Bridging AGI Theory and Practice with Galois Connections

Ben Goertzel[1,2](✉)

[1] OpenCog Foundation, Rockville, USA
ben@goertzel.org
[2] SingularityNET Foundation, Amsterdam, The Netherlands

Abstract. Multiple cognitive algorithms posited to play a key role in AGI (forward and backward chaining inference, clustering and concept formation, evolutionary and reinforcement learning, probabilistic programming, etc.) are given a common formulation as recursive discrete decision processes involving optimizing functions defined over metagraphs, in which the key decisions involve sampling from probability distributions over metagraphs and enacting sets of combinatory operations on selected sub-metagraphs. This forms a bridge between abstract conceptions of general intelligence founded on notions of algorithmic information and complex systems theory, and the practical design of multi-paradigm AGI systems.

1 Introduction

The pursuit of AGI has an abstract theoretical aspect, in which the focus is understanding *what intelligence is* at a fundamental level going beyond any particular biological organism or engineered system. It also has an acutely practical aspect, in which one is trying to build particular systems with specific resources and application foci, much like building any other machine (albeit with some unique aspects given that this sort of machine is expected to take over its own redesign and re-engineering process).

Connecting the theoretical and practical aspects of AGI is a major challenge, which if done well can enhance both aspects. Here we present some ideas aimed at fleshing out this connection, in the specific context of cross-paradigm metagraph-based AI approaches like the OpenCog family of systems.

Perhaps the best known approach to the abstract formulation of AGI is the algorithmic-information-theory-driven angle, as represented by AIXI [10], Godel Machine [13] and their relatives. These abstract AGI systems have the general form of "reinforcement learning" or "experiential interactive learning" algorithms, meaning that they operate via iteratively observing the world, then choosing actions that they expect will give them maximum reward based on the world's reactions, etc. They are unrealistic because their action selection is uncomputable or at best computationally intractable (though efforts have been made to scale them down [1,14,16]).

https://doi.org/10.1007/978-3-031-33469-6_12

An alternate way of conceiving AGI is Weaver's notion of "Open Ended Intelligence" [17], which considers intelligences as complex, self-organizing systems that interact with their environments in such a way as to pursue the two complementary, often conflicting goals of *individuation* (maintaining system boundaries and coherence) and *self-transcendence* (developing and acquiring new properties and aspects, including those that would be incomprehensible to earlier system versions). This approach does not make unrealistic assumptions, but possesses a fluidity that renders its rigorous application to specific cases an interesting challenge.

In recent papers such as *The General Theory of General Intelligence* [7] and *Patterns of Cognition* [6] I have sought to provide one sort of conceptual and mathematical bridge between these general-purpose AGI frameworks and practical real-world AGI-oriented systems, via looking at formulations of the AI algorithms playing key roles in the OpenCog AI system in terms of abstract recursive discrete decision systems. This paper gives a concise overview of a few of the key ideas from these longer works.

The DDSs (Discrete Decision Systems) we propose on the one hand can be straightforwardly viewed as scaled down versions of AIXI^{tl} or time-bounded Godel Machine type systems, but on the other hand can be used to drive concrete thinking about functional programming implementations of OpenCog algorithms, and understood as seeds for the self-modifying self-organization conceived in Open-Ended Intelligence theory.

The *Patterns of Cognition* analysis involves representing various cognitive algorithms as recursive discrete decision processes involving optimizing functions defined over metagraphs, in which the key decisions involve sampling from probability distributions over metagraphs and enacting sets of combinatory operations on selected sub-metagraphs. A variety of recursive decision process called a COFO (Combinatory Function Optimization) algorithm plays a key role. One can view a COFO as being vaguely like Monte-Carlo-AIXI, but within the context of a combinatory computational model – and with the added twist that the Monte Carlo sampling based estimations are augmented by estimations using other probabilistic algorithms that are themselves implemented using COFO. There are close connections to modern probabilistic programming theory [11], but with more of an emphasis on recursive inference algorithms and less reliance on simplistic sampling methods.

