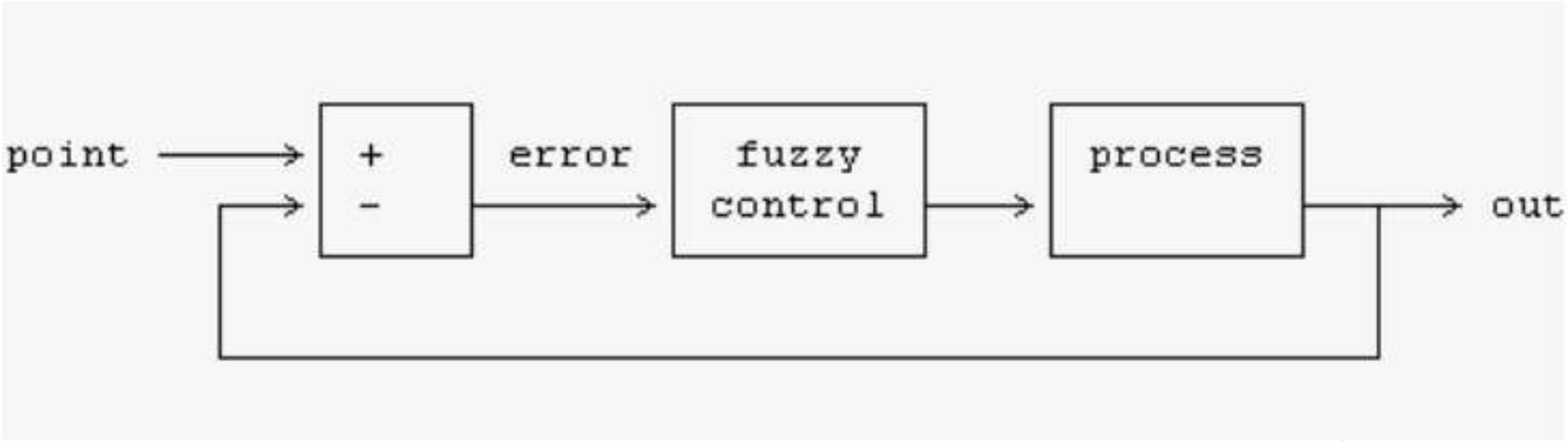
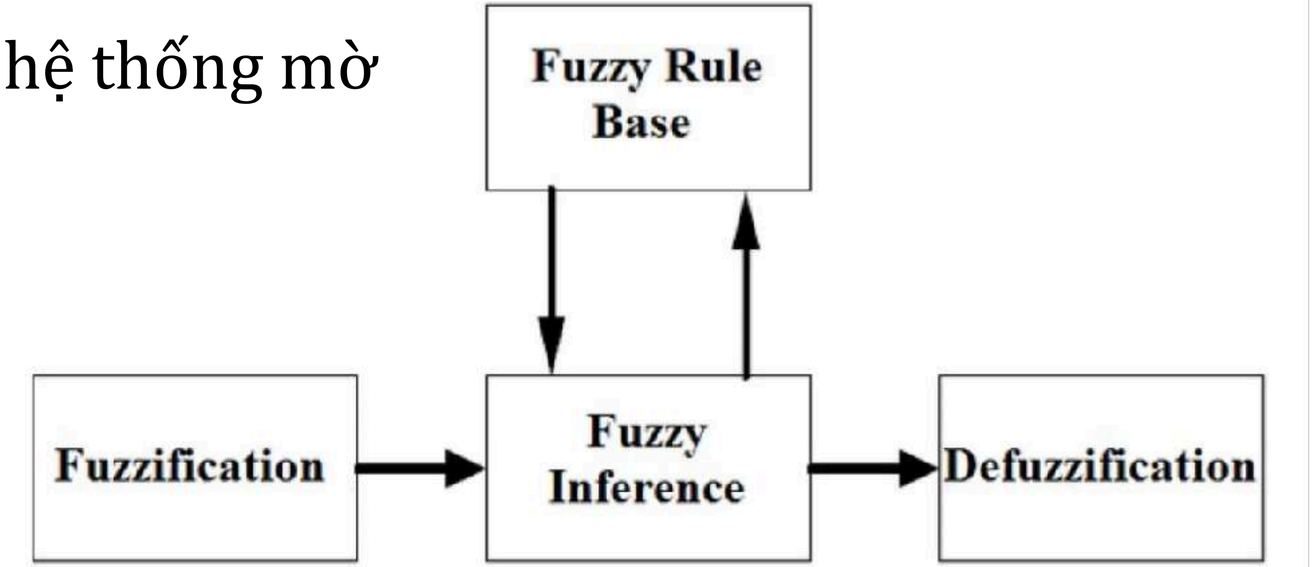
Điều khiển mờ

