Artificial Intelligence: From Programs to Solvers

Hector Geffner
ICREA & Universitat Pompeu Fabra
Roc Boronat 138, 55.213
08018 Barcelona, Spain
E-mail: hector.geffner@upf.edu

Artificial Intelligence is a brain child of Alan Turing and his universal programmable computer. During the 60's and 70's, AI researchers used computers for exploring intuitions about intelligence and for writing programs displaying intelligent behavior. A significant change occurred however in the 80's, as many AI researchers moved from the early AI paradigm of writing programs for ill-defined problems to writing solvers for well-defined mathematical models like Constraint Satisfaction Problems, Strips Planning, SAT, Bayesian Networks, Partially Observable Markov Decision Processes, and General Game Playing. Solvers are programs that take a compact description of a particular model instance and automatically compute its solution. Unlike the early AI programs, solvers are general as they must deal with any instance that fits the model. Many ideas have been advanced to address this crisp computational challenge from which a number of lessons can be drawn. In this paper, I revisit the problem of generality in AI, look at the way in which this 'Models and Solvers' agenda addresses the problem, and discuss the relevance of this agenda to the grand AI goal of a computational account of intelligence and human cognition.

Keywords: Models and solvers, Planning, General Intelligence, Cognitive Science

1. Programming, AI, and AI Programming

Artificial Intelligence is a brain child of Alan Turing and his universal computer [58]. Turing was not only a logician and the father of the modern computer, but also the first programmer and the first to envision the possibilities that the programmable computer opened for the study and replication of human intelligence [11]. Programming played indeed a key role in the 1956 meeting at Darmouth where AI got its name [42]. An early collection of AI papers describing programs for playing chess and checkers, for proving theorems in logic and geometry, and for many other tasks, appeared in the seminal book *Computers and Thought*. The role of programming in these early days of AI is made explicit in the preface of the original 1963 edition of the book, where the editors explain the criterion that they followed in selecting the papers:

> (We) have tried to focus on papers that report results. In this collection, the papers ... describe actual working computer programs ... Because of the limited space, we chose to avoid the more speculative ... pieces [18].

The point is reemphasized in the preface of the more recent 1995 edition:

> A critical selection criterion (for inclusion in the collection) was that the paper had to describe ... a running computer program ... All else was talk, philosophy not science ... (L)ittle has come out of the talk [19].

Of course, not everyone in AI agreed with these views and methods, in particular John McCarthy, who was putting the basis for the logical approach to AI that placed the emphasis on the semantics and expressivity of representations rather than in their use [39,41]. Yet such dissenting views were not common and cannot be found in the volume.

It can actually be argued that several of the key AI contributions in 60's, 70's, and early 80's, had to do with *programming* and the *representation of knowledge in programs*. These contributions include Lisp and Functional Programming, Prolog and Logic Programming, Rule-based Programming, Interactive Programming Environments and Lisp Machines, Frame, Scripts, and Semantic Networks, and Expert Systems Shells and Architectures [60,7,46]. To a large extent, it was common to do AI then by picking up a task and domain X, such as story understanding, humor, scientific discovery, or circuit analysis, analyzing then how the task is done by humans either by introspection or by interviewing an expert, and capturing this reasoning in a computer program [45,52]. The work was a theory about X along with a program implementing the theory which was usually tested over some examples.

2. The Problem of Generality

Many great ideas came out of this early work in AI but there was a problem: theories expressed as programs could not be proved wrong, and in particular, when a program failed on new examples like a new joke or a new story, it was natural to put the blame on the knowledge that the program was missing. This is a version of the *generality problem* discussed at length by John McCarthy [40]. This problem was dealt by AI programmers in three different ways. Some narrowed down the domain and scope of the programs so that all the relevant knowledge could be made explicit. This was the 'expert systems' approach: programs designed to display expert performance over one specific domain, leaving all other domains and any notion of general intelligence, unaddressed [31]. Some took the programs as illustrations or demonstrations of potential capabilities which were not actually delivered. The problem with this approach is the limited scientific value of such demos [14]. Finally, some decided to write down all the relevant knowledge. This was the motivation underlying projects like Cyc [37], which haven't yet helped to deliver general intelligence.

