Self-Modeling Agents Evolving in Our Finite Universe

Bill Hibbard

SSEC, University of Wisconsin, Madison, WI 53706, USA
and Machine Intelligence Research Institute
test@ssec.wisc.edu

Abstract: This paper proposes that we should avoid infinite sets in definitions of AI agent and their environments. For agents that evolve to increase their finite resources it proposes a self-modeling agent definition that avoids assumptions about the agent's future form. And it proposes a consistent and complete logical theory for reasoning by AI agents in our finite universe.

1 Finitely Computable Agents

According to current physics [1] our universe has a finite information capacity of no more than 10^{120} bits (10^{90} bits excluding gravitational degrees of freedom). Modeling artificial intelligence (AI) agents and environments with infinite sets, such as Peano arithmetic and the infinite tape of a Turing machine, introduces unnecessary theoretical complexity into our understanding of AI. Thus in my recent papers [2, 3] I have replaced the universal Turing machine in Hutter's [4] universal AI with finite stochastic programs (limited to finite memory, for which the halting problem is decidable).

Specifically, at each of a discrete series of time steps $t \in \{0, 1, 2, ..., T\}$, for some large T, the agent sends an action $a_t \in A$ to the environment and receives an observation $o_t \in O$ from the environment, where A and O are finite sets. Let $h_t = (a_1, o_1, ..., a_t, o_t) \in H$ be an interaction history where H is the set of all histories for $t \leq T$.

The agent's actions are motivated by a *utility function* $u : H \rightarrow [0, 1]$ which assigns utilities between 0 and 1 to histories. Future utilities are discounted according to a *geometric temporal discount* $0 < \gamma < 1$. The agent computes a prior probability $\rho(h)$ of history h. The value $v(h)$ of a possible future history h is defined recursively by:

$$v(h) = u(h) + \gamma \max_{a \in A} v(ha) \quad (1)$$

$$v(ha) = \sum_{o \in O} \rho(o \mid ha)\, v(hao) \quad (2)$$

The recursion terminates with $v(h_t) = 0$ for $t > T$. The agent (or policy) π is defined to take, after history h_t, the action:

$$\pi(h_t) := a_{t+1} = \operatorname{argmax}_{a \in A} v(h_t a) \quad (3)$$

Given a history h_t, the agent models the environment by the program [2]:

$$q_t = \lambda(h_t) := \text{argmax}_{q \in Q} P(h_t \mid q)\, \rho(q) \quad (4)$$

Here Q is a prefix-free language for finite stochastic programs, $\rho(q) = 2^{-|q|}$ is the prior probability of program $q \in Q$ where $|q|$ is the length of q in bits, and $P(h_t \mid q)$ is the probability that q computes the history h_t (this is the probability that the stochastic program q computes the observations o_i in response to the actions a_i for $1 \le i \le t$). Then the prior probability of a possible future interaction history h for use in (2) is:

$$\rho(h) = P(h \mid q_t) \quad (5)$$

2 Self-Modeling Agents

Limited resources are essential to Wang's [5] definition of intelligence and a practical reality for agents in our universe. Although the model $\lambda(h_t)$ in (4) can be finitely computed [3], the resources necessary to compute it grow exponentially with the length of history h_t. Furthermore computing the value $v(h)$ of a possible future history h in (1) and (2) requires an expensive recursion. Hence an agent with limited resources must compute approximations. Increasing the accuracy of these approximations will improve the agent's ability to maximize its utility function, and hence the agent will choose actions to increase its computing resources and so increase accuracy.

Such self-improvement must be expressible by actions in set A. However, the agents of Section 1 cannot adequately evaluate self-improvement actions. If the agent is computing approximations to the model $\lambda(h_t)$ and to values $v(h)$ using its limited computing resources, then it cannot use those limited resources to compute and evaluate what it would compute with greater resources. In real time interactions between the agent and the environment, the environment will not wait for the agent to slowly simulate what it would compute with greater resources.

In the agents of Section 1 values $v(ha)$ are computed by future recursion in (1) and (2). Here we define a revised agent in which values $v(ha)$ are computed for initial subintervals of the current history and in which the environment model includes the computation of such values. Given $h_t = (a_1, o_1, ..., a_t, o_t)$, for $i \le t$ define:

$$ov(h_{i-1}a_i) = discrete((\sum_{i \le j \le t} \gamma^{j-i}\, u(h_j)) / (1 - \gamma^{t-i+1})) \quad (6)$$

Here $h_j = (a_1, o_1, ..., a_j, o_j)$, $discrete()$ samples real values to a finite subset R of the reals (e.g., floating point numbers) and division by $(1 - \gamma^{t-i+1})$ scales values of finite sums to values as would be computed by infinite sums. Define $o'_i = (o_i, ov(h_{i-1}a_i))$ and $h'_t = (a_1, o'_1, ..., a_t, o'_t)$. That is, values $ov(h_{i-1}a_i)$ computed from past interactions are included as observables in an expanded history h'_t so the model $\lambda(h'_t)$ includes an algorithm for computing them:

$$q_t = \lambda(h'_t) := \text{argmax}_{q \in Q} P(h'_t \mid q)\, \rho(q) \quad (7)$$

Define $\rho(h') = P(h' \mid q_t)$. Then compute values of possible next actions by:

$$v(h_t a) = \sum_{r \in R} \rho(ov(h_t a) = r \mid h'_t a)\ r \tag{8}$$

Here $h'_t = (a_1, o'_1, ..., a_t, o'_t)$ and $h_t = (a_1, o_1, ..., a_t, o_t)$. As in (3) define the agent's policy $\pi(h_t) = a_{t+1} = \text{argmax}_{a \in A}\ v(h_t a)$. Because $\lambda(h'_t)$ models the agent's value computations I call this the *self-modeling agent*. It is finitely computable. There is no look ahead in time beyond evaluation of possible next actions and so no assumption about the form of the agent in the future. $\lambda(h'_t)$ is a unified model of agent and environment, and can model how possible next actions may increase values of future histories by any modification of the agent and its embedding in the environment [6].