- Điều khiển mờ (*Fuzzy Control*) là một lĩnh vực ứng dụng được Zadeh đề cập đến như là những ứng dụng đầu tiên, rồi sau đó được cộng đồng các nhà khoa học Châu Âu phát triển.
- Thực chất điều khiển mờ là ***điều khiển dựa trên lập luận mờ*** (*Fuzzy Logic Control*).
- Nên mục tiêu của điều khiển mờ là quản lý các tiến trình theo mệnh lệnh thông qua việc tác động lên các đại lượng mô tả tiến trình *dựa trên tri thức chuyên gia* hoặc các *nhà kỹ thuật có kinh nghiệm*.
- Đó là sự khác biệt so với điều khiển thông thường chỉ dựa trên các tham số vật lý thu nhận được.

- Nguyên tắc của điều khiển mờ như hình

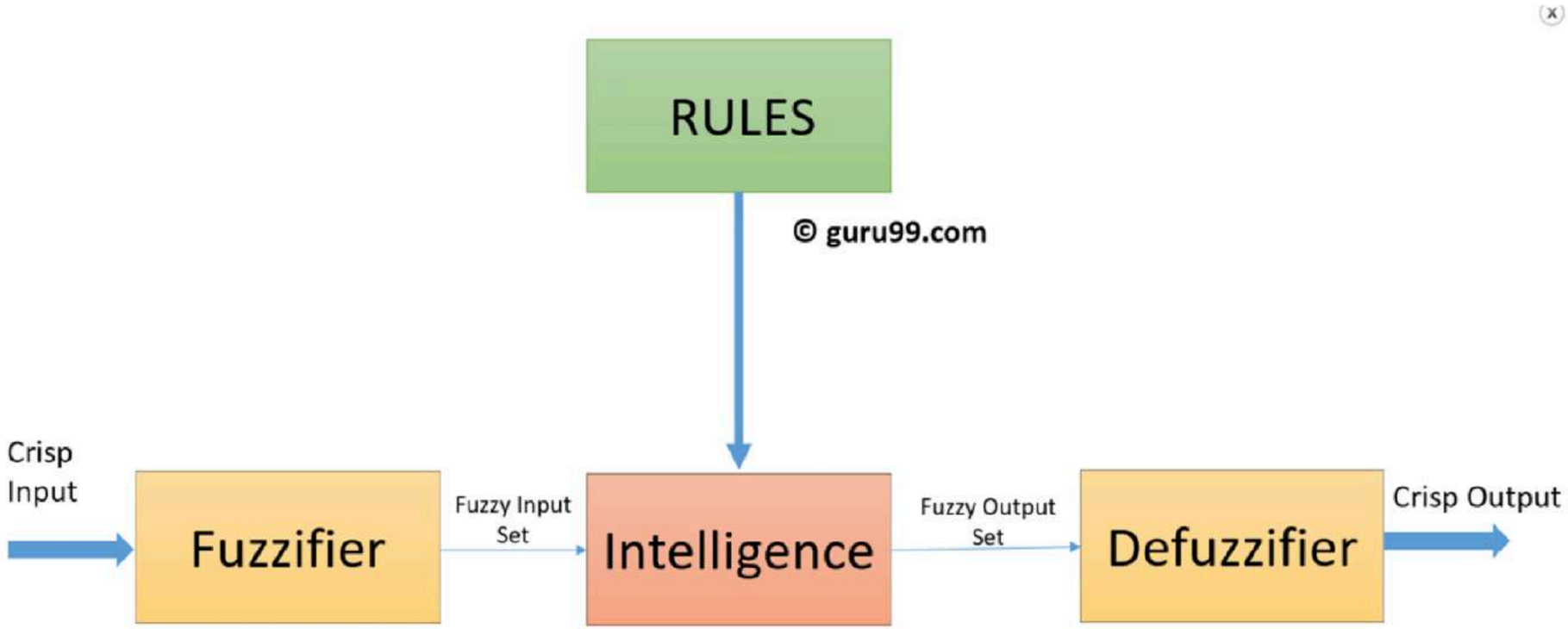


- Còn bên trong hệ thống mờ



- Như vậy, có thể nói điều khiển logic mờ là một cách tiếp cận heuristic (dò dẫm) mà có thể dễ dàng đưa kiến thức và các đặc trưng trong suy nghĩ của con người vào việc thiết kế các bộ điều khiển phức tạp
- Hệ thống này không cần phải thiết lập một mô hình toán học dưới dạng phương trình, hệ phương trình, một phiếm hàm cần tìm cực trị, v.v... để tìm thuật giải tương ứng
- Mà chỉ cần tuân thủ theo những bước như cho mọi bài toán

