Artificial Intelligence (LISP)

Introduction

Artificial Intelligence (AI) is a broad field, and means different things to different people. It is concerned with getting computers to do tasks that require human intelligence. However, having said that, there are many tasks which we might reasonably think require intelligence - such as complex arithmetic - which computers can do very easily. Conversely, there are many tasks that people do without even thinking - such as recognizing a face - which are extremely complex to automate. AI is concerned with these difficult tasks, which seem to require complex and sophisticated reasoning processes and knowledge.

People might want to automate human intelligence for a number of different reasons. One reason is simply to understand human intelligence better. For example, we may be able to test and refine psychological and linguistic theories by writing programs which attempt to simulate aspects of human behavior. Another reason is simply so that we have smarter programs. We may not care if the programs accurately simulate human reasoning, but by studying human reasoning we may develop useful techniques for solving difficult problems.

Is AI Possible?

Artificial intelligence research makes the assumption that human intelligence can be reduced to the (complex) manipulation of symbols, and that it does not matter what medium is used to manipulate these symbols - it does not have to be a biological brain! This assumption does not go unchallenged among philosophers etc. Some argue that true intelligence can never be achieved by a computer, but requires some human property which cannot be simulated. There are endless philosophical debates on this issue.

The most well known contributions to the philosophical debate are Turing's ``Turing test'' paper, and Searle's ``Chinese room''. Turing considered how you would be able to conclude that a machine was really intelligent. He argued that the only reasonable way was to do a test. The test involves a human communicating with a human and with a computer in other rooms, using a computer for the communication. The first human can ask the other human/computer any questions they like, including very subjective questions like ``What do you think of this Poem''. If the computer answers so well that the first human can't tell which of the two others is human, then we say that the computer is intelligent.

Searle argued that just behaving intelligently wasn't enough. He tried to demonstrate this by suggesting a thought experiment (the ``Chinese room''). Imagine that you don't speak any Chinese, but that you have a huge rule book which allows you to look up Chinese sentences and tells you how to reply to them in Chinese. You don't understand Chinese, but can behave in an apparently intelligent way. He claimed that computers, even if they appeared intelligent, wouldn't really be, as they'd be just using something like the rule book of the Chinese room.

Many people go further than Searle, and claim that computers will never even be able to appear to be really intelligent. There are therefore a number of positions that you might adopt:

• Computers will never even appear to be really intelligent, though they might do a few useful tasks that conventionally require intelligence.
• Computers may eventually appear to be intelligent, but in fact they will just be simulating intelligent behavior, and not really be intelligent.
• Computers will eventually be really intelligent.
• Computers will not only be intelligent, they'll be conscious and have emotions.

We can conclude that, though computers can clearly behave intelligently in performing certain limited tasks, full intelligence is a very long way off and hard to imagine. However, these philosophical issues rarely impinge on AI practice and research. It is clear that AI techniques can be used to produce useful programs that conventionally require human intelligence, and that this work helps us understand the nature of our own intelligence. This is as much as we can expect from AI for now, and it still makes it a fascinating topic!

Some AI Tasks

Human intelligence involves both ``mundane'' and ``expert'' reasoning. By mundane reasoning we mean all those things which (nearly) all of us can routinely do (to various abilities) in order to act and interact in the world. This will include:

• Vision: The ability to make sense of what we see.
• Natural Language: The ability to communicate with others in English or another natural language.
• Planning: The ability to decide on a good sequence of actions to achieve your

goals.
• Robotics: The ability to move and act in the world, possibly responding to new perceptions.

By expert reasoning we mean things that only some people are good at, and which require extensive training. It can be especially useful to automate these tasks, as there may be a shortage of human experts. Expert reasoning includes:

• Medical diagnosis.
• Equipment repair.
• Computer configuration.
• Financial planning.

Expert Systems are concerned with the automation of these sorts of tasks. AI research is concerned with automating both these kinds of reasoning. It turns out, however, that it is the mundane tasks that are by far the hardest to automate.

AI Techniques

There are various techniques that have evolved that can be applied to a variety of AI tasks .These techniques are concerned with how we represent, manipulate and reason with knowledge in order to solve problems.
Knowledge Representation

Knowledge representation is crucial. One of the clearest results of artificial intelligence research so far is that solving even apparently simple problems requires lots of knowledge. Really understanding a single sentence requires extensive knowledge both of language and of the context. For example, today's (4th Nov) headline ``It's President Clinton'' can only be interpreted reasonably if you know it's the day after the American elections. Really understanding a visual scene similarly requires knowledge of the kinds of objects in the scene. Solving problems in a particular domain generally requires knowledge of the objects in the domain and knowledge of how to reason in that domain - both these types of knowledge must be represented.

