Machine intelligibility and the duality principle

Stephen Muggleton
Oxford University Computing Laboratory
Parks Road
Oxford, OX1 3QD,
United Kingdom.

Donald Michie
Professor Emeritus
University of Edinburgh
United Kingdom.

Abstract

The scale and diversity of networked sources of data and computer programs is rapidly swamping human abilities to digest and even locate relevant information. The high speed of computing has compounded this problem by the generation of even larger amounts of data, derived in ways that are generally opaque to human users. The result is an increasing gulf between human and computer abilities. Society's ever wider-scale dependence on rapidly growing networked sources of software threatens severe breakdowns if machine intelligibility issues are not given high priority.

In this paper we argue that lack of machine intelligibility in human-computer interactions can be traced directly to present approaches to software design. According to the duality principle in this paper, software involved in human-computer interaction should contain two distinct layers: a declarative knowledge-level layer and a lower-level functional or procedural-knowledge layer. This extends the formal methods separation of specification and implementation by requiring that the declarative layer be capable of extensive human interrogation at runtime. The declarative layer should support simple deductive and inductive inference. The ease with which declarative knowledge can be translated to natural language could be used to provide a human-comprehensible "window" into the properties of the underlying functional layer. Adaptation of the declarative knowledge in response to human interaction could be supported by modern machine learning mechanisms. In addition, declarative knowledge could be used to facilitate human-comprehensible communication between programs. Existing well-developed technologies can be used to implement the declarative layer. The obvious language of choice is pure

Prolog, augmented with machine learning mechanisms based on Inductive Logic Programming. The underlying functional layer would be composed of normal procedurally encoded computer programs. It is argued that the duality principle in software design is a necessity for dealing with the demands of wide-scale computer usage in the information age and should be an urgent goal for computer science research at the start of the 21st century.

1 Introduction

In 1964 Gordon Moore, then president of Fairchild, stated that the component density of integrated circuits would continue doubling every year (presently every two years). Such exponential change in both memory cost and computer speed has held true consistently over the last 40 years of computing. This dramatic increase in computing power has led to the widely heralded "information revolution", the social implications of which are rivaled only by the industrial revolution of the 18th and 19th centuries. Knock-on effects include a shift within the industrialised world from manufacturing-based economies to information-based ones.

The consequences of low cost memory and computer speed have not all been positive. The scale and diversity of data sources is rapidly swamping human abilities to digest and even locate relevant information. The high speed of computing has compounded this problem by generation of even larger amounts of data, derived in ways that are generally opaque to human users. The result is an ever widening gulf between human and computer abilities.

Thus although the parameters of computer hardware abilities are increasing exponentially, those of their human users remain constant and are already dwarfed in all respects. Figure 1 (borrowed from [22]) tabulates some of the information parameters of the human brain.

It is really human comprehension which is the key bottleneck in a number of the related problems of human-computer incompatibility. *What is required is both increased intelligence on the part of machines and increased compatibility in human-computer interaction.*

The need to increase the compatibility between computers and their human users necessitates the practical application, integration and extension of existing technologies.

This paper is structured as follows. The motivations for BT's Machine Intelligence initiative are introduced in Section 2. As a response to the urgent need for increased machine intelligibility we introduce the principle of software duality in Section 3. In Section 4 previous large-scale Artificial Intelligence projects are reviewed in the light of the duality principle and lessons are drawn for BT's ongoing Machine Intelligence initiative [34]. Sections 5 and 6 review the relevant ongoing research in autonomous agents and machine learning. It is argued that

1	Rate of information transmission along any input or output channel	30 bits per second
2	Maximum amount of information explicitly storable by the age of 50	10^{10} bits
3	Number of mental discriminations per second during intellectual work	18
4	Number of addresses which can be held in short-term memory	7
5	Time to access an addressable 'chunk' in long-term memory	2 seconds
6	Rate of transfer from long-term to short-term memory of successive elements of one 'chunk'	3 elements per second

Figure 1: Some information parameters of the human brain. Estimation errors can be taken to be around 30 per cent. (Main sources: Miller (1956) *Psychology Review*, 63, pp. 81–97, Stroud (1966) Ann N.Y. Academy, and sources cited by Chase and Simon (1974) *Cognitive Pyschology*, 4 pp. 55-81.)

present research falls far short of addressing the challenges raised by the problems in human-computer incompatibility we have described above. A radically new approach allowing the development of adaptive collaborative software[1] is called for. This approach should both build on and transform existing software technologies. Section 7 summarises and concludes the paper.

