POP-11 PROGRAMMING FOR ARTIFICIAL INTELLIGENCE

Mike Burton
Nigel Shadbolt

POP-11 PROGRAMMING FOR ARTIFICIAL INTELLIGENCE

Mike Burton
Nigel Shadbolt
University of Nottingham

ADDISON-WESLEY
PUBLISHING
COMPANY

Wokingham, England · Reading, Massachusetts · Menlo [illegible]
Don Mills, Ontario · Amsterdam · Bonn · Sydney · Singapore
Tokyo · Madrid · Bogota · Santiago · San Juan

Cover graphic by kind permission of Apollo Computer Inc.
Photo-typset at the University of Nottingham.
Printed in Great Britain by The Bath Press.

British Library Cataloguing in Publication Data

Burton, Mike
Pop-11 programming for artificial intelligence.–(International computer science series)
1. Artificial intelligence–Data processing 2. Pop-11 (Computer program language)
I. Title II. Shadbolt, Nigel III. Series
006.3'02855133 Q336

ISBN 0-201-18049-9

Library of Congress Cataloging in Publication Data

Burton, Mike, 1959–
POP-11 programming for artificial intelligence.

(International computer science series)
Bibliography; p.
Includes index.
1. Artificial intelligence–Data processing.
2. POP (Computer program language) I. Shadbolt, Nigel, 1956– . II. Title. III. Title: POP-eleven programming for artificial intelligence. IV. Series.
Q336.B88 1987 006.3'028'55133 86-32059
ISBN 0-201-18049-9

INTERNATIONAL COMPUTER SCIENCE SERIES

OTHER TITLES IN THE SERIES:

Programming in Ada (2nd Edn.) *J G P Barnes*
Computer Science Applied to Business Systems *M J R Shave and K N Bhaskar*
Software Engineering (2nd Edn.) *I Sommerville*
A Structured Approach to FORTRAN 77 Programming *T M R Ellis*
The Cambridge Distributed Computing System *R M Needham and A J Herbert*
An Introduction to Numerical Methods with Pascal *L V Atkinson and P J Harley*
The UNIX System *S R Bourne*
Handbook of Algorithms and Data Structures *G H Gonnet*
Office Automation: Concepts, Technologies and Issues *R A Hirschheim*
Microcomputers in Engineering and Science *J F Craine and G R Martin*
UNIX for Super-Users *E Foxley*
Software Specification Techniques *N Gehani and A D McGettrick* (eds.)
Data Communications for Programmers *M Purser*
Local Area Network Design *A Hopper, S Temple and R C Williamson*
Prolog Programming for Artificial Intelligence *I Bratko*
Modula-2: Discipline & Design *A H J Sale*
Introduction to Expert Systems *P Jackson*
Prolog *F Giannesini, H Kanovi, R Pasero and M van Caneghem*
Programming Language Translation: A Practical Approach *P D Terry*
System Simulation: Programming Styles and Languages *W Kreutzer*
Data Abstraction in Programming Languages *J M Bishop*
The UNIX System V Environment *S R Bourne*
The Craft of Software Engineering *A Macro and J Buxton*

Preface

This book serves two purposes. In Part One we provide an introductory course in the programming language POP-11. In Part Two we show how POP-11 can be used to develop artificial intelligence (AI) programs. We do this by building parts of classic AI programs and discussing their relevance to the discipline as a whole. The book should thus be taken as a course in *practical* AI.

The course, as presented here, has grown out of our experiences in teaching AI to psychology undergraduates. In POP-11 we found a language which is sufficiently easy to learn and at the same time has sufficient power to build genuinely interesting programs. While POP-11 has a great deal of documentation associated with it, we felt the need for a structured text. Therefore we have aimed to produce a course in the language rather than a complete description of its full capabilities. No programming experience is assumed and though, of course, we had psychological applications in mind when writing, we believe that the text will be of use to the increasing number of people turning to computer modelling in many fields.

In our AI chapters we hope to plug another gap. Most texts on AI provide a coverage of *what* has been done, without saying *how* the programs can be realized. In building small, but real, AI programs we take the reader through the relevant steps and end up with working programs. This is what we mean in saying that we aim to provide a course in practical AI.

Almost everyone who uses POP-11 will do so as part of the POPLOG system. POPLOG is a programming environment consisting of three languages (POP-11, PROLOG and LISP), an editor (VED) and a great deal of on-line documentation. The documentation ranges from very simple introductory articles, to formal descriptions of the internal workings of POP-11. In Appendix C we provide readers with an outline guide to POPLOG and point to on-line documentation which may be useful.

