Fuzzy Logic

- “the essence of fuzzy logic is that everything is a matter of degree”
- *Imprecision in data*…

  and *uncertainty in solving problems*
- Fuzzy logic vs. Boolean logic
  - FL uses 50%-80% less rules than traditional BL rule-based systems, to accomplish identical tasks
- Discredited by academic AI: better to use probability to represent any kind of uncertainty

Fuzzy Logic in Games

- **Example of uses:**
  - **To control movement of bots/NPCs (to *smooth out movements based on imprecise target areas*)**
  - **To *assess threats posed by players* (to make further strategic decisions)**
  - **To *classify player and NPCs in terms of some useful game information* (such as health level, combat ability, or defensive prowess)**

Degrees of Belonging

- **Fuzzy set** (a predicate or description of something, with degree value)
- Fuzzy logic ***gives a predicate a degree value***.
- Instead of belonging, or not, to a set of having a particular predicate (1 or 0, Boolean logic), ***everything can partially belong to a set***, and **some things belong more in the fuzzy set than others**
- A character with **a hurt value of 0.7 will be more hurt than one with a value of 0.3**.

Fuzzy Sets

- Fuzzy sets – the numeric value is called the *degree of membership* (these values are **NOT probability values!**)
- For each fuzzy set, a degree of membership of 1 given to something *completely in the set*. Degree membership of 0 given to something ***completely outside the fuzzy set***
- Typical to use integers in implementation instead of floating-point values (between 0 and 1), *for fast computation in game*
- **Note:** *Anything can be a member of multiple fuzzy sets at the same time*

Fuzzy Control / Inference Process

- **3 basic steps** in a fuzzy control or fuzzy inference process

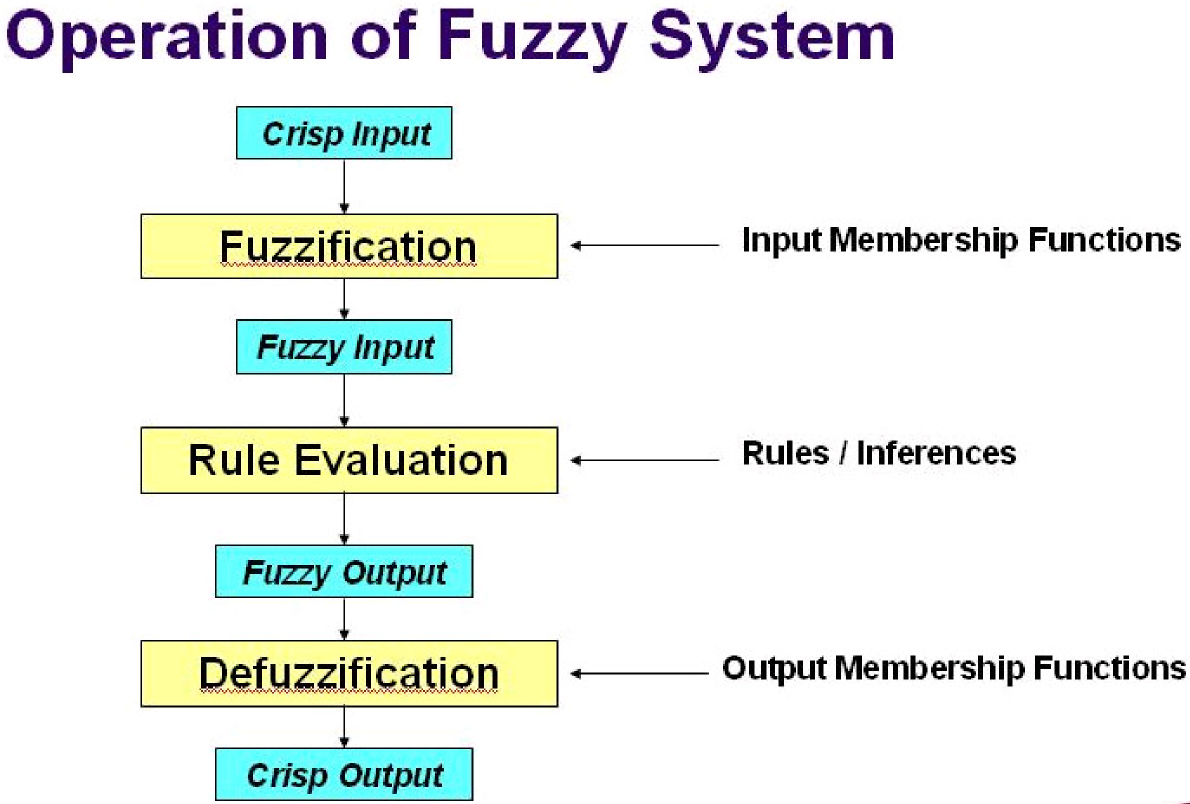
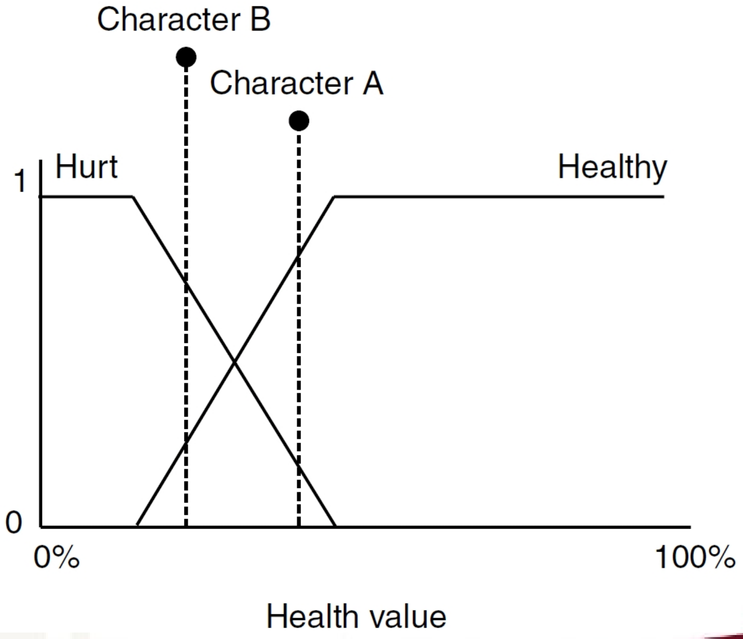
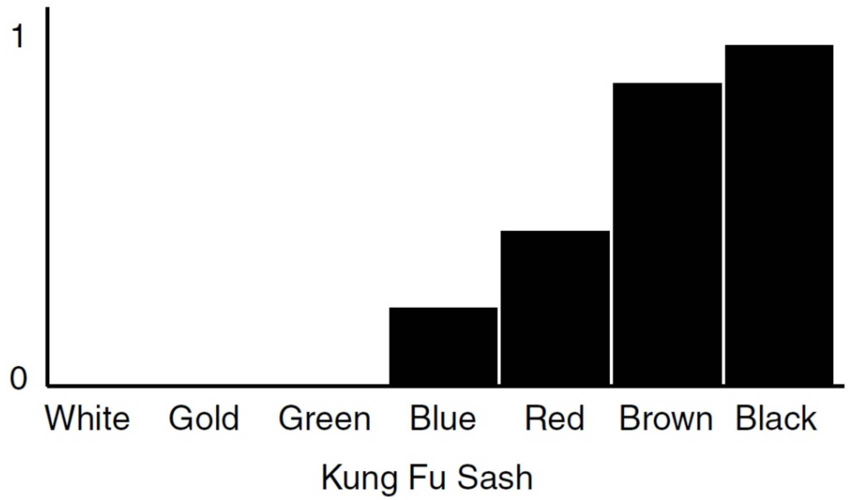