Behind the scenes of the COFO framework is a core insight drawn from the body of theory behind the OpenCog system – that a combinatory computational model defined over metagraphs is an especially natural setting in which to formalize various practical AGI-oriented algorithms. From a sufficiently abstract perspective, all Turing-complete computational models are equivalent, and all general-purpose computational data structures are equivalent. But from a practical AGI implementation and teaching perspective, it makes a difference which computational models and data structures one chooses; the argument for metagraphs as the core data structure for AGI has been laid out in [3] and references therein such as [2], and the argument for combinatory computing as the core approach for AGI has been laid out in [9] and earlier papers referenced therein.

2 Discrete Decision Systems

To bridge the gap between abstract AGI agent models and practical AGI systems, we introduce a basic model of a *discrete decision system (DDS)* – a process defined on n stages in which each stage $t = 1, \ldots, n$ is characterized by

– an **initial state** $s_t \in S_t$, where S_t is the set of feasible states at the beginning of stage t;
– an **action** or "decision variable" $x_t \in X_t$, where X_t is the set of feasible actions at stage t – note that X_t may be a function of the initial state s_t;
– an **immediate cost/reward function** $p_t(s_t, x_t)$, representing the cost/reward at stage t if s_t is the initial state and x_t the action selected;
– a **state transition function** $g_t(s_t, x_t)$ that leads the system towards state $s_{t+1} = g_t(s_t, x_t)$.

The mapping of the simple agents model given above into this framework is fairly direct: environments determine which actions are feasible at each point in time and goals are assumed decomposable into stepwise reward functions. Highly generally intelligent agents like AIXI^{tl} fit into this framework, but so do practical AI algorithm frameworks like greedy optimization and deterministic and stochastic dynamic programming. As we shall see, with some care and further machinery the various cognitive algorithms utilized in the OpenCog framework can be interpreted as DDSs as well.

To express greedy optimization in this framework, one begins with an initial state, chosen based on prior knowledge or via purely randomly or via appropriately biased stochastic selection. Then one chooses an action with a probability proportional to immediate cost/reward (or based on some scaled version of this probability). Then one enacts the action, the state transition, and etc.

An interesting case of "greedy" style DDS dynamics in an AGI context is the adaptive spreading of attention through a complex network. OpenCog's attentional dynamics subsystem, ECAN (Economic Attention Networks), involves spreading of two types of attention values through a knowledge metagraph – Short-Term Importance (STI) and Long-Term Importance (LTI) values, representing very roughly the amount of processor time an Atom should receive in the near term, and the criticalness of keeping an Atom in RAM in the near term. In this case: an **initial state** is a distribution of STI and LTI values across the Atoms in an Atomspace; an **action** is the spreading of some STI or LTI from one Atom to its neighbors; an **immediate cost/reward function** is the degree to which a given spreading action causes the distribution of STI/LTI values to better approximate the actual expected utilities of assignation of processor time and RAM to the Atoms in Atomspace; a **state transition function** is the updating of the overall set of STI/LTI values; and the ECAN equations in the OpenCog system embody a greedy heuristic for executing this DDS.

To express dynamic programming in this DDS framework is a little subtler, as in DP one tries to choose actions with probability proportional to overall expected cost/reward. Estimating the overall expected cost/reward of an action sequence requires either an exhaustive exploration of possibilities (i.e. full-on dynamic programming) or else some sort of heuristic sampling of possibilities (approximate stochastic dynamic programming).

To handle concurrency in this framework, one can posit underlying atomic actions $w_t \in W_t$, and then define the members of X_t as subsets of W_t. In this case each action x_t represents a set of w_t being executed concurrently.

3 Combinatory-Operation-Based Function Optimization

To frame the sorts of cognitive algorithms involved in OpenCog and related AGI architectures in terms of general DDS processes, [6] introduces the notion of COFO, Combinatory-Operation-Based Function Optimization. Basically, a COFO process wraps a combinatory computational system of the sort considered in [4] and [7] within a DDS, by using the combinatory system as the method of choosing actions in a discrete decision process oriented toward optimizing a function. The hypothesis is then made that this particular sort of DDS plays a core role in practical AGI systems operating in environments relevant to our physical universe and the everyday human world.

