ mô hình hợp nhất (UML)

Liên hệ là những mối quan hệ lỏng lẻo giữa các đối tượng. Ví dụ, chúng có thể được sử dụng để đại diện cho bố cục không gian của các đối tượng, có thể bằng cách đặt tên cho đối tượng có liên quan dưới dạng thuộc tính đối tượng. Một dạng liên kết khác phát sinh khi một đối tượng sử dụng đối tượng khác bằng cách gửi thông điệp đến nó (hoặc, trong trường hợp C ++, gọi các hàm thành viên của nó hoặc truy cập các thành viên dữ liệu của nó) - xem Phần 4.9. Ví dụ, các xung có thể gửi thông báo đến đối tượng hiển thị để chúng có thể được hiển thị lại. Người gửi tin nhắn được gọi là đại lý và người nhận là máy chủ. Các đối tượng vừa gửi và nhận tin nhắn đôi khi được gọi là đại lý, mặc dù đây là cách sử dụng từ này gây nhầm lẫn, vì đại lý có một ý nghĩa khác được mô tả trong Chương 5.

Ngôn ngữ mô hình hóa hợp nhất (UML) [2] bao gồm biểu diễn dạng sơ đồ cho các mối quan hệ giữa các lớp, được tóm tắt trong Hình 4.13. Bất cứ khi nào thích hợp, ký hiệu này đã được sử dụng trong suốt chương này.

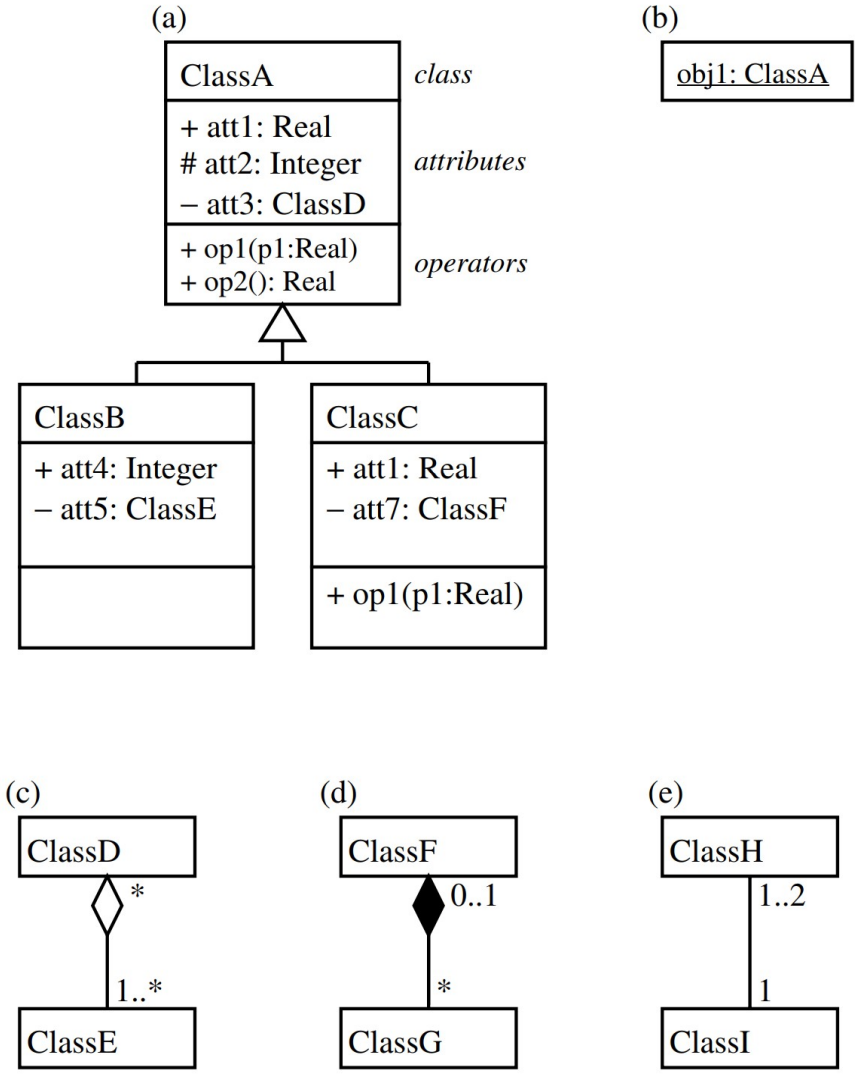
Liên kết/ràng buộc động (hoặc trễ)

Ba tính năng cần thiết của ngôn ngữ OOP, như được định nghĩa trong Phần 4.3, là trừu tượng hóa dữ liệu, kế thừa và đóng gói. Tính năng cần thiết thứ tư và cuối cùng là ràng buộc động (hoặc ràng buộc muộn). Mặc dù các lớp cha của các đối tượng có thể được biết đến tại thời điểm biên dịch, nhưng các lớp thực tế (dẫn xuất) có thể không. Lớp thực tế không bị ràng buộc với tên đối tượng tại thời điểm biên dịch, nhưng thay vào đó, ràng buộc được hoãn lại cho đến thời gian chạy. Điều này được gọi là ràng buộc động và ý nghĩa của nó được thể hiện rõ nhất bằng ví dụ.

Giả sử rằng chúng ta có một phương thức, truyền, được định nghĩa trong C ++ cho lớp, Feature:

```
void Feature::propagate(Sonic_pulse* p)
{
  float x, y;
  x = getx();
```

Liên kết/ràng buộc động (hoặc trễ)



Liên kết/ràng buộc động (hoặc trễ)

Cũng giả sử rằng shear1 và long1 lần lượt là các thể hiện của Shear_pulse và Longitudinal_pulse. Khi chương trình được chạy, việc truyền shear1 và long1 bởi một tính năng được thực hiện bằng các lệnh gọi để truyền, với các con trỏ đến mỗi xung được truyền dưới dạng tham số. Các kiểu tham số là đúng, vì shear1 là một thể hiện của Shear_pulse, có nguồn gốc từ Sonic_pulse. Tương tự, long1 là một thể hiện của Longitudinal_pulse, cũng có nguồn gốc từ Sonic_pulse. Phương thức truyền lệnh gọi phương thức di chuyển, nhưng điều này có thể chuyên biệt khác nhau đối với shear1 và long1. Tuy nhiên, định nghĩa chính xác sẽ được chọn trong mỗi trường hợp. Đây là một ví dụ về liên kết trễ, vì phương thức di chuyển thực tế sẽ được sử dụng được xác định mỗi khi bước truyền đó được gọi, thay vì khi chương trình được biên dịch.

Liên kết/ràng buộc động (hoặc trễ)

Hiệu quả kết hợp của tính kế thừa và ràng buộc động là cùng một lệnh gọi hàm (di chuyển trong ví dụ trên) có thể có nhiều hơn một ý nghĩa và ý nghĩa thực tế không được giải thích cho đến thời gian chạy. Hiệu ứng này được gọi là đa hình.

Để hiểu tại sao tính đa hình lại quan trọng như vậy, chúng ta nên xem xét cách chúng ta giải quyết vấn đề trên bằng một ngôn ngữ mà ràng buộc là tĩnh (hoặc sớm). Trong các ngôn ngữ như vậy, ý nghĩa chính xác của mỗi lệnh gọi hàm được xác định tại thời điểm biên dịch, và được gọi là tính đơn hình. Trong phương thức tuyên truyền, chúng ta sẽ phải kiểm tra lớp đối số của nó và gọi một hành vi cho phù hợp. Tùy thuộc vào ngôn ngữ, chúng ta có thể cần bao gồm một biến lớp trên Sonic_pulse để gắn thẻ cho mỗi cá thể với tên của lớp của nó (có thể là Sonic_pulse hoặc một lớp được dẫn xuất từ nó).

Liên kết/ràng buộc động (hoặc trễ)

Hai nhược điểm là rõ ràng ngay lập tức. Đầu tiên, mã sử dụng tính đơn hình dài hơn nhiều và có khả năng bao gồm sự trùng lặp đáng kể, vì mã để di chuyển Shear_pulse sẽ tương tự như mã để di chuyển Longitudinal_pulse. Ngược lại, tính đa hình tránh được sự trùng lặp và cho phép tính tương đồng giữa các lớp được thể hiện rõ ràng. Thứ hai, mã này khó bảo trì hơn mã đa hình, bởi vì sau đó việc bổ sung một lớp xung mới sẽ yêu cầu những thay đổi đối với phương thức truyền trong đặc tính lớp. Điều này đi ngược lại triết lý đóng gói, bởi vì việc thêm một lớp sẽ không yêu cầu thay đổi đối với các lớp hiện có.
Tác dụng của tính đa hình là ngôn ngữ sẽ luôn chọn nghĩa hợp lý cho một lệnh gọi hàm. Vì vậy, một khi tầm quan trọng của nó đã được hiểu rõ, tính đa hình không cần thiết phải làm người lập trình khó chịu.

Gọi hàm và truyền bản tin

Trong C ++, các hàm thành viên (tương đương với các phương thức) được truy cập theo cách tương tự với bất kỳ hàm nào khác, ngoại trừ việc chúng ta cần phân biệt đối tượng mà hàm thành viên đó thuộc về. Trong ví dụ C ++ sau, các khuyết tật d1 và d2 được tạo ra bằng cách sử dụng hai kỹ thuật được giới thiệu trong Phần 4.4.5, và sau đó chức năng thành viên truyền bá sẽ được truy cập.

Một lệnh gọi đến một hàm thông thường cũng được bao gồm để so sánh:

```
result = somefunc(parameter);
    // call a conventional function, defined elsewhere.

Defect* d1 = new Defect();   // make a new instance d1 of Defect.
Sonic_pulse* p1 = new Sonic_pulse();
Shear_pulse* p2 = new Shear_pulse();
                  // make two new instances of pulses, p1 and p2.
d1->propagate(p1);
    // call member function, 'propagate', with parameter p1.

Defect d2; // make a new instance d2 of Defect.
d2.propagate(p2);
    // call member function, 'propagate', with parameter p2.
```

Gọi hàm và truyền bản tin

Trong Smalltalk và các ngôn ngữ OOP khác, các phương thức được gọi bằng cách chuyển các thông báo. Thuật ngữ này nhấn mạnh khái niệm đóng gói. Mỗi đối tượng là độc lập và phụ trách các phương pháp riêng của nó. Tuy nhiên, các đối tượng có thể tương tác. Một đối tượng có thể kích thích đối tượng khác kích hoạt một phương pháp bằng cách gửi một thông điệp đến nó. Một ví dụ về việc chuyển thông báo trong Smalltalk là:

D1 propagate: p1.

Điều này được biên dịch là:

Gửi tới phiên bản d1 thông điệp propagate:, với tham số p1.

Cú pháp Smalltalk trở nên khó hiểu hơn một chút khi có nhiều tham số: p1 move: x and: y.

Các đối số được xen kẽ với tên phương thức, làm cho thông báo được đọc giống như tiếng Anh. Ví dụ này có nghĩa là:

Gửi tới phiên bản p1 thông điệp move:and: với các tham số x và y.

Khi nhận được một tin nhắn, một đối tượng sẽ gọi phương thức tương ứng bằng các tham số được cung cấp.

Gọi hàm và truyền bản tin

Biến ảo/biến giả:

Smalltalk và một số ngôn ngữ OOP khác bao gồm các từ dành riêng cung cấp dạng viết tắt cho "đối tượng này" và "lớp cha của đối tượng này", do đó tránh được sự cần thiết phải đặt tên các lớp một cách rõ ràng. Trong Smalltalk, các từ dành riêng lần lượt là self và super. Chúng được gọi là biến giả; chúng khác với các biến bình thường bởi vì chúng không thể được người lập trình gán giá trị trực tiếp. C ++ bao gồm biến giả này, tương đương với bản thân của Smalltalk. Để minh họa việc sử dụng super, hãy xem xét ví dụ về đặc biệt truyền phương thức, được trình bày trong Phần 4.5.3 bằng cách sử dụng C ++. Đây là một tương đương với Smalltalk, được định nghĩa cho lớp Defect:

```
propagate: inst1
    super propagate: inst1.
"Propagate a pulse, inst1, using inherited version of propagate"

    inst1 attenuate: 0.5.    "Attenuate the pulse"
"The method 'attenuate' must be defined for the class of inst1"
```

Gọi hàm và truyền bản tin

Biến ảo/biến giả:
Phiên bản của truyền được xác định cho Feature lớp cha lần đầu tiên được gọi. Sau đó, inst1 được gửi thông báo giảm dần :, hướng dẫn nó áp dụng phương pháp giảm dần :, do đó làm giảm biên độ của nó. Người ta giả định rằng inst1 là một xung sonic.
Tương tự, phương thức tương tác được mô tả cho lớp Defect mô tả sự tương tác tổng thể giữa một xung và khiếm khuyết. Phương thức này bao gồm các lệnh gọi đến các phiên bản truyền và phản ánh được xác định cục bộ, được chỉ định trong Smalltalk bằng cách sử dụng tự:

```
interact: inst1
    self propagate: inst1.
"Propagate a pulse, inst1, using the locally defined method"


    self reflect: inst1.
"Generate a new pulse and send it in the opposite direction"
```

Chương 5

Đại lý thông minh

Các đặc tính của một đại lý thông minh

Khi thế giới thông tin ngày càng mở rộng, con người ngày càng ít có khả năng hành động hơn với số lượng ngày càng tăng của thông tin được cung cấp cho họ. Một cách giải quyết vấn đề này là xây dựng các đại lý thông minh - các trợ lý phần mềm đảm nhiệm các nhiệm vụ cụ thể cho chúng ta. Ví dụ: nếu bạn muốn tìm kiếm một phần thông tin cụ thể trên World Wide Web, bạn có thể sử dụng một đại lý thông minh để tham khảo lựa chọn các công cụ tìm kiếm và lọc các trang web cho bạn. Bằng cách này, bạn chỉ được trình bày với hai hoặc ba trang phù hợp chính xác với nhu cầu của bạn. đại lý thông minh thuộc loại này, tự cá nhân hóa theo yêu cầu cá nhân của bạn bằng cách tìm hiểu thói quen và sở thích của bạn, được gọi là đại lý người dùng.

Các đặc tính của một đại lý thông minh

Tương tự, phần lớn giao dịch trên các sàn giao dịch chứng khoán của thế giới được thực hiện bởi các đại lý thông minh. Việc đạt được lợi ích của một số loại giao dịch cổ phiếu phụ thuộc vào việc phản ứng nhanh chóng với những biến động giá nhỏ. Vào thời điểm một nhà giao dịch con người đã đồng hóa dữ liệu và đưa ra quyết định, cơ hội sẽ mất đi.

Các đặc tính của một đại lý thông minh

Cũng giống như các đại lý thông minh cung cấp một cách để giảm bớt sự phức tạp trong thế giới thực, chúng cũng thực hiện một vai trò tương tự trong các hệ thống máy tính.

Khi các hệ thống phần mềm ngày càng trở nên lớn hơn và phức tạp hơn, việc duy trì chúng như là các hệ thống tập trung được thiết kế và thử nghiệm trong mọi trường hợp sẽ trở nên kém khả thi hơn.

Một cách tiếp cận thay thế là lấy ý tưởng về phần mềm mô-đun, cụ thể là, biến các mô-đun thành các đại lý tự trị có thể đưa ra quyết định thông minh của riêng họ trong nhiều trường hợp. Các đại lý như vậy cho phép hệ thống phần lớn tự quản lý, vì chúng có thể được cung cấp kiến thức về cách đối phó trong các tình huống cụ thể, thay vì được lập trình rõ ràng để xử lý mọi tình huống có thể thấy trước.

Các đặc tính của một đại lý thông minh

Lưu ý rằng không phải tất cả các đại lý đều thông minh, Wooldridge đưa ra định nghĩa sau cho một đại lý:

Đại lý là một hệ thống máy tính được đóng gói nằm trong một số môi trường và có khả năng hoạt động linh hoạt, tự chủ trong môi trường đó để đáp ứng các mục tiêu thiết kế của nó.

Định nghĩa trên cho thấy rằng ba đặc điểm chính của một đại lý **là tính tự chủ, tính bền bỉ và khả năng tương tác với môi trường của nó**.
Quyền tự chủ đề cập đến khả năng của một nhân viên để đưa ra quyết định của riêng mình dựa trên kinh nghiệm và hoàn cảnh của riêng họ, đồng thời kiểm soát trạng thái và hành vi nội bộ của chính họ.
Định nghĩa ngụ ý rằng một đại lý hoạt động liên tục trong môi trường của nó, tức là nó bền bỉ theo thời gian. Các đại lý cũng được cho là có cơ sở, tức là họ đáp ứng các yêu cầu của môi trường và có khả năng hành động theo yêu cầu đó.
Tương tác với môi trường vật chất yêu cầu nhận thức thông qua cảm biến và hành động thông qua bộ truyền động hoặc bộ tạo hiệu ứng. Tương tác với môi trường phần mềm thuần túy đơn giản hơn, chỉ yêu cầu quyền truy cập và thao tác dữ liệu và chương trình.

Các đặc tính của một đại lý thông minh

Trí thông minh có thể được thêm vào ở một mức độ lớn hơn hoặc thấp hơn, nhưng chúng ta có thể mong đợi một cách hợp lý một đại lý thông minh là tất cả những điều sau:

- ✓ Hồi đáp nhanh,
- ✓ Mục tiêu hướng dẫn,
- ✓ Thích nghi,
- ✓ Có khả năng xã hội

Các đặc tính của một đại lý thông minh

Năng lực xã hội đề cập đến khả năng hợp tác và thương lượng với các đại lý khác (hoặc con người).
Khá dễ dàng để hình dung một đại lý hoàn toàn là phản ứng, ví dụ: đại lý có vai trò duy nhất là đặt cảnh báo trên màn hình máy tính của bạn khi máy in hết giấy. Hành vi này tương tự như một daemon. Tương tự như vậy, các mô-đun của mã máy tính thông thường có thể được coi là hướng đến mục tiêu theo nghĩa hạn chế rằng chúng đã được lập trình để thực hiện một nhiệm vụ cụ thể bất kể môi trường của chúng là gì. Vì nó là tự chủ, một đại lý thông minh có thể quyết định các mục tiêu của riêng mình và lựa chọn các hành động của riêng mình để theo đuổi các mục tiêu đó. Đồng thời, nó cũng phải có khả năng phản ứng với những thay đổi bất ngờ trong môi trường của nó. Do đó, nó phải cân bằng giữa hành vi phản ứng và hướng đến mục tiêu, thường thông qua sự kết hợp giữa giải quyết vấn đề, lập kế hoạch, tìm kiếm, ra quyết định và học hỏi thông qua kinh nghiệm.

Các đặc tính của một đại lý thông minh

Không có lý do để một đại lý phải ở lại vĩnh viễn trên một máy tính. Nếu một máy tính được kết nối với mạng, một nhân viên di động có thể di chuyển đến các máy tính từ xa để thực hiện nhiệm vụ được chỉ định trước khi trở về nhà với nhiệm vụ đã hoàn thành. Nhiệm vụ điển hình của một nhân viên di động có thể là xác định kế hoạch du lịch của một người. Điều này sẽ yêu cầu thông tin về lịch trình tàu và hãng hàng không, cùng với tình trạng sẵn có của khách sạn. Thay vì chuyển một lượng lớn dữ liệu qua mạng từ các công ty xe lửa, hãng hàng không và khách sạn, hiệu quả hơn là đại lý tự chuyển đến các trang web từ xa này, tìm thông tin họ cần và trả lại. Rõ ràng, có khả năng sử dụng độc hại các đại lý di động, do đó, bảo mật là yếu tố hàng đầu được xem xét đối với các trang web chấp nhận chúng.

Đại lý và đối tượng

Với cachs tiếp cận tổ chức hệ thống theo đối tượng và khung cho phép các vấn đề phức tạp được chia nhỏ thành các thành phần đơn giản hơn trong khi vẫn duy trì tính toàn vẹn của hệ thống tổng thể. Mặc dù lịch sử của lập trình dựa trên đại lý có thể bắt nguồn từ những năm 1970, nhưng hiện nay nó được nhiều người coi là sự phát triển logic tiếp theo từ lập trình hướng đối tượng (OOP). Nếu các đối tượng được xem như những người hầu ngoan ngoãn, thì các đặc vụ thông minh có thể được coi là những sinh vật độc lập. Thật vậy, họ thường được gọi là các đại lý tự trị. Khi một đại lý nhận được yêu cầu thực hiện một hành động, nó sẽ đưa ra quyết định của riêng mình, dựa trên niềm tin của mình và theo đuổi mục tiêu của mình. Do đó, nó cư xử giống một cá nhân hơn với tính cách của riêng mình. Do đó, các hệ thống dựa trên đại lý tương tự như các xã hội hoặc tổ chức của con người.

Đại lý và đối tượng

Là một thực thể phần mềm được đóng gói, một đại lý thông minh có một số điểm tương đồng với một đối tượng. Tuy nhiên, nó khác nhau ở ba điểm chính:

Quyền tự chủ: Khi một đối tượng đã khai báo một phương thức là công khai, vì nó phải để phương thức trở nên hữu ích, nó sẽ mất quyền tự chủ của nó. Các đối tượng khác sau đó có thể gọi phương thức đó bằng cách gửi tin nhắn. Ngược lại, các đại lý không thể gọi các hành động của một đại lý khác, họ chỉ có thể đưa ra các yêu cầu. Quyết định về hành động cần thực hiện thuộc về người nhận tin nhắn chứ không phải người gửi. Quyền tự chủ không bắt buộc trong hệ thống hướng đối tượng vì mỗi đối tượng được thiết kế để thực hiện một nhiệm vụ nhằm theo đuổi mục tiêu chung của nhà phát triển. Tuy nhiên, không nhất thiết phải giả định rằng các đại lý sẽ chia sẻ một mục tiêu chung.

Thông minh: Mặc dù hành vi thông minh có thể được xây dựng trong một đối tượng, nhưng nó không phải là một yêu cầu của mô hình OOP.

Đại lý và đối tượng

Tính bền bỉ: các đối tượng có thể được tạo ra để tồn tại từ một lần chạy chương trình này sang lần chạy chương trình khác bằng cách lưu trữ chúng. Ngược lại, các đại lý tồn tại giống như liên tục được "bật" và do đó chúng hoạt động đồng thời. Trong khi đó, một hệ thống hướng đối tượng tiêu chuẩn có một luồng điều khiển duy nhất, với các đối tượng thực hiện các hành động một cách tuần tự.

Trong khi các ngôn ngữ OOP như C ++ và Smalltalk được định nghĩa khá chặt chẽ, không có phương pháp tiếp cận tiêu chuẩn nào để triển khai các hệ thống dựa trên đại lý. Một cách tiếp cận hướng đối tượng thường có thể được sử dụng trong việc triển khai hệ thống dựa trên đại lý, trong khi điều ngược lại là không chắc.