2 BT's Machine Intelligence initiative

Machine Intelligence is the software technology of user friendliness at the level of concepts, rather than the level of keystrokes and error message. The following quote from [34] provides the central motivation behind BT's Machine Intelligence initiative.

> Man-to-machine communications is a major business opportunity. The rapid growth in the use (and processing power) of computers both in the home and in the workplace is leading to the situation where the market for "man-to-machine traffic" is growing fast.

The development of high-speed communications and cheap high powered PCs is rapidly steering the dominant use of information technology into the hands of users who are not computer professionals. Despite the wide-scale use of window and mouse interfaces, the level of machine intelligibility is extremely low. The following is a short list of interaction issues familiar to any computer user.

[1] This terminology is derived from Hyacinth Nwana's classification [31] of intelligent agents. Nwana also notes that no existing agents are both collaborative and adaptive.

Program purpose. It is generally not possible to query the purpose of a computer program's actions.

Human intentions. Computer programs generally have no model of their human user's aims and motivations, and thus lack the ability to make helpful suggestions.

Failure diagnosis. Execution failures are generally hard to track down.

Correction. Erroneous software behaviour will be repeated indefinitely, since present-day software has no way to incorporate corrective feedback.

Brittleness. Computer programs are unable to make intelligent conjectures in the face of slightly incomplete or incorrect data.

Hidden interactions. The communication between programs are opaque to the human user.

Many of these features have for some time acted as nagging irritants to computer users. However, with the wider-scale use of computers such problems start to impinge not only on business profitability, but also in some situations on human safety.

In the next section we urge the need for software involved in human-computer interaction to have an extra declarative layer defining program specifications, reaction speeds of sub-modules, goals, intentions as well as models of other computer programs and human users.

3 Machine intelligibility and the duality principle

We view *machine intelligibility* as a state of human-computer interaction. It can be defined as follows.

> A two-way interaction with a machine is intelligible to a human user if the (growing) set of the machine's concept descriptions used in the interaction maintain logical equivalence with a (growing) subset of the human's concepts.

We see the *duality principle* of software design as a necessary requisite for machine intelligibility. The software duality principle can be stated as follows.

> Software involved in human-computer interaction should be designed at two interconnected levels: a) a declarative, or self-aware level, supporting ease of adaptation and human interaction and b) a procedural, or skill level, supporting efficient and accurate computation.

3.1 Diary example

As an example of software designed without the duality principle, consider the humble electronic diary, available on most PCs and workstations. New entries are put in such diaries by pointing with a mouse at a representation of an open page, clicking and typing text. However, suppose one has a regular meeting every Thursday at 4pm for a period of 8 weeks. Rather than forcing the user to put in each entry separately some advanced models of electronic diary will provide a simple mechanism for specifying repetitive entries of this kind. However, you cannot specify that these meetings must avoid public holidays, or ask for it to be taken into account that one might want to miss meetings that fall on one's child's birthday. Nor will they take into account the simplest of temporo-spatial rules, such as the fact that a participant of an event cannot be in two distant places at the same time. Present electronic diaries have no facilities to build up such human-oriented concepts, nor to make intelligent human-checkable conjectures in terms of already defined concepts. This is no doubt due to the difficulties of programming such abilities in a procedural programming language.

However, imagine the situation if the duality principle had been applied in the design of an electronic diary. Events with their associated participants and spatial designations are easy and efficient to represent within a declarative language such as the datalog subset of Prolog. Such events could be asserted into a deductive database as a side-effect of mouse clicking. Integrity constraints such as the requirement for a person to be at only one place at any time are again straightforward to encode in datalog. Interaction in a commonly defined declarative representation with other declaratively encoded diary programs/birthday planners/travel planners would allow integrity checking to extend beyond the local knowledge-base. One would certainly not want to burden a user by requiring that they learn mathematical logic before they can operate their electronic diary! However, rules that state that a series of meetings happens every Thursday except during public holidays and your children birthday's could be conjectured and refined by an Inductive Logic Programming (ILP) [30] system from a handful of examples and counter-examples entered by mouse-clicking. Such rules could be easily translated into natural language for user certification. Within such a diary procedural encoding would still be vital for numerical calculations involving time and space, graphical display and mouse input.

