CS4386 AI Game Programming

Lecture 09
Fuzzy Logic

Semester B, 2020-2021
Department of Computer Science
City University of Hong Kong

Motivation for Fuzzy Logic

- Consider an AI for a character moving through a dangerous environment
- In a finite state machine approach, we could choose two states:
  1. Cautious: the character sneaks slowly along, keeping an eye out for trouble
  2. Confident: the character walks normally
  - As the character moves through the level, it will switch between the two states which may appear odd
  - We might think of the character getting gradually braver, but the change occurs suddenly
- Fuzzy logic allows us to blur the line between cautious and confident, giving us a whole spectrum of confidence levels
  - With fuzzy logic we can still make decisions like "walk slowly when cautious" but both "slowly" and "cautious" can include a range of degrees

Traditional Logic

- For example, a character may be hungry or not hungry; it may be hurt or not hurt
- In other words, the character may belong to the set hungry, or not belong to this set; it may belong to the set hurt, or not belong to this set

Fuzzy Logic

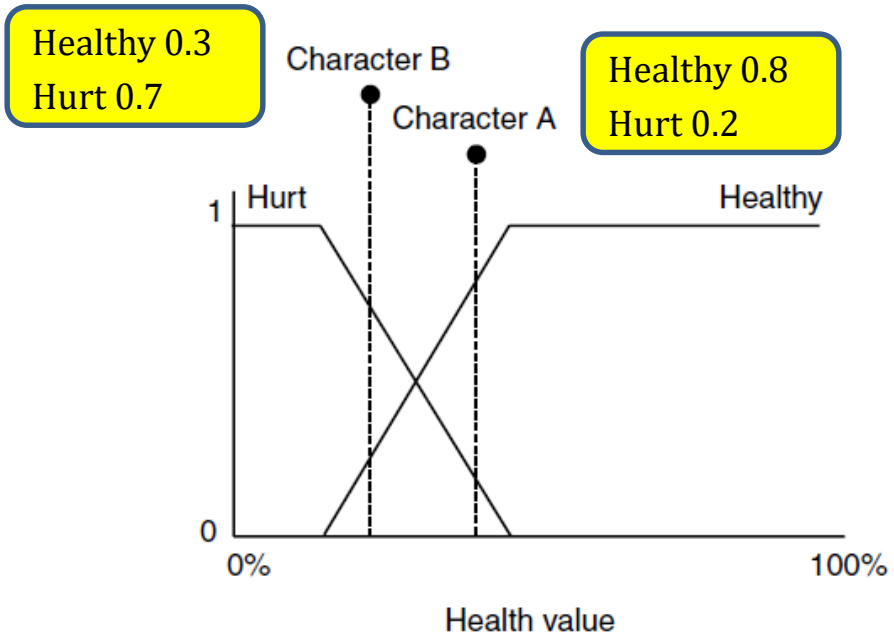
- Rather than belonging to a set or being excluded from it, everything can partially belong to the set, and some things can belong to more than others
  - A character can be hurt with a value of 0.5, or hungry with a value of 0.9
  - A character with a hurt value of 0.7 will be more hurt than one with a value of 0.3
  - Here hurt and hungry are fuzzy sets and the numeric value is known as degree of membership
- A common mistake is to interpret the value as a probability or a percentage
  - the results of applying fuzzy logic techniques will rarely be the same as if you applied probability techniques

Membership of Multiple Sets

- Anything can be a member of multiple sets at the same time
  - E.g., a character may be both hungry and hurt, which is the same for both classical and fuzzy sets
- In traditional logic some sets are mutually exclusive
  - E.g., a character cannot be both hurt and healthy
- In fuzzy logic, this is no longer the case
  - A character can be hurt and healthy, it can be tall and short, and it can be confident and curious
  - The character will simply have different degrees of membership for each set (e.g., it may be 0.5 hurt and 0.5 healthy)
  - The fuzzy equivalent of mutual exclusion is the requirement that membership degrees sum to 1

Fuzzification (1)