Many people have contributed to this book in a variety of ways. First of all we must thank Roger Henry and Hugh Smith for introducing us to POP-11 in the first place. These two have commented on early drafts of the book, as have Vicki Bruce, Aaron Sloman and a number of anonymous reviewers. It goes without saying that we claim responsibility

for any blunders still present. Penny Lightburn provided willing secretarial support, and Simon Plumtree gave sound editorial advice. In the final stages, George Paechter was most helpful in performing the typesetting.

Perhaps our final word of thanks should go to our students, who have never been slow to criticize any flaw in our didactic style. Their criticism has contributed substantially to this book.

Mike Burton and Nigel Shadbolt
July 1986

Contents

Chapter 1 Artificial Intelligence and Computational Modelling

1.1 Introduction

Thirty years ago a small conference was held at Dartmouth College in the United States. The delegates included computer scientists, psychologists, linguists and philosophers. They met to discuss the computer simulation of intelligence. In 1956 this seemed no more than a dream. Computers were in their infancy, the hardware and software was primitive. No one understood the nature of what was to be simulated. The field of enquiry did not even have a name. By the time the conference ended one thing at least had changed; John McCarthy had coined the term 'Artificial Intelligence'.

Today rather more has changed. Computers are thousands of times more powerful whilst the cost of hardware continues to plummet. The programs running on today's machines are hugely more sophisticated than their distant forbears. Furthermore, we now understand rather more about the processes which support intelligent behaviour. These developments have, in part, contributed to the emergence of artificial intelligence (AI) as a science in its own right.

This book is aimed at students who need to understand the process of AI modelling, and in particular cognitive modelling. It is divided into two parts. The first is an introduction to programming in POP-11. It provides a course in the 'nuts and bolts' of the language without much reference to AI applications. It is not a complete and exhaustive description of the language. Rather the aim is to provide an understanding of the most important concepts in POP-11.

The second part presents POP-11 as an AI modelling tool. Various AI techniques and applications which have captured the imagination of the academic community, are discussed, and aspects of these are presented as examples, exercises and projects.

The emphasis throughout is on POP-11 as a tool for AI modelling, not as an abstract formal construction. For this reason the book will also be of interest to those in industry and commerce exploring the possibilities of AI for the first time.

1.2 AI in context

As hardware and software developed through the late fifties and early sixties, AI work gathered momentum. This has resulted in a proliferation of applications and techniques. Application areas include vision, natural language understanding, robotics and expert systems. AI techniques comprise methods for searching 'problem spaces', planning sequences of actions to solve problems, methods of reasoning and deduction, and techniques for representing knowledge. Any particular application will typically call on numerous AI methods and techniques. Historically a proportion of AI research has been application driven whilst other work has looked at extending the available methods and techniques.

One of the earliest application areas was machine translation. Early optimism that a simple solution could be found was soon dashed. It was not sufficient to look up word by word a passage of Russian to generate an English equivalent. Much more substantial knowledge was required by the system. This included ways of representing the rules of syntax and semantics (rules of grammar and meaning). The principled inclusion of knowledge into a system is a distinguishing feature of AI applications. Moreover, such applications require the development of new methods and techniques. Research into machine translation, for example, led to many new techniques for parsing – the systematic analysis of strings of symbols into constituents according to a set of grammatical principles.

Many application areas attempted to integrate various AI subsystems, for example, visual and robotic systems. The famous SHAKEY system, a robot developed at the Stanford Research Institute, was able to perform elementary visual analysis, and to work out ways of moving in and acting on its environment. Another integrative project was Winograd's (1973) SHRDLU system, a simulated robot with problem solving abilities which included some natural language understanding.

Such systems might appear very humble given the expectations generated by Hollywood stars such as HAL, C3PO and R2D2! However one of the most enduring lessons of AI has been to increase our respect for 'mundane intelligence'. While the solutions to artificial 'perception' and 'comprehension' are still far from satisfactory, many activities which we prize as intellectual the machines have found easy. Playing chess, solving problems in crypto-arithmetic, solving differential equations are all behaviours which can now be accomplished by machine. Why is this? Much of it has to do with a property of these domains sometimes called the 'closed world assumption'.

AI has progressed largely inasmuch as it has restricted itself to 'closed worlds' or 'micro-worlds'. To illustrate a closed world, consider the Tower of Hanoi problem (Figure 7.1, page 84). In this closed world we have three posts and a set of discs with holes in them, each disc has a different radius. At the start of the problem all of the discs are on one

post, each disc resting on the one just bigger than it. The task is to move all of the discs to one of the other posts. Only a single disc may be moved at a time and all the other discs must be on one of the posts. At no time during the process of solving the problem may a disc be placed on top of a disc that is smaller than it. The third post can be used as a temporary resting place for the discs. This problem gets increasingly difficult the larger the number of discs used.