Knowledge must be represented efficiently, and in a meaningful way. Efficiency is important, as it would be impossible (or at least impractical) to explicitly represent every fact that you might ever need. There are just so many potentially useful facts, most of which you would never even think of. You have to be able to infer new facts

from your existing knowledge, as and when needed, and capture general abstractions which represent general features of sets of objects in the world.

Knowledge must be meaningfully represented so that we know how it relates back to the real world. A knowledge representation scheme provides a mapping from features of the world to a formal language. (The formal language will just capture certain aspects of the world, which we believe are important to our problem - we may of course miss out crucial aspects and so fail to really solve our problem, like ignoring friction in a mechanics problem). Anyway, when we manipulate that formal language using a computer we want to make sure that we still have meaningful expressions, which can be mapped back to the real world. This is what we mean when we talk about the semantics of representation languages.

Search

Another crucial general technique required when writing AI programs is search. Often there is no direct way to find a solution to some problem. However, you do know how to generate possibilities. For example, in solving a puzzle you might know all the possible moves, but not the sequence that would lead to a solution. When working out how to get somewhere you might know all the roads/buses/trains, just not the best route to get you to your destination quickly. Developing good ways to search through these possibilities for a good solution is therefore vital. Brute force techniques, where you generate and try out every possible solution may work, but are often very inefficient, as there are just too many possibilities to try. Heuristic techniques are often better, where you only try the options which you think (based on your current best guess) are most likely to lead to a good solution.

Artificial Intelligence Programming

AI programs can, in principle, be written in any programming language. However, as with any programming task, there are languages that have features that make AI programming easy, and languages that make it difficult. It is therefore useful to go quickly through some of the characteristics of AI programming that influence the choice of language.

Support for Symbolic Computation

First, AI programming involves (mainly) manipulating symbols, and not numbers. These symbols might represent objects in the world, and relationships between those objects - complex structures of symbols are needed to capture our knowledge about the world.

Symbol structures are often represented using the list data structure, where an element of a list may be either a symbol, or another list. For example, (friends jim (joe mary anne)) is a list. Manipulating symbol structures often involves pattern matching, where two patterns which partially specify symbol structures are matched against each other to test for compatibility. For example (friends jim X) might be matched with the above list, where X is a variable which can match anything. AI languages should therefore support flexible list-based data structures and pattern matching.

Support for Exploratory Programming

For many AI programming problems, software engineering techniques such as stepwise refinement and development from formal specifications may not be that helpful. It is seldom possible to give a complete specification of an AI program before building at least a prototype - we just don't understand the nature of the problems well enough. AI programming is therefore inherently exploratory. We develop a program to explore the nature of the problem and domain, and discover good solution strategies as we go. This is not to say that we should abandon good software engineering techniques, just that we cannot rely on them, and that we require an environment that supports more exploratory styles of program development.

Features of a programming language/system that support exploratory programming include: extensibility - the ability to develop special purpose interpreters for solving classes of problems; and interactive development environments - where the programmer can flexibly and interactively test small sections of their program.

The Main AI Languages

The main programming languages used in AI are Lisp and Prolog. Both have features which make them suitable for AI programming, such as support for list processing, pattern matching and exploratory programming. Both are also widely used - Prolog especially in Europe and Japan, and Lisp in the US. This wide use

within the field is another reason to choose Lisp or Prolog for AI implementations.

Lisp can be viewed as the grandfather of functional programming, developed in the late 1950s and based on function definitions. Although somewhat looked down upon by functional programming purists, it remains a useful, powerful and widely used programming language. Lisp uses the list as its fundamental representation for data structures and programs (function definitions), and provides a wide range of built in functions for manipulating lists. (In fact Lisp stands for LISt Processing.) Unlike pure functional languages, data structures may be modified - possibly theoretically undesirable but in practice sometimes very useful! Of course, it is still possible to use Lisp in a purely functional way, but this is not forced on you. Using lists to represent programs allows special purpose interpreters to be easily written, as there is little distinction between prolog and data. Lists also allow complex symbol structures (representing AI knowledge) to be easily manipulated, and pattern matchers can be easily be written to match list structures.

Prolog is a language based on logic. (Prolog = PROgramming in LOGic). In particular, it is based on first order predicate calculus. Writing simple Prolog programs is similar to writing statements in predicate calculus, and `running' them involves setting queries for Prolog to `prove' using its special purpose theorem prover.

Source: http://www.go4expert.com/articles/artificial-intelligence-lisp-t274/