Kiến trúc đại lý

Bất kỳ kỹ thuật nào được đáp ứng cho đến nay trong tài liệu này đều có thể được sử dụng để cung cấp khả năng đại diện và lý luận bên trong của một đại lý. Tuy nhiên, có thể xác định một số kiểu tiếp cận khác nhau. Có ít nhất bốn trường phái suy nghĩ khác nhau về cách đạt được sự cân bằng thích hợp giữa hành vi phản ứng và hành vi hướng đến mục tiêu.

Kiến trúc dựa trên tính logic

Là cách tiếp cận ủng hộ suy luận logic dựa trên sự biểu thị mang tính biểu tượng của môi trường. Cách tiếp cận này rất tao nhã và chặt chẽ, nhưng nó dựa trên sự không thay đổi của môi trường trong quá trình lập luận. Nó cũng đưa ra những khó khăn đặc biệt trong việc đại diện một cách tượng trưng cho môi trường và lý luận về nó.

Kiến trúc đại lý

Kiến trúc dựa trên ứng xử

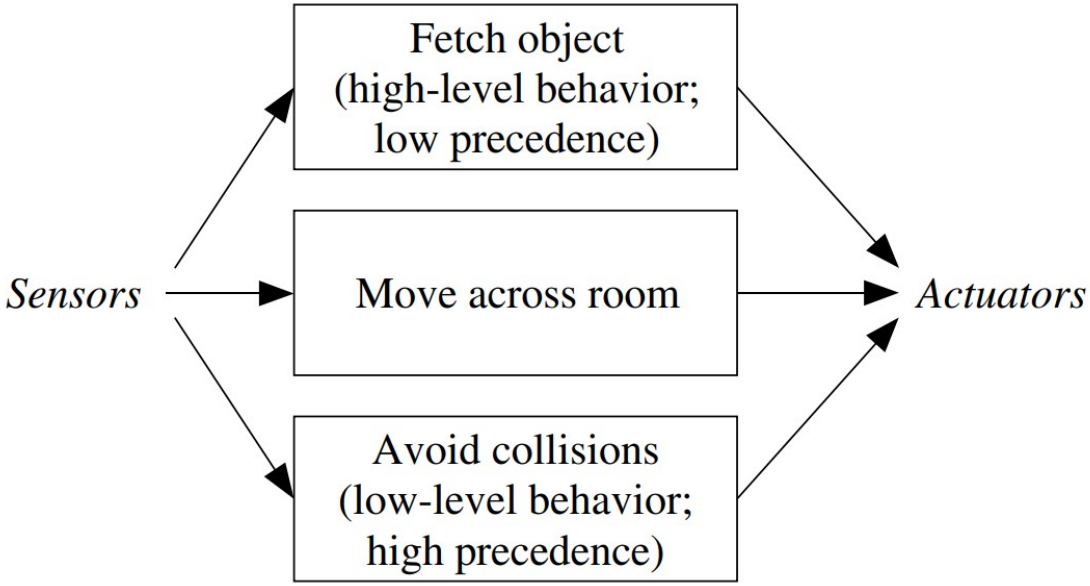
Ngược lại, các nhà nghiên cứu khác đề xuất rằng suy luận logic về môi trường là không thích hợp chi tiết và tốn thời gian. Ví dụ, nếu một vật nặng rơi về phía bạn, ưu tiên nên di chuyển ra khỏi đường hơn là phân tích và chứng minh quan sát. Các nhà nghiên cứu này gợi ý rằng các đại lý chỉ cần một tập hợp các phản ứng để phản ứng với hoàn cảnh và hành vi thông minh sẽ xuất hiện từ sự kết hợp của các phản ứng đó.

Loại kiến trúc này dựa trên các đại lý phản ứng, tức là các đại lý không bao gồm mô hình thế giới biểu tượng cũng như khả năng thực hiện lý luận biểu tượng phức tạp. Một ví dụ nổi tiếng về cách tiếp cận này là kiến trúc phụ của Brooks, chứa các mô-đun hành vi liên kết các hành động với các tình huống được quan sát mà không cần suy luận gì cả.

Kiến trúc đại lý

Kiến trúc dựa trên ứng xử

Các hành vi được sắp xếp thành một hệ thống phân cấp, trong đó hành vi cấp thấp như "tránh đối tượng" được ưu tiên hơn các hành vi hướng tới mục tiêu cấp cao hơn như "di chuyển khắp phòng". Cách tiếp cận đơn giản và thiết thực này có thể mang lại hiệu quả cao. Hạn chế chính của nó là sự tập trung vào môi trường địa phương có thể dẫn đến việc thiếu nhận thức về bức tranh toàn cảnh



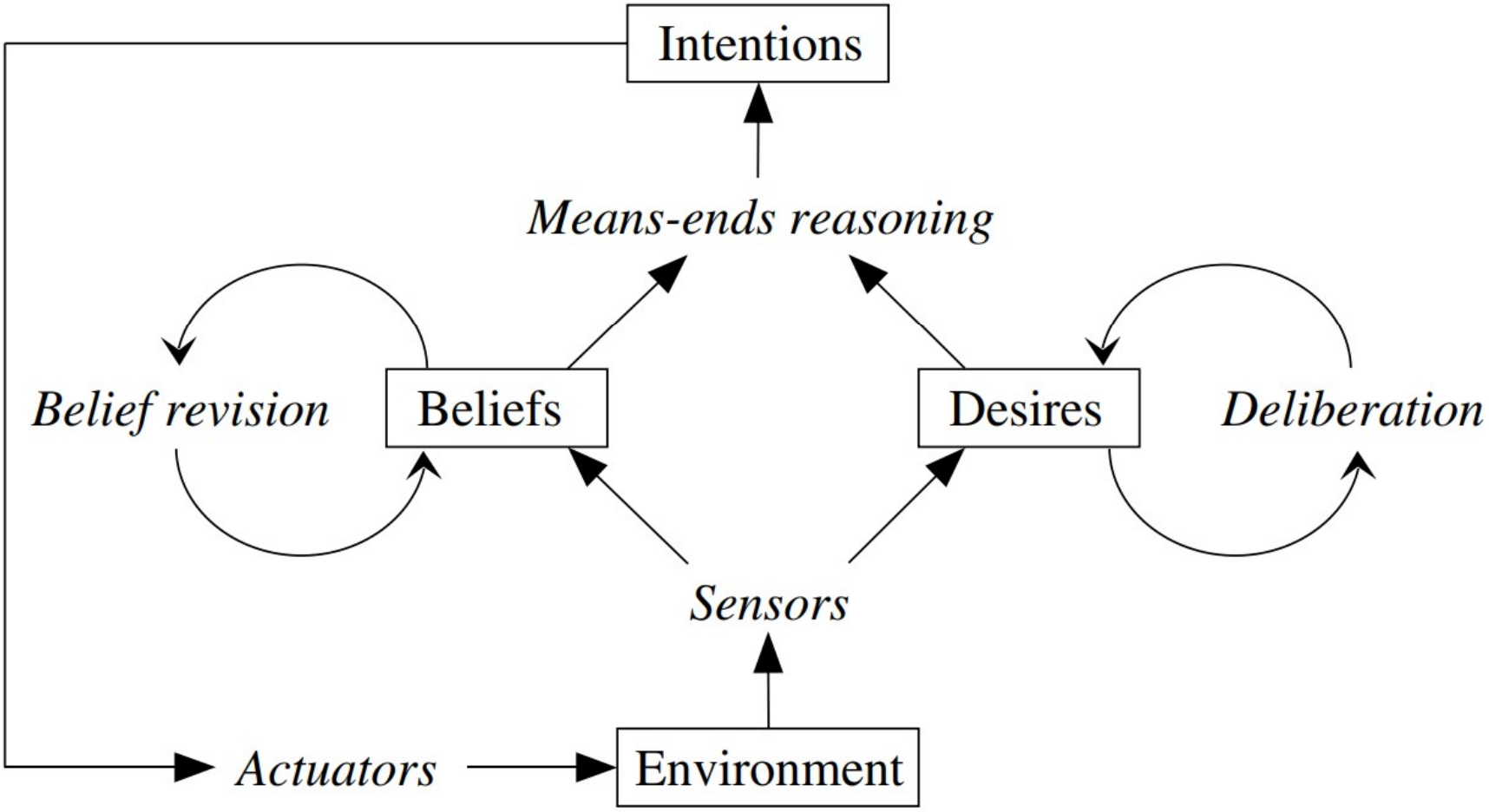
Kiến trúc đại lý

Kiến trúc theo cấp độ tri thức

Loại kiến trúc thứ ba, dựa trên các đại lý cấp tri thức, coi mỗi đại lý thông minh như một hệ thống dựa trên tri thức trong mô hình thu nhỏ. Chương 1 cho thấy rằng hệ thống dựa trên tri thức chứa cơ sở tri thức khác biệt và tách biệt với công cụ suy luận của nó, do đó, loại đại lý này được mô hình hóa với cấp độ tri thức, khác biệt với các cơ chế cơ bản của nó. Những đại lý như vậy được cho là có chủ ý. Chúng là phản đề của các đại lý phản ứng, vì chúng đại diện một cách rõ ràng cho một mô hình biểu tượng của thế giới và đưa ra quyết định thông qua suy luận logic dựa trên sự đối sánh mẫu và thao tác biểu tượng. Kiến thức của đại lý có chủ ý xác định hành vi của nó theo Nguyên tắc hợp lý của Newell, trong đó nêu rõ rằng:

Nếu một đại lý biết rằng một trong những hành động của họ sẽ dẫn đến một trong những mục tiêu của họ, thì đại lý sẽ chọn hành động đó.

Kiến trúc đại lý



Kiến trúc đại lý

Kiến trúc theo cấp độ tri thức

Một trong những biểu hiện quan trọng nhất của cách tiếp cận này được gọi là kiến trúc niềm tin - mong muốn - ý định (BDI). Ở đây, kiến thức về môi trường được coi là "niềm tin" và mục tiêu tổng thể là "mong muốn". Cùng với nhau, những điều này hình thành "ý định", tức là các phương án đã chọn mà hệ thống tự cam kết đạt được (Hình). Các ý định chỉ được thực hiện chừng nào chúng vẫn phù hợp với mong muốn và có thể đạt được theo niềm tin. Quá trình xác định phải làm gì, tức là mong muốn hoặc mục tiêu, là quá trình cân nhắc. Quá trình xác định cách thực hiện, tức là kế hoạch hoặc ý định, là phân tích phương tiện. Trong kiến trúc này, sự cân bằng giữa khả năng phản ứng và tính hướng đến mục tiêu có thể được lập lại thành một giữa việc xem xét lại các ý định thường xuyên (như một đại lý thận trọng có thể) và không thường xuyên (như một đại lý táo bạo hoặc ung dung có thể).

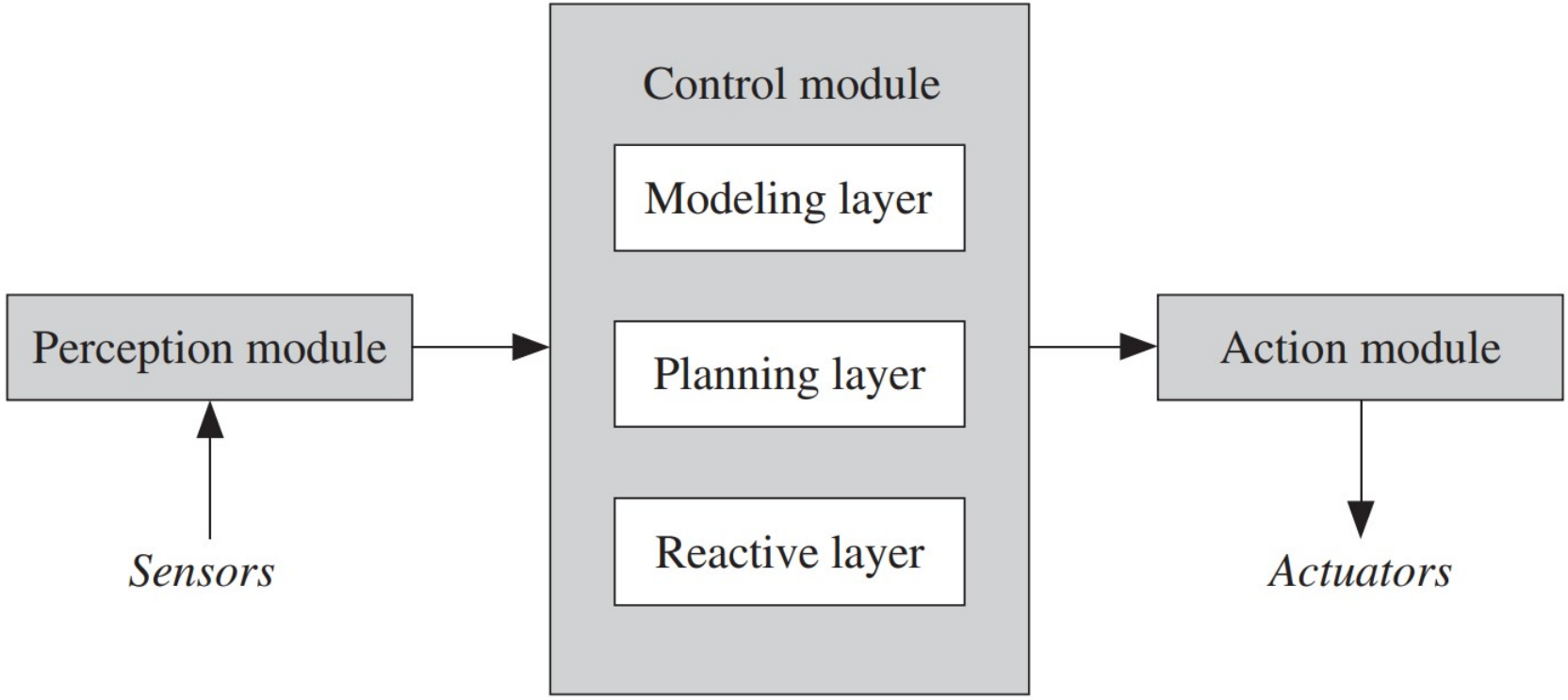
Kinny và Georgeff nhận thấy rằng cách tiếp cận thận trọng hoạt động tốt nhất trong môi trường thay đổi nhanh chóng và cách tiếp cận táo bạo hoạt động tốt nhất trong môi trường thay đổi chậm.

Kiến trúc đại lý

Kiến trúc theo lớp

Cách tiếp cận cuối cùng để cân bằng hành vi phản ứng và hướng đến mục tiêu là kết hợp các mô-đun áp dụng hai lập trường khác nhau. Đây là cơ sở của kiến trúc phân lớp. Trên thực tế, các đại lý này chứa ba lớp cụ thể: lớp phản ứng, lớp lập kế hoạch cho hành vi hướng đến mục tiêu và lớp mô hình hóa để mô hình hóa môi trường. Vấn đề cân bằng các lớp vẫn còn; trong trường hợp này, một hệ thống con điều khiển thông minh đảm bảo rằng mỗi lớp có một phần quyền lực thích hợp.

Hệ thống đa đại lý



Hệ thống đa đại lý

Về cơ bản đại lý thông minh riêng lẻ có thể thực hiện các chức năng hữu ích, để mở rộng khả năng này cần xem xét sự phối hợp cùng làm việc của nhiều nhóm đại lý thông minh. Trong Chương 1, cho rằng việc nghiên cứu trí tuệ nhân tạo được lấy cảm hứng từ những nỗ lực bắt chước suy luận logic của con người. Theo cách tương tự, một nhánh của AI được gọi là trí tuệ nhân tạo phân tán (DAI) đã được truyền cảm hứng từ những nỗ lực bắt chước một xã hội con người làm việc cùng nhau. Hệ thống đa đại lý (MAS), đôi khi được gọi là hệ thống hướng đại lý hoặc dựa trên đại lý, là một trong những cách tiếp cận quan trọng nhất đối với DAI; hệ thống bảng đen (xem Chương 9) là một hệ thống khác.

Hệ thống đa đại lý

Hệ thống đa đại lý có thể được định nghĩa là một hệ thống trong đó một số đại lý thông minh, tương tác theo đuổi một tập hợp các mục tiêu được tổ chức riêng lẻ hoặc thực hiện một tập hợp các nhiệm vụ riêng lẻ. Mặc dù định nghĩa này là một điểm khởi đầu hữu ích, nó đặt ra một số câu hỏi chính sẽ được giải quyết trong phần còn lại của chương này, đáng chú ý là:

- Những lợi ích là gì?
- Các đại lý thông minh tương tác như thế nào?
- Họ theo đuổi mục tiêu và thực hiện nhiệm vụ như thế nào?

Hệ thống đa đại lý

Lợi ích của hệ thống đa đại lý

Có một số vấn đề mà hệ thống đa đại lý đưa ra cách tiếp cận khả thi duy nhất:

Các vấn đề vốn có phức tạp: Những vấn đề như vậy đơn giản là quá lớn để có thể giải quyết bằng một hệ thống phần cứng hoặc phần mềm duy nhất. Do các đặc vụ được cung cấp thông tin tình báo để xử lý nhiều tình huống khác nhau, nên sẽ có một số không chắc chắn về cách chính xác hệ thống đại lý sẽ hoạt động như thế nào trong một tình huống cụ thể. Tuy nhiên, các đại lý được thiết kế tốt sẽ đảm bảo rằng Thiết bị truyền động cảm biến Lớp mô hình hóa Lớp kế hoạch Lớp phản ứng Mô-đun điều khiển Mô-đun cảm nhận Mô-đun hành động Hình Tham quan Kiến trúc máy mọi tình huống đều được xử lý theo cách thích hợp mặc dù có thể không được dự đoán rõ ràng.

Hệ thống đa đại lý

Lợi ích của hệ thống đa đại lý

Vấn đề phân tán cố hữu: Ở đây dữ liệu và thông tin có thể tồn tại ở các vị trí vật lý khác nhau, hoặc tại các thời điểm khác nhau, hoặc có thể được nhóm lại thành các nhóm yêu cầu các phương pháp xử lý hoặc ngữ nghĩa khác nhau. Những loại vấn đề này yêu cầu một giải pháp phân tán, có thể được cung cấp bởi các đại lý chạy đồng thời, mỗi đại lý có luồng điều khiển riêng.

Hệ thống đa đại lý

Lợi ích của hệ thống đa đại lý

Một hệ thống đa đại lý (MAS) mang lại những lợi ích chung sau:

- Một cái nhìn tự nhiên hơn về trí thông minh.
- Tăng tốc độ và hiệu quả, do các đại lý có thể hoạt động đồng thời và giao tiếp không đồng bộ.
- Mạnh mẽ và đáng tin cậy: Không có đại lý nào là quan trọng miễn là có những đại lý khác có thể đảm nhận vai trò của nó trong trường hợp nó thất bại. Do đó, hiệu suất của MAS sẽ giảm sút một cách đáng kể nếu các đại lý riêng lẻ bị lỗi.
- Khả năng mở rộng: Hệ thống DAI thường có thể được mở rộng quy mô chỉ đơn giản bằng cách thêm các thành phần bổ sung. Trong trường hợp MAS, các đại lý bổ sung có thể được thêm vào mà không ảnh hưởng xấu đến những đại lý đã có.

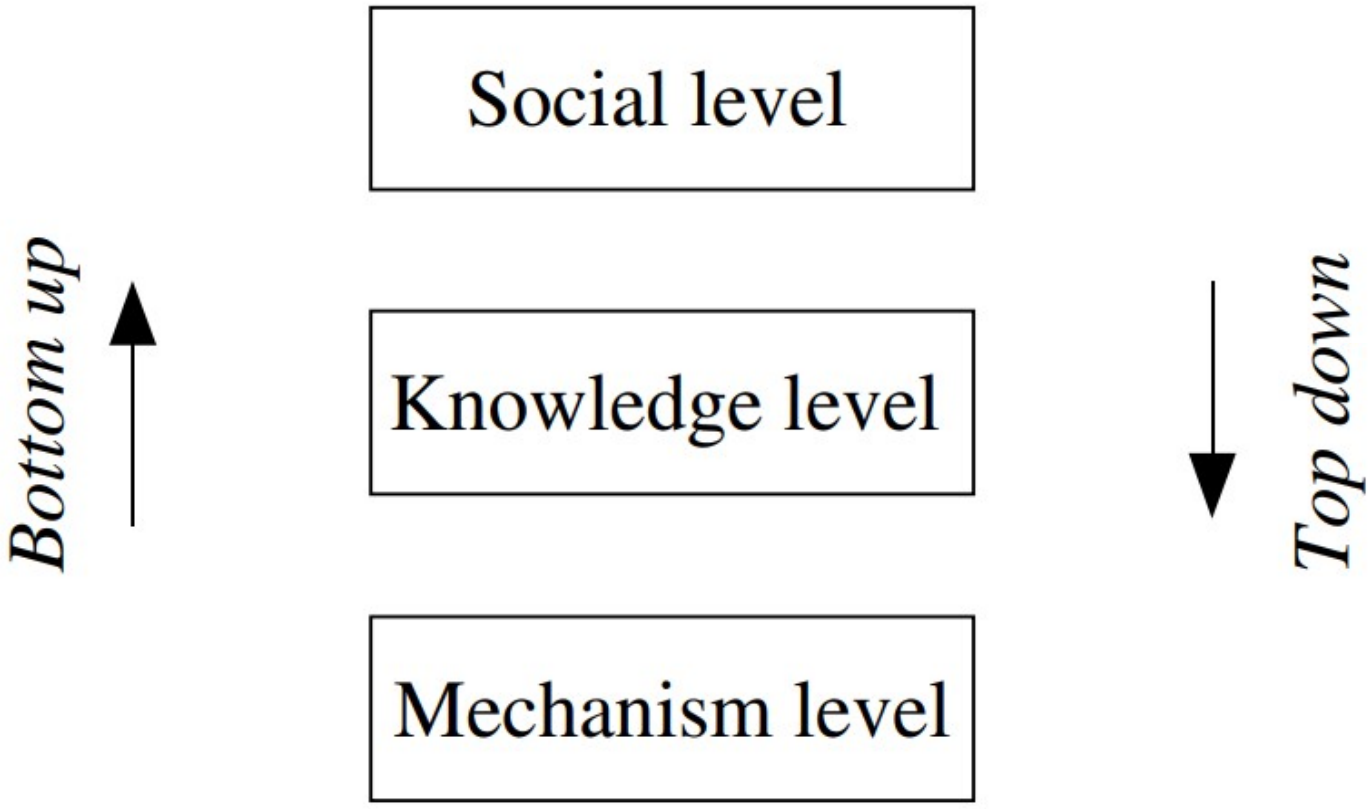
Hệ thống đa đại lý

Lợi ích của hệ thống đa đại lý

- Tính chi tiết: Các đại lý có thể được thiết kế để hoạt động ở mức độ chi tiết thích hợp. Nhiều đại lý "chi tiết" có thể được yêu cầu làm việc trên các chi tiết nhỏ của một vấn đề, trong đó mỗi đại lý giải quyết một chi tiết riêng biệt. Đồng thời, một số đặc vụ “thô sơ”, tinh vi hơn có thể tập trung vào chiến lược cấp cao hơn.
- Dễ phát triển: Như với OOP, tính đóng gói cho phép các đại lý riêng lẻ được phát triển riêng biệt và được sử dụng lại ở bất cứ nơi nào có thể áp dụng.
- Chi phí: Một hệ thống bao gồm nhiều đại lý xử lý nhỏ có khả năng rẻ hơn một hệ thống tập trung lớn.