3.2 Declarative and procedural knowledge

The divisions between declarative and procedural knowledge has its counterparts not only throughout the behavioural and brain sciences but also throughout computer science. Thus, for instance, Hoare [14] distinguishes between *specifications* and *program* as follows.

> Given specification S, the task is to find a program P which satisfies it in the sense that every possible observation of every possible

behaviour of the program P will be among the behaviours described by (and therefore permitted by) the specification S.

Note that, unlike programs, specifications are both declarative and human-oriented. This is typically achieved by making use of variants of first-order predicate calculus such as Z [35]. However, the purpose of specifications is largely fulfilled once the program has been correctly implemented. Specifications are not intended to be used for runtime interrogation and they do not include knowledge about the program's use and environment beyond its input-output behaviour.

Within the field of AI, the knowledge representation debate has polarized between advocates of purely declarative representations (such as McCarthy) and those supporting purely reactive representations (such as Brooks). This split and its potential resolution via the duality principle will be explored in more detail in Section 5.

Intelligent interaction has been a central theme within Artificial Intelligence. For this reason we review some of the large-scale AI projects and relevant sub-disciplines of AI in the next section.

4 Review of relevant AI research

4.1 Review of previous large-scale Artificial Intelligence projects

BT's Machine Intelligence initiative is both timely and important in attempting to address the needs for intelligent software in the 21st Century. We believe that such a project could substantially alter the landscape of computing. However, to do so it will be necessary to take account of the advances and mistakes of previous large-scale Artificial Intelligence projects. For these purposes we review the Japanese Fifth Generation project and the US CYC project.

4.1.1 Japanese Fifth Generation Project

The Japanese Fifth Generation Computing Systems (FGCS) project [6] was one of a number of large-scale computing projects of the 1980s aimed at the construction of intelligent machinery. Intensive planning and research led to the identification of Horn clause logic as the best candidate for a single software representation for all parts of the project. Since this representation was aimed at underpinning a whole new generation of machines, the vision acted as a spur to the Logic Programming community to remodel all the major constituents of computer science within a logic programming framework. The FGCS project aimed at using logic programming for implementing man-machine communication aids, visual and speech input-output, sensory robot interfaces, and natural language database interfaces. Efficiency was to be achieved by massive parallelism, based on Personal Inference Machines (PIMs) whose assembly language was an augmented version of Prolog. Cognitive compatibility between the new ultra-powerful machines and their users would be ensured by the use of knowledge-based programming techniques. The result would be intelligent machines which made the fruits of large-scale knowledge bases available to their users via interactive natural language interfaces. What went right and what went wrong?

The project met many of its technical aims. At the end of the project large-scale parallelism based on a Guarded Horn Clause (GHC) language was demonstrated on a number of impressive applications, including a theorem prover which proved an open result in number theory. Despite this success large-scale knowledge bases were not built and the vision of intelligent natural language oriented machines was never realised. In addition, the hardware and software were not taken up by Japanese industry due partly to their incompatibility with existing software.

The lasting value of the FGCS project can be largely attributed to its choice of a single declarative knowledge representation, which connected a large number of disparate problems into a coherent whole[2]. As a side-effect intense international research in the 1980s enabled Logic Programming to encompass the disparate computer science topics of knowledge representation, semantics, databases, program termination, formal methods, program synthesis, debugging, modularity, constraint solving, induction and natural language processing (see Ten Year Review Special Issue of the Journal of Logic Programming, Volumes 19/20, 1994). On the other hand, the failure of FGCS to construct large-scale knowledge-bases and natural language capabilities has been attributed [23] to the failure of the planners to recognise the central role that should have been played by machine learning. Prof. K. Furukawa, Research Director of the Japanese Fifth Generation project, has admitted privately that a central error of theirs was the initial omission of computer induction when planning the project. Since 1992, Prof. Furukawa [10] has focussed his main research area on the application of "Inductive Logic Programming systems to various problems including natural language processing, skill acquisition and economics".

