Alan Turing and the development of Artificial Intelligence

Stephen Muggleton
Department of Computing, Imperial College London

December 19, 2012

Abstract

During the centennial year of his birth Alan Turing (1912-1954) has been widely celebrated as having laid the foundations for Computer Science, Automated Decryption, Systems Biology and the Turing Test. In this paper we investigate Turing's motivations and expectations for the development of Machine Intelligence, as expressed in his 1950 article in *Mind*. We show that many of the trends and developments within AI over the last 50 years were foreseen in this foundational paper. In particular, Turing not only describes the use of Computational Logic but also the necessity for the development of Machine Learning in order to achieve human-level AI within a 50 year time-frame. His description of the Child Machine (a machine which learns like an infant) dominates the closing section of the paper, in which he provides suggestions for how AI might be achieved. Turing discusses three alternative suggestions which can be characterised as: 1) AI by programming, 2) AI by *ab initio* machine learning and 3) AI using logic, probabilities, learning and background knowledge. He argues that there are inevitable limitations in the first two approaches and recommends the third as the most promising. We compare Turing's three alternatives to developments within AI, and conclude with a discussion of some of the unresolved challenges he posed within the paper.

1 Introduction

In this section we will first review relevant parts of the early work of Alan Turing which pre-dated his paper in *Mind* [42].

1.1 Early work: the Entscheidungsproblem

Turing's initial investigations of computation stemmed from the programme set out at the 1928 International Mathematical Congress by David Hilbert. Hilbert presented three key open questions for logic and mathematics. Was mathematics

1. *complete* in the sense that any mathematical assertion could either be proved or disproved.

2. *consistent* in the sense that false statements could not be derived by a sequence of valid steps.

3. *decidable* in the sense that there exists a definite method to decide the truth or falsity of every mathematical assertion.

Within three years Kurt Gödel [13] had shown that not even simple forms of arithmetic are both complete and consistent and by 1937 both Alonzo Church [5] and Alan Turing [43] had demonstrated the undecidability of particular mathematical assertions.

While Gödel and Church had depended on demonstrating their results using purely mathematical calculi, Turing had taken the unusual route of considering mathematical proof as an artifact of human reasoning. He thus considered a physical machine which emulated a human mathematician using a pen and paper together with a series of instructions. Turing then generalised this notion to a universal machine which could emulate all other computing machines. He used this construct to show that certain functions cannot be computed by such a universal machine, and consequently demonstrated the undecidability of assertions associated with such functions.

At the heart of Turing's universal machine is a model of human calculation. It was this choice which set the scene for later discussions on the degree to which computers might be capable of more sophisticated human-level reasoning.

1.2 Bletchley Park

The outbreak of the Second World War provided Turing with an opportunity and resources to design and test a machine which would emulate human reasoning. Acting as the UK's main wartime decryption centre, Bletchley Park had recruited many of the UK's best mathematicians in an attempt to decode German military messages. By 1940 the Bombe machine, designed by Turing and Welchman [7], had gone into operation and was efficiently decrypting messages using methods previously employed manually by human decoders. In keeping with Turing's background in Mathematical Logic, the Bombe design worked according to a *reductio ad absurdum* principle which simplified the hypothesis space of 26^3 possible settings for the Enigma machine to a small number of possibilities based on a given set of message transcriptions.

The hypothesis elimination principle of the Bombe was later refined in the design of the Colossus I and II machines. The Tunny report [15] (declassified by the UK government in 2000), shows that one of the key technical refinements of Colossus was the use of Bayesian reasoning to order the search through the space of hypothetical settings for the Lorenz encryption machine. This combination of logical hypothesis generation tied with Bayesian evaluation were later to become central to approaches used within Machine Learning (see Section 5). Indeed strong parallels exist between decryption tasks on the one hand, which

involve hypothesising machine settings from a set of message transcriptions and modern Machine Learning tasks on the other hand, which involve hypothesising a model from a set of observations. Given their grounding in the Bletchley Park decryption work it is hardly surprising that two of the authors of the Tunny report, Donald Michie (1923-2007) and Jack Good (1916-2009), went on to play founding roles in the post-war development of Machine Intelligence and Subjective Probabilistic reasoning respectively. In numerous out-of-hours meetings at Bletchley Park, Turing discussed the problem of machine intelligence with both Michie and Good. According to Andrew Hodges [16], Turing's biographer