The domain described has a fixed starting point and a determinate goal state. It has a fixed number of rules which specify how moves can be made. In fact one can, in principle, enumerate all the possible moves in any Tower of Hanoi problem. What one needs is a systematic way of searching this set of moves (or 'problem space') to find the goal state. Much of the power of AI resides in providing ways of searching only that part of the space which contains moves likely to provide a solution.

As we can see, closed worlds in problem solving have an enumerable set of states, well understood start and end states, and a finite set of rules or legal actions which will move one through the problem space. Closed worlds were the starting point in many successful AI applications. Winograd's SHRDLU system had a small number of objects to manipulate, a small repertoire of actions, limited methods of planning, and understood only a 'restricted' set of the English language. Restricted in the sense that the grammar accounted for a small set of the permissible sentences in English and the meanings of the terms were fixed.

Once we move outside a micro-world and abandon the closed world assumption things become very much harder. In the generalized problem of visual recognition we cannot assume, as early systems did, that all objects in the world are rectilinear, that lighting is uniform, that objects are stationary and so on. The general case is 'recognize scene X'. This type of task is one we perform without much conscious reflection – this is 'mundane intelligence'. AI has demonstrated that we should respect it.

What makes the generalized cases so hard? We have already hinted at some of the problems – these are simply the converse of closed world properties. There are many objects in the domains of interest, some of which will not have been experienced before. The relationships and properties of these objects are numerous and vague. Contingencies or rules in the domain do not always hold reliably – there may often be exceptions to any rules that do exist. Faced with an 'open world', how do we cope? How do intelligent systems succeed in dealing with the natural environment? This brings us to another distinction which has become evident in the course of AI research. The contrast between so-called 'weak' and 'strong' methods.

Much of the early work in AI was aimed at developing a battery of 'weak methods'. These methods were general mechanisms which could be applied to numerous domains – they were not 'strongly' tied to the particulars of a domain, nor dependent on strong assumptions about knowledge

of that domain. Good examples of this are some of the search algorithms described in Chapter 7. In contrast, methods which utilize specific knowledge about the domain are known as 'strong AI methods'.

The search for a set of weak methods capable of providing the general foundations of intelligent behaviour was in the end frustrated. Weak methods work reasonably well in closed domains but not on real world problems. When we solve problems and act on our environment we use a great deal of knowledge about the world; how it is structured, what actions are appropriate in what circumstances, what exceptional cases might arise. In short, our problem solving is immensely context determined. There seems to be no universal theory of problem solving or knowledge representation which underlies everything else.

The lessons of AI have been important in shaping our understanding of the character of human intelligence. Nevertheless, we must beware of expecting too much. AI is furnishing a tool-box of methods and techniques which provide ways of exploring cognitive processes. It is not about to deliver the millennial Theory of Psychology neatly wrapped and packed.

The principal research tool which AI offers psychology is that of computational modelling – models of the possible processes of cognition. And it is to the question of models and modelling that we now turn.

1.3 Commonsense models

What do we mean by a model? A good place to start to answer this question is to look at our commonsense notion of a model. What does one think of when one hears the word model? Possible candidates include: the model aeroplanes made by children; the scale models made by architects for display to clients; the balsa wood bridges that school children make as part of their school physics curriculum. These three models fulfill different purposes. Model aeroplanes are meant to be played with, the client can understand the layout of his new housing estate by looking at the scale model, and the bridges are built only to be destroyed by hanging weights from them in order to test their strength.

What characteristics do these three disparate examples share which make them all candidates for our normal use of the term model? They are all representations of the modelled objects, they maintain certain of the original's characteristics and not others. The decision as to what is and is not represented depends on what the model is to be used for.

Consider the architect's scale model. The characteristics it will share with the final estate are quite easy to name. The relative distances between landmarks will be the same in model and world; buildings will retain the same relative positions; hills and depressions will share the same relative heights and the same slopes. This model then, serves the purpose

of allowing one to inspect the estate at a glance, and to establish the relative positions of various elements.

Now consider the possibility that we might use the model for another purpose. Suppose we want to build another house on some unoccupied site. The model will provide a certain amount of information, e.g. how far the site is from other buildings, and so forth. However, we certainly cannot start construction on the basis of this information. Certain aspects of the world will not be represented in this model, e.g. where underground power cables lie, what the soil type is on our site, etc. In fact, an altogether different model would be needed for this project, or perhaps a collection of different types of models.

The moral of the example is simply this: there are often very many different models available (or possible) as representations of the same modelled thing. Which model one decides to use, or build, depends entirely on what one wants to use it for. A pen with a ruler lain across it may be a perfectly good model of an aeroplane for some purposes, whereas for different purposes one may need a model which looks and flies like the real thing. There is nothing magical about a model – it simply represents certain aspects of something else.

So far we have only considered physical models – but a model does not have to be a physical object. A model can be made of words, for instance. Consider the following model of a train.

