Basic Principles of Symbolic AI

Conventional Computing

- Any program involves three things: objects to work on (data), operations to perform on those objects, and a control strategy
  - The control strategy decides which operations to perform on which objects, and in what order
- Think of how you solve a problem using a computer program. What do you do?
- To write the code, you first develop an algorithm for solving the problem – that is, you decide how the computer's going to do it – then write some code (or both together ☺)
- Where's all the intelligence in this?

Conventional Computing

- Essentially you solve the problem yourself (do all the intellectual work), then translate that algorithm into a computer program (purely mechanical work at this point)
- Your algorithm is combined with the objects and operators into a control strategy in the form of program code
- Why can't conventional strategies be used easily for something like recognizing a scene, or deciphering speech, or driving a car?

Where Conventional Computing Fails

- We can't dream up an algorithm for these!
- Conventional computing strategies are fast and development techniques are well understood, but if no algorithm is known, they're not useful!
- In AI, we use a *declarative* computing model
- Declarative computing separates the objects and operators from the control strategy
- We declare the objects and operators in some form of representation
- Where does the control strategy come from? Let's look at an example…

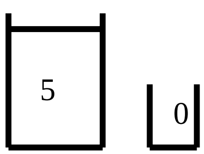
The Water-Jugs Problem

- You have two jugs of different sizes
- You have a quantity of the beverage of your choice in each
  - These are our objects (think of some representation)
- We also have operators: we can pour from one jug into the other until the latter is full or the former is empty (2 operators) or we can empty either jug on the ground
- You desire to have some specific arrangement of water in each jug
- This is an abstract problem – we can make instances of it by specifying details

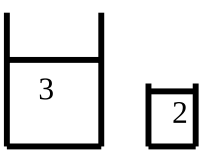
One Situation

- 5L and 2L jugs, with 5L of whatever in the 5L jug, and none in the 2L jug
- Our operators are already specific
- We want to have 1L of water in the 5L jug, and the 2L be empty
- Now SAY you want to write a computer program for this – we first develop an algorithm. We'd sit down on paper and figure out what would need to be done…

Finding an Algorithm



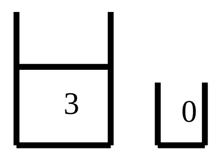
"well, we could empty the 5, but then We'd be at a dead end. It wouldn't make sense to empty the 2 or pour from 2 to 5, so pouring from 5->2 is our only option



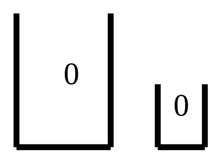
"ok, now we could pour from 2->5, but We'd be back where we started. We could empty 5, or we could empty 2. Pouring from 5->2 doesn't make sense.
2 valid choices. For convenience I'll pick Empty 2.

Incidentally, why would you pick empty 2 over 5 at this point?

Finding an Algorithm (cont)

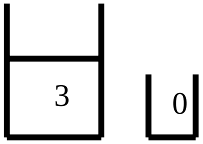


Now we can pour from 5->2 again, Or we can empty 5. Pouring 2->5 or Emptying 2 makes no sense. Let's Empty 5.

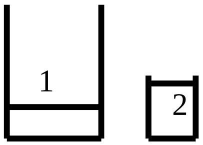


Dang. That didn't help. Let's try that one again and take the other choice…

Finding an Algorithm (cont)

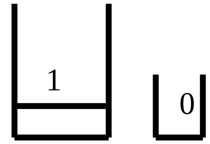


Now we can pour from 5->2 again, or we can empty 5 (but we already tried that. Pouring 2->5 or emptying 2 makes no sense. So there's only one choice now: we pour from 5->2.



Now we can pour from 2->5, but that would just put us back to the last step again. Pouring 2->5 makes no sense. We can empty 5 or empty 2. Let’s Empty 2.

Finding an Algorithm (cont)



TA DAH! Ok, this is no huge intellectual milestone, but we’ve achieved what we wanted.

- We have an algorithm: Pour 5->2, empty 2, pour 5->2, empty 2.
- Now what?

Now

- Now that we’ve developed an algorithm, we could write a computer program for it and spit out the answer. But again *WE* did all the work.
- Suppose the specific problem changed – the size of the jugs, new operators, the amount of water? We’d need to develop a NEW algorithm
- What we WANT is the ability for a program to take over the intelligent part – to do what we did!

