Artificial Intelligence I

Matthew Huntbach, Dept of Computer Science, Queen Mary and Westfield College, London, UK E1 4NS. Email: mmh@dcs.qmw.ac.uk.

Introductory Notes

This is a set of notes introducing the subject of Artificial Intelligence, and discussing it in general terms. Most of the rest of the course will be of a more technical nature. Further printed notes will be provided, but you should buy at least one of the following books:

T.J.M.Bench-Capon *Knowledge Representation: An Approach to Artificial Intelligence* Academic Press (1990). A little simplistic, but short and not over-technical. Covers roughly the same material as will be in the Artificial Intelligence 1 course.

I.Pratt *Artificial Intelligence* Macmillan (1994).A good book if you want a thorough coverage of the subject, but you would have to work hard with it as it's takes a heavy formal approach.

I.Bratko *Prolog Programming for Artificial Intelligence* Addison-Wesley (1986).Unlike the two books above, this approaches the subject of Artificial Intelligence through the Prolog language, rather than introducing Prolog as one aspect of Artificial Intelligence.

P.Flach *Simply Logical* Wiley (1994). Another Prolog-based book, but slightly more oriented towards formal logic. Something of a compromise between Pratt and Bratko.

Introduction and Overview

Philosophers, psychologists, linguists and so on have often sought to find rules which describe human behaviour. A linguist, for example, might want to draw up rules which tell us whether a sentence is correct grammatically or not and what exactly we mean by it. A psychologist interested in human vision might want to find out how we interpret the light patterns that reach our eyes as physical objects in the world. A philosopher interested in logic might ask how it is that given some facts we can reach some further conclusions.

Computer scientists too are interested in drawing up rules. Writing a program which fits some specification amounts to drawing up a set of rules which enable a computer to produce the desired output from the given input.

Artificial Intelligence fills the gap between the scientists of human behaviour and the computer scientists. One thing we know about computers is that given a set of rules written in the programming language of the computer the rules will always be obeyed exactly. So human scientists can test their theories about human behaviour by converting their rules to computer programs and seeing if the behaviour of the computer in executing these programs is like the natural behaviour of a human being, or at least that small subset of human behaviour they are studying. Computer scientists can look at modelling human behaviour as a challenge to their programming abilities: if a person can do something, can we write a computer program which does the same thing?

The name "Artificial Intelligence" is rather an emotive one, though it has become established now and is unlikely to be changed. It may be interesting to debate whether human beings are just very complex "meat machines" or whether it would be theoretically possible to create a real "artificial intelligence", that is a computer program which really does behave as a full human being, but studying Artificial Intelligence does not involve commiting oneself to such beliefs. In fact, one thing that Artificial Intelligence has quite firmly established is that human beings are more complex than some rather naïve analyses have suggested. Predictions that human-like artificial intelligences are soon to be built are often just media sensationalism, and have so far proved wrong.

It is often said that though the space programme which led to the landing on the moon did not in itself lead to a useful end, it spun off much in the way of useful technological development. This appears to be the case with Artificial Intelligence: the real benefits in research in the field come in the spinoffs which advance computer science in general. In particular, artificial intelligence researchers have often faced particularly difficult programming problems, so many of the concepts in programming language design and programming methodology which are now general in computer science have their origin in Artificial Intelligence programming.

So the emphasis of the Artificial Intelligence I course will be from the Computer Scientist's angle, building on what has already been covered in the first year Computer Science courses. There will be a particular emphasis on knowledge representation. You will already have seen that computer programs will to some extent model some aspects of the "real world". A program which stores and recalls data will be able to answer questions which correspond to the objects in the world the data represents. A program which interacts with a user will have some internal representation of what the user sees on the screen (or, as we enter the world of virtual reality, hears, feels, etc). A program which searches for the solution to some planning problem will have a representation of the component parts of that problem. What distinguishes artificial intelligence systems from, say, database systems, is that they not only represent knowledge internally, they are also able to manipulate it and draw conclusions which we would say need "intelligence" to reach.