The limitations of AI programs for exhibiting some form of general intelligence, not tied to toy worlds or narrow domains led to an impasse in the 80's and to several debates and controversies. One criticism then, coming mainly from philosophers, was that AI, renamed as Good Old Fashioned AI, had been conceived as rules and rule application, while human intelligence is something completely different [30]. A second criticism, coming from engineers, was that knowledge representation and inference were getting in the way, and AI systems should instead follow nature and "just do it"; placing emphasis on how percepts are transformed into behavior, not on thought and deliberation [6]. A third criticism came from the reborn field of neural networks and its emphasis on robust and tractable forms of soft inference and learning, as opposed to rigid, symbolic, and intractable forms of logical inference [49]. In my view, all these criticisms contained more than a grain of truth about the state of mainstream AI research in the early 80's. It's clear that intelligence cannot be rules "all the way down" with rules indicating how and when the other rules should be used, and also that intelligence must be linked to action and behavior, and not just to abstract knowledge and inference. Likewise, intelligent action involves processes such as sensing and categorization that are inherently partial and noisy and cannot be handled naturally in a pure logical setting.

These debates and controversies have not died out, and indeed, expressions such as "Good Old Fashioned AI (GOFAI)" are still used sometimes not only to refer to AI research in the 60's and 70's but to mainstream AI as a whole. What I argue below, however, is that many of these debates and controversies are simply dated, as mainstream AI has changed a great deal in the last 30 years. Moreover, important lessons have been learned that those who are still interested in understanding the human mind from a computational perspective, should not ignore.

3. Models and Solvers

If we look at the main conferences, journals, and textbooks, current AI research appears as a long and fragmented list of areas including Search and Planning, SAT and Constraints, Probabilistic Reasoning and Planning, Knowledge Representation and Reasoning, Machine Learning, Natural Language Processing, Multiagent Systems, and Robotics. The work on many of these areas, however, is no longer about writing programs for ill-defined problems but about writing *solvers for perfectly well-defined mathematical models.* These models include Constraint Satisfaction Problems, Strips Planning, SAT, Bayesian Networks, Partially Observable Markov Decision Processes, General Game Playing, and Answer Set Programming among others [13,25,1,48,34,24,5]. Solvers are programs that take a convenient description of a particular model instance (a planning problem, a CSP instance, and so on) and automatically compute its solution. Unlike the early AI programs, solvers are *general* as they must deal with any problem that fits the model: any Strips planning problem, any CSP, etc. This presents a crisp computational challenge as all models are computationally intractable, with all complete algorithms running in time that is exponential in the number of problem variables in the worst case. The computational challenge is to push this exponential explosion as far as possible. For this, general domain-independent solvers must automatically find ways to exploit the structure of the given problems, so that their performance over a given domain instance can approach the performance of a domain-specific solver. A still more basic requirement is that these solvers should not break down on a problem merely due to its size as large problems are not necessarily hard.

Work in CSPs, SAT, Planning, Bayesian Networks, and to a less extent POMDPs has uncovered techniques for *scaling up by automatically exploiting the structure of problems.* While these techniques apply to specific type of mathematical models, they are general in the sense that they are not tailored to specific domains. The computational value of these techniques is assessed *experimentally* through benchmarks, and almost in all cases, by means of competitions. There are currently periodic competitions for SAT, QBF, CSPs, Planning, Bayesian Networks, MDPs, POMDPs, General Game Playing, Answer Set Programming, and so on, with reports regularly appearing in the top AI journals. In these competitions, the solvers are tested over known and unknown model instances, with solvers being ranked in terms of the number of problems solved, the quality of solutions (when non-optimal solutions are allowed), and speed. The various competitions have helped to generate hundreds of problems that are used as benchmarks, have set standards for the syntactical encoding of problems, and have facilitated the empirical evaluation of algorithms which is crucial when the challenge is scalability over models that are computationally intractable.

The focus on models and solvers that can scale up has acted as *a powerful filter on ideas and techniques*, setting up a clear distinction between the ideas that look well from those that work well. Of course, the set of benchmarks over which these ideas and techniques are evaluated is always limited and biased, yet the focus on benchmarks provides a clear bottom line for assessing progress, even if it also tends to discourage sometimes excursions that do not address this bottom line.

