An Attempt to Generalize AI Part 15: A Complete Description

By Paul Almond

10 July 2010

Website: http://www.paul-almond.com
E-mail: info@paul-almond.com

This is the fifteenth in a series of articles attempting an overview of how minds may work and how similar systems could be implemented in computers. For readers wanting an understanding of what is being suggested, this article is an ideal starting point, as it gives a complete description of the cognitive model as it stands at the time of writing. This article brings everything together so that it can be understood by reading just this article, instead of having to read fourteen articles and having to read about ideas that are changed later, or read through a long discussion in which the concepts are developed. The cognitive model and approach to artificial intelligence uses a probabilistic hierarchy based on *patterns*. A pattern has a specification describing a set, or population, of *pattern instances*, distributed throughout a hierarchy containing the pattern instances of all the patterns. Each pattern's set of pattern instances is used to obtain statistical information for probabilistic predictions. Each pattern's population of pattern instances is to be described in a very general way, to provide a very general ontology. The hierarchical model is actually used to plan the system's actions, and this implies that what people regard as the "self" is really an object in the hierarchical model. The hierarchy needs to be relevant, with pattern instances that are useful being featured in the hierarchy, and used as a basis for exploratory extension of the hierarchy, while less useful pattern instances are removed, so that the hierarchy "grows" into high-relevancy regions, and this is achieved by a back-propagation relevance measurement process which assigns relevance values to pattern instances. Relevance can also be provided by reflexive outputs: special outputs that are made in the same way as normal outputs, but which alter the hierarchical model.

Table of Contents

1 Introduction 8
2 The Basic Structure of the Hierarchy 10
2.1 Patterns and Pattern Instances 10
2.2 Pattern Specifications 13
2.2.1 The General Idea of Pattern Specifications 13
2.2.2 The Logic Specification of a Pattern 15
2.2.3 The Construction Specification of a Pattern 16
2.2.4 Generation of Pattern Specifications 19
3 The Conceptual and Actual Hierarchies and the Atemporal Nature of the Conceptual Hierarchy 20
3.1 The Conceptual Hierarchy 20
3.2 The Actual Hierarchy 20
4 Probabilistic Propagation in the Actual Hierarchy 22
4.1 The General Idea of Probabilistic Propagation in the Actual Hierarchy 22
4.2 Logic Application 22
4.3 Statistics Generation 25
4.4 Statistics Application 26
4.5 Notes on Logic Application, Statistics Generation and Statistics Application 28
4.6 Convective Delusion 30
5 Planning of Actions 32
5.1 The General Idea Behind Planning of Actions 32
5.2 The Action Selection Process 33
5.2.1 Computation of Evaluation Function Score 33
5.2.2 Input of Evaluation Function Score 33
5.2.3 Selecting Actions 33
5.3 How the Action Selection Process Works 34
5.3.1 The *real* planning process 34
5.3.2 Why the Action Selection Process is Needed 36
5.3.2.1 Problem 1: Initially, there is no history of suitable behavior. 36
5.3.2.2 Problem 2: Random drift would occur. 36
5.3.2.3 The Same Problem 36
5.3.2.4 How the Action Selection Process Gets the System Started 36
5.3.2.5 Learning as Modeling 37
5.3.2.6 How the Action Selection Process Prevents Random Drift 38
5.3.3 Why the Evaluation Function Score is Used as an Input 39

5.4 Possible Variations on the Action Selection Process 40
5.4.1 The Possibility of "Giving the System More of a Start" 40
5.4.2 The Possibility of "Coming in Higher Up" in Action Selection 40
6 The "Self" 42
7 Geometry 44
7.1 Breaking the Strong, Geometrical Analogy 44
7.2 Weak Geometry 45
8 Relevance 47
8.1 The General Idea Behind Processing for Relevance 47
8.2 What parts of the hierarchy are relevant? 47
8.2.1 Analyzing the Actual Hierarchy 47
8.2.2 The Purpose of the Hierarchy 48
8.2.3 The Importance of the Action Selection Process 48
8.2.4 Does the propagation of information *from* previous inputs/outputs tell us anything useful? 49
8.2.5 Does the propagation of information *into* pattern instances tell us anything useful? 50
8.2.5.1 Let's just consider the structure… 50
8.2.5.2 But what if we consider probabilities too? 50
8.2.5.3 The amount of effect is not just determined by probability. 53
8.3 The Relevance Measurement Process (RMP) 53
8.3.1 Description of the Relevance Measurement Process (RMP) 54
8.3.2 Determining the Amount of Effect of a Pattern Instance 55
8.4 The Basic, Exploratory Relevance Process (BERP) 57
8.4.1 Starting with a Basic, Exploratory Process 57
8.4.2 What Would Happen With an Unrestricted Hierarchy. 57
8.4.3 The Objective of the Basic, Exploratory Relevance Process 57
8.4.4 What the Basic, Exploratory Relevance Process Will Do 58
8.4.4.1 Removal of Pattern Instances 58
8.4.4.2 Addition of Pattern Instances 58
8.4.5 Why add more pattern instances in higher-relevance regions? 59
8.4.6 How the Hierarchy Should Develop 60
8.4.6.1 How the Structure of the Hierarchy Changes 60
8.4.6.2 Exploration is not random. 61
8.4.7 Use of the Relevance Measurement Process by the Basic, Exploratory Relevance Process 61
8.5 Description of the Basic, Exploratory Relevance Process 63

