15

Building Symbolic Representations of Intuitive Real-time Skills from Performance Data

D. Michie and R. Camacho

The Turing Institute,

Glasgow, UK

Abstract

Real-time control skills are ordinarily tacit — their possessors cannot explicitly communicate them. But given sufficient sampling of a trained expert's input–output behaviour, machine learning programs have been found capable of constructing rules which, when run as programs, deliver behaviours similar to those of the original exemplars. These 'clones' are in effect symbolic representations of subcognitive behaviours.

After validation on simple pole-balancing tasks, the principles have been successfully generalized in flight-simulator experiments, both by Sammut and others at UNSW, and by Camacho at the Turing Institute. A flight plan switches control through a sequence of logically concurrent sets of reactive behaviours. Each set can be thought of as a committee of subpilots who are respectively specialized for rudder, elevators, rollers, thrust, etc. The chairman (the flight plan) knows only the mission sequence, and how to recognize the onset of each stage.

This treatment is essentially that of the 'blackboard model', augmented by machine learning to extract subpilot behaviours (seventy-two behaviours in Camacho's auto-pilot for a simulated F-16 combat plane). A 'clean-up' effect, first noted in the pole-balancing phase of this enquiry, results in auto-pilots which fly the F-16 under tighter control than the human from whom the behavioural records were sampled.

Table 15.1. Criteria of strong and weak AI

	Strong	Weak
Feasibility of goals	Human-level intelligence will be achieved in machines within foreseeable time.	Human-level intelligence will be implemented only in some unimaginable future, or perhaps never.
Forms of implementation	All thought can be mechanized as sequential logical reasoning from axiomatic descriptions of the world. The 'physical symbol system hypothesis': all agents, including intelligent, are best implemented symbolically.	Most thought is intuitive, not introspectable, non-logical, associative, approximate and 'fuzzy': best modelled by brain-like ultra-parallel networks.
Personnel	Vintage AI professionals, e.g. Turing, Simon, Newell, McCarthy, Feigenbaum, Nilsson, and their followers.	Members of other professions, particularly in linguistics, neurobiology, physics, and philosophy.

1. INTRODUCTION

The labels 'strong AI' and 'weak AI' have sometimes been used to differentiate two schools. Criteria are summarized in Table 15.1.

The taxonomy in Table 15.1 lays emphasis on the 'physical symbol system hypothesis' of Newell and Simon (1976). Their intended interpretation restricts symbol systems to those which can transparently support communication with human users. Thus the lists of numerical weights in which neural nets express themselves constitute 'symbols' of a sort, but not in the sense intended by the above authors. This restriction has persuaded some practitioners that the physical symbol system hypothe-

sis excludes intuitive processes from AI's domain of discourse. Such separatism is unsafe, since much knowledge-based thought seems irredeemably intuitive and sub-articulate (for a recent commentary see French, 1990). For its subcognitive processes, there is no direct evidence that the brain employs a symbolic regime. Hence those who accept subarticulate expertise as a proper AI concern may wonder whether for this purpose they should abandon symbolic representations as untrue to nature. The present chapter advocates a different position, namely that a conceptually transparent symbolic style offers a way of improving on nature. By representing intuitive processes symbolically, inductive inference can do something which is both non-brainlike and also highly useful, catering to the client who says: 'My in-house experts may be 'intuitive'. But I want an expert system to formulate its reasons more explicitly than that.'

2. KNOWLEDGE AND THOUGHT

In industrial knowledge systems the implementer has to distinguish between thought as something to be communicated and thought as problem solving. Choice of representation remains a developer's option. In implementing intuition, he or she may decide that it is something over which to draw a veil. The veil may be woven of neural nets, or of hand-crafted spaghetti-code, or of something else. But suppose that the developer has to supply the customer also with means to draw the veil aside, for purposes of interrogation about goals, plans, evidence, justification, and the like. At the price of being less true to nature, he or she might then be better off not to have veiled it in the first place. Like cognitive and brain scientists, knowledge engineers also study the structure of expertise. Unlike cognitive and brain scientists, they do this (or should do) for the purpose not of emulating but of transcending the brain's limitations. First among these is the relative inarticulacy of what both cognitive scientists and knowledge engineers call 'procedural knowledge', thus distinguishing it from 'declarative'.

2.1. Declarative knowledge

It is characteristic of the retrieval and use of declarative knowledge that it is ordinarily done in conscious awareness. From a wealth of neurobiological observations concerning the effects of brain lesions on memory, L. R. Squire (1987, chapter 11) distinguishes declarative memory from procedural as 'memory that is directly accessible to conscious recollection'. By contrast, the hall-mark of a highly trained expert brain is that it does much of its work intuitively. 'Dialogue elicitation' of rules for building expert systems may therefore be frustrated whenever a given expertise involves strategies stored in procedural memory. Inaccessibility to consciousness of even parts of a targeted expertise can then cause serious problems for large knowledge engineering projects, such as Japan's ambitious 'Fifth Generation' (Michie, 1988). Differentiation of the two forms is thus desirable.

Declarative knowledge comprises whatever lends itself to logical formulation: goals, descriptions, constraints, possibilities, hypotheses. The declarative category also includes facts. When these relate directly or indirectly to events in the agent's own experience, their place of storage is referred to as 'episodic' memory. Another subdivision of declarative knowledge is held to reside in 'semantic' memory, which Squire defines as follows:

> Semantic memory refers to knowledge of the world. This system represents organised information such as facts, concepts, and vocabulary. The content of semantic memory is explicitly known and available for recall. Unlike episodic memory, however, semantic memory has no necessary temporal landmarks. It does not refer to particular events in a person's past. A simple illustration of this difference is that one may recall the difference between episodic and semantic memory, or one may recall the encounter when the difference was first explained.

A school founded by John McCarthy (1959) aims to extend formal logic to serve as a vehicle for mechanizing declarative knowledge (see a recent collection edited by Ginsberg, 1987). We will say little further about the project, beyond expressing respect for such work. Its philosophical importance is matched

only by its difficulty. Our theme is closer to the name and nature of expert systems. These are not so much to do with giving computers knowledge of the world, as with equipping them with useful know-how. In face of the difficulties which confront the McCarthy project, there is something to be said for separately studying the mechanization of procedural knowledge and only later integrating the two levels.

2.2. Nature of procedural knowledge

In Anderson's (1990) text on cognition, skilled procedures are pictured as arising in part by derivation from pre-existing mental descriptions. No direct evidence is offered. Knowledge engineers concerned with real-time skills have been led by practical experience in a rather different direction. The empirical picture is one of inductive compilation from sensorimotor data gathered in the course of trial and error. In this picture the role of higher-level knowledge is not to participate directly, but to steer the learning process, setting and adjusting the frame within which skill-bearing rules are constructed.