Figure from: www.binoybnair.blogspot.com/2007/03/fuzzification-and-defuzzification.html

Step 1 - Fuzzification

- **Mapping Process – converts *crisp input* (real numbers) *to fuzzy input* (degree of membership)**
- **E.g.: To find the *degree to which a character is hurt or healthy*, or their degree of membership in the "fearsome fighter" fuzzy set.**





Membership Functions

- **<u>Membership functions</u>** ***map input variables to a degree of membership***, **in a fuzzy set between 0 and 1**
- **<u>Any function can be used</u>, and the <u>shape usually is governed by</u>** ***desired accuracy***, **the** ***nature of problem***, **or** ***ease of implementation***.
- **Boolean logic membership function (m/f)**

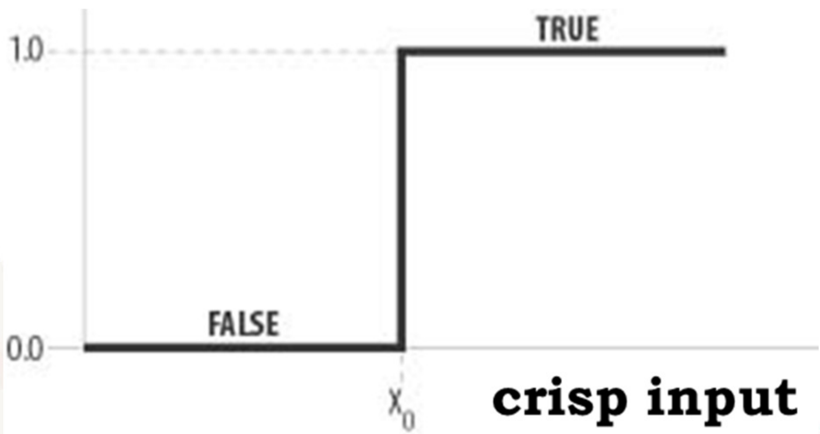
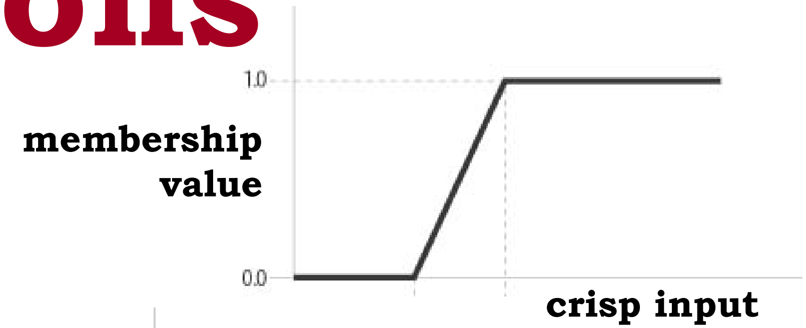
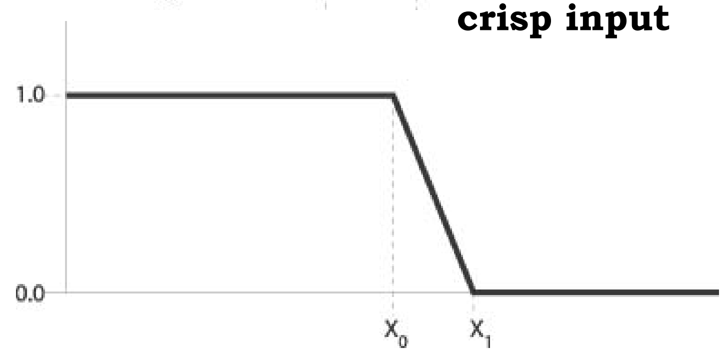


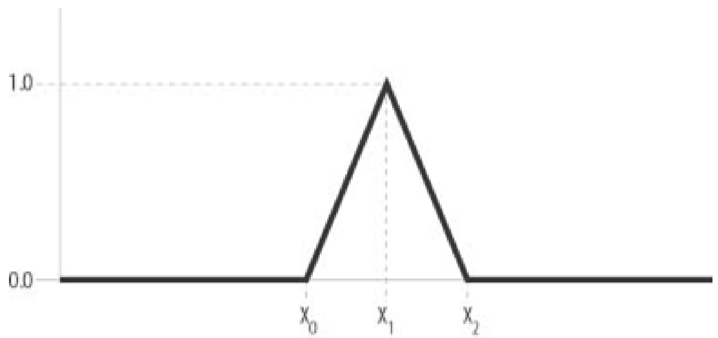
Figure 10-2 from AI for Game Development, David M. Bourg, Glenn Seeman