More specifically, we envision a cognitive system controlling an agent in an environment to be roughly describable as a DDS (the "top-level DDS"), and then envision the cognitive processing used for action selection in the DDS as comprising: 1) A memory consisting of a set of entities that combine with each other to produce other entities, i.e. a combinatory system embodied in a knowledge metagraph; 2) Cognitive processes instantiated as COFO processes, i.e. as DDSs whose goals are function optimizations and whose actions are function evaluations, all leveraging a common metagraph as background knowledge and as a dynamic store for intermediate state; 3) One or more DDSs carrying out attention allocation on the common metagraph (the core DDS here using greedy heuristics but supplemented by one or more additional DDSs using more advanced cognition), spanning the portions of the metagraph focused on by the various COFO processes.

So practical intelligent systems are modeled as multi-level DDSs where the subordinate DDSs operating within the outer-loop agent control DDS are mostly COFO processes. In [7] some effort is taken to explore how the various COFO-like processes involved in human-like cognition appear to interoperate in human cognitive architecture, and more specifically how the OpenCog Hyperon design explicitly interleaves COFO processes in its attempt to manifest advanced AGI.

A COFO process, more explicitly, involves making of a series of decisions involving how to best use a set of combinatory operators C_i to gain information about maximizing a function F (or Pareto optimizing a set of functions $\{F_i\}$) via sampling evaluations of F ($\{F_i\}$). For simplicity we'll present this process in the case of a single function F but the same constructs work for the multiobjective case. It is shown in [6] how COFO can be represented as a discrete decision process, which can then be enacted in greedy or dynamic programming style.

Given a function $F : X \to R$ (where X is any space with a probability measure on it and R is the reals), let $\mathcal{D}$ denote a "dataset" comprising finite subset of the graph $\mathcal{G}(F)$ of F, i.e. a set of pairs $(x, F(x))$. We want to introduce a measure $q_F(\mathcal{D})$ which measures how much guidance $\mathcal{D}$ gives toward the goal of finding x that make $F(x)$ large. The best measure will often be application-specific; however as shown in [6] one can also introduce general-purpose entropy-based measures that apply across domains and problems.

We can then look at greedy or dynamic programming processes aimed at gradually building a set D in a way that will maximize $q_{\rho,F}(D)$. Specifically, in a cognitive algorithmics context it is interesting to look at processes involving combinatory operations $C_i : X \times X \to X$ with the property that $P(C_i(x,y) \in M_\rho^D | x \in M_\rho^D, y \in M_\rho^D) \gg P(z \in M_\rho^D | z \in X)$. That is, given $x, y \in M_\rho^D$, combining x and y using C_i has surprisingly high probability of yielding $z \in M_\rho^D$.

Given combinatory operators of this nature, one can then approach gradually building a set D in a way that will maximize $q_{\rho,F}(D)$, via a route of successively applying combinatory operators C_i to the members of a set D_j to obtain a set D_{j+1}.

Framing this COFO process as a form of recursive Discrete Decision System (DDS), we obtain:

1. A **state** s_t is a dataset D formed from function F
2. An **action** is the formation of a new entity z by
   (a) Sampling x, y from X and C_i from the set of available combinatory operators, in a manner that is estimated likely to yield $z = C_i(x,y)$ with $z \in M_\rho^D$
      i. As a complement or alternative to directly sampling, one can perform probabilistic inference of various sorts to find promising (x, y, C_i). This probabilistic inference process itself may be represented as a COFO process, as we show below via expressing PLN forward and backward chaining in terms of COFO
   (b) Evaluating $F(z)$, and setting $D^* = D \cup (z, F(z))$.
3. The **immediate reward** is an appropriate measure of the amount of new information about making F big that was gained by the evaluation $F(z)$. The right measure may depend on the specific COFO application; one fairly generic choice would be the relative entropy $q_{\rho,F}(D^*, D)$
4. **State transition**: setting the new state $s_{t+1} = D^*$

A concurrent-processing version of this would replace 2a with a similar step in which multiple pairs (x, y) are concurrently chosen and then evaluated.

