
An Attempt to Generalize AI Part 4: Modeling Efficiency

By Paul Almond

20 March 2010

Website:

<http://www.paul-almond.com>

E-mail:

info@paul-almond.com

This is the fourth in a series of articles attempting to give an overview of how minds may work and how similar systems could be implemented in computers. The first article described a probabilistic, hierarchical modeling system, based on patterns, which are sets of pattern instances, intended to provide a general ontology. The second article described the use of this for planning actions. A serious issue with AI systems is ensuring that the computation that is done is useful. A system like this finds patterns, then patterns based on the patterns, and so on. Only a small amount of this computation will be relevant. Unless something is done to prevent it, there will be a huge proliferation of pattern instances that are of little use in the making of predictions. This article explores this issue, and starts to consider, very broadly, what may be involved in reducing the number of pattern instances in the model. This will be built on in later articles, which will discuss specific methods of achieving what is discussed here.

Table of Contents

1 Introduction	5
2 General Idea of the Hierarchy.....	6
3 The Impracticality of Computing Everything	9
4 Possible Ways of Reducing the Amount of Computation.....	10
4.1 The Need to Reduce the Number of Pattern Instances.....	10
4.2 Making Particular Patterns More or Less Represented	10
4.3 Making Groups of Patterns More or Less Represented	10
4.4 Removing Individual Pattern Instances	11
4.5 Varying the Density of the Hierarchy “Locally”	11
5 The Actual Hierarchy and the Conceptual Hierarchy	13
5.1 The two hierarchies are different.	13
5.2 Requiring Fully Specified Pattern Instances.....	13
6 Issues with Not Requiring Fully Specified Pattern Instances	15
7 A Word on Neurons	16
8 Conclusion.....	17
9 Bibliography	18

Table of Figures

Figure 1: Patterns and the Hierarchy.....	7
Figure 2: Actual and Conceptual Hierarchies.....	14

List of Abbreviations

AI artificial intelligence

1 Introduction

This article is the fourth in a series about artificial intelligence (AI) and how our own minds might work. The first article, *An Attempt to Generalize AI - Part 1: The Modeling System*, should be read before this one and is available at <http://www.paul-almond.com/AI01.pdf>.¹ The second article, *An Attempt to Generalize AI - Part 2: Planning and Actions*, is at <http://www.paul-almond.com/AI02.pdf>.² The third article, *An Attempt to Generalize AI - Part 3: Forgetting*, is at <http://www.paul-almond.com/AI03.pdf>.³ (Reading the second and third articles before this one is suggested, but not absolutely necessary, as the focus here is on the hierarchy itself, rather than actions or forgetting.)

These three articles together were intended to give an idea of how humans may model the world, plan actions and discard information from the model when it is no longer useful.

The system as described so far is impractical because there is nothing to focus modeling on what is important. All pattern instances of a pattern are explicitly represented, whether useful or not. The system will always be finding irrelevant relationships, and building irrelevant layers of abstraction on top of them.

