cis20.2
design and implementation of software applications 2
spring 2010
lecture # IV.1
introduction to intelligent systems

"AI is the science of making machines do things that would require intelligence if done by [humans]" (Marvin Minsky, 1963)

today's topics:

- foundations of artificial intelligence
- knowledge representation
- reasoning

foundations of artificial intelligence

- Artificial intelligence (AI) is both science and engineering. It is the *science* of understanding intelligent entities—of developing theories which attempt to explain and predict the nature of such entities; and it is the *engineering* of intelligent entities.
- There are four main views of AI in the literature, listed below.
  - AI means *acting humanly*, i.e., acting like a person. The classic example of this is the "Turing test" (details on a later slide).
  - AI means *thinking humanly*, i.e., thinking like a person. The field of *Cognitive Science* delves into this topic, trying to model how humans think.
    The difference between "acting humanly" and "thinking humanly" is that the first is only concerned with the actions, the outcome or product of the human's thinking process; whereas the latter is concerned wtih modeling human thinking processes.
  - AI means *thinking rationally*, i.e., modeling thinking as a *logical* process, where conclusions are drawn based on some type of symbolic logic.
  - AI means *acting rationally*, i.e., performing actions that increase the value of the state of the agent or environment in which the agent is acting. For example, an agent that is playing a game will act rationally if it tries to win the game.

acting humanly: Turing Test

- This is a problem that has greatly troubled AI researchers for years. They ask the question "when can we count a machine as being intelligent?"
- The most famous response is attributed to Alan Turing, a British mathematician and computing pioneer. The famous "Turing Test" was named after him, based on ideas he expressed in a paper published in 1950.

  Human interrogates entity via teletype for 5 minutes. If, after 5 minutes, human cannot tell whether entity is human or machine, then the entity must be counted as intelligent.
- To date, no program has yet passed the Turing Test! However, there is the annual Loebner Prize which awards scientists for getting close. See `http://www.loebner.net/Prizef/loebner-prize.html` for more information.
- In order to pass the Turing Test, a program that succeeded would need to be capable of: speech recognition, natural language understanding and generation, and speech synthesis; knowledge representation; learning; and automated reasoning and decision making. (Note that the basic Turing Test does not specify a *visual* or *aural* component.)

acting humanly: Searle's Chinese Room

- Another famous test is called the "Chinese Room" which was proposed by John Searle in a paper published in 1980.
- Suppose you have a computer in a room that reads Chinese characters as input, follows a program and outputs (other) Chinese characters. Suppose this computer does this so well that it passes the Turing Test (convinces a human Chinese speaker that it is talking to another human Chinese speaker). *Does the computer understand Chinese*?
- Suppose Searle is in the room, and he uses a dictionary to translate the input characters from Chinese to English; he then constructs his answer to the question, translates that back into Chinese and delivers the output—*does Searle understand Chinese*?
- Of course not.
- This is Searle's argument: the computer doesn't *understand* it either, because all it is doing is translating words (symbols) from one language (representation) to another.

thinking humanly

- Thinking humanly means trying to understand and model how the human mind works.
- There are (at least) two possible routes that humans use to find the answer to a question:
  - We reason about it to find the answer. This is called *"introspection"*.
  - We conduct experiments to find the answer, drawing upon scientific techniques to conduct controlled experiments and measure change.
- The field of *Cognitive Science* focuses on modeling how people think.

thinking rationally

- Trying to understand how we *actually* think is one route to AI.
- But another approach is to model how how we *should* think.
- The "thinking rationally" approach to AI uses *symbolic logic* to capture the laws of rational thought as symbols that can be manipulated.
- *Reasoning* involves manipulating the symbols according to well-defined rules, kind of like algebra.
- The result is an *idealized* model of human reasoning. This approach is attractive to theoretists, i.e., modeling how humans should think and reason in an ideal world.

acting rationally

- Acting rationally means acting to achieve one's goals, given one's beliefs or understanding about the world. An *agent* is a system that *perceives* an environment and *acts* within that environment. An intelligent agent is one that acts *rationally* with respect to its goals. For example, an agent that is designed to play a game should make moves that increase its chances of winning the game.
- When constructing an intelligent agent, emphasis shifts from designing the *theoretically best* decision-making procedure to designing the best decision-making procedure possible within the circumstances in which the agent is acting.
- Logical approaches may be used to help find the best action, but there are also other approaches.
- Achieving so-called *"perfect rationality"*, making the best decision theoretically possible, is not usually possible due to limited resources in a real environment (e.g., time, memory, computational power, uncertainty, etc.).
- The trick is to do the best with the information and resources you have. This represents a shift in the field of AI from *optimizing* (early AI) to *satisfying* (more recent AI).

