Robot Technology

Volume 6: Decision and Intelligence

Robot Technology

A Series in Eight Volumes
Series Editor and Consultant: Philippe Coiffet
English Language Series Consultant:
I. Aleksander, Imperial College of Science and Technology, London, England

Volume 6

DECISION AND INTELLIGENCE

Igor Aleksander, Henri Farreny and Malik Ghallab

Kogan Page
London

Translated by Meg Tombs

Copyright Chapters 1, 2 and 9 © Igor Aleksander
Copyright Chapters 3–8 © Hermes Publishing (France) 1986

English language translation copyright Chapters 3–8
© Kogan Page Ltd 1986
English language edition first published 1986
by Kogan Page Ltd, 120 Pentonville Road, London N1 9JN
All rights reserved.

British Library Cataloguing in Publication Data
Aleksander, Igor
Decision and intelligence.—(Robot technology; v. 6)
1. Robots, Industrial
I. Title II. Farreny, Henri III. Ghallab, Malik IV. Fonction décision et intelligence. *English* V. Series
629.8′92 TS191.8

ISBN 0 85038 651 9

Printed and bound in Great Britain
by Biddles Ltd, Guildford

Contents

Chapter 1 **Prospects for knowledge-based robots** 9

Introduction, 9
Three levels of robot 'intelligence', 10
  First generation, 10
  Second generation, 11
  Third generation, 12
The 'fifth generation' of computers in relation to robots, 14
  Technological advances, 14
  Fifth generation strategy, 15
Conclusion, 17

Chapter 2 **Robots and artificial intelligence: parallel developments** 19

Introduction, 19
Early hopes for 'intelligent' automation, 20
Motivating notions in artificial intelligence, 21
Game playing, 23
  MINIMAXING, 24
  ALPHA-BETA processing, 26
Learning, 26
Problem solving, 28
Blocks worlds, 31
The Marr legacy, 33
The Barlow principle, 34
Near-natural language communication with robots, 36
Expert systems, 39
Conclusion, 40

Chapter 3 **Expert systems and knowledge-based languages** 41

The place of knowledge-based languages in expert systems, 41
Many languages, few comparisons, 41
Categories of expert knowledge, 42
  Facts, 42
  Rules, 43
  Metarules (or rules on rules), 47
  Assertional/operational objects, 48
  Procedures, 49
  Demons, 50
  Use of variables, 50
  Representation of uncertain or imprecise knowledge, 51
Conclusion, 52

Chapter 4 **Production-rule expert systems** 55

Introduction, 55
What is an expert system? 56
  The role of expert systems, 57

Basic components of expert systems, 57
Pattern matching in expert systems, 59
Production rule systems: organization and operation, 60
Production rules, 60
Basic cycle of a production rule system, 61
Operation of a production rule microsystem, 63
General qualities of production rule systems, 65
Examples of production rule systems, 66
Expert systems, 66
General systems (shells), 69
General comment, 71
Differentiation of production rule systems, 71
Strategies for chaining production rules, 71
Invoking rules and facts by pattern matching, 73
Monotony or non-monotony of knowledge, 75
Methods for the resolution of conflicts, 76
Imprecise or uncertain knowledge; approximate reasoning, 76

Chapter 5 **Introduction to search techniques** **79**
Introduction, 79
Formulation of the problem, 79
Convergence, 81
Admissibility, 81
Complexity, 82
Heuristics, 82
Strategies for search organization, 84
Choice of strategy, 85
Further discussion, 86
Notes, 87

Chapter 6 **Heuristic graph searching** **89**
General statement of the problem, 89
Complete development algorithms, 91
Search ordered by the evaluation of cost f: the A* algorithm, 91
Depth-first searching, 99
Breadth-first searching, 101
Non-evaluated search, 102
Search ordered by the heuristic **h**, 102
Search with 'pruning', 104
Search without redevelopment, 105
Bidirectional searching, 105
Partial development algorithms, 108
Redundant search algorithms – tree searching, 111
ϵ-admissible algorithms, 114

Chapter 7 **AND/OR graphs** **121**
The search problem using an AND/OR graph, 121
Algorithms for non-evaluated search, 128
Admissible search: the AO* algorithm, 132
Convergence, 137
Admissibility, 138
Example of an AND/OR graph with circuits, 140
Complexity of the algorithm, 141
Implementation, 141
Redundant search – search in an AND/OR graph, 142
Search with pruning, 143
Search with partial development, 143
ϵ-admissible searching: the AO algorithm, 145
Convergence, ϵ-admissibility and complexity of AO_ϵ, 151
Search in a space H, for which all the solution sub-graph is a tree structure, 153

