BIG PICTURE LECTURE

Announcements

- Reading
  - Levesque Paper, link on the main course web page

OK, we are done with our study of search *per se*. While search will remain important in much of what we do, it will not be the focus of attention so much as a background tool.

Now let's step back and ask a big question, one that has interested humans for millennia, and that many AI researchers are also fascinated by:

Is thinking an algorithmic (mechanical) process?

There are many ways to phrase this; for instance, one could ask: is there a logic for thought in general? Here is a brief summary of some important historical efforts:

– Aristotle –

(~350BCE) Formulated *Aristotelian (subject-predicate) Logic*, that was held in high regard for roughly 2000 years.

– Leibniz –

(~1700) Proposed a *rational calculus* to capture general laws of thought. But was too busy with other things (such as co-inventing math calculus!) to carry this out.

– Boole –

(1847, 1854) Published *Analysis of Logic*, and *Laws of Thought*, involving a preliminary treatment of quantification.

– Frege –

(1879) Published the first account of what is now called *First-Order Logic*, going well beyond Aristotelian Logic; FOL is still the central form of mathematical logic today.

– Turing –

(1950) Published the paper "Computing Machinery and Intelligence."

– McCarthy –

(1958) Published "Programs with Common Sense."

Now, many many others have written very deep contributions to the overall question, and we will consider some of these toward the end of the semester. For now, we will take a quick glance at ideas of Turing and McCarthy.

Alan Turing is sometimes called the *father of computer science*. He built one of the first computers, as part of a British effort to decode German messages during World War II. He was quick to realize that the methods could generalize to arbitrary kinds of computation, and he formulated a theoretical characterization of what counts as computation: whatever a so-called Turing Machine can do. A **Turing Machine** is an imaginary device with a tape of infinite length on which symbols (for example 0s and 1s, though any symbols could be used) can be *written, read, erased* according to a *finite list of precise instructions*. Such a simple process may hardly seem like computation, but *he showed that all known computations could be simulated that way*. This proof, that all computations can be performed on a Turing Machine is known as **Turing's Thesis,** and it is widely accepted as correct. He also showed that *there are problems that cannot be solved by computation* (such as the famous **Halting Problem**). But his 1950 work in Computing Machinery and Intelligence was of a different sort. There he speculates on the possible computational nature of intelligence and proposes a test (now called the **Turing Test**): suppose a human judge can communicate (by typed English messages) with two agents in another room, one being human and the other a computer. If the judge can tell which is human and which is not, then the computer – so Turing suggests – is as good as the human at English conversation, and so we might as well say it is intelligent.

This has been widely accepted and also widely criticized. The Loebner Prize awards a sum of money each year to the program that comes closest to passing the Turing Test. See http://www.cs.umd.edu/class/fall2013/cmsc421/turing for more information on the Turing Test and the Loebner Prize, including links to some past winning programs (in some cases you can actually chat with them online). Hector Levesque is one person (of many) who has argued that the Turing Test is not a good measure of intelligence; his paper on this I have assigned as reading.

In addition to Levesque's critique, one can do a simple counting argument. Suppose we want to Turing-test a given program on "conversations" of up to 100 words, 10 per sentence, 5 sentences per agent (one human and one machine). And suppose we limit them to only ten word-choices per word. That is, each sentence consists of ten words, and each word has to be one of a given list (for that position in the sentence, eg the subject noun can be one of: rooster, cat, Fido, Sally, house, and so on for ten items; and just ten verbs, etc. That means there could be in all exactly 10^100 such conversations, which in more than the estimated number of electrons in the known universe. No chance at all of storing that in computer memory! And these were just the very limited conversations based on a very tiny set of word choices. Thus while theoretically one could store every possible conversation and just have the computer parrot back a new part when the conversation *so far* matches a conversation in memory, it is totally impractical. Thus if someone did program a machine to converse in a human-seeming way in general (not just

once in a while) that would be a very impressive feat and we'd want to see how it did that. It would be virtually unbelievable that it had all the responses memorized.

In 1958 John McCarthy – of Dartmouth Conference fame – published a paper suggesting that the design of an intelligent machine might better be done in well-crafted steps rather than all at once. In particular, he argued that a precursor to a problem-solving kind of intelligence is that of an **understanding intelligence**: one that is able to understand solutions when explained to it. His points were that (i) without the ability to *understand* a solution, there is little hope of coming up with one, and (ii) the ability to understand a solution may be a lot easier to build into a program than that of devising a solution. In fact, he proposed a particular research program with just that aim; he called such a program an **advice-taker***: a system that can be taught.*

Just what would be involved in building an advice-taking program? Quite a lot: *natural-language processing* (the advice is to be in English); *reasoning* (so the advice can be used when appropriate); *learning* (the advice amounts to a change in possible future behaviors); *planning* (such as when and how and for what to seek advice); and *knowledge representation* (a convenient general format in which advice is to be stored). These in fact comprise very close to a complete list of the major subfields of modern AI research – all foreshadowed in one 1958 paper! And thus McCarthy's vision of building an advice-taker has not been accomplished even today, although a great deal of progress has been made.

