Self-Programming = Learning about Intelligence-Critical System Features

Ben Goertzel

Novamente LLC, Rockville MD

Abstract. A novel characterization of "self-programming" is presented. A system is said to carry out an instance of self-programming when it undergoes learning regarding some element of its "cognitive infrastructure", where the latter is defined as the fuzzy set of "intelligence-critical" features of the system; and the intelligence-criticality of a system feature is defined as its "feature quality," considered from the perspective of solving the optimization problem of maximizing the intelligence of a multi-feature system.

1 Introduction

When considering what sort of paper to write for this workshop on "Self-Programming in AGI Systems," my first impulse was to write something exploring the possibilities for advanced self-programming implicit in the OpenCog [1] architecture. But the more I thought about that topic, the more I realized how difficult it is to crisply distinguish between self-modification and learning. After much hand-wringing, I concluded the distinction was fundamentally a fuzzy one, and set about trying to formulate this fuzzy distinction in a clear and precise way. Finally, rather than writing a paper about self-programming in OpenCog, I produced the brief paper you are reading, which addresses the question of the basic meaning of "self-programming," and the extent to which it's distinct from plain ordinary learning.

First, what does "self-programming" mean conceptually and informally? Most of the Google hits on "self-programming" refer to human beings trying to alter their own "programming" so as to achieve more conscious control over their behavior – the distinction from "learning" here being that the people involved are trying to alter their fundamental nature, their personalities and ingrained habits, rather than just trying to learn some new facts or tricks. And intuitively, what we mean when we say "self-programming" in an AGI context seems fairly close to this – something like: A cognitive system modifying the fundamental "program code" that guides its thinking, rather than just (as in ordinary learning) modifying the knowledge that's used by this program code in the course of its thinking. The concept is clearly related to the familiar concept of "self-modifying code," but the latter has generally been used to refer to simple cases such as low-level code that modifies itself for improved efficiency (while preserving identical input-output behavior, except for timing), rather than to advanced

cognitive code that self-modifies in such a way as to affect the actual results it produces for given inputs.

The informal notion of self-programming makes sense as far as it goes, but when one digs into the details, one finds the distinction between self-programming and ordinary learning is not so easy to pin down. So, my main goal in this paper is to provide a qualitative, conceptual theory of "self-programming", providing a reasonable answer to the question of what "self-programming" actually means. Along the way, I will give some specific suggestions regarding how one can turn this qualitative theory into a mathematical criterion for distinguishing self-programming from generic learning, relying on some of my prior work formalizing the notion of general intelligence, and the concept of feature quality. However, there are many ways to formalize the concepts presented here, and I am not yet sure which is the best; this may become clear only in the course of practical attempts to analyze advanced self-programming AGI systems, beyond the current state of the art. So my focus here is on qualitative understanding rather than formalization.

2 Self-Programming versus Learning

To see why the distinction between self-programming and learning is so slippery, consider that the crux of learning is the adaptive modification of a system's internal state, in response to its environment, and often associated with the system's effort to achieve certain goals. And it's not so clear how to distinguish a system's "internal state", generally speaking, from its "programming."

First of all, to compare learning with self-programming, it's beneficial to restrict the latter concept a bit. "Learning", as I'll interpret it here, has to do with a system modifying its internal state in a way that causes increase in its intelligence (although there is some subtlety here, because learning one thing could cause forgetting of another thing). Self-programming, on the other hand, could actually decrease a system's intelligence. So that's one big difference between learning and self-programming, if the latter is crudely formulated! However, if one restricts attention to intelligence-increasing self-programming, then the distinction between learning and (for short) "I-I self-programming" becomes more interesting, and more difficult to clarify.

In some cognitive architectures, there may be a clear distinction between an "infrastructural" layer of "programming", versus a higher level of "cognitive content" implemented atop this infrastructural layer. But different architectures could draw this line in different places, making the matter somewhat confusing. For instance, consider the case of two systems, A and B, where:

- A contains a basic "infrastructural programming" layer which is a very simple "agent operating system", in which multiple cognitive agents may interact and adapt to each other
- B contains a specific cognitive architecture, with a collection of 10 cognitive processes that create and share cognitive content, and collaboratively try to create intelligence

Next, suppose that one seeds A with some specific cognitive agents, corresponding closely to the 10 core cognitive processes in B.

In this case, the same 10 agents are "cognitive infrastructure" for B, but "cognitive content" for A. So, modification of these agents in a way that improves intelligence would be I-I self-programming in the context of B, but mere learning in the context of A.