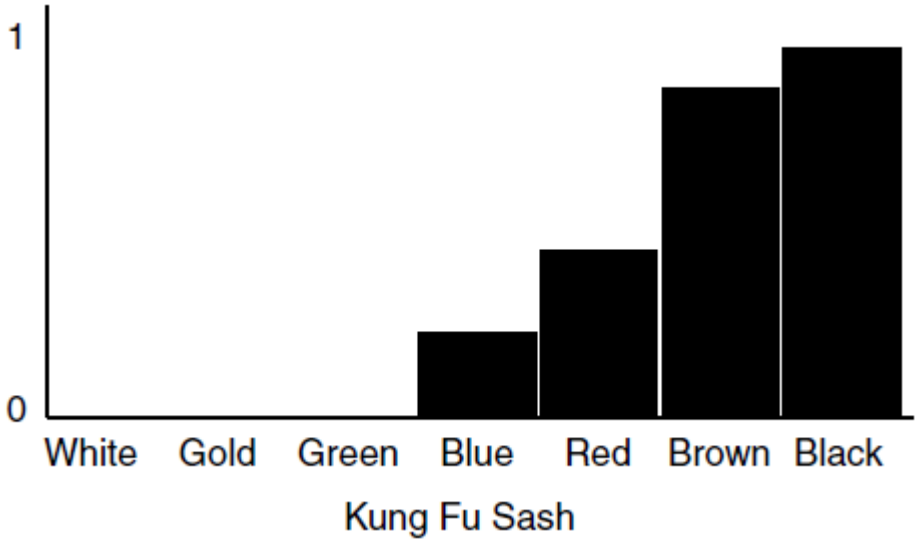
- Fuzzification is the process of turning regular data into degrees of membership
- Numeric fuzzification
  - turning a numeric value into the membership of one or more fuzzy sets with a membership function



Fuzzification (2)

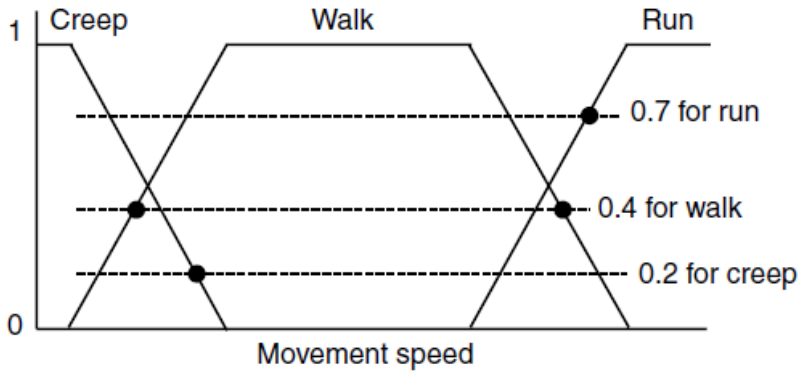
- Fuzzification of other data types
  - If the fuzzy set corresponds directly to the Boolean value (if the fuzzy set is "possession of powerful artifact," for example), then the membership values will be 0 and 1
  - For enumerated values, where there are more than two options: each possible value has a pre-determined stored membership value

Degree of membership for the fuzzy set "fearsome fighter"



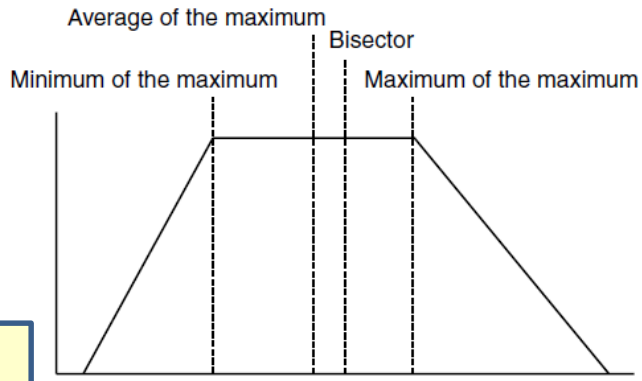
Defuzzification

- Defuzzification is the process to turn a set of membership values for fuzzy sets back into useful data
  - Defuzzification involves turning a set of membership values into a single output value which is almost always a number
  - We are trying to reverse the fuzzification method: to find an output value that would lead to the membership values we know we have
  - It is often not directly possible, e.g., membership values of 0.2, 0.4, and 0.7 for the fuzzy sets "creep," "walk," and "run", no possible speed value that can give these membership values