– Comment on Programming Projects –

I will reveal here that one of your four programming projects will probably be a partial attempt to build an advice taker, along these lines: your programs will accept English input such as "Fido is a dog" and "Dogs are pets" and "Is Fido a pet?" – and will output English such as "OK" and "I understand" and "Yes, Fido is a pet". That is, the user will act as a teacher of your program, where the teacher both instructs and tests.

McCarthy's vision actually breaks problem-solving down into *two parts: having ability to understand a solution* (this is the advice-taker), and *having the ability to come up with a solution* (this is a sophisticated mix of planning and reasoning and perhaps other things). Much work has been done on both parts (and as we saw above, the advice-taker part itself involves many abilities).

We will in fact soon (next week) turn to the Monkey and Bananas problem to motivate our study of automated reasoning/problem-solving. But for now, I will mention a few aspects of natural-language processing, and what might be called **The NLP problem:** given inputs consisting of English sentences, produce appropriate outputs that are English sentences and/or actions. Notice how *ill-defined* this is: how is it decided that an output is appropriate? Yet we manage to decide this, routinely, whenever we converse. AI problems are often like this, without sharply specified conditions for correctness.

However, things are not quite as bad as this may make it seem. For one, **linguists** have been hard at work for along time, and have broken the study of language into key pieces, including:

- **Phonology** – the study of a language's basic sounds (roughly syllable-level or finer)
- **Morphology** – the study of word-formation (such as how plurals are formed, etc)
- **Syntax** – the study of how words fit together into sentences
- **Semantics** – the study of meaning, (e.g., how word meanings determine sentence-meaning)
- **Pragmatics** – how context contributes to meaning

Each of these is the subject of intense ongoing study by researchers in NLP. And some aspects will enter into your programming projects, along with reasoning and other issues.

One thing we have not emphasized up until now is the idea of a **knowledge base** (often abbreviated KB). This is the *repository of information that an agent has available to draw upon* – whether built in or learned or inferred. Arguably, the presence of a KB (and the ability to add to it, change it, use it) is what makes an agent an agent. And it will play an ever-more-important role as we go forward.

ACTIONS:

We introduced search in terms of planning: how an agent might choose a sequence of actions that ought to lead to a given goal. But we did not say a great deal about what constitutes an action, except that it takes the world from one state (or situation) to another. We need to look more closely at actions, since they can be complex. For one thing, many actions cannot be done at all except under special conditions. Thus "raise left arm" cannot be done by an agent that has no arms; nor by one whose left arm motion is blocked by concrete barriers, etc. We speak of *preconditions* that must hold in order for a given action to be possible. Equally complex can be the results of an action. While there is often a primary or intended result (the one that should lead toward the goal), there often are many unintended ones as well. Thus raising my left arm affects the motion of air molecules; it may bump into a glass and knock it over, etc.

Here is a more telling example: the "jump in the water" action is possible only if there is water nearby, and only if jumping is something the agent can do. Indeed, not any old sort of jumping, but jumping that can actually move the agent into the water (and that might be very hard to specify in detail). And the results are many: the agent is wet, the agent's clothes are wet, agents and object nearby are likely wet from splashing. And if the agent then gets out of the water, it will not be dry right away, nor will its clothes. And wherever the agent steps there will be water for awhile, but not above the agent. Etc.

People have tried to formalize all this sort of thing – and some are still trying. But it seems hopeless to get it completely right. There always seem to be exceptions. So perhaps instead we need to get a basic set of principles that are right most of the time and then have a robust error-correcting process to help us fine-tune particular cases as needed. This is at the frontiers of research currently.

Another technique is case-based reasoning, where the agent memorizes a set of previously encountered cases, and uses those to guide future reasoning. One intriguing idea is to combine this with a kind of perceptual reasoning with images of some sort. A recent paper by Patrick Winston (MIT) is suggestive along these lines: Winston describes installing a table saw with help from a friend who cautions him not to wear gloves when using the saw. Winston is puzzled and then "sees" the point by envisioning what might happen if the glove – with his hand in it – is caught by a spinning sawblade. Whether it really endangers the hand or protects it is not the point. The point is that we can reason like that, very rapidly, using images tracked over time together with notions of causality (and perhaps probability). This is a topic of great importance and almost no research so far. It should also help a great deal even in far simpler cases (e.g., block-stacking) and might lead to a new powerful AI paradigm. Vision and reasoning (and memory) seem like natural partners just waiting to be combined – but the work is challenging.

However, a part of this is reasoning by itself, and we will spend a few weeks on that, solving the Monkey and Bananas problem in the process.