Some models are closely related to other models (e.g., SAT and CSPs), others can be expressed as instances of another model (e.g., Finite-horizon Planning as SAT, SAT as Planning, Probabilistic Planning as Probabilistic Inference), yet all these models address features that appear naturally in many problems of interest. Some models are more narrow and limited, like SAT, while others are almost AI-Complete such as POMDPs, that feature many of the characteristics of agents that act in the real world: goals and preferences, stochastic actions, and partial and noisy feedback from the environment.

It is important to notice that *solvers are not architectures*; i.e., ways to organize software or hardware in settings that often have the expressive power of Turing machines. Solvers deal with models that are decidable in general and hence are less expressive than TMs, yet within this restriction, they must solve the model instances automatically. This is in contrast with modern computers or TM-equivalents like rule-based systems that can solve more difficult problems but need to be programmed. In this sense, an instance that fails to be solved by a general solver points to a limitation of the *mechanism* embodied in the solver, while an instance that fails to be solved by a knowledge-based program normally points to a failure in the *knowledge-base.* Solvers thus provide a different way for addressing the generality problem in AI.

4. Lessons from Planning

There are important lessons that can be drawn from the Models and Solvers agenda and the results achieved so far in the quest for *general scalable methods*. Some of these lessons pertain to the old debates surrounding AI and its relevance to the attempts to understand the human mind from a computational perspective. I focus first on lessons learned from Planning, my research area, then on lessons learned from other models and solvers.

Planning is the model-based approach to action selection where actions for achieving goals are selected using a model of the actions and sensors. [50,25,22]. There is a variety of planning models as actions can be deterministic or not, feedback can be full, null, or partial, and in addition, the initial situation may be fully known or not. A planner is a solver that accepts arbitrary instances of a particular planning model in compact form, and automatically produces a plan or controller that drives the agent to the goal. The most basic planning model is the one that underlies *classical planning* and assumes a fully known initial state, deterministic actions, and no feedback. A plan then is an action sequence that transforms the initial state into a goal state. From a computational point of view, the classical planning problem can be mapped into a path-finding problem over an implicit directed graph whose nodes are the possible states and whose directed edges stand for the state transitions made possible by the actions. There are well-known algorithms for path-finding in graphs such as Dijkstra's, that run in time that is polynomial in the size of the graph [9], yet for planning this is not good enough as the number of nodes is *exponential* in the number of problem variables. For finding paths to the goal in such huge graphs, the search must proceed with a sense of direction. In AI, this is achieved by means of *heuristic functions*: functions

$h(s)$ that provide a quick-and-dirty estimate of the distance or cost separating a state s from the goal [47,50]. A key development in modern planning research was the realization that useful heuristics can be derived automatically from the representation of the problem in a domain-general language [43,3,2]. It does not matter what the problem is about, a *relaxed problem* is obtained directly and effectively from the problem representation, whose solution from a state s, computed in low polynomial time, yields an informed estimate $h(s)$ of the cost-to-go from s to the goal.

The most common and useful domain-independent relaxation in planning is the *delete-relaxation* that maps a planning problem $P(s)$, with initial state s, into a planning problem $P^+(s)$ that is exactly like $P(s)$ but where the actions do not 'delete' from the state the atoms that they make false. That is, the delete-relaxation is a domain-independent transformation that takes a planning problem $P(s)$ and produces another problem $P^+(s)$ where atoms are added exactly like in $P(s)$ but are never deleted. The relaxation implies, for example, that objects and agents can be in "multiple places" at the same time, as when an object or an agent moves into a new place, the atom encoding the old location is not deleted. Relaxations, however, are not aimed at providing accurate models of the world; quite the opposite, simplified and even meaningless models of the world that while not accurate yield useful heuristics [54,44,47]. The *heuristic* $h(s)$ is obtained as an approximation of the optimal cost of the delete-relaxation $P^+(s)$ that is derived from the cost of a plan that solves $P^+(s)$ not necessarily optimally [32]. The reason that an approximation is needed is because finding an optimal plan for a delete-free planning problem like $P^+(s)$ is also NP-hard. On the other hand, finding just one plan for the relaxation whether optimal or not, can be done quickly and efficiently. The property that allows for this is *decomposability:* a problem without deletes is decomposable in the sense that a plan for a joint goal G_1 and G_2 can always be obtained from a plan for G_1 and a plan for G_2.