Chapter 8 **First order predicate logic** **155**

The syntax of first order predicate language, 155
  Alphabet, 155
  Terms, 156
  Atoms (or atomic formulae), 156
  Well-formed formulae (or formulae, abbreviated as wffs), 157
Inference rules, 157
Semantics of first order predicate language, 158
  Interpretations, 158
  Formula value according to an interpretation, 159
  Validity (and invalidity), inconsistency (and consistency) of a formula, 160
  Indecidability and semi-decidability of first order predicate logic, 161
  Equivalent formulae, 161
  Logical consequence formulae, 162
Relationship between the idea of theorems and of logical consequences, 162
  Sound groups of inference rules, 163
  Complete groups of inference rules, 163
  Sound or complete: rationale for application, 163
The resolution principle, 163
  Clauses, 163
  Normal prefix form of a wff, 164
  Transformation in clause form or clausal form, 165
  Relationship between a set of wffs and a set of clause forms of these wffs, 167
  The resolution principle applied to concrete clauses, 168
  Unification, 168
  The resolution principle applied to arbitrary clauses, 172
  A variant in the presentation of the resolution principle, 173
  General properties of the resolution principle, 173
Systems for refutation by resolution, 175
  General (non-determinist) procedure for refutation by resolution, 175
  Complete resolution strategies for refutation, 176
  Derivation, search and refutation graphs, 177
  Strategy for breadth-first resolution, 178
  Strategies for resolution by refutation, or set of support resolution, 179
  Strategies for resolution by refutation (lock resolution), 180
  Strategies for resolution by refutation (linear resolution), 181
  'LUSH' resolution strategies, 183

Chapter 9 **Future prospects for knowledge-based robots** **185**

Introduction, 185
Structure of the Alvey Programme, 185
The design-to-product demonstrator, 186
  The design system, 187
  The factory system, 188
Other demonstrator products, 189
Support from chips and software, 190
Knowledge system development, 191
Pattern analysis, 191
Conclusions, 192

Bibliography **193**

Index **201**

Chapter 1

Prospects for knowledge-based robots

1.1 Introduction

This volume describes the principles of the advanced programming techniques involved in decision making. Such principles are founded in mathematical logic and are an example of the way in which robotics demands a knowledge of a wide variety of disciplines. Automated decision making in the context of robotics can adopt many aspects. At the most obvious level, a robot may have to plan a sequence of actions on the basis of signals obtained from changing conditions in its environment. These signals may be quite complex, for example, the input of visual information from a television camera.

At another level, automated planning may be required to schedule the entire work cycle of a plant that includes many robots as well as other types of automated machinery. The often-quoted dark factory is an example of this, where not only some of the operations (such as welding) are done by robots, but also the transport of part-completed assemblies is automatically scheduled as a set of actions for autonomic transporters and cranes. It is common practice for this activity to be pre-programmed to the greatest detail. Automated decision-making is aimed at adding flexibility to the process so that it can absolve the system designer from having to foresee every eventuality at the design stage.

Frequent reference is made in subsequent chapters to *artificial intelligence* (AI), *knowledge-based* and *expert systems*. Although these topics are more readily associated with computer science than with robotics, it is the automated factory, in general, and the robot, in particular, that will benefit from success in these fields. In this chapter we try to sharpen up this perspective, while in Chapter 2 we aim to discuss the history of AI.

Chapter 3 deals with expert systems and knowledge-based languages which are the areas of current practical achievement in the field. Chapters 4 to 8 discuss the detailed principles associated with logical programming, with particular reference to those techniques that accelerate the computation. This is of vital importance if real-time decision-making is to be achieved. In Chapter 9 we return to discuss the industrial prospects for the field.

1.2 Three levels of robot 'intelligence'

1.2.1 FIRST GENERATION

Ever since the installation of the first industrial robot for die casting in 1961, it has been customary to compare human ability with the prowess of the robot. Most such comparisons are based on the mechanical power of the robot which, as one might expect, exceeds that of man in its capacity for handling heavy weights and its high accuracy at reasonable speeds. Such was the concern to develop these properties that less time was spent on the development of the brain, or the computer of the robot.

In his scene-setting book *Robotics in Practice*, Joseph Engelberger (1980), among 13 areas of activity, makes the man-machine comparison in the following way:

'Human operative:
uses any one or all of his five senses to follow the operation of the machine and to activate the controls as necessary. Has a memory with which he can learn the sequence and timing of the operations.

Robot operative:
a robot must be pre-programmed to carry out its operations according to a timed sequence. A man has to do the teaching, and the robot has to have an internal memory to store the information. Computer technology has made this possible.'