The action step in a COFO process is in essence carrying out a form of probabilistic programming [11] (which is clear from the discussion of probabilistic programming in a dependent type context given in [5]). Finding the right conglomeration of combinatory operators to produce a given output is formally equivalent to finding the right program to produce a given sort of output, and here as in probabilistic programming one is pushed to judiciously condition estimates on prior knowledge.

In the case where one pursues COFO via dynamic programming, it becomes *stochastic* dynamic programming because of the probabilistic sampling in the action. If probabilistic inference is used along with sampling, then one may have a peculiar sort of approximate stochastic dynamic programming in which the step of choosing an action involves making an estimation that itself may be usefully carried out by stochastic dynamic programming (but with a different objective function than the objective function for whose optimization the action is being chosen).

Basically, in the COFO framework one looks at the process of optimizing F as an explicit dynamical decision process conducted via sequential application of an operation in which: Operations C_i that combine inputs chosen from a distribution induced by prior objective function evaluations, are used to get new candidate arguments to feed to F for evaluation. The reward function guiding this exploration is the quest for reduction of the entropy of the set of guesses at arguments that look promising to make F near-optimal based on the evaluations made so far.

The same COFO process can be applied equally well the case of Pareto-optimizing a set of objective functions. The definition of M_ρ^D must be modified accordingly and then the rest follows.

Actually carrying out an explicit stochastic dynamic programming algorithm according to the lines described above, will prove computationally intractable in most realistic cases. However, we shall see below that the formulation of the COFO process as dynamic programming (or simpler greedy sequential choice based optimization) provides a valuable foundation for theoretical analysis.

4 Cognitive Processes as COFO-Guided Metagraph Transformations

COFO is a highly general framework, and to use it to structure specific AI systems one has to take the next step and introduce specific sets of combinatory operations, often associated with specific incremental reward functions in the spirit of (but often not identical) the information-theoretic reward approach hinted above. In [6] explicit discussion is given to the COFO expression of a variety of cognitive algorithms used in the OpenCog AGI approach: Logical reasoning, evolutionary program learning, metagraph pattern mining, agglomerative clustering and activation-spreading-based attention allocation.

We will focus mainly here on AGI architectures such as OpenCog that have metagraphs as core meta-representational data structures – thus placing metagraphs in a dual role: 1) As a fundamental means of analyzing what the AGI system is doing from a conceptual and phenomenological perspective; 2) As the core data structure the AGI system uses to store various sorts of information as it goes about its business.

In this sort of AGI architecture, the expression of logical inference, program learning and pattern mining in combinatory-system terms ties directly back to the discussion of distinction metagraphs and associated patterns in [7]. Logical inference rules can be considered as transformations on distinction metagraphs. Bidirectional inference rules (expressed using coimplication) are rules mapping between two distinction metagraphs that have different surface form but ultimately express the same distinctions between the same observations. Programs can be viewed, using Curry-Howard type mappings, as series of steps for enacting these logical-inference-rule transformation on metagraphs, where the steps are to be carried out on an assumed reference machine. The reference machine itself may also be represented as a distinction metagraph with temporal links used to express the transitions involved in computations. Pattern mining can be expressed in terms of formal patterns in metagraphs. Clustering can be viewed as a sort of metagraph transformation that creates new ConceptNodes grouping nodes into categories. Etc.

In this context, COFO presents itself as a way of structuring processes via which sub-metagraphs transform other sub-metagraphs into yet other sub-metagraphs, where the submetagraphs are interpreted as combinators and are combined via a systematic recursive process toward the incremental increase of a particular reward function. And the common representation of multiple COFO processes involved in achieving the overall multiple-goal-achieving activities of a top-level DDS in terms of a shared typed metagraph is one way to facilitate the cognitive synergy needed to achieve high levels of general intelligence under practical resource constraints. The reliance on a common metagraph representation makes it tractable for the multiple cognitive algorithms to share intermediate state as they pursue their optimization goals, which enables the cognitive-synergy dynamic in which each process is able to call on other processes in the system for assistance when it runs into trouble.

