How the *What* becomes the *How* - Edward A. Feigenbaum's Turing Award lecture from 1994

Jan Herold

Informatik (Nebenfach M.A.)

5. Semester

February 4, 2002

Abstract

In 1994 Edward A. Feigenbaum and Raj Reddy were honored by the ACM with the Turing Award - an anual prize for the most outstanding researcher(s) in computer science.

Edward Feigenbaum and Raj Reddy were given the Alan M. Turing Award for both had been leaders in defining the emerging field of applied artificial intelligence (AI) and had been demonstrating it's technological significance. In their Turing Award lectures they give an insight in their professional development and what they're expecting as outcomes of further AI-research.

In this paper I will present Feigenbaum's lecture and will extend it by taking a closer look on Alan M. Turing contribution to artificial intelligence.

Contents

1 Introduction

Artificial Intelligence, commonly abbreviated with "AI", is a combination of computer science, physiology and philosophy. It is a broad topic, which covers different fields from machine vision to expert systems, each element of that field sharing a common feature created by AI: the ability to *"think"*.

To classify machines as being able to *think* requires defining the term *intelligence*, which in human understanding is a prerequisite for thinking. Research into areas related to intelligence such as language, learning process and sensory perception have helped scientists to design and build intelligent machines. As AI has come a long way in history even before the first electronic device was in use, many mathematicians and philosophers such as George Boole[1] have contributed to its scientific foundations. AI research experienced a boom with the invention of the computer in 1943 when suddenly the technology was available, or at least it seemed, to simulate intelligent behaviour and until now it appears to be one of the most challenging fields in computer science.

2 Turing Test

One station of Feigenbaums scientific career has been the National Physical Laboritory[2], the working place of Alan Turing from 1945 until 1948. When, in 1994, referring to this time he states that still "after decades, these memories of NPL and of Turing's ghostly presence are vivid" ([Feigenbaum, 1996], 97). Having made such a lasting impression without even being there anymore, is there anything more one needs to say about Alan Turing and his relevance for computer science ? Let us at least reconsider one of his most crucial questions concerning AI: Can machines think ?

[1]Boole (1815-1864) introduced the algebra of logic (Boolean algebra) which now finds application in computer construction, switching circuits etc.

[2]Feigenbaum went there on a Fulbright scholarship in 1959

In 1950 Turing wrote an article, "Computation machinery and intelligence", where he set up an imitation game to find an answer to his question. When referring to this game nowadays, one usually speaks about the "Turing Test".

The game he introduced is played with three people, a man (A), a woman (B) and an interrogator (C). The interrogator stays in a room separated from A and B. The object of the game for the interrogator is to determine which of the two others is the man and which is the woman. He is allowed to pose questions to A and B, which, since he doesn't know who's who, are labelled with X and Y. At the end of the game the interrogator should either state that "X is A (a man) and Y is B (a woman)" or that "Y is A and X is B". The object for A is to cause C to make the wrong identification. His answers therefore might be not true, while B, whose aim is to help the interrogator, can be expected to tell the truth. So, for example, if C poses the question "Will X please tell me the length of his or her hair" A (I suppose that he has been labelled with 'X') might answer "My hair is shingled, and the longest strands are about nine inches long" and B could try to help C by convincing him of her identity. The conversation between (A,B) and (C) has to be neutralized in such a way[3], that C cannot guess the right person by voice or manner.

Turing proposes to replace the original question by asking "What will happen when a machine takes the part of A in this game ?" Turing therefore defines intelligence in terms of flawless imitation of human cognitive behaviour: A machine can be classified as intelligent if it succeeds to deceive a human into believing that it is human. This first definition of intelligence in relation with machines was an important step in AI research: as mentioned in the introduction already, it is the basis for any further development of "thinking" machines.