8.5.1 Addition of Pattern Instances 63
8.5.2 Removal of Pattern Instances 63
8.5.3 Ghost Pattern Instances 64
8.6 Relevance and Patterns 65
8.6.1 The General Idea 65
8.6.2 More Sophistication in Achieving Pattern Relevance 66
8.7 A General View of the Exploratory Relevance Process 67
8.7.1 The ERP is exploratory. 67
8.7.2 The ERP is *directed* by the hierarchy. 67
8.7.3 High relevance is not everything. 69
9 Possible Improvements to the Basic, Exploratory Relevance Process 70
9.1 The General Idea of Possible Improvements to the Basic, Exploratory Relevance Process 70
9.2 Update Frequency Dependent on Relevance 70
9.2.1 The General Idea of Update Frequency Dependence on Relevance 70
9.2.2 Update Frequency for Probabilistic Propagation Dependent on Relevance 70
9.2.3 Update Frequency for Relevance Back-Propagation Dependent on Relevance 71
9.2.4 Long and Short-Term Relevance 71
9.2.5 Will this be of any use? 72
9.3 Pattern Instance Addition *Explicitly* Dependent on Relevance 73
9.3.1 The Basic Approach 73
9.3.2 Clarification 73
9.3.3 A Possible Change 74
9.4 Relevance Computed for *Regions* of the Hierarchy 75
10 Forgetting 77
10.1 The General Idea Behind Forgetting 77
10.2 Relevance Propagation and Degree of Uncertainty 77
10.2.1 The Relevance Measurement Process 77
10.2.2 The Special Case of a Pattern Instance with No Uncertainty 78
10.2.3 A Pattern Instance with *Almost* Complete Certainty 79
10.2.4 A Pattern Instance with *Any* Degree of Certainty 80
10.3 Modifying the Relevance Measurement Process to Take Account of Degree of Certainty 80
10.3.1 The Basic Modification 80
10.3.1.1 Total Amount of Relevance 80

10.3.1.2 Allocation of Relevance 81
10.3.2 The Role of Ghost Pattern Instances in Preventing Loss of Information..... 81
10.4 Issues with Forgetting 82
10.4.1 What if we just remember the probability? 82
10.4.2 Isn't this inconsistent? 83
10.4.2.1 Two Different Standards 83
10.4.2.2 Why there is No Inconsistency 83
10.4.2.3 An Analogy 84
10.4.2.4 Relevance is about allocation of computing resources. 84
10.5 Bottom-Level Pattern Instances and Forgetting 84
11 Reflexive Outputs as a Way of Providing Relevance 86
11.1 The Idea of Reflexive Outputs 86
11.2 Why Reflexive Outputs Would Work 86
11.3 Analogies for Reflexive Outputs 87
11.3.1 Analogy 1: Indirect Manipulation of the System 87
11.3.2 Analogy 2: Obtaining a Better View 87
11.4 A Philosophical View of Reflexive Outputs 88
11.4.1 View 1: Reflexive Outputs are the "Self" Controlling the Hierarchy 88
11.4.2 View 2: Actions just seem to be initiated by the "self", but the "self" is just part of the model. 89
11.4.3 Which view of reflexive outputs is correct? 90
11.5 Complications with Implementation of Reflexive Outputs 90
12 Pattern Instance Construction Alternatives 91
12.1 The General Idea 91
12.2 The Previously Described Approach to Pattern Instance Construction 91
12.3 The Possibility of Different Construction Methods 92
12.4 Different Ways in Which the Construction Specification Could Work 92
12.4.1 Pattern instances are *made* by patterns. 92
12.4.2 Pattern instances are made *randomly* and *selected* by patterns. 93
12.4.3 Pattern instances are made by some *exploratory process* and *selected* by patterns. 93
12.4.4 The connection of pattern instances is *influenced* by patterns. 93
12.5 The general idea is the same for the different pattern instance construction methods. 94
13 Conclusion 95
14 Bibliography 98