The final phase of skill-learning, described by Anderson and others as 'automatization', does not ordinarily support introspective report by the expert performer, hence the 'knowledge-acquisition bottleneck' of applied AI. Procedural knowledge, as we have seen, limits itself to the 'how to' of skilled tasks, whether physical as in making a chair, or more abstract as in prediction of sterling rates against the dollar or the diagnosis of acute abdominal pain. A common synonym for such knowledge is know-how, and its manifestation in observable behaviour is called 'skill'. One difficulty is that observed task-performance does not necessarily reveal whether a given expert's behaviour really exemplifies a skill in the procedural sense or whether he or she is using declarative-semantic memory to form action-plans on the fly. Squire's earlier-cited definition supplies a test, namely the ability to give a verbal account of the way in which each decision was made, possible only for declarative memory. A second criterion is the frequency of the recognize–act cycle: this may simply be too fast for 'what-if' inferential planning to be feasible.

For those concerned to recover procedural rules, as in build-

ing expert systems, lack of verbal access (on which Anderson also remarks) is a problem. Yet there is widespread faith among knowledge engineers that special methods of 'dialogue elicitation' can be found which will permit the construction of rule-based systems on the scale of such inductively built systems as the GASOIL (Slocombe *et al.*, 1986) and BMT programs of Table 15.2.

Is rule induction from expert-supplied data nevertheless in some sense a second-best option for building systems on the BMT scale? On the contrary. Experts can rapidly and effectively communicate their skills (as in the BMT case) solely via illustrative responses to selected cases. Does he or she thereby omit something indispensable? Certainly the practitioner's explicit and communicable awareness is basic to expertise in some task domains. But other domains, which lack this property, can be found not only among a rather wide variety of industrial tasks, but even in such purely 'mental' forms of expertise as playing a strong game of checkers (see below).

As a paradigm of procedural knowledge, Feigenbaum and McCorduck (1983, p.55) give the example of tying one's shoes. It is interesting that once this skill has reached the stage known as automatization it can continue unaffected by destruction of the individual's brain mechanisms for acquiring and handling important forms of declarative knowledge. Damasio describes a patient named Boswell. The following summary is from Patricia Smith Churchland (personal communication).

> In addition to losing the hippocampal structures, he has massive damage to frontal cortex. He can identify a house, or a car, but he cannot identify his house or his car; he cannot remember that he was married, that he has children, and so forth. He seems to have no retrograde episodic memory, as well as no anterograde episodic, ... Boswell can still play a fine game of checkers, though when asked he says it is bingo. He cannot learn new faces and does not remember 'pre-morbid' faces such as that of his wife and his children... Boswell can play checkers, tie his shoes, carry on a conversation, etc.

Of considerable interest is the survival of Boswell's checkers skills. Evidently what we shall later term 'fast' skills are

Table 15.2. Of the world's three largest expert systems the two latest (GASOIL and BMT) were not constructed from rules obtained in dialogue fashion, but by automated induction from expert-supplied data. In each case the induction engineer trained the system in the desired skill in the style that the master of a craft trains an apprentice, by a structured sequence of selected examples. Rates of code production are typically in excess of 100 lines of installed Fortran, C, Pascal, etc., per programmer day. The methodology allows validation to be placed on a user-transparent basis (Michie 1989), and maintenance costs are in many cases trivialized. Tabulation is from Slocombe *et al.* (1986) with 1990 data on BMT added. The BMT program is described on p.10 of Pragmatica, vol. 1 (ed. J.E. Hayes Michie), Glasgow, UK: Turing Institute Press.

	APPLICATION	NO. OF RULES	DEVELOP. MAN-YRS	MAINTENANCE MAN-YRS/YR	INDUCTIVE TOOLS
MYCIN	medical diagnosis	400	100	N/A	N/A
XCON	VAX computer configuration	8,000	180	30	N/A
GASOIL	hydrocarbon separation system configuration	2,800	1	0.1	ExpertEase and Extran 7
BMT	configuration of fire-protection equipment in buildings	>30,000	9	2.0	1st Class and RuleMaster

not the only ones for which procedural knowledge may dominate over declarative. In contrast to chess skill, checkers was already known not to lend itself to the planning approach and to be essentially 'intuitive'. When A.L. Samuel was engaged in his classic studies of machine learning using the game of checkers, he had numerous sessions with leading checkers masters directed towards dialogue acquisition of their rules and principles. Samuel reported (personal communication) that he had never had such frustrating experiences in his life. In terms of relationship to what the masters actually did, the verbal material which he elicited contained almost nothing which he could use or interpret. In similar vein, Feigenbaum and McCorduck (loc. cit., p.82) describe this type of expert response in the following terms: 'That's true, but if you see enough patient/rocks/chip-designs/instrument readings, you see that it is not true after all.' They conclude 'At this point, knowledge threatens to become ten thousand special cases.'

The message from clinical studies is that skilled performance of even sophisticated tasks can still be manifested, and learned, when the brain is so damaged that knowledge of new happenings cannot be retained and previously stored facts and relations (declarative-semantic memory) are seriously disrupted. Another circumstance under which the mediation of declarative memory is at least equally disabled can be observed in the normal brain by imposing a sufficiently restrictive constraint on the time available for the recognize–act cycle, as in touch-typing. This skill does not depend on the storage and retrieval of declarative knowledge, and can be acquired and executed in its virtually complete absence. Recall that when copy-typing at speed the typist does not need to understand the words as he or she reads them. Indeed, after a speed test little or nothing of the text's content can be recalled. Moreover, educated onlookers are surprised, although they should not be, by the outcome of a request to the typist (supposing that he or she has been using a typewriter with unlabelled keys) to label the keyboard correctly with the proper alphanumeric symbols. Lacking a declarative model, the touch-typist is ordinarily unable to do so (see, for example, Posner, 1973), other than by deliberately typing a symbol and

observing where the finger went!

Simon (in press) has recently re-emphasized that simple recognition of a familiar object takes at least 500 milliseconds. Operations involving reference to a semantic model of the task domain require retrieval from long-term memory of relatively complex knowledge-structures and an associated apparatus for inferring, storing, and utilizing intermediate results. Such elaborate transactions are to be found only in the 'slow lane'. Here seconds, minutes, or even hours are required to incubate a decision. The bare bones of an explicit rationale for a slow-lane decision, when it comes, can usually be elicited from the expert by verbal report. Not so in the fast lane, to which the present discussion is confined. 'Fast' skills cannot be accessed by 'dialogue elicitation' methods. How then are expert systems to be built for these skills? A solution is to record behavioural traces from the expert subject. Inductive inference then reconstructs from recorded decision-data rule-based models of the brain's hidden strategies. As reported in this review, machine execution of data-derived models has been found to generate performance exceeding in reliability the trained subject's own.

2.3. Postulates of skill acquisition

Experimental work which will now be described was animated by a point of view about brains, summarized below as a list of postulates. Declarative knowledge is abbreviated to 'D' and procedural to 'P'. P designates only procedural knowledge which has already reached the automatized stage.

