Foundations of Artificial Intelligence and Effective Universal Induction

Jörg Zimmermann and Armin B. Cremers

Contents

Abstract

The term Artificial Intelligence was coined in 1956. Since then, this new research area has gone through several cycles of fast progress and periods of apparent stagnation. Today, the field has broadened and deepened significantly, and developed a rich variety of theoretical approaches and frameworks on the one side, and increasingly impressive practical applications on the other side. While a thorough foundation for a general theory of cognitive agents is still missing, there is a line of development within AI research which aims at foundational justifications for the design of cognitive agents, enabling the derivation of theorems characterizing the possibilities and limitations of computational cognitive agents.

Keywords

Artificial intelligence · Machine learning · Computational cognitive agents · Universal induction · Algorithmic transparency

J. Zimmermann (✉) · A. B. Cremers
Institute of Computer Science, University of Bonn, Bonn, Germany
e-mail: jz@cs.uni-bonn.de

Introduction

In its most general form, artificial intelligence is an area of computer science which is concerned with the design and analysis of agents acting within an open, partially, or completely unknown environment. The agent and the environment are coupled by observations and actions, i.e., the agent observes the environment and executes actions which can affect the environment. Additionally, the agent has an internal state, which can serve as memory and as a resource for internal reflection. The environment, too, has a state, which in general is not directly accessible by the agent. Only by observations the agent gets indirect and partial information about the state of the environment.

In total, the agent–environment system is a coupled dynamical system, which can be described by the following two functions:

$$E : In_E \times State_E \to State_E \times Out_E,$$

$$A : In_A \times State_A \to State_A \times Out_A,$$

where E is the function defining the dynamics of the environment and A is the function defining the agent. These two functions are coupled by setting $Out_E = In_A$ and $Out_A = In_E$. Typically, the elements of the input set of the agent are called percepts, and the elements of the output set of the agent actions. The agent function is often referred to as *agent policy* (Fig. 1).

In order to define good or even optimal agent policies, it is necessary to introduce the concept of goal or reward. An agent policy is optimal if it reaches a goal with minimal resources or maximizes reward. Ans *intelligent agent* is now defined as an agent which achieves goals in a wide range of environments. This definition was extracted by Legg and Hutter from more than 70 informal definitions occurring in cognitive science and AI research (Legg & Hutter, 2007a). In Legg and Hutter (2007b) they introduce the first general, formal definition of the intelligence of a computational agent. With the Υ-functional and its successors, e.g. for the incorporation of spatio-temporal aspects, see Orseau and Ring (2012), there are finally formal definitions of the core concept of artificial intelligence. The formal definition of intelligence by Legg and Hutter is briefly discussed in section "Defining Intelligence".

Learning from Data: The Problem of Induction

The problem of induction, which can be informally described as extracting rules from examples, leads to the following question:

- What set of possible models of the data generating process should a learning agent consider?

To answer this question in its full generality, it is necessary to explore the notion of "all possible models" from a mathematical and computational point of view, and discuss the question of effective learnability in the context of

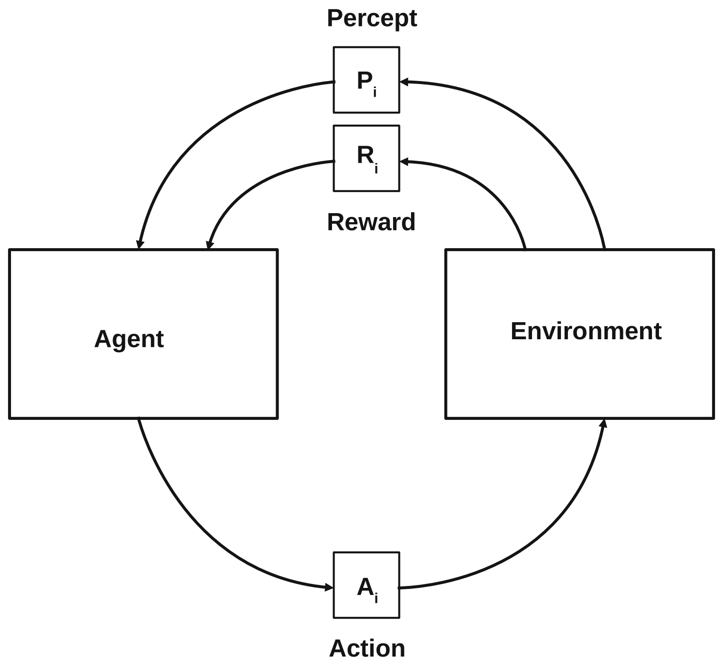

Fig. 1 Reinforcement learning agent

such generic model spaces. In Zimmermann and Cremers (2012) we showed that within the learning framework introduced by Solomonoff (1964a,b), Li and Vitányi (2008) the notion of "all possible models" cannot be defined in an absolute sense, but only with regard to a reference proof system. This dependence is used to establish a relationship between the *time* complexity of the data generating process and the *logical* complexity—defined as the proof-theoretic strength of a background axiom system—of the algorithmic learning system, thus shedding new light on the undecidability of the induction scheme introduced by Solomonoff.

The incomputability of Solomonoff induction can be traced back to the fact that the learning system does not know how much time has passed between two observations, i.e., how much time the data generating process has "invested" in order to produce the next observation. Such learning frameworks, where the generator and the learner are suspended while the other one is busy, will be called *asynchronous learning frameworks*. If one introduces a synchrony condition, which couples the time scales of the generator and the learner, one gets a *synchronous learning framework* and we will show that within such a learning framework effective and universal induction is possible, i.e., every effectively generated data sequence can be effectively learned.

Learning Frameworks

Every formal analysis of learning has to define a framework which specifies the exact type of learning problems considered and what successful learning means within this framework. The details of such a learning framework can have major implications for the question which learning tasks are solvable and which are not. In the following we will introduce two learning frameworks and we will show that these frameworks answer the same question—are universality and effectivity compatible properties?—differently.

The Asynchronous Learning Framework

A widely used model for analyzing sequential learning or decision tasks is, for example, defined in Hutter (2005), p. 126:

Definition 1 An agent is a system that interacts with an environment in cycles $k = 1, 2, 3, \ldots$. In cycle k the action (output) $y_k \in \mathcal{Y}$ of the agent is determined by a policy p that depends on the I/O history $y_1 x_1 \cdots y_{k-1} x_{k-1}$. The environment reacts to this action, and leads to a new perception (input) $x_k \in \mathcal{X}$ determined by a deterministic function q or probability distribution μ, which depends on the history $y_1 x_1 \cdots y_{k-1} x_{k-1} y_k$. Then the next cycle $k + 1$ starts.

Here $\mathcal{X}$ is a set containing all possible perceptions and $\mathcal{Y}$ is a set containing all possible actions of the agent. If the actions affect the future observations, then we call the above model an *asynchronous agent framework*, and if the actions are predictions which do not affect future observations, we call it an *asynchronous learning framework*.