Interestingly, the heuristics developed in domain-independent planning for achieving both generality and scalability, tie up quite closely with current themes in Cognitive Science concerning the role of emotions and gut feelings in rational behavior, and to the limitations of early AI research that missed the crucial role played by unconscious inference in everyday cognition [59,29,17]. Domain-independent heuristics are fast (low-polynomial time) and effective, as the 'fast and frugal' heuristics advocated by Gigerenzer and others [27,26], and yet, they are general too: they apply indeed to all the problems that fit the model and problems that can be cast in that form. In addition, the derivation of these heuristics sheds light on why appraisals may be opaque from a cognitive point of view, and thus not conscious. This is because the heuristic values are obtained from a relaxed model where the meaning of the symbols is different than the meaning of the symbols in the 'true' model. Last, the heuristics provide the agent with a sense of direction or 'gut feeling' that guide the action selection in the presence of many alternatives, while avoiding an infinite regress in the decision process. Indeed, emotions long held to interfere with the decision process and rationality, are now widely perceived as a requisite in contexts where it is not possible to consider all alternatives, where they appear to act as the 'invisible hand' that guides us in these mental labyrinths [36,16]. The 'rationality of the emotions' has been defended on theoretical grounds by philosophers [12,15], and on empirical grounds by neuroscientists that have studied the impairments in the decision process that result from lesions in the frontal lobes [10]. The link between emotions and automatically derived evaluation functions in planning point to their possible computational role as well. The work in planning and more generally the work in solvers, thus show that not only the twin goals of generality and scalability constitute a crisp challenge that can be addressed head on through suitable models and methods, but that the results can be a critical source of insight for understanding general intelligence in both humans and machines.

5. Other Lessons

An old discussion surrounding AI is about *symbolic vs. subsymbolic* representation and inference. Clearly, rule-based systems were symbolic, and neural networks were subsymbolic. Yet, what about more recent models like Bayesian Networks or MDPs that have been at the center of AI research for more than two decades? Are they symbolic or subsymbolic? Is the distinction actually relevant at all? Indeed, models such as Bayesian Networks or MDPs can be used "symbolically" over variables and states associated with meaningful symbols, or subsymbolically, with variables and states not correlated with meaningful symbols at all. The latter case is common when the models are learned, as when Hidden Markov Models, a special type of Bayesian Network, are learned from data

in applications such as speech recognition, or when reinforcement learning algorithms are used to learn to act by trial-and-error in MDPs with unknown parameters [57]. Indeed, Bayesian Networks have been used in Development Psychology for understanding how babies acquire and use causal relations much before that they learn to speak [28], and Reinforcement Learning algorithms have been used in Neuroscience for interpreting the activity of dopamine cells in the brain [53].

While the distinction between symbolic and subsymbolic processing does not appear to be as crucial now as in the past, there are other distinctions that are important now that were not regarded as important in the past. Indeed, there is a crucial difference between *programs* and *architectures*, on the one hand, and *models* like Bayesian Networks or MDPs on the other. The former do not have a clearly defined scope and must be programmed by hand; the latter represent an infinite but perfectly well defined class of problems which must be solved automatically.

What about other distinctions such as *probabilistic vs. logic inference*? Logic and probabilistic inference have been traditionally thought as very different forms of reasoning requiring completely different types of algorithms. This, however, is no longer as clear-cut. Some of the best probabilistic reasoning engines are currently based on ideas coming from logic, while some of the best logical inference engines are based on ideas coming from probabilistic reasoning. The two areas are no longer disjoint and there is a very active exchange of ideas and techniques among the two different fields, with interesting consequences for understanding the mechanisms required for a domain-general intelligence.