Turing, although no machine met the challenge at his lifetime (nor do they today), never gave up the vision that they would do sometime:

[3]For example, questions and answers can be transmitted by a neutral person

"People assume computers are just glorified calculating machines. Not so.[...] A computer is a universal machine and I have proved how it can perform any task that can be described in symbols. I would go further. It is my view a computer can perform any task that the human brain can carry out. Any task" [Whitemore, 1987]

3 What is artificial intelligence ?

Artificial intelligence research has been developed in the 50ies of the last century in the U.S.A., dominated by two main motivations [Rechenberg/Pomberger, 1999]:

- The attempt to model the human brain in order to learn how it works (cognitive science) and to
- Design Software that mimics the human thinking process and thus improves its problem-solving abilities

Let's reconsider the term *intelligence*: As for most of us intelligence means the ability to talk about subjects that cover a broad range of human experience, but not the usage of skills to solve complex problems or deep reasoning. In this understanding intelligence can best be compared to *common sense* - but still, the most important intellectual behaviour in our society is the behaviour of expertise, that is the knowledge of specialists in their domains of work.

For example, a physician is not necessarily more intelligent in general than the average, but surely exceeds the average in the field of medicine. Imagine a turing test in medicine: what would have to be done to the machine to pass this test ? It must be given a great amount of domain-specific knowledge.

Since human intelligence is best modelled as software for a physical symbol system (e.g., a digital computer), intelligent programs coexist with other software. Feigenbaum in that context introduces a "What to How"-Scale, one end of the scale representing the "What" and the other referring to the "How".

Software in general and ever more *intelligent* programs in specific are situated on the "What"-end of the Scale. They deal with peoples' expressions of their needs, desires and goals they wish the computer to accomplish for them, it can therefore be seen as the interaction part between human and computer. The "How"-end, on the opposite of the scale consists of the computer hardware, that achieves those needs. During its history, computer science has made a shift from the "How"-end to the "What"-end and today, computer science is largely a software field. Historical steps have been assembler and first compilers, later came languages such as Fortran and Cobol, which enabled the user to instruct a computer in a more understandable language (for fellow humans). With the first Fortran program, AI programs were born at the "What"-end, for it was easier to express needs to the computer.

4 Knowledge Principle

As Software comes closer to human needs, therefore steadily approaching the "What"-end, one wants to understand how this approach works. In short, its based on increasing domain specificity of software - with other words, the softwares ability to access and represent an amount of knowledge within a certain domain of our lives, be it work or leisure.

Feigenbaum together with his colleagues made up a hypothesis about the impact of domain-specific knowledge on AI. Their suspicion was that knowledge played an until then underestimated role for AI software. They set up several experiments and concluded from their result the *knowledge principle*:

> The power of AI programs to perform at high levels of competence is primarily a function of the program's knowledge of its task domain, and not of the program's reasoning processes.

Reasoning methods , which have been intensively focused on in AI research,

from there on were considered as rather "weak methods" [Newell, 1982], until connected to a knowledge base. The importance of knowledge, not reasoning, may become obvious by drawing parallels to "real live": It is a physician, who treats sick patients, and not a logician, and correspondingly common sense is *mostly* based on knowledge, not on logic. Technically, the knowledge principle can be seen as a measure to prioritize on and evaluate AI research efforts and to transform them in real-world applications of AI ([Feigenbaum, 1996], 101)

The knowledge principle also applies to the What-to-How-Spectrum: needful software consists of knowledge details and exceptions, its not about algorithms and generalized procedures. Feigenbaum reckons domain-specific software architecture to be the future for software specialists in computer science, a development that demands a change in current engineering. Since domain-specific knowledge takes the biggest share in a certain software architecture, domain-specificity requires most attention in design. Distancing from domain-specific details is to distance oneself from relevance of the software world. The key task therefore, is to design tools,

- that allow users to express and capture domain-specific details and concepts,
- that represent those details and concepts appropriately and
- that specify architectures and processes using this knowledge

Examples for successful software products are domain-specific software: Tax-programs, word-processing and presentation software.