This is a characteristic of what are now called *first generation* robots: the presence of a computer memory is seen as the sum total of the required 'intelligence' of the machine. This, indeed, is an improvement on some of the earliest robots whose actions were controlled by a 'pinboard' of meticulously worked out connections scanned mechanically in sequence. That could be truly said to represent the 'zeroth' level of intelligence.

The ability to teach a robot a sequence of actions that could then be repeated in perpetuity with a considerable degree of accuracy was all that was necessary. Probably, 80% of the robots in use up to the late 1980s are likely to rely on this open-loop type of intelligence. But while this is adequate for work such as spot welding or pick-and-place labouring, it cannot be applied to even simple seam welding or depalletizing tasks. The reason for the deficiency is simple; the actions of the robot must be based on measured contingencies in the work place. No two seam welds are likely to be the same, and without sensory feedback the welding robot might apply its gun to fresh air, or ram it into the work piece with costly consequences.

This book is not about robot sensors; Volume 2 of this series entitled *Interaction with the Environment: Robot Sensors and Sensing*, covers this area copiously (Coiffet, 1983). This book is concerned with what the computer 'brain' of the robot can do in response to a changing environment. In programming terms, this means being weaned from a sequence of pre-programmed instructions to being capable of modifying the instruction path according to environmental occurrences. Programmers call this *branching*.

1.2.2 SECOND GENERATION

Robot designers in the mid 1970s found themselves in the same position as those dreaming about computers before the Von Neumann report of 1946; the stored program revolution was still to come (Burks *et al.*, 1946). Branching is often complex and it is the crux of general programming. So the implication is that robots should become fully programmable in the same way as a general-purpose computer, rather than being just programmable sequencers. But the robot designers of the 1970s were more fortunate than the computer pioneers of the pre 1950s, they could buy microprocessors as components out of which they could structure robot 'brains'. In other words, the evolution of the general-purpose computer into a component that barely occupies a cubic inch (fully packaged) combined with the considerable dexterity of robot manipulators led to the appearance of *second generation* robots.

The vexing question was whether these systems should be programmed in a conventional high-level language, or whether special languages directly aimed at robots needed to be developed and marketed. The needs of the robot were not only those of decisions to be made on environmental contingencies, but also those of having to control the trajectories of six or so limbs in real time.

Ideally, the robot programmer would like to write block-structured programs where the blocks were procedures such as:

```
MOVE GRIPPER      FROM X1 Y1 Z1 TO X2 Y2 Z3
                  WITH END ORIENTATION X3 Y3 Z3
```

or

```
SLOW FORWARD      IN X1 Y1 Z1 UNTIL ENDSTOP
                  THEN GRIPPING PROCEDURE P
```

Within these procedures the system programmer will wish to perform standard computing functions. This may mean that he would be well served by pre-declared data types such as speed, torque force, velocity etc. This clearly spells out the need for special-purpose robot languages, written in some efficient run-time way so that reactions in real time may be obtained.

Thus the second generation robot era was characterized by the development of such languages. They are well documented in Volume 5 of this series entitled *Logic and Programming* (Parent and Laurgeau, 1984),which describes how languages such as VAL are being adopted in Unimation robots while PLAW is being developed in Japan specifically for welding robots. The latter allows welding sensor input so that the weld head can be prevented from leaving the task.

Of the dozen or more robot control languages the following facilities (with % of coverage across the languages) were provided:

PROGRAMMING FACILITIES
Structured programming (50%)
Parallel limb drive (50%)
Move interrupt (67%)
Handshaking with robot signals (85%)
Real time clocking (25%)
Substantial program memory: mainly floppy disk (75%)

ROBOT CONTROL FACILITIES
Coordinate transformations (50%)
Trajectory control: linear or circular (100%)
Maximum effort drive (50%)
Compliance feedback (8%)

ENVIRONMENTAL FEEDBACK
Differential feedback (100%)
Manual training (90%)
Network facilities (67%)
Vision:low resolution and embryonic (75%)
Strain gauges (67%)

In summary, it is the ability of robotic programs to branch, that is to obey 'if. . .then' statements, which is characteristic of second generation robots. The main feature that distinguishes second from *third generation* robots is the complexity and interaction of the 'if. . .then' statements.

1.2.3 THIRD GENERATION

Consider the robot in Figure 1.1. It has stored in its memory the target of the task to be achieved. The 'if. . .then' ability needs to be applied to the planning of the task. For example, the correct actions for the program are to go through a reasoning process that includes:

IF I remove C THEN B and D will fall down (avoid)
IF I remove B THEN . . .