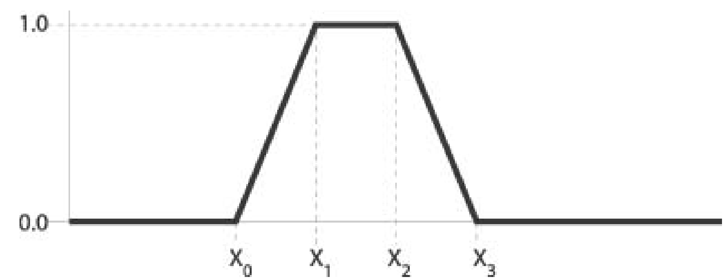
Common Membership Functions

- ❑ *Grade* **m/f**
- ❑ *Reverse* **grade m/f**
- ❑ *Triangular* **m/f**
- ❑ *Trapezoid* **m/f**









Figures 10-5 to 10-7 from AI for Game Development, David M. Bourg, Glenn Seeman

Membership Functions

- E.g., to find *the degree to which a person is* *underweight*, *overweight* or at ideal weight, a set of membership functions may be used to represent a person's weight

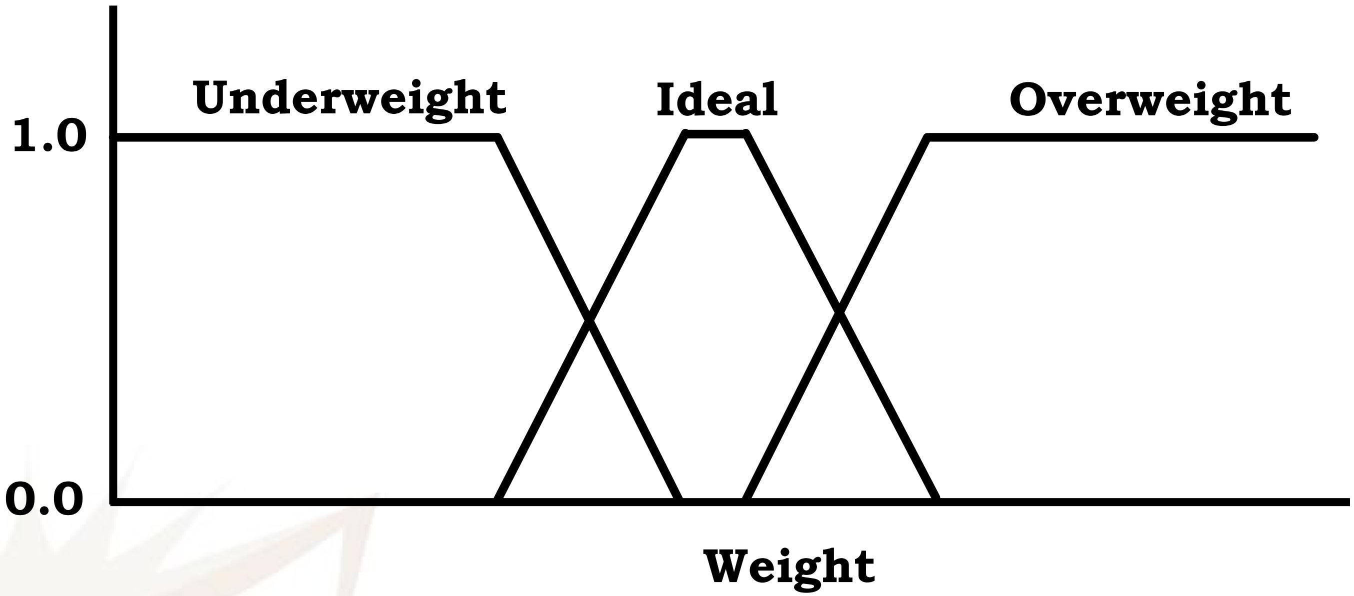


Figure 10-4 from AI for Game Development, David M. Bourg, Glenn Seeman

Step 2 – Fuzzy Rule Base

- Once all inputs are expressed in fuzzy set membership, combine them *using logical fuzzy rules* to determine degree to which each rule is true (*i.e.*, degree of membership for an output fuzzy set)
- E.g., given a person's weight and activity level as input, define rules to make a health decision
  - If overweight AND NOT active then *frequent exercise*
  - If overweight AND active then *moderate diet*

Combining Facts

- In traditional logic we use a truth table

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

- To apply usual logical operators to fuzzy input, we need the following *basic fuzzy axioms*:
- A OR B = MAX(A, B)
- A AND B = MIN(A, B)
- NOT A = 1 – A