5 Artificial intelligence and expert systems

Programs with a high performance in solving problems of intellectual difficulty were began to be called "expert systems" in the 1960s. Their performance power

was based on domain-specific knowledge of particular application fields - since 1970s expert systems make the biggest and best known part of AI applications. Feigenbaum defines an ES as being "essentially a symbolic computer model of human expertise in a specific domain of work"([Feigenbaum, 1996], 102)

Being the core part of an expert system, the knowledge base (KB) contains formal and informal knowledge of one or more experts. The knowledge bases amount of knowledge makes the difference between a novice and an expert, for both may use the same logical reasoning methods.

For example, when looking at a top-down reasoning process[4], the more elements a database consists of, the more possible solutions in might find. More generally spoken, an expert system works much like a detective solves a mystery. Using the information, and logic or rules, an expert system can solve the problem.

Beside its amount of knowledge another core feature of a knowledge base is its ability to present knowledge - the knowledge representation technology. In fact, AI and expert systems helped to pioneer important data representation techniques in the software world: objects, inheritance of properties, rule-based representations (e.g. business logic). Overall, the factors of success of an expert system are knowledge, representation of knowledge and, finally, the management of knowledge.

6 Expert Systems: Applications

Today, needs of millions of users in all domains of life are being served by expert systems, and, respectively AI. For example, "Tax Cut", a tax-saving software, contains a rule-based tax-advisor of a thousand rules, updated by only a few tax lawyers. Although expert systems make a big part of computer-software, still

[4]This is the procedural exclusion of solution possibilities by comparing the given features of an unknown element one by one with features of each of the elements in the knowledge database, finally resulting in the output of only those elements, that meet the query criteria

many niches need to be filled out by expert systems technology, for example

- diagnosis of system failures or equipment failures
- financial analysis and decision aids for transactions
- scheduling and planning of operations, manufacturing, logistics and similar problems that are subject to a large number of complex interlocking constraints (e.g. NASA SpaceShuttle refurbishing operations in off cited landmark)
- Configuration of equipment that is ordered and manufactured in a semi-custom manner (mass-customization in modern economy)

Another interesting aspect are the economical impacts of expert systems, most of them promising return of investments in months, not years. This is due to Augustine's Law or "power of expertise law" , saying that most of the high level perfomance in a company is performed only by few people, who, in addition, outperform the weakest performers by at least a factor of 10. The expert systems approach tries to make expert knowledge (top-performers) available to non-experts(mediocre-performers), causing productivity to improve rapidly.

7 Conclusion

Turings vision of finding a computational model of intelligence is not yet realized, but still remains a challenge. Since 1991, every year computer scientists meet in order to compete for the achievement of Turings goal. The Loebner Prize, with which the best-performing machine participating in a Turing Test is annually rewarded, is only just one of many legacies, that Alan Turing left for the science community researching on AI.

Unfortunately, research on AI is not very high on the agenda of the public (and the media), which is wrong:

"Modelling thought is on the same grand scale as modelling the evolution of the universe from the Big Bang, or modelling the development of live form DNA and proteins." ([Feigenbaum, 1996], 104)

References

[Feigenbaum, 1996] Edward A. Feigenbaum: *How the "What" becomes the "How"*, CACM 39/5, 1996

[Laplante, 1996] Alan M. Turing: *Computing Machinery and Intelligence*, in: Phillip Laplante: *Great Papers in Computer Science*, St.Paul, 1996, S.629-646

[Naur, 1992] Peter Naur: *Computing: A human activity*, New York, 1992

[Lane/Chisholm, 1991] N.D.Lane / M.E.Chisholm: *Information Technology*, Boston, 1991

[Rechenberg/Pomberger, 1999] Rechenberg / Pomberger: *Informatik-Handbuch*, München, Wien, 1999

[Whitemore, 1987] Whitemore, H *Breaking the code*, London, 1987

[Newell, 1982] Newell, A. *The knowledge level*, Artifical Intelligence 18, 1982, 87-127