> These meetings were an opportunity for Alan to develop the ideas for chess-playing machines that had begun in his 1941 discussions with Jack Good. They often talked about mechanisation of thought processes, bringing in the theory of probability and weight of evidence, with which Donald Michie was by now familiar. ... He (Turing) was not so much concerned with the building of machines designed to carry out this or that complicated task. He was now fascinated with the idea of a machine that could *learn*.

2 Turing's 1950 paper in Mind

2.1 Structure of the paper

The opening sentence of Turing's 1950 paper [42] declares

> I propose to consider the question, "Can machines think?"

The first six sections of the paper provide a philosophical framework for answering this question. These sections are briefly summarised below.

1. **The Imitation Game.** Often referred to as the "Turing test", this is a form of parlour game involving a human interrogator who alternately questions a hidden computer and a hidden person in an attempt to distinguish the identity of the respondents. The Imitation Game is aimed at providing an objective test for deciding whether machines can think.

2. **Critique of the New Problem.** Turing discusses the advantages of the game for the purposes of deciding whether machines and humans could be attributed with thinking on an equal basis using objective human judgement.

3. **The Machines Concerned in the Game.** Turing indicates that he intends digital computers to be the only kind of machine permitted to take part in the game.

4. **Digital Computers.** The nature of the new digital computers, such as the Manchester machine, is explained and compared to Charles Babbage's proposals for an Analytical Engine.

5. **Universality of Digital Computers.** Turing explains how digital computers can emulate any discrete-state machine.

6. **Contrary Views on the Main Question.** Nine traditional philosophical objections to the proposition that machines can think are introduced and summarily dismissed by Turing.

2.2 Learning machines - Section 7 of Turing paper

The task of engineering software which addresses the central question of Turing's paper have dominated Artificial Intelligence research over the last sixty years. In the final section of the 1950 paper Turing addresses the motivation and possible approaches for such endeavours. His transition from the purely philosophical nature of the first six sections of the paper is marked as follows.

> The only really satisfactory support that can be given for the view expressed at the beginning of section 6, will be that provided by waiting for the end of the century and then doing the experiment described. But what can we say in the meantime?

Turing goes on to discuss three distinct strategies which might be considered capable of achieving a thinking machine. These can be characterised as follows: 1) AI by programming, 2) AI by *ab initio* machine learning and 3) AI using logic, probabilities, learning and background knowledge. In the next three sections we discuss these strategies of Turing in relation to various phases of AI research as it has been conducted over the past half century.

3 Version 1: AI by programming [1960s-1980s]

3.1 Storage capacity argument

Turing considers an argument concerning the memory requirements for programming a digital computer with similar capacity to a human being.

> As I have explained, the problem is mainly one of programming. Advances in engineering will have to be made too, but it seems unlikely that these will not be adequate for the requirements. Estimates of the storage capacity of the brain vary from 10^{10} to 10^{15} binary digits. I incline to the lower values and believe that only a very small fraction is used for the higher types of thinking. Most of it is probably used for the retention of visual impressions, I should be surprised if more than 10^9 was required for satisfactory playing of the imitation game, at any rate against a blind man. (Note: The capacity of the Encyclopaedia Britannica, 11th edition, is 2×10^9). A storage capacity of 10^7, would be a very practicable possibility even by present techniques. It is probably not necessary to increase

> the speed of operations of the machines at all. Parts of modern machines which can be regarded as analogs of nerve cells work about a thousand times faster than the latter. This should provide a "margin of safety" which could cover losses of speed arising in many ways. Our problem then is to find out how to programme these machines to play the game. At my present rate of working I produce about a thousand digits of programme a day, so that about sixty workers, working steadily through the fifty years might accomplish the job, if nothing went into the wastepaper basket. Some more expeditious method seems desirable.