4.1.2 US CYC Project

In response to the Japanese FGCS project, the US started its own privately funded Artificial Intelligence project called CYC at MCC in Austin Texas. CYC was under the technical directorship of Doug Lenat, who strongly advocated the need to build up a massive, diverse knowledge source. Unlike FGCS Lenat believed that it was not important to choose a particular knowledge representation from the start, but rather to let one evolve.

[2] This was a lesson lost on the so-called Japanese Sixth Generation (or real-world computing) project, which after 4 years has yet to make a significant impact outside Japan.

The first author of this paper visited MCC in 1984 while carrying out a consultancy at Radian Corporation, also in Austin Texas. Radian's approach to building knowledge bases, much more pedestrian than CYC's, involved the use of "structured" rule induction. During the 1980s Radian's RuleMaster product was used to build a number of expert systems including an autolander for the Space Shuttle and what is still one of the world's largest expert systems, BMT. 30,000 rules were developed in 9 man years of development time. The BMT system was installed at Siemens and is still in full-time use. Simple "first-generation" machine learning techniques have achieved many other similar successes in application [18].

By comparison, after 10 years effort and 100s of million of dollars of funding the CYC [19] system failed to find large-scale industrial application. Like FGCS, the self-imposed requirement to enter all knowledge by hand proved overwhelming. Unlike FGCS, the failure to choose a sufficiently expressive common representation language was discovered as an oversight, near the end of the project. The following is a quote from Lenat [19] on this topic.

> Another point is that a standard sort of frame-and-slot language proved to be awkward in various contexts: ... Such experiences caused us to move toward a more expressive language, namely first order predicate calculus with a series of second-order extensions ...

Thus the CYC project eventually concurred with the original design choice of FGCS, to use first order predicate calculus as the central declarative knowledge representation.

4.1.3 Lessons for BT's Machine Intelligence initiative

Can BT's Machine Intelligence project succeed where projects like FGCS and CYC have failed to meet expectations? We would argue that the answer is yes, but only if sufficient note is taken of the lessons to be learned from FGCS and CYC. From the above discussion we take the following to be necessary, though not sufficient design choices for the project.

- **Declarative representation.** Choose first order predicate calculus as the central declarative knowledge representation, as FGCS did and CYC eventually had to do. In practice this implies coding knowledge-bases in Prolog-like programming languages. This representation is both flexible and supports comprehensibility due to its close relationship with natural language.

- **Learning.** Do not rely on hand-coding of knowledge. Machine Learning, when properly applied can generate hundreds of lines of validated code per day. The advent of ILP [4] ties this point to that of representation, since efficient systems now exist [32, 28] which use logic programming as their sole representation for example databases, background knowledge and constructed theories.

- **Duality.** Build links to existing efficient procedural computer software to avoid recoding and enhance take-up in existing user communities. This approach is exemplified to a limited degree by constraint logic programming [15], in which logical theorem proving interacts with efficient procedurally encoded constraint solving.

An alternative to CYC's approach for obtaining large-scale knowledge structures might be imagined by analogy with the World-Wide-Web. By this analogy we might expect an interconnected network of interacting simple programs, such as the declarative diary program in Section 3.1, to build up a massive executable network of personal and social knowledge by continuous interaction with a population of human users. The inferential power of such a "knowledge network" would be substantially beyond the hopes or aspirations of either FGCS or CYC.

Two of the most pertinent topics in present Machine Intelligence research relevant to increased intelligibility in human computer interaction are Intelligent Agents and Machine Learning. Issues related to both topics are thrown up by the declarative diary example of Section 3.1. These topics will be reviewed in the following two sections.

5 Review of Agents

A rift has opened up in Artificial Intelligence circles between the advocates of, on the one hand, declarative knowledge-based systems and, on the other hand, reactive knowledge-free systems. These two approaches can be exemplified by the views expressed by John McCarthy (Stanford) and Rodney Brooks (MIT). The distinction between these disparate approaches explains some of the diversity in the usage of the term "agents" in the recent Artificial Intelligence (AI) literature.