If the knowledge we are interested in is well-structured, we can use conventional database or data structure techniques to represent it. Human knowledge, though, is not well structured. If we were asked to list all the things we know we could not methodically do it in the way a database could be methodically printed out. If we are to model and manipulate even a fraction of the complexity of human knowledge, we must have some more sophisticated way of doing so than a straightforward data structure of the type we have seen in first year programming courses.

Perhaps the most important form of knowledge representation is predicate logic, which you will already have seen in the Introduction to Logic course. We know that predicate logic is a useful way to store and manipulate knowledge because it was devised by mathematicians and philosophers for this purpose. Predicate logic has the advantage of precision because we can define precisely what we mean by a sentence in logic and define precise rules for drawing further conclusions from sets of sentences in logic. As you have seen from the Introduction to Logic course, it is possible to translate English sentences to logic and vice versa. Furthermore, although logic was devised as a notation to formalise human reasoning, rules for the manipulation of logic can be obeyed automatically by a computer. This leads to the suggestion that logic itself may be used as a programming language.

The language Prolog was devised as a cut-down form of logic that could be used directly as a programming language. A major part of the Artificial Intelligence I course will be learning to use the Prolog language, looking at it both in terms of its relationship to logic and as a general-purpose programming language for Artificial Intelligence applications. Prolog has similarities to the Miranda language you saw in the Functional Programming courses. Both languages are based on the idea of starting with human mathematical notation and rules to manipulate it, and moving down towards implementing it on the machine hardware, rather than the conventional programming language approach of starting with what the electronics of the machine can do for us and building upwards. Both languages emphasise a programming style where the programmer is encouraged to reason about the actual problem and declare a solution in terms of relationships rather than have to think in terms of issuing a stream of instructions to update a bank of store. For this reason, functional and logic languages can be grouped together as "declarative languages".

Logic can be criticised as in some ways unnatural. Intuitively we cluster information together and build links between items. An alternative form of knowledge representation, semantic networks, was devised based on this idea, and we shall give some consideration to it in the Artificial Intelligence I course. Semantic networks are graphs where the nodes represent idividual items or concepts, and the arcs represent relationships between them. Semantic networks have been criticised as lacking in precision – whereas using logic forces us to say exactly what we mean by some English sentence and exactly how we can draw further conclusions from our knowledge, a more intuitive representation has the problem that hidden assumptions in our own thinking become hidden assumptions in our representation. However they can be used for styles of reasoning which are difficult to manage in formal logic. The idea of classes of objects linked in a tree structure using "inheritance" which is central to semantic networks was influential in the development of the object-oriented style of programming and programming languages which is now important in mainstream programming.

A further form of knowledge representation we shall look at briefly is Truth Maintenance Systems. The importance of these systems is that they enable us to draw conclusions based on things we assume to be true, rather than only on things we know to be true as with predicate logic. The reasons why we have drawn some conclusion are stored as part of the knowledge, which enables us to easily make any necessary changes should our assumptions change, either because we have learnt new

information or because we have found that the assumptions we first made lead to conclusions we know to be contradictory and therefore we have to make changes to our assumptions.

Declarativism v. Neural Networks

We note above that the design of knowledge representation systems might be influenced by intuition on how humans think about problems. This leads to a major question in Artificial Intelligence – are we trying to construct something which actually works in the same way that people work by modelling what happens internally in the human mind, or would we be content with something which given a task to do gives us the same output or external behavior as a human would give?

As a good example, the ability to play games like chess is often taken to be a measure of human intelligence. So game-playing programs have traditionally been an important aspect of Artificial Intelligence. The chess-playing programs that do best in beating human players work on a fairly simple principle. All they do is note from the current position all the possible positions that can be made from all possible moves, then all positions that can be made from all possible moves from them and so on to a certain number of moves in advance. All that then has to be done is to search through all the possible positions that can be reached to which of the original moves leads to the best position where it is not possible for the opponent to make a move which leads to a better position for him. There are many variations on this theme but all are based on searching through large numbers (hundreds or thousands) of possible board states. Can we therefore suppose that a similar search mechanism is built into the brain of a human chess-player? There is no evidence that it is, but there is evidence that human chess players make more use of intuition which involves a broader consideration of patterns on the chess board.