1. human agents are able verbally to report their own D;
2. human agents cannot verbally report their P;
3. D can be augmented by being told, and also by deduction;
4. P is built by learning, whether by imitation or by trial and error;
5. P can be executed independent of D, but not vice versa;
6. decision-taking via P is fast relative to use of D;
7. sufficiently fast control skills depend on P alone;

8. even for some slow skills P is sufficient for expert performance; rule-induction can extract an explicit form of P from behavioural traces.

Experiments on dynamical control have yielded illustrations of the listed postulates, culminating in a test of 9 above, namely induction of rules from silent brains. But a comment is first requisite on the undoubted existence of expert systems (EXCON was mentioned earlier) whose rule-bases have, with whatever difficulty, been constructed by dialogue acquisition.

Many observers have noted that experts seek to escape from the requirement of rule-formulation (which they find uncongenial) by supplying 'rules' of such low-level form that they constitute no more than concocted sample cases, i.e. specimen decision-data. The phenomenon has been described by Sterling and Shapiro (1986) in their description of the construction of a credit evaluation expert system. The finance specialists continually gravitated towards concrete instances rather than general rules. This has indeed been a universal finding in knowledge engineering, in line with the known facts concerning procedural memory and its mode of access.

But what if knowledge engineers in search of improvements on raw formulations were consciously or unconsciously to apply their own powers of inductive inference to such sample cases? They could then themselves create the kind of high-level rule structures that they had hoped to elicit. The result would of course be testimony more to their own powers of inductive generalization than evidence that experts can introspect their own rules. In a recent aerospace application two knowledge engineers were able, by deliberately exploiting this style of 'rule-conjecture and test', to construct a rule-based solution with no more than a black-box simulator of the task domain to provide corrective feed-back. No set of rules pre-existed, either in an expert's brain or anywhere else.

3. AN EXPERIMENT IN RULE-BASED CONTROL

The role of the systems developer postulated above requires only a reactive oracle. This source need not be an expert. Indeed, it

need not be human. As will be described it could be a simulator on which the developers can play 'what-if' games with their latest conjectured rules (what if we modify the rules like this? ... what would result from that adjustment? ... etc.). In an R & D contract for a US space consortium Sammut and Michie (1991) were given access to just such an interactive oracle.

When building a controller for a physical process, traditional control theory requires a mathematical model to predict the behaviour of the process. Many processes are either too complicated to model accurately or insufficient information is available about the process environment. Space-craft attitude control is an example of the latter. The client was interested in the development by machine learning of a rule-structured controller. A check was desirable as to whether dynamical control tasks can be satisfactorily handled by production rules at all, whether these are captured by learning algorithms or developed in some other way.

If the attitude of a satellite in low Earth orbit is to be kept stable by means of thrusters, the control system must interact with many unknowns. For example, although very thin, the Earth's atmosphere can extend many hundreds of kilometers into space. At different times, the solar wind can cause the atmosphere's density to change, thus altering the drag and aerodynamic torques on the vehicle. These are factors which earth-bound designers cannot predict and even after three decades of space flight, attitude control is still a major problem.

The client required a trial of rule-based control, using a computer simulation of an orbiting space-craft under 'black box' conditions. By this is meant that knowledge of the simulation's structure and parameters was unavailable to the developers and hence to the controller. Constraints and assumptions included minimal human supervision. Only one ground station was to be used for control. The ground crew therefore have only a 16-minute window in each 90-minute orbit during which they can communicate with the space-craft. A premium was thus placed on the controller's aptness for generating intelligible reports.

The client's 'black box' simulated three-axis rigid body attitude control with three non-linear coupled second order differen-

tial equations, and was supplied as Fortran object code. The use of pseudo-random generators introduced various time-varying disturbances, not only concerned with aerodynamic effects of solar wind variations and of atmospheric density and altitude changes, but also effects of propellant expenditure, payload redistribution, solar array articulation, extension and retraction of the gravity gradient boom and the motion of robotic and other on-board manufacturing appliances. Due to such unpredictabilities and to the possibility of a failure while out of communication with the ground, interest in a rule-based back-up controller centred on robustness, simplicity, and conceptual transparency.

The BOXES adaptive rule-based control algorithm (Michie and Chambers, 1968; Chambers and Michie, 1969) was recently the subject of new work by Sammut (1988) who also reviewed trials of other algorithms for learning rule-based solutions to the 'pole and cart' problem. A rigid pole is hinged to a cart which is free to move along a track of fixed length. The learning system attempts to keep the pole balanced, and the cart within the limits of the track, by applying to the cart a force of constant magnitude but variable sign, either right or left ('bang-bang' control). The pole and cart system is characterized by four state variables which make up a four-dimensional space. By dividing each dimension into intervals, the state space is filled by four-dimensional 'boxes'. With each box (i.e. local region of state-space, or 'situation' in the terminology of situation-action rules) is associated a setting which indicates that for any point within the given box the cart should be pushed either to the left or to the right. Essentially this representation was tested on the client's simulated spacecraft.

3.1. The black box

The task was to drive the system from its initial state to the specified final state and maintain that state. Included in the black box was a fourth order Runge–Kutta numerical algorithm which integrated the dynamics of the equations of motion. The time step had a fixed value of 10 seconds. The black box kept track of time and randomly injected various time-dependent disturbances as earlier described.

The state variables:

Attitudes: yaw (x), roll (y), pitch (z)

Body rates: ω_x, ω_y, ω_z.

Initial values of the state variables:

$x = y = z = 10$ deg

$\omega_x = \omega_y = \omega_z = 0.025$ deg/sec

The desired state:

$x = y = z = 0 \pm 3$ deg

$\omega_x = \omega_y = \omega_z = 0.005$ deg/sec

Failure conditions:

x or y or z exceeds $\pm$ 30 deg

ω_x or ω_y or ω_z or exceeds $\pm$ 0.05 deg/sec

A flag is turned on if any of these go out of bounds.

Available control inputs:

Torque: T_x, T_y, T_z.

Torque was applied by the firing of thrusters which were aligned to the body axes. Although other attitude control devices (momentum exchange systems) will be used on the satellite in addition to thrusters, this work only addressed the use of thrusters. The following are minimum and maximum torques which can be applied by the thrusters:

$T_x(\text{Min}) = T_y(\text{Min}) = T_z(\text{Min}) = 0$ ft-lbf

$T_x(\text{Max}) = \pm$ 0.5 ft/lbf; $T_y(\text{Max}) = T_z(\text{Max}) = \pm$ 1.5 ft/lbf.

3.2. The rules

The first trial was made by directly adapting a set of BOXES-derived rules from the pole-and-cart domain to a sequential logic suggested by hand-derived rules due to Makarovic (1987, 1991). In each recognize–act cycle rule-matching follows a certain priority order, cycling through the state variables until an action is selected. For each in turn the rule first checks that the first derivative does not exceed certain bounds. If it does, then a

force is applied to oppose it. If it does not, then with respect to the same variable check its magnitude. If it exceeds given bounds then a force is applied accordingly.