The game of chess provides an example of learning to model value as a function of computing resources. Ferreira [7] demonstrated an approximate functional relation between a chess program's ELO rating and its search depth, which can be used to predict the performance of an improved chess-playing agent before it is built. Similarly the self-modeling agent will learn to predict the increase of its future utility due to increases in its resources.

Utility functions defined in terms of the environment model $\lambda(h'_t)$ are a way to avoid the unintended behavior of self-delusion [8, 2]. They are also natural for complex AI agents. Rather than having preprogrammed environment models, complex AI agents must explore and learn models of their environments. But the designers of an AI agent will express their intentions for the agent's behavior choices in terms of their own knowledge of the agent's environment. Thus it is natural that they define an agent's utility function in terms of a procedure to be applied to the agent's learned environment model. I presented an example of such a procedure at AGI-12 [3]. Defining its utility function in terms of its environment model introduces a potential circularity in the self-modeling agent: $ov(h_{i-1}a_i)$ depends on $u(h_j)$ in (6), $u(h_j)$ depends on $\lambda(h'_t)$ in defining the utility function in terms of the model, and $\lambda(h'_t)$ depends on $ov(h_{i-1}a_i)$ in (7). This circularity can be avoided by defining the utility function u used in equation (6) in terms of an environment model from a previous time step.

The agent's environment model is an approximation because the model is based on a limited history of interactions with the environment and, for agents in our universe, because of limited resources for computing the model. Thus a utility function computed from the model is also an approximation to an ideal utility function which is the true expression of the intention of the agent designers. Such an approximate utility function is a possible source of AI behavior that violates its design intention.

3 Consistent and Complete Logic for Agents

Any AI agent based on a logical theory that includes Peano arithmetic (PA) faces problems of decidability, consistency and completeness. Yudkowsky and Herreshoff [9] discuss such problems related to Löb's Theorem for a sequence of evolving agents. Our universe has finite information capacity [1]. I suggest that an agent can pursue its

goal in such a finite environment without any need for PA and its theoretical problems.

An environment with finite information capacity has a finite number of possible states. In this case it is reasonable to assume a finite limit on the length of interaction histories, that the action set *A* and the observation set *O* never grow larger than the information capacity of the environment, and that probabilities and utility function values are constrained to a finite subset of the reals (only a finite subset can be expressed in a finite environment). Then, for a given finite limit on environment size, there are finite numbers of possible objects of these types: environments (expressed as Markov decision processes), histories, utility functions, policies, environment models that optimize equation (4) or (7) for possible histories, and agent programs (assuming agent memory is limited by the information capacity of the environment, there are a finite number of programs and their halting problem is decidable). There are also finite numbers of possible predicates and probability distributions over these types of objects and combinations of them. So, for a given finite limit on environment size, the theory of these types of objects, predicates and probability distributions is decidable, consistent and complete (quantifiers over finite sets can be eliminated, reducing the theory to propositional calculus). In our universe with no more than 10^{120} bits, agents can use this theory to avoid the logical problems of PA. I suggest that more serious problems for agents in our universe are the inaccuracy of their environment models and the limits on their memory capacity and speed for reasoning in real time.

References

1. Lloyd, S. Computational Capacity of the Universe. Phys.Rev.Lett. 88 (2002) 237901.
2. Hibbard, B. 2012a. Model-based utility functions. J. Artificial General Intelligence 3(1), pp. 1-24.
3. Hibbard, B. 2012b. Avoiding unintended AI behavior. In: Bach, J., and Iklé, M. (eds) AGI 2012. LNCS (LNAI), vol. 7716, pp. 107-116. Springer, Heidelberg.
4. Hutter, M. 2005. Universal artificial intelligence: sequential decisions based on algorithmic probability. Springer, Heidelberg.
5. Wang, P. 1995. Non-Axiomatic Reasoning System --- Exploring the essence of intelligence. PhD Dissertation, Indiana University Comp. Sci. Dept. and the Cog. Sci. Program.
6. Orseau, L. and Ring, M. 2012. Space-Time Embedded Intelligence. In: Bach, J., and Iklé, M. (eds) AGI 2012. LNCS (LNAI), vol. 7716, pp. 209-218. Springer, Heidelberg.
7. Ferreira, D. R. 2013. The Impact of Search Depth on Chess Playing Strength, ICGA Journal 36(2), pp. 67-80.
8. Ring, M., and Orseau, L. 2011. Delusion, survival, and intelligent agents. In: Schmidhuber, J., Thórisson, K.R., and Looks, M. (eds) AGI 2011. LNCS (LNAI), vol. 6830, pp. 11-20. Springer, Heidelberg.
9. Yudkowsky, E., and Herreshoff, M. 2013. Tiling Agents for Self-Modifying AI, and the Löbian Obstacle. http://intelligence.org/files/TilingAgents.pdf