This sort of dichotomy is particularly acute in the context of a flexible architecture like OpenCog, where there are multiple avenues for implementing any given cognitive process. In OpenCog, one can implement core cognitive processes, e.g., as

- C++ or Python code, "hard-wired" into the infrastructure of the system and not currently modifiable based on learning
- so-called "schema", implemented in a LISP-like language called Combo, which may do the same things as cognitive processes implemented C++ or Python code, but currently run significantly slower (and are difficult to develop and debug due to the lack of a good development environment and debugger for Combo)

Supposing that OpenCog were given the capability to import its C++ or Python infrastructural code into its declarative knowledge base, and then reason about this code (as is planned), and export modified versions in C++/python, and then recompile itself to use the modified versions. This would be a nice software engineering advance over the current OpenCog framework, but if we consider the two cases

- Case A: OpenCog modifies its PLN inference process, which is implemented in C++, via modifying that C++ source-code
- Case B: OpenCog modifies its PLN inference process, which is implemented as a cognitive schema

would it fundamentally be doing anything different in the two cases? Not really.

One could, of course, simply let each cognitive architecture separately define the boundary between cognitive infrastructure and cognitive content, and then define self-programming in the context of architecture X as learning that impacts the cognitive infrastructure of X – but this seems a bit unsatisfying. However, what this example indicates is that, if the notion of self-programming is to be assigned a meaning in a way that's portable across multiple cognitive architectures, some care must be taken.

My suggestion is that "cognitive infrastructure" should be taken as a fuzzy concept, specifically as the fuzzy set of "intelligence-critical" features of a system. The degree to which an instance of learning counts as I-I self-programming, then has to do with the extent to which the thing being learned about is intelligence-critical. This provides a way of thinking about self-programming, that goes beyond any particular cognitive architectures. The following section presents further details along these lines.

3 Characterizing Self-Programming as a Fuzzy Property of Instances of Learning

Suppose one has a system so that

- one can quantitate the intelligence of the system and its close variants (some specific ways to do this will be referenced below)
- one can usefully divide the state of the system, over an interval of time, into (not necessarily disjoint) "features" representing fairly distinct aspects of the system's knowledge base or operational infrastructure

Taking OpenCog as an example, one way to decompose the system into disjoint features would be to look at

- Atoms (nodes and links) or schema (small programs)
- C++/python objects and functions

, and their states during time intervals, as features

In this context, given a system S, a feature x of S and an interval of time T, I suggest one may roughly define

$$self\_programming(x;T) = amount\_of\_learning\_about(x;T) * intelligence-criticality(x;T)$$

That is: the degree to which x is self-programmed by S during interval T, is the product of the degree to which S learns about x during interval T, and the degree to which x is intelligence-critical for S during interval T.

To make this notion meaningful I must specify what I mean by amount of learning, and by intelligence-criticality.

First of all, we may take $amount\_of\_learning\_about(x;T)$ to mean the degree that S has adapted x via learning, during interval T. Most simply, I suggest, this means that x has changed in a particular way that correlates with an increase in S's intelligence. One might formalize this crudely by looking at the intelligence of systems similar to S where x did change in this particular way, versus systems similar to S where x did not. To the extent that the intelligence of systems in the former class is significantly greater, I may say that S has *learned* about x during the interval T.

To define this formally in a more sophisticated way, using the formal agents model in [3], one could look at an ensemble of possible world-histories, and compare S's intelligence averaged over those world-histories where x has changed in the same way as it did in the current instance (during T), to S's intelligence averaged over all world-histories. Or, one could look at an ensemble of possible intelligent systems containing feature x, and compare the intelligence of all systems in the ensemble over those world-histories where x has changed in the same way as it did for S in the current instance (during T), to the general average intelligence of a system in an ensemble over an arbitrary world-history. In each

case there is a need to define a probability distribution over world-histories, and in the second case over systems as well, in order to weight the averages.

Next, how does one define the "intelligence-criticality" of an element of an AGI system, in a way that doesn't depend on the particulars of any AGI architecture?

This is a subtle matter and my suggested solution may not appeal to everyone. But it seems some such definition must be posited, or the notion of "self-programming" becomes relatively uninteresting due to the inability to distinguish it from learning in a way that spans multiple AGI approaches. So I will suggest a working definition of intelligence-criticality that seems useful in practice.

Qualitatively, I suggest that a feature x of an intelligent system S is intelligence-critical to the extent that changes in x are useful for predicting changes in the intelligence of S. According to this idea, an act of learning then qualifies as an act of "self-programming" to the extent that its learning activity modifies those aspects of itself, on which its intelligence most critically relies.

One can make this more precise via combining two formalisms presented in prior works.

First, in [3] a mathematical characterization of general intelligence is given, which specifies the degree of general intelligence possessed by a given agent, relative to a probability distribution over (environment, goal) pairs. This is not a practical "AGI IQ test", but it's a formal definition of intelligence suitable to use in developing other formal notions, such as, in this instance, self-programming. It could be used as the basis for various practical tests, but that's a matter for other papers.