- Đó là: xác định các biến đầu vào và đầu ra của bộ điều khiển để mờ hoá, xây dựng các tập mờ. Tiếp theo là dùng các luật hay quy tắc điều khiển được xây dựng theo thuật ngữ của ngôn ngữ tự nhiên; rồi cuối cùng là giải mờ để tìm ra giá trị rõ.
- Sơ đồ của một hệ thống điều khiển mờ như hình



Luật mờ

- Việc suy diễn mờ dựa trên các luật mờ là vấn đề quan trọng.
- Chẳng hạn, với quy tắc ***suy diễn khẳng định*** trong lập luận cổ điển: $\big((P \Rightarrow Q) \wedge P\big) \Rightarrow Q$, trong đó:
  - Luật hoặc tri thức: $P \Rightarrow Q$ (có nghĩa $P \Rightarrow Q$ là chân lý)
  - Cùng sự kiện có thật: P (nghĩa là P đúng)
  - Thì kết luận được: Q là chân lý
- Mà luật hay tri thức thường được diễn tả bằng những thuật ngữ mang tính định tính (*luật mờ*)

- Chẳng hạn, cho $P = \{x = A\}, Q = \{y = B\}$ với luật (mờ): $P \Rightarrow Q$
- Khi có sự kiện mờ $P' = \{x = A'\}$ xác định bởi tập mờ A' trên không gian nền X và với luật mờ $P \Rightarrow Q$ thì có kết luận mờ $Q' = \{y = B'\}$ với B' là tập mờ trên không gian nền Y.
- Ví dụ: Luật mờ $P \Rightarrow Q$ là "*cánh quạt quay* ***nhanh*** *thì gió* ***nhiều***" (do có 2 khái niệm *nhanh* và *nhiều*)
  - Luật: P = "*cánh quạt quay nhanh*", Q = "*gió nhiều*"
  - Sự kiện: P = "*cánh quạt quay nhanh*"
  - Kết luận: Q = "*gió nhiều*"