Defuzzification: Using the Highest Membership

- We can simply choose the fuzzy set that has the greatest degree of membership and choose an output value based on that
  - In the example on previous slide, the "run" membership value is 0.7, so we could choose a speed that is representative of running
  - Four commonly chosen points as the output value:
    1. the minimum value at which the function returns 1 (i.e., the smallest value that would give a value of 1 for membership of the set)
    2. the maximum value
    3. the average of 1. and 2.
    4. The bissector of the function, calculated by integrating the area under the curve of the membership function and choosing the point which bisects this area (can be pre-computed offline)



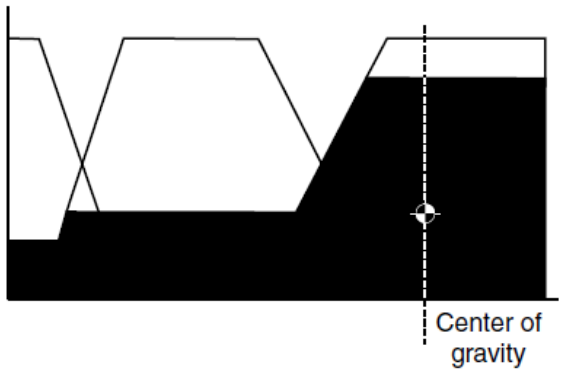
Pro: fast technique and simple to implement
Con: only provides a coarse defuzzification, e.g., 0 creep, 0 walk, 1 run will have exactly the same output speed as 0.33 creep, 0.33 walk, 0.34 run

Defuzzification: Blending Based on Membership

- A simple way around this limitation is to blend each characteristic point based on its corresponding degree of membership
  - A character with 0 creep, 0 walk, 1 run will use the characteristic speed for the run set (calculated in any of the ways we saw above: minimum, maximum, bisector, or average)
  - A character with 0.33 creep, 0.33 walk, 0.34 run will have a speed given by (0.33 * characteristic creep speed) + (0.33 * characteristic walk speed) + (0.34 * characteristic run speed)

Defuzzification: Center of Gravity

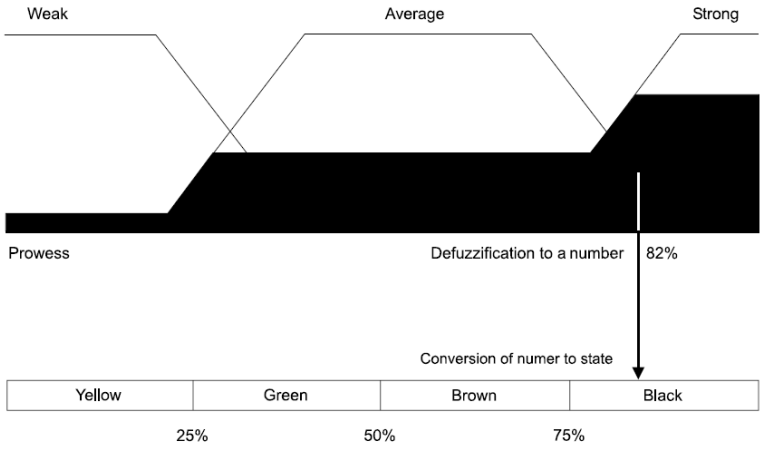
- This technique is also known as centroid of area
- This method takes into account all the membership values, rather than just the largest
  - First, each membership function is cropped at the membership value for its corresponding set, e.g., if a character has a run membership of 0.4, the membership function is cropped above 0.4
  - The center of mass of the cropped regions is then found by integrating each in turn and this point is used as the output value



- Unlike the bisector of area method, we cannot do the integration offline because we do not know in advance what level each function will be cropped at
  - The resulting integration (often numeric, unless the membership function has a known integral) can take time

Defuzzification to an Enumerated Value

- The method for defuzzifying an enumerated value depends on whether the different enumerations form a series or if they are independent categories
- Enumerations that can be ordered are often defuzzified as a numerical value
  - Each of the enumerated values corresponds to a non-overlapping range of numbers
  - The defuzzification is carried out exactly as for any other numerical output, and then an additional step places the output into its appropriate range, turning it into one of the enumerated options