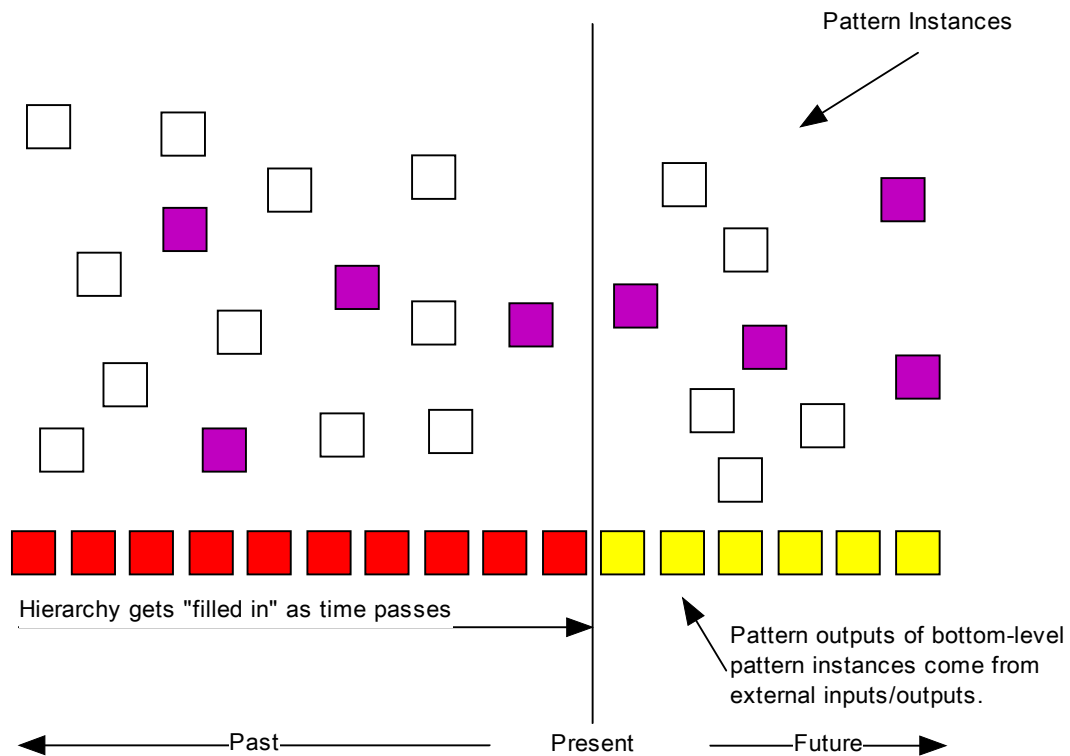
¹ Almond, P. (2010). An Attempt to Generalize AI - Part 1: The Modeling System. *paul-almond.com*. <http://www.paul-almond.com/AI01.pdf>. (Also available at <http://www.paul-almond.com/AI01.doc>.)

² Almond, P. (2010). An Attempt to Generalize AI - Part 2: Planning and Actions. *paul-almond.com*. <http://www.paul-almond.com/AI02.pdf>. (Also available at <http://www.paul-almond.com/AI02.doc>.)

³ Almond, P. (2010). An Attempt to Generalize AI - Part 3: Forgetting. *paul-almond.com*. <http://www.paul-almond.com/AI03.pdf>. (Also available at <http://www.paul-almond.com/AI03.doc>.)

2 General Idea of the Hierarchy

The system described in the previous articles is a hierarchy based on patterns. Each pattern is a set of pattern instances, distributed throughout the hierarchy and described by a pattern specification. The pattern specification consists of a logic specification, describing how each pattern instance of a pattern determines its pattern output from its labeled inputs, and a construction specification, describing how each pattern instance connects its labeled pattern inputs to the pattern outputs of other pattern instances. The bottom level of the hierarchy consists of special pattern instances, each corresponding to some external input/output event occurring at some instant. These pattern instances record the inputs/outputs that have occurred over a period of time. The rest of the hierarchy is built on top of this according to the pattern specifications of patterns. (See Figure 1, on page 7.)



Key

- Pattern instances of the same pattern. All these pattern instances are related in some way.
- Pattern instances of other patterns.
- Bottom-level pattern instances corresponding to external inputs/outputs that have already occurred.
- Bottom-level pattern instances corresponding to external inputs/outputs that have yet to occur.

Figure 1: Patterns and the Hierarchy

Because the pattern instances of a pattern are all related by being described by the same pattern specification, statistical observations of pattern instances with known pattern inputs can be used to assign probabilities to pattern instances with partially and probabilistically known pattern outputs. This can be used to propagate probabilities up the hierarchy to determine probabilities for high-level pattern instances, based on previous inputs/outputs, and down the hierarchy, to fill in the parts of the hierarchy corresponding to future inputs/outputs.

The hierarchy is a collection of probability values – pattern instances – each of which collapses at some point into a known state, with this collapse process being ongoing as inputs and outputs occur. When a pattern instance has collapsed – when its state is determined, directly or indirectly, by inputs or outputs that have already occurred – it is *fixed*. The fixed pattern instance is then used to provide statistical data for the pattern to which it belongs, and it also serves as a pattern input for any other pattern instances that use it.