This process goes under several fashionable titles, for example *automated reasoning*, *knowledge-based processing*, *problem solving*, *automatic planning*. In computer science this activity used to fall under the heading of AI, and the clear characteristic of third generation robots is the coming together of robot programming and AI. Without wishing to pre-empt subsequent chapters of this book (particularly Chapter 2 where the parallel develop-

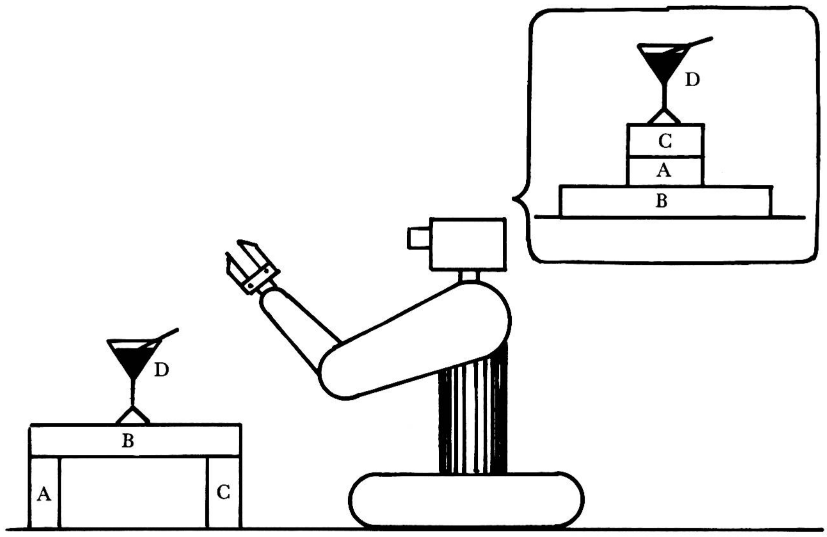

Figure 1.1 *A problem solving robot*

ment of AI and robotics is discussed), a few introductory points should be mentioned at this stage.

In theoretical terms, the step between second and third generation robots is much more significant than the increased degree of complexity of 'if. . . then' statements implied above. Interestingly, the complexity can be unravelled by the use of mathematical logic. For example, a statement such as:

'It is true that A and B are true or that A is true and B is false'

may be simplified to 'A is true'. It is this type of simplification that forms the theoretical basis of running programs as described in Chapters 7 and 8 of this book. The significance of this in terms of robot programming lies in the applicability of a totally novel style of programming: the *declarative* style.

As will be seen in subsequent chapters, new languages such as LISP and PROLOG emerged from AI research. Their objective is to allow their user to state directly or *declare* the nature of a logical problem and the rules that may be applied to its manipulation and then sit back while the machine solves the problem. In conventional programming, the programmer is under obligation to include explicitly in his problem the *method* of solution. In declarative programming, the machine searches for a solution using implicit methods that are well founded in mathematical logic.

Clearly this may not improve on-line dynamic control of robot limbs; that will always reside as an *imperative* (ie not declarative) program in a competent machine. However, such imperative programs are subordinate to the declarative ones. It is also the declarative program, particularly if PROLOG is the medium, that acts as an efficient man-machine interface. The style of such programs is based on a series of logical statements of the 'if. . .then' type. These may be written directly and executed as such by the machine. This is in contrast with the need to write structures of procedures as was indicated in the case of second generation declarative programs.

1.3 The 'fifth generation' of computers in relation to robots

There is bound to be confusion over generation numbers. Above we have written of third generation robots, whereas most readers will have heard of *fifth generation* computers. This does not mean that the development of robots is behind by a couple of generations. On the contrary, fifth generation computers are being used to provide the 'brainpower' for third generation robots. But why is the term 'fifth generation' used? And why has the development of fifth generation computing become the mainstay of political involvement in high technology in most Western nations?

1.3.1 TECHNOLOGICAL ADVANCES

The technological escalation over five stages is quite simple. The first four generations refer largely to the hardware that constitutes computing machines. The *first generation* describes the first commercially available computing machines. Their circuits used thermionic valves which not only required vast storage space but also created a heat removal problem. A first generation computer not only required a large hall, but also required another hall of equal size to house the heat extracting plant. This era had more or less come to an end by the time the first industrial robot had been installed in 1961.

In the *second generation* machines, the transistor replaced the valve and ferrite rings of one millimetre in diameter replaced pairs of valves for storing individual bits of information. The heat dissipation problem was solved, and the digital computer began to be seen not only as a tool for scientific research but as a vehicle for making businesses more effective. This was the age of the mainframe and the creation of large number-crunching systems which distributed their computing power mainly through a 'hand-it-in-then-come-and-get-it' shop. Towards the end of the 1960s, the concept of multi-entry machines and multi-user terminals began to be seen as a better way of distributing computer power. But for the roboticist, second generation computer philosophy held little joy, since should he need to control his machine, he would need an umbilical connection to an expensive mainframe that would almost certainly not react at sufficient speed.