In these asynchronous frameworks the resources, especially time, needed for generating the perceptions or the actions and predictions by the environment (the data generating process) or the agent are not modeled. This, for example, does imply that an agent does not know whether a new observation has arrived after 1 s or after one billion years, or, more importantly, that it has to wait longer and longer for each new observation. This last implication means that the time scales of the environment and the agent are not coupled, that, in a way, they belong to different universes. This decoupling of time scales is the reason why we call the framework asynchronous, and we will see that this property has deep implications.

Figure 2 illustrates the coupling of a learning system and an environment in the asynchronous learning framework.

The following notions are based on definitions in Zimmermann and Cremers (2012). Real-valued probabilistic learning systems are a specific type of learning system within the asynchronous learning framework:

Definition 2 A *real-valued probabilistic learning system* is a function

$$\Lambda : \{0,1\}^* \times \{0,1\} \to [0,1]_{\mathbf{R}}, \quad \text{with } \Lambda(x,0) + \Lambda(x,1) = 1 \text{ forall } x \in \{0,1\}^*.$$

A real-valued probabilistic learning system has bits as perceptions and the predictions are probabilities for the next bit. One can extend the prediction horizon of Λ by feeding it with its own predictions. This leads to a learning system $\Lambda^{(k)}$ which makes probabilistic predictions for the next k bits (xy is the concatenation of strings x and y):

$$\Lambda^{(1)} = \Lambda,$$
$$\Lambda^{(k+1)}(x, y1) = \Lambda^{(k)}(x,y) \cdot \Lambda(xy, 1), \quad x \in \{0,1\}^*, y \in \{0,1\}^k,$$
$$\Lambda^{(k+1)}(x, y0) = \Lambda^{(k)}(x,y) \cdot \Lambda(xy, 0).$$

Finally, the learnability of an infinite bit sequence s ($s_{i:j}$ is the subsequence of s starting with bit i and ending with bit j) is defined as follows:

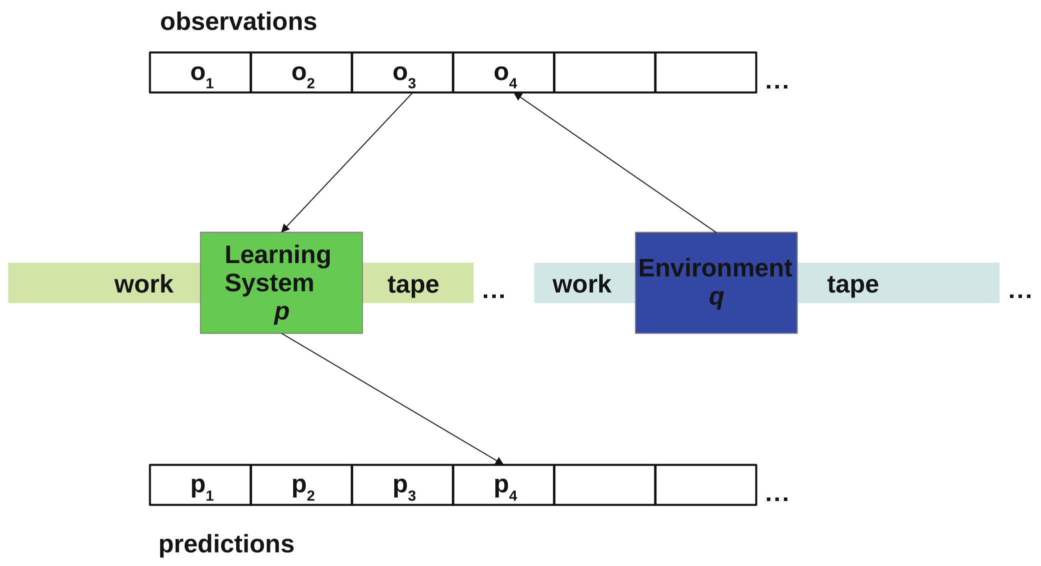

Fig. 2 Asynchronous learning framework

Definition 3 An infinite bit sequence s is learnable in the limit by the probabilistic learning system Λ, if for all $\epsilon > 0$ there is an n_0 so that for all $n \geq n_0$ and all $k \geq 1$:

$$\Lambda^{(k)}(s_{1:n}, s_{n+1:n+k}) > 1 - \epsilon.$$

This type of learnability criterion (learning in the limit) only requires that the learning system eventually will be nearly correct, but says nothing about the prediction accuracy on initial segments of the bit sequence.

Solomonoff Induction

The induction scheme introduced by Solomonoff (1964a,b) can be seen as a real-valued probabilistic learning system within an asynchronous learning framework. Solomonoff induction can learn (in the sense of Definition 3) all bit sequences generated by Turing machines. In this sense it is universal. In the following we will analyze the incomputability of Solomonoff induction and discuss why this incomputability cannot be resolved within the asynchronous learning framework.

The possible environments for Solomonoff induction can be described as programs p (represented as finite binary strings) executed by a fixed universal Turing machine U. Specifically, the universal Turing machine U has a one-way read-only input tape, some work tapes, and a one-way write-only output tape (such Turing machines are called monotone). The choice of the specific universal Turing machine affects space complexity only by a constant factor and time complexity at most by a logarithmic factor (Arora & Barak, 2009). Since the resources for generating the percepts are not modeled in an asynchronous learning framework, these effects are irrelevant and we can use any universal Turing machine as our reference machine. The program strings are chosen to be prefix-free, i.e. no program string is the prefix of another program string. This is advantageous from a coding point of view, and does not restrict universality (Li & Vitányi, 2008).

A program p is a generator of a possible world, if it outputs an infinite stream of bits when executed by U. Unfortunately, it is not decidable whether a given program p has this well-definedness property. This is the reason why Solomonoff induction is incomputable: the inference process uses the whole set of programs (program space) as possible generators, even the programs which are not well-defined in the above sense. It follows that either one restricts the model space to a decidable set of well-defined programs, which leads to an effective inference process but ignores possibly meaningful programs, or one keeps all well-defined programs, but at the price of necessarily keeping ill-defined programs as well.

The Synchronous Learning Framework

We will now introduce a learning framework where the learning system gets information about the time the data generating process has used in order to produce the next observation. This concept is inspired by an analysis of real-world sequential learning situations, where both the environment and the learning system are not suspended while the other one is busy. But first we need the notion of the *generator time function*, generator function for short, of a program p (see Zimmermann & Cremers 2012):

Definition 4 The *generator time function* $G_p^{(U)} : \mathbf{N} \to \mathbf{N} \cup \{\infty\}$ of a program p wrt. the universal reference machine U assigns every $n \in \mathbf{N}$ the number of transitions needed to generate the first n bits by the reference machine U executing p. If n_0 is the smallest number for which p does not generate a new bit, then $G_p^{(U)}(n) = \infty$ for all $n \geq n_0$.