5 COFO Processes as Galois Connections

For some of the cognitive algorithms treated in COFO terms in [6] one requires a variety of COFO that uses greedy optimization to explore the dag of possibilities, for others one requires a variety of COFO that uses some variation on approximation stochastic dynamic programming. In either case, one can use the “programming with Galois connections” approach from [12] to formalize the derivation of practical algorithmic approaches. Roughly, in all these cases, Galois connections are used to link search and optimization processes on directed metagraphs whose edge targets are labeled with probabilistic dependent types, and one can then show that – under certain assumptions – these connections are fulfilled by processes involving metagraph chronomorphisms (where a chronomorphism is a fold followed by an unfold, where both the fold and unfold are allowed to accumulate and propagate long-term memory as they proceed).

5.1 Greedy Optimization as Folding

Suppose we are concerned with maximizing a function $f : X \to R$ via a “pattern search” approach. That is, we assume an algorithm that repeatedly iterates a pattern search operation such as: Generates a set of candidate next-steps from its focus point a, evaluates the candidates, and then using the results of this evaluation, chooses a new focus point a^*. Steepest ascent obviously has this format, but so do a variety of derivative-free optimization methods as reviewed e.g. in [15].

Evolutionary optimization may be put in this framework if one shifts attention to a population-level function $f_P : X^N \to R$ where X^N is a population of N elements of X, and defines $f_P(x)$ for $x \in X^N$ as e.g. the average of $f(x)$ across $x \in X^N$ (so the average population fitness, in genetic algorithm terms). The focus point a is a population, which evolves into a new population a^* via crossover or mutation – a process that is then ongoingly iterated as outlined above.

The basic ideas to be presented here work for most any topological space X but we are most interested in the case where X is a metagraph. In this case the pattern search iteration can be understood as a walk across the metagraph, moving from some initial position in the graph to another position, then another one, etc.

We can analyze this sort of optimization algorithm via the Greedy Theorem from [12],

Theorem 1 *(Theorem 1 from [12]).* *$(\!|S \upharpoonright R|\!) \subseteq (\!|S|\!) \upharpoonright R$ if R is transitive and S satisfies the "monotonicity condition" $R^\circ \leftarrow SFR^\circ$*

which leverages a variety of idiosyncratic notation: $R \overset{S}{\leftarrow} FR$ indicates $S \cdot FR \subseteq R \cdot S$; $(\!|S|\!)$ means the operation of folding S ; $\langle \mu X :: fX \rangle$ denotes the least fixed point of f ; T° means the converse of T, i.e. $(b, a) \in R^\circ \equiv (a, c) \in R$; $S \upharpoonright R$ means "S shrunk by R", i.e. $S \cap R/S^\circ$. Here S represents the local candidate-generation operation used in the pattern-search optimization algorithm, and R represents the operation of evaluating a candidate point in X according to the objective function being optimized.

If the objective function is not convex, then the theorem does not hold, but the greedy pattern-search optimization may still be valuable in a heuristic sense. This is the case, for instance, in nearly all real-world applications of evolutionary programming, steepest ascent or classical derivative-free optimization methods.

5.2 Galois Connection Representations of Dynamic Programming Decision Systems Involving Mutually Associative Combinatory Operations

Next we consider how to represent dynamic programming based execution of DDSs using folds and unfolds. Here our approach is to leverage Theorem 2 in [12] which is stated as

Theorem 2 *(Theorem 2 from [12]).* *Assume S is monotonic with respect to R, that is, $R \overset{S}{\leftarrow} F_R$ holds, and $dom(T) \subseteq dom(S \cdot FM)$. Then*

$$M = ((\!|S|\!) \cdot (\!|T|\!)^\circ) \upharpoonright R \Rightarrow \langle \mu X :: (S \cdot FX \cdot T^\circ) \upharpoonright R \rangle \subseteq M$$