Another example involves the grammatical rules of natural language. Linguists have drawn up elaborate rules which tell us whether a sentence in a language is grammatical or not. But is doubtful whether such rules are actually built into our minds. Certainly the experience of anyone learning a language in the traditional way using declensions and conjugations (in particular the classical languages such as Latin and Ancient Greek have usually been taught in this way) will know how complex and unnatural these rules seem. When we speak our own language, we know from intuition whether what we are saying is grammatical, in fact it is often quite hard to sit down and work out rules for it. We might use conjugation tables today to understand Latin, but surely the ordinary inhabitant of ancient Rome did not think in these terms when he or she spoke, any more than we are aware of the rules for English which may be found in academic journals of linguistics when we use the language in conversation.

The doubt over the validity of rule-based and search-based Artificial Intelligence has led to interest in an alternative form of Artificial Intelligence involving the connection of simple components in a network resembling the way neurones are connected in the human brain, thus this sort of system is termed a neural net. Each component in a neural net will receive signals from its inputs, and if these signals reach some level send signals on its outputs. Thus a neural net given an input will cause signals to move around the network until an output is produced. The network is modified by adjusting the levels at which signals are received and sent according to whether the output is correct for the given input, and will move closer to a system which always gives the desired output for any input as it is given more examples.

Although some success has been achieved with neural networks, at their best they reproduce the problem with human behaviour – we can produce the correct response to a given input, but we cannot explain how we arrived at it, it is simply "intuition". Similarly, a neural net will have no specific rules for specific purposes which we can investigate and modify if desired or use as an explanation. Rather the information it stores is distributed throughout the network in a way which cannot be divided up and analysed.

To give an example, it has been suggested that neural nets could be used for personal decision making, such as a bank deciding whether to grant someone a loan. An example closer to home would be to train a neural net system to simulate a university Admissions Tutor. The input to the neural net could be the qualifications of applicants, the output an accept/reject decision. The system could be trained by giving it examples of real decisions and aletering it so that the decisions it makes match the real decisions made. We would successfuly have built an artificial Admissions Tutor, but we would not know what rules it was applying because we would have no internal representation of the

rules. If the real admissions tutor used for training had a bias we wanted to correct, say placing an overemphasis on one subject and an underemphasis on another, we could not easily do so on the artificial one. If, however we built a rule-based system in which the rules the real admissions tutor used for selection were programmed in directly, it would be a simple matter to change the programmed-in rules to correct the bias. Thus an important aspect of rule-based systems is that they can give an explanation for decisions they arrive at.

So this course will not cover Artificial Intelligence from the neural net approach. We will use the "traditional AI approach" as this fits in best with our aim of extending what you have already learnt in programming and logic courses. The systems we build, though not necessarily working in the way a human brain actually works, are considered "intelligent" because they perform tasks of reasoning of the sort people can perform. As the systems are actual computer programs we know how they work, and thus however clever they may seem, we can break down their operations into simpler steps and know exactly why they are behaving as they are. The traditional approach is referred to more technically as "declarativism", meaning that the computer programs we write have identifiable symbols which represent concepts in the world it is modelling.

Microworlds v. Common Sense

One of the results of early AI work was the realisation that tasks which are difficult problems for human intelligence may be easy for artificial intelligence, and vice versa. We regard a human being who can perform tasks of mental arithmetic quickly as intelligent, whereas we have now become used to calculators which can quickly do arithmetic. It is fairly easy to write computer programs to solve the sort of puzzles that make up IQ tests, but extremely difficult to write programs that can do things human beings, even those we would regard as of well below average intelligence, can do naturally. For example, writing computer programs to understand simple sentences in human languages, or to interpret simple pictures, is a difficult task which has by no means been fully solved yet.