All this enables the hierarchy to predict future inputs/outputs probabilistically, based on previous inputs/outputs and this principle is used in selecting outputs.

3 The Impracticality of Computing Everything

The hierarchical system as described so far makes no distinction between relevant and irrelevant information: It would compute everything. Low-level inputs/outputs are abstracted into higher-level pattern instances, which are abstracted still further into still higher-level pattern instances and so on. Some low-level pattern instances would be almost completely irrelevant to what is going on, yet would still be the basis for abstraction upon abstraction. This would be wasteful of computing resources.

Human brains could never function in this way. As an example, imagine that you see a competition in a local store. A carpet is displayed, with some intricate pattern, and some kind of message is encoded into this pattern. Finding the message would require a lot of analysis. You want to win the competition, and you may do, because your brain is analyzing the pattern in the carpet. Now, imagine that you are disturbed at night by a thief in your house. You go downstairs to challenge the thief, or you go to get help, and you happen to see your own carpet, which is also intricately patterned. Your brain gives the carpet the same attention as the carpet in the competition, looking for patterns, and patterns made from those patterns and so on, abstracting the basic elements in the carpet in many ways and to a great extent. Doing this would be pointless. At the same time, you would be subjecting a mark on the wall to the same abstraction. If you detected any relationship between the sounds of the thief and the pattern in the carpet you would start abstracting this, and then start abstracting the abstractions.

This is all unworkable. A human brain could not work like this, and nor can a practical AI system. There needs to be a way of limiting the number of pattern instances that are explicitly represented in the system, and ensuring that the ones which are represented are relevant.

4 Possible Ways of Reducing the Amount of Computation

4.1 The Need to Reduce the Number of Pattern Instances

Any data storage and computation in the hierarchy is based on pattern instances, so any approach aimed at reducing the amount of computation in the hierarchy must be aimed at reducing the number of pattern instances in the hierarchy.

This reduction can take a number of forms:

- making particular patterns more or less represented.
- making groups of patterns more or less represented.
- removing individual pattern instances.
- varying the density of the hierarchy locally.

I will now say more about each of these.

4.2 Making Particular Patterns More or Less Represented

A pattern has a set of associated pattern instances. As the pattern instances of a pattern become fixed, they eventually become obsolete and get removed. New pattern instances must be continually added by the pattern. We might not treat all pattern's equally. We might allow patterns to have different numbers of pattern instances represented at any time, according to some criteria. This could be done by controlling the rates at which different patterns can introduce new pattern instances into the hierarchy, but it could also involve removing pattern instances from the hierarchy after they have been added, if the permitted level of representation of a pattern is reduced.

4.3 Making Groups of Patterns More or Less Represented

This is similar to the above approach, except that it involves assigning pattern instances to groups, and dealing with them on a group basis. We might assign some group of pattern instances a permitted level of representation, which is varied, and as it is varied so are the numbers of pattern instances which patterns in that group have.

4.4 Removing Individual Pattern Instances

An individual pattern instance is removed from the hierarchy, if it is decided, according to some criteria, that computing resources are being wasted on it.

4.5 Varying the Density of the Hierarchy “Locally”

This involves changing the density of the hierarchy in a given “region”: increasing the number of pattern instances within a given part of the hierarchy. This use of spatial terms like “locally” or “region” needs qualification. As explained in previous articles, the hierarchy is not based on a strong, geometrical analogy. A pattern instance does not have to have some coordinates in some imaginary space. All that matters about a pattern instance is what it is connected to.