The appearance of the minicomputer in the late 1960s heralded the *third generation* of computing machinery. The development of early silicon-integrated circuitry, particularly memory chips as opposed to magnetic devices, was responsible for the design of machines that could be afforded by small university and research departments and were not much larger than a desk and a few filing cabinets. Robotics research laboratories began to think of the possibilities of positioning the smaller versions of such machines on board experimental mobile robots. Also, as mini-computers developed further, sensible control boxes were being fitted to commercial, stationary manipulator arms confirming the introduction of second generation robots.

In synchronism with these developments, there was increased activity among language designers, particularly block-structured languages such as ALGOL and interactive languages such as BASIC. Despite the fact that BASIC is hardly ever used by computer scientists (due to its unstructured nature), it did make programming available to a large number of people. It also became an early standard in minicomputers. Indeed, in robotics, ROL (RObot Language) and VAL (Vic Arm Language) are BASIC-like languages. The first is still available for IBM PC machines, while the second was originally designed for PDP 11 minicomputers and it now runs on LSI 11 microcomputers.

Of course, it is the microcomputer that is the trademark of the *fourth generation* machines. Silicon technology in the mid 1970s was pushed to such an extent of miniaturization that it came up with a complete computer on a chip which occupies less than a cubic inch when fully packaged (where, for roughly the same computing power, a first generation machine would typically cover 3,000 cubic feet), weighs a fraction of an ounce (where the other was 30 tons) and uses about 2.5 watts (as against 140,000 watts). For the robot designer, the computer on a chip offered opportunities of deploying computing power where it was most needed, possibly both to control the individual arms of a manipulator and to coordinate the work of these distributed processors from an additional processor.

In terms of robot generations, this development in computing simply boosted the second generation. For example, the Unimation Pumas built since 1979, in addition to the use of an LSI 11 for program execution, use seven 6502 processor chips that are set off independently to ensure optimal trajectory control. These machines have also been designed to be programmed in VAL II, a nicely structured language in the style of Pascal.

1.3.2 FIFTH GENERATION STRATEGY

But this book is about *fifth generation* computing techniques for robots. Why has this generation merited so much more attention than any of the others? A good account of the historical factors associated with the fifth generation phenomenon is found in Feigenbaum (1983). Here we extract some of the features that are of relevance to a study of robotics.

Where the first four computer generations were defined by technological advances, the fifth generation is based on strategy with respect to potential for technological advance. The thrust of this effort, as is well known, came from Japan. In October 1981 the Japanese government announced its $1,000 million programme in collaboration with industry aimed at a new computer concept with some machines planned for production by 1990. The concept centres on computers that can converse with humans in natural language and be capable of seeing and hearing. They would be capable of using input data to reason and make inferences in human-like ways.

It is interesting that this decision was made at a time when Japan was well established as a leader in consumer electronics, and was beginning to make major advances in heavy industry: automotive production and shipping. It was the USA that led the world in fourth generation computers. The announcement pointed to the fact that the effort was not only intended to foster creative computer design but also to provide Japan with bargaining power. Feigenbaum describes the fifth generation as 'an exquisite piece of economic strategy'.

When the news of this announcement reached the USA, there was much debate and agitation particularly among those academics who were involved in AI research. Several calls for an effort of national priority went largely unheeded. There may well have been a very good reason for this, since the leading intellectual effort in the field was securely lodged in the major universities, with a few notable exceptions in industry. However, the Japanese event gave researchers more political power in obtaining funding for AI, and encouraged the private sector to strengthen its investment. So, due to the good base of funding that already existed, plus this strengthening, the total investment in what may best be called *knowledge systems* equalled, if not exceeded, that in Japan.

It was in Europe that more directed programmes were announced in response to the Japanese fifth generation plans. It was soon after the Japanese announcement that the Department of Trade and Industry in London rushed through the formation of a committee chaired by John Alvey of British Telecommunications. It contained representatives from government, industry and academic institutions. The target was a broad one and dubbed *advanced information technology*. This was further subdivided into four fields: *very large-scale integrated systems* (the furthering of the fabrication of advanced silicon chips), *intelligent knowledge-based systems* (the AI heart of the operation), *software engineering* (the production of advanced software) and *man-machine interfaces* (the improvement of computer usability). When announced in 1982, the programme was to be given £450 million over five years, of which the government would provide