5.1 The declarative school

On the declarative side John McCarthy [21] claims that nothing short of second-order predicate calculus will do for implementing "robot consciousness". He underlines the necessity of agents being able to reason about other agents' beliefs, including their own. Despite its declarative elegance this approach runs into problems with tractability and completeness of inference. Genesereth [11] has followed McCarthy's lead and produced specific plans for agent-oriented software based on the logical communication languages KQML and KIF [9]. These languages, which are extensions of first-order Horn clause logic (in a LISP-like syntax) allow declarative definition and communication of agent knowledge. Genesereth advocates the idea of "software wrappers" which allow existing legacy software to be converted to agent-oriented form by embedding within a KQML/KIF exterior. For this aspiration to be met on a large-scale would require converting software vendors across the board to the new KQML/KIF standards. This approach is related to the duality principle, though it lays no stress on the importance of machine learning or human-computer communication. Also the KQML/KIF attempts to standardise predicate usage in inter-agent communication have been contended even within the limits of the AI community [12]. It is our belief that, in contrast to Genesereth's top-down approach, large-scale usage of agent software—like large-scale usage of the WWW—is more likely to be achieved bottom-up, by way of adaptive personalised agent interfaces.

The one limited form of adaptive personalised agent interface that already exists is the Softbot framework developed by Etzioni and Weld [8]. Softbots employ a first-order predicate calculus representation and are primarily planners. Given an imprecisely specified goal, such as "set up a meeting on agents with Mitchell at CMU", the Softbot plans a precise set of commands to execute this goal. For example this might include determining the email address of Tom Mitchell at CMU. Although Softbots use an expressive representation they do not communicate with one another. They perform learning, but only by rote memorisation of plans that succeeded or details that proved useful in the past.

5.2 The reactive school

At MIT Rodney Brooks is advocating and leading a programme for building reactive insect-like robots [5]. High-level cognitive compatibility with humans is abandoned in favour of stimulus-response architectures, which lend themselves to straightforward robot implementations imported from mechanical engineering. Some work in Pattie Maes' group at MIT appears to be exploiting some of Brooks' ideas where it is applied to software agents [20, 3]. This approach is sometimes referred to as *artificial life* and is summarised in the following quote from Maes [20].

> The goal of building an autonomous agent is as old as the field of AI itself. The artificial life community has initiated a radically different approach to this goal, which focuses on fast, reactive behaviour, rather than on knowledge and reasoning, as well as on adaptation and learning. Its approach is largely inspired by biology, and more specifically the field of ethology, which attempts to understand the mechanisms animals use to demonstrate adaptive and successful behaviour.

Maes and her colleagues have demonstrated an artificial dog as well as various reactive agent utilities which learn within a non-declarative representation for tasks such as email filters.

At CMU Mitchell and colleagues have developed a reactive agent called *Web-Watcher* [2] that learns about user interests on the World-Wide-Web. Like other reactive agents, although it adapts it cannot communicate learned knowledge to other agents or to users. *WebWatcher* has the merit of being freely available over the Web from the following site.

http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-6/web-agent/www/project-home.html

Another class of agents that best fits within this category is that of Knowbots [7]. Knowbots perform a task similar to that of Softbots (Section 5.1) except that rather than employing planning, they carry out hard-wired commands which are encoded procedurally. Communication between Knowbots is opaque to human users and Knowbots do not learn from experience.

Genesereth's approach lends itself to high-level knowledge communication, inter-robot modelling and collaboration. However, due to the nature of the languages used no learning has been attempted within this framework. On the other hand, Maes' approach does not support high-level communication but has strengths in the ease of incorporation of low-level learning. However, despite the incorporation of learning within a number of the Maes' agent projects, the assumptions made and learned are opaque to human users.

5.3 Bridging the gap using the duality principle

The second author [24, 25] has proposed a pragmatic position on conscious agents which bridges McCarthy's and Brooks' programmes. In settings in which agents need to interact with human users there is a need to integrate both Brooksian subcognitive software with McCarthyite self-articulate 'conscious' abilities. This approach supports the duality principle of Section 3. We see a clear need for adaptive learning at both cognitive and subcognitive levels. Followers of McCarthy's programme have achieved agents which communicate and are autonomous but do not learn. Followers of Brooks have constructed agents which are autonomous and learn but do not communicate with each other. Neither of these approaches can be seen to follow the duality principle of Section 3. The papers [33, 26] appear to be the only ones in the literature to have demonstrated autonomous agents which communicate with one another and learn. These agents each control an independent control surface of a simulated Cessna aircraft. Inter-agent communication is achieved via a common 'blackboard' which describes the current state of the aircraft. The control strategy of individual agents is defined by situation/action mappings represented as decision trees. These trees are learned independently from observations of human pilot traces. Since the low-level of decision tree representation impedes communicability of the learned knowledge, [25] suggests the future use of first-order predicate calculus and Inductive Logic Programming for the construction of human comprehensible rules.