In the case of the pole and cart, there was a clear priority to the order in which dimensions were checked. It was critical that the angular velocity and the angle of the pole were considered before the cart variables, since neglect of the pole leads to failure much more rapidly than neglecting to keep the cart away from the ends of the track. If this principle is applicable to the case of the space-craft then it is necessary to determine which of the state variables changes most rapidly. This was done, yielding rules expressible in 'if-then-else' form, thus:

if $\omega_z <$ -0.002 then	apply a T_z of +1.5
else if $\omega_z >$ 0.002 then	apply a T_z of -1.5
else if z < -2 then	apply a T_z of +1.5
else if z > 2 then	apply a T_z of -1.5
else if $\omega_y <$ -0.002 then	apply a T_y of +1.5
. . . and so on ...	

Note the use of 'bang-bang' control, i.e. the torquers were set either fully positive or fully negative just as in the pole-balancing experiments. With a space vehicle there are three dimensions, not one, to which a control motor (torquer) can apply a positive or negative thrust, corresponding to the yaw, roll, and pitch dimensions of rotation respectively. The thresholds for the variables were determined by choosing an arbitrary value slightly within the bounds given for the desired values of the variables.

This control strategy proved to be successful but slow, requiring 8700 seconds to bring the vehicle within desired bounds, and it also consumed 11.2 units of propellant. The question arose whether the control of each dimension could be decoupled. The cited rule only allows one thruster to be fired at any one time. If each axis of the craft were considered separately then all three thrusters could be fired simultaneously. This modification resulted in rules which brought the vehicle under control very quickly, requiring only 4090 seconds. But propellant consumption, although improved, was still too high, using 7.68 units

Table 15.3. A decision array for control of the yaw dimension.

Yaw too-positive	$T_x/4$	0	$-T_x/4$	$-T_x/2$	$-T_x$
Yaw OK	$T_x/2$	0	0	0	$-T_x/2$
Yaw too-negative	T_x	$T_x/2$	$T_x/4$	0	$-T_x/4$
	Yaw-rate too-neg.	Yaw-rate negative	Yaw-rate OK	Yaw-rate positive	Yaw-rate too-pos.

before the vehicle became stable. Therefore a partial retreat was made from pure 'bang-bang', with a view to replacing it with finer control of the thrusters.

The resulting strategy is best understood by a decision array. For example, yaw control can be displayed as in Table 15.3 and the resulting performance as in Figure 15.1. Each of the 15 boxes corresponds to one control rule. Thus the box in the top left hand corner states that if the yaw is positive (i.e. above the bounds on the desirable yaw) and the yaw rate ω_x is well below the bounds of desirability then apply a quarter of the full torque in the positive direction. Thresholds were set for angles at ± 2 deg and for angular velocities they were ± 0.002 and ± 0.003. The decision arrays for roll and pitch dimensions were of the same form. The resulting control behaviour was highly satisfactory. The pitch dimension was the slowest of the three to be brought within the desirability zone.

The client's engineers stated that both in speed of recovery and in propellant expenditure results were close to calculated optima. Since however it appeared that the satellite had greater inertia in the z-axis (pitch) than in the other two the thrust of the z-torquer was increased. This brought the vehicle under control in 5290 seconds, somewhat more slowly than the previous

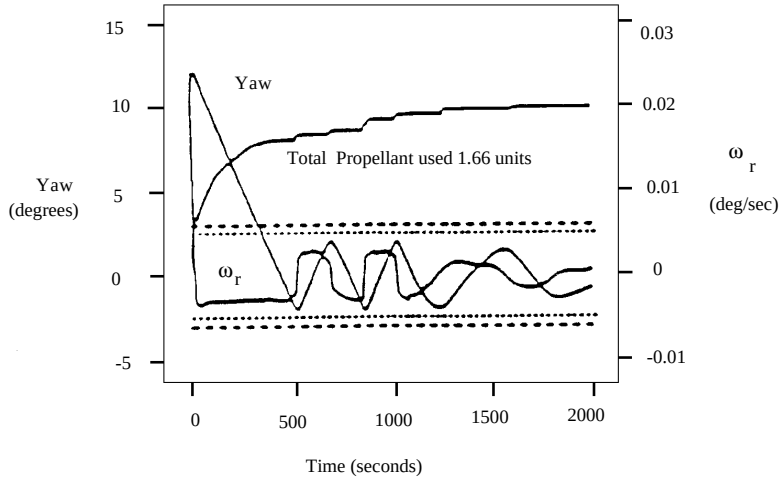

Figure 15.1. Plot over time of vehicle's yaw behaviour (see text)

controller. But it only required 1.7 units of propellant, a substantial saving. Also calculations and simulations by the client's engineers made the result appear slightly better than optimal. This doubtless arose from minor approximations and/or distributional assumptions made in their numerical work. Time did not permit the point to be elucidated. But the broad conclusion was seen as extremely encouraging. An industrial-strength problem had shown that the simplicity, robustness, and conceptual transparency of rule-based control does not have to be purchased at the cost of significant degradation of performance.

4. EXPERIMENTS WITH SKILL-GRAFTING

Supported by the freedom interactively to test each conjectured modification on the simulator, Sammut and Michie found their own powers of inductive conjecture adequate. But tasks of higher complexity, such as remote control of pilotless aircraft, would demand a less primitive approach. Present ideas are oriented towards the industry's use of interactive simulators for training pilots. A simulator-trained performer cannot tell you his or her strategy, but can demonstrate it. What is demonstrated can be automatically recorded. What is recorded can be inductively analysed by computer. With psychology-trained col-

leagues, Michael Bain, Jean Hayes-Michie, and Chris Robertson, one of us (D.M.) engaged in an investigation into the use of the rule-induction algorithm C4.5 (see Quinlan, 1987) to uncover effective control rules from such behavioural records. Experimental subjects were trained on an interactive simulation of a task illustrated in Figure 15.2. Control was exercised through a joystick of a pole-and-cart simulation which refreshed the screen approximately 20 times per second. New results together with earlier findings with this experimental system (Chambers and Michie 1969) lead to conclusions as follows (details are available in Michie, Bain, and Hayes-Michie 1990).

4.1. Conclusions from pole-balancing

First conclusion: role of problem representation. Chambers and Michie used two regimes of training, identical except for the graphical animation seen by the subject. In one variant the picture was as shown. In the other the subject saw only a display of four separate horizontal lines, along each of which a pointer wandered to and fro. The subjects in this second variant were kept in ignorance of the nature of the simulated physical system. Unknown to them, the pointers actually represented the current status of four state variables, namely position of cart, velocity of cart, angle of pole, and angular velocity of pole. Our hypothesis was that when the system is run fast, leaving only time for use and up-dating of procedural memory, then there will be no difference in the learning curves of subjects using the two different representations. Although not explicitly reported in their paper, an indication of this was observed by Chambers and Michie. In recent work a rate was additionally used sufficiently slow for subjects to report the task as having a major 'planning' component. This slow-trained group learned more slowly, at least in the initial stages. In the new work trials have not yet been made of the lines-and-pointers representation.