Declarative Computing

- We want to give it the objects and operators, and let IT develop it’s own algorithm for solving the problem. We’d like it to do what we did in our example
- If you had to pick a word for the process I illustrated, what would it be?
- SEARCH!
- There were any number of irrelevant combinations of operators (exponential!), and we effectively steered our way through the space of possibilities to arrive at the correct combination

SEARCH is fundamental to AI

- We want our program to search through the space of possible combinations just as we did.
- Worst case: try all possible combinations. Nasty because we have an exponential problem (it'd even be infinite if we allowed for the refilling of jugs!)
  - This is called a Toy Problem because we could fairly easily try all possibilities. I'm just using it to illustrate a point though – the same thing goes for harder stuff like context in language processing for example
- We didn't try everything. What were some of the things we DID do?

Our Search

- We eliminated considering operators that would put us back in a state we've already been in
- We eliminated (all but once) operators that would lead to dead ends
- Where one did lead to a dead end, we backed up to the last choice point and kept going
- We also (more subtly) tried to choose operators that were most likely to lead to our goal

Search and Intelligence

- We did a few things that caused us to eliminate most of the possible states we could have visited (most of the Problem Space)
- If we watched somebody solve it by trying everything, we'd point out the knowledge that would let you avoid dealing with paths that made no sense
- What does this say about intelligence?
- *INTELLIGENCE IS AVOIDING SEARCH*

Comparing Conventional and Declarative approaches

- Speed?
  - Conventional will be faster – but only because we do all the thinking ahead of time!
  - Declarative will naturally be slower to execute as part of it's execution IS solving the problem
  - Somewhat unfair comparison because we don't count the person-hours to solve the problem in conventional systems!
- BUT REMEMBER – we use declarative approaches because there isn't a conventional algorithm

A few things to note…

- We'll come back to this stuff but while we're here…
- If this was a physical situation problem we'd have problems backing up (can't pick up spilled water off the ground!)
- If this was a physical problem our system would be figuring out a PLAN and then carrying it out
  - this is a planning problem, not unlike planning to build a house or planning a a street route

A Few things to Note

- If our system is searching for a solution, the system itself still employs an algorithm – whatever search process it uses to find its solution to the problem. No different from us when we think up a plan – we're systematic to some degree. The point is in AI the solution for the problem is developed dynamically by the system.

Generality vs. Specialization

- In our case we used a very general approach to find a solution: one that could be used in many problems
- We'd call this a *weak technique* in AI: one that would succeed on many simple (toy) problems like this one, but fail as things got more complex
- e.g. if we tried to use this for driving
- We call this the *scaling problem*: general techniques don't work as problems get more complex

Generality vs. Specialization

- We said in the history of AI discussions that this realization led to the development of more powerful specialized techniques
- We'll see these as we get into more specialized subfields: the same general ideas apply, but they're adapted to be much, much more powerful for certain types of problems
- Now, I want to go back to the idea of Symbols and the Physical Symbol System Hypothesis

Back to Symbols

- When we solved that problem we used internal symbols for rules (empty5, pour5into2), and symbols for what those rules meant
- We also visualized (symbolically, through drawings in this case) what the state of the problem was
- We used and associated symbols, a la the Physical Symbol System hypothesis
  - Albeit in a toy problem: the principles are exactly the same for more complex problems, though!

Achieving this in a Computer Program

- The idea of using a search process should be familiar to you from other courses
- What you have likely not thought about before was the symbolic form underlying your own problem solving activities
- If we're going to get a computer system to solve problems in an intelligent fashion, we need to have some form of symbolic representation for the system to work with
- Our own such representations can get extremely complex (often multiple reps)

PRINCIPLES OF SYMBOLIC AI

- REPRESENTING a problem in symbolic form, allowing for the association of symbols and the construction of new symbols
- SEARCH through a space of possible solutions
- AVOIDING SEARCH through the appropriate application of symbolic knowledge
- We're going to look at all of these, but to be able to understand and implement search for problem solving, we need to start with symbolic representation

Readings So Far

- Chapter 1 in the text
- We've actually covered a few sections in later chapters as well – you'll hit them when we get there and recognize it
- Next readings: Representation and Logic
- Chapters 2 and 15
  - We won't be covering all of 15 (Prolog)
  - Specifically, a lot of the data structures and so on we won't get to in Prolog, but it's worth skimming
  - Also interesting as a further to to those who have taken 3440 (non-imperative programming)