SAT is the problem of determining whether there is a truth assignment that satisfies a propositional logical formula in Conjunctive Normal Form (CNF) [1]. The problem is NP-Complete, which in practice means that the worst-case behavior of complete SAT algorithms is exponential in the number of variables. Still current SAT solvers manage to solve problems with thousands of variables and clauses, and are used widely. The way that SAT solvers achieve this is by interleaving an incremental and complete search for solutions with two types of efficient and cost-effective inference: Unit Resolution, where the resolution inference rule is applied only when one of the parent clauses is a unit clause, and Conflict-based learning, by which clauses are learned during the search by analyzing the reasons that caused a partial assignment to fail. It is important to emphasize that many other ideas are logically possible but do not scale up as well; ideas like bypassing inference entirely by generating and testing each possible assignment one by one, or bypassing search entirely by applying full resolution without the unit restriction. More interestingly, the techniques found to be successful in SAT have been applied to SAT variations, like Weighted MAX-SAT, where the task is to find the assignment that minimizes the additive cost of the clauses violated by the assignment, and Weighted Model Counting, where the task is to add up the weights of the satisfying assignments, each such weight being the product of the weights of the true literals in the assignment. Moreover, these SAT variants are intimately related to probabilistic reasoning tasks; in particular, the computation of beliefs in a Bayesian Networks can be easily reduced to a weighted model counting problem, while the computation of the most probable explanation in a Bayesian Network can be easily reduced to a Weighted Max-SAT problem. Indeed, some of the best exact BN solvers build on this formulation [51,8]. More generally, the worst-case complexity of a number of operations over SAT, CSP, and Bayesian Network tasks, can be all expressed in terms of the same treewidth parameter that measures how tree-like is the graph induced by the factors of the problem: clauses in SAT, constraints in the CSP, and each variable with its parent set in Bayesian Networks [20,48,13]. Also, belief propagation algorithms derived for tree-like Bayesian Networks have been used not only as approximation algorithms for general Bayesian Networks [61], but also as part of exact algorithms for SAT and CSPs [4].

6. Conclusions

The relevance of the early work in AI to cognitive science was based on *intuition:* programs provided a way for specifying intuitions precisely and for trying them out. The more recent work on *domain-independent solvers* is more technical and experimental, and is focused not on reproducing intuitions but on *scalability.* This may give the impression that recent work is less relevant to the original AI goals of understanding human and general intelligence from a computational perspective. This impression, however, may prove wrong for two reasons. First, intuition is not what it used to be, and it is now regarded as the tip of an iceberg whose bulk is made of massive amounts of shallow, fast, but unconscious inference mechanisms that cannot be rendered explicit [59,29,26,35]. Second,

whatever these mechanisms are, they appear to work pretty well and to scale up. This is no small feat, given that most methods, whether intuitive or not, do not. By focusing then on the study of meaningful models and the computational methods for dealing with them *effectively,* AI may prove its relevance to human cognition in ways that may go well beyond the rules, cognitive architectures, and knowledge structures of the 80's, while providing also the basis for a general form of artificial intelligence.

There are still a number of challenges. I focus briefly on open problems related to planning. One challenge there is planning in the presence of other agents that plan, often called multiagent planning. People do this naturally all the time: walking on the street, driving, etc. The first question is how plans should be then defined. This is a subtle problem and many proposals have been put forward, often building on equilibria notions from game theory, yet no models, algorithms, or implementations of domain-independent planners able to plan meaningfully and efficiently are available in such a setting. This is not surprising though given the known limitations of game theory as a descriptive theory of human behavior. It is possible indeed that multiagent domain-independent planners based on a narrow view of human rationality just cannot get off the ground, trapped in a prisioner's dilemma type of paralysis. Eventually, a working theory of multiagent planning could shed light on the computation, nature, and role of social emotions in multiagent settings, very much as single agent planning appears to shed light on the computation, nature, and role of goal appraisals and heuristics in single-agent settings. A second open problem is learning the planning models themselves by interacting with the environment. There has been considerable progress in learning model parameters and rewards as in model-based reinforcement learning, yet the harder problem is learning the states themselves. Last, solutions are often learned or inferred that work not just for one problem but for many problems, including all possible instances of a particular domain. For instance, finding a plan for an instance of blocks world is different than inferring a general strategy for solving all block world instances. This more general problem has been called generalized planning [33,55]. The solutions to such problems can often be encoded as memoryless policies that map problem features into actions. The question is how to get such features and policies in the model-based approach, and how to get them effectively, in particular in domains that admit compact solutions over the right features. An early approach that does this in the blocks world constructs a large pool of compound features from the primitive predicates in the domain using a description logic, and then looks for compact policies represented as decision lists in the resulting language using Rivest's learning algorithm over data obtained from the solution of small instances [38]. A different approach, popular in Game AI, uses a given feature space and looks for controllers represented by neural networks using a form of evolutionary search [56].