- Giờ đây chỉ có sự kiện, với P' = "*cánh quạt quay* ***khá nhanh***", thì có thể kết luận Q' "*gió* ***khá nhiều***".
- Hiện thực bằng ngôn ngữ của tập hợp mờ:
  - Ở đây có quan hệ mờ $\mathfrak{R} = \{P \Rightarrow Q\}$ trên không gian nền $X \times Y$ với hàm thuộc $\mu_{\mathfrak{R}}(x, y) = \mu_{P \Rightarrow Q}(x, y) = \min\limits_{(a,b) \in X \times Y} \left\{\mu_A(a), \mu_B(b)\right\}, \forall (x, y) \in X \times Y$
  - Từ đây, với tập mờ B' trên hàm mờ (quan hệ $\mathfrak{R}$) thì $B' = A' \circ \mathfrak{R}$ với $\mu_{B'}(y) = \max\limits_{a \in X} \left\{min\left\{\mu_{A'}(a), \mu_{\mathfrak{R}}(a, y)\right\}, \forall y \in Y\right.$

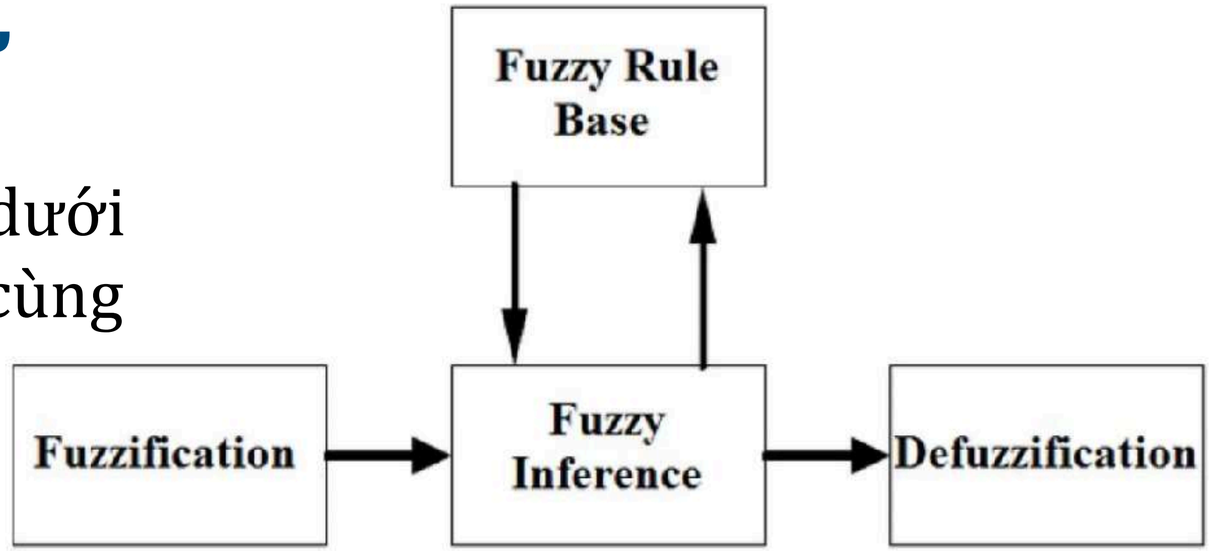
- Tương tự như vậy, với ***quy tắc phủ định***: $((P \Rightarrow Q) \wedge \neg Q) \Rightarrow \neg P$ với sự kiện mờ $\neg Q$ có kết luận mờ $\neg P$.
- Nên với sự kiện mờ $\neg Q = \{y = \neg B'\}$, cùng với luật mờ $P \Rightarrow Q$, thì có kết luận là $\neg P = \{x = \neg A'\}$
- Chẳng hạn, cũng với luật như trên: $P \Rightarrow Q$ là "*cánh quạt quay nhanh thì gió nhiều*", và với sự kiện "*gió không nhiều lắm*" thì kết luận là "*cánh quạt quay không nhanh lắm*"

Bài toán điều khiển mờ

- Để giải bài toán mờ, các lập luận mờ dưới dạng các luật mờ thường liên kết lại cùng nhau qua phép toán luận lý.
- Chẳng hạn, có các luật

$$x = A_1 \Rightarrow y = B_1$$
$$x = A_2 \Rightarrow y = B_2$$
$$\dots$$
$$x = A_n \Rightarrow y = B_n$$

- Trong đó A_i, B_i là các tập mờ trên không gian nền X_i, Y_i tương ứng.