> A train is a long tubular vehicle which travels along iron rails at speeds of 0 to 125 m.p.h. A train may carry passengers who may embark and alight only at railway stations.

This model of a train may be adequate for certain purposes. For instance, if we wish to know how fast trains travel we can consult this model, and discover that the range is 0 to 125 m.p.h. However, just as with the models mentioned above, this one only has a limited usefulness and will not help us if we want to know how to design a faster train.

All of the models considered so far are models of the physical characteristics of objects, they are **structural** models. There is another, very important type, and this is called a **functional** model. An easy (though rather oversimplified) way to think of this distinction is that a structural model represents what something *is*, whereas a functional model represents what something *does*. Consider the following model of a thermostat for instance. This is the type of device used in the boiler of a central heating or hot water system.

Step 1. Check the temperature of the water.

Step 2. If the temperature is above the required level, then do nothing.

Step 3. If the temperature is below the required level, then apply heat to the boiler.

Step 4. Start again at Step 1.

This is an interesting model of a dynamic process – one which changes all the time. The boiler operates in the following way; if after a check the temperature is found to be above the required level, the mechanism does not apply heat – thus allowing the water to cool down. If, on a subsequent check, the temperature has fallen below the criterion level, then it is heated up. The temperature inside the boiler then, is not constant, but constantly fluctuating around the criterion temperature. The amount of this fluctuation will depend upon how long it takes to go through the four steps, how quickly water cools down, how much heat is applied in Step 3, and so forth. These are not values that we know, but nevertheless the model does provide insight into the workings of a simple thermostat. Moreover, it includes no information about how a thermostat is built, what it looks like, or how its parts are connected to one another. This is a functional model – it represents the functions rather than the structure of a thermostat.

Let's take stock of what we have learnt by considering various simple sorts of everyday model. A model is a representation of something, it preserves certain characteristics of the original, whilst ignoring certain others. Exactly which characteristics are preserved is left to the builder or user of that model. A model can be a physical object, or a more abstract representation – such as a description in ordinary English. Furthermore, the thing modelled can be a physical object, or some abstract process.

When we talk of 'formal models' in cognitive science and AI we are using the same underlying ideas we have discussed here. The difference is that we wish to be very precise about what each part of the model means. We will look in more detail at how formal models aim to do this.

1.4 Formal models

Models are essentially representations. To understand our models or representations we must be clear about the following:

1. What is the medium of representation?
2. What is the original object we are trying to represent?
3. What aspects of the medium are doing the representing?
4. What aspects of the original do we want to model?
5. What are the precise correspondences between elements of the model and the original?

The modelling mediums we have looked at so far have been various, ranging from physical components in the scale model, to explicit languages in the case of the thermostat and the train. In each case the medium of modelling or representation possesses certain constituents which are

placed into correspondences with certain features of the object or process being modelled.

Now, we can regard the modelling medium as having a structure itself. The clearer and more precise this structure the better able it will be to represent in a clear and precise manner those features of the original which we wish to represent.

A modelling medium with which we are all familiar is the language of simple mathematics. Let us consider a very elementary formal model – a simple algebraic model. Suppose we take the expression '$x = y/z$'. What are the constituents of this expression? There are the variables x, y and z which we know, via the conventions of this medium, can take ranges of values. There is the concept of division and the concept of an equivalence relation.

This formal expression can serve as a model. This is done by providing a mapping between the things in the medium and the objects in the real world we wish to model. Let us suppose that x is the current value of a bicycle, z is the age of the bicycle and y is the price of the equivalent bicycle brand new. We can propose the algebraic expression as a model for a property of the real world. Namely, a relationship between the age, value, and new price of a particular bicycle. The older the bicycle, the less value it has as a proportion of its price brand new.

Now we have no idea whether this is a good model, whether it reflects a real underlying relationship between the quantities. That is for the model builder to demonstrate by using the model against known data. It is likely that this model suffers from all sorts of weaknesses – it may oversimplify, it may leave out crucial features by not including enough variables. However, it remains a model, whether good or bad.

A further interesting feature of formal modelling mediums is that we can use them and interpret them in very different ways. Indeed in the extreme case we can use the same construction in the medium of representation to represent something else entirely. Our simple algebraic model could be used for instance to model the effect of some drug. Let x be the level of the drug in the bloodstream, y represents the amount of the drug administered and z represents the patient's weight.

When a particular model is able to sustain a wide range of interpretations we might be tempted to speculate that we have modelled some common law or process which underlies a variety of different real world situations. So for example we can imagine a generalized version of our model of a thermostat which could be seen as underlying any regulatory process.

Step 1. Check the value of some property of the object.

Step 2. If this value is above the required level, then do nothing.

Step 3. If this value is below the required level, then apply some process to increase the property's value.

Step 4. Start again at Step 1.