Fuzzy Rules

- **Earlier example on weight (and now, including height)**

overweight AND tall = MIN(0.7, 0.3) = 0.3

overweight OR tall = MAX(0.7, 0.3) = 0.7

NOT overweight = 1 – 0.7 = 0.3

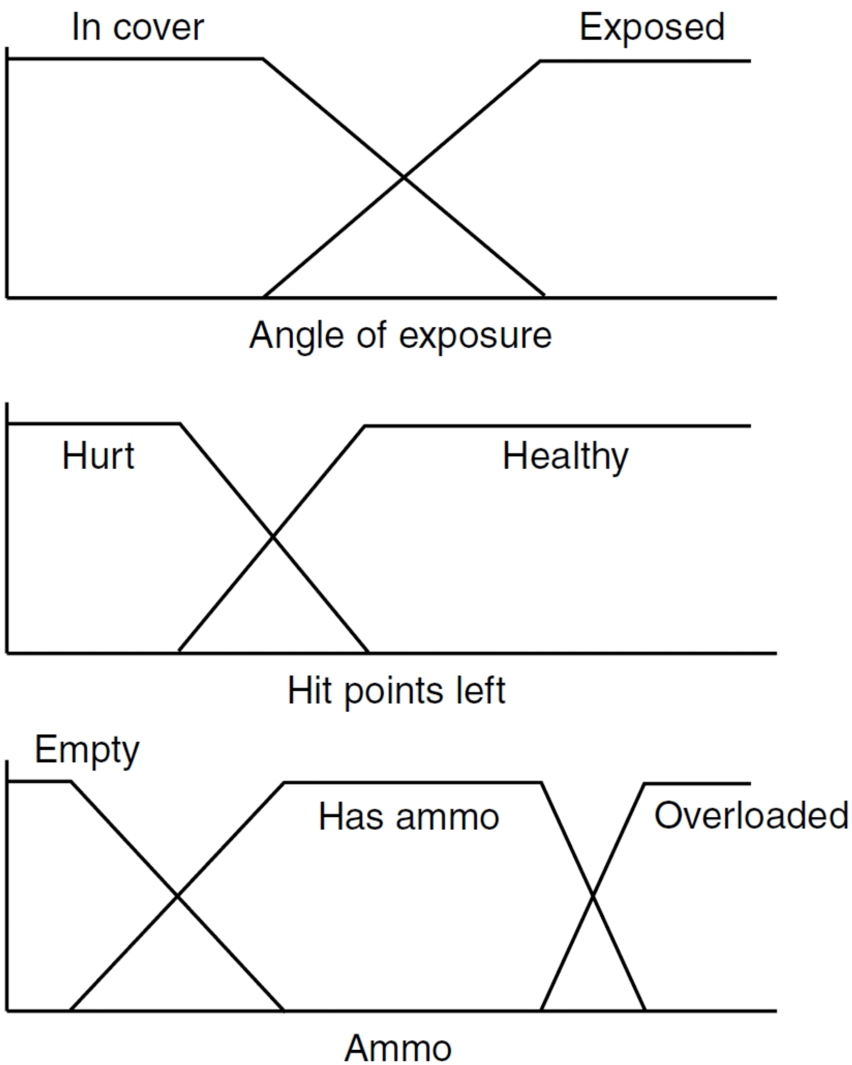
NOT tall = 1 – 0.3 = 0.7

NOT (overweight AND tall) = 1 – MIN(0.7, 0.3) = 0.7

- **Note that these fuzzy axioms (AND, OR, NOT) are not the only definition of the logical operators. There are *other definitions that can be used*.**

Université Concordia University

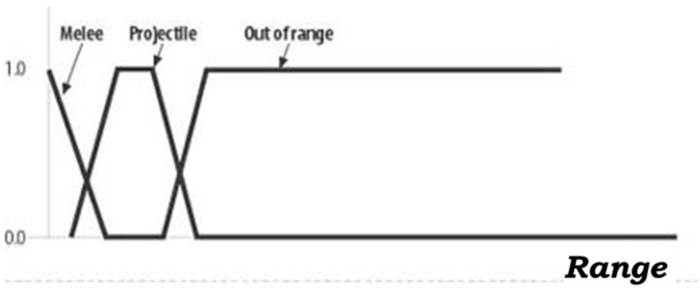
Fuzzy Rules

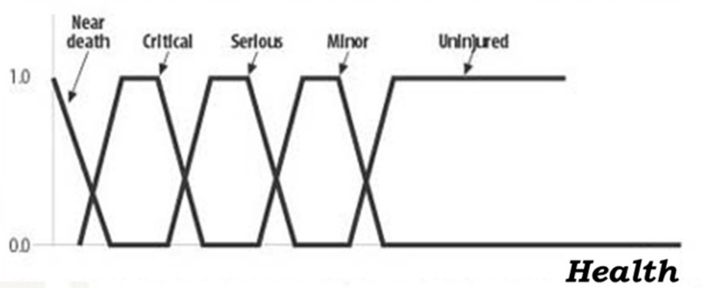


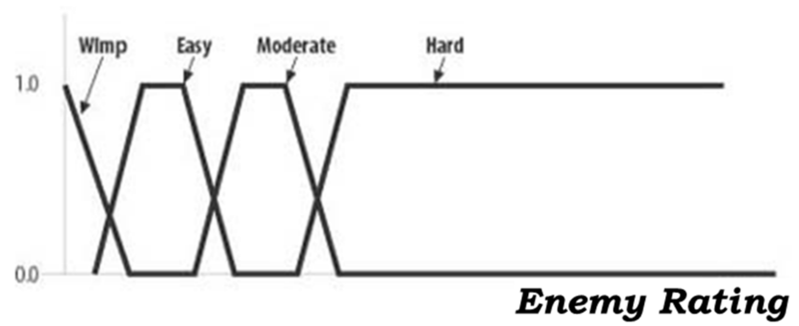
- **With the m/f to the left, for each input variable, *common requirement* is to *construct a complete set of* *all possible combinations* of inputs. In this case, we need 12 rules (2x2x3)**

Rule Evaluation (Creature example)

- **We have an AI fuzzy decision making system, which needs to evaluate *whether a creature should attack the player*.**
  Input variables: *range*, *health*, *opponent rating*







Rule Evaluation

- **Rule base:**
  - **If (in *melee range* AND *uninjured*) AND NOT *hard* (*enemy rating*) then attack**
  - **If (NOT in *melee range*) AND *uninjured* then do nothing**
  - **If (NOT *out of range* AND NOT *uninjured*) AND (NOT *wimp*) then flee**
- **Given specific degrees for the input variables, we might get outputs (membership value) that are:**
  - **Attack degree: 0.2**
  - **Do nothing degree: 0.4**
  - **Flee degree: 0.7**

Université Concordia University

Misc.: Dealing with Complex Rule Base

- We may have multiple rules in our rule base that *will results in the same output membership fuzzy set*.
- E.g.
  - *Corner-entry* **AND** *going-slow* **THEN** accelerate
  - *Corner-exit* **AND** *going-fast* **THEN** accelerate
  - *Corner-exit* **AND** *going-slow* **THEN** accelerate
- How do we deal with such situations? Which output membership value for accelerate to choose?

Example

Corner-entry AND *going-slow* THEN accelerate *Corner-exit* AND *going-fast* THEN accelerate *Corner-exit* AND *going-slow* THEN accelerate

- If we have the following degrees of membership:

  Corner-entry: 0.1   Corner-exit: 0.9

  Going-fast: 0.4   Going-slow: 0.6

- Then the results from each rule are

  Accelerate = min(0.9, 0.4) = 0.4

  Accelerate = min(0.1, 0.6) = 0.1

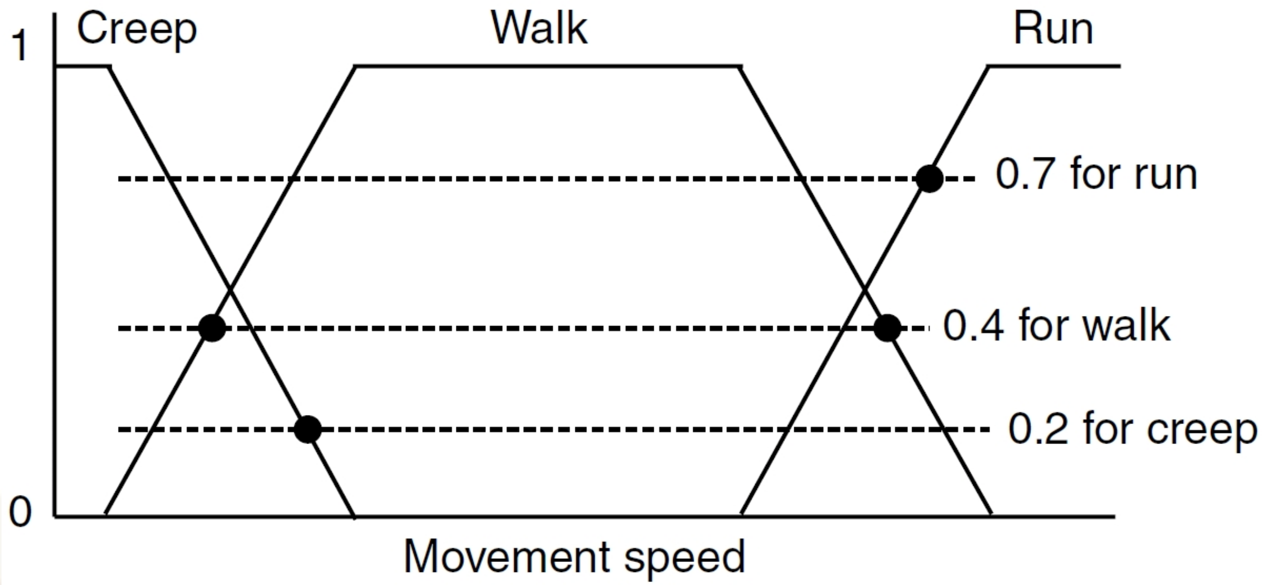
  Accelerate = min(0.9, 0.6) = 0.6

- So, the final value for accelerate is the maximum of the degrees given by each rule, namely, 0.6.

Université Concordia University

Step 3 – Defuzzification

- **Defuzzification Process**: ***Fuzzy output*** ➔ ***Crisp output*** **(a single output value)**
- **Consider an example where we have membership values of 0.2, 0.4, and 0.7 for the output fuzzy sets "creep," "walk," and "run."**



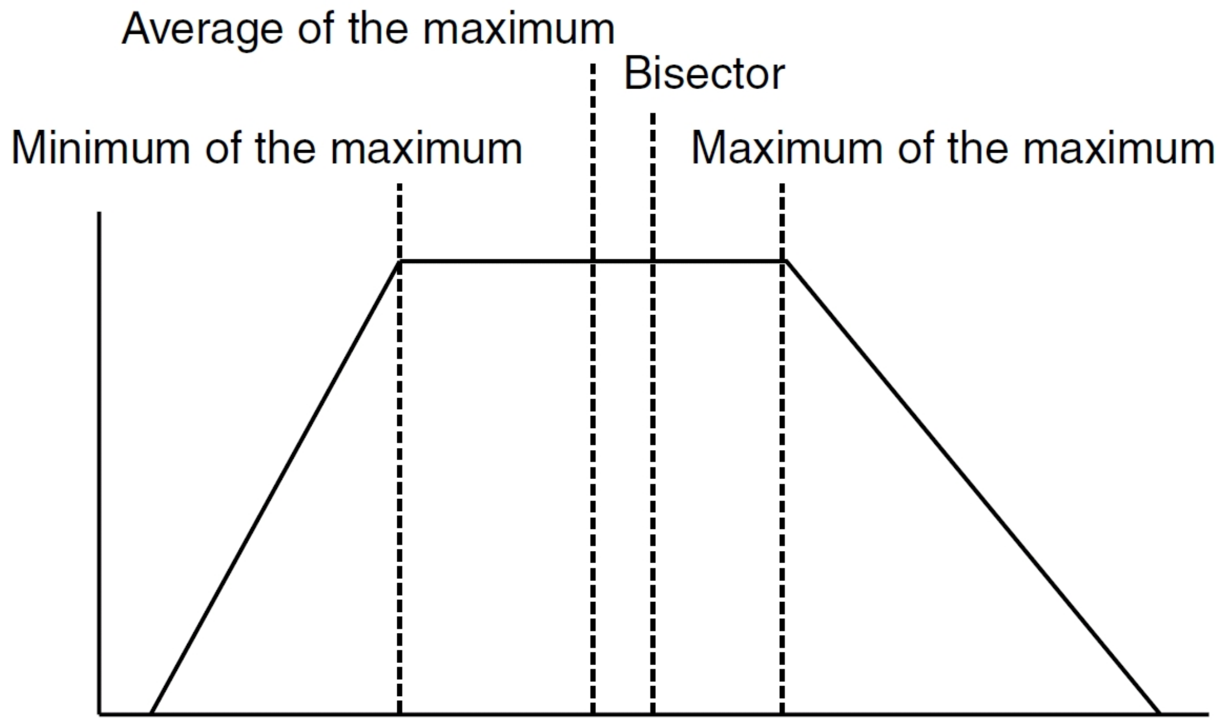
Step 3 – Defuzzification

Using the Highest Membership

- We can simply choose the fuzzy set that has the <u>greatest degree of membership</u> and choose an output value based on that m/f only.
- In the example above, the "run" membership value is 0.7, so we could <u>choose a speed</u> that is representative of running.
- Choose 1 out of 4 common characteristic points:
  - the *minimum* value at which the function returns 1; the *maximum* value (calculated the same way); the average of the two; and the <u>bisector</u> of the (area under the) function

Step 3 – Defuzzification

- This figure shows all four values for an example:



- **Drawback**: A character with membership values of 0 creep, 0 walk, 1 run ***will have exactly the same output speed*** as a character with 0.33 creep, 0.33 walk, 0.34 run.