Second conclusion: induction of rules from behaviour. Machine learning by imitation of a trained human was first shown for the inverted pendulum by Donaldson (1960) and partially reproduced under bang-bang conditions by Widrow and Smith (1964). Our concern was to test the ability of modern induction

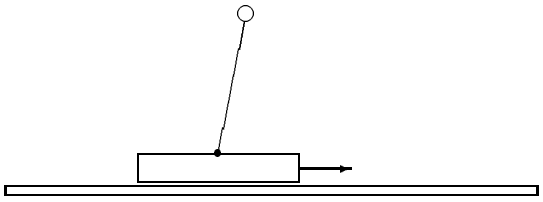

Figure 15.2. Diagram of the pole and cart task

algorithms to extract from the behavioural record the kinds of rules believed to accumulate in procedural memory during skill-learning. Results have been positive. A task was investigated where the object was to cross the centre of the track as often as possible in an allotted time-span without dropping the pole or crashing the cart. When induction-extracted rules were installed in the computer as an 'auto-pilot', performance on the task was similar to that of the trained human who had generated the original behavioural trace, but more dependable, as described below.

Third conclusion: the clean-up effect. Rules induced from a behavioural record can be assessed in two different ways. Predictive mode tests the ability of a rule-set correctly to predict other behaviour sampled from the same source. Performance mode tests the ability of the rules to substitute for the human source in executing the skilled task.

Induced rule-sets performed satisfactorily in the second mode while consistently showing high prediction error, often exceeding 20 per cent. One of the team, Mr. Michael Bain, pointed out that when watching a machine-generated rule-set's performance on the screen one is struck by an appearance of super-human precision and stability. A trained human skill, although controlled by an equally precise and stable set of production rules, is obliged to execute via an error-prone sensorimotor system. Inconsistency and moments of inattention would then be stripped away by the averaging effect implicit in inductive generalization, thus restoring to the experimenters a cleaned-up

Table 15.4. Clean-up effect shown by induced control rules over a 5-minute test period. x = position, θ = angle: 'dot' denotes first derivatives. These results are typical, and have been many times confirmed in test runs with the same, and with other, subjects.

	x	$\dot{x}$	θ	$\dot{\theta}$
Trained human (ranges)	2.79	4.85	0.562	5.021
Induced rule (ranges)	0.46	1.83	0.134	2.276
Range differences	2.33	3.02	0.428	2.745
'Clean-up'	83%	62%	76%	55%

version of the original production rules. When tested in predictive mode, such a rule-set can do no better than the cumulative sum of human perceptual and execution errors allow. But in performance mode one would expect a super-reliable stereotype of the behaviour of the human exemplar. Direct confirmation of this idea was obtained by calculating the magnitude of the pole and cart's excursions during a control session along each of the four dimensions of the state space. Observed ranges tabulated in Table 15.4 were obtained from a behavioural trace recorded from Mr. Bain's own trained performance.

The findings suggest that 'skill-grafting' from behavioural traces may be possible for more demanding tasks, such as those encountered in aircraft flight control. The key idea is that if we could look inside the head of the ground-based pilot of a remotely controlled aircraft, or of the on-board pilot of a difficult vehicle such as a helicopter, we might see a neural encoding of a fully sufficient skill, but degraded in real-time execution by sensorimotor delays and errors. Recovery of a logically equiva-

lent rule structure and its transplantation to an error-free device (i.e. to a control computer) then offers a source of enhanced and more reliable performance. In advanced rotorcraft control there is a current need for libraries of individual autopilot manouevres ('circle at 50 feet', 'fly slowly sideways for one minute', etc.) which the pilot could activate in difficult weather or other conditions, so as to free his attention for some main task in hand, visual search of water surface, target acquisition, etc.

4.2. Learning to fly

Sammut and colleagues have recently been able to reproduce the 'skill-grafting' phenomenon in the complex task of flying a simulated aircraft (Sammut, Hurst, Kedzier, and Michie, 1992). Using a flight simulator developed by Silicon Graphics, three subjects trained themselves by repeatedly piloting a simulated Cessna through the successive stages of a defined flight plan, consisting of the following manouevres:

1. Take off and fly to an altitude of 2000 feet.
2. Level out and fly to a distance of 32 000 feet from the starting point.
3. Turn right to a compass heading of approximately 330°.
4. At a North/South distance of 42 000 feet, turn left to head back towards the runway.
5. Line up on the runway.
6. Descend to the runway, keeping in line.
7. Land on the runway.

Taking 'events' as being signalled by the occurrence of control actions, then up to 1000 events were recorded per flight. Each of three trained subjects performed 30 flights, so that the complete data comprised about 90 000 events. For each event the control action was recorded, together with values of state variables measured at a moment selected 1–3 seconds earlier. The 'offset' makes approximate allowance for the pilot's delay in responding to complex stimuli. To give a rough impression of the data, the following are names of recorded variables:

boolean variables: *on-ground, g-limit, wing-stall*;

*twist, elevation, azimuth, roll-speed,*integer variables:
climb-speed, fuel, thrust, flaps;
real variables: *E/W distance, altitude, N/S distance,*
rollers, elevator.

The simulation program was modified to log the subjects' actions during flight. Log files from trained subjects were used to create the input to an inductive rule-learning program. The learning program was Quinlan's (1987) C4.5. Its output took the form of separate decision trees for each of the four different control actions, further sub-divided into the seven stages listed above. For example, to quote from the original paper,

> The critical rule at take-off is the elevator rule:
>
> elevation $>$ 4: level-pitch
> elevation $\leq$ 4
> airspeed $\leq$ 0: level-pitch
> airspeed $>$ 0: pitch-up-5
>
> This states that as thrust is applied and the elevation is level, pull back on the stick until the elevation increases to 4°. Because of the delay, the final elevation usually reaches 11° which is close to the values usually obtained by the pilot. 'pitch-up-5' indicates a large elevator action, whereas 'pitch-up-1' would indicate a gentle elevator action. The other significant control at this stage is flaps:
>
> elevation $\leq$ 6: full-flaps
> elevation $>$ 6: no-flaps

Once the aircraft has reached an elevation angle of 6°, the flaps are raised.

The 28 decision trees were automatically converted to C-code routines, arranged as a suite of seven flight control modules, each responsible for all aspects of a given stage. A new module was

invoked as soon as a pre-programmed precondition was satisfied for the onset of the next stage. Within each module, four sets of if–then rules separately supervized the four separate control actions.

An autopilot was generated in this fashion from each of the trained subjects. Tests were made by running the simulator in autopilot mode, substituting as autopilot code one or another of the three inductively synthesized program suites. The entire flight plan was executed with conspicuous competence, but with individual mannerisms characteristic of the flying styles of the individual human data source. Indications of the 'clean-up effect' (see earlier) were also evident, particularly during the approach stage.

4.3. Learning to fly straight

What is the significance of the foregoing experiment? Primarily that a suitable decomposition of the problem allows the skill-grafting methodology to be scaled up. Inductive skill-grafting evidently is not just applicable to pole-balancing but also to more complex domains such as flight control.