The reason comes down to the fact that computers are essentially machines for obeying rules speedily and accurately. Human beings work by intuition in a way that psychologists still do not fully understand. We find that if we are asked to consciously memorise more than a small number of items or rules we cannot do so, but we can when solving problems which cannot be broken down into simple rules bring to mind all sorts of information from our lifetime's experience which may be of assistance.

As an example, let us consider the analysis of two simple pieces of English:

"I stood on the table to paint the window. It broke and I fell through onto the flowerbed below"

"I stood on the table to paint the window. It broke and I fell through onto the carpet below".

This is a simple example of one of the ways in which human language is ambiguous – when we use pronouns such as "it" we do not always know which object is meant by them. In both cases "it" could be the table or the window. If the sentences were part of a larger piece of English "it" could refer to something else first mentioned in an earlier sentence; if the second sentence in the example were, say, "I should not have done it, as I damaged the table", "it" refers to the action of the first sentence rather than to some specific object (the old song "If I said you had a beautiful body, would you hold it against me?" makes deliberate use of this ambiguity!).

However, in the first example we will naturally assume that "it" means the window, in the second that "it" means the table. The reason for this is that we know that flowerbeds are more likely to be the other side of windows from tables, and carpets more likely to be below tables. From this it can be seen that analysis of natural language sentences requires common-sense knowledge of the world. If we wrote a computer program to interpret natural language statements and store the information required, perhaps as sets of statemements in logic, it would be a difficult task to ensure it always interpreted "it" to mean the thing that a human would naturally interpret it to mean. In fact it would be almost impossible to devise rules to cover every possible situation.

As another example, consider the fact that a fancy restaurant may have a sign saying "Ties must be worn", while an escalator may have a sign saying "Dogs must be carried". The word "must" in these two sentences is used to mean different things (a recent paper in a linguistics journal discussed precisely this problem of what exactly is mean by "must" in all the ways we use it). We could

interpret the second sign as saying that people are only allowed to use the escalator if they are carrying a dog, but we know that would be ridiculous. Similarly we know it would be ridiculous to interpret the first sign as "if you happen to be carrying a tie with you, wear it, otherwise come in tieless". Again, it is only social context and our common-sense knowledge of the world as a whole which enables us to correctly interpret the ambiguity. It would be difficult to program in all the knowledge that would be required so that in all situations like this a computer language-understanding situation gave the correct meaning to "must", and we are left with the problem of how exactly this knowledge would be represented and accessed.

In both these cases the problem is one of openness: we cannot easily draw boundaries between the knowledge that is necessary to resolve the ambiguity and everything else we know. Yet human beings do it without thinking unless their attention is drawn to the problem of ambiguity. It is things like this that are done without thinking that seem to be the really difficult things to capture when trying to build an artificial intelligence.

For this reason, much practical AI work concentrates on "micro-worlds", that is situations where we know precisely everything there is to know: all the objects in the situation and all the rules governing them. Games like chess are microworlds, because we they operate to exact rules, we know all the pieces that occur in them and we never have to deal with any unexpected intervention from the outside world. Dealing with a real combat situation though(for example a program designed to assist a general making troop movement plans) is not a micro-world because we can at best operate with rules-of-thumb while always being aware that the unexpected might happen.

From this it follows that a legitimate criticism of AI is that in concentrating on micro-worlds it is not really looking at intelligence in the sense of modelling human behaviour – it is simply about solving tricky problems. The problem is perhaps more with the name "Artificial Intelligence". We can surely regard it as legitimate and useful to look into ways of writing computer programs which can solve tricky problems, whether or not those computer programs can be considered building blocks towards a larger goal of building "artificial human beings".