Jennings đã lập luận rằng MAS, một mặt, phù hợp với việc thiết kế và xây dựng các hệ thống phần mềm phân tán, phức tạp và mặt khác, phù hợp với tư cách là một mô hình kỹ thuật phần mềm chính thống.

Hệ thống đa đại lý



Hệ thống đa đại lý

Xây dựng hệ thống đa đại lý

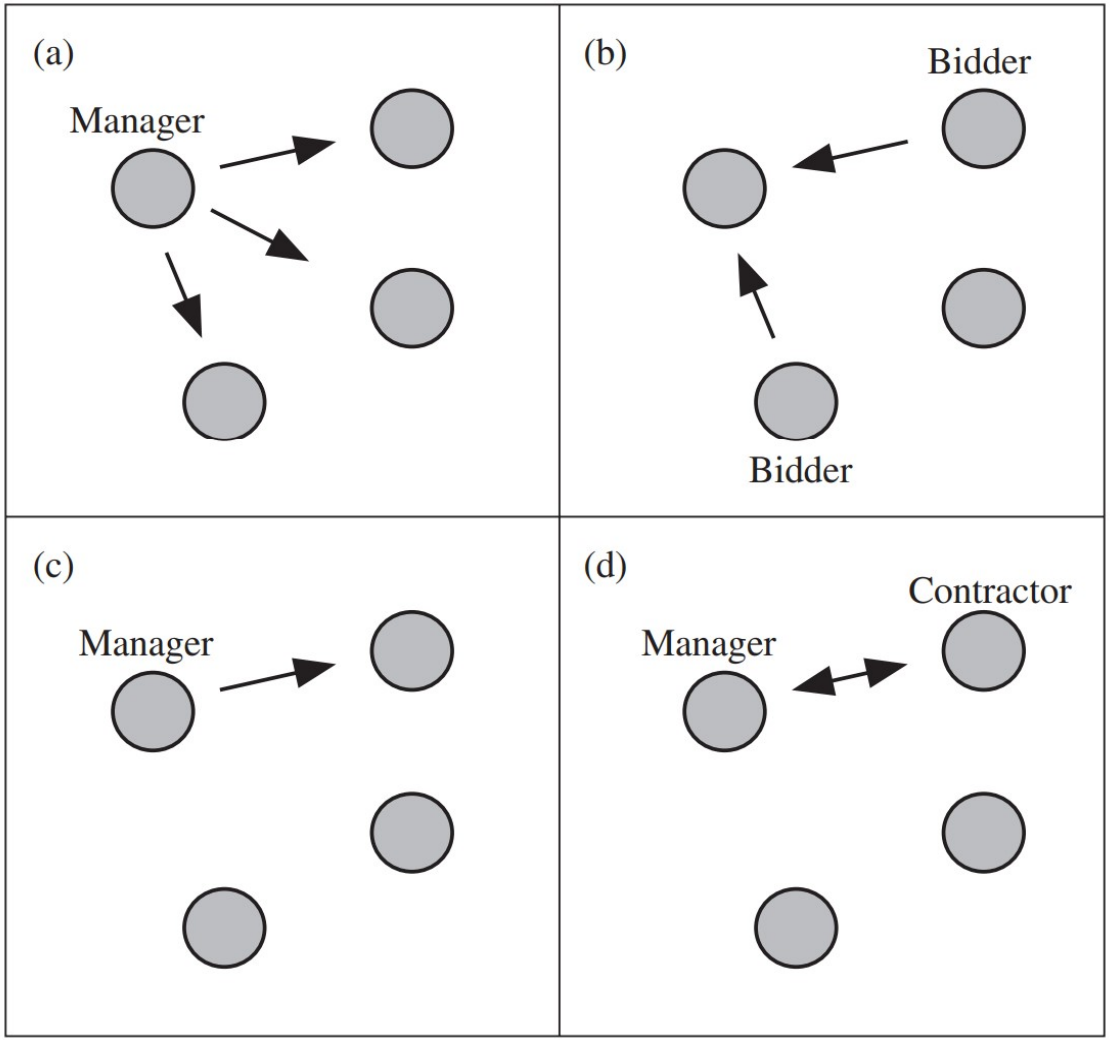
Một hệ thống đa đại lý phụ thuộc vào sự tương tác giữa các đại lý thông minh. Do đó, có một số quyết định thiết kế chính cần được thực hiện, ví dụ, khi nào, như thế nào và các đại lý nên tương tác với ai? Trong các mô hình hợp tác, một số đại lý cố gắng kết hợp nỗ lực của họ để hoàn thành như một nhóm những gì mà các cá nhân không thể. Trong các mô hình cạnh tranh, mỗi đại lý cố gắng đạt được thứ mà chỉ một số đại lý có thể có. Trong cả hai loại mô hình, các đại lý thường được cho là trung thực.

Hệ thống đa đại lý

Xây dựng hệ thống đa đại lý

Để đạt được sự đồng nhất, các hệ thống đa đại lý có thể được thiết kế từ dưới lên hoặc từ trên xuống. Theo cách tiếp cận từ dưới lên, các đại lý được ban tặng với đầy đủ các khả năng, bao gồm cả các giao thức truyền thông, để cho phép họ tương tác một cách hiệu quả. Hiệu suất tổng thể của hệ thống sau đó xuất hiện từ những tương tác này. Theo cách tiếp cận từ trên xuống, các quy ước - đôi khi được gọi là chuẩn mực xã hội - được áp dụng ở cấp độ nhóm để xác định cách các đại lý nên tương tác. Một ví dụ có thể là nguyên tắc dân chủ, đạt được bằng cách trao cho các đại lý quyền bầu cử. Nếu chúng ta xem một đại lý có mức kiến thức được trừu tượng hóa bên trên các cơ chế bên trong của nó, thì những quy ước này có thể được coi là nằm ở mức trừu tượng vẫn cao hơn, đó là mức xã hội.

Hệ thống đa đại lý



Hệ thống đa đại lý

Xây dựng hệ thống đa đại lý

Hệ thống đa nhân tố thường được thiết kế dưới dạng mô hình máy tính về các vai trò chức năng của con người. Ví dụ, chúng ta có thể có một cấu trúc kiểm soát phân cấp trong đó một đại lý là cấp trên của các đại lý cấp dưới khác. Mối quan hệ nhóm ngang hàng, chẳng hạn như có thể tồn tại trong một tổ chức dựa trên nhóm, cũng có thể. Phần này sẽ đề cập đến ba mô hình quản lý tương tác đại lý, được gọi là lưới hợp đồng/hợp tác, giải quyết vấn đề hợp tác (CPS) và quản lý ma trận chuyển dịch (SMM). Sau khi xem xét lần lượt từng mô hình này, ngữ nghĩa của giao tiếp giữa các đại lý sẽ được giải quyết.

Hệ thống đa đại lý

Xây dựng hệ thống đa đại lý – lưới hợp tác

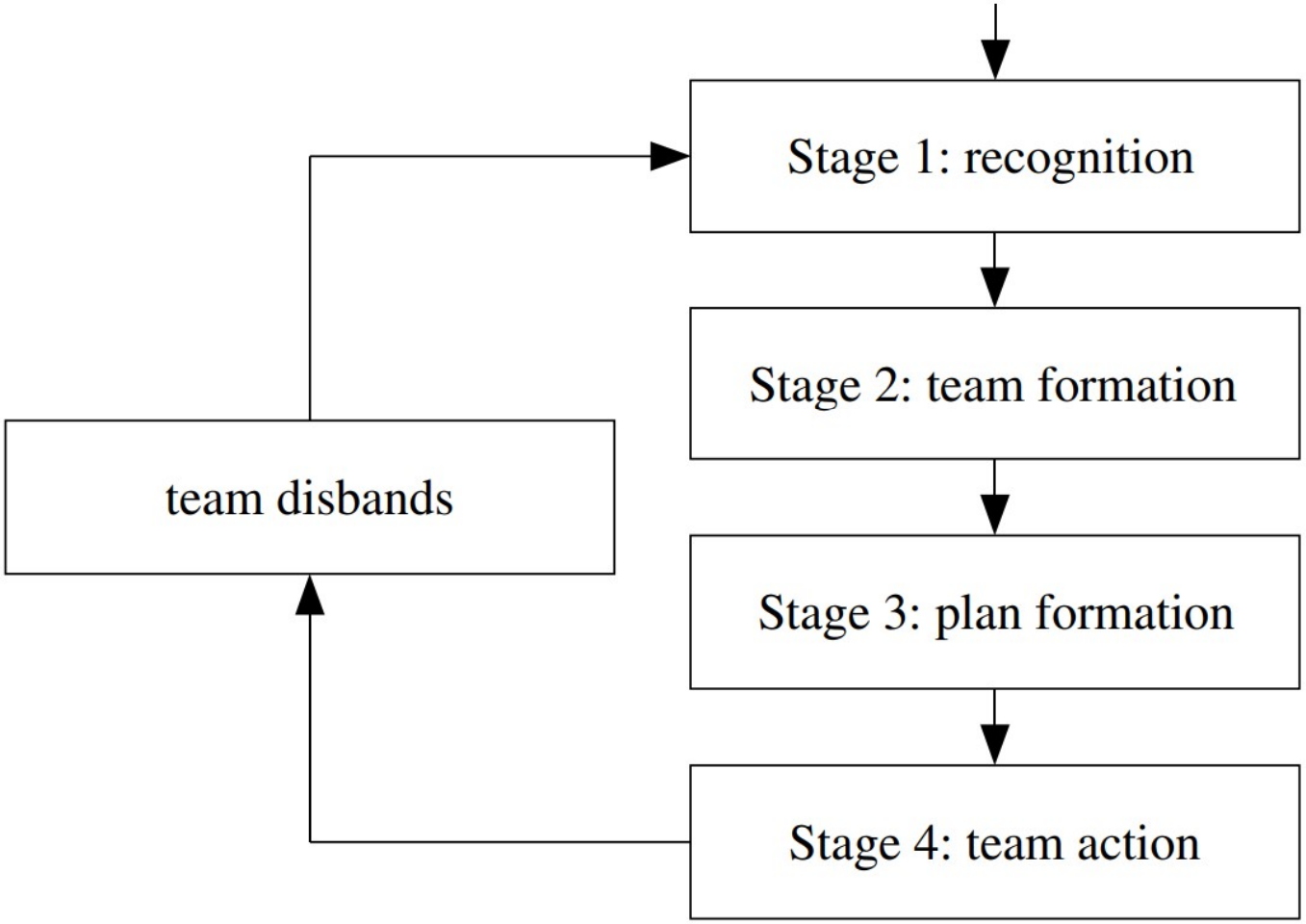
Hãy tưởng tượng rằng bạn đã quyết định xây dựng một ngôi nhà của riêng mình. Bạn chưa chắc đã tự mình đảm nhận mọi công việc. Bạn có thể sẽ thuê các chuyên gia để vẽ các quy hoạch kiến trúc, xin phép quy hoạch theo luật định, đặt nền móng, xây tường, lắp đặt sàn, xây dựng mái nhà và kết nối các tiện ích khác nhau. Mỗi chuyên gia này có thể lần lượt sử dụng một nhà thầu phụ cho một số khía cạnh của công việc. Sự sắp xếp này tương tự như khung ròng hợp đồng về hợp tác đại lý. Tại đây, đại lý quản lý tạo ra các nhiệm vụ và chịu trách nhiệm giám sát việc thực hiện của chúng. Người quản lý ký kết các thỏa thuận rõ ràng với các đại lý nhà thầu sẵn sàng thực hiện các nhiệm vụ. Các đại lý riêng lẻ không được chỉ định trước làm người quản lý hoặc nhà thầu. Đây chỉ là những vai trò và bất kỳ đại lý nào cũng có thể đảm nhận một trong hai vai trò này trong quá trình giải quyết vấn đề.

Hệ thống đa đại lý

***Xây dựng hệ thống đa đại lý* – lưới hợp tác**

Để thiết lập một hợp đồng, người quản lý đại lý quảng cáo sự tồn tại của các nhiệm vụ cho các đại lý khác. Các đại lý là nhà thầu tiềm năng sẽ đánh giá các thông báo nhiệm vụ và nộp hồ sơ dự thầu cho những công việc mà họ phù hợp. Người quản lý đánh giá các hồ sơ dự thầu và trao các hợp đồng thực hiện nhiệm vụ cho các đại lý mà họ xác định là phù hợp nhất. Do đó, người quản lý và nhà thầu được liên kết bởi một hợp đồng và giao tiếp riêng tư trong khi hợp đồng đang được thực hiện. Người quản lý cung cấp hầu hết thông tin nhiệm vụ và nhà thầu báo cáo tiến độ và kết quả cuối cùng của nhiệm vụ. Quá trình thương lượng có thể lặp lại nếu một nhà thầu chia nhỏ nhiệm vụ của mình và trao hợp đồng cho các đại lý khác mà họ là người quản lý.

Hệ thống đa đại lý



Hệ thống đa đại lý

Xây dựng hệ thống đa đại lý – Khung CPS

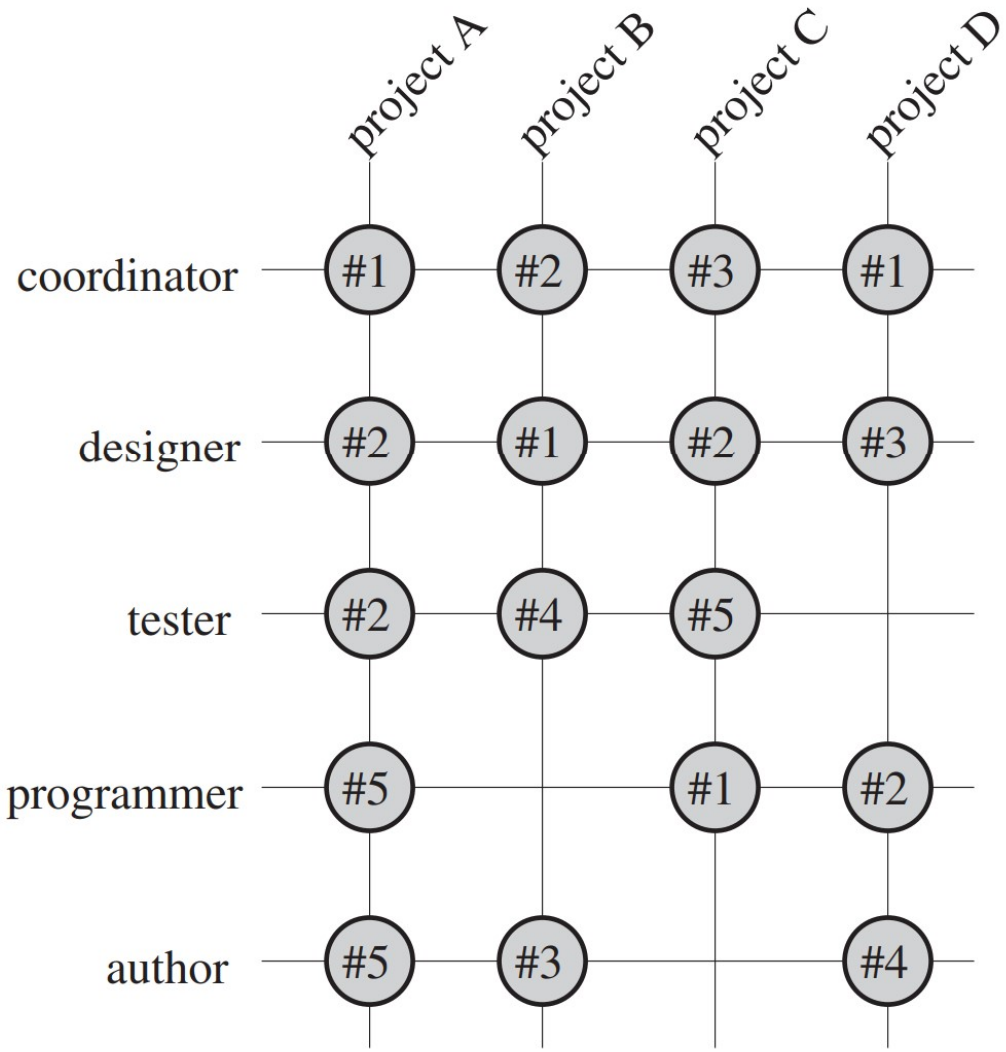
Khung hợp tác giải quyết vấn đề (CPS) là một mô hình từ trên xuống để hợp tác đại lý. Như trong mô hình BDI, ý định của đại lý đóng một vai trò quan trọng. Họ xác định hành vi cá nhân của đại lý bất cứ lúc nào, trong khi ý định chung kiểm soát hành vi xã hội của họ. Ý định của một đại lý được hình thành bởi cam kết của họ và ý định chung của họ bởi quy ước xã hội của họ. Khung bao gồm bốn giai đoạn sau, cũng được thể hiện trong Hình trên.

Hệ thống đa đại lý

Xây dựng hệ thống đa đại lý – Khung CPS

Giai đoạn 1: công nhận. Một số đại lý nhận ra tiềm năng hợp tác với một đại lý đang tìm kiếm sự hỗ trợ, có thể vì họ có một mục tiêu mà họ không thể đạt được một cách cô lập.
Giai đoạn 2: thành lập đội. Một đại lý nhận ra tiềm năng hợp tác ở Giai đoạn 1 sẽ đề nghị hỗ trợ thêm. Nếu thành công, giai đoạn này kết thúc với việc một nhóm có cam kết chung về hành động tập thể.
Giai đoạn 3: hình thành kế hoạch. Các đại lý cố gắng thương lượng một kế hoạch chung mà họ tin rằng sẽ đạt được mục tiêu mong muốn.
Giai đoạn 4: hành động theo nhóm. Kế hoạch hành động chung mới được thống nhất được thực hiện. Bằng cách tuân thủ một quy ước xã hội đã được thống nhất, các đại lý duy trì một mối quan hệ chặt chẽ xuyên suốt.

Hệ thống đa đại lý

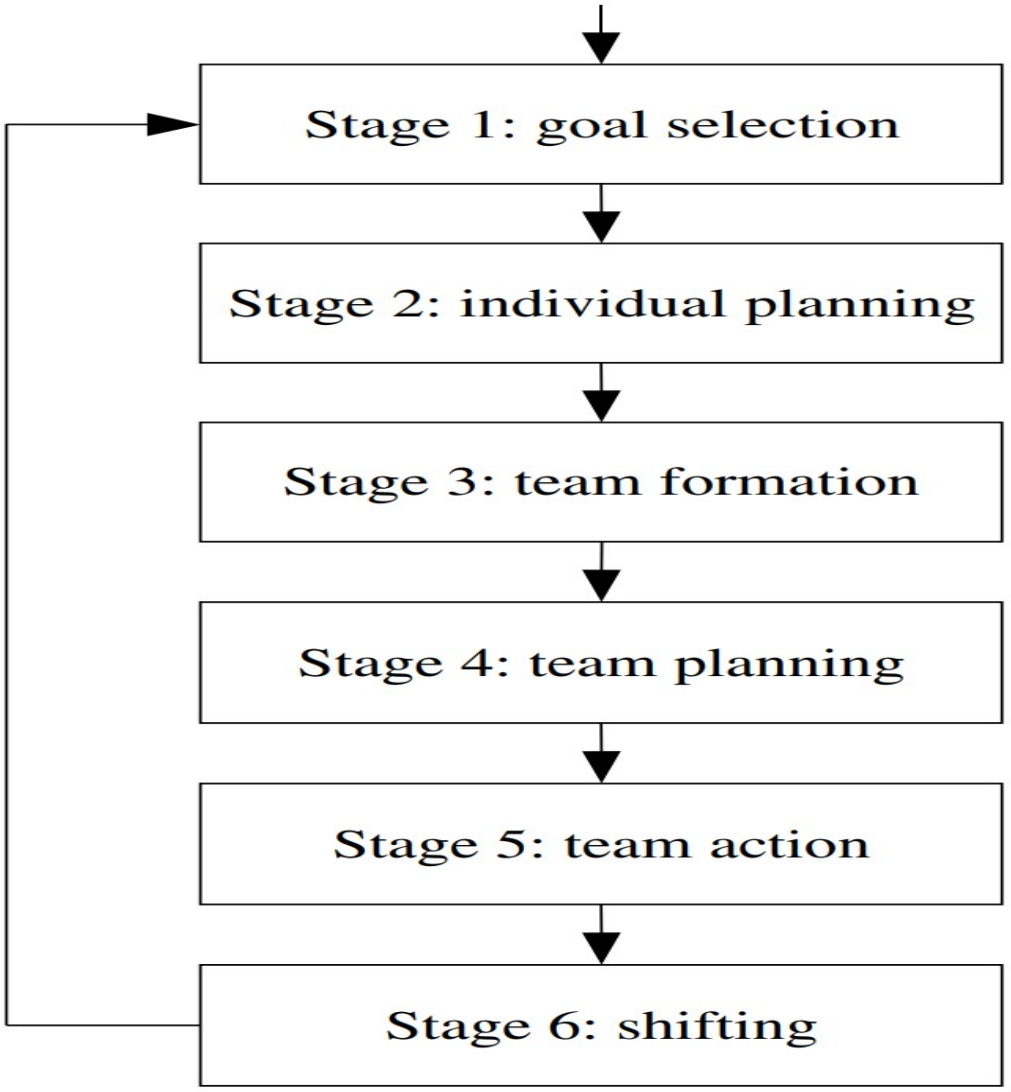


Hệ thống đa đại lý

Xây dựng hệ thống đa đại lý – Quản lý ma trận dịch chuyển (SMM)

SMM là một mô hình điều phối đại lý được lấy cảm hứng từ mô hình quản lý cơ cấu tổ chức của Mintzberg's Shifting Matrix, như được minh họa trong Hình. Không giống như hệ thống phân cấp quản lý truyền thống, quản lý ma trận cho phép nhiều dòng quyền hạn, phản ánh nhiều chức năng được mong đợi của một lực lượng lao động linh hoạt. Quản lý ma trận thay đổi đưa ý tưởng này lên một bước xa hơn bằng cách coi các dòng quyền hạn là tạm thời, thường thay đổi khi các dự án khác nhau bắt đầu và kết thúc. Ví dụ, trong Hình, cá nhân số 1 là điều phối viên cho dự án A và nhà thiết kế cho dự án B. Để áp dụng những ý tưởng này vào hợp tác đại lý, một khuôn khổ gồm sáu giai đoạn đã được đưa ra và được phác thảo bên dưới. Các đại lý được phân biệt bởi động cơ, chức năng và kiến thức khác nhau của họ. Những khác biệt này xác định sự đa dạng của các trạng thái tinh thần đối với mục tiêu, niềm tin và ý định.