The same workers also reported indications of the 'clean-up' effect earlier found in the pole-balancing experiments, but these indications were of a preliminary nature only. We now report a more detailed examination of this phenomenon independently conducted by Camacho (1992) using a more challenging flight control task. He used a computer simulation (ACM public-domain software down-loaded onto a Sun Sparcstation 2) of the F-16 combat aircraft. Using Quinlan's C4.5 (see Quinlan, 1987) decision-tree induction package Camacho not only found that clean-up was operating, but was also able to show that in his experimental context it played a very large, almost dominating, role.

Camacho followed a similar methodology to that of Sammut *et al.*, details being as follows.

Flight plan stages:

1. Take off.

2. Climb to 1500 feet.
3. Reduce climbing angle and thrust attaining level flight at 2 kilo-feet.
4. Fly parallel to the runway's long axis for a distance of 200 kilo-feet.
5. Turn left 270°.
6. Turn right to line up with the runway.
7. As soon as distance to runway is less than 70 kilo-feet, start descent to runway keeping in line.
8. Land on the runway.

Variables sampled.

real: magnitude of airspeed (knots) (Geoparallel system)
real: y coordinate of airspeed (knots) (Geoparallel system)
integer: x position (ft) (Geoparallel system)
integer: y position (ft) (Geoparallel system)
integer: altitude (ft)
real: climb rate (ft/h)
real: g-force vector in acft system (only z coordinate)
real: roll rate (rad/sec)
real: pitch rate (rad/sec)
real: yaw rate (rad/sec)
real: heading (rad) /* Euler angles for acft */
real: pitch (rad) /* Euler angles for acft */
real: roll (rad) /* Euler angles for acft */
real: angle of attack (rad)
real: angle of sideslip (rad)
real: elevators setting (radl)
real: ailerons setting (rad)
real: rudder setting (rad)
real: elevator trim setting (NOT used)
real: flaps setting (rad)
real: speedBrake setting (rad)
integer: throttle
boolean: gear handle
boolean: brakes

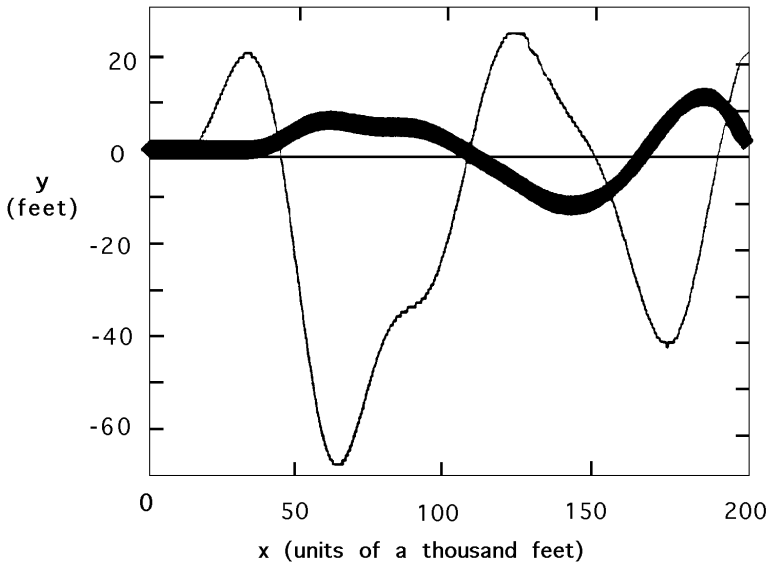

Figure 15.3. The 'clean-up' effect. Plotted lines show distances travelled in the horizontal plane from take-off by human pilot (light lines) and the autopilot (heavy line) using the ACM flight simulator of the F-16 combat plane (see text): the y axis represents deviations in the horizontal plane from straight flight.

boolean: afterBurner

Control commands used were: elevators, rollers, rudder, flaps, speed brake angle, throttle, gear handle (boolean), brakes (boolean), after burner (boolean). Thus for each of the flight plan's eight stages nine separate decision trees were synthesized.

From each of typically twenty missions, successive 'state-vectors' were sampled and written to file, making about 213 000 'state-records' in all. As a post-processing operation, between 1100 and 1600 'events' were then machine-selected from each of these, making about 25 000 'events-records'. As in Sammut *et al.*, only those state vectors were selected which precede by a fixed interval in the file the subsequent record of a control action. The set of events so constructed formed the 'training set' for inductive synthesis of a complete autopilot of the form: flight plan plus 72 decision trees.

The earlier-mentioned clean-up effect became evident when

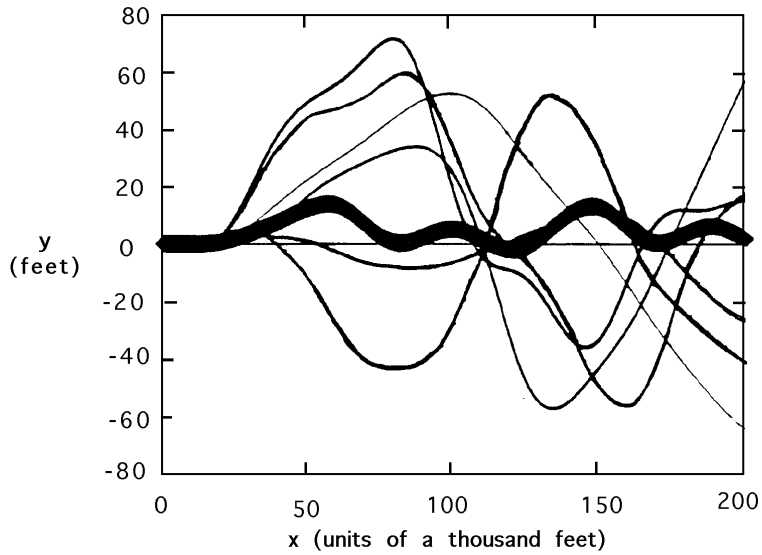

Figure 15.4. Further measurements of the 'clean-up' effect, see previous figure. The six thin-line plots represent the first six missions of a total of 20 flown by the human pilot (R. Camacho) to form the training set of about 25 000 events. The general appearance of the remaining 14 was very similar when plotted in the same way (see text): x represents distance travelled in the horizontal plane from take-off; the y axis represents deviations in the horizontal plane from straight flight.

autopilots synthesized according to the above formulation were substituted for human control. The magnitude of this gain in steadiness of control can be appreciated by a study of Figures 15.3 and 15.4, which relate only to one dimension, namely control of horizontal deviation from flight plan during the first four stages (straight-line flight on a constant bearing).

This has so far been the main result of an investigation still in its early stages. It should be emphasized that on the other criteria there are local stages of the total mission where improvement is needed. In particular, probably because the human pilot himself (R. C.) has not yet adequately mastered stage eight, the autopilot induced from these records has not either. Self-training, as well as autotraining, is currently continuing. Since the foregoing was written, both human and clone have become able routinely to land the simulated F–16 without mishap.