Further we call two programs p and q *observation equivalent* if they generate the same bit sequence s. The equivalence class of all programs corresponding to an infinite bit sequence s will be denoted by $[s]$. According to the Oxford Dictionaries Online (2013), *synchrony* can be defined as:

> The state of operating or developing according to the same time scale as something else.

This is a good description of what we have in mind, so we call bit sequences having the following property *synchronous*:

Definition 5 s is *synchronous* (wrt. U) if $\limsup_{n\to\infty} \frac{G_p^{(U)}(n)}{n} < \infty$ for at least one $p \in [s]$.

As stated in section "Solomonoff Induction", the time complexity between different universal Turing machines can vary by a logarithmic factor, so we have to define the notion of synchrony relative to a fixed universal Turing machine U. A bit sequence s is called *synchronous*, if there is a universal Turing machine U so that s is synchronous wrt. U.

Synchrony entails that the time scales of the learning system and the environment are coupled, that they cannot ultimately drift apart. As long as one not assumes a malicious environment, i.e., an environment that decelerates the computing speed of the learning system more and more, synchrony seems to be a natural property. A setting where observable bit sequences can be assumed to be synchronous will be called a *synchronous learning framework*.

Effective Universal Induction

We will now show that the problem of universal induction in the synchronous learning framework is effective and discuss implications of this result. The first step is formulated by the following theorem:

Theorem 1 *All synchronous bit sequences are learnable in the limit by an effective learning system.*

Proof This can be shown straightforward by using the generator-predictor theorem proved in Zimmermann and Cremers (2012), which states that a bit sequence s is learnable in the limit by a learning system $\Lambda(\Sigma)$, if Σ (a background axiom system) proves the totality of a recursive functions which dominates the generator function of at least one program in $[s]$.

Now combining the synchrony condition wrt. a specific universal Turing machine and the fact that the time complexities of different universal Turing machines vary at most by a logarithmic factor, it suffices to find a background axiom system which proves the totality of a function which dominates $c \cdot n \cdot \log(n)$ for all positive constants c. Because the function n^2 will eventually be greater than $c \cdot n \cdot \log(n)$ for all fixed c, and the axiom system RCA_0 (Recursive Comprehension Axiom, see Zimmermann & Cremers 2012) proves the totality of n^2, the effective learning system $\Lambda(RCA_0)$ will learn all synchronous bit sequences in the limit. □

The next idea is that via a process called *clockification* an arbitrary computable bit sequence can be transformed into a synchronous one (see Fig. 3). Clockification is a process by which a learning system extends in regular time intervals (measured by its internal transitions) an observed bit sequence s by inserting "clock signals" (coding a clock signal by "00" and the original observed bits by "10" and "11") marking the passing of time. The resulting bit sequence is a synchronous one.

Theorem 2 *Within a synchronous learning framework, all effectively generated bit sequences can be effectively learned in the limit.*

Proof By combining clockification and Theorem 1 we will get the desired result. □

Caveats

The previous section has established an important result: all effective generators can eventually be effectively learned within the synchronous learning framework. This implies, for example, that if a universe can be described by a Turing machine, and we assume the assumptions of the synchronous learning framework as valid, then there is an effective learning system Λ which would converge to the "theory of everything" (TOE). This is an interesting result, but here is a list of caveats which help to put this theorem into perspective:

1. Λ converges to the TOE, but we will never know when this has happened or how close the current predictions are to the truth.
2. The true model probably is not useful, learnability and predictability fall apart, i.e., the true model could be extremely complex, its evaluation would take so long that its predictions would only arrive after the fact.
3. Even having a TOE does not mean that one can answer all questions: there are cellular automata like "Game of Life"

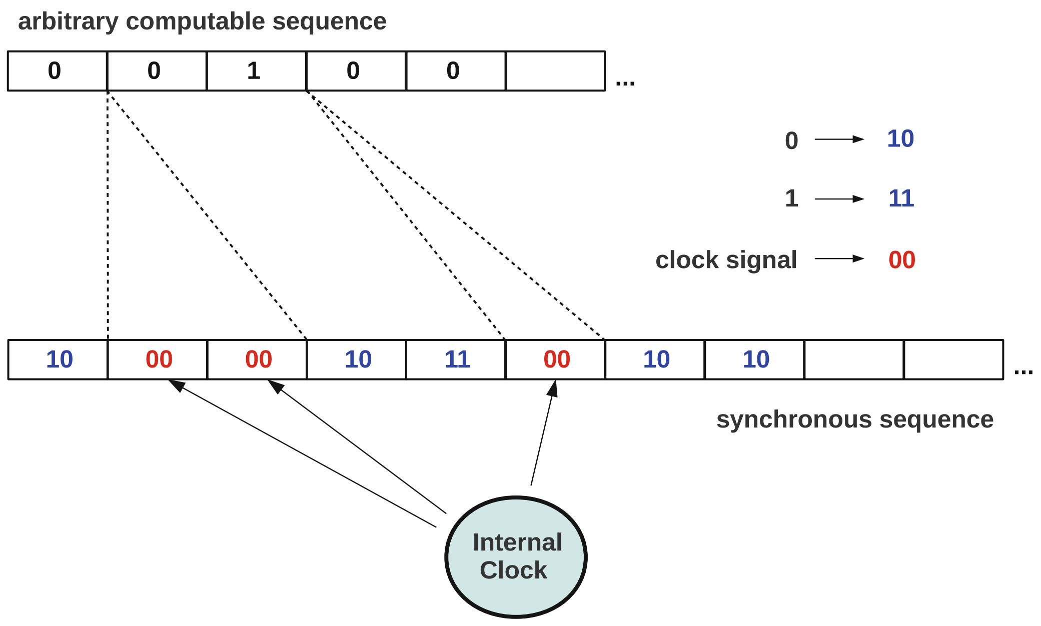

Fig. 3 Clockification: using an internal clock transforms all computable bit sequences into synchronous bit sequences

(Berlekamp, Conway, & Guy, 2001) or "Langton's Ants" (Langton, 1986) which can be seen as possible universes, and the transition rules define the TOE of these universes. But questions like "Are there self-reproducing patterns?" or "Does this ant build a highway (i.e., a certain repetitive pattern)?" cannot be answered in general, despite the fact that we know the TOE of the "Game of Life" and the ant world.

4. Finally, the information content of the universe could be infinite: imagine a Turing machine which has a work tape initialized to an infinite random bit sequence. Then the transition process is effective, but the output stream could still be incomputable by using ever more bits of the random bit sequence.

The second caveat can be termed the "postdiction problem": one can in principle predict the future exactly, but the resources needed to compute the predictions are prohibitive: they would arrive long after the predicted event has happened. This situation, where the notions of determinism and predictability fall apart, is discussed, for example, in Rummens and Cuypers (2010).

In summary, the compatibility of universality and effectiveness of inductive inference within the synchronous learning framework is an interesting theoretical finding, but has no immediate practical implications. However, it can shed some light on the path towards learning systems which are both efficient and extremely general at the same time.