Hệ thống đa đại lý



Hệ thống đa đại lý

***Xây dựng hệ thống đa đại lý* – Quản lý ma trận dịch chuyển (SMM)**

Giai đoạn 1: lựa chọn mục tiêu. Sự kiện chọn các nhiệm vụ họ muốn thực hiện, dựa trên trạng thái tinh thần ban đầu của họ.
Giai đoạn 2: lập kế hoạch cá nhân. Các đại lý lựa chọn một cách để đạt được mục tiêu của họ. Đặc biệt, một đại lý nhận ra mục tiêu đã định của mình là phổ biến đối với các đại lý khác sẽ phải quyết định theo đuổi mục tiêu một cách riêng lẻ hay phối hợp với các đại lý khác.
Giai đoạn 3: thành lập đội. Các đại lý đang tìm kiếm sự hợp tác cố gắng tự tổ chức thành một nhóm. Việc thành lập một nhóm đòi hỏi phải có một quy tắc ứng xử đã được thống nhất, một cơ sở để chia sẻ các nguồn lực và một thước đo chung về hiệu suất.
Giai đoạn 4: lập kế hoạch cho nhóm. Khối lượng công việc được phân bổ giữa các thành viên trong nhóm.
Giai đoạn 5: hành động theo nhóm. Kế hoạch của nhóm được các thành viên thực hiện theo quy tắc ứng xử của nhóm.

Hệ thống đa đại lý

Xây dựng hệ thống đa đại lý – Quản lý ma trận dịch chuyển (SMM)

Giai đoạn 6: chuyển dịch. Giai đoạn cuối cùng của quá trình hợp tác, đánh dấu sự tan rã của nhóm, liên quan đến việc thay đổi mục tiêu, vị trí và vai trò của các đại lý.

Mỗi đại lý cập nhật xác suất làm việc nhóm của mình với các đại lý khác, tùy thuộc vào việc trải nghiệm làm việc nhóm hoàn chỉnh với đại lý đó có thành công hay không. Kiến thức cập nhật này rất quan trọng, vì việc lặp lại qua sáu giai đoạn diễn ra cho đến khi hoàn thành tất cả các nhiệm vụ.

Hệ thống đa đại lý

Giao tiếp giữa các đại lý

Các đại lý thông minh có thể được thiết kế và triển khai cũng như có thể hợp tác và thương lượng trong một xã hội. Hãy xem xét cách các đại lý giao tiếp với nhau, cả đồng bộ và không đồng bộ. Giao tiếp đồng bộ giống như một cuộc trò chuyện - sau khi gửi một tin nhắn, đại lý gửi sẽ đợi phản hồi từ người nhận. Giao tiếp không đồng bộ giống như gửi một email hoặc một bức thư - mặc dù bạn có thể mong đợi một hồi âm vào một thời điểm nào đó trong tương lai, nhưng bạn không mong đợi người nhận sẽ đọc hoặc hành động ngay lập tức.

Các đại lý có thể được thực hiện bởi những người khác nhau vào những thời điểm khác nhau trên các máy tính khác nhau, nhưng vẫn phải giao tiếp với nhau. Do đó, đã có một động lực để chuẩn hóa cấu trúc thông báo giữa các đại lý, bất kể miền mà chúng đang hoạt động.

Hệ thống đa đại lý

Giao tiếp giữa các đại lý

Trong KQML, cấu trúc chứa ít nhất các thành phần sau:
Một công năng. Đây là một từ đơn mô tả mục đích của tin nhắn, ví dụ: cho biết, hủy bỏ, đánh giá, quảng cáo, hỏi một người, đăng ký, trả lời.

- Danh tính của đại lý là người gửi.
- Danh tính của đại lý là người nhận.
- Ngôn ngữ được sử dụng trong nội dung tin nhắn. Mặc dù KQML xác định dạng tổng thể của thông báo, nhưng bất kỳ ngôn ngữ lập trình nào cũng có thể được sử dụng cho nội dung miền cụ thể.

Hệ thống đa đại lý

Giao tiếp giữa các đại lý

- Bản thể luận, hoặc từ vựng, của thông điệp. Điều này cung cấp bối cảnh mà trong đó nội dung thư sẽ được diễn giải. Ví dụ, vấn đề chọn một polyme để đáp ứng yêu cầu thiết kế kỹ thuật được sử dụng trong Chương 10 để chứng minh các ngôn ngữ Lisp và Prolog. Ở cấp độ ngôn ngữ lập trình, một chương trình để giải quyết vấn đề này chỉ đơn thuần là một tập hợp các từ và ký hiệu được tổ chức dưới dạng các câu lệnh. Ví dụ, nó sẽ vẫn đúng về mặt cú pháp nếu mỗi tên polyme được thay thế bằng tên của một loại trái cây riêng biệt. Các câu lệnh chỉ trở nên có nghĩa khi chúng được diễn giải bằng từ vựng của các polyme kỹ thuật.
- Nội dung tin nhắn. Trong thế giới lựa chọn polyme được đề cập ở trên, agent1 có thể muốn nói với agent2 về các đặc tính của polystyrene, được mã hóa trong Prolog. Sử dụng KQML, nó có thể làm như vậy với thông báo sau.

Hệ thống đa đại lý

```
(tell
  :sender    agent1
  :receiver  agent2
  :language  prolog
  :ontology  polymer-world
  :content
    "materials_database(polystyrene, thermoplastic,
      [[impact_resistance, 0.02],
       [flexural_modulus, 3.0],
       [maximum_temperature, 50]]).")
```

Chương 6

Mạng nơ ron (Neural networks)

Giới thiệu

Mạng nơron nhân tạo là một họ các kỹ thuật cho **học số**, giống như các thuật toán tối ưu hóa, nhưng trái ngược với các kỹ thuật học ký hiệu. Chúng bao gồm nhiều **phần tử tính toán phi tuyến** tạo thành các **nút mạng hoặc nơron**, được **liên kết** bằng các **kết nối có trọng số**. Chúng có cấu trúc tương tự như hệ thống thần kinh ở động vật, được tạo thành từ các mạng thần kinh thực chứ không phải là nhân tạo. Các mạng nơron nhân tạo thực tế đơn giản hơn nhiều so với các mạng sinh học, vì vậy **sẽ không thể kỳ vọng chúng tạo ra các hành vi tinh vi như của con người hoặc động vật**. Tuy nhiên, chúng có thể thực hiện một số nhiệm vụ nhất định, đặc biệt là **phân loại** theo cách hiệu quả nhất. Trong môn học sẽ sử dụng mạng nơ-ron biểu thức có nghĩa là một mạng nơ-ron nhân tạo. Kỹ thuật sử dụng mạng nơ-ron được mô tả như là cơ chế kết nối.

Giới thiệu

Mỗi nút trong mạng nơ-ron có thể có một số **đầu vào**, mỗi đầu vào có trọng số liên quan. Nút thực hiện một phép tính đơn giản trên các giá trị đầu vào của nó, là các số nguyên đơn lẻ hoặc số thực, để tạo ra một giá trị số duy nhất làm **đầu ra** của nó. Đầu ra từ một nút có thể tạo thành đầu vào cho các nút khác hoặc là một phần của đầu ra từ toàn bộ mạng. Hiệu ứng tổng thể là mạng nơ-ron tạo ra một mẫu số ở các đầu ra của nó để đáp ứng với một mẫu số ở các đầu vào của nó. Các mẫu số này là mảng một chiều được gọi là vectơ,
ví dụ: (0,1, 1,0, 0,2).

Giới thiệu

Mỗi **tế bào/nút** thần kinh thực hiện tính toán của nó một cách độc lập với các tế bào thần kinh khác, ngoại trừ đầu ra từ một số tế bào thần kinh có thể tạo thành đầu vào cho những tế bào thần kinh khác. Do đó, **mạng nơ-ron có cấu trúc song song cao**, cho phép chúng khám phá nhiều giả thuyết cạnh tranh đồng thời. Tính song song này cho phép mạng nơ-ron tận dụng lợi thế của các **máy tính xử lý song song**. Chúng cũng có thể chạy trên các máy tính tuần tự thông thường – tuy nhiên sẽ mất nhiều thời gian hơn. Mạng nơ-ron có khả năng chịu đựng sự thất bại của các nơ-ron riêng lẻ hoặc kết nối với nhau. Hiệu suất của mạng sẽ giảm sút nhanh chóng nếu những lỗi cục bộ xảy ra trong mạng.

Giới thiệu

Trọng số trên các kết nối nút, cùng với cấu trúc liên kết tổng thể, xác định vectơ đầu ra được mạng bắt nguồn từ một vectơ đầu vào nhất định. Các trọng số không cần biết trước, nhưng có thể học được bằng cách tự động điều chỉnh chúng thông qua thuật toán huấn luyện. Trong trường hợp **học có giám sát**, các trọng số được suy ra bằng cách liên tục trình bày cho mạng một tập các vectơ đầu vào mẫu/ví dụ cùng với vectơ đầu ra mong muốn tương ứng cho mỗi chúng. Các trọng số được điều chỉnh với mỗi lần lặp cho đến khi đầu ra thực tế cho mỗi đầu vào gần với vectơ mong muốn. Trong trường hợp **học không giám sát**, các ví dụ được trình bày mà không có bất kỳ vectơ đầu ra mong muốn tương ứng nào. Với một thuật toán huấn luyện phù hợp, mạng sẽ điều chỉnh trọng số của nó phù hợp với các mẫu xuất hiện tự nhiên trong dữ liệu. Sau đó, vectơ đầu ra đại diện cho vị trí của vectơ đầu vào trong các mẫu dữ liệu được phát hiện.

Giới thiệu

Một phần của sự hấp dẫn của mạng nơ-ron là khi được hiển thị với dữ liệu nhiễu hoặc không đầy đủ, chúng sẽ tạo ra một giải đáp gần đúng hơn là một giải đáp không chính xác. Tương tự, khi được hiển thị với dữ liệu không quen thuộc nằm trong phạm vi của các ví dụ đã thấy trước đây của nó, mạng nói chung sẽ tạo ra một đầu ra là phép nội suy hợp lý giữa các đầu ra ví dụ. Tuy nhiên, mạng nơ-ron không thể ngoại suy một cách đáng tin cậy ngoài phạm vi của các ví dụ đã thấy trước đây. Nội suy cũng có thể đạt được bằng logic mờ (xem Chương 3). Do đó, mạng nơron và logic mờ thường đại diện cho các giải pháp thay thế cho một vấn đề kỹ thuật cụ thể và có thể được kết hợp trong một hệ thống lai (xem Chương 9).

Ứng dụng của mạng nơ ron

Mạng nơ-ron có thể được áp dụng cho sự đa dạng của các nhiệm vụ. Nói chung, mạng liên kết một vectơ đầu vào nhất định (x1, x2,… xn) với một vectơ đầu ra cụ thể (y1, y2,… ym), mặc dù hàm liên kết có thể không xác định và có thể rất phi tuyến tính. (Một hàm tuyến tính là một hàm có thể được biểu diễn dưới dạng f (x) = mx + c, trong đó m và c là các hằng số; một hàm phi tuyến tính có thể bao gồm các số hạng bậc cao hơn cho x, hoặc các hàm lượng giác hoặc logarit của x.)

Ứng dụng của mạng nơ ron

Ước tính phi tuyến

Mạng nơron cung cấp một kỹ thuật hữu ích để xác định giá trị của các biến không thể đo lường dễ dàng, nhưng được biết là phụ thuộc theo một số cách phức tạp vào các biến khác dễ tiếp cận hơn. Các biến có thể đo lường tạo thành vectơ đầu vào của mạng và các biến chưa xác định tạo thành vectơ đầu ra-được gọi là sử dụng ước lượng phi tuyến. Mạng được huấn luyện ban đầu bằng cách sử dụng một tập hợp các ví dụ được gọi là dữ liệu huấn luyện. Học có giám sát được sử dụng, vì vậy mỗi ví dụ trong khóa đào tạo bao gồm hai vectơ: vectơ đầu vào và vectơ đầu ra mong muốn tương ứng của nó. (Điều này giả định rằng một số giá trị cho biến ít truy cập hơn đã được lấy để tạo ra các kết quả đầu ra mong muốn.) Trong quá trình đào tạo, mạng học cách liên kết các vectơ đầu vào ví dụ với các vectơ đầu ra mong muốn của chúng. Khi nó được trình bày sau đó với một vectơ đầu vào chưa từng thấy trước đó, mạng có thể nội suy giữa các ví dụ tương tự trong dữ liệu huấn luyện để tạo ra một vectơ đầu ra.

Ứng dụng của mạng nơ ron

Phân loại

Thường thì vectơ đầu ra từ mạng nơ-ron được sử dụng để đại diện cho một trong một tập hợp các kết quả có thể có, tức là mạng hoạt động như một bộ phân loại. Ví dụ, một hệ thống nhận dạng giọng nói có thể được tạo ra để nhận ra ba từ khác nhau: có, không và có thể. Âm thanh được số hóa của các từ sẽ được xử lý trước theo một số cách để tạo thành vectơ đầu vào. Vectơ đầu ra mong muốn sau đó sẽ là (0, 0, 1), (0, 1, 0) hoặc (1, 0, 0), đại diện cho ba lớp từ.

Một mạng như vậy sẽ được đào tạo bằng cách sử dụng một tập hợp các ví dụ được gọi là dữ liệu đào tạo. Mỗi ví dụ sẽ bao gồm một câu nói được số hóa của một trong các từ làm vectơ đầu vào, sử dụng một loạt các giọng nói khác nhau, cùng với vectơ đầu ra mong muốn tương ứng. Trong quá trình đào tạo, mạng học cách liên kết các vectơ đầu vào tương tự với một vectơ đầu ra cụ thể. Khi nó được hiển thị sau đó với một vectơ đầu vào chưa từng thấy trước đó, mạng sẽ chọn vectơ đầu ra cung cấp kết quả phù hợp nhất. Kiểu phân loại này sẽ không đơn giản nếu sử dụng các kỹ thuật không liên kết, vì dữ liệu đầu vào hiếm khi tương ứng chính xác với bất kỳ ví dụ nào trong dữ liệu huấn luyện.

Ứng dụng của mạng nơ ron

Phân cụm

Phân cụm là một hình thức học tập không giám sát, tức là, dữ liệu đào tạo bao gồm một tập hợp các vectơ đầu vào mẫu mà không có bất kỳ vectơ đầu ra mong muốn tương ứng nào. Khi các vectơ đầu vào liên tiếp được trình bày, chúng được nhóm lại thành N nhóm, trong đó số nguyên N có thể được xác định trước hoặc có thể được phép tăng theo sự đa dạng của dữ liệu. Ví dụ, các từ nói được xử lý trước được số hóa có thể được trình bày trên mạng. Mạng sẽ học cách tập hợp các ví dụ mà nó cho là tương tự với nhau theo một nghĩa nào đó. Trong ví dụ này, các cụm có thể tương ứng với các từ khác nhau hoặc các giọng nói khác nhau.

Khi các cụm đã hình thành, một mạng nơ-ron thứ hai có thể được huấn luyện để liên kết mỗi cụm với một đầu ra mong muốn cụ thể. Hệ thống tổng thể sau đó trở thành một bộ phân loại, trong đó mạng đầu tiên không được giám sát và mạng thứ hai được giám sát. Phân cụm hữu ích cho việc nén dữ liệu và là một khía cạnh quan trọng của khai thác dữ liệu, tức là tìm kiếm các mẫu trong dữ liệu phức tạp.

Ứng dụng của mạng nơ ron

Bộ nhớ định địa chỉ nội dung

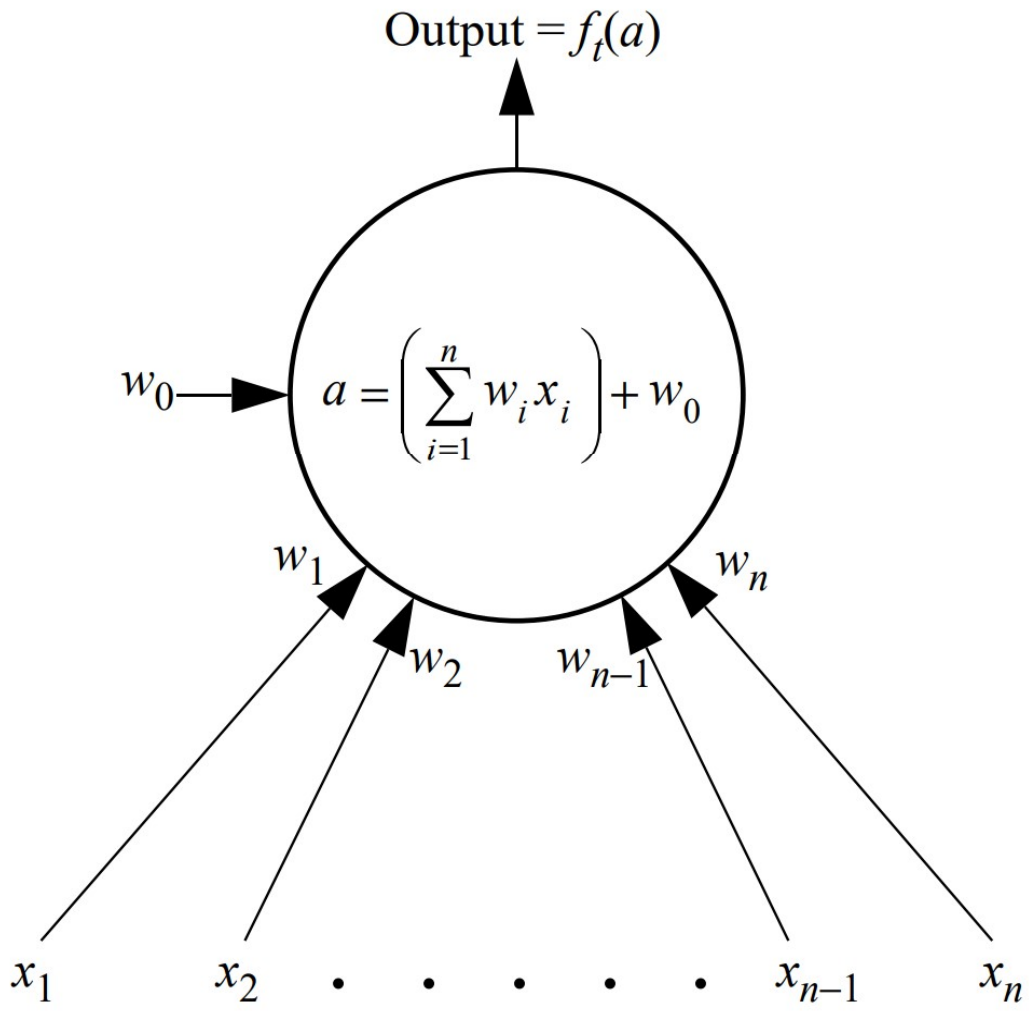
Việc sử dụng mạng nơ-ron làm bộ nhớ địa chỉ nội dung là một hình thức học tập không giám sát khác, do đó, không có vectơ đầu ra mong muốn nào được liên kết với dữ liệu huấn luyện. Trong quá trình đào tạo, mỗi vectơ đầu vào ví dụ sẽ được lưu trữ ở dạng phân tán thông qua mạng.

Khi một vectơ chưa từng thấy trước đó được hiển thị trên mạng, nó được coi là một phiên bản chưa hoàn chỉnh hoặc có lỗi của một trong các ví dụ được lưu trữ. Vì vậy, mạng sẽ tạo lại ví dụ được lưu trữ gần giống nhất với vectơ đã trình bày. Đây có thể được coi là một kiểu phân loại, trong đó mỗi ví dụ trong dữ liệu huấn luyện thuộc về một lớp riêng biệt và mỗi ví dụ đại diện cho vectơ lý tưởng cho lớp đó. Nó rất hữu ích khi các lớp có thể được đặc trưng bởi một ví dụ lý tưởng hoặc hoàn hảo. Ví dụ: văn bản in sau đó được quét để tạo thành hình ảnh số hóa sẽ chứa các ví dụ về ký tự in bị nhiễu và không hoàn hảo. Đối với một phông chữ nhất định, một phiên bản lý tưởng của mỗi ký tự có thể được lưu trữ trong bộ nhớ địa chỉ nội dung và được tạo ra dưới dạng đầu ra của nó bất cứ khi nào một phiên bản không hoàn hảo được trình bày làm đầu vào của nó.

Nút và kết nối

Mỗi nút, hay nơ-ron, trong mạng nơ-ron là một phần tử tính toán đơn giản có một phía đầu vào và một phía đầu ra. Mỗi nút có thể có các kết nối định hướng đến nhiều nút khác ở cả hai phía đầu vào và đầu ra của nó. Mỗi đầu vào Xi được nhân với trọng số Wi liên quan của nó. Thông thường, vai trò của nút là tổng hợp từng đầu vào có trọng số của nó và thêm một số hạng ưu tiên W0 để tạo thành một đại lượng trung gian được gọi là kích hoạt, a. Sau đó, nó chuyển kích hoạt thông qua một hàm phi tuyến được gọi là hàm truyền hoặc hàm kích hoạt. Hình dưới cho thấy chức năng của một nơron đơn lẻ.