This paper further endorses the bridge-building approaches previously proposed by the second author. We accept the necessity within agents for efficient and reactive low-level functionality, communication and learning. However, adaptive collaborative agents which interact with human users need a corresponding layer of declarative social and self-knowledge which can be autonomously learned and succinctly and exactly communicated to humans and other agents.

6 Review of Learning

Unlike the topic of software agents, Machine Learning (ML) is a well established field of AI with many international conferences and its own reputable journal. Topics covered range from the highly theoretical to the technical intricacies of implementation design and comparative testing. ML largely involves highly reactive representations for the learned knowledge, ranging from statistical regression equations to neural networks, decision trees, boolean functions and finite state automata. The 1990s have seen the rapid development of Inductive Logic Programming (ILP) [30], the only subarea of ML directed at the declarative representations favoured by McCarthy and Genesereth (Section 5.1).

6.1 Surveys of applied ML

For the purposes of this paper we will narrow the field to that of real-world applications of ML. Despite the fact that ML is one of the largest and fastest growing areas of AI, until recently relatively few of the numerous industrial applications of ML had been detailed in the computer science literature. There are several reasons for this.

1. **Confidentiality.** Successful ML applications often involve commercially sensitive data (eg. personal credit limit information) and often give the users a substantial edge which would be lost on communication of the resulting knowledge to competitors.

2. **Trade publication.** Unlike academia, there is little advantage to company employees who publish in learned journals. ML applications are often published in the relevant trade journals, or described in advertising material.

However, two recent issues of the Communications of the ACM (March 1994 and November 1995) provide in-depth surveys of real-world applications of ML. These surveys cover case-based learning [1], neural networks [36], genetic algorithms [13], rule induction [18] and ILP [4]. As a prime example Langley and Simon describe a chemical process control application of rule induction which saved Westinghouse ten million dollars per year.

All the surveyed ML techniques except ILP are highly efficient and reactive, use relatively simple feature-based representations, employ search techniques with implicit biases which are usually not understood by the user and generally produce results which are relatively opaque. These techniques are ideal for "black-box" reactive agents. By contrast, ILP is relatively slow in generation of hypotheses, but allows a rich relational representation, has an explicit search bias (background knowledge) and is the only one to have produced discoveries which are not only comprehensible to human experts but also represent new knowledge publishable on its own account in top scientific journals [29, 16, 17].

6.2 The Statlog project

Whereas the quality of new declarative knowledge can be established by its publishability in refereed journals, the contending claims of "black-box" reactive learning mechanisms can best be judged by head-to-head performance comparisons on extensive datasets. Statlog [27], the largest such comparative machine learning study to date, was an ESPRIT project that ran from 1990 to 1993 and involved 6 academic and 6 industrial laboratories. Around 20 reactive rule induction, neural and statistical classification algorithms were each tested and compared on around 20 large-scale industrial datasets to produce $20 \times 20 = 400$ large scale experiments. The results were compared and analysed by academic statisticians. Reactive rule induction algorithms were found to have the best average performance, though they could be outstripped on particular datasets by statistical algorithms which made assumptions appropriate to the dataset. Unfortunately, no attempt was made to assess quantitatively the comprehensibility of the learned information.

The general lesson seems to be that 'black box' learning should be carried out with whatever algorithm, or combination of algorithms (as long as they are reasonably efficient), give highest predictive accuracy (or minimum cost given a cost/benefit matrix). However, when learned knowledge must be intelligible to the user, a concise, exact and declarative representation, such as that used by ILP, is ideally suited. It should also be noted that Statlog exclusively investigated unstructured learning. Meanwhile, as is the case with good software engineering practice, the interesting machine learning technologies for large-scale knowledge development will be incremental and develop highly structured representations by hierarchical decomposition.