Step 3 – Defuzzification

- **Better: *Blending Based on Membership***
  - **Blend each characteristic point *based on its corresponding degree of membership***
  - **E.g. Character with 0.2 creep, 0.4 walk, 0.7 run will produce crisp output given by**

    Output Speed = (0.2 * creep speed)
    + (0.4 * walk speed)
    + (0.7 * run speed)

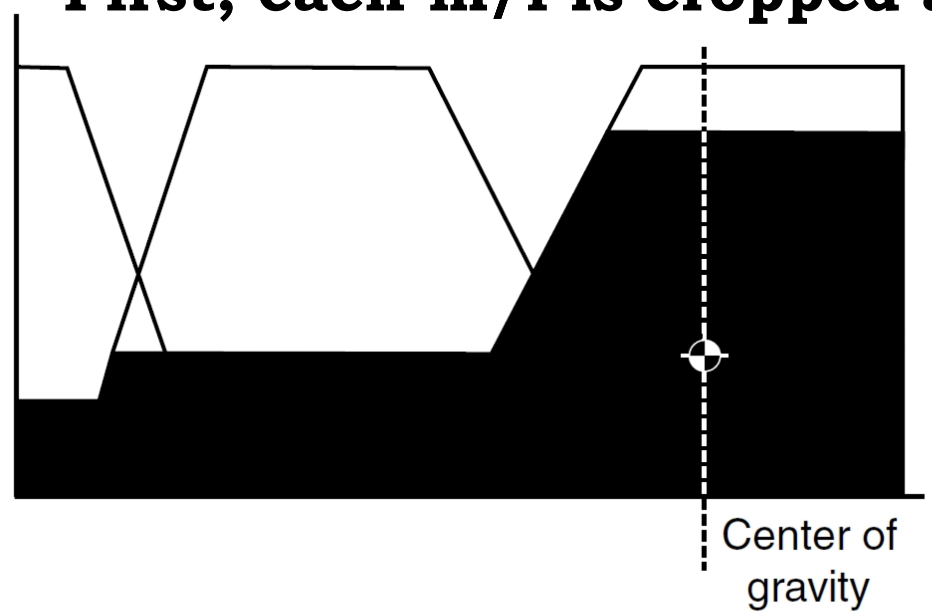
  - **Make sure that the eventual result is normalized (otherwise *result may be over-the-bounds or unrealistic*)**

Step 3 – Defuzzification

- **Blending Based on Membership, continued…**
  - **Common normalization technique: Divide total blended sum by the sum of fuzzy output values**
  - **E.g., Output speed = [(0.2 * creep speed) + (0.4 * walk speed) + (0.7 * run speed)] / (0.2 + 0.4 + 0.7)**
- **Possible blends and common names for them:**
  - **Minimum values blended (called: Smallest of Maximum, *SoM*; or Left of Maximum, *LM*)**
  - **Maximum values blended (Largest of Maximum, *LoM* or occasionally *LM!*; or Right of Maximum)**
  - **Average values blended (Mean of Maximum, MoM)**

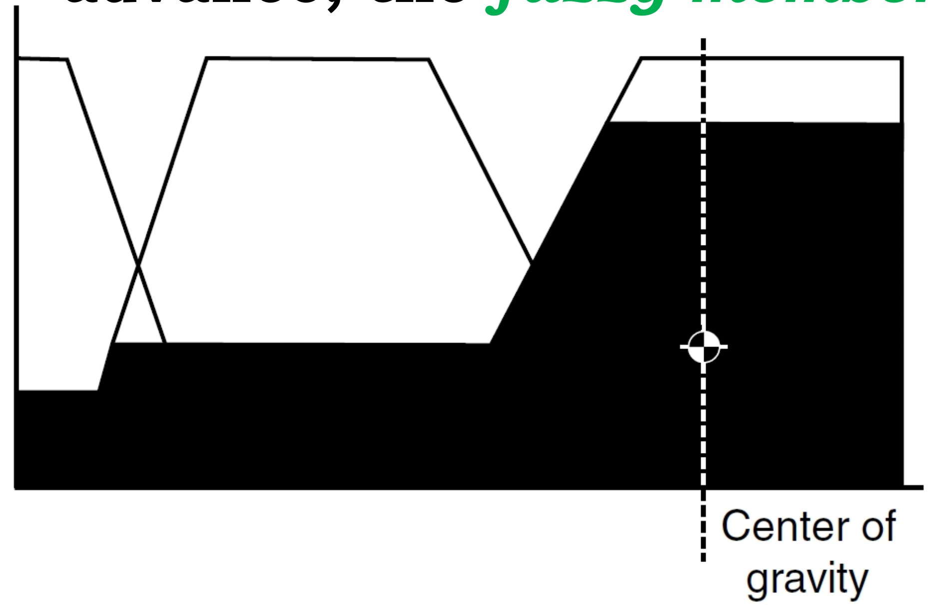
Step 3 – Defuzzification

- **Center of Gravity**
  - **Also known as Centroid of Area method → *Takes into account all membership values*, rather than specific characteristic ones (largest, smallest, average, etc.)**
  - **First, each m/f is cropped at the membership value of its set**



Step 3 – Defuzzification

- **Center of Gravity, continued...**
  - **Center of mass is found by integrating each in turn. This point is chosen as output crisp value**
  - **Unlike bisector of area method, we *can't compute this offline* since we do not know in advance, the *fuzzy membership values*, and *how m/f will be cropped***



Goal-Oriented Behaviour

- **<u>Beyond hard-coding stimulus-response pairs</u>**
- ***Seeks to satisfy internal goals*** **(*e.g.*, hunger, threat, gold, *etc.*)**

Goal-Oriented Behaviour

- **Goals** (motives)
  - Different levels of importance (insistence)
  - High insistence affects behavior more
  - Character *tries to fulfill goals to reduce insistence*

www.alphasims.com/wp-content/uploads/1311121163-68.jpg

- **Actions**
  - Possible actions to reach goal
  - Have Expected Impact on insistence of each Goal
  - Depend on *current state of the game*
  - Might need planning a few steps ahead

Goal-Oriented Behaviour

Example

- Goals with Insistence Values:

  Eat = 9, Kill Enemy = 8,

  Get Healthy = 4

- Actions with *Impact on Insistence Values of Goals*:

  Get Food (Eat: -5)

  Kill Enemy (<u>Kill Enemy</u>: -8, <u>Get Healthy</u>: +4)

  Get Health Pack (Get Healthy: –2)

- <u>Actions might affect other Goals</u>

Goal-Oriented Behaviour

- To pick the best action, ***consider the whole character state***
  - *i.e.*, pick the action that has the best net effect
- One approach is to ***use an overall discontentment measure*** that is calculated from the ***insistence values of the goals***
  - Pick the action that lowers the discontentment the most
- Scale certain insistence values (by using a weighted sum) so ***higher values contribute more***
  - *e.g.*, killing the enemy might be more important than eating ***unless the need to eat is critical***