However, following from these criticisms there has been a growing trend in AI work to try and capture some aspects of "common-sense" reasoning. For instance, standard predicate logic can be criticised for not being able to capture the fact that we often make default assumptions. We know, for example, that if we have the facts `p(x)→q(x)` where `x` is a variable, and `p(a)` where `a` is a constant we can conclude `q(a)`. As a typical example, suppose we say "all Computer Science students take Logic" we can express that by `CompSci(X)→Logic(X)`, then if we have the fact "John is a Computer Science student", expressed by `CompSci(John)`, we can conclude `Logic(John)` i.e. "John takes Logic". It is, though, a well-known fallacy that one can conclude `p(a)` from `q(a)` and `p(x)→q(x)`. If we have the fact that "Bill takes Logic" or `Logic(Bill)`, we can't conclude `CompSci(Bill)` meaning "Bill is a Computer Science student", because it may be the case that Maths students take Logic as well, expressed by the fact `Maths(X)→Logic(X)`.

People often make this false step in an argument though, and we cannot always just dismiss them as illogical. It may be the case that all Computer Science students take Logic, but there are one or two students from other departments who also take Logic. It would therefore be a reasonable common sense conclusion if we have the fact "Bill takes Logic" to say that Bill is a Computer Science student unless we have some reason to conclude he is not. Ordinary predicate logic cannot make this sort of jump, but there has been research on variations of logic that can. In this case there is an element of probability involved: the assumption is a reasonable common sense one under the conditions given above, but it would no be reasonable if we also knew that Computer Science students formed only a small fraction of those taking Logic.

Related to the idea of microworlds is the "closed world assumption", that is the assumption that the data we have represents all that can be known. This enables us to assume that if we cannot prove something true it must be false. This may be reasonable in a microworld, but is less so when we are reasoning about the real world. For example, if the only rules we have about who takes logic are `CompSci(X)→Logic(X)` and `Maths(X)→Logic(X)`, we can conclude if we don't have the fact `CompSci(Fred)` or `Maths(Fred)` that `Logic(Fred)` is false (i.e. Fred does not take Logic). This conclusion will always be correct under the closed world assumption, but the real world is not closed, and in practice there may reasons that we do not know about for a student to take Logic other than he or she is a Maths or a Computer Science student. Of course, we can still make the assumption that Fred does not take logic, but like the above case it is an assumption we hold only until we obtain further knowledge.

Principles of Knowledge Representation

Assuming that we are concerned with declarative AI, and recognising that any AI system must in a sense be a microworld, since the only completely adequate model of the complete universe would be the universe itself, we can consider what particular properties a knowledge representation system should have.

As a minimum requirement, there should be some correspondence between the knowledge representation system and the knowledge it is attempting to represent. As we said, the defining feature of declarative AI is that in the computer system we build there are identifiable symbols representing the things being modelled. This property can be referred to as *metaphysical adequacy*.

But a further property is that our knowledge representation system must be able to represent at least all the knowledge we might ever want to represent. A knowledge representation system would be inadequate if there was no way of drawing a distinction between two different states which are recognisably different in the microworld we are modelling. This property can be referred to as *epistemical adequacy*.

We have also already mentioned, in connection with the problems with neural networks, that we want any representation system to have the property that it can express the reasoning that is gone through in solving a problem. This can be referred to as *heuristic adequacy*.

Clearly, also any representation must be capable of being manipulated in a computer to arrive at solutions to problems. This property can be referred to as *computational tractability*. For example, natural language is not computationally tractable as it requires a human being to fully interpret it correctly. The other side of this property is *human clarity*, that is a knowledge representation system should be capable of being presented to a human programmer or user of the data in such a way that it is capable of being understood. For example, the raw data in a computer expressed as a hexidecimal dump of its memory may be computationally tractable, but fails totally on human clarity.

We need to ensure that no expression in the knowledge representation system is capable of being interpreted as representing more than one possible state in the world we are modelling, that is it should be *unambiguous*. Again, natural language is unsatisfactory as it has many cases of ambiguity, such as those given as examples above. Note the difference between this proprty and epistemic adequacy – a system is epistemically inadequate if it is <u>impossible</u> to use it to mark a distinction between two different states, it is epistemically adequate but ambiguous if it is possible but not necessary to mark a distinction in representation of two different states. For example, in natural language it is always possible to express ourselves in more detail so as to resolve ambiguities.