Conceptually, T° transforms input into subproblems, e.g. for backward chaining inference, it chooses (x, y, C) so that $z = C(x, y)$ has high quality (e.g. CWIG); for forward chaining, it chooses x, y, C so that z = C(x, y) has high interestingness (e.g. CWIG).
FX figures out recursively which combinations give maximum immediate reward according to the relevant measure. These optimal solutions are combined and then the best one is picked by $\upharpoonright R$, which is the evaluation on the objective function. Caching results to avoid overlap may be important here in practice (and is what will give us histomorphisms and futumorphisms instead of simple folds and unfolds).

The fix-point based recursion/iteration specified by the theorem can of course be approximatively rather than precisely solved – and doing this approximation via statistical sampling yields stochastic dynamic programming. Roughly speaking the approach symbolized by $M = (\llparenthesis S \rrparenthesis \cdot \llparenthesis T \rrparenthesis^\circ) \upharpoonright R$ begins by applying all the combinatory operations to achieve a large body of combinations-of-combinations-of-combinations-..., and then shrinks this via the process of optimality evaluation. On the other hand, the least-fixed-point version on the rhs of the Theorem iterates through the combination process step by step (executing the fold).

6 Associativity of Combinatory Operations Enables Representing Cognitive Operations as Folding and Unfolding

A key insight reported in *Patterns of Cognition* is that the mutual associativity of the combinatory operations involved in a cognitive process often plays a key role in enabling the decomposition of the process into folding and unfolding operations. This manifests itself for example in the result that

Theorem 3. *A COFO decision process whose combinatory operations C_i are mutually associative can be implemented as a chronomorphism.*

This general conclusion regarding mutual associativity resonates fascinatingly with the result from [4] mentioned above, that mutually associative combinatory operations lead straightforwardly to subpattern hierarchies. We thus see a common mathematical property leading to elegant and practically valuable symmetries in both algorithmic dynamics and in knowledge-representation structure. This bolsters confidence that the combinatory computational model is a good approach for exploring the scaling-down of generic but infeasible AGI models toward the realm of practically usable algorithms.

This conclusion regarding mutual associativity also has some practical implications for the particulars of cognitive processes such as logical reasoning and evolutionary learning. For instance, one can see that mutually associativity holds among logical inference rules if one makes use of reversible logic rules (co-implications rather than implications), and for program execution processes if one makes use of reversible computing. It is also observed that where this mutual associativity holds, there is an alignment between the hierarchy of subgoals used in recursive decision process execution and subpattern hierarchies among patterns represented in the associated knowledge metagraph.

In the PLN inference context, for example, the approach to PLN inference using relaxation rather than chaining outlined in [8] is one way of finding the fixed point of the recursion associated with the COFO process. What the theorem suggests is that folding PLN inferences across the knowledge metagraph is another way, basically boiling down to forward and backward chaining as outlined above. However, it seems this can only work reasonably cleanly for crisp inference if mutual associativity among inference rules holds, which appears to be the case only if one uses PLN rules formulated as co-implications rather than one-way implications.

Further, when dealing with the uncertainty-management aspects of PLN rules, one is no longer guaranteed associativity merely by adopting reversibility of individual inference steps. One must heuristically arrange one's inferences as series of co-implications whose associated distributions have favorable independence relationships.

7 Challenges and Prospects

The assumptions needed to get from the symmetry properties of discrete decision processes to fold and unfold operations are not entirely realistic – for instance, to get the derivations to work in their most straightforward form, one needs to assume the underlying metagraph remains unchanged as the folding and unfolding processes proceed. If the metagraph changes dynamically along with the folding and unfolding – e.g. because inference processes are drawing conclusions from the nodes and links created during the folding process, and these conclusions are being placed into the metagraph concurrently with the folding process proceeding – then one loses the straightforward result that simple approximate stochastic dynamic programming algorithms will approximate the optimal result of the decision process. This is a serious limitation, but it must also be understood that in many cases the real-time changes to the metagraph incurred by the folding and unfolding process are not a significant factor. Creating rigorous theory connecting abstract AGI theory to pragmatically relevant cognitive algorithms and their implementations is a complex matter inevitably involving some simplifications and approximations; the trick is to choose the right ones.