Nút và kết nối



Nút và kết nối

Hoạt động của mạng nơ-ron phụ thuộc vào cấu trúc liên kết của nó, trọng số, điều kiện ưu tiên và chức năng truyền. Trọng số và độ lệch có thể được học, và hành vi học tập của một mạng phụ thuộc vào thuật toán đào tạo đã chọn. Thông thường, một hàm sigmoid được sử dụng làm hàm truyền, như thể hiện trong Hình 8.2 (a). Hàm sigmoid được đưa ra bởi:

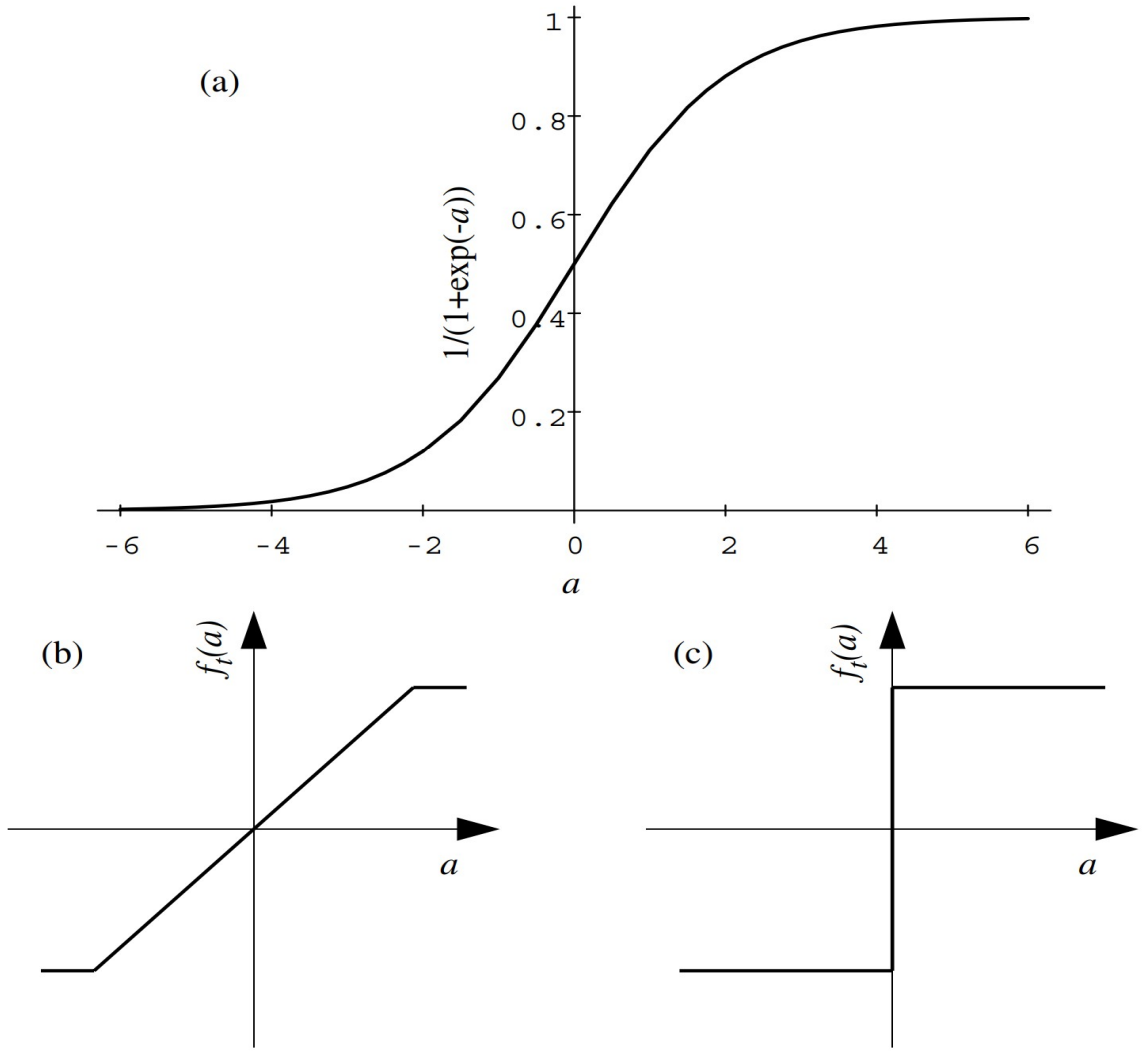
$$f_t(a) = \frac{1}{1+e^{-a}}$$

Đối với một tế bào thần kinh, kích hoạt a được đưa ra bởi:

$$a = \left(\sum_{i=1}^{n} w_i x_i \right) + w_0$$

trong đó n là số lượng đầu vào và số hạng ưu tiên W0 được xác định riêng cho mỗi nút. Hình dưới cho thấy các hàm dốc và bước, là các hàm phi tuyến thay thế đôi khi được sử dụng như các hàm truyền.

Nút và kết nối



Nút và kết nối

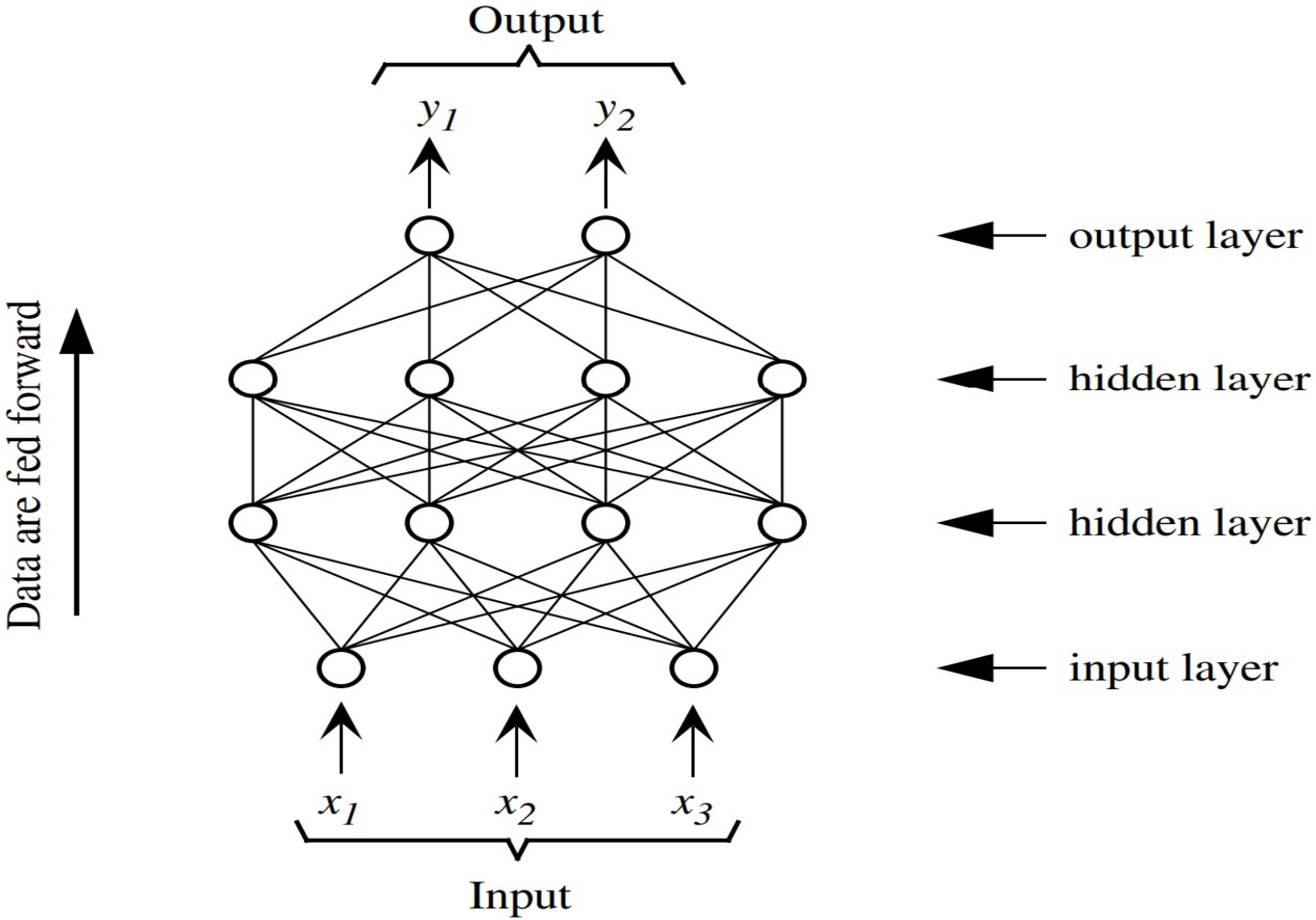
Có thể có nhiều cấu trúc liên kết mạng, ở đây sẽ tập trung vào một lựa chọn minh họa một số ứng dụng khác nhau cho mạng nơ-ron.
Xem xét các phần tử/lớp đơn và nhiều lớp, có thể được sử dụng để phân loại hoặc nói chung hơn là ánh xạ phi tuyến. Các mạng khác là mạng Hopfield để sử dụng như một bộ nhớ địa chỉ nội dung; MAXNET để chọn giá trị tối đa trong số các đầu vào của nó; mạng Hamming để phân loại; và cuối cùng là mạng ART1, mạng tự tổ chức Kohonen và mạng chức năng cơ sở xuyên tâm, tất cả đều được sử dụng để phân cụm.

Nhận thức đơn lớp và đa lớp

Cấu trúc mạng

Cấu trúc liên kết của phần tử/lớp nhiều lớp (MLP) được thể hiện trong Hình. Các tế bào thần kinh được tổ chức thành các lớp, sao cho mỗi tế bào thần kinh được kết nối hoàn toàn với các tế bào thần kinh ở các lớp trên và dưới, nhưng không liên kết với các tế bào thần kinh trong cùng một lớp. Các mạng này còn được gọi là mạng chuyển tiếp, mặc dù thuật ngữ này có thể được áp dụng chung hơn cho bất kỳ mạng nào mà hướng của luồng dữ liệu luôn là "chuyển tiếp", tức là hướng tới đầu ra. MLP có thể được sử dụng để phân loại hoặc làm công cụ ước lượng phi tuyến. Số lượng nút trong mỗi lớp và số lớp được xác định bởi trình xây dựng mạng, thường trên cơ sở thử và sai. Luôn luôn có một lớp đầu vào và một lớp đầu ra; số lượng nút trong mỗi nút được xác định bởi số lượng đầu vào và đầu ra đang được xem xét. Có thể có bất kỳ số lớp nào giữa hai lớp này. Không giống như các lớp đầu vào và đầu ra, các lớp ở giữa thường không có ý nghĩa rõ ràng liên quan đến chúng và chúng được gọi là các lớp ẩn. Nếu không có lớp ẩn, mạng là một phần tử/lớp lớp đơn (SLP). Mạng thể hiện trong Hình có ba nút đầu vào, hai lớp ẩn với mỗi nút bốn nút và lớp đầu ra gồm hai nút. Do đó, nó có thể được mô tả là MLP 3–4–4–2.

Nhận thức đơn lớp và đa lớp



Nhận thức đơn lớp và đa lớp

Một MLP hoạt động bằng cách cung cấp dữ liệu chuyển tiếp dọc theo các kết nối từ lớp đầu vào, qua các lớp ẩn, đến lớp đầu ra. Ngoại trừ các nút trong lớp đầu vào, các đầu vào cho một nút là các đầu ra từ mỗi nút trong lớp trước đó. Tại mỗi nút ngoài những nút trong lớp đầu vào, dữ liệu được tính trọng số, tính tổng, thêm vào độ lệch và sau đó được chuyển qua hàm truyền.

Có một số điểm không nhất quán trong tài liệu về việc đếm các lớp, xuất phát từ thực tế là các nút đầu vào không thực hiện bất kỳ quá trình xử lý nào mà chỉ đơn giản là đưa dữ liệu đầu vào vào các nút ở trên. Vì vậy, mặc dù mạng trong Hình rõ ràng là mạng bốn lớp, nhưng nó chỉ có ba lớp xử lý. SLP có hai lớp (lớp đầu vào và đầu ra) nhưng chỉ có một lớp xử lý, cụ thể là lớp đầu ra.

Nhận thức đơn lớp và đa lớp

Phần tử nhận thức là bộ phân loại

Nói chung, mạng nơ-ron được thiết kế sao cho có một nút đầu vào cho mỗi phần tử của vectơ đầu vào và một nút đầu ra cho mỗi phần tử của vectơ đầu ra. Vì vậy, trong một ứng dụng phân loại, mỗi nút đầu ra thường sẽ đại diện cho một lớp cụ thể. Một đại diện điển hình cho một lớp sẽ là giá trị gần bằng 1 xuất hiện ở nút đầu ra tương ứng, với các nút đầu ra còn lại tạo ra giá trị gần bằng 0. Cần có quy tắc quyết định đơn giản kết hợp với mạng, ví dụ: người chiến thắng lấy tất cả quy tắc chọn lớp tương ứng với nút có đầu ra cao nhất. Nếu vectơ đầu vào không thuộc bất kỳ lớp nào, không có giá trị đầu ra nào có thể rất cao. Vì lý do này, quy tắc quyết định phức tạp hơn có thể được sử dụng, ví dụ: quy tắc chỉ định rằng đầu ra từ nút chiến thắng cũng phải vượt quá ngưỡng định trước, chẳng hạn như 0,5.

Nhận thức đơn lớp và đa lớp

Phần tử nhận thức là bộ phân loại

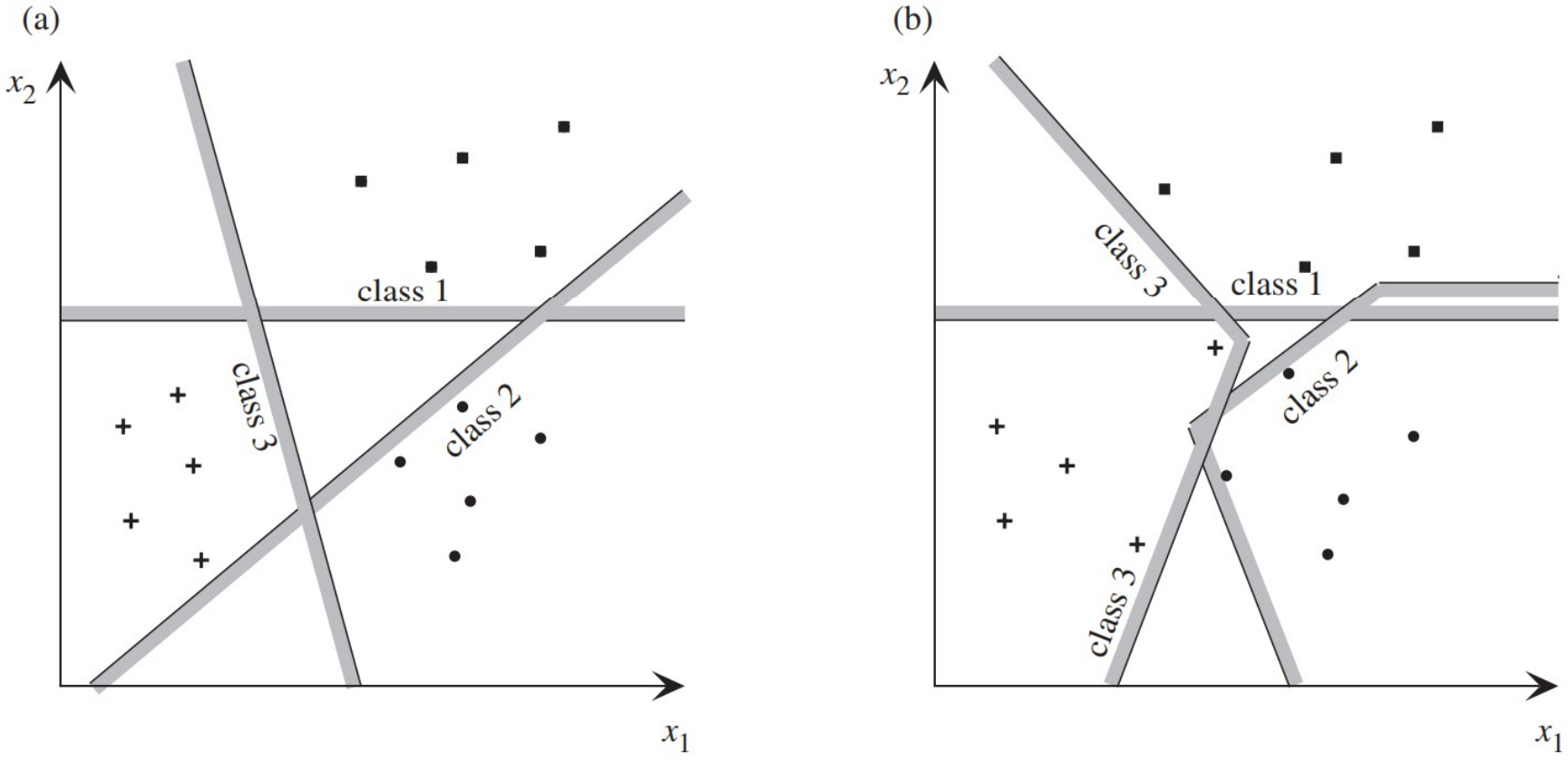
Các biểu diễn nhỏ gọn hơn cũng có thể thực hiện được cho các bài toán phân loại. Hallam và cộng sự chỉ sử dụng hai nút đầu ra để đại diện cho bốn lớp. Điều này đạt được bằng cách xử lý cả hai đầu ra với nhau, sao cho bốn khả năng tương ứng với bốn lớp là (0,0), (0,1), (1,0) và (1,1). Một nhược điểm của cách tiếp cận này là khó giải thích kết quả đầu ra không phù hợp chặt chẽ với một trong những khả năng này (ví dụ, kết quả đầu ra của (0,5, 0,5) sẽ đại diện cho điều gì?).

Nhận thức đơn lớp và đa lớp

Phần tử nhận thức là bộ phân loại

Trương hợp thông thường trong đó mỗi nút đầu ra đại diện cho một lớp riêng biệt. Nếu vectơ đầu vào có hai phần tử, nó có thể được biểu diễn dưới dạng một điểm trong không gian trạng thái hai chiều, đôi khi được gọi là không gian mẫu. Quá trình phân loại sau đó là một trong việc vẽ các đường phân chia giữa các vùng. Một phần tử/lớp lớp đơn, với hai tế bào thần kinh ở lớp đầu vào và cùng số lượng tế bào thần kinh ở lớp đầu ra khi có các lớp, có thể liên kết với mỗi lớp một đường phân chia thẳng duy nhất, như thể hiện trong Hình. Các lớp có thể được phân tách theo cách này được cho là có thể phân tách tuyến tính. Tổng quát hơn, vectơ đầu vào n chiều là các điểm trong siêu không gian n chiều. Nếu các lớp có thể được phân tách bằng (n – 1) siêu mặt phẳng, chúng có thể phân tách tuyến tính.

Nhận thức đơn lớp và đa lớp



Nhận thức đơn lớp và đa lớp

Phần tử nhận thức là bộ phân loại

Để xem cách SLP phân chia không gian mẫu với siêu mặt phẳng, hãy xem xét một nơ-ron xử lý đơn của SLP. Đầu ra của nó, trước khi áp dụng hàm truyền, là một số thực được cho bởi Công thức. Các vùng của không gian mẫu rõ ràng thuộc về lớp được đại diện bởi nơron sẽ tạo ra giá trị dương mạnh và các vùng rõ ràng không thuộc về lớp sẽ tạo ra giá trị âm mạnh. Việc phân loại ngày càng trở nên không chắc chắn khi kích hoạt a trở nên gần bằng 0 và tiêu chuẩn phân chia thường được giả định là a = 0. Điều này sẽ tương ứng với kết quả đầu ra là 0,5 sau khi áp dụng hàm truyền sigmoid (Hình) . Do đó, siêu phẳng phân tách hai vùng được cho bởi:

$$\left(\sum_{i=1}^{n} w_i x_i\right) + w_0 = 0$$

Nhận thức đơn lớp và đa lớp

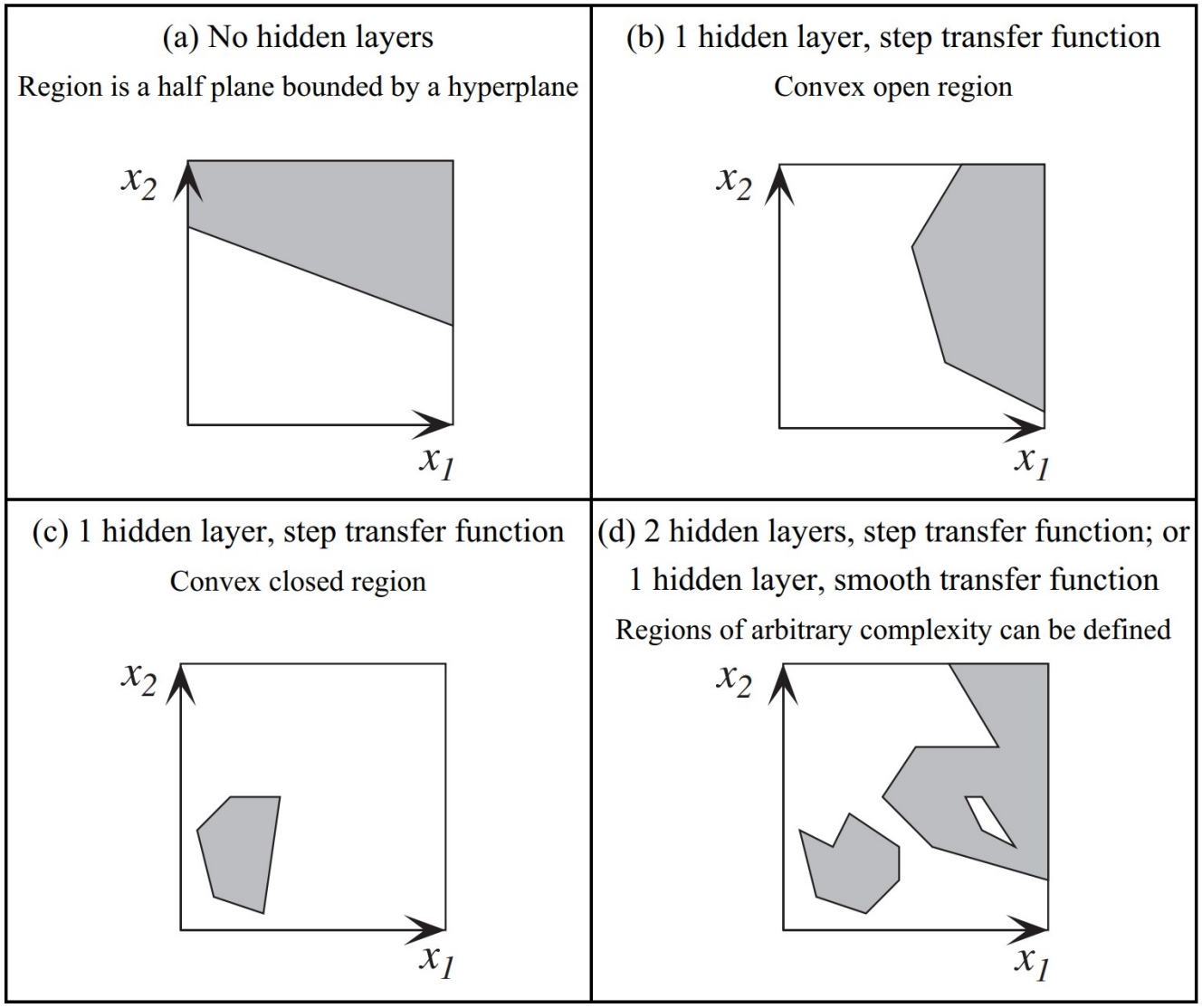
Phần tử nhận thức là bộ phân loại

Trong trường hợp có hai đầu vào, Công thức đơn giản trở thành phương trình của một đường thẳng, vì nó có thể được sắp xếp lại như sau:

$$x_2 = \frac{-w_1}{w_2} x_1 - \frac{w_0}{w_2}$$

trong đó – w1/w2 là gradient (độ dốc) và – w0/w2 là điểm giao nhau trên trục x2.