The Structure of Uncertainty

One central aspect of learning from experience is the representation and processing of uncertain knowledge. In the absence of assumptions about the world, there is no nontrivial logical conclusion which can be drawn from the past on any future event. Accordingly, it is of foundational interest to analyze the structure of uncertainty as a question in its own right, and it has spawned a subfield of research within artificial intelligence and philosophy. A plethora of approaches has emerged over the last century to address this question, for example, Dempster–Shafer theory (Dempster, 1967; Shafer, 1976), Possibility theory (Dubois & Prade, 1988; Dubois, 2006), Revision theory (Gärdenfors, 1992), Ranking theory (Spohn, 1999, 2009), and non-monotonic logic (Ginsberg, 1987). A survey and discussion of many of the existing approaches is given in Huber and Schmidt-Petri (2009).

In the following we discuss an approach to reasoning under uncertainty by introducing a small axiom system describing necessary conditions for uncertainty measures. Furthermore, this axiom system does not define the structure of uncertainty explicitly, e.g. that uncertainty can be measured by one real number, but entails the algebraic structure of uncertainty values. This approach, which can be called *algebraic uncertainty theory*, enables a unifying perspective on reasoning under uncertainty. A good overview and a discussion with examples of this algebraic approach can be found in Arnborg (2016).

Formalizing Uncertainty

First we have to discuss a subtle issue of terminology. Above we have used the notion "uncertainty values" to denote generalized truth values. Unfortunately, there is the following problem when using this term in a formalized context: no uncertainty about a proposition can be identified with sure knowledge, but maximal uncertainty about a proposition is *not* certainty with regard to the negation of the proposition. The domains of truth values we want to axiomatize contain a greatest and a least element, where the greatest element should represent certainty and the least element impossibility, i.e. certainty of the negated proposition. For this reason, we adopt the notion "confidence measure" instead of uncertainty measure in the following definitions and axioms.

The Algebra of Truth Bearers

Before delving into the structure of uncertainty, we have to define the objects and their relations which are capable to take on truth values, the *truth bearers*. In a context of crisp events, i.e., after the fact it is unambiguously decidable if the event has occurred or not, the algebra of truth bearers is normally considered to be a Boolean algebra, but when truth bearers are not crisp, then another proposition algebra has to be considered, i.e., a fuzzy logic where the law of complementation is not valid: $x \vee \neg x \neq 1$, or quantum logic. The propositional algebra in quantum logic is "formally indistinguishable from the calculus of linear subspaces of a Hilbert space with respect to set products, linear sums, and orthogonal complements" corresponding to the roles of *and*, *or*, and *not* in a Boolean algebra. These linear subspaces form orthomodular lattices which in general do not satisfy the distributivity laws, see Padmanabhan and Rudeanu (2008), page 128ff. The investigation of uncertainty measures for non-Boolean proposition algebras is open to future research.

Uncertainty: The Boolean Case

A *conditional confidence measure* for a Boolean Algebra $\mathbf{U}$ and a domain of confidence values $\mathcal{C}$ is a mapping $\Gamma : \mathbf{U} \times \mathbf{U} \setminus \{\perp\} \rightarrow \mathcal{C}$. Let $A, B \in \mathbf{U}$, then the expression $\Gamma(A|B)$ reads: "the confidence value of A given B (wrt. Γ)". The domain of confidence values is partially ordered and has a greatest ($\top\!\!\top$) and a least ($\perp\!\!\perp$) element. A *confidence space* is a triple $(\mathbf{U}, \Gamma, \mathcal{C})$. One of the following axioms (Extensibility) for confidence measures deals with relations between confidence spaces defined over different Boolean algebras. Thus it is necessary to introduce a *set of confidence spaces* all sharing the same domain of confidence values. Such a set of confidence spaces we will call a *confidence universe*, and the following axiom system is concerned with such confidence universes, and not single confidence spaces. This seemingly technical shift in perspective is essential for the formalization of natural properties like extensibility, which plays a crucial role as an intuitive axiom complementing Cox's assumptions.

In Zimmermann (2012) seven axioms are introduced, which can be grouped in three connective axioms, two order axioms, and two "infrastructure axioms," where the connective axioms concern properties of the logical connectives, the order axioms relate the order structures of a proposition algebra and the confidence domain, and the infrastructure axioms deal with the combinability of confidence spaces and a closure property. Here we only state two of the seven axioms as examples; for a complete list of axioms and a discussion, see Zimmermann (2012).

Axioms for Uncertainty

In the following, we use $\Gamma(A)$ as an abbreviation for $\Gamma(A|\top)$.

(Not) For all $(\mathbf{U_1}, \Gamma_1, \mathcal{C})$ and $(\mathbf{U_2}, \Gamma_2, \mathcal{C})$:
If $\Gamma_1(A_1) = \Gamma_2(A_2)$, then $\Gamma_1(\bar{A}_1) = \Gamma_2(\bar{A}_2)$.

The axiom **Not** expresses that the information in the confidence value of a statement A is sufficient to determine the confidence value of $\bar{A}$. This is justified by the requirement that every piece of information which is relevant for the confidence value of A is relevant for the confidence value of $\bar{A}$ and vice versa.

The other two connective axioms concern similar properties for the conjunction of two propositions. The next axiom states that if a proposition A implies a proposition B (the implication relation defines an order relation on a proposition algebra), denoted by $A \leq B$, then the confidence in B is at least as high as the confidence in A.

(Order$_1$) For all $(\mathbf{U}, \Gamma, \mathcal{C})$ and all $A, B \in \mathbf{U}$: If $A \leq B$, then $\Gamma(A) \leq \Gamma(B)$.

The order axioms connect the implication ordering of the proposition algebra with the ordering on the confidence domain, where **Order**$_1$ specifies the forward direction and a second order axiom specifies the backward direction (Fig. 4).

The infrastructure axioms require the extensibility of domains of discourse, i.e., two independently defined confidence spaces shall be embeddable into one frame of reference, and a closure property of conditioning which assures that for every confidence measure conditioned on some background knowledge there is an equivalent unconditional confidence measure.

For the justification of the axioms it is important to interpret the expression $\Gamma(A|B)$ as: "*all* that can be said about the confidence of A given B (wrt. Γ)." Given this interpretation, the common justification of the connective axioms is that a violation of these axioms will necessarily lead to a loss of relevant information. Note that the axioms use only equations

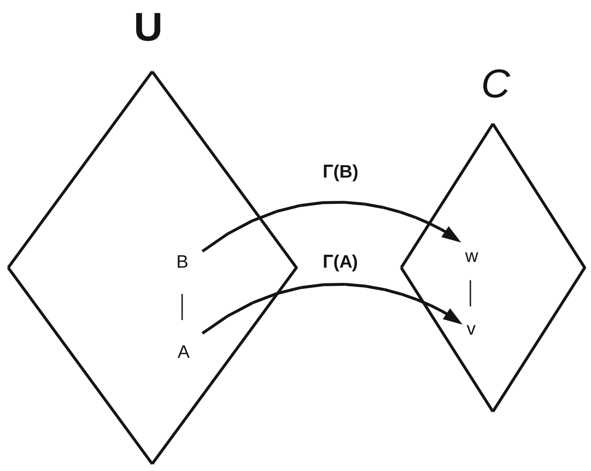

Fig. 4 Ordered confidence values v and w with corresponding propositions in a suitably chosen confidence space $(\mathbf{U}, \Gamma, \mathcal{C})$

and inequalities between confidence values, because there are no algebraic operations defined on the domain of confidence values yet.