Nevertheless, we can still apply a *weak, geometrical analogy*, using the concept of *logical distance* between pattern instances. If one pattern instance were directly connected to another, using it as a pattern input, we would say that those two pattern instances were “close together”. Two pattern instances connected together through an intermediate pattern instance would be less close. We could generalize this by saying that the logical distance between any two pattern instances is the number of connections between pattern instances which must be traversed to move between them. Logical distance is not an arbitrary concept. This is only one way, and the simplest, of determining logical distance. A more sophisticated method might involve averaging over different paths, or determining the logical distance between two pattern instances by determining the logical distance between each and some third pattern instance, and doing this a number of times, changing the third pattern instance each time.

However we compute it, the concept of logical distance allows us to impose a weak, geometrical analogy on the system: It allows us to look at a pattern instance and say which pattern instances are “nearby” in some sense, without having to impose any well-defined spatial relationships on the system.

A weak, geometrical analogy such as this can be used to vary the local density of the hierarchy, because it gives a non-spatial meaning to the term “local”. We might set a limit on the number of pattern instances that patterns can create within a certain logical distance of some pattern instance, or we might remove pattern instances that have already been placed there. We could combine this with one of the approaches above. For example, if we are reducing the local density in some region of the hierarchy, and also allowing patterns to have different amounts of representation, we may select pattern instances for removal according to these levels of representation.

The idea of this is that most of the hierarchy that could be represented can end up not being explicitly represented. If we can define appropriate methods for manipulating the distribution of pattern instances in the hierarchy, we can produce a model which only

An Attempt to Generalize AI – Part 4: Modeling Efficiency

represents what is relevant about reality. Being careful not to go too far with a spatial analogy, we might imagine the hierarchy being like cheese – being full of holes.

5 The Actual Hierarchy and the Conceptual Hierarchy

5.1 The two hierarchies are different.

There is a *conceptual hierarchy* describing a hierarchy of relationships between the AI system's inputs and outputs and the *actual hierarchy* set up by the patterns in the AI system is a representation of this. The conceptual and actual hierarchies will not be the same. The conceptual hierarchy would contain every possible relationship, whereas the actual hierarchy will contain only those relationships associated with its patterns. The conceptual hierarchy will extend without limit, whereas the actual hierarchy will be limited in extent. If we start to limit particular patterns, or vary the local density of the actual hierarchy this will make it still more different to the conceptual hierarchy. If the right parts of the actual hierarchy are constructed, this should not be a problem, though. The idea is to construct an actual hierarchy which represents the conceptual hierarchy well enough for the restrictive task of predicting the particular inputs/outputs that interest us.

5.2 Requiring Fully Specified Pattern Instances

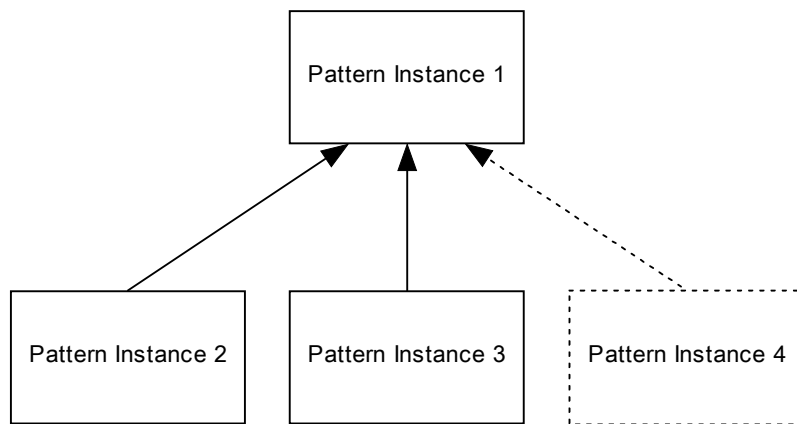
As the actual hierarchy is not exactly the same as the conceptual hierarchy, in fact, only ever being a small part of it, the "wiring" of the actual hierarchy is incomplete. So far, however this series of articles has implicitly assumed that every pattern instance in the actual hierarchy will have all its pattern inputs connected to other pattern instances in the actual hierarchy: A pattern input will never be left "undefined" because the input is assumed to come from another pattern instance in the actual hierarchy.