Table of Figures

Figure 1: A Pattern Instance 10
Figure 2: The Hierarchy 12
Figure 3: Patterns and the Hierarchy 13
Figure 4: The pattern instances of a pattern are generated and described by the pattern specification. 14
Figure 5: Pattern instances of the same pattern can be connected differently. 16
Figure 6: Roles of the logic specification and construction specification 17
Figure 7: Statistics Application 28
Figure 8: Convective Delusion 31

List of Abbreviations

AI	artificial intelligence
BERP	basic, exploratory relevance process
EF	evaluation function
EFS	evaluation function score
ERP	exploratory relevance process
RMP	relevance measurement process

1 Introduction

This article is the fifteenth in a series about artificial intelligence (AI) and how our own minds might work. For anyone who has not read all the previous articles to date, this article is an ideal starting point, because it gives a complete description of the cognitive model that has been developed so far. Reading through all the previous articles would be time consuming, and the reader would be exposed to repetition and false starts, as I have introduced features of the system and then changed them later. This article brings everything together, as it currently stands, in a single article. I should be clear that this article has been mainly made by copying parts of the previous articles, but I have also ordered things for easier understanding, omitted obsolete text and edited as appropriate. I am not trying to be original in this article, but merely trying to arrange the previous work in a more easily understood form. This article *is* somewhat long, but it deals with a subject with very wide scope and, while being equivalent to most of the contents of the first fourteen articles in the series, it is only about as long as the first four articles combined and is therefore an improvement. I may write a summary later.

The previous articles have been used to develop a model of cognition, and approach to AI, and discuss some general philosophical issues that relate to this.

The cognitive model involves a hierarchy based on *patterns*, which are sets of *pattern instances*.[1] The system is not there just to model the world: It needs to plan actions. Planning is provided by the *action selection process*, which involves using the modeling system to model the system itself to plan its behavior.[2]

The hierarchy needs to be *relevant*, which means including the pattern instances that represent those features of the world which are most useful in planning actions and excluding others. This requires a way of measuring the relevance of pattern instances, and this is provided by the *relevance measurement process* (RMP).[3] The RMP starts with particular pattern instances, used in the action selection process and corresponding to future inputs, being assigned relevance and this relevance is then back-propagated so that other pattern instances are assigned relevance. The relevance values are used in a *basic, exploratory relevance process* (BERP), which involves continual removal of low-relevance pattern instances, while the hierarchy is extended from the pattern instances

---

[1] Almond, P., 2010. *An Attempt to Generalize AI - Part 1: The Modeling System*. [Online] paul-almond.com. http://www.paul-almond.com/AI01.pdf or http://www.paul-almond.com/AI01.doc.
Almond, P., 2010. *An Attempt to Generalize AI - Part 5: A Completely Probabilistic Hierarchy*. [Online] paul-almond.com. http://www.paul-almond.com/AI05.pdf or http://www.paul-almond.com/AI05.doc. pp.11-20. (Part 5 supersedes the description in Part 1, which referred to the (now obsolete) concept of "fixing".

[2] Almond, P., 2010. *An Attempt to Generalize AI - Part 2: Planning and Actions*. [Online] paul-almond.com. http://www.paul-almond.com/AI02.pdf or http://www.paul-almond.com/AI02.doc.

[3] Almond, P., 2010. *An Attempt to Generalize AI - Part 6: Measuring Relevance*. [Online] paul-almond.com. http://www.paul-almond.com/AI06.pdf or http://www.paul-almond.com/AI06.doc.

that remain.[4] "Forgetting" – the removal of obsolete pattern instances – is also provided by the BERP.[5]

There are various ways in which the sophistication of the BERP might be increased, giving an improved *exploratory relevance process* (ERP).[6]

Relevance can also be provided by *reflexive outputs*: special outputs which, while made in the same way as other outputs, are directed inwards, acting on the hierarchy itself.[7]

[4] Almond, P., 2010. *An Attempt to Generalize AI - Part 7: A Basic, Exploratory Relevance Process.* [Online] paul-almond.com. http://www.paul-almond.com/AI07.pdf or http://www.paul-almond.com/AI07.doc.