Nhận thức đơn lớp và đa lớp



Nhận thức đơn lớp và đa lớp

Phần tử nhận thức là bộ phân loại

Đối với các vấn đề không thể phân tách tuyến tính, như trong Hình, các vùng có độ phức tạp tùy ý có thể được vẽ trong không gian trạng thái bởi phần tử/lớp nhiều lớp với một lớp ẩn và một hàm truyền có thể phân biệt, tức là, trơn tru, chẳng hạn như hàm sigmoid (Hình). Lớp xử lý đầu tiên của MLP có thể được coi là phân chia không gian trạng thái bằng các đường thẳng (hoặc siêu mặt phẳng) và lớp xử lý thứ hai tạo thành các vùng đa diện bởi các tổ hợp Boolean (AND, OR và NOT) của các vùng được phân tách tuyến tính. Do đó, người ta thường chấp nhận rằng chỉ một lớp ẩn là cần thiết để thực hiện bất kỳ ánh xạ hoặc phân loại phi tuyến nào với MLP sử dụng chức năng truyền sigmoid (Hình). Đây là Định lý tồn tại của Kolmogorov. Tương tự, không cần nhiều hơn hai lớp ẩn nếu sử dụng chức năng chuyển bước. Tuy nhiên, khả năng học hỏi từ một tập hợp các ví dụ không thể được đảm bảo và do đó, cấu trúc liên kết chi tiết của mạng chắc chắn liên quan đến một số lần thử và sai nhất định. Một cách tiếp cận thực dụng đối với thiết kế mạng là bắt đầu với một mạng nhỏ và mở rộng số lượng các nút hoặc lớp nếu cần thiết.

Nhận thức đơn lớp và đa lớp

Đào tạo một lớp nhận thức

Một trong những thuật toán huấn luyện được sử dụng phổ biến nhất là thuật toán lan truyền lỗi ngược, đôi khi được gọi là quy tắc đồng bằng tổng quát. Đây là một kỹ thuật giảm độ dốc tỷ lệ thuận (xem Chương 7), và nó dựa vào việc hàm truyền là liên tục và có thể phân biệt được. Hàm sigmoid (Hình) là một lựa chọn đặc biệt thích hợp vì đạo hàm của nó được cho bởi:

$$f_t'(a) = f_t(a)\,(1 - f_t(a))$$

Việc sử dụng thuật toán lan truyền lỗi ngược để tối ưu hóa trọng số và điều kiện sai lệch có thể được làm rõ ràng hơn bằng cách coi các sai lệch là trọng số trên các kết nối từ các nút giả, có đầu ra luôn là 1, như thể hiện trong Hình. Lưu đồ mô tả thuật toán lan truyền lỗi ngược được trình bày trong Hình, sử dụng danh pháp thể hiện trong Hình.

Nhận thức đơn lớp và đa lớp

Đào tạo một lớp nhận thức

Cốt lõi của thuật toán là quy tắc delta xác định các sửa đổi đối với trọng số, ∆wBij:

$$\Delta w_{Bij} = \eta \delta_{Bi} y_{Aj} + \alpha(w_{Bij})$$

cho tất cả các nút j trong lớp A và tất cả các nút i trong lớp B, trong đó A = B - 1. Tế bào thần kinh trong lớp đầu ra và trong các lớp ẩn có một thuật ngữ lỗi liên quan, δ (phát âm là delta). Khi sử dụng hàm truyền sigmoid, δAj được cho bởi:

$$\left.\begin{array}{l} \text{output layer :} \\ \delta_{Ai} = f_t'(y_{Ai})(d_i - y_{Ai}) = y_{Ai}(1 - y_{Ai})(d_i - y_{Ai}) \\ \\ \text{hidden layers :} \\ \delta_{Aj} = f_t'(y_{Aj}) \sum_i \delta_{Bi} w_{Bij} = y_{Aj}(1 - y_{Aj}) \sum_i \delta_{Bi} w_{Bij} \end{array}\right\}$$

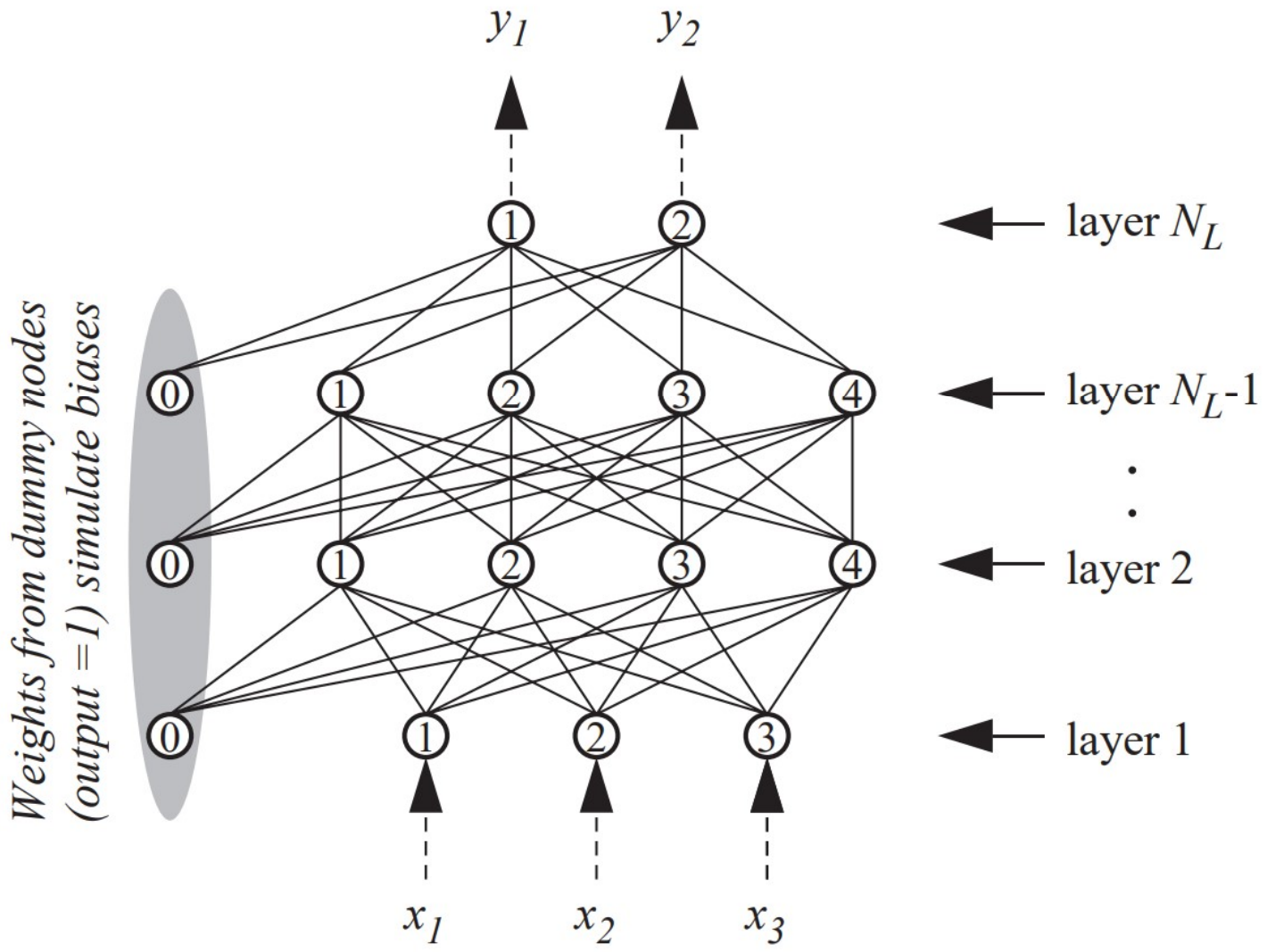
Nhận thức đơn lớp và đa lớp

Đào tạo một lớp nhận thức

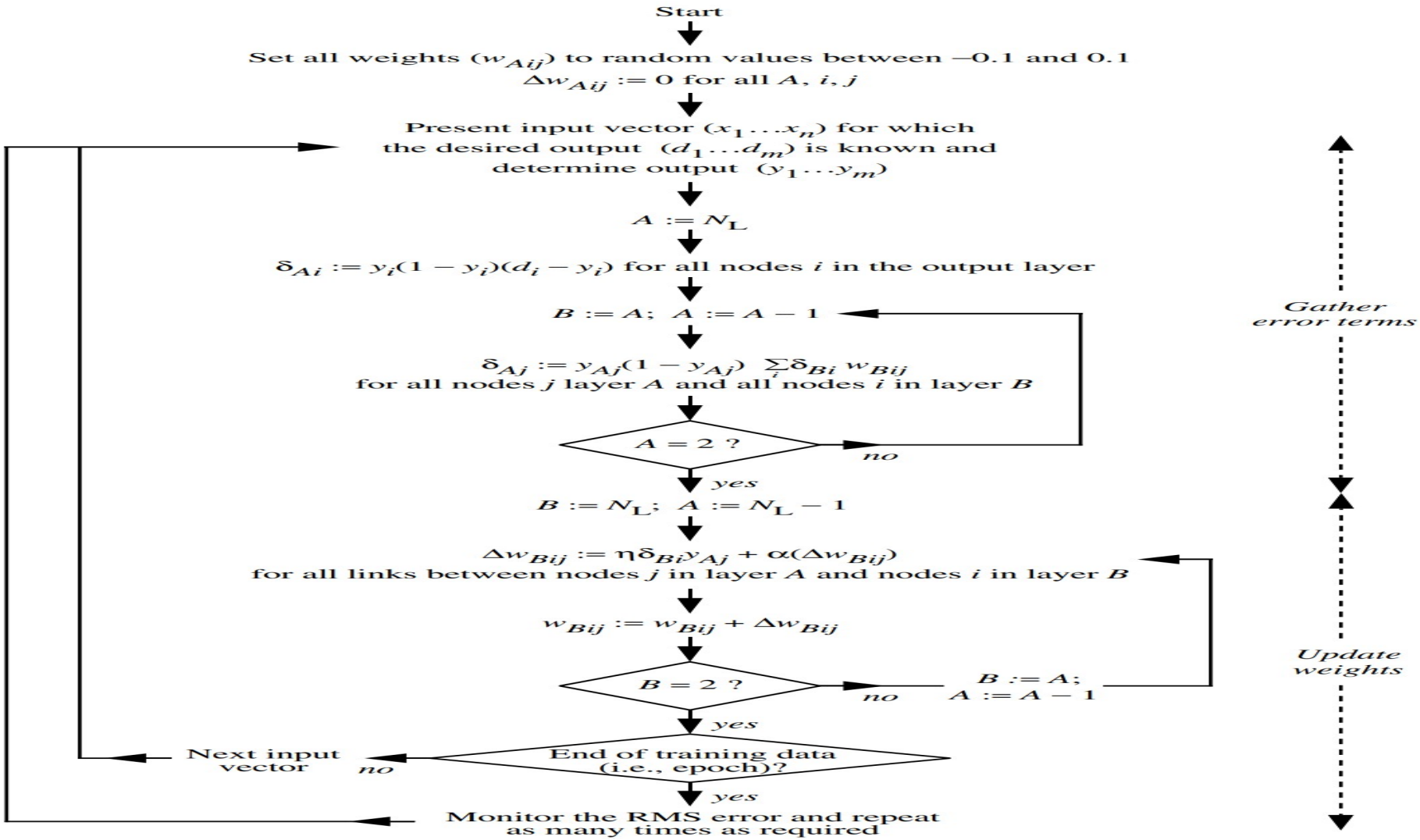
Tốc độ học tập, η, được áp dụng cho các giá trị được tính toán cho δAj. Knight đề xuất giá trị cho η khoảng 0,35. Như được viết trong Công thức, quy tắc delta bao gồm hệ số động lượng, α, mặc dù điều này đôi khi bị bỏ qua. Kỹ thuật giảm độ dốc tỷ lệ thuận có thể không hiệu quả, đặc biệt là gần với mức tối thiểu trong hàm chi phí, trong trường hợp này là lỗi RMS của đầu ra. Để giải quyết vấn đề này, một thuật ngữ xung lượng buộc những thay đổi về trọng lượng phụ thuộc vào những thay đổi về trọng lượng trước đó. Giá trị của hệ số xung lượng phải nằm trong khoảng 0–1. Knight gợi ý rằng α được đặt thành 0,0 trong vài lần luyện tập đầu tiên và sau đó tăng lên 0,9.

Các thuật toán huấn luyện khác cũng đã được áp dụng thành công cho các phần tử cảm nhận. Ví dụ, Willis và các cộng sự ủng hộ thuật toán chemotaxis, thuật toán này kết hợp một yếu tố thống kê ngẫu nhiên.

Nhận thức đơn lớp và đa lớp



Nhận thức đơn lớp và đa lớp



Nhận thức đơn lớp và đa lớp

Cây phân cấp nhận thức

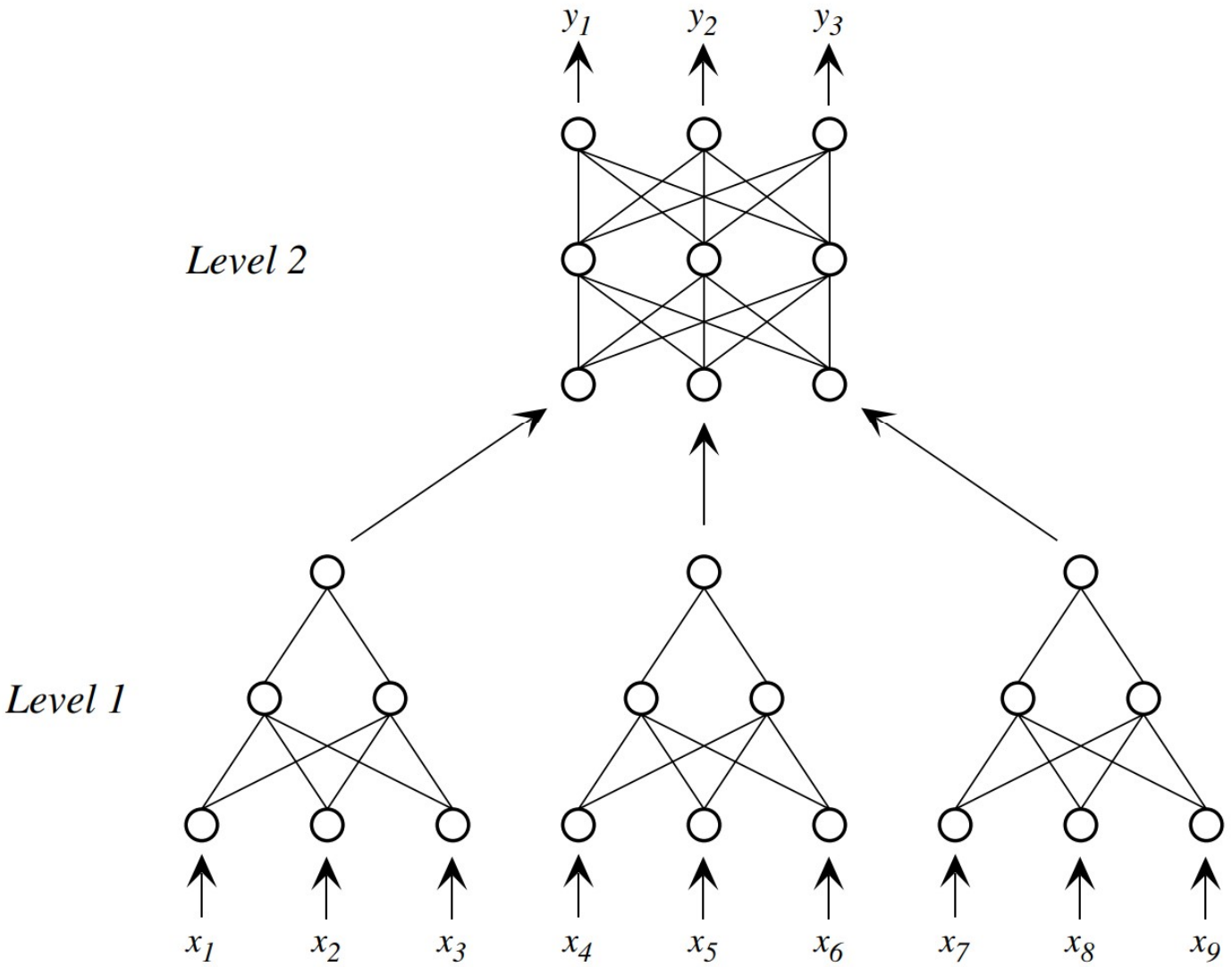
Trong các vấn đề phức tạp liên quan đến nhiều đầu vào, một số nhà nghiên cứu khuyên nên chia MLP thành nhiều MLP nhỏ hơn được sắp xếp theo thứ bậc, như thể hiện trong Hình. Trong ví dụ này, hệ thống phân cấp bao gồm hai cấp. Các đầu vào được chia sẻ giữa các MLP ở cấp 1 và đầu ra từ các mạng này tạo thành các đầu vào cho MLP ở cấp 2. Cách tiếp cận này thường hữu ích nếu các biến trung gian có ý nghĩa có thể được xác định là đầu ra từ các MLP cấp 1. Ví dụ, nếu đầu vào là phép đo từ cảm biến cho thiết bị giám sát, thì đầu ra mức 1 có thể biểu thị chẩn đoán bất kỳ lỗi nào và đầu ra mức 2 có thể đại diện cho các hành động kiểm soát được khuyến nghị. Trong ví dụ này, về nguyên tắc, một MLP lớn duy nhất có thể được sử dụng để ánh xạ trực tiếp từ các phép đo cảm biến đến các hành động kiểm soát được khuyến nghị. Tuy nhiên, sự hội tụ của các mạng nhỏ hơn trong MLP phân cấp có thể đạt được dễ dàng hơn. Hơn nữa, vì các mạng cấu thành trong MLP phân cấp độc lập với nhau, chúng có thể được đào tạo riêng biệt hoặc song song.

Nhận thức đơn lớp và đa lớp

Lưu ý khi áp dụng

Đôi khi, dừng quá trình đào tạo trước thời điểm không thể giảm thêm lỗi RMS. Điều này là do có thể đào tạo quá mức mạng, để nó trở thành chuyên gia trong việc đưa ra kết quả đầu ra chính xác cho dữ liệu đào tạo, nhưng ít chuyên gia hơn trong việc xử lý dữ liệu mới. Đây có thể là sự cố nếu mạng đã được huấn luyện trong quá nhiều chu kỳ hoặc nếu mạng quá phức tạp đối với nhiệm vụ đang thực hiện. Ví dụ, việc bao gồm các lớp ẩn bổ sung hoặc số lượng lớn tế bào thần kinh bên trong các lớp ẩn sẽ có xu hướng thúc đẩy quá trình đào tạo. Ảnh hưởng của việc huấn luyện quá mức được thể hiện trong Hình đối với ánh xạ phi tuyến của một tham số đầu vào đơn lẻ lên một tham số đầu ra duy nhất và Hình cho thấy tác động của việc huấn luyện quá mức bằng cách sử dụng dữ liệu phân loại có thể phân tách phi tuyến tính từ Hình.

Nhận thức đơn lớp và đa lớp

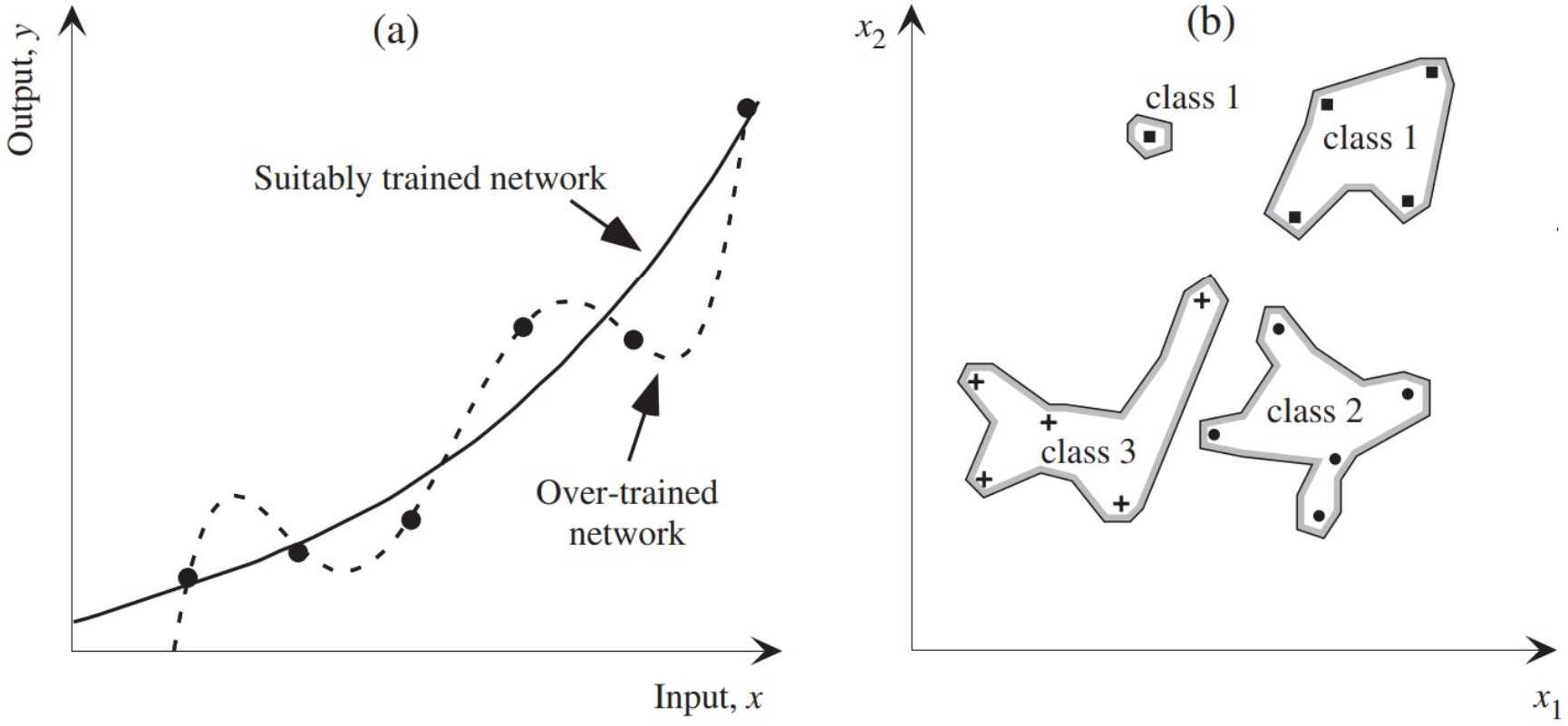


Nhận thức đơn lớp và đa lớp

Lưu ý khi áp dụng

Một cách để tránh đào tạo quá mức là chia dữ liệu thành ba bộ, được gọi là dữ liệu đào tạo, kiểm tra và xác thực. Quá trình đào tạo diễn ra bằng cách sử dụng dữ liệu đào tạo và lỗi RMS với những dữ liệu này được theo dõi. Tuy nhiên, vào những khoảng thời gian định trước, quá trình huấn luyện sẽ bị tạm dừng và số tạ hiện tại được lưu lại. Tại những điểm này, trước khi tiếp tục đào tạo, mạng được hiển thị với dữ liệu kiểm tra và một lỗi RMS được tính toán. Lỗi RMS đối với dữ liệu đào tạo giảm dần cho đến khi nó ổn định. Tuy nhiên, lỗi RMS đối với dữ liệu thử nghiệm có thể vượt qua mức tối thiểu và sau đó bắt đầu tăng trở lại do ảnh hưởng của việc huấn luyện quá mức, như thể hiện trong Hình. Ngay khi lỗi RMS đối với dữ liệu thử nghiệm bắt đầu tăng lên, mạng đã được huấn luyện quá mức, nhưng tập trọng số được lưu trữ trước đó sẽ gần với mức tối ưu. Cuối cùng, hiệu suất của mạng có thể được đánh giá bằng cách kiểm tra nó bằng cách sử dụng dữ liệu xác thực chưa từng thấy trước đó.