Acknowledgements

I thank Luc De Raedt and Maria Fox for the invitation to talk at the Turing Session, ECAI 2012, and for the chance to put these ideas in writing for AI Communications. I delivered previous versions of this talk at the University of Edinburgh in 2007 and at UMass in 2010 as part of their Distinguished Lecture Series. I thank Michael Fourman, Johanna Moore, Andy Barto, and Shlomo Zilberstein for the invitations and their hospitality. I've touched on some of these issues in [21,23]. My work is partially supported by grant EC-7PM-SpaceBook.

References

[1] A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

[2] B. Bonet and H. Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1–2):5–33, 2001.

[3] B. Bonet, G. Loerincs, and H. Geffner. A robust and fast action selection mechanism for planning. In *Proc. AAAI-97*, pages 714–719, 1997.

[4] A. Braunstein, M. Mezard, and R. Zecchina. Survey propagation: An algorithm for satisfiability. *Random Structures and Algorithms*, 27(2):201–226, 2005.

[5] G. Brewka, T. Eiter, and M. Truszczyński. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.

[6] R. Brooks. Intelligence without representation. *Artificial Intelligence*, Volume 47(1–2):139–159, 1991.

[7] E. Charniak, C. Riesbeck, and D. McDermott. *Artificial intelligence programming*. L. Erlbaum Associates, 1980.

[8] M. Chavira and A. Darwiche. On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172(6–7):772–799, 2007.

[9] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2009.

[10] A. Damasio. *Descartes' Error: Emotion, Reason, and the Human Brain*. Quill, 1995.

[11] M. Davis. *The universal computer: The road from Leibniz to Turing*. W. W. Norton & Company, 2000.

[12] R. De Sousa. *The rationality of emotion*. MIT Press, 1990.

[13] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[14] H. Dreyfus. *What computers can't do: A critique of artificial reason*. Harper & Row, New York, 1972.

[15] J. Elster. *Alchemies of the Mind: Rationality and the Emotions*. Cambridge Univ. Press, 1999.

[16] D. Evans. The search hypothesis of emotion. *British J. Phil. Science*, 53:497–509, 2002.

[17] J. Evans. Dual-processing accounts of reasoning, judgment, and social cognition. *Annual Review of Pschycology*, 59:255–258, 2008.

[18] E.A. Feigenbaum and J. Feldman. *Computers and thought*. McGraw-Hill, 1963.

[19] E.A. Feigenbaum and J. Feldman. *Computers and thought*. MIT Press, 1995.

[20] E. Freuder. A sufficient condition for backtrack-free search. *Journal of the ACM*, 29(1):24–32, 1982.

[21] H. Geffner. Heuristics, planning, cognition. In R. Dechter, H. Geffner, and J.Y. Halpern, editors, *Heuristics, Probability and Causality. A Tribute to Judea Pearl*. College Publications, 2010.

[22] H. Geffner. The model-based approach to autonomous behavior: A personal view. In *Proc. AAAI*, 2010.

[23] H. Geffner. Computational models of planning. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2, 2013.

[24] M. Genesereth, N. Love, and B. Pell. General game playing: Overview of the aaai competition. *AI magazine*, 26(2):62, 2005.

[25] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: theory and practice*. Morgan Kaufmann, 2004.

[26] G. Gigerenzer. *Gut feelings: The intelligence of the unconscious*. Viking Books, 2007.

[27] G. Gigerenzer and P. Todd. *Simple Heuristics that Make Us Smart*. Oxford, 1999.

[28] A. Gopnik, C. Glymour, D. Sobel, L. Schulz, T. Kushnir, and D. Danks. A theory of causal learning in children: Causal maps and Bayes nets. *Psychological Review*, 111(1):3–31, 2004.