7 Conclusion

In this paper we have investigated some of the general issues related to computer science research at the end of the 20th century, and tried to indicate some of the central issues for the next century. In particular we have stressed the human overload dangers inherent in computing technology. These stem from the mass availability of high information capacity, transmission rates and computation rates. Without sufficient intelligent support, human users will be increasingly overpowered by computer power. This is a central motivation for BT's Machine Intelligence initiative. However, we note that similar 1980s projects, namely FGCS and CYC, failed to deliver their promise. We trace this failure largely to the absence of declarative machine learning techniques within these projects. The renaissance of Artificial Intelligence techniques under the guise of the Intelligent Agent's movement, is in danger of repeating the same mistake as FGCS and CYC. Thus most agent technology has no learning capacity, and those agent techniques that do incorporate learning, do not allow for the communication of learned knowledge to human beings.

Our suggestion is that a new approach be taken to designing software which interacts with human beings. We believe such software should incorporate declarative machine learning at its core. According to our "duality principle" machine intelligibility can only be obtained by clearly separating the declarative and procedural knowledge components of a program. While the procedural part of the program should carry out low-level communication inaccessible tasks, this should be directed by high-level, declarative knowledge. This declarative knowledge should be encoded in a logic language, thus making it easily translatable to natural language. This would not only actively support high level interaction with human beings, but also make inter-program communication transparent to inspection. Using today's machine learning technology, learning at the declarative level would necessarily be carried out using Inductive Logic Programming.

We urge the need for software involved in human-computer interaction to have an extra declarative layer defining program specifications, reaction speeds of sub-modules, goals, intentions as well as models of other computer programs and human users.

We believe that if BT were to take this route with the Machine Intelligence initiative, they would not only be carrying out developments at the forefront of emerging computer science technologies, but also might help transform the communication and computing industries of the 21st Century.

Acknowledgements

The authors would like to thank Nader Azarmi, Robin Smith, Hyacinth Nwana, Ben Azvine and Brian Tester for enlightening discussions on agents and other related topics at British Telecom Research Labs at Martlesham Heath. Thanks are also due for discussions on the topics in this paper with John McCarthy, David Page and Lincoln Wallen. Stephen Muggleton and David Page's work on this topic was supported by a period of consultancy at British Telecom during 1995-1996, as well as by the Esprit Long Term Research Action ILP II (project 20237), EPSRC grant GR/J46623 on Experimental Application and Development of ILP, EPSRC grant GR/K57985 on Experiments with Distributionbased Machine Learning and an EPSRC Advanced Research Fellowship held by Stephen Muggleton. Discussions on these topics between the authors was supported at Oxford University Computing Laboratory by EPSRC Visiting Fellowships granted to Prof. Michie in 1994 and 1995. During the discussion period Stephen Muggleton and Donald Michie were supported respectively by an ongoing Research Fellowship and a Visiting Scholarship at Wolfson College Oxford.

References

[1] B.P. Allen. Case-based reasoning: business applications. *Communications of the ACM*, 37(3):40–44, 1994.

[2] R. Armstrong, D. Freitag, T. Joachims, and T. Mitchell. Webwatcher: a learning apprentice for the World Wide Web. In *AAAI Spring symposium on Information Gathering from Heterogeneous, Distributed Environments*, Stanford, 1995. http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-6/web-agent/www/project-home.html.

[3] B. Blumberg. Lessons from ethology for autonomous agent architectures. In K. Furukawa, D. Michie, and S. Muggleton, editors, *Machine Intelligence 15: intelligent agents.* Oxford University Press, Oxford, 1996. (To appear).

[4] I. Bratko and S. Muggleton. Applications of inductive logic programming. *Communications of the ACM*, 38(11):65–70, 1995.

[5] R. Brooks. A robust layerd control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2:14–23, 1986.

[6] JIPDEC Fifth Generation Computing Committee. Preliminary report on study and research on fifth-generation computers in 1979-1980. Technical report, Japan Information Processing Development Centre, Tokyo, 1981.

[7] R. Droms. The Knowbot information service. FTP Report - Corporation for National Research Initiatives (CNRI), December 1989.

[8] O. Etzioni and D. Weld. A softbot-based interface to the internet. *Communications of the ACM*, 37(7):72–76, July 1994.

[9] T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML - a language and protocol for knowledge and information exchange. Technical Report CS-94-02, Computer Science Department, University of Maryland and Valley Forge Engineering Center, Unisys Corporation, Computer Science Department, University of Maryland, UMBC Baltimore MD 21228, 1994.