In retrospect it is amazing that Turing managed to foresee that "Advances in engineering" would lead to computers with a Gigabyte of storage by the end of the twentieth century. It is also noteworthy that Turing suggests that in terms of hardware, it is memory capacity rather than processing speed which will be critical.

However, the final sentence of the quote above indicates that Turing could already foresee that manual composition of a program which could pass the Turing test was not the most "expeditious" method, despite the fact that a dedicated group of around "sixty" programmers might complete the task within "fifty years"" if "nothing went into the wastepaper basket". Turing must already have been accutely aware, from his work with the early pilot ACE computer, that plenty goes in the waste basket in the process of debugging computer programs.

3.2 Programming approach to AI and the Machine Intelligence series

Turing's influence on the development of AI from the 1960s to the 1980s is particularly evident in the *Machine Intelligence* book series, which acted as a vanguard of cutting edge AI research during this period. The series Executive Editor, Donald Michie has already been mentioned as one of Turing's Bletchley colleagues. Michie was also the founder of Europe's first Department of Artificial Intelligence in the 1960s in Edinburgh, and later also founded the Turing Institute (an AI research institute) in the 1980s in Glasgow. Michie specifically chose topics for the Machine Intelligence workshops which were closely related to those which he and Jack Good had discussed with Turing during the war. Indeed Jack Good was a frequent contributor to the series on Turing-inspired topics such as Computer Chess [14]. To open the *Machine Intelligence 5* volume Michie selected "Intelligent machinery" [44], a previously unpublished article, in which Turing discussed the idea of designing intelligent robots which could "roam the countryside" and learn from their experience.

Turing's Version 1 Programming approach to Artificial Intelligence was the dominating paradigm for Artificial Intelligence research up until the mid-1980s. Research during this period can largely be divided into broad areas associated with 1) Reasoning, 2) Physical perception and 3) Physical action.

Reasoning Simon and Newell's General Problem Solver (GPS) [30] was an early and influential attempt to program a universal problem solver which could be applied to a variety of formal symbolic reasoning problems such as theorem proving, geometry and chess playing. It was clear that although GPS could solve simple problems, with more complex tasks, its reasoning was rapidly swamped by the combinatorics of the search. Throughout the 1960s-1980s a variety of other more specific approaches were taken to the problems of improving the efficiency of search (eg [24, 9]) and planning (eg [8, 11, 17]). Additionally a variety of more special purpose techniques were developed for both theorem proving (eg [34, 20]) and chess playing (eg [38, 14]).

During the same period, attempts to address the difficulties, foreseen by Turing, of writing effective and efficient AI programs led to the rise of a number of high-level languages. The methodologies on which these were based varied from the use of λ-calculus (eg LISP) [21] to the development of stack-based languages (eg POP1) [6] as well as languages based on first-order predicate calculus (eg Prolog) [46]. The approach of heuristic programming, developed in systems such as Dendral [4] and MYCIN [41], used constraints in the form of rules to produce systems which could reason at the level of human experts. These expert systems became a key demonstrator for the achievements of Artificial Intelligence in the early 1980s.

Physical perception The 1960s-1980s witnessed a number of early and bold attempts to write programs which could recognise three-dimensional objects within a digital image (eg [19, 3].) However, these were generally limited to analysis of simple polygons and it was unclear how they could be extended to recognise real-world objects such as trees, cars or people.

In the same period considerable advances were made in natural language generation and understanding (eg [35, 36, 37]). Early systems directly addressed one of the key assumptions of Turing's imitation game, by supporting answering of questions posed in natural language. However, just as with the initial attempts at computer vision, these natural language systems were limited by the complexity of grammars provided by their programmers.