If one wishes to explore open-ended, evolutionary AGI systems in which multiple algorithms constructed on diverse principles interact within a common meta-representational fabric, then the conceptual and mathematical approach presented here provides an avenue for relatively elegant and concise formalization, putting diverse AI methods in a common framework. This framework has potential to ease practical complexity and performance analysis, and also connects practical operational systems with broader conceptions of AGI.

References

1. Franz, A., Gogulya, V., Löffler, M.: WILLIAM: a monolithic approach to AGI. In: Hammer, P., Agrawal, P., Goertzel, B., Iklé, M. (eds.) AGI 2019. LNCS (LNAI), vol. 11654, pp. 44–58. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-27005-6_5
2. Gibbons, J.: An initial-algebra approach to directed acyclic graphs. In: Möller, B. (ed.) MPC 1995. LNCS, vol. 947, pp. 282–303. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-60117-1_16
3. Goertzel, B.: Folding and unfolding on metagraphs (2020). https://arxiv.org/abs/2012.01759
4. Goertzel, B.: Grounding Occam's razor in a formal theory of simplicity. arXiv preprint arXiv:2004.05269 (2020)
5. Goertzel, B.: Paraconsistent foundations for probabilistic reasoning, programming and concept formation. arXiv preprint arXiv:2012.14474 (2020)
6. Goertzel, B.: Patterns of cognition: cognitive algorithms as Galois connections fulfilled by chronomorphisms on probabilistically typed metagraphs. arXiv preprint arXiv:2102.10581 (2021)
7. Goertzel, B.: Toward a general theory of general intelligence: a patternist perspective. arXiv preprint arXiv:2103.15100 (2021)
8. Goertzel, B., Pennachin, C.: How might probabilistic reasoning emerge from the brain? In: Proceedings of the First AGI Conference, vol. 171, p. 149. IOS Press (2008)
9. Goertzel, B., Pennachin, C., Geisweiller, N.: Engineering General Intelligence, Part 1: A Path to Advanced AGI via Embodied Learning and Cognitive Synergy. Atlantis Thinking Machines, Springer, Heidelberg (2013). https://doi.org/10.2991/978-94-6239-027-0
10. Hutter, M.: Universal Artificial Intelligence: Sequential Decisions Based on Algorithmic Probability. Springer, Heidelberg (2005). https://doi.org/10.1007/b138233
11. van de Meent, J.W., Paige, B., Yang, H., Wood, F.: An introduction to probabilistic programming. arXiv preprint arXiv:1809.10756 (2018)
12. Mu, S.C., Oliveira, J.N.: Programming from Galois connections. J. Log. Algebraic Program. **81**(6), 680–704 (2012)
13. Schmidhuber, J.: Godel machines: fully self-referential optimal universal self-improvers. In: Goertzel, B., Pennachin, C. (eds.) Artificial General Intelligence. COGTECH, pp. 119–226. Springer, Heidelberg (2006). https://doi.org/10.1007/978-3-540-68677-4_7
14. Schmidhuber, J.: Optimal ordered problem solver. Mach. Learn. **54**(3), 211–254 (2004). https://doi.org/10.1023/B:MACH.0000015880.99707.b2
15. Torczon, V.: Pattern search methods for nonlinear optimization. In: SIAG/OPT Views and News. Citeseer (1995)
16. Veness, J., Ng, K.S., Hutter, M., Uther, W., Silver, D.: A Monte-Carlo AIXI approximation. J. Artif. Intell. Res. **40**, 95–142 (2011)
17. Weinbaum, D., Veitas, V.: Open ended intelligence: the individuation of intelligent agents. J. Exp. Theor. Artif. Intell. **29**(2), 371–396 (2017)