- Mỗi luật xác định một quan hệ kéo theo ($\Rightarrow$) giữa 2 tập $\{A_i, B_i\}$ tương ứng
- Từ đây có quan hệ $\mathfrak{R}(A_i, B_i)$ mờ đại diện cho một một luật.

- **Tính chất 9**: Các tập mờ A_i trong tiền đề của luật được đặc trưng bởi độ thuộc μ_{A_i}, khi tiền đề của luật gồm nhiều thành phần kết hợp lại theo kiểu gom lại của tích Descartes $A = A_1 \times A_2 \times \ldots \times A_n$, thì độ thuộc của A là $\mu_A(x) = \min\left\{\mu_{A_1}(x_1), \mu_{A_2}(x_2), \ldots, \mu_{A_k n}(x_n)\right\}$
- Chẳng hạn, khi đánh giá một viên đá quý, có luật mờ (*Fuzzy rule*) như:
  - Trọng lượng cao, kích thước lớn và tinh khiết thì giá cao.
  - Giả sử có 3 tập mờ A_n, B_l, C_t, G_c tương ứng với trọng lượng nặng, kích thước lớn, tinh khiết và giá cao. Thì $\mu_{\Re}(x) = min\left\{\mu_{A_c}(x_1), \mu_{B_l}(x_2), \mu_{C_t}(x_3)\right\}$
  - Từ đó dùng $\mu_{\Re}$ là căn cứ như là trọng số của $\mu_{G_c}(x)$ để tính toán

Giải bài toán điều khiển mờ dùng skfuzzy

- Tổng kết lại, có các bước chi tiết sau đây:
  - Xác định miền xác định của các hàm thuộc (là không gian nền)
  - Xây dựng những tập mờ có trong tiền đề của các luật mờ
  - Đưa luật mờ vào
  - Nhập giá trị như là dữ liệu nhập đề suy đoán kết quả đầu ra có trong các kết luận của luật mờ
  - Tính các trọng số dựa vào những luật mờ
  - Thiết lập hàm thành viên về giá trị cần suy đoán
  - Tính giá trị cần suy đoán

Hiện thực qua thư viện *Scikit-Fuzzy*

- Trong thư việc Scikit-Fuzzy có những module rất thuận lợi cho việc xử lý, chúng ta chỉ cần tuần tự theo các bước sau đây là có được một hệ thống hoàn chỉnh.
  - **Bước 1**: Xây dựng không gian nền tương ứng (dùng mảng *numpy.array*([]) để lưu trữ).
  - **Bước 2**: Xác định đâu là tập tiền đề đâu là tập kết luận qua hàm trong module *control*: *fuzzy.control.Antecendent*([],_), *fuzzy.control.Consequent*([],_)
  - **Bước 3**: Có thể dùng *numpy.arange*(_,_,_) hay *numpy.array*([]) để tạo tập mờ, hoặc dùng một trong các hàm thành viên như *fuzzy.trimf*([],[]), *fuzzy.trapmf*([],[]), *fuzzy.gbellmf*([],[]), v.v...

- **Bước 4**: Dùng để đưa các luật *rules* vào bằng *fuzzy.control.Rule*(.,.)
- **Bước 5**: Tạo mô hình (*model*) suy diễn với *fuzzy.control.ControlSystemSimulation*(*fuzzy.control.ControlSystem*(*rules*))
- **Bước 6**: Nhập dữ liệu như là sự kiện cần suy đoán thông qua phương thức *input*() từ *model* ở trên, rồi dùng phương thức *compute*() để xử lý sau đó xuất kết quả bằng phương thức *output*()