Nhận thức đơn lớp và đa lớp

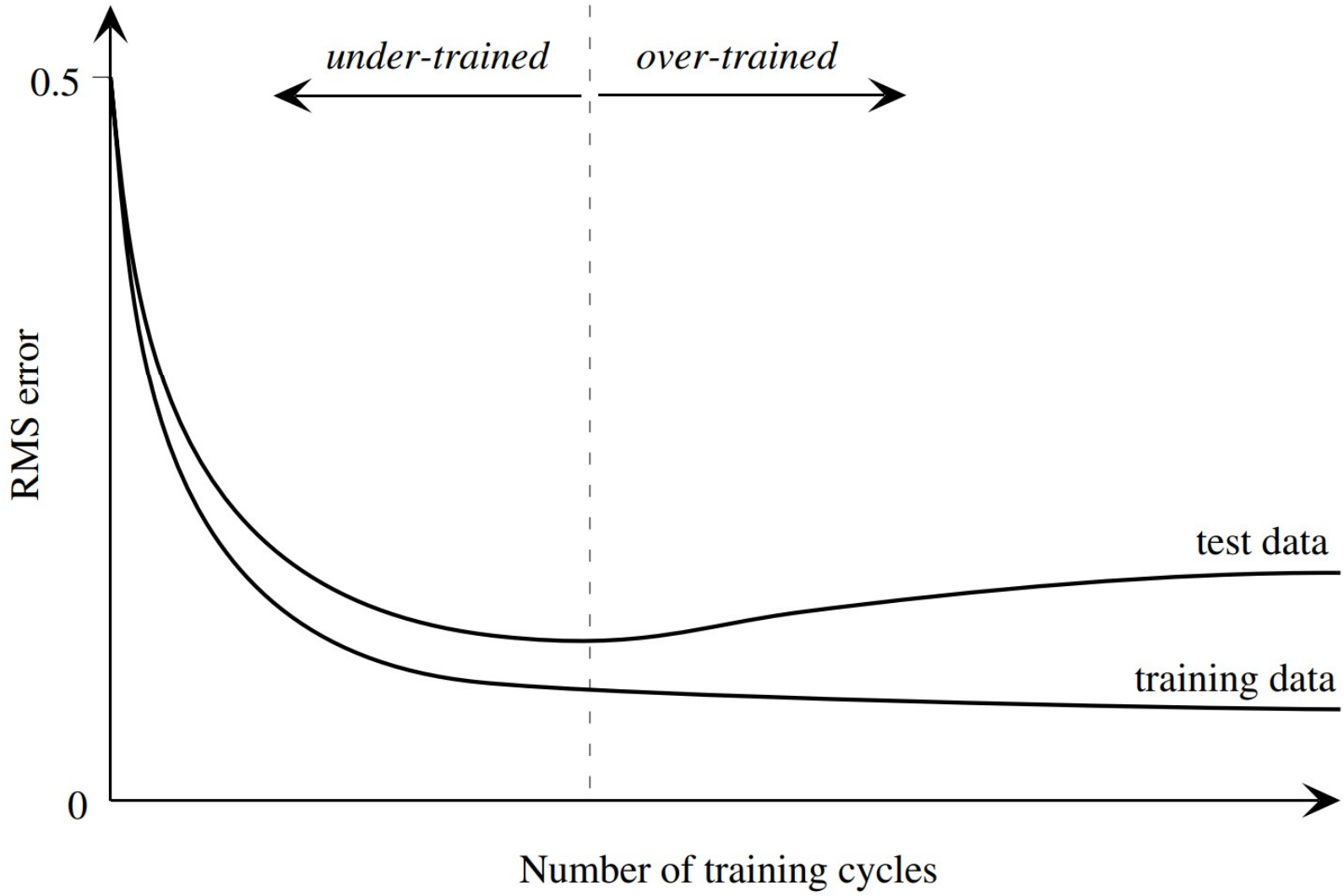


Nhận thức đơn lớp và đa lớp

Lưu ý khi áp dụng

Một vấn đề thường xuyên gặp phải trong các ứng dụng thực tế là thiếu dữ liệu phù hợp để đào tạo và kiểm tra mạng nơ-ron. Hopgood và cộng sự mô tả một vấn đề phân loại trong đó chỉ có 20 ví dụ phù hợp, cần được chia sẻ giữa dữ liệu đào tạo và kiểm tra. Họ đã sử dụng một kỹ thuật gọi là bỏ đi như một cách để giảm bớt ảnh hưởng của vấn đề này. Kỹ thuật này liên quan đến việc đào tạo liên tục về tất cả trừ một trong những ví dụ và thử nghiệm trên cái còn thiếu. Vì vậy, trong trường hợp này, ban đầu mạng sẽ được đào tạo trên 19 ví dụ và thử nghiệm trên ví dụ còn lại. Quy trình này được lặp lại 19 lần nữa: bỏ qua một ví dụ khác nhau mỗi lần khỏi dữ liệu huấn luyện, đặt lại trọng số về giá trị ngẫu nhiên, đào tạo lại và sau đó kiểm tra trên ví dụ đã bỏ qua. Kỹ thuật bỏ đi rõ ràng là tốn thời gian vì nó liên quan đến việc đặt lại trọng số, đào tạo, kiểm tra và cho điểm mạng nhiều lần - trong ví dụ này là 20 lần. Ưu điểm của nó là hiệu suất của mạng có thể được đánh giá bằng cách sử dụng mọi ví dụ có sẵn như thể nó là dữ liệu thử nghiệm chưa từng thấy trước đây.

Nhận thức đơn lớp và đa lớp



Nhận thức đơn lớp và đa lớp

Lưu ý khi áp dụng

Mạng nơ-ron chấp nhận số thực chỉ hiệu quả nếu các giá trị đầu vào bị giới hạn trong phạm vi phù hợp, thường là từ 0 đến 1 hoặc từ 1 đến 1. Phạm vi đầu ra phụ thuộc vào chức năng truyền đã chọn, ví dụ: phạm vi đầu ra là giữa 0 và 1 nếu hàm sigmoid được sử dụng. Trong các ứng dụng thực tế, các giá trị đầu vào và đầu ra thực tế có thể nằm ngoài các dải này hoặc có thể bị giới hạn trong một dải hẹp bên trong chúng. Trong cả hai trường hợp, dữ liệu sẽ cần được chia tỷ lệ, thường là tuyến tính, trước khi được trình bày cho mạng nơ-ron. Một số gói mạng nơ-ron thực hiện việc mở rộng quy mô một cách tự động.

Mạng Hopfield

Mạng Hopfield chỉ có một lớp và các nút được sử dụng cho cả đầu vào và đầu ra. Cấu trúc liên kết mạng được thể hiện trong Hình. Mạng thường được sử dụng như một bộ nhớ địa chỉ theo nội dung trong đó mỗi ví dụ huấn luyện được coi như một vector mô hình hoặc ví dụ, được mạng lưu trữ. Mạng Hopfield sử dụng các giá trị đầu vào nhị phân, thường là 1 và 1. Bằng cách sử dụng độ phi tuyến tính của bước được thể hiện trong Hình khi hàm truyền ft, đầu ra cũng buộc phải duy trì trạng thái nhị phân. Nếu kích hoạt a là 0, tức là ở bước này, đầu ra là không xác định, vì vậy cần có quy ước để mang lại đầu ra là 1 hoặc 1.

Mạng Hopfield

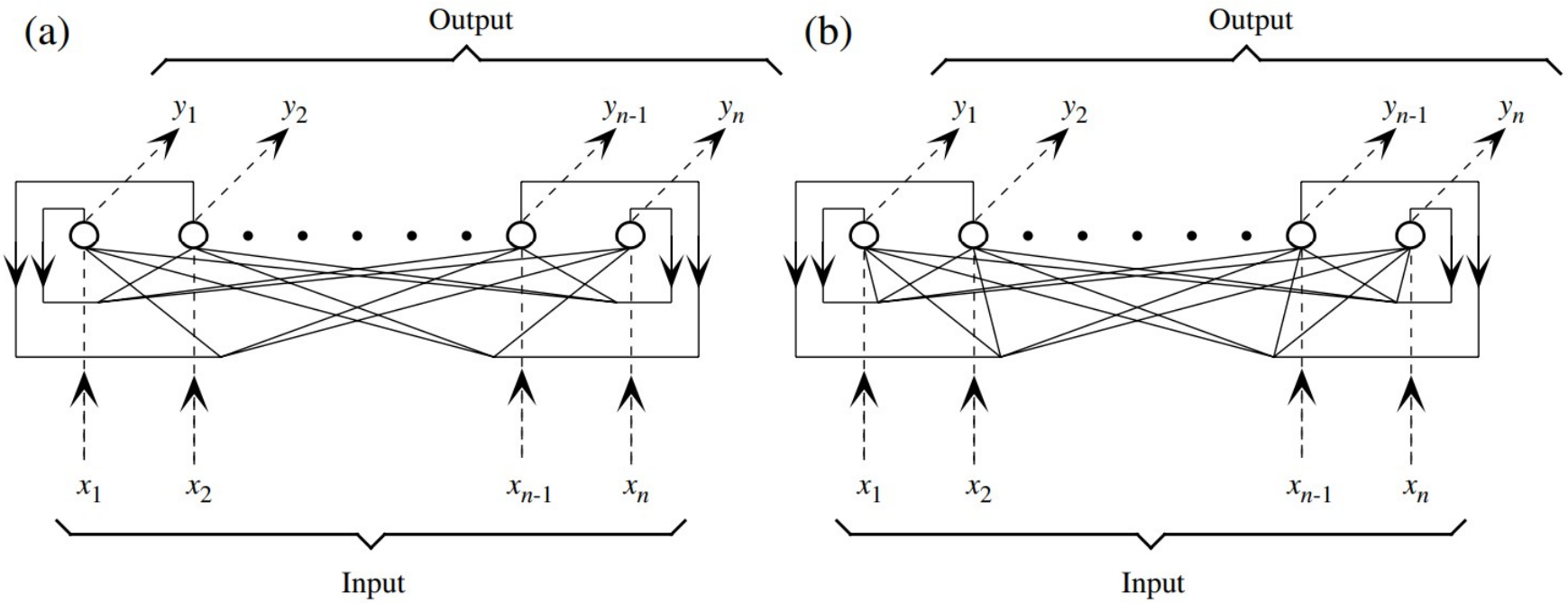
Nếu mạng có Nn nút, thì cả đầu vào và đầu ra sẽ bao gồm một vectơ gồm Nn chữ số nhị phân. Nếu có các mẫu Ne được lưu trữ, trọng số và độ lệch của mạng được thiết lập theo các phương trình sau:

$$w_{ij} = \begin{cases} \sum_{k=1}^{N_e} x_{ik} x_{jk} & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases}$$

$$w_{i0} = \sum_{k=1}^{N_e} x_{ik}$$

trong đó wij là trọng số trên kết nối từ nút i đến nút j, wi0 là độ lệch trên nút i và xik là chữ số thứ i của ví dụ k. Không có kết nối vòng nào từ một nút đến chính nó, do đó wij = 0 trong đó i = j.

Mạng Hopfield



Mạng Hopfield

Việc thiết lập các trọng số theo cách này tạo thành giai đoạn học tập, và kết quả là các giá trị mẫu được lưu trữ theo kiểu phân tán trong mạng. Nếu một vectơ mới sau đó được trình bày dưới dạng đầu vào, thì vectơ này ban đầu cũng là đầu ra, vì các nút được sử dụng cho cả đầu vào và đầu ra. Chức năng nút (Hình) sau đó được thực hiện song song trên mỗi nút. Nếu điều này được lặp lại nhiều lần, đầu ra sẽ được sửa đổi dần dần và sẽ hội tụ về mẫu gần giống nhất với vectơ đầu vào ban đầu. Để lưu trữ đáng tin cậy ít nhất một nửa số mẫu, Hopfield đã ước tính rằng số lượng mẫu (Ne) không được vượt quá khoảng 0,15Nn.

Nếu mạng được sử dụng để phân loại, thì cần phải thực hiện thêm một giai đoạn nữa trong đó kết quả được so sánh với các mẫu để chọn ra kết quả phù hợp.

MAXNET

MAXNET (Hình) có cấu trúc liên kết giống hệt với mạng Hopfield, ngoại trừ trọng số trên các kết nối hình tròn, wii, không phải lúc nào cũng bằng không vì chúng nằm trong mạng Hopfield. MAXNET được sử dụng để nhận biết đầu vào nào của nó có giá trị cao nhất. Trong vai trò này, nó đôi khi được sử dụng kết hợp với các mạng khác, chẳng hạn như phần tử/lớp nhiều lớp, để chọn nút đầu ra tạo ra giá trị cao nhất. Giả sử rằng phần tử/lớp nhiều lớp có bốn nút đầu ra, tương ứng với bốn danh mục khác nhau. Một MAXNET để xác định giá trị đầu ra tối đa (và do đó, giải pháp cho nhiệm vụ phân loại) sẽ có bốn nút và bốn mẫu đầu ra thay thế sau khi hội tụ, tức là bốn ví dụ:

Trong đó X> 0. MAXNET sẽ điều chỉnh giá trị đầu vào cao nhất của nó thành X và giảm các giá trị khác xuống 0. Lưu ý rằng MAXNET được xây dựng để có cùng số nút (Nn) với số mẫu (Ne). Ngược lại điều này với mạng Hopfield, mạng này cần số nút nhiều gấp bảy lần so với mạng mẫu.

$$\begin{matrix} X & 0 & 0 & 0 \\ 0 & X & 0 & 0 \\ 0 & 0 & X & 0 \\ 0 & 0 & 0 & X \end{matrix}$$

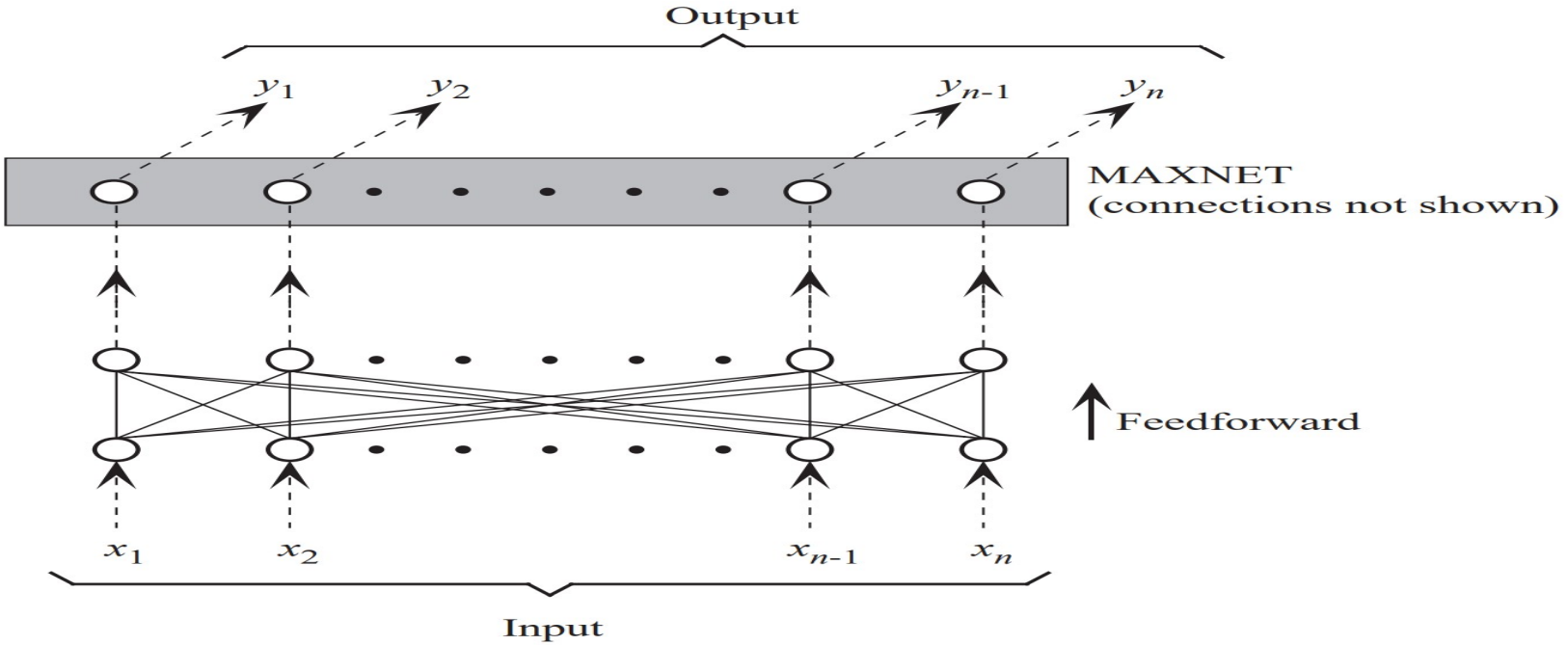
MAXNET

Sử dụng cùng một ký hiệu như Công thức 8.8 và 8.9, trọng số kết nối được đặt như sau:

$$w_{ij} = \begin{cases} -\varepsilon & \text{where } \varepsilon < \dfrac{1}{N_n} \quad \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

Mạng Hamming

Mạng Hamming có hai phần - một mạng truyền thẳng hai lớp và một MAXNET, như trong Hình. Mạng chuyển tiếp được sử dụng để so sánh vectơ đầu vào với từng ví dụ, đưa ra điểm phù hợp cho từng ví dụ. Sau đó, MAXNET được sử dụng để chọn ra ví dụ đạt điểm cao nhất. Hiệu quả tổng thể là mạng có thể phân loại vector đầu vào của nó.

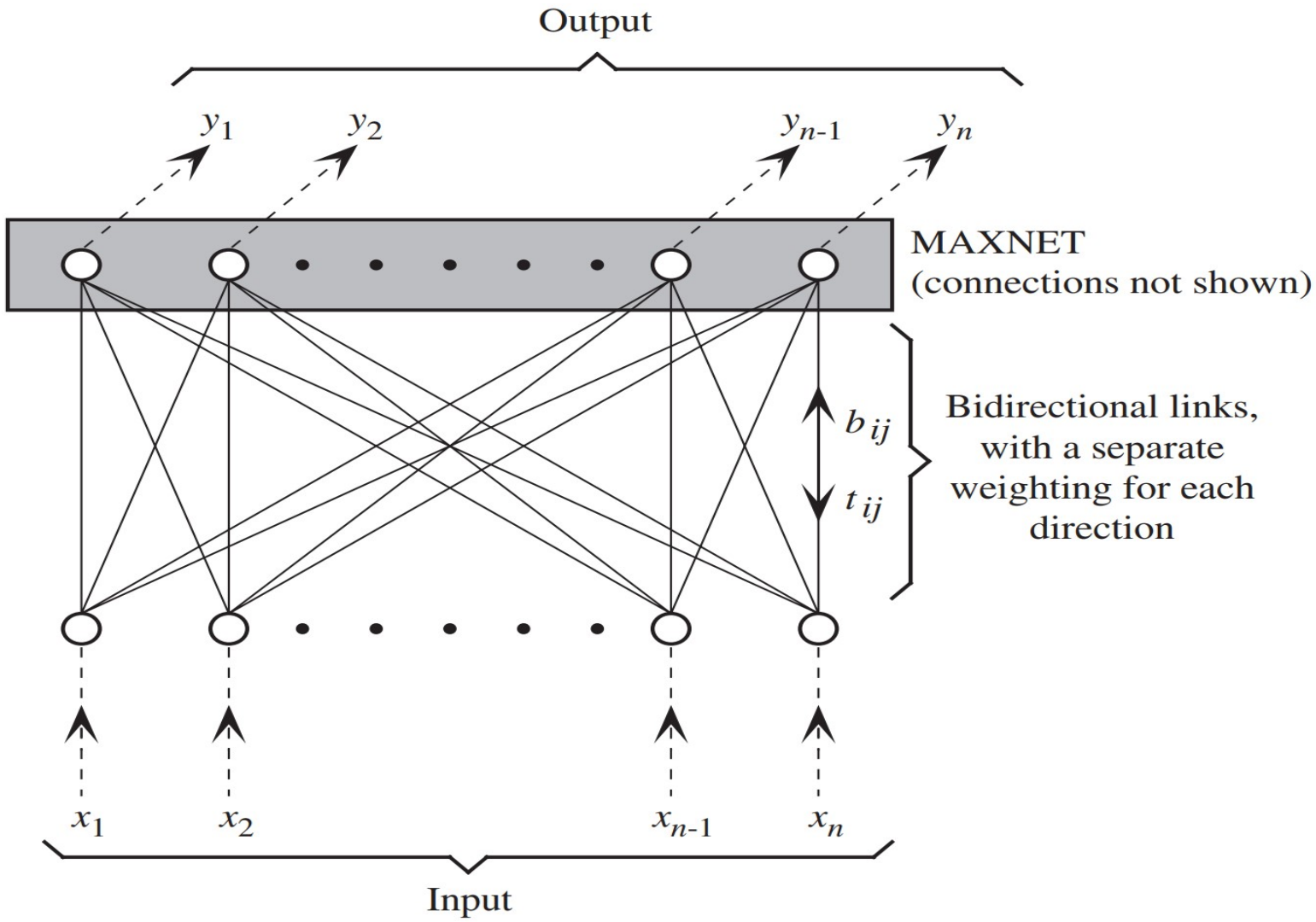


Mạng lý thuyết cộng hưởng thích ứng (ART)

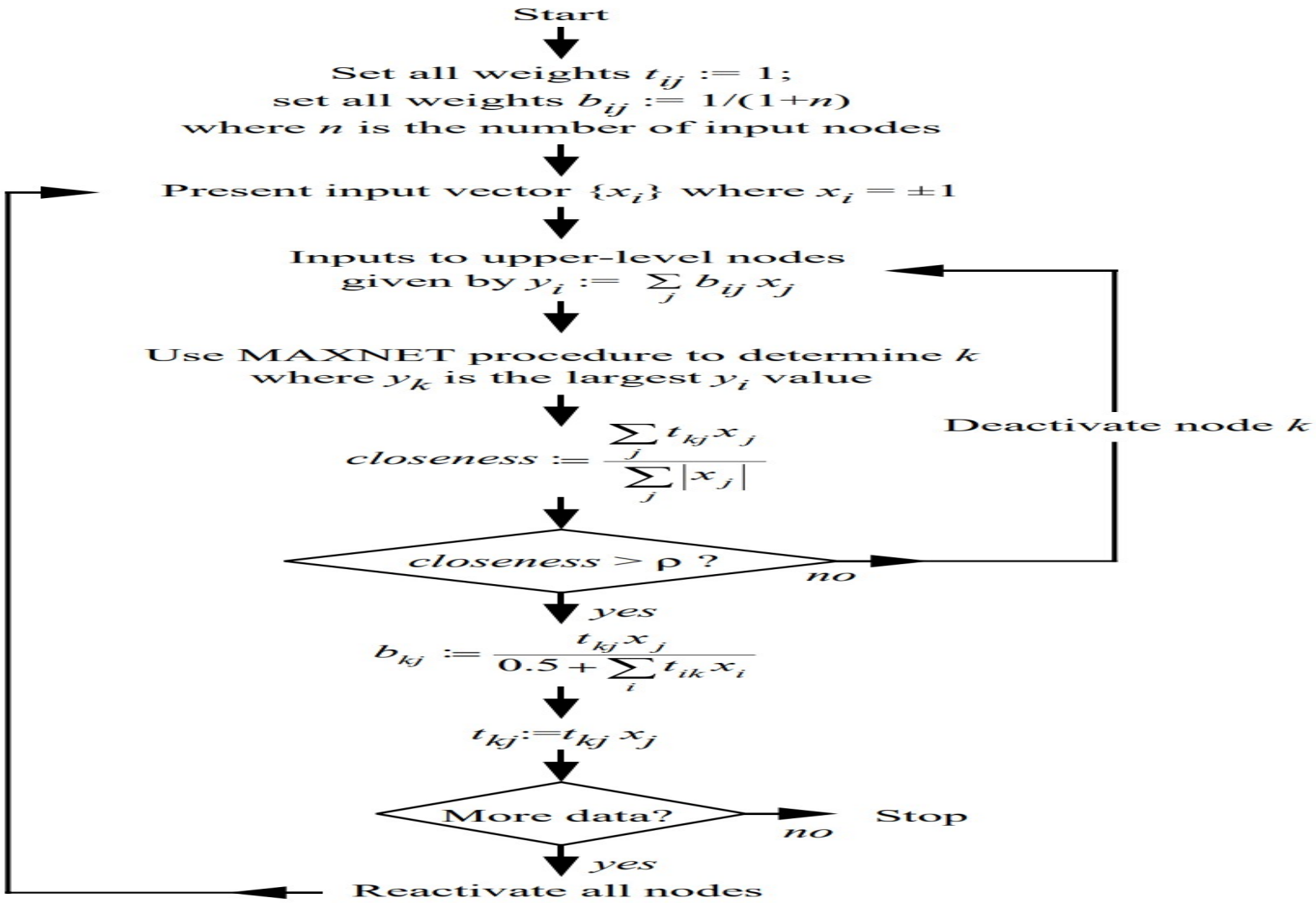
Các mạng Lý thuyết Cộng hưởng Thích ứng (ART1 và ART2) của Carpenter và Grossberg đáng được đề cập đến vì chúng là ví dụ về các mạng học mà không cần giám sát. Cấu trúc liên kết mạng ART1 bao gồm các kết nối hai chiều giữa một tập hợp các nút đầu vào và MAXNET, như thể hiện trong Hình.