Physical action As mentioned previously Turing [44] had discussed the idea of intelligent machines which could roam the countryside, learning for themselves. Probably the best known mobile robotics project from the early years was Stanford's Shakey project (1966-1972) [31]. By contrast, in the Edinburgh Freddy assembly robot [2, 1] the robot arm and associated digital camera remained in a fixed position while a platform containing sequentially assembled parts was directed to move past it by the computer.

4 Version 2: AI by ab initio machine learning

In his 1950s paper Turing had already anticipated the difficulties of developing AI by manually programming a digital computer. His suggested remedy was

that machines must learn in the same way as a human child.

> Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's? If this were then subjected to an appropriate course of education one would obtain the adult brain. Presumably the child brain is something like a notebook as one buys it from the stationers. Rather little mechanism, and lots of blank sheets. (Mechanism and writing are from our point of view almost synonymous.) Our hope is that there is so little mechanism in the child brain that something like it can be easily programmed. The amount of work in the education we can assume, as a first approximation, to be much the same as for the human child.

4.1 The ab initio Machine Learning movement [1980s-1990s]

During the 1970s the success of the expert systems movement (see Section 3.2) became increasingly stifled by the cost of involving experts in the development and maintenance of large rule-based systems. This problem became known as "Feigenbaum's bottleneck" [10]. However, early experiments with Meta-Dendral [4], and later Michalski's Soy Bean expert system [23], showed that rules could be automatically learned by machines from observations. Moreover, Michalski demonstrated that not only was this a more efficient method of building and maintaining expert systems, but it could also result in rules which were more accurate than existing human experts. This resulted in the start of a new series of workshops called *Machine Learning* [22] led by Ryszard Michalski, Jaime Carbonell and Tom Mitchell. The workshops, which later developed into the International Conference on Machine Learning, were originally based on the format of Donald Michie's *Machine Intelligence* workshops.

4.2 The limits of positive and negative examples

A common feature of systems developed within the standard Machine Learning framework is that, in Turing's words, learning is conducted *ab initio* (Turing's phrase is from "blank sheets") using a set of vectors associated with positive and negative classifications. Turing provides a mathematically-inspired warning about such an approach.

> The use of punishments and rewards can at best be a part of the teaching process. Roughly speaking, if the teacher has no other means of communicating to the pupil, the amount of information which can reach him does not exceed the total number of rewards and punishments applied. By the time a child has learnt to repeat "Casabianca" he would probably feel very sore indeed, if the text could only be discovered by a "Twenty Questions" technique, every "NO" taking the form of a blow.

Turing's knowledge of information theory [39] had led him to anticipate some of the limitations later uncovered in the 1980s by Valiant's theory of the learnable [45]. That is, effective *ab initio* machine learning is necessarily confined to the construction of relatively small chunks of knowledge. However, Valiant also demonstrated that the expected accuracy of the learned knowledge can be arbitrarily high given sufficient examples. So, unfortunately we have to return to Turing's original question of how to programme the 10^{12} bits of memory required to achieve human-level intelligence.

5 Version 3: AI using logic, probabilities, learning and background knowledge

Turing's answer to the problems which beset *ab initio* machine learning follows immediately on from the quote given in the previous Section.

> It is necessary therefore to have some other "unemotional" channels of communication. If these are available it is possible to teach a machine by punishments and rewards to obey orders given in some language, e.g., a symbolic language. These orders are to be transmitted through the "unemotional" channels. The use of this language will diminish greatly the number of punishments and rewards required.

Turing's claim is that by employing an "unemotional" symbolic language it should be possible to reduce the number of examples required for learning.

5.1 Logic-based learning with background knowledge

The obvious question is the appropriate form and function of the symbolic language to be employed. Again Turing's suggestions follow immediately on from the last quote.

> Opinions may vary as to the complexity which is suitable in the child machine. One might try to make it as simple as possible consistent with the general principles. Alternatively one might have a complete system of logical inference "built in". In the latter case the store would be largely occupied with definitions and propositions.