Example

- Goals with (current) Insistence Values:

  Eat = 9, Kill Enemy = 8, Get Healthy = 4
- Actions with Impact on Insistence Values of Goals:

  Get Food (Eat: -5)
  Kill Enemy (Kill Enemy: -8, Get Healthy: +4)
  Get Health Pack (Get Healthy: –2)
- Discontentment = Eat+2*Kill Enemy + Get Healthy

Action	Outcome
Get Food	Eat = 4; Kill Enemy = 8; Get Healthy = 4 Discontentment = 24
Kill Enemy	Eat = 9; Kill Enemy = 0; Get Healthy = 8 Discontentment = 17
Get Health Pack	Eat = 9; Kill Enemy = 8; Get Healthy = 2 Discontentment = 27

Time Dependent Insistence Values

- (Some) Insistence Values should change with time.
- Each Action takes time; ***Update all Insistence Values that are time dependent*** after each action.
- Goals with (initial) Insistence Values:

  Eat = 9, Kill Enemy = 8, Get Healthy = 4
- Goals with Insistence Values after "Get Health Pack" Action:

  Eat = 9, Kill Enemy = 8, Get Healthy = 2
- Goals with (current) Insistence Values after update

  Eat = 9+0.1 = 9.1, Kill Enemy = 8,
  Get Healthy = 2

Goal-Oriented Action Planning

- Suppose a character is ***under attack*** and can **pick up a weapon** that allows it to ***more effectively shoot the enemies***
- **Goals with Insistence Values:**

  Eat = 7, Kill Enemy = 8, Get Healthy = 4, NewWeapons = 1
- **Actions with Impact on Insistence Values of Goals:**

  Get Food (Eat: -5, Get Healthy: +2)
  Kill Enemy (Kill Enemy: -8, Get Healthy: +7)
  Get Health Pack (Get Healthy: –2, Eat: +2)
  Get Weapon (NewWeapons: -1, Get Healthy: +8 plus Kill Enemy action will have no impact on Health)

Goal-Oriented Action Planning

- The outcome of an action could enable or disable other actions
- So, consider multiple actions in sequence
  - *Assume no action is repeated*
- Find the sequence that best fulfills a character's goals in the long term
- Need to simulate future state of the world

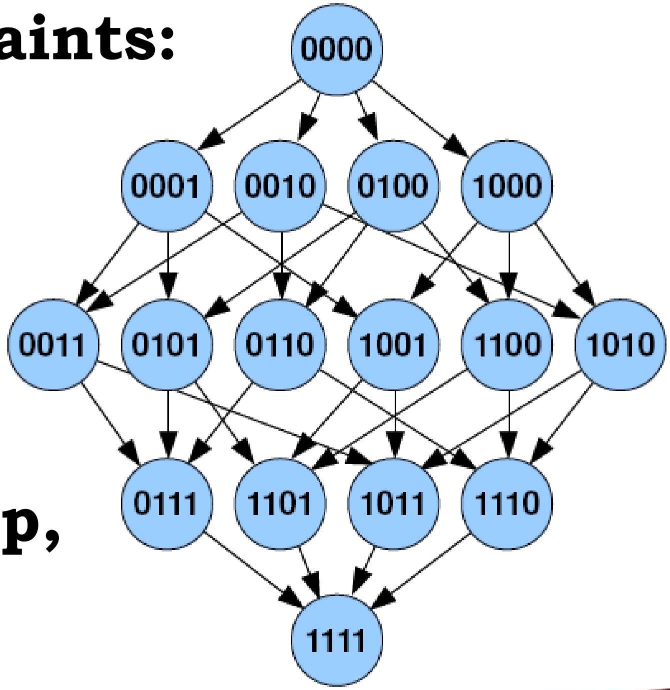
How to compute?

Optimal Sequence:

Get Health Pack, Get Weapon, Kill Enemy

Goal-Oriented Action Planning

- **Beyond single-step decisions**
- ***Search-space*** **(graph) for finding a goal that satisfies the necessary constraints:**



Example:

- **initial state is 0000**
- **goal state is 1111**

  (e.g., bit 0: food is picked up,

  bit 1: health pack is picked up,

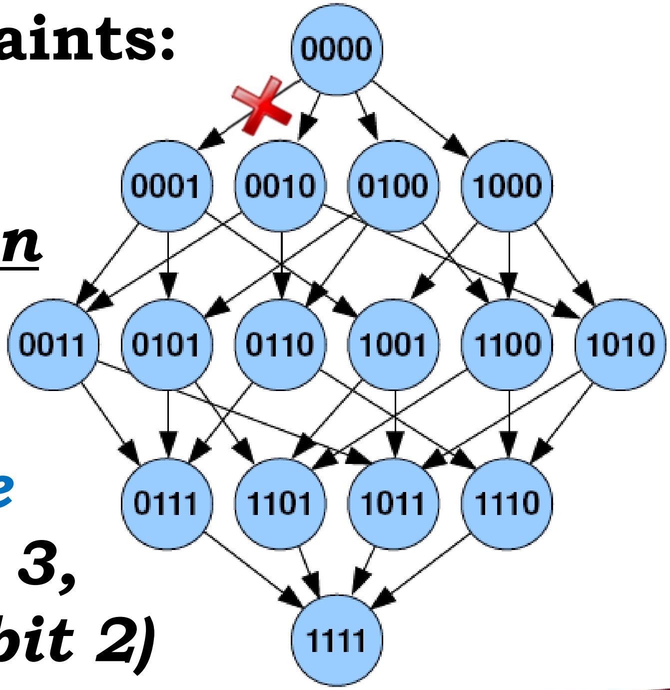
  bit 2: weapon is picked up,

  bit 3: enemy is shot)

[no action is repeated]