Mạng phân loại dữ liệu đến thành các cụm. Khi ví dụ đầu tiên được hiển thị trên mạng, nó sẽ được lưu trữ dưới dạng một mẫu ví dụ hoặc mẫu mô hình. Ví dụ thứ hai sau đó được so sánh với ví dụ mẫu và được coi là tương tự đủ để thuộc cùng một cụm hoặc được lưu trữ dưới dạng một ví dụ mới. Nếu một ví dụ được coi là thuộc về một cụm đã được xác định trước đó, thì ví dụ cho cụm đó sẽ được sửa đổi để tính đến thành viên mới. Hiệu suất của mạng phụ thuộc vào cách thức đo lường sự khác biệt, tức là thước đo mức độ gần gũi và ngưỡng hoặc cảnh giác, ρ, ngoài ra còn có thể lưu trữ một ví dụ mới. Khi mỗi vectơ mới được trình bày, nó được so sánh song song với tất cả các vectơ hiện tại. Số lượng các mẫu ngày càng tăng khi mạng được sử dụng, tức là mạng học các mẫu mới. Hoạt động của mạng ART1, sử dụng các đầu vào nhị phân, được tóm tắt trong Hình. ART2 cũng tương tự nhưng lấy các đầu vào liên tục khác nhau.

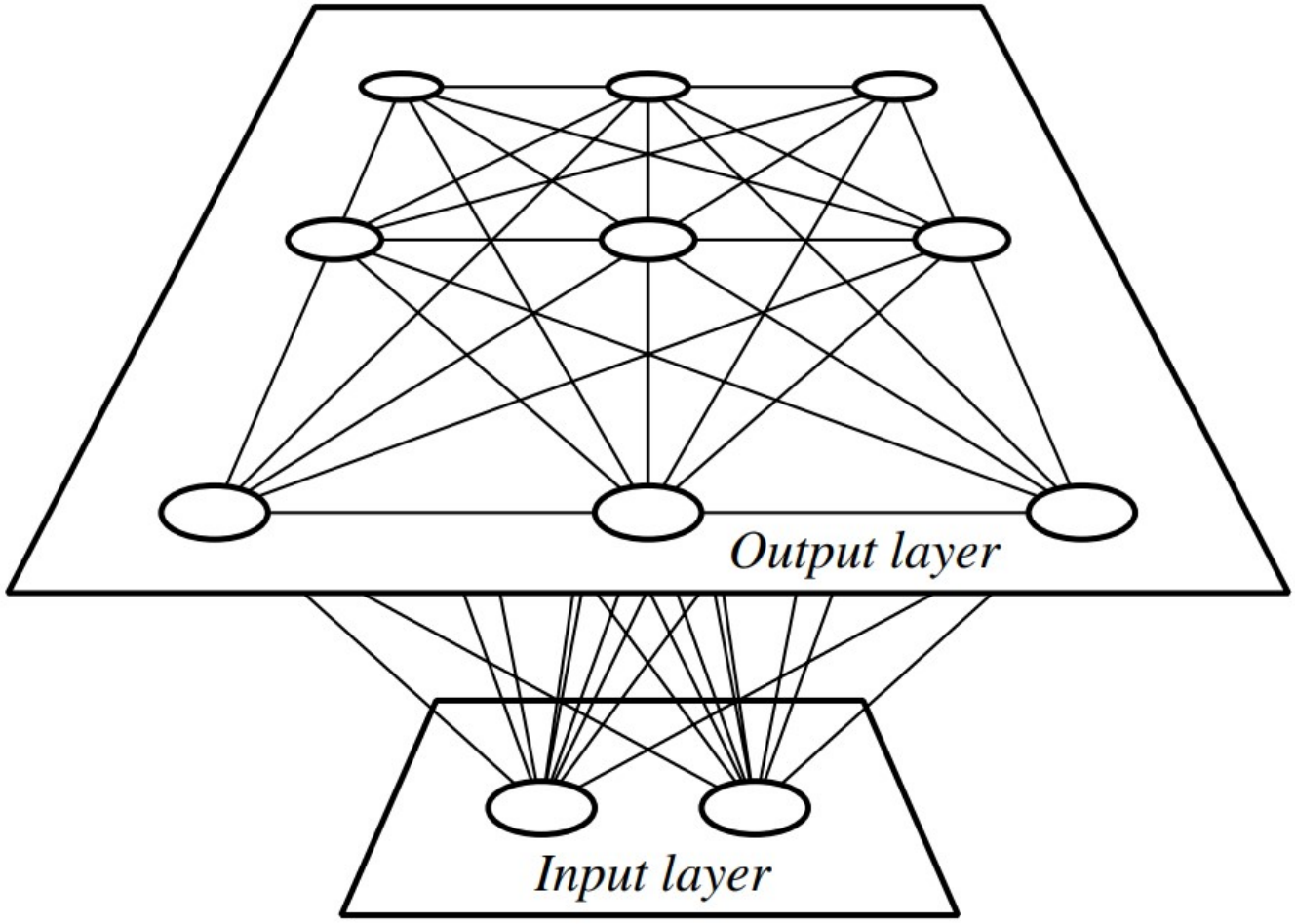
Mạng lý thuyết cộng hưởng thích ứng (ART)



Mạng lý thuyết cộng hưởng thích ứng (ART)



Mạng lý thuyết cộng hưởng thích ứng (ART)



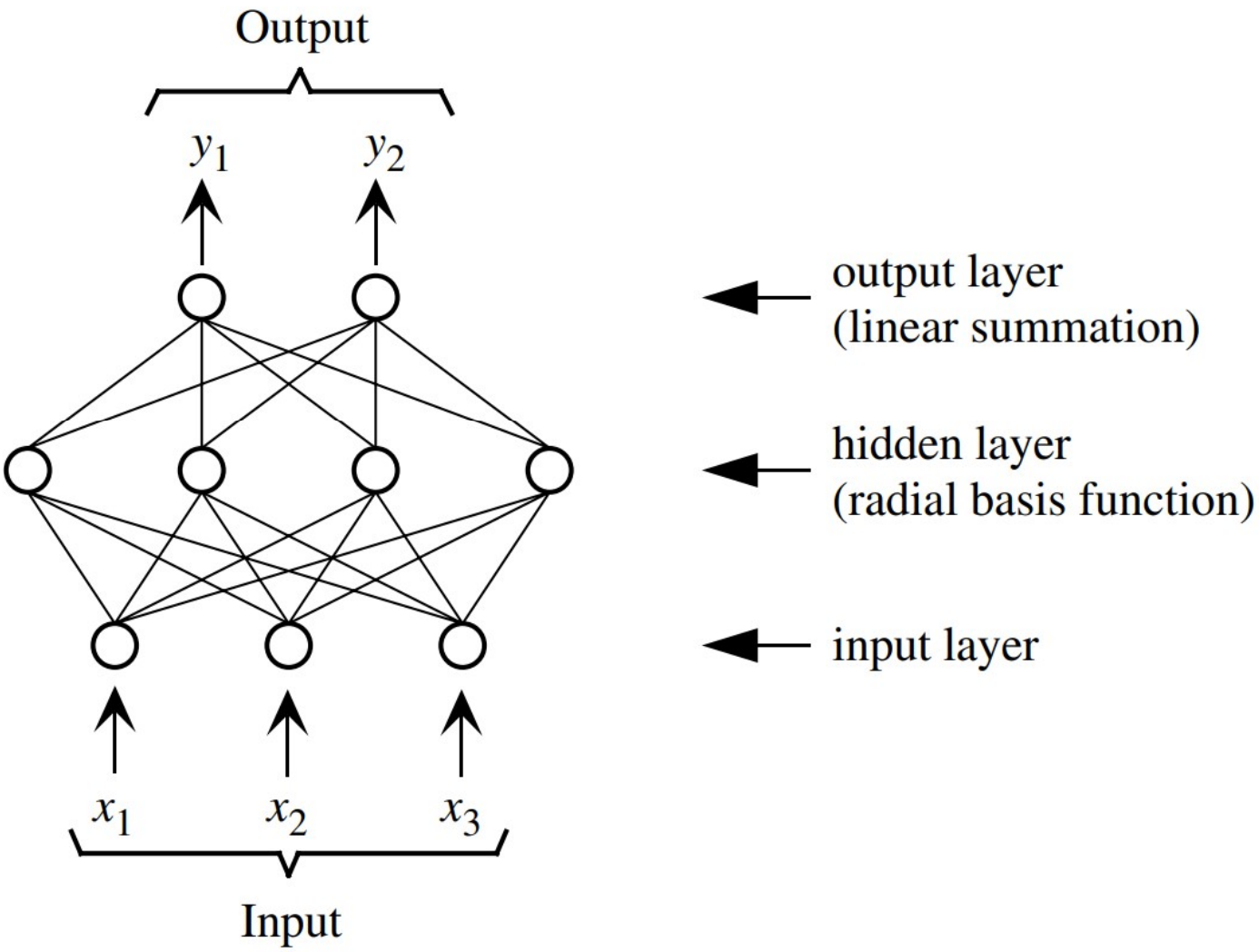
Mạng tự tổ chức Kohonen

Các mạng tự tổ chức của Kohonen, đôi khi được gọi là bản đồ tự tổ chức (SOM), cung cấp một ví dụ khác về các mạng có thể học hỏi mà không cần giám sát. Có thể hình dung các nút xử lý được sắp xếp trong một mảng hai chiều, được gọi là lớp Kohonen (Hình). Ngoài ra còn có một lớp một chiều riêng biệt của các nút đầu vào, nơi mỗi nút đầu vào được kết nối với mỗi nút trong lớp Kohonen. Như trong MLP, các nơ-ron đầu vào không thực hiện quá trình xử lý nào mà chỉ chuyển các giá trị đầu vào của chúng cho các nơ-ron xử lý, nơi áp dụng trọng số.

Mạng tự tổ chức Kohonen

Giống như trong các mạng ART, mạng Kohonen học cách tập hợp các mẫu tương tự lại với nhau. Cơ chế học tập liên quan đến sự cạnh tranh giữa các tế bào thần kinh để đáp ứng với một vectơ đầu vào cụ thể. "Người chiến thắng" có trọng số của nó được đặt để tạo ra sản lượng cao, tiến gần đến 1. Trọng số trên các nơ-ron lân cận cũng được điều chỉnh để tạo ra giá trị cao, nhưng trọng số trên "kẻ thua cuộc" được để riêng. Các tế bào thần kinh ở gần người chiến thắng tạo thành một vùng lân cận.

Mạng tự tổ chức Kohonen



Mạng tự tổ chức Kohonen

Khi mạng đã huấn luyện được hiển thị với một mẫu đầu vào, một tế bào thần kinh trong lớp Kohonen sẽ tạo ra kết quả đầu ra lớn hơn các tế bào khác và được cho là đã kích hoạt. Khi một mẫu tương tự thứ hai được trình bày, cùng một nơ-ron hoặc một nơ-ron trong vùng lân cận của nó sẽ kích hoạt. Khi các mô hình tương tự gây ra kích hoạt các nơron gần về mặt cấu trúc liên kết, nên việc phân nhóm các mô hình tương tự sẽ đạt được. Hiệu quả có thể được chứng minh bằng cách huấn luyện mạng sử dụng các cặp tọa độ Descartes. Mạng được huấn luyện có đặc tính là sự phân bố của các nơ-ron kích hoạt tương ứng với các tọa độ Descartes được biểu diễn bằng vectơ đầu vào. Do đó, nếu các phần tử đầu vào nằm trong phạm vi từ –1 đến 1, thì vectơ đầu vào của (–0,9, 0,9) sẽ khiến một nơ-ron gần một góc của lớp Kohonen kích hoạt, trong khi vectơ đầu vào là (0,9, –0,9) sẽ khiến một tế bào thần kinh gần góc đối diện phát hỏa.

Mạng tự tổ chức Kohonen

Mặc dù các mạng này không được giám sát, nhưng chúng có thể tạo thành một phần của mạng kết hợp cho việc học có giám sát. Điều này đạt được bằng cách chuyển tọa độ của nơ-ron kích hoạt tới MLP. Theo cách sắp xếp này, việc học diễn ra trong hai giai đoạn khác nhau. Đầu tiên, mạng tự tổ chức Kohonen học cách liên kết các vùng trong không gian mẫu với các cụm tế bào thần kinh trong lớp Kohonen mà không cần giám sát. Thứ hai, một MLP học cách liên kết tọa độ của nơ-ron kích hoạt trong lớp Kohonen với lớp mong muốn.

Mạng hàm cơ sở xuyên tâm

Mạng chức năng cơ sở xuyên tâm (RBF) cung cấp một phương pháp học tập không giám sát thay thế khác. Chúng là các mạng chuyển tiếp, kiến trúc tổng thể tương tự như kiến trúc của một phần tử/lớp ba lớp, tức là một MLP với một lớp ẩn. Kiến trúc mạng RBF được thể hiện trong Hình. Các tế bào thần kinh đầu vào và đầu ra tương tự như của một phần tử/lớp, nhưng các tế bào thần kinh ở lớp ẩn hoàn toàn khác nhau. Các nơ-ron đầu vào không thực hiện bất kỳ quá trình xử lý nào mà chỉ đơn giản là đưa dữ liệu đầu vào vào các nút ở trên. Các nơ-ron ở lớp đầu ra tạo ra tổng trọng số của các đầu vào của chúng, thường được chuyển qua một hàm truyền tuyến tính, trái ngược với các hàm truyền phi tuyến được sử dụng với phần tử/lớp.

Mạng hàm cơ sở xuyên tâm

Các nơ-ron xử lý được xem xét cho đến nay trong chương này tạo ra một đầu ra là tổng trọng số của các đầu vào của chúng, được chuyển qua một hàm truyền. Tuy nhiên, trong mạng RBF, các nơ-ron trong lớp ẩn, đôi khi được gọi là lớp nguyên mẫu, hoạt động khác. Đối với một vectơ đầu vào (x1, x2,… xn), một nơ-ron thứ i trong lớp ẩn tạo ra một đầu ra, yi, được cho bởi:

$$y_i = f_r(r_i)$$

$$r_i = \sqrt{\sum_{j=1}^{n} (x_j - w_{ij})^2}$$

Mạng hàm cơ sở xuyên tâm

trong đó chúng ta là trọng số trên các đầu vào của nơ-ron i và fr là một hàm đối xứng được gọi là hàm cơ sở xuyên tâm (RBF). RBF được sử dụng phổ biến nhất là một hàm Gaussian:

$$f_r(r_i) = \exp\left(\frac{-r_i^2}{2\sigma_i^2}\right)$$

trong đó Vi là độ lệch chuẩn của một phân phối được mô tả bởi hàm (Hình). Mỗi nơ-ron, i, trong lớp ẩn có giá trị riêng biệt đối với Vi.

Mạng hàm cơ sở xuyên tâm

Khoảng cách Euclide giữa hai điểm là độ dài của một đoạn thẳng giữa chúng. Nếu tập hợp các trọng số (wi1, wi2,… win) trên một nơ-ron i nhất định được coi là tọa độ của một điểm trong không gian mẫu, thì ri là khoảng cách Euclide từ đó đến điểm được biểu diễn bởi vectơ đầu vào (x1, x2 ,… Xn). Trong quá trình học tập không giám sát, mạng sẽ điều chỉnh trọng số - được gọi một cách chính xác hơn là các trung tâm trong mạng RBF - để mỗi điểm (wi1, wi2,… win) đại diện cho trung tâm của một cụm điểm dữ liệu trong không gian mẫu. Tương tự, nó xác định kích thước của các cụm bằng cách điều chỉnh các biến Vi (hoặc các biến tương đương nếu sử dụng RBF khác với Gaussian). Các điểm dữ liệu trong một phạm vi nhất định, ví dụ: 2Vi, từ trung tâm cụm có thể được coi là thành viên của cụm. Do đó, cũng giống như phần tử/lớp một lớp có thể được coi là phân chia không gian mẫu hai chiều thành các đường, hoặc không gian mẫu n chiều bằng siêu mặt phẳng, vì vậy mạng RBF có thể được coi như là vẽ các vòng tròn xung quanh các cụm theo mô hình hai chiều. không gian, hoặc siêu cầu trong không gian mẫu n chiều. Một cụm như vậy có thể được xác định cho mỗi nơ-ron trong lớp ẩn. Hình cho thấy một hàm Gaussian trong không gian mẫu hai chiều, từ đó có thể thấy rằng giá trị đầu ra cố định (ví dụ: 0,5) xác định một vòng tròn trong không gian mẫu.

Mạng hàm cơ sở xuyên tâm

Quá trình học tập không giám sát trong lớp ẩn được theo sau bởi một giai đoạn học tập có giám sát riêng biệt, trong đó các tế bào thần kinh đầu ra học cách liên kết mỗi cụm với một lớp cụ thể. Bằng cách liên kết một số cụm tròn có tâm và kích thước khác nhau với một lớp duy nhất, các hình dạng tùy ý cho các vùng lớp có thể được xác định (Hình).

Tổng kết

Học số dựa trên việc điều chỉnh các tham số số để đạt được sự phù hợp chặt chẽ giữa đầu ra mong muốn và đầu ra thực tế. Mạng nơron là một loại kỹ thuật học số quan trọng. Chúng có thể được sử dụng để mô hình hóa bất kỳ ánh xạ phi tuyến nào giữa các biến và thường được sử dụng trong các nhiệm vụ phân loại. Khi được trình bày với dữ liệu nằm giữa dữ liệu đã gặp trước đó, mạng nơ-ron thường sẽ nội suy để tạo ra kết quả đầu ra giữa những dữ liệu đã tạo trước đó. Do đó, mạng nơron có thể thay thế cho logic mờ trong một số ứng dụng. Tính song song của mạng nơ-ron làm cho chúng phù hợp một cách lý tưởng với các máy tính xử lý song song.

Tổng kết

Mạng nơ-ron có thể được sử dụng để giải quyết một vấn đề theo đúng nghĩa của nó hoặc như một thành phần của một hệ thống lớn hơn. Việc thiết kế một mạng phù hợp cho một ứng dụng nhất định có thể đòi hỏi một lượng lớn các thử nghiệm và sai sót. Việc một mạng có hội tụ hay không, tức là học các trọng số phù hợp, sẽ phụ thuộc vào cấu trúc liên kết, x1 x2 Hình 8.19 Mạng RBF có thể xác định các hình dạng tùy ý cho các vùng trong hàm truyền không gian mẫu của các nút, giá trị của các tham số trong thuật toán huấn luyện và dữ liệu đào tạo. Nó thậm chí có thể phụ thuộc vào thứ tự trình bày dữ liệu đào tạo. Mặc dù mạng nơ-ron có thể giảm bớt “nút thắt thu nhận kiến thức” được mô tả trong Chương 6, chúng cũng có thể tạo ra “nút thắt thông số mạng” khi người dùng đấu tranh để định cấu hình mạng cho một vấn đề cụ thể.

Tổng kết

Một nhược điểm khác của mạng nơ-ron là lý luận của chúng không rõ ràng. Các trọng số đã học hiếm khi có thể được hiểu theo một cách có nghĩa, mặc dù về nguyên tắc, các quy tắc có thể được rút ra từ chúng. Do đó, mạng nơ-ron thường được coi như một "hộp đen" chỉ đơn giản là tạo ra một đầu ra từ một đầu vào nhất định. Tuy nhiên, bằng cách giới hạn việc sử dụng mạng nơ-ron cho các nhiệm vụ phụ trong một vấn đề, chiến lược giải quyết vấn đề tổng thể có thể vẫn rõ ràng.

Tổn kết môn học và Nội dung ôn tập

Ứng dụng và xu thế phát triển
của điều khiển thông minh