- Enumerations that cannot be ordered are usually defuzzified by making sure a fuzzy set corresponds to each possible option
  - There may be a fuzzy set for “eat,” another for “sleep,” and another for “watch movie”
  - The set that has the highest membership value is chosen, and its corresponding enumerated value is output

Defuzzification to a Boolean Value

- To arrive at a Boolean output, we use a single fuzzy set and a cut-off value
- If the degree of membership for the set is less than the cut-off value, the output is considered to be false; otherwise, it is considered to be true
- If several fuzzy sets need to contribute to the decision, then they are usually combined using a fuzzy rule into a single set, which can then be defuzzified to the output Boolean

Combining Facts – Traditional Logic

- Under traditional logic, logical operators (such as AND, OR, and NOT) are used to combine the truth of simple facts to understand the truth of complex facts
  - If we know the two separate facts "it is raining" and "it is cold," then we know the statement "it is raining and cold" is also true
- In traditional logic we use a truth table, which tells us what the truth of a compound statement is based on the different possible truth values of its constituents
  - AND is represented as:

A	*B*	*A* AND *B*
false	false	false
false	true	false
true	false	false
true	true	true

Combining Facts – Fuzzy Logic (1)

- Unlike traditional logic, now each simple fact is not true or false, but is a numerical value — the degree of membership of its corresponding fuzzy set
  - It might be partially raining (membership of 0.5) and slightly cold (membership of 0.2)
  - We need to be able to work out the truth value for compound statements such as "it is raining and cold"
- The fuzzy logic for AND is $m_{(A \text{ AND } B)} = \min(m_A, m_B)$
  where m_A is the degree of membership of set A (i.e., the truth value of A) and m_B is the degree of membership of set B

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

This fuzzy rule is also applicable to the truth table of AND under traditional logic

Combining Facts – Fuzzy Logic (2)

- The fuzzy logic operators:

Expression	Equivalent	Fuzzy Equation
NOT A		$1 - m_A$
A AND B		$\min(m_A, m_B)$
A OR B		$\max(m_A, m_B)$
A NOR B	NOT(A OR B)	$1 - \max(m_A, m_B)$
A NAND B	NOT(A AND B)	$1 - \min(m_A, m_B)$

- The fuzzy logic corresponding to some traditional logic is still applicable, e.g., De Morgan's Law

$$A \text{ OR } B = \text{NOT}(\text{NOT } A \text{ AND NOT } B)$$

- How about the fuzzy equation for A XOR B ?

Fuzzy Rule Example

- Fuzzy rules relate the known membership of certain fuzzy sets to generate new membership values for other fuzzy sets
- For example, "If we are close to the corner and we are traveling fast, then we should brake"
- This rule relates two input sets: "close to the corner" and "traveling fast" and determines the degree of membership of the third set, "should brake"

$$m_{(\text{Should Brake})} = \min(m_{(\text{Close to the Corner})}, m_{(\text{Traveling Quickly})})$$

- If we knew that we were "close to the corner" with a membership of 0.6 and "traveling fast" with a membership of 0.9, then we would know that our membership of "should brake" is 0.6

Fuzzy Logic Decision Making (1)

- In many problems a set of different actions can be carried out, but it isn't always clear which one is best
  - Often, the extremes are very easy to call, but there are gray areas in the middle
  - It is particularly difficult to design a solution when the set of actions is not on/off but can be applied with some degree
- For example, the actions available for driving the car include steering and speed control (acceleration and braking), both of which can be done to a range of degrees
  - It is possible to brake sharply to a halt or simply tap the brake to shed some speed
  - It is clear on some extreme cases what action should be taken:
    - If the car is traveling headlong at high speed into a tight corner, then it is pretty clear we should brake
    - If the car is out of a corner at the start of a long straightaway, then we would like to accelerate
  - However, exactly when to brake and how hard to hit the pedal are gray areas that differentiate the great drivers from the mediocre

Fuzzy Logic Decision Making (2)