Alan Robinson's introduction [34] of *resolution*-based automatic theorem proving in 1965 led to an explosion of interest in the use of first-order predicate calculus as a representation for reasoning within AI systems. In line with Turing's idea of using "built-in" logical definitions, Gordon Plotkin's thesis [32] used resolution theorem proving as the context for investigating a form of machine learning which involves hypothesising logical axioms from observations and background knowledge. Within the era of *Logic Programming* [18] in the 1980s, these early investigations by Plotkin were taken up again by Shapiro [40] in the context of using inductive inference for automatically revising Prolog programs. However, it was not until the 1990s that the school of *Inductive*

Logic Programming [25, 26, 28] started to investigate this approach in depth as a highly expressive Machine Learning paradigm. A recent survey of the field [29] points to the maturity of theory, implementation and applications in this area.

5.2 Uncertainty and probabilistic learning

Turing makes some interesting observations concerning the uncertainty of learned rules.

> Processes that are learnt do not produce a hundred per cent certainty of result; if they did they could not be unlearnt.

Over the last decade there has been increasing interest in including probablities into *Inductive Logic Programming* [33, 12]. These probability values are used to give an indication of the uncertainty of learned rules. Turing also makes the following point concerning the ephemeral nature of learning.

> The idea of a learning machine may appear paradoxical to some readers. How can the rules of operation of the machine change? They should describe completely how the machine will react whatever its history might be, whatever changes it might undergo. The rules are thus quite time-invariant. This is quite true. The explanation of the paradox is that the rules which get changed in the learning process are of a rather less pretentious kind, claiming only an ephemeral validity.

It is in the nature of a Universal Turing machine that it acts as a meta-logical interpreter. It is this property which allows rules to be treated as data, allowing them to be altered and updated. A recent paper [27] by the author demonstrates that the meta-interpretive nature of the Prolog Logic Programming language can be used to efficiently support the introduction of auxilliary 'invented'' predicates and recursion within the context of learning complex grammars.

6 The challenge of "super-criticality"

The previous sections indicate that many of the issues which Turing discusses in the last section of the paper have since been explored in the AI literature. However, one of the Machine Learning challenges which Turing mentions is still entirely open.

> Another simile would be an atomic pile of less than critical size: an injected idea is to correspond to a neutron entering the pile from without. Each such neutron will cause a certain disturbance which eventually dies away. If, however, the size of the pile is sufficiently increased, the disturbance caused by such an incoming neutron will very likely go on and on increasing until the whole pile is destroyed.

> Is there a corresponding phenomenon for minds, and is there one for machines? There does seem to be one for the human mind. The majority of them seem to be "subcritical," i.e., to correspond in this analogy to piles of subcritical size. An idea presented to such a mind will on average give rise to less than one idea in reply. A smallish proportion are supercritical. An idea presented to such a mind may give rise to a whole "theory" consisting of secondary, tertiary and more remote ideas. Animals' minds seem to be very definitely subcritical. Adhering to this analogy we ask, "Can a machine be made to be supercritical?"

Turing's challenge to make a machine which is "super-critical" seems to only makes sense in the context of an extreme setting of the Version 3 approach (see Section 5) to Artificial Intelligence. The situation in which a new observation "leads to a theory consisting of secondary, tertiary and more remote ideas" requires both an alert mind, but also one which is abundantly stocked with relevant background knowledge. Providing such abundant background knowledge to a machine is challenging, though the advent of the World-Wide-Web offers an obvious source, as long as the available information can be accessed for purposes of inductive reasoning.

7 Conclusion

Turing closes the *Mind* paper with the following statement.

> We can only see a short distance ahead, but we can see plenty there that needs to be done.