4.4. A Blackboard-like Model for Coordination Among Agents

In both of the flight control implementations reviewed above, there is a two-level hierarchy of control: a high level 'chairman' (the flight plan) and a set of low-level 'agents' (decision trees). So far the only role played by the chairman is to monitor the stage of flight and switch the subset of active agents according to context (stage of flight). Each low-level agent has a very specialized task of deciding upon one control in one given stage. All active low-level agents have the same view of the situation (inspect all the state variables) including access to the decision values of their peer agents. Despite the two-level design of the current controller there is no supervized coordination of the low-level agents. How then is the work done?

There is a strong similarity between the community of low-level agents and the AI paradigm of the blackboard (see Nii, 1986, for review). Each agent behaves like a blackboard's 'knowledge source' responsible for a specialized problem solving activity (decide one of the controls in one particular stage of flight). The blackboard (shared memory) role is played by the variables of the aircraft depicting the overall situation. Since all variables are visible to every agent the agents have the same view of the situation and, most importantly, have information about

their peer agents by watching their corresponding decision values in the 'blackboard'. This latter facility is responsible for the coordination among low-level agents. As an example, if the rudder agent decides to change its value (moving the physical rudder) the change will be noted on the 'blackboard' and the roller agent, seeing it, may compensate the banking effect of the physical movement of the rudder. For achieving this coordination effect the decision values of the other agents must also be used as attributes during the learning phase. The chairman represents explicit knowledge that is easily articulated and therefore can be hand-crafted. The specialist agents, on the other hand, implement low-level real-time control skills that, in a human, are not performed at a conscious level and therefore cannot be articulated. To create this kind of knowledge, each agent is separately derived by inductive learning from recorded human performance of the skilled task — a step which is an extension of previous blackboard models. The current implementation of the low-level agents may be effective if some small variations to 'normal conditions' appear (mild wind). But if the wind is abnormally strong (serious exception to normal flying conditions) then an understanding of the situation is needed and possibly a reformulation of some current goal, like make a slight change in the bearing to accommodate the wind component in the final velocity. Therefore to improve the skills of the chairman and to incorporate planning capabilities, a deep model (possibly qualitative, as suggested by Sammut (1992)) will be needed and the capability of reasoning from first principles using it. In a way similar to the human counterpart the computer high-level agent should be silent most of the time, just monitoring the overall situation and making small corrections from time to time. It should be fully activated only when the situation requires considerable replanning and deep reasoning for dealing with exceptions for which the low-level agents have no decision.

There is obviously a strong case for implementation of the chairman in a logic programming language. Sammut (1992) has suggested that the low-level agents should *also* be coded in a first order language and constitute a library of primitive actions that

the high-level planner could use, setting values for parameters and defining goals.

4.5. Conclusions from autopilot induction

Models extracted as above from decision-data by rule learning are purely heuristic in form. They incorporate no explicit references to time or causality. Yet as reviewed earlier, real-time human problem-solving involves co-operation between two separate kinds of mental process driven from two separate memory systems, updated by separate kinds of learning. The dichotomy is recognized in AI under the labels 'heuristic' and 'causal'. These roughly correspond to the neuropsychologist's procedural/declarative distinction. The balance in practice is set by the time-constraints imposed by different tasks. A fast situation–action cycle allows time only for executing heuristics and virtually none for reasoning about causes.

As the autopilot experiments demonstrate, complex skills can be built entirely from heuristics. Bears can learn to ride bicycles, and humans can fly combat planes through mission phases which allow no time for analysis. Under such circumstances, everything goes by pattern-invocation. The formal identity between pattern classification and control then stands out clearly. This identity has recently been discussed (Michie, 1991) in connection with a definition of learning which says:

> a learning system uses sample data (the training set) to generate an up-dated basis for improved classification of subsequent data from the same source.

The above-cited discussion continues:

> Notice that the definition, although phrased strictly in terms of classification, logically extends to acquisition of improved performance on tasks which do not look at all like classification. Iterative situation–action tasks come to mind such as riding a bicycle, solving an equation, or parsing a sentence. The extension becomes obvious when for the decision classes we choose names which refer to partitions of the space of situations as 'suitable for action A', 'suitable for action B', etc.

Why should one want, in addition to finding a machine-efficient representation of the above mapping, to construct an operationally redundant superstructure to capture causal relations and to support 'what-if' planning? Answers suggest themselves as soon as one moves to the more demanding definition which animates the characteristically AI approach to learning:

> a learning system uses sample data to generate an up-dated basis for improved classification of subsequent data from the same source and expresses the new basis in intelligible symbolic form.

The requirement for social communication of the 'improved basis' now forces the issue. If synthetic autopilots are to show 'understanding' of flight situations and their own responses, then however necessary heuristic models may continue to be for the sub-structures of skill, insightful performance and explanation at higher and more strategic levels demands causal modelling of a sophisticated kind. It is towards this difficult objective that much work of the kind here reviewed is now turning (see Bratko, 1991).

5. SUMMING UP

In the debate between symbolic and neural-net representations, the two sides have tended to overlook the possibility that different parts of the brain, specialized to address different purposes, employ different representations. Specifically such differing purposes can be broadly grouped under two contrasted main heads:

(1) 'run-time' thinking;
(2) communication of the process and its outcome.

For (1), there is no obvious biological reason to expect symbolic representations to have evolved, in the sense in which 'symbolic' is here used. Indeed there is little evidence that such structures are employed in the brain's real-time problem-solving, some of which is critically supported by varieties of visual and spatial reasoning associated with the brain's right cerebral cortex, and by subcognitive procedures. But the social dissemination of knowledge and thought listed under (2) is

of such predominant importance in our species that elaborate symbolic mechanisms have emerged to support the execution of this function. The evolutionary processes have partly been biological and partly cultural. Eccles (in Popper and Eccles, 1987) paints a picture of divergent specialization between the two hemispheres of the brain according to which the 'minor' (usually the right) hemisphere plays roles central to run-time problem-solving, involving pattern-handling and spatial and social orientation. Yet this hemisphere almost wholly lacks capabilities of symbolic reasoning, notably those associated with language and logic. The dominant (usually left) hemisphere, by contrast, not only fluently handles the decipherment of linguistic and logical expressions, but is also the clearing-house for reports on subgoal attainment during problem-solving. Eccles argues that, although 'consciousness' is also manifested by the right hemisphere in the sense of a diffuse awareness, the focussed and organized forms of goal-oriented awareness which we associate with 'self' are functions of the left brain. More recently the possibility has been aired in neurobiological circles (see Benjamin Libet's observations and associated discussion in Behavioural and Brain Science, 1988-89) that the seat of consciousness acts more as a news room than as a planning headquarters, putting a coherent retrospective gloss on the consequences of decision. The decisions themselves, in this model, emanate from activities localized elsewhere. An elaboration of this view has recently been developed by Dennett (1992). Whatever the neural nature of functions (1) and (2) above, modern brain science sees them as operationally and topographically distinct. In such a view, the mechanisms of (2) face a serious problem. Modules specialized to symbolic reporting must interface with dissimilar, even alien, architectures if explanations of the 'self's problem-solving decisions are to be generated. When required to support the more intuitive field of real-time skills, the brain's explanation module tends to fail, or resorts, when pressed by the dialogue-elicitation specialist, to confabulation.