It is now possible to characterize the algebraic structure of a confidence domain as the [0, 1]-interval of a partially ordered ring. Rings are algebraic structures which generalize fields. For example, the real numbers with addition and multiplication form a field. In a field all elements except zero have a multiplicative inverse, in a ring this is not required, i.e., a ring can contain elements other than 0 which are not invertible. Confidence measures satisfy the analogs of the axioms of probability, but with regard to the ring operations. This is stated by the following theorem:

Ring Theorem The domain of confidence values $\mathcal{C}$ of a confidence universe satisfying the connectivity, order, and infrastructure axioms can be embedded into a partially ordered commutative ring. All confidence measures Γ of the confidence universe satisfy:

$$\hat{\Gamma}(\top) = 1\,, \tag{1}$$

$$\hat{\Gamma}(A \vee B) = \hat{\Gamma}(A) \oplus \hat{\Gamma}(B)\,, \qquad \text{if } A \wedge B = \bot\,, \tag{2}$$

$$\hat{\Gamma}(A \wedge B) = \hat{\Gamma}(A|B) \odot \hat{\Gamma}(B)\,. \tag{3}$$

In the next chapter we discuss a model for a general computational agent called AIXI, which was introduced by Hutter (2005). This agent satisfies certain optimality conditions with regard to its long-term behavior within the class of computational agents. AIXI combines Solomonoff induction and reinforcement learning, which captures also interactions of an agent with the environment generating its perceptions. AIXI, like Solomonoff induction, uses the Bayesian framework for representing and processing uncertainty, which does not utilize the full generality of uncertainty calculi discussed in this chapter, like infinitesimal or incomparable uncertainty values, but Bayesian inference is a possible model of the axioms introduced in Zimmermann (2012). How uncertainty calculi using the full expressiveness of confidence domains can be combined and integrated with the AIXI agent model is open to future research.

A General Agent Architecture: AIXI

The framework of universal induction introduced by Solomonoff only treats the modeling and predicting aspect of learning, but the agent does not act based on its predictions, so in the Solomonoff framework the environment affects the learning agent, but not vice versa. In this sense, the loop between agent and environment is not closed (no sensomotoric loop). Enhancing the Solomonoff framework in order to incorporate the possibility of actions leads to a framework introduced by Hutter (2005), which can be seen as an integration of the reinforcement learning framework (Sutton, 1984) and the framework of Solomonoff. Now the agent acts based on its predictions, and these actions can affect the environment and change its future course, thus also changing future observations of the agent. In order to define the quality of an agent policy, we need generalization of the loss function used to evaluate the predictions of learning agents. Success is now defined by the environment and is the second feedback channel, besides the percepts, from the environment to the agent.

The search for optimal policies in this framework leads to a generalization of Solomonoff induction, and agents following such an optimal policy are called AIXI agents. AIXI is a reinforcement learning agent which maximizes the expected total rewards received from an environment. It simultaneously considers every computable environment as a possible generator of its perceptions. In each time step, it looks at every computable environment and evaluates how many rewards that environment generates depending on the next action taken. The expected rewards are then weighted by the subjective belief that this program constitutes the true environment. This belief is computed from the length of the program describing the environment: longer programs are considered less likely, in line with Occam's razor. AIXI then selects the action that has the highest expected total reward in the weighted sum of all these programs.

However, in Leike and Hutter (2015) it is shown, that a bad prior for inductive inference can affect the agent behavior indefinitely, because it does not sufficiently incite the agent to explorative behavior. Accordingly, no theorem comparable to the invariance theorem for Solomonoff induction is available, and the choice of the reference machine becomes crucial. Unfortunately, investigations into suitable reference machines are still in an early stage and have not yet resulted in a clear candidate for a reference machine on which to base a general cognitive agent.

Defining Intelligence

Legg and Hutter (2007b) used the combination of a general reinforcement learning agent and Solomonoff induction to define an intelligence functional Υ by assigning every agent policy π an intelligence score describing its expected reward averaged over all possible computable environments. It is an attempt to translate their informal definition of intelligence, "the ability to achieve goals in a wide range of environments," in a quantitative intelligence measure.

Let $\mathcal{X}$ be a set of perceptions, $\mathcal{R}$ be a set of rewards, and $\mathcal{Y}$ be a set of actions of an agent. A deterministic agent policy assigns to all possible sequences of percepts from $\mathcal{X}$ and rewards from $\mathcal{R}$ an action from $\mathcal{Y}$. A probabilistic policy assigns to all percept/reward sequences a probability distribution on the action set $\mathcal{Y}$. The total reward $V_\mu(\pi)$ of a policy π for an environment μ is the accumulated reward an agent executing policy π in environment μ collects during its lifetime.

Now the computable environment μ can be seen as a binary program running on a suitable universal Turing machine used as a reference machine. Solomonoff induction assumes that the prior probability of an environment μ is proportional to $2^{-|\mu|}$, where $|\mu|$ is the length of the binary program describing μ (Li & Vitányi, 2008). Thus simpler environments, meaning that there is a shorter program to describe them, get a higher prior probability. These prior probabilities are used to define the expected reward of policy π over all computable environments:

$$\Upsilon(\pi) := \sum_{\mu \in E} 2^{-|\mu|} \cdot V_\mu(\pi),$$

where E is the set of all computable environments. Legg and Hutter call $\Upsilon(\pi)$ the *universal intelligence* of an agent using policy π. The first aspect of their informal definition of intelligence, "achieving goals," is encoded in the value $V_\mu(\pi)$ of policy π with regard to each environment, the second aspect, "in a wide range of environments," is represented by averaging over all computable environments. This measure was the first formal definition of the intelligence of a general computational agent, and thus represents an important milestone in the foundations of artificial intelligence.

The Quest for a Standard Reference Machine

The results of Leike and Hutter (2015) made it abundantly clear that in order to make progress in understanding the simplicity or complexity of finite objects it is necessary to reach a consensus on a meaningful reference machine, i.e., which operations are available and executable in unit time. Such a consensus on a reference machine could serve as a standard for measuring descriptive and computational complexity. Like today's physical units, such a standard reference machine would contain contingent elements, but if it is chosen in a "natural" way it could nevertheless be tremendously useful.