Goal-Oriented Action Planning

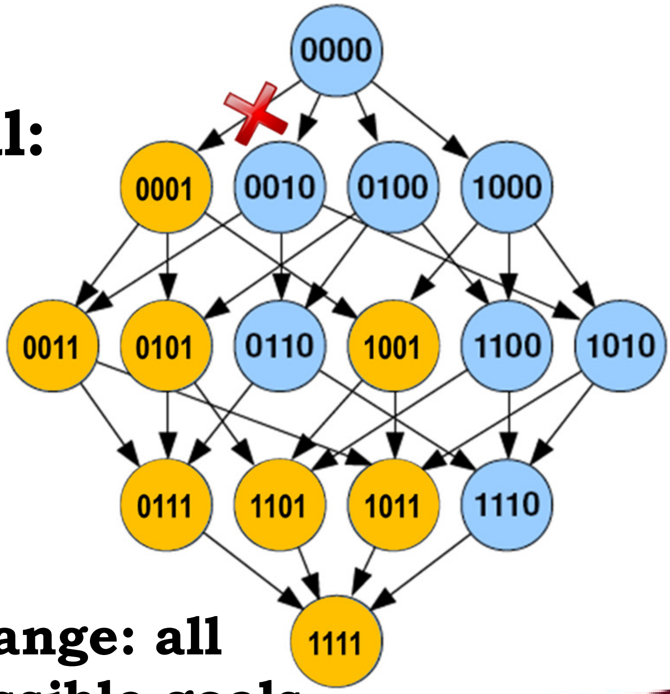
- **Beyond single-step decisions**
- ***Search-space*** **(graph) for finding a goal that satisfies the necessary constraints:**



- **Transitions:**
  - ***What could be the transition costs?***
  - ***The validity of a transition depends on the source state (e.g., can't shoot enemy, bit 3, if no weapon is picked up, bit 2)***

Goal-Oriented Action Planning

- What if the goal is to shoot enemy (whether health pack or food are picked up or not)?
- ***Search-space*** (graph) for finding a **partially-defined** goal:

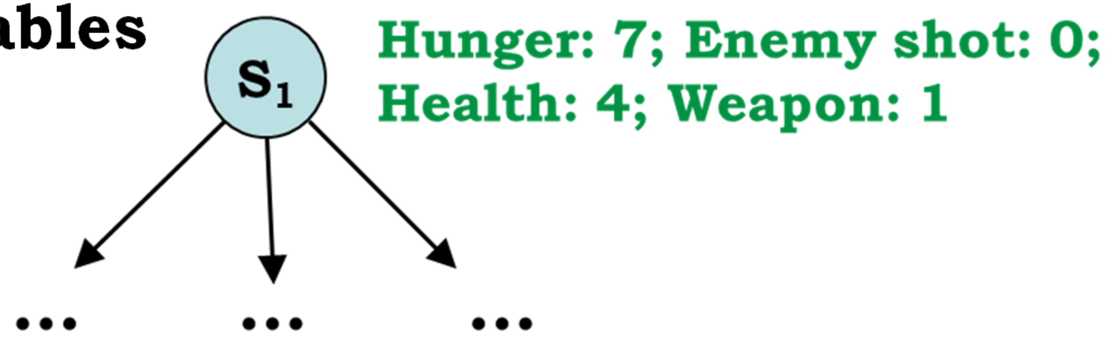


Orange: all possible goals

- What is an **efficient way** *to represent the game state vectors*?

Goal-Oriented Action Planning

- Working out Example with some non-binary variables

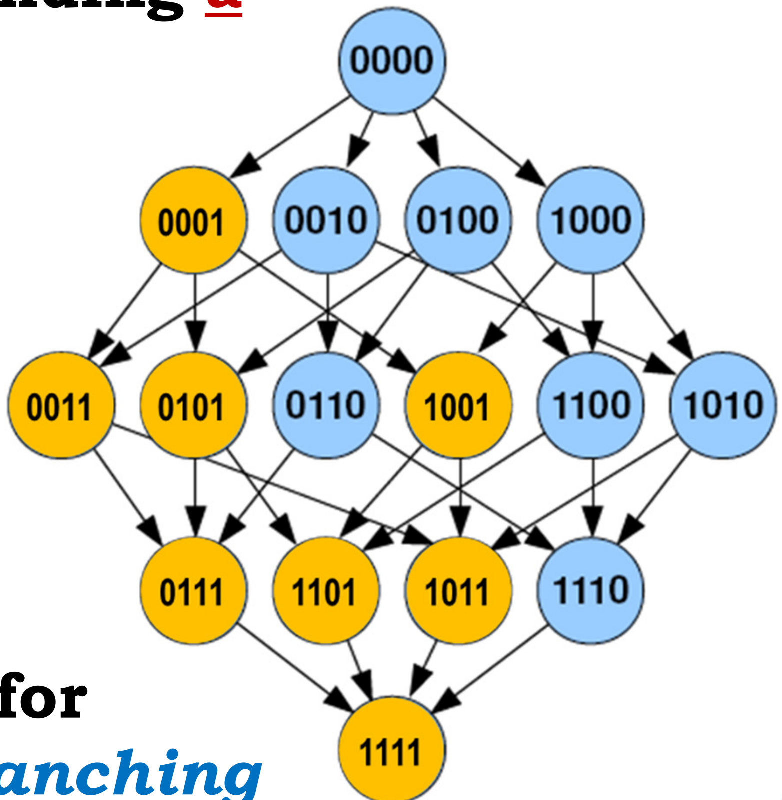


- Start with model of world and get available actions
- Choose an action and ***simulate the new world***
- Continue simulation ***until max depth***
- Iterate over ***all possible actions***
- Find sequence that creates the lowest discontentment level

→ **Not efficient!!!**

Goal-Oriented Action Planning

- ***Search-space*** (graph) for finding **a partially-defined goal**:
- **How to search the graph?**
- **BFS, DFS, A*, *etc.***
- **Problems with BFS and A* for large state vectors?**
- **IDA*: Very popular search for state-spaces *with large branching factors and shallow goals***



IDA* (Iterative Deepening A*)

- IDA* (Iterative Deepening A*)
  1. Set maxDiscont = 1 (or *some other small value*)
  2. Traverse the graph in DFS fashion without expanding states with discount > maxDiscont
  3. If *path to goal found*, *return the best path found*
  4. Else maxDiscont += 1 and go to step 2
- Complete and Optimal in any state-space (*with positive costs*)
- Memory: $O(bl)$, where b – max. branching factor, l – length of optimal path
- Complexity: $O(kbl)$, where k is the number of times DFS is called

Other Decision Making Topics

- Markov Systems (§ 5.6)
- Rule-based Systems (§ 5.8)
- Blackboard Architectures (§ 5.9)
- Scripting (§ 5.10)–Dropped from Ed.3 of AI for G.
- Action Execution (§ 5.11)
- Utility Functions