It is now time to question the idea of requiring such fully specified pattern instances. This requirement creates two possible issues:

The first issue is that if a pattern instance is removed from the hierarchy, everything "above" that pattern instance has to be removed. Any pattern instance using it as a pattern input needs to be removed, as it is now missing a pattern input. Any pattern instances using this pattern instance in their pattern inputs need to be removed and so on. The removal of a single pattern instance, at a low level of the hierarchy, could trigger the removal of other pattern instances all the way up the hierarchy. A single, high-level pattern instance may be indirectly connected to many low-level pattern instances, and a single, low-level pattern instance may be indirectly connected to many high-level pattern instances. The removal of a single, low-level pattern instance could cause the removal of many high-level pattern instances, losing a lot of the hierarchy. The hierarchy may then be reconstructed by patterns, but this is undesirable.

The second issue is that constructing new pattern instances, and connecting their own pattern inputs to other pattern instances, may be more complicated if all the pattern inputs of a pattern instance must always be connected. Patterns themselves will not be very sophisticated. They may be generated randomly, or by some kind of Darwinian evolution process, and we should expect as little of them as possible.

Both of these issues, together, raise a question: Should we consider allowing pattern instances to exist with incomplete wiring? This would mean allowing a pattern to construct a pattern instance and to connect it to the rest of the hierarchy without specifying all the pattern inputs, and it would mean allowing a pattern instance to remain in the hierarchy when one or more of the pattern instances serving as pattern inputs for it have been erased. In both cases, the unspecified pattern inputs would really be connected to pattern instances in the conceptual hierarchy, making them for all practical purposes, non-existent in the actual hierarchy. (See Figure 2, below. In this diagram a pattern instance is shown, Pattern Instance 1, with two pattern inputs coming from Pattern Instance 2 and Pattern Instance 3 in the actual hierarchy. A third pattern input is left undefined, but it can be assumed to be coming from another pattern instance, Pattern Instance 4, in the conceptual hierarchy.)



Key

- Actual hierarchy and conceptual hierarchy
- Conceptual hierarchy only

Figure 2: Actual and Conceptual Hierarchies

6 Issues with Not Requiring Fully Specified Pattern Instances

If the description of the AI system is altered to allow partially connected pattern instances, two issues arise.

Firstly, the logic application stage, as it has been described, could not be used to assign a pattern output value to a pattern instances unless all of its pattern inputs are defined and have known values. Logic application cannot assign a pattern output value to a pattern instance if it, or any of the pattern instances on which it depends, directly or indirectly, have any missing pattern inputs. The likelihood of a single pattern instance high up the hierarchy being indirectly connected to many pattern instances lower down means that hardly any pattern instances will have their states assigned in logic application. The logic application process *in its current form* would become unworkable.

Secondly, the basic forgetting procedure described in the third article⁴ could not be applied, as it stands, for the same reason. It relies on pattern instances which have been “fixed” in logic application – and if partially specified pattern instances are allowed there will be few such pattern instances.

This does not mean that we cannot allow partially specified pattern instances. What I have said previously in this article makes an argument for allowing them. If we allowed them, however, the system would need modifying to remove a sharp distinction between “fixed” pattern instances and probabilistic ones. It would still work in basically the same way, but would deal completely with probabilities. Such a change to the system should now be considered.

⁴ Almond, P. (2010). An Attempt to Generalize AI - Part 3: Forgetting. *paul-almond.com*. <http://www.paul-almond.com/AI03.pdf>. (Also available at <http://www.paul-almond.com/AI03.doc>.)