Next, in [2] a general theory of "feature quality" is given. Specifically, a definition is given of the "feature quality" of an element x of a feature structure X being used as input to a program f, relative to an objective function Φ evaluated with f as input. Roughly, what the feature quality of x tells you is how much information the value of x tells you about the output of a program $g(X)$, averaged over all g that are (fuzzily) good at optimizing Φ (with the fuzzy degree of "goodness" toggled by a parameter p). For instance, a feature x of intelligent systems would have high "feature quality" for predicting system intelligence (relative to a particular distribution over (environment, goal) pairs) if it has a high impact on the output of systems $g(X)$ averaged over all g that are reasonably intelligent. That is, with this set-up, x's feature quality measure the extent to which x makes a difference to the output of intelligent systems. This seems one reasonable way to measure the degree to which x is intelligence-critical.

That is, I map feature quality theory into intelligence theory via equating

- Degree of general intelligence = objective function Φ
- Intelligent agent = program f
- Aspect of intelligent agent's internals = feature x

Putting these pieces together, I then have a characterization of self-programming as a fuzzy property of systems, in a manner that applies across any sort of AGI system.

4 Gauging the Self-Programming-Orientation of an AGI system

So far, I have defined the degree to which an instance of learning qualifies as self-programming, rather than the degree to which a system as a whole is engaged in self-programming. However, one way to think about the latter is simply to ask, for a randomly chosen act of learning by the system, how intelligence-critical are the features being adapted via that act of learning? Or one could choose a random act of learning *with probability proportional to the amount learned*, and then asking how intelligence-critical are the features being adapted via that act of learning. These measures tell you something about how much of the system's learning is self-programming-oriented, and how much involves learning of less intelligence-critical aspects of the system.

An interesting question is: What sorts of systems, in what sorts of environments, will find it optimal to adopt a strong orientation toward self-programming? Intuitively it seems that if a system is in an environment which changes fairly significantly over time, and in which it can generally achieve significantly greater intelligence via using all its resources than via using a small fraction of them, then the system will probably gain significant advantage via self-programming.

5 Conclusion

I have presented a novel characterization of "self-programming", which implies a fuzzy but precisely-defined distinction between self-programming and ordinary learning. Roughly, a system is said to carry out an instance of self-programming when it undergoes learning regarding some element of its "cognitive infrastructure", where the latter is defined as the fuzzy set of "intelligence-critical" features of the system; and the intelligence-criticality of a system feature is defined as its "feature quality," considered from the perspective of solving the optimization problem of maximizing the intelligence of a multi-feature system.

But how *useful* is the notion of self-programming for AGI? This is, of course, a different question – an interesting and important one, but beyond the scope of the present paper! But I will close with a few possibly relevant comments.

One pertinent issue that the present paper raises, though, is whether the designers or creators of a system will necessarily recognize when an AGI system of their invention is self-programming versus merely learning. How easy is, it, in practice, to tell cognitive infrastructure from mere cognitive content, in a complex, intelligent, self-organizing system? An AGI system may well originate cognitive processes on its own, which then become so critical to its thought as to be considered "infrastructural", perhaps more so than the particular lower-level processes created by human programmers.

Another issue is that one might want an AGI system to be particularly careful about making changes to its cognitive infrastructure. So, for this reason, it might be good for an AGI system to acquire a talent for recognizing the intelligence-criticality of its various features, including those which it created or heavily modified. Then it would know to be extra-judicious in tweaking those aspects of itself, or thoroughly backing itself up before it makes the tweaks.

All in all, I currently remain ambivalent about whether the notion of "self-programming" per se is an important one for AGI. However, I do feel that intelligence-criticality may be a worthwhile quality to think about, and for AGI systems to be instrumented to recognize in themselves. In this view, the key question regarding self-programming is whether AGI systems need special learning mechanisms to learn about intelligence-critical things, or whether they can just use the same learning mechanisms in these cases as in all the rest. In the OpenCog design at present we make the latter assumption, but other cognitive architectures may do it differently, and at this stage it's certainly worth exploring various possibilities.

References

1. Goertzel, B.: Opencogbot: An integrative architecture for embodied agi. Proc. of ICAI-10, Beijing (2010)
2. Goertzel, B., Geisweiller, N.: What are good features, and how can we find them faster via metalearning? Draft article in preparation (2011), `http://goertzel.org/GoodFeatures.pdf`
3. Goertzel, B., Ikle', M.: Toward a formal definition of real-world general intelligence. In: Proc. of the Third Conference on Artificial General Intelligence (2010)