[29] R. Hassin and J. Uleman J.Bargh. *The new unconscious*. Oxford University Press, USA, 2005.

[30] J. Haugeland. *Artificial intelligence: The very idea*. MIT press, 1993.

[31] F. Hayes-Roth, D. Waterman, and D. Lenat. *Building expert systems*. Addison-Wesley, Reading, MA, 1984.

[32] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[33] Y. Hu and G. De Giacomo. Generalized planning: Synthesizing plans that work for multiple environments. In *Proc. IJCAI*, 2011.

[34] L. Kaelbling, M. Littman, and T. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998.

[35] D. Kahneman. *Thinking, fast and slow*. Farrar, Straus and Giroux, 2011.

[36] T. Ketelaar and P. M. Todd. Framing our thoughts: Evolutionary psychology's answer to the computational mind's dilemma. In H.R. Holcomb III, editor, *Conceptual Challenges in Evolutionary Psychology*. Kluwer, 2001.

[37] D. Lenat, R. Guha, K. Pittman, D. Pratt, and M. Shepherd. Cyc: toward programs with common sense. *Communications of the ACM*, 33(8):30–49, 1990.

[38] M. Martin and H. Geffner. Learning generalized policies in planning using concept languages. In *Proc. 7th Int. Conf. on Knowledge Representation and Reasoning*, 2000.

[39] J. McCarthy. Programs with common sense. In *Proc. Symp. on Mechanization of Thought Processes*, National Physical Laboratory, Teddington, England, 1958. At http://en.wikipedia.org/wiki/Advice_taker.

[40] J. McCarthy. Generality in artificial intelligence. *Communications of the ACM*, 30(12):1030–1035, 1987.

[41] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, Vol. 4, 1969.

[42] J. McCarthy, M.L. Minsky, N. Rochester, and C.E. Shannon. A proposal for the Dartmouth Summer Research Project on Artificial Intelligence, August 31, 1955. *AI Magazine*, 27(4):12, 2006.

[43] D. McDermott. A heuristic estimator for means-ends analysis in planning. In *Proc. AIPS-96*, pages 142–149, 1996.

[44] M. Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961.

[45] M. Minsky. *Semantic information processing*. The MIT Press, 1969.

[46] P. Norvig. *Paradigms of artificial intelligence programming: case studies in Common LISP*. Morgan Kaufmann, 1992.

[47] J. Pearl. *Heuristics*. Addison Wesley, 1983.

[48] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[49] D. Rumelhart and J. McClelland, editors. *Parallel distributed processing: explorations in the microstructure of cognition. Vol. 1*. MIT Press, 1986.

[50] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2009. 3rd Edition.

[51] T. Sang, P. Beame, and H. Kautz. Performing bayesian inference by weighted model counting. In *Proc. AAAI*, pages 475–482, 2005.

[52] R. Schank and C. Riesbeck. *Inside computer understanding: Five programs plus miniatures*. L. Erlbaum Associates, 1981.

[53] W. Schultz, P. Dayan, and P.R. Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997.

[54] H. Simon. A behavioral model of rational choice. *The quarterly journal of economics*, 69(1):99–118, 1955.

[55] S. Srivastava, N. Immerman, and S. Zilberstein. A new representation and associated algorithms for generalized planning. *Artificial Intelligence*, 175(2):615–647, 2011.

[56] K. Stanley, B. Bryant, and R. Miikkulainen. Real-time neuroevolution in the nero video game. *Evolutionary Computation, IEEE Transactions on*, 9(6):653–668, 2005.

[57] R. Sutton and A. Barto. *Introduction to Reinforcement Learning*. MIT Press, 1998.

[58] A. Turing. On computable numbers, with an application to the Entscheidungsproblem. In *Proc. of the London Mathematical Society*, volume 42, pages 230–265, 1936.

[59] T. Wilson. *Strangers to ourselves*. Belknap Press, 2002.

[60] P. Winston and R. H. Brown, editors. *Artificial Intelligence: An MIT Perspective, Vols 1 & 2*. MIT Press, 1982.

[61] J. Yedidia, W. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. In G. Lakemeyer and B. Nebel, editors, *Exploring artificial intelligence in the new millennium*, pages 236–239. Morgan Kaufmann, 2003.