Ideally any knowledge representation system should be *formally equivalent* to the microworld it is trying to model. Two systems can be defined as formally equivalent if:

1) For each distinct possible state in one there is exactly one distinct possible state in the other.

2) For each possible move or change of state in one there is exactly one move or change of state in the other.

Haugeland, in the book referenced in the "Further Reading" section below, gives a good example of some of the problems regarding representation, noting that formal equivalence is not always obvious. He suggests considering the following game:

1) The game consists of two baskets, one black, one white, and thirty three chips marked with a letter from the range A-G and a number from the range 1-7.

2) Initially, all the chips are in the white basket except for one which is in the black.

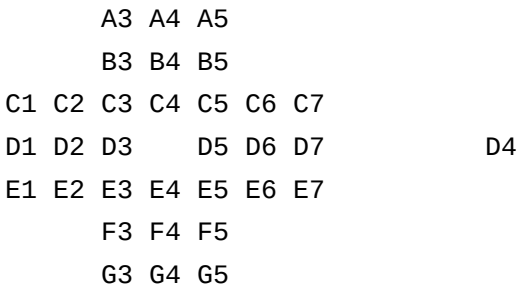
3) A move consists of exchanging two chips from the white basket for one from the black with the restriction that:

a) *Either* all three chips must have the same letter and sequential numbers *or* they must have the same number and sequential letters

b) *And* the middle letter or number in the sequence can't be on the one chip going from the black basket to the white.

The chips are labelled A3, A4, A5, B3, B4, B5, C1, C2, C3, C4, C5, C6, C7, D1, D2, D3, D4, D5, D6, D7, E1, E2, E3, E4, E5, E6, E7, F3, F4, F5, G3, G4, G5, with D4 being the one initially in the black basket.

The game may sound obscure, but when it is presented in a visual two dimensional way, it becomes immediately apparent that it is equivalent to the game Solitaire:



Artificial Intelligence Programming

The conventional software engineering approach to programming emphasises starting with requirements, developing formal specifications, breaking the problem into parts and coding at a late stage in the development. Programming Artificial Intelligence systems, however, does not fit so easily into conventional software engineering models. By their very nature, Artificial Intelligence problems cannot be formally specified. The correctness of an AI system is measured not by its performing according to some formal specification, but by its closeness to human performance. This leads to an approach to programming described as "exploratory" or "prototyping". This means that programs are developed in a less formal way than with conventional software engineering, but may be regarded as experimental prototypes rather than semi-completed products. An AI program may be tested against a range of human examples, and if necessary modified so that it performs as a human would on those examples. Some modifications may prove successful in improving performance (as measured by closeness to human performance) over a variety of examples. Others may prove to be useful only in one or two examples, and to degrade performance in others, and so be abandoned. However, as modifications accumulate, a program structure may become complex as modifications interact with other modifications. Eventually, the point may be reached where the program is abandoned but those lessons learnt from building it are kept and used to build a cleaner program. This process may go through several cycles, each cycle building a better prototype. This has been described as the "Run-Understand-Debug-Edit" (RUDE) approach to programming.

If we define Artificial Intelligence as those aspects of programming which of necessity are programmed using a RUDE method rather than a more formal software engineering approach, then chess-playing, for example, which previously we suggested was only doubtfully part of Artificial Intelligence, can be seen as firmly part of AI. We may now be able to write programs which can beat all but the most exceptional human chess-players. But we cannot write a formal specification which defines the best next move to make from any given position. Improving our basic chess-playing model involves adding variations to the underlying search procedure which are tested in an experimental way by running a complete program against human chess-players.

It might be noted that the software engineering process of producing a program from formal specifications is itself an AI problem, otherwise we would not have to employ human programmers to do it, we could do it automatically. "Automatic programming" is itself a branch of AI. Note that this is a term which has changed its meaning as time has passed. When human programmers wrote in machine code, the compilers which translated from the first high-level languages, such as FORTRAN, into machine code were described as "automatic programming" systems. Nowadays we reserve the term for those systems which produce executable programs from non-executable specifications. Those specifications may be in logic or some other formal specification language, but another branch of automatic programming "synthesis from examples" produces programs from an informal specification consisting of examples of inputs and the expected outputs for these inputs.