Are we, as engineers of cognition, obliged to burden intelligent artifacts with similar problems? On the contrary, to do so would seem the height of folly. Moreover, from such work as has

here been reviewed, an alternative strategy is available. We can treat expert sub–cognition as a 'black box' from which articulate models can be extracted. The product: symbolic models of sub-symbolic behaviour, or, more concretely, machine-executable yet articulate skills from 'silent' brains.

Acknowledgments

Thanks are due to Professor Quinlan for active assistance in the use of his C4.5 algorithm. The preparation of the review also benefited from criticisms and suggestions made by Dr. Patricia Smith Churchland, Mrs. J.E. Hayes Michie, Prof Colwyn Trevarthen, and by colleagues in the helicopter division of the Royal Aerospace Establishment, Bedford, UK, and at Advanced Rotorcraft Technology Inc., Mountain View, USA. We were helped by facilities at the Turing Institute, UK, at the Department of Computer Science, University of New South Wales, Australia, and at the Department of Statistics, Virginia Tech, USA. Thanks are also due to the Japan Society for Artificial Intelligence, in whose 1991 Proceedings some of the material of this article first appeared. Rui Camacho is supported by a scholarship from Junta Nacional de Investigação Científica e Tecnológica (JNICT) in Portugal.

REFERENCES

Anderson, J.R. (1990). *Cognitive Psychology and its Implications* (Third Edition), W.H. Freeman & Co.

Bratko, I. (1991). *Qualitative modelling: learning and control*. Proc. AI-91, Prague. Copies also available from Electr. Eng. and Comp. Sci., Ljubljana University, Slovenia.

Camacho, R. (1992). *Laboratory notes on experiments with the ACM flight simulator*. Available from the author: Fax no: +44-865-273839. e-mail: camacho@prg.ox.ac.uk.

Chambers, R.A. and Michie, D. (1969). *Man–machine co-operation on a learning task*. In Computer Graphics: Techniques and Applications (eds. R. Parslow, R. Prowse and R. Elliott-Green), London: Plenum.

Dennett, D.C. (1992). *Consciousness Explained*. Little, Brown and Co.

Donaldson, P.E.K. (1960). *Error decorrelation: a technique for*

matching a class of functions, in Proc. Third Internat. Conf. on Medic. Electronics, 173-178.

Feigenbaum, E.A. and McCorduck, P. (1983). *The Fifth Generation:* Artificial Intelligence and Japan's Computer Challenge to the World, Reading, MA: Addison-Wesley.

French, R.M. (1990). *Subcognition and the limits of the Turing Test*. In Mind, *99*, pp.53-65.

Ginsberg, M.L. (ed. 1987). *Readings in Nonmonotonic Reasoning*, Los Altos, CA: Morgan Kaufman.

McCarthy, J. (1959). *Programs with common sense*. In Mechanization of Thought Processes *Vol. I*, London: Her Majesty's Stationary Office. Reprinted in M. Minsky, ed. (1960), *Semantic Information Processing*, Cambridge, MA: MIT Press.

Makarovic, A. (1987). *Pole-balancing as a benchmark problem for qualitative modelling*. Technical Report DP-4953, Ljubljana: Josef Stefan Institute. Revised as (1991): A qualitative way of solving the pole-balancing problem. In Machine Intelligence, *12* (eds. J.E. Hayes, D. Michie and E. Tyugu), Oxford University Press.

Michie, D. (1988). *The Fifth Generation's unbridged gap*. In A Half-Century of the Universal Turing Machine (ed. R. Herken), Oxford University Press.

Michie, D. (1989). *Problems of computer-aided concept formation*. In Applications of Expert Systems 2 (ed. R. J. Quinlan). Wokingham and Reading, MA: Addison Wesley, pp 310-333.

Michie, D. (1991). *Methodologies from machine learning in data analysis and software*. Computer Journal, *34*, 559-565.

Michie, D., Bain, M. and Hayes-Michie, J. E. (1990). *Cognitive models from subcognitive skills*. In Knowledge-based Systems in Industrial Control (eds. Grimble,M., McGhee, S. and Mowforth, P.), Peter Peregrinus.

Michie, D. and Chambers, R.A. (1968) *BOXES: an experiment in adaptive control*, In Machine Intelligence *2* (eds. E. Dale and D. Michie), Edinburgh: Edinburgh University Press.

Newell, A. and Simon, H.A. (1976). *Computer science as empirical inquiry: symbols and search*. Commun. of the ACM, *19*, 113-126.

Nii, H. P. (1986). *Blackboard systems: the blackboard model of problem solving and the evolution of blackboard architectures*. In AI Magazine, *7*, 38-53.

Popper, K.R. and Eccles, J.C. (1977). *The Self and its Brain*. London and New York: Routledge and Kegan Paul.

Posner, M.I. (1973). *Cognition: An Introduction*, Glenview, IL: Scott, Foresman.

Quinlan, J.R. (1987). *Generating production rules from decision trees*. Internat. Joint Conf. on Art. Intell. 1987 (IJCAI-87), Los Altos, CA: Kaufmann, pp.304-307.

Sammut, C. (1988). *Experimental results from an evaluation of algorithms that learn to control dynamic systems*. In Proc. Fifth Internat. Conf. on Machine Learning (ed. J. Laird), San Mateo: Morgan Kaufmann.

Sammut, C. (1992). *Automatically constructing control systems by observing human behaviour*. In Proc. of the Internat. Workshop on Inductive Logic Programming, June 6-7, 1992, Tokyo, Japan.

Sammut, C., Hurst, S., Kedzier, D. and Michie, D. (1992). *Learning to fly*. In Proc. Ninth Intern. Machine Learning Conf. (eds. D.H. Sleeman and P. Edwards), San Mateo, CA: Morgan Kaufman, pp. 385-393.

Sammut, C. and Michie, D. (1991). *Controlling a 'black box' simulation of a space craft*. The AI Magazine, Amer. Assoc. for Artif. Intell., *Vol. 12* (Part 1, Spring), pp. 56-63.

Simon, H.A (in press) *Machine as mind*, In Proceedings of the Turing 1990 Colloquium, 3-6 April 1990, Brighton, UK (ed. Millican, P.) to appear.

Slocombe, S., Moore, K. and Zelouf, M. (1986). *Engineering expert system applications*. Presented at BCS Annual Conference, December 1986.

Squire, L. R. (1987). *Memory and Brain*, Oxford University Press.

Sterling, L. and Shapiro, E. (1986). *The Art of Prolog*, Cambridge, MA: The MIT Press, p.357.

Widrow, B. and Smith, F.W (1964). *Pattern recognising control systems*, In Computer and Information Sciences (eds. Tou, J.T. and Wilcox, R.H.). Clever Hume Press.