differing views of AI

- There are three basic approaches to AI systems.
- *Connectionist* approaches represent intelligence as emergent behaviors that are the result of a network of interconnected simple (numeric) units. Examples include *neural networks* and *perceptrons* (simple neural networks). Inputs to the network are represented by numbers, which are mapped to numeric outputs. Inside the network, a mathematical function is computed, by multiplying and adding the inputs to a set of internal *weights* and *biases*. The earliest work on neural networks was done by Warren McCulloch and Walter Pitts at Princeton in the 1940's. Seminal work was done by James Rumelhart and David McClelland in the 1970-80's, which forms the foundations of the field today.
- *Logic-based* approaches represent intelligence by resolving and unifying logic symbols and statements using techniques like *"deduction"*. This approach was developed by John McCarthy at Stanford in the 1950's and later.
- *Symbolic* approaches represent intelligence by explicit structures that symbolize different bits of knowledge and activities. Explicit *rules* control intelligent behavior. The first work using this approach was developed by Allen Newell and Herbert Simon at Carnegie Mellon University in the 1950's and later.

implementing intelligent systems

- No matter which approach is taken, there are many components that need to be addressed in order to implement an intelligent system.
- The problem of *representation* looks at how to map the environment in which an agent is acting into symbols that can be manipulated. For example, if you want to build an intelligent agent to play the game of tic-tac-toe, how would you represent the board using symbols?
- The problem of *reasoning* looks at how to manipulate the symbols to make decisions. For example, how does your tic-tac-toe player decide which move to make next?
- The problem of *tractability* looks at how to make decisions about what to do in a reasonable amount of time.

knowledge representation

- *Knowledge Representation (KR)* focuses on designing specific mechanisms for representing the knowledge in ways that are useful for AI techniques.
- Knowledge about a domain allows problem solving to be focused, so that we don't have to perform exhaustive search (i.e., for an agent to decide what to do).
- *Explicit* representations of knowledge allow a *domain expert* to understand the knowledge a system has, add to it, edit it, and so on. This is called *knowledge engineering*.
- Comparatively simple algorithms can be used to *reason* with the knowledge and derive "new" knowledge from it.
- How do we represent knowledge in a form that a computer can use?
  A list of desirable features is contained on the next page.

- *representational adequacy*— A KR scheme must be able to represent the knowledge appropriate to the problem being addressed.
- *inferential adequacy*— A KR scheme must allow new *inferences* to be made from old knowledge. Inferences should be: *sound* (new knowledge follows from old) and *complete* (makes all the right inferences). Soundness is easier; completeness is hard!
- *inferential efficiency*— A KR scheme should be *tractable*, i.e., one should be able to make inferences from it in reasonable (polynomial) time Unfortunately, *any* KR scheme with interesting *expressive power* is probably not efficient. Often, the more *general* a KR scheme is, the *less efficient* it is.
- *well-defined syntax and semantics*— A KR scheme should have *well-defined syntax*, i.e., it should be possible to tell whether any construction is "grammatically correct" and how a particular construction should be understood (no ambiguity). A KR scheme should have *well-defined semantics*, i.e., it should be possible to precisely determine, for any given construction, exactly what its meaning is. Defining syntax is relatively easy; but semantics is hard!
- *naturalness*— Ideally, a KR scheme should closely correspond to our way of thinking, reading, and writing. A KR scheme should allow a *knowledge engineer* to read and check the *knowledge base* (database that contains the knowledge represented). The *more general* a KR scheme is, the less likely it is to be readable and understandable.

production systems

- There are many types of Knowledge Representations. One simple, early model is called a *production system*. Knowledge is represented as a collection of *production rules*.
- Each rule has the form: $condition \longrightarrow action$
  which may be read "if $condition$ then $action$"
  where the $condition$ (i.e., antecedent) is a *pattern*
  and the $action$ (i.e., consequent) is an operation to be performed if rule *fires* (i.e., when the rule is true).
- A "production system" is essentially a list of rules and it is used by matching a system's state against the conditions in the list.
- An action can be something the agent does, or it can be an action that manipulates the agent's memory, such as adding a rule to its database.
- The mechanism that fires rules is frequently called an *inference engine*.

- Here's a simple example:

```
R1:  IF animal has feathers
     THEN animal is a bird
R2:  IF animal is a bird
     THEN animal can fly
R3:  IF animal can fly
     THEN animal is not scared of heights
```

- A set of rules like these can be used in two ways to perform reasoning:
  - *forward chaining*—data driven: reasoning forward from conditions to actions
  - *backward chaining*—goal driven: reasoning backward from goals back to facts
- Sometimes a system has to make choices about which rules should fire, if more than one is applicable at a given time. This is called *"conflict resolution"*. There are a number of approaches to conflict resolution: most specific rule first (with most matching antecedents); most recent first (i.e., rule that became true most recently); user/programmer specified priorities; or use "meta-knowledge" (i.e., knowledge about knowledge..., which can be encoded into the system).

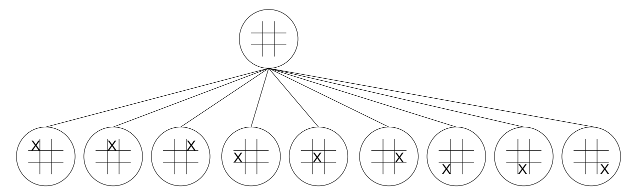
graph representations

- Knowledge can also be represented in a graph.
- A *directed graph* is a set of variables and a set of directed arcs between them.
- A directed graph is *acyclic* if it is not possible to start at a node, follow the arcs in the direction they point, and end up back at the starting node. We will only talk about *directed acyclic graphs (DAG)*.
- A is the *parent* of B if there is a directed arc from A to B.
- B is the *child* of A if A is the parent of B.
- Any node with no parents is known as a *root* of the graph.
- Any node with no children is known as a *leaf* of the graph.
- The parents of node A and the parents of those parents, and the parents of those parents, and so on, are the *ancestors* of A.
- If A is an ancestor of B, then B is a *descendent* of A.
- We say that there is a *link* between two nodes A and B if A is a parent of B or B is a parent of A.