Reference Machines and Initial Complexity

In order to analyze the computational complexity (or simplicity) of a computational object (algorithm, model, agent), it is necessary to define a reference machine which executes the computations. The first precisely defined mathematical model of computation, an abstract machine, was introduced by Alan Turing in 1936. There were many different attempts to define a model of computation, for example, the λ-calculus or Markov algorithms, but they were all found to equivalent to or weaker than Turing machines. This led to the formulation of the Church–Turing thesis, that all conceivable mathematical models of computation are equivalent to the Turing machine. The thesis found widespread acceptance, and today Turing machines are seen as defining an absolute notion of computability. Turing also showed that there are incomputable problems, of which the halting problem is the most famous. Another important discovery was the existence of universal Turning machines, i.e., Turning machines which are capable to simulate all other Turing machines. For a discussion of the Church–Turing thesis, universal Turing machines, and related topics, see Herken (1994).

If one is only interested whether a problem can be solved by computation or not, one can use any universal Turing machine U as a reference machine and if there is a program for U which solves the problem, then the problem is computable, otherwise not. So for questions of computability any universal Turing machine can be used and will lead to the same answers. But things become much more complicated when one is not only interested in computability, but also in complexity, i.e. the resources needed to actually execute the computations. Typically, one is interested in time and space complexity, and a central theorem relates the time and space complexity of a universal Turing machine to any Turing machine (Arora & Barak, 2009):

Theorem *There exists a TM U such that for every $x, p \in \{0, 1\}^*$, with $U(x, p) = M_p(x)$, where M_p denotes the TM represented by p.*

Furthermore, if M_p halts on input x within T steps, then $U(x, p)$ halts within $C \cdot T \cdot log(T)$ steps, where C is a number independent of $|x|$ and depending only on M_p's alphabet size, number of tapes, and number of states.

This means if one is interested only in the general growth of the time complexity with the input length, i.e., with the asymptotic behavior, a suitably chosen UTM can serve as a reference machine for analyzing the time complexity of computational problems. Current computational complexity theory tries to classify problems with regard to the asymptotic complexity, and for this goal the above specification of a reference machine is sufficient. For example, one of the most important problem classes, P, i.e., the problems solvable in polynomial time, does not change when one changes from one UTM U_1 to another UTM U_2, provided they can simulate all other TM's within polynomial time. This has led to a very successful theory of computational complexity, which can help to classify the hardness of a computational problem. The famous $P = NP$ problem is one of the major open questions of this field, and problems which can be shown to be $NP - hard$ are generally believed to have no efficient algorithms to solve them (Arora & Barak, 2009).

For questions aiming at the asymptotic growth of needed resources depending on the size of the input, this is a suitable resolution of computational complexity. But for questions regarding the computational complexity of a finite problem, like the computational complexity of a good strategy for a game like Go, or for deciding which of two binary strings has a shorter description, we need to look closer at the reference machine.

Iterated Boolean Circuits

We now introduce a proposal for a reference machine inspired by the basic functionality of current computing devices, but also by striving for mathematical simplicity. Current computing devices can be seen as the iterative application of a hardwired Boolean circuit to a vector of bits. Accordingly, an *iterated Boolean circuit* is defined as a Boolean function on B^n, the set of n-bit vectors, which then is applied iteratively, generating a sequence of bit vectors. Additionally, the Boolean circuit is build entirely of NAND-gates, i.e., the Boolean function which is false if both inputs are true and otherwise true. The NAND-gate is a Boolean base, so all Boolean functions can be expressed entirely with NAND-gates. Interestingly, a similar machine model was already defined by Alan Turing in a National Physical Laboratory Report "Intelligent Machinery" published in 1948. He called networks of binary nodes connected by NAND-gates "Unorganized Machines," and introduced them as a possible model for information processing in the human brain. This report is reproduced in Cooper and Leeuwen (2013), pp. 501–516.

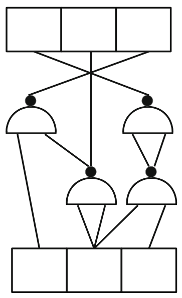
Fig. 5 A Boolean circuit consisting of 4 NAND-gates

These iterated Boolean circuits are now used to generate sequences of output bits, and for an observed bit sequence the learning problem is to find a small (measured by the number of NAND-gates) Boolean circuit which, when iterated, generates the observed bit sequence. As an example, consider the following bit sequence: 00010001000. There is a Boolean circuit with 4 NAND-gates which generates this sequence, see Fig. 5. The leftmost bit is considered the output bit. In Fig. 6 the sequence of output bits generated by the Boolean circuit after 1 and after 11 iterations is depicted. Finally, when the output sequence matches the observed sequence, we can just continue with the iterated applications of the Boolean circuit to generate predictions, see Fig. 7. In this case, the prediction for the 12th bit is "1."

The problem of finding a generating Boolean circuit matching an observed sequence can be seen as an inversion problem. Inversion problems often lead to a combinatorial search space, where no exhaustive strategy is applicable. We now discuss an approach to deal with such combinatorial search problems based on recent advances in machine learning.

Outlook: Search in Circuit Space

The number of possible circuits grows like $2^{O(n^2)}$, i.e., super-exponentially fast with the number n of gates. Even for small numbers (like 10) an exhaustive search is not possible anymore. The current advancements in combining deep learning, a variant of artificial neural networks using many hidden layers, with reinforcement learning can lead the way how to explore huge combinatorial search spaces with limited resources (Silver et al., 2018). In March 2016 a Go program based on deep learning and a self-play loop won against one of the best professional Go-players. This progress of Computer Go was not expected within the next decade, which

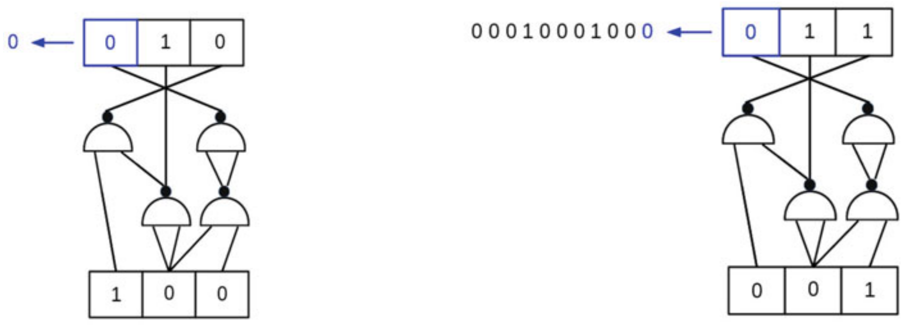

Fig. 6 Left: output sequence after one iteration. Right: output sequence after 11 iterations is matching the observed sequence

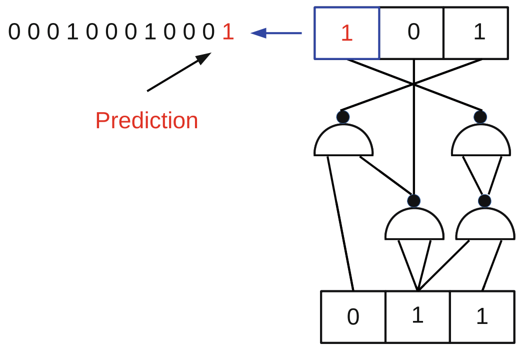

Fig. 7 The twelfth bit is the prediction generated by the iterated Boolean circuit

is a reason to hope that the basic principles of AlphaGo, and its subsequent generalization AlphaZero, can be applied to other combinatorial search problems as well. The core idea is to use deep reinforcement learning to focus the exploration of combinatorial search spaces on the most promising parts (Fig. 8).