- A fuzzy logic decision maker should help to represent the gray areas
  - We can use fuzzy rules written to cope with the extreme situations
  - These rules should generate sensible (although not necessarily optimal) conclusions about which action is best in any situation
- Here we will consider decision making structure that uses only rules involving the fuzzy logic AND operator
  - The set of crisp inputs is mapped into lots of states, which can be arranged in mutually inclusive groups
  - We also have a set of output states. These output states are normal fuzzy states, representing the different possible actions that the character can take
  - Linking the input and output states is a set of fuzzy rules. Typically, rules have the structure:

    input 1 state AND . . . AND input n state THEN output state

Fuzzy Logic Decision Making (3)

- To generate the output, we go through each rule and calculate the degree of membership for the output state
  - This is simply a matter of taking the minimum degree of membership for the input states in that rule (since they are combined using AND)
  - The final degree of membership for each output state will be the maximum output from any of the applicable rules
- For example, with 2 inputs (corner position and speed) where each input has 2 possible states

Rule Block

corner-entry AND going-fast THEN brake
corner-exit AND going-fast THEN accelerate
corner-entry AND going-slow THEN accelerate
corner-exit AND going-slow THEN accelerate

Degrees of Membership

Corner-entry: 0.1
Corner-exit: 0.9
Going-fast: 0.4
Going-slow: 0.6

Results of Each Rule

$\text{Brake} = \min(0.1, 0.4) = 0.1$
$\text{Accelerate} = \min(0.9, 0.4) = 0.4$
$\text{Accelerate} = \min(0.1, 0.6) = 0.1$
$\text{Accelerate} = \min(0.9, 0.6) = 0.6$

Weakness

- Lack of scalability
  - It works well for a small number of input variables and a small number of states per variable
  - To process a system with 10 input variables, each with 5 states, would require almost 10 million rules
  - This is well beyond the ability of anyone to create
- For larger systems of this kind, we can either use a small number of general fuzzy rules, or we can use the Combs method for creating rules, where the number of rules scales linearly with the number of input states

Combs Method (1)

- The Combs method relies on a simple result from classical logic: a rule of the form

  x AND y ENTAILS z

  can be expressed as:

  (x ENTAILS z) OR (y ENTAILS z)

  where ENTAILS is a Boolean operator with the following truth table

a	b	a ENTAILS b
true	true	true
true	false	false
false	true	true
false	false	true

Combs Method (2)

- This is equivalent of saying that

  IF a AND b THEN c

  can be expressed as:

  (IF a THEN c) or (IF b THEN c)

  we can then split it into 2 separate rules:

  IF a THEN c

  IF b THEN c

- More generally

  IF a_1 AND … AND a_n THEN c

  can be expressed as:

  IF a_1 THEN c

  ⋮

  IF a_n THEN c

Combs Method (3)

- If rules can always be decomposed into this form, then why bother with the block format rules at all?
- Look at the following example with 2 block format rules:
  - Block format rule 1

    IF corner-entry AND going-fast THEN brake

    can be expressed as:

    IF corner-entry THEN brake

    IF going-fast THEN brake
  - Block format rule 2

    IF corner-exit AND going-fast THEN accelerate

    can be expressed as:

    IF corner-exit THEN accelerate

    IF going-fast THEN accelerate

Inconsistent actions when the 2 block format rules are considered at the same time

Combs Method (4)

- When there is more than 1 block rule, we cannot obtain the general set of rules automatically through decomposition (unless by coincidence that the results do not have inconsistency)
- The Combs method instead starts from scratch
  - the fuzzy logic designers build up rules, limiting themselves to the Combs format only
  - The overall sophistication of the fuzzy logic system will inevitably be limited, but the tractability of creating the rules means they can be tweaked more easily

Combs Method (5)

- For example, in block format

  corner-entry AND going-fast THEN brake
  corner-exit AND going-fast THEN accelerate
  corner-entry AND going-slow THEN accelerate
  corner-exit AND going-slow THEN accelerate

  could be expressed as

  corner-entry THEN brake
  corner-exit THEN accelerate
  going-fast THEN brake
  going-slow THEN accelerate

  with inputs

  Corner-entry: 0.1
  Corner-exit: 0.9
  Going-fast: 0.4
  Going-slow: 0.6

Brake = 0.1
Accelerate = 0.6

Brake = 0.4
Accelerate = 0.9

When both sets of results are defuzzified, they are both likely to lead to a modest acceleration