[10] K. Furukawa, D. Michie, and S. Muggleton. *Machine Intelligence 13: machine intelligence and inductive learning.* Oxford University Press, Oxford, 1994.

[11] M. Genesereth and S.P. Ketchpel. Software agents. *Communications of the ACM*, 37(7):48–53, 1994.

[12] M. Ginsberg. Knowledge interchange format: The kif of death. *AI Magazine*, 12(3):57–63, Fall 1991.

[13] D.E. Goldberg. Genetic and evolutionary algorithms come of age. *Communications of the ACM*, 37(3):113–119, 1994.

[14] C.A.R. Hoare. Programs are predicates. In *Proceedings of the Final Fifth Generation Conference*, pages 211–218, Tokyo, 1992. Ohmsha.

[15] J. Jaffar and M.J. Maher. Constraint logic programming: a survey. *Journal of Logic Programming*, 19/20:503–582, 1994.

[16] R. King, S. Muggleton, R. Lewis, and M. Sternberg. Drug design by machine learning: The use of inductive logic programming to model the structure-activity relationships of trimethoprim analogues binding to dihydrofolate reductase. *Proceedings of the National Academy of Sciences*, 89(23), 1992.

[17] R. King, S. Muggleton, A. Srinivasan, and M. Sternberg. Structure-activity relationships derived by machine learning: the use of atoms and their bond connectives to predict mutagenicity by inductive logic programming. *Proceedings of the National Academy of Sciences*, 93:438–442, 1996.

[18] P. Langley and H. Simon. Applications of machine learning and rule induction. *Communications of the ACM*, 38(11):54–64, 1995.

[19] D.B. Lenat. CYC: a large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

[20] Pattie Maes. Artificial life meets entertainment: lifelike autonomous agents. *Communications of the ACM*, 38(11):108–114, 1995.

[21] J. McCarthy. Making robots conscious. In K. Furukawa, D. Michie, and S. Muggleton, editors, *Machine Intelligence 15: intelligent agents*. Oxford University Press, Oxford, 1996. (To appear).

[22] D. Michie. The superarticulacy phenomenon in the context of software manufacture. *Proceedings of the Royal Society of London*, A 405:185–212, 1986.

[23] D. Michie. The Fifth Generation's unbridged gap. In Rolf Herken, editor, *The Universal Turing machine : a half-century survey*, pages 467–489. Oxford University Press, Oxford, 1988.

[24] D. Michie. Consciousness as an engineering issue, part 1. *Journal of Consciousness Studies*, 1(2):182–195, 1994.

[25] D. Michie. Consciousness as an engineering issue, part 2. *Journal of Consciousness Studies*, 2(1):52–66, 1995.

[26] D. Michie and C. Sammut. Behavioural clones and cognitive skill models. In K. Furukawa, D. Michie, and S. Muggleton, editors, *Machine Intelligence 14: applied machine intelligence.* Oxford University Press, Oxford, 1995.

[27] D. Michie, D.J. Spiegelhalter, and C.C. Taylor. *Machine learning, neural and statistical classification.* Ellis Horwood, London, 1994.

[28] S. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.

[29] S. Muggleton, R. King, and M. Sternberg. Protein secondary structure prediction using logic-based machine learning. *Protein Engineering*, 5(7):647–657, 1992.

[30] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19,20:629–679, 1994.

[31] Hyacinth Nwana. 'Smart' software agents: an overview. *Knowledge Engineering Review*, 11(3), 1996.

[32] J.R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[33] C. Sammut, S. Hurst, D. Kedzier, and D. Michie. Learning to fly. In D. Sleeman and P. Edwards, editors, *Proceedings of the Ninth International Workshop on Machine Learning*, pages 385–393, San Mateo, CA, 1992. Morgan Kaufmann.

[34] R. Smith, J. Callaghan, and B. Azvine. Development of a Machine Intelligence capability in BT. Techical feasibility report, British Telecom Research Laboratories, Ipswich, 1995.

[35] J.M. Spivey. *The Z notation : a reference manual.* Prentice-Hall, New York, 1992. (2nd edition).

[36] B. Widrow, D.E. Rumelhart, and M.A. Lehr. Neural networks: applications in industry, business and science. *Communications of the ACM*, 37(3):93–105, 1994.