By introducing operators on circuit space (like adding a gate, removing a gate, rewire a connection,…) the inversion problem can be transformed into a reachability problem for graphs and will thus be accessible to AlphaZero-like search strategies (Fig. 9).

Conclusions and Outlook

Despite foundational results on learnability within the synchronous and asynchronous learning frameworks, an axiomatization of uncertain reasoning, a formal definition of intelligence, and many results on general reinforcement learning agents, there is still no unifying axiomatization of general cognitive agents comparable, for example, to the axiomatic foundations of probability theory or set theory. Especially the topics of a standard reference machine and cognition with bounded resources have to be explored much further in order to reach a meaningful and integrated foundational framework for artificial intelligence.

Nevertheless, the theoretical and practical advance of artificial intelligence has reached a state where ethical questions and the impact on society become pressing issues. In the following outlook we will discuss the emerging landscape of ethical and social questions arising from the expansion of AI systems to increasingly critical applications.

Algorithmic Accountability, Transparency, and Fairness

The increase of computational resources and available data on the one side, and the latest advancements in machine learning, notably deep learning, on the other side have now reached a critical level where AI systems start to leave highly specialized and controlled environments and become part—now or in the foreseeable future—of our daily lives, on an individual and a societal level. Examples are autonomous driving, natural language processing, and applications in the judicial system. The prospect of general AI systems which are not limited to narrow applications has led to growing concerns about safety and trustworthiness. See Everitt, Lea, and Hutter (2018) for a comprehensive review of current literature.

The potential impact of AI applications on individuals and society as a whole leads to an increased need for transparency and accountability of AI systems which keeps pace with the technical development. For example, autonomous driving can lead to moral dilemmas when during an accident the loss of human life becomes unavoidable, but the autonomous driving system still can influence whose life will be endangered (Awad et al., 2018). Natural language processing can be used to facilitate fraud or to wield political influence, e.g. via bots in social networks (Simonite, 2019). One especially controversial decision support system already used by the US judicial system is COMPAS, a system which assesses the likelihood of a defendant becoming a recidivist. These risk assessments can inform decisions about who will be set free and who is not. Even if the race of the defendant is not part of the variables considered by COMPAS, reports

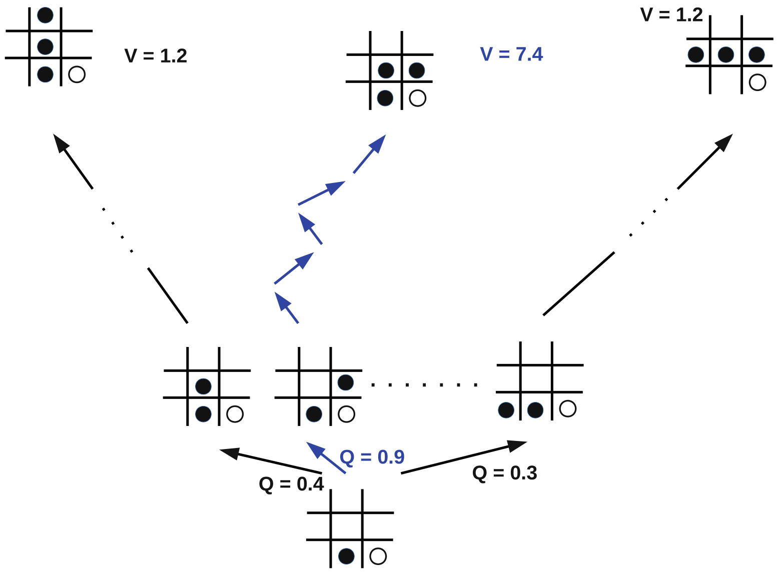

Fig. 8 The search strategy of AlphaGo

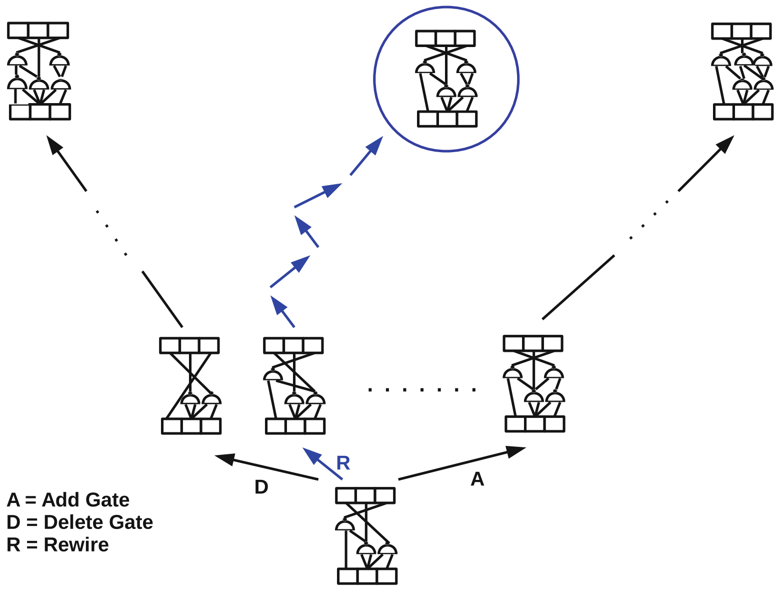

Fig. 9 The exploration of Boolean circuits using an AlphaZero-like search strategy

have emerged that COMPAS risk levels are racially biased (Angwin, Larson, Mattu, & Kirchner, 2016). A closer look shows that the exact definition of unbiasedness or fairness is instrumental, and different definitions can lead to different outcomes (Corbett-Davies, Pierson, Feller, & Goel, 2016). In this case, no decision system can be simultaneously unbiased or fair with regard to all desirable definitions of unbiasedness or fairness, and only an emerging consensus on which definition is the "right" or the "least problematic" one can mitigate this dilemma.

From Association Learning to Causal Learning

The need for algorithmic transparency, accountability, and unbiasedness adds new urgency to a topic which has affected machine learning and statistics from the beginning: the learned relationships are in general only association relations and not causal relations, i.e., the observed covariation between two variables A and B is caused by an unknown third variable C. When actions based on predictions significantly feed back into the observed system, association learning cannot answer important questions arising with regard to the consequences of the executed actions. In order to develop and apply standards of transparency, accountability, and unbiasedness, the result of learning has to identify the *causal* factors that determine the predictions. The notion of causality and the detection of causal relationships is a longstanding problem in machine learning and statistics, but recently there has been some progress, most notably the theory of causal inference by Pearl, Glymour, and Jewell (2016), but also attribution science (Otto, 2017) and causal deconvolution (Zenil, Kiani, Zea, & Tegnér, 2019) are interesting developments.