7 A Word on Neurons

In the first article of this series, I stated, *“It may be tempting to think of a pattern instance as corresponding to a neuron, but I would caution against this. For reasons that will be explained later, the structure of the hierarchy is not fixed. Only a small number of pattern instances that could be represented will be explicitly represented in the hierarchy at any time, suggesting that the relationship between pattern instances and neurons will be more complex.”*⁵ That statement may not be correct. The reason I made it was because the structure of pattern instances needs to be dynamic, with pattern instances being created when needed, and disposed of when they represent irrelevant abstraction. That seemed to suggest that any mapping between pattern instances and neurons would not be simple, with pattern instances being more like “software-level” things and neurons being more like “hardware-level” things. However, the structure of the brain *is* dynamic. Connections between neurons can change. What if a pattern instance *could* correspond in some simple way to a neuron or group or neurons, with the connections of neurons changing to represent changes in the actual hierarchy? If a pattern instance is irrelevant we should hardly expect a neuron to be destroyed – neurons would be getting destroyed all the time if that were the case – but what if, when a pattern instance ceased to exist, the corresponding neuron or neurons became free for use in representing another pattern instance?

If the mapping between pattern instances and neurons were as simple as this, some method would be needed of sharing the statistical data for a pattern among its pattern instances. This would seem to suggest that the neurons would have to do that as well, or that there would have to be a second network serving this purpose.

The main issue here is how quickly the hierarchy would need to be rewired to keep it relevant enough, and whether the connections between neurons could be changed quickly enough. For now, I will leave the issue of mapping between pattern instances and neurons open.

⁵ Almond, P. (2010). An Attempt to Generalize AI - Part 1: The Modeling System. *paul-almond.com*. <http://www.paul-almond.com/AI01.pdf>. p10. (Also available at <http://www.paul-almond.com/AI01.doc>.)

8 Conclusion

The AI system, as described in previous articles in this series, has no way of only performing relevant abstractions. A system like this finds abstractions, then abstractions based on the abstractions, and so on. Only a small amount of the computation performed by such a system would be relevant. There would be a huge proliferation of pattern instances that are of little use in the making of predictions.

Measures need to be taken to reduce the number of pattern instances appearing in the hierarchy, ensuring that those pattern instances that do appear are particularly relevant. Specific procedures for doing this have not yet been described, but the general ways in which numbers of pattern instances could be limited have been discussed.

The concepts of the *conceptual hierarchy* and the *actual hierarchy* have also been discussed. The conceptual hierarchy is a theoretical hierarchy that would exist with no limits on computation and no attempt to limit the number of patterns or pattern instances. The conceptual hierarchy is the hierarchy of all possible abstractions of the low-level input/output data. The *actual* hierarchy is the smaller version of this that exists in the AI system. Any method of limiting the number of pattern instances should produce an actual hierarchy represents the parts of the conceptual hierarchy most relevant in making the required predictions.

One issue is whether to allow pattern instances in the actual hierarchy to have some of their pattern inputs corresponding to pattern instances that exist in the conceptual hierarchy, but not in the actual hierarchy. In practical terms, this means allowing pattern instances with incompletely specified pattern inputs. Advantages of this are that it allows the removal of pattern instances without requiring the removal of huge amounts of the hierarchy and that it makes addition of pattern instances simpler – meaning less demands are made of whatever relatively simple construction system is being used by patterns. It causes a problem in that few pattern instances would be completely dependent on pattern instances in the actual hierarchy and the process of fixing would become impractical. The solution to this would be to modify the hierarchy to be completely probabilistic, with no fundamental distinction between fixing and other operations. The hierarchy would still work in basically the same way, however. This should now be considered.

9 Bibliography

Almond, P. (2010). An Attempt to Generalize AI - Part 1: The Modeling System. *paul-almond.com*. Retrieved 20 March 2010 from <http://www.paul-almond.com/AI01.pdf>. (Also available at <http://www.paul-almond.com/AI01.doc>.)

Ibid. p10.

Almond, P. (2010). An Attempt to Generalize AI - Part 2: Planning and Actions. *paul-almond.com*. Retrieved 20 March 2010 from <http://www.paul-almond.com/AI02.pdf>. (Also available at <http://www.paul-almond.com/AI02.doc>.)

Almond, P. (2010). An Attempt to Generalize AI - Part 3: Forgetting. *paul-almond.com*. Retrieved 20 March 2010 from <http://www.paul-almond.com/AI03.pdf>. (Also available at <http://www.paul-almond.com/AI03.doc>.)