Combs Method (6)

- The Combs method is surprisingly practical in fuzzy logic systems
  - If the Combs method were used in classical logic (building conditions for state transitions, for example), it would end up hopelessly restrictive
  - But, in fuzzy logic, multiple fuzzy states can be active at the same time, and this means they can interact with one another (we can both brake and accelerate, for example, but the overall speed change depends on the degree of membership of both states)
  - This interaction means that the Combs method produces rules that are still capable of producing interaction effects between states, even though those interactions are no longer explicit in the rules

Fuzzy State Machines (1)

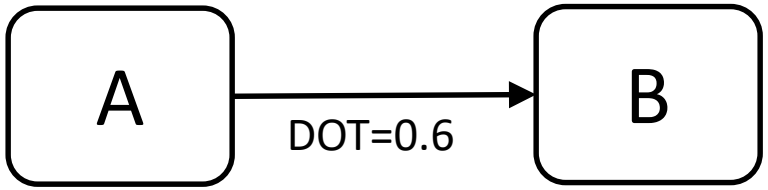
- There may be difference when various developers consider fuzzy state machines
  - A fuzzy state machine can be any state machine with some element of fuzziness
  - It can have transitions that use fuzzy logic to trigger, or it might use fuzzy states rather than conventional states. It could even do both
- Here we will look at a simple state machine with fuzzy states, but with crisp triggers for transitions
  - a state machine that can sensibly handle state transitions while allowing a character to be in multiple states at the same time

Fuzzy State Machines (2)

- Each state has its own degree of membership (DOM)
- The currently active states have a DOM of greater than 0
- At each iteration of the state machine, the transitions belonging to all active states are given the chance to trigger
  - This means that multiple transitions can happen in one iteration
  - This is essential for keeping the fuzziness of the machine

Fuzzy State Machines (3)

- If a transition fires, it can transition to any number of new states
  - The transition itself also has an associated degree of transition
  - The DOM of the target state is given by the DOM of the current state ANDed with the degree of transition (DOT)
- For example

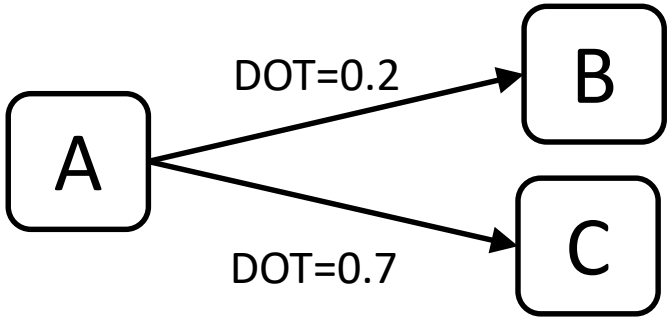


Assume that at time t,
M_A=0.4
M_B=0.3

At time t+1,
$M'_B = M_{B \text{ OR } (A \text{ AND } T)}$ = max(0.3, min(0.4, 0.6)) = 0.4
$M'_A = M_{A \text{ AND NOT } T}$ = min(0.4, 1-0.6) = 0.4

Fuzzy State Machines (4)

- Another example with multiple degrees of transition



Assume that at time t,
M_A=0.5
M_B=0.6
M_C=0.4

At time t+1,
$M'_B = M_{B \text{ OR } (A \text{ AND } T[A \text{ to } B])} = \max(0.6, \min(0.5, 0.2)) = 0.6$
$M'_C = M_{C \text{ OR } (A \text{ AND } T[A \text{ to } C])} = \max(0.4, \min(0.5, 0.7)) = 0.5$
$M'_A = M_{A \text{ AND NOT } (T[A \text{ to } B] \text{ OR } T[A \text{ to } C])} = \min(0.5, 1-\max(0.2,0.7)) = 0.3$

Reference

- Artificial Intelligence for Games
  - Chapter 5.5