Attribution science, or probabilistic event attribution (PEA), is an emerging field that assigns probabilities to possible causes for observed effects, especially in the context of climate change, but is still in an early stage and the validation of its claims is subject to further research.

We are convinced that effective universal induction can play an important role in causal learning by identifying generators of observed data and not only associations within the observed data. The importance of universal induction was emphasized by one of the founding figures of artificial intelligence, Marvin Minsky, during a discussion panel in 2010:

"It seems to me that the most important discovery since Gödel was the discovery by Chaitin, Solomonoff, and Kolmogorov of the concept called Algorithmic Probability which is a fundamental new theory of how to make predictions given a collection of experiences and this is a beautiful theory, everybody should learn it, but it has got one problem, that is, that you cannot actually calculate what this theory predicts because it is too hard, it requires an infinite amount of work. However, it should be possible to make practical approximations to the Chaitin, Kolmogorov, Solomonoff theory that would make better predictions than anything we have today. Everybody should learn all about that and spend the rest of their lives working on it."

References

Angwin, J., Larson, J., Mattu, S., & Kirchner, L. (2016). Machine bias. *ProPublica*.

Arnborg, S. (2016). Robust Bayesian analysis in partially ordered plausibility calculi. *International Journal of Approximate Reasoning, 78*, 1–14.

Arora, S., & Barak, B. (2009). *Complexity theory: A modern approach*. Cambridge: Cambridge University Press.

Awad, E., Dsouza, S., Kim, R., Schulz, J., Henrich, J., Shariff, A., et al. (2018). The moral machine experiment. *Nature, 536*, 59–64.

Berlekamp, E. R., Conway, J. H., & Guy, R. K. (2001). *Winning ways for your mathematical plays* (2nd ed.). Natick: A K Peters.

Cooper, S. B., & van Leeuwen, J. (2013). *Alan Turing: His work and impact*. Amsterdam: Elsevier Science.

Corbett-Davies, S., Pierson, E., Feller, A., & Goel, S. (2016). A computer program used for bail and sentencing decisions was labeled biased against blacks. It's actually not that clear. *The Washington Post*.

Dempster, A. P. (1967). Upper and lower probabilities induced by a multivalued mapping. *The Annals of Mathematical Statistics, 38*(2), 325–339.

Dubois, D. (2006). Possibility theory and statistical reasoning. *Computational Statistics & Data Analysis, 51*(1), 47–69.

Dubois, D., & Prade, H. (1988). *Possibility theory*. New York: Plenum Press.

Herken, R. (Ed.). (1994). *The universal Turing machine: A half-century survey*. Berlin: Springer.

Everitt, T., Lea, G., & Hutter, M. (2018). AGI safety literature review. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18), Stockholm, Sweden* (pp. 5441–5449).

Gärdenfors, P. (Ed.). (1992). *Belief revision*. Cambridge: Cambridge University Press.

Ginsberg, M. (Ed.). (1987). *Readings in nonmonotonic reasoning*. Los Altos, CA: Morgan Kauffman.

Huber, F., & Schmidt-Petri, C. (Ed.) (2009). *Degrees of belief*. Berlin: Springer.

Hutter, M. (2005). *Universal artificial intelligence: Sequential decisions based on algorithmic probability*. Berlin: Springer.

Langton, C. G. (1986). Studying artificial life with cellular automata. *Physica D: Nonlinear Phenomena, 22*(1–3), 120–149.

Legg, S., & Hutter, M. (2007a). A collection of definitions of intelligence. In B. Goertzel & P. Wang (Eds.), *Advances in artificial general intelligence: Concepts, architectures and algorithms*. Frontiers in Artificial Intelligence and Applications (Vol. 157, pp. 17–24). Amsterdam, NL: IOS Press.

Legg, S., & Hutter, M. (2007b). Universal intelligence: A definition of machine intelligence. *Minds & Machines, 17*(4), 391–444.

Leike, J., Hutter, M. (2015). Bad universal priors and notions of optimality. *Journal of Machine Learning Research, 40*, 1244–1259.

Li, M., & Vitányi, P. M. B. (2008). *An introduction to Kolmogorov complexity and its applications*. Graduate texts in computer science (3rd ed.). Berlin: Springer.

Orseau, L., & Ring, M. (2012). Space-time embedded intelligence. In *Proceedings of the 5th International Conference on Artificial General Intelligence, AGI'12* (pp. 209–218). Berlin: Springer.

Otto, F. E. L. (2017). Attribution of weather and climate events. *Annual Review of Environment and Resources, 42*, 627–646.

Oxford Dictionaries Online. (2013). *Synchrony, noun*. Oxford: Oxford University Press. Retrieved May 11, 2013 from http://oxforddictionaries.com/definition/english/synchrony

Padmanabhan, R., & Rudeanu, S. (2008). *Axioms for Lattices and Boolean algebras*. Singapore: World Scientific.

Pearl, J., Glymour, M., & Jewell, N. P. (2016). *Causal inference in statistics: A primer*. Hoboken: Wiley.

Rummens, S., & Cuypers, S. E. (2010). Determinism and the paradox of predictability. *Erkenntnis, 72*(2), 233–249.

Shafer, G. (1976). *Mathematical theory of evidence*. Princeton: Princeton University Press.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science, 362*(6419), 1140–1144.

Simonite, T. (2019). The AI text generator that's too dangerous to make public. *Wired*.

Solomonoff, R. (1964a). A formal theory of inductive inference, part I. *Information and Control, 7*(1), 1–22.

Solomonoff, R. (1964b). A formal theory of inductive inference, part II. *Information and Control, 7*(2), 224–254.

Spohn, W. (1999). Ranking functions, AGM style. In B. Hansson, S. Halldén, N.-E. Sahlin, & W. Rabinowicz (Eds.), *Internet festschrift for Peter Gärdenfors*, Lund. http://www.lucs.lu.se/spinning

Spohn, W. (2009). A survey of ranking theory. In F. Huber & C. Schmidt-Petri (Eds.), *Degrees of belief*. Berlin: Springer.

Sutton, R. S. (1984). *Temporal Credit Assignment in Reinforcement Learning*. PhD Thesis, University of Massachusetts, Amherst, MA.

Zenil, H., Kiani, N. A., Zea, A. A., & Tegnér, J. (2019). Causal deconvolution by algorithmic generative models. *Nature Machine Intelligence, 1*, 58–66.

Zimmermann, J. (2012). *Algebraic Uncertainty Theory*. PhD Thesis, Rheinische Friedrich-Wilhelms-Universität Bonn.

Zimmermann, J., & Cremers, A. B. (2012). Making Solomonoff induction effective or you can learn what you can bound. In S. B. Cooper, A. Dawar, and B. Löwe (Eds.), *How the world computes*. Lecture Notes in Computer Science (Vol. 7318). Berlin: Springer.