As the present article indicates, Turing's vision was far from myopic. Indeed he foresaw many of the key issues which dominated Artificial Intelligence research over the last fifty years. However, it could still be argued that there has been no convincing demonstration of a computer passing the Turing test to date. Modern computers are typically not well-equipped with deep natural language facilities capable of playing the kind of parlour game which Turing describes. On the other hand, when most people these days are faced with an arcane (or even simple) question which they cannot immediately solve they turn to the closest computer or smart phone to find an answer. The implicit assumption is that the collective power of the World Web Web provides a greater degree of intelligence than that provided by asking the same question of whichever person is closest to hand. Computers instantly search through voluminous encylopedias, find objects in images, learn patterns of user behaviour and provide reasonable translations of text in foriegn languages. Many of the techniques used in these tasks grew out of the research carried out by Artificial Intelligence laboratories. We have Turing to thank not only for the concept of the Universal Turing machine, which gave rise to the computer industry, but also his visions of intelligent machines, which inspired the development of much of the software

behind the digital assistants which we find around us everywhere in the modern world.

Acknowledgements

The author would like to thank Donald Michie and other colleagues for their inspiring discussions on Alan Turing's views on Machine Intelligence and Machine Learning. The author would also like to thank the Royal Academy of Engineering for funding his present 5 year Research Chair.

References

[1] A.P. Ambler, H.G. Barrow, C.M. Brown, R.M. Burstall, and R. J. Popplestone. A versatile system for computer controlled assembly. *Artificial Intelligence*, 6(2):129–156, 1975.

[2] H.G. Barrow and S.H. Salter. Design of low-cost equipment for cognitive robot research. In B. Meltzer and D. Michie, editors, *Machine Intelligence 5*, pages 555–566. Edinburgh University Press, 1969.

[3] H.G. Barrow and J.M. Tenenbaum. Interpreting line drawings as three-dimensional surfaces. *Artificial Intelligence*, 17:75–116, 1981.

[4] B. Buchanan, E. Feigenbaum, and N. Sridharan. Heuristic theory formation: data interpretation and rule formation. In B. Meltzer and D. Michie, editors, *Machine intelligence 7*, pages 267–290. Edinburgh University Press, 1972.

[5] A. Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58:345–363, 1936.

[6] E. Dale and D. Michie. Pop-1: an on-line language. 1968.

[7] D. Davies. The Bombe a remarkable logic machine. *Cryptologia*, 23(2):108–138, 1999.

[8] J.E. Doran. Planning and robots. In B. Meltzer and D. Michie, editors, *Machine Intelligence 5*, pages 519–532. Edinburgh University Press, 1969.

[9] E.W. Elcock and D. Michie. Achieving several goals simultaneously. In E.W. Elcock and D. Michie, editors, *Machine Intelligence 8*, pages 94–136. Ellis Horwood, Edinburgh, 1977.

[10] E.A. Feigenbaum. Themes and case studies of knowledge engineering. In D. Michie, editor, *Expert Systems in the Micro-electronic Age*, pages 3–25. Edinburgh University Press, Edinburgh, 1979.

[11] R. Fikes and N. Nilsson. Strips: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[12] L Getoor and B. Taskar, editors. *Introduction to Statistical Relational Learning*. MIT Press, Cambridge, Massachusetts, 2007.

[13] K. Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter System I. *Monats. Math. Phys.*, 32:173–198, 1931.

[14] I.J. Good. Analysis of the machine chess game. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 267–270. Edinburgh University Press, 1969.

[15] J. Good, D. Michie, and T. Geoffrey. General report on Tunny: with emphasis on statistical methods. Bletchley Park Report HW 25/4, HW 25/5, UK Public Records Office, London, 1945.

[16] A. Hodges. *The enigma of intelligence*. Unwin Paperbacks, Hemel Hempstead, 1985.

[17] K. Konolige. A first-order formalisation of knowledge and action for a multi-agent planning system. In J.E. Hayes, D. Michie, and Y-H Pao, editors, *Machine Intelligence 10*, pages 41–72. Ellis Horwood, Chichester, UK, 1982.

[18] R.A. Kowalski. *Logic for Problem Solving*. North Holland, 1980.

[19] L.G.Roberts. *Machine Perception of Three-Dimensional Solids*. Garland Publishing, New York, 1963.