As mentioned previously, AI programming has often made use of declarative languages which attempt to be closer to a human way of reasoning about problems than conventional programming languages. The exploratory nature of AI programming means that it is essential that AI programs retain their understandibility and do not get bogged down in the details required to manipulate conventional computer store. Much early work in AI was done in the language Lisp, a forerunner to the functional language Miranda. The argument for using Lisp was that the use of lists for storing data and the emphasis on symbol manipulation was more appropriate for AI work than the arrays

and numerical data of scientific/engineering (principally FORTRAN) and data processing (principally Cobol) languages then in use.

Prolog, now the preferred language for many AI programmers, has the list processing and symbolic manipulation qualities of Lisp. The ideal of the developers of Prolog was that it would be simply machine-executable logic, which would mean that a Prolog program would be its own specification. Thus an AI program developed in an exploratory way would still be fully specified. The equation ALGORITHM=LOGIC+CONTROL was suggested, suggesting that the programmer need simply specify logic relationships between the data and the control would be a separate aspect, managed by the computer, which manipulated the data according to the rules expressed in the logic. It was suggested that imperative languages are confusing because they mix up the logic aspects of programming with the control aspects.

We shall in this course investigate the extent to which Prolog as a practical programming language meets these ideals. One important aspect of the separation of logic and control that has an important place in AI programming is the distinction between language and meta-language. Conventional programming makes a separation between the data of the program and the program itself written in a language for manipulating that data. AI programming often makes a three fold separation into data, rules for manipulating the data, and rules for saying how the rules to manipulate the data are to be used. A meta-program is this top-level set of rules: a program which treats a program as data.

Further Reading

The following books, which may be found in the QMW library, all serve as introductions to Artificial Intelligence in a way that emphasises the psychological and philosophical connections of the subject. They can be considered as forming part of the "Sussex" approach to the subject (the authors of all three are associated with the University of Sussex Cognitive Studies programme which specialises in these aspects of Artificial Intelligence), and might be read as a useful contrast to the more technical Computer Science approach to Artificial Intelligence which will be given in the rest of this course.

Margaret Boden *Artificial Intelligence and Natural Man* Harvester Press. An influential work, but now rather dated, the second edition (1986) is slightly updated from the first (1977).

Mike Sharples et al. *Computers and Thought* (1989) MIT Press. Based on notes originally used for an introductory course in AI at the University of Sussex specifically designed for students without a traditional Maths or Computing background. At a slight disadvantage due to its heavy use of the Pop-11 language, which is not widely used outside Sussex where it was developed.

Masoud Yazdani *Artificial Intelligence: Principles and Applications* Chapman Hall (1986) A collection of introductory articles by various authors covering a wide range of subfields within AI, including natural language processing, vision and robotics under the general heading of "Applications", and articles on the philosophical and social implications of AI.

The following three books give strong arguments on philosophical grounds on the case of whether it is possible to build an artificial human-like intelligent system (known as the "strong AI" thesis). The first is for, the second against, the third adopts neither position:

J.L.Pollock *How to Build a Person* (1989) MIT Press

R.Born et al *Artificial Intelligence: the Case Against* (1987) Croom Helm

J.Haugeland *Artificial Intelligence: the Very Idea* (1985) MIT Press

I particularly recommend the Haugeland book as a very readable expansion of some of the ideas covered only briefly in this set of notes.

A collection of articles which argues on similar lines, but concentrates on the neural network approach to Artificial Intelligence, rather than the declarative approach which dominates in the other references, is:

S.R.Graubard *The Artificial Intelligence Debate* (1989) MIT Press

The issues in the last section on Artificial Intelligence programming are discussed further in:

D.Partridge *Engineering Artificial Intelligence Software* Intellect Press (1992).