- Two nodes A and C in a graph have a *path* between them if it is possible to start at A and follow a series of links through the graph to reach C.
- Note that in defining a path we ignore the direction of the arrows; in other words we are considering the underlying undirected graph.
- A graph is said to be *singly-connected* if it includes no pairs of nodes with more than one path between them.
- A graph which is not singly-connected is *multiply-connected*.
- A singly-connected graph with one root is called a *tree*.
- A singly-connected graph with several roots is called a *polytree*. A polytree is sometimes called a *forest*.

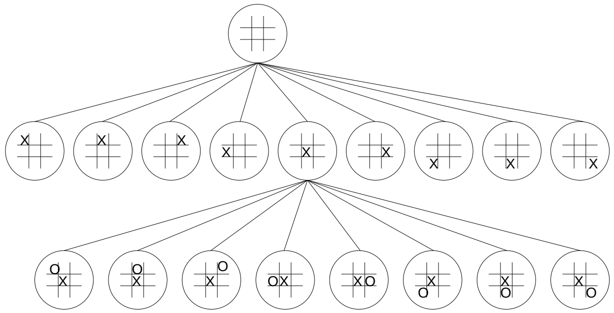
game trees

- A common way to represent knowledge for a game-playing agent is using a graph called a *game tree*.
- The *root* of the game tree contains a represention of the starting *state* of the environment.
- The children of the root represent possible moves from the start state.
- Here is an example for the game of tic-tac-toe:



- The root of the tree is the empty board at the beginning of the game.
- The children of the root represent all the possible moves for the "X" player, who goes first in the game.

- If the "X" player chooses to go in the middle, then the game tree can be *expanded* from that node, and all the possible moves for the "O" player are shown:



reasoning

- Knowledge representations are used for agents to reason and make decisions about what to do.
- A game tree can be used by game players to reason and make decisions about what move to make next. This is often referred to as a player's *strategy*.
- The entire tree could be expanded (this would be very big for tic-tac-toe!), and it would be possible to determine which sets of moves lead to winning the game.
- All the board positions can be given *values*, indicating their *utility*.
- All final board positions (when all the squares in the tic-tac-toe board are filled) are represented as *leaves* in the tree. A winning board position could be given a positive value and a losing board position could be given a negative value. Then, traversing backwards from the leaf previous board positions can be assigned positive or negative values that indicate whether they lead to winning or losing positions.
- An agent can *reason* about what to do, i.e., which move to make, by choosing nodes with high utility values.

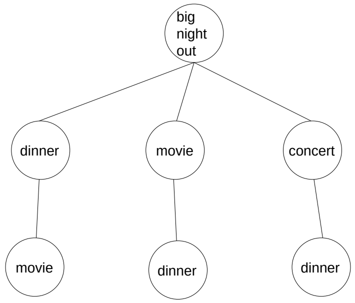
- The size of the tree, if all possible nodes are expanded, is referred to as its *branching factor*. The game tree for Chess has a really big branching factor!
- It is not tractable to fully expand the Chess game tree (even tic-tac-toe is big).
- So AI looks for ways to make decisions without fully expanding the tree.

search

- *Search* is a technique used to visit the nodes in the tree and calculate the cost and utility of taking certain paths.
- There are different *search algorithms* that are used in AI.
- *Depth-first* search means traversing the tree to the deepest path first, and then going back up to shallower branches (i.e., first child of root, then first grandchild of root, then first great grandchild, etc.).
- *Breadth-first* search means traversing the tree by visiting all the children of the root, then all the grandchildren of the root, etc.
- *Best-first* search means traversing the tree by selecting the child of the root with the highest utility and lowest cost; then selecting that child's child with the highest utility, etc.
- Search techniques can help *prune* a tree, to avoid expanding all the nodes.

decision trees

- A game tree is an example of a *decision tree*. The nodes in a game tree represent the state of a game board, and the links between nodes represent moves in the game.
- In a decision tree, the nodes represent the result of making a choice, and the links between nodes represent the results of making combinations of choices.
- Below is an example of a decision tree for planning a big night out:



- Typically a decision tree includes *costs* of going from one node to another. With the "big night out" example, a very real cost (in dollars) is associated with each node.
- A decision tree can also represent *constraints*. In this example, suppose that there are only certain times at which each event can begin. Suppose that dinner can only be at 7pm or 9pm, movie can only be at 7pm or 9pm, concert can only be at 7pm. If the person planning the "big night out" wants to go to dinner at 7pm, then s/he loses out on the opportunity to go to the concert. This is represented in the decision tree.