[20] D. Luckham. The resolution principle in theorem-proving. In N.L. Collins and D. Michie, editors, *Machine Intelligence 1*, pages 47–61. Oliver and Boyd, Edinburgh, 1967.

[21] J. McCarthy. Recursive functions of symbolic expressions and their computation by machine, part i. *CACM*, 3(4):184–195, 1960.

[22] R.S. Michalski. A theory and methodology of inductive learning. In R. Michalski, J. Carbonnel, and T. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 83–134. Tioga, Palo Alto, CA, 1983.

[23] R.S Michalski and R.L. Chilausky. Learning by being told and learning from examples: an experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis. *International Journal of Policy Analysis and Information Systems*, 4(2):125–161, 1980.

[24] D. Michie. Strategy-building with the graph traverser. In N.L. Collins and D. Michie, editors, *Machine Intelligence 1*, pages 135–152. Oliver and Boyd, Edinburgh, 1967.

[25] S.H. Muggleton. Inductive Logic Programming. *New Generation Computing*, 8(4):295–318, 1991.

[26] S.H. Muggleton, editor. *Inductive Logic Programming*. Academic Press, 1992.

[27] S.H. Muggleton, D. Lin, D. Pahlavi, and A. Tamaddoni-Nezhad. Meta-interpretive learning: application to grammatical inference. In *Proceedings of the 22nd International Conference on Inductive Logic Programming*, 2012. To appear.

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

[29] S.H. Muggleton, L. De Raedt, D. Poole, I. Bratko, P. Flach, and K. Inoue. ILP turns 20: biography and future challenges. *Machine Learning*, 86(1):3–23, 2011.

[30] A. Newell, J.C Shaw, and H.A. Simon. Report on a general problem-solving program. In *Proceedings of the International Conference on Information Processing*, pages 256–264, 1959.

[31] N.J. Nilsson. Shakey the robot. Technical Note 323, SRI International, 1984.

[32] G.D. Plotkin. *Automatic Methods of Inductive Inference*. PhD thesis, Edinburgh University, August 1971.

[33] L. De Raedt, P. Frasconi, K. Kersting, and S.H. Muggleton, editors. *Probabilistic Inductive Logic Programming*. Springer-Verlag, Berlin, 2008. LNAI 4911.

[34] J.A. Robinson. A machine-oriented logic based on the resolution principle. *JACM*, 12(1):23–41, January 1965.

[35] Isard S. and Longuet-Higgins H.C. Question-answering in english. In B. Meltzer and D. Michie, editors, *Machine Intelligence 6*, pages 243–254. Edinburgh University Press, 1971.

[36] E. Sandewall. Representing natural language information in predicate calculus. In B. Meltzer and D. Michie, editors, *Machine Intelligence 6*, pages 255–280. Edinburgh University Press, 1971.

[37] R.C. Schank and G. Dejong. Purposive understanding. In J.E. Hayes, D. Michie, and L.I. Mikulich, editors, *Machine Intelligence 9*, pages 459–465. Ellis Horwood, 1979.

[38] J.J. Scott. A chess-playing program. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 255–266. Edinburgh University Press, Edinburgh, 1969.

[39] C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.

[40] E.Y. Shapiro. *Algorithmic program debugging*. MIT Press, 1983.

[41] E.H. Shortliffe and B. Buchanan. A model of inexact reasoning in medicine. *Mathematical Biosciences*, 23:351–379, 1975.

[42] A. Turing. Computing machinery and intelligence. *Mind*, 59(236):435–460, 1950.

[43] A.M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936.

[44] A.M. Turing. Intelligent machinery. In B. Meltzer and D. Michie, editors, *Machine Intelligence 5*, pages 3–23. Edinburgh University Press, Edinburgh, 1969. Written in September 1947 and submitted in 1948 to the National Physical Laboratory.

[45] L.G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

[46] D.H.D Warren, L.M. Pereira, and F. Pereira. Prolog - the language and its implementation compared with lisp. *ACM SIGART Bulletin*, 64:109–115, 1977.