[5] Almond, P., 2010. *An Attempt to Generalize AI – Part 8: Forgetting as Part of the Exploratory Relevance Process*. [Online] paul-almond.com. http://www.paul-almond.com/AI08.pdf or http://www.paul-almond.com/AI08.doc.

[6] Almond, P., 2010. *An Attempt to Generalize AI - Part 9: Improving the Exploratory Relevance Process*. [Online] paul-almond.com. http://www.paul-almond.com/AI09.pdf or http://www.paul-almond.com/AI09.doc.

[7] Almond, P., 2010. *An Attempt to Generalize AI - Part 13: Reflexive Outputs*. [Online] paul-almond.com. http://www.paul-almond.com/AI13.pdf or http://www.paul-almond.com/AI13.doc.

2 The Basic Structure of the Hierarchy

2.1 Patterns and Pattern Instances

The system uses a hierarchical model, made of *pattern instances*.[8]

Each pattern instance is a simple, computational unit. It has a set of labeled *pattern inputs* (e.g. A, B, C, etc.). It follows a set of rules to generate a *pattern output* value of 0 or 1.[9] Each pattern input for a pattern instance is the pattern output from another pattern instance, which can (and usually will) be a pattern instance of a different pattern. (See Figure 1: A Pattern Instance, below.)

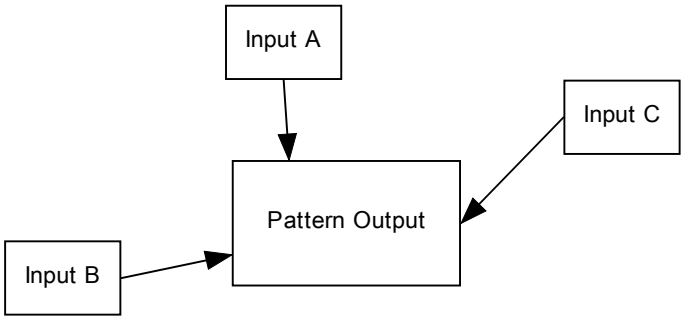


Figure 1: A Pattern Instance

The bottom level of the hierarchical model consists of special pattern instance values, corresponding to the history of values of the external inputs and outputs that the AI system has received or made previously, *with each pattern instance corresponding to an external input or output that has occurred at some specific time*. These can be considered to be arranged in arrays containing the input and output values that have occurred over a period of time. For example, if the AI system has a video camera, generating a 2D array of values, this could be represented in the bottom level of the hierarchy as a 3D array containing 2D images captured by the camera at different instants, each pixel state at some instant corresponding to a different pattern instance. External output values are represented as well. For example, if the system has an electrical motor, then output values sent to this motor will be stored in an array. If the system has multiple input/output devices, then there will be multiple arrays representing the history of their values. As well as representing the history of input/output events, these arrays represent the future: They are extended to contain

[8] Almond, P., 2010. *An Attempt to Generalize AI - Part 5: A Completely Probabilistic Hierarchy*. [Online] paul-almond.com. http://www.paul-almond.com/AI05.pdf or http://www.paul-almond.com/AI05.doc. pp.11-20.

[9] This could be generalized for values other than 0 or 1, but 0 or 1 will be assumed in this article for simplicity.

elements corresponding to inputs/outputs that have not yet occurred. Of course, the values of future inputs/outputs can only be known probabilistically, and that is what this is all about. In most respects, the inputs/outputs on the bottom level of the hierarchy are treated as if they were pattern instances: The only difference between these and the other pattern instances is that their values are set by external input/output events, rather than by the pattern instance's computation (See Figure 2: The Hierarchy, on page 12. In that diagram, the shade of blue is used to indicate the uncertainty in a pattern instance: Deep blue pattern instances are ones which are significantly dependent, directly or indirectly, on previous inputs/outputs, and about which there is little uncertainty, whereas white pattern instances are ones which are dependent on future inputs/outputs, about which little is known. As inputs/outputs occur, pattern instances gradually become more blue: They might be considered to be known about with "almost complete certainty", but there is no point where we draw any line: Pattern instances are known about with varying degrees of confidence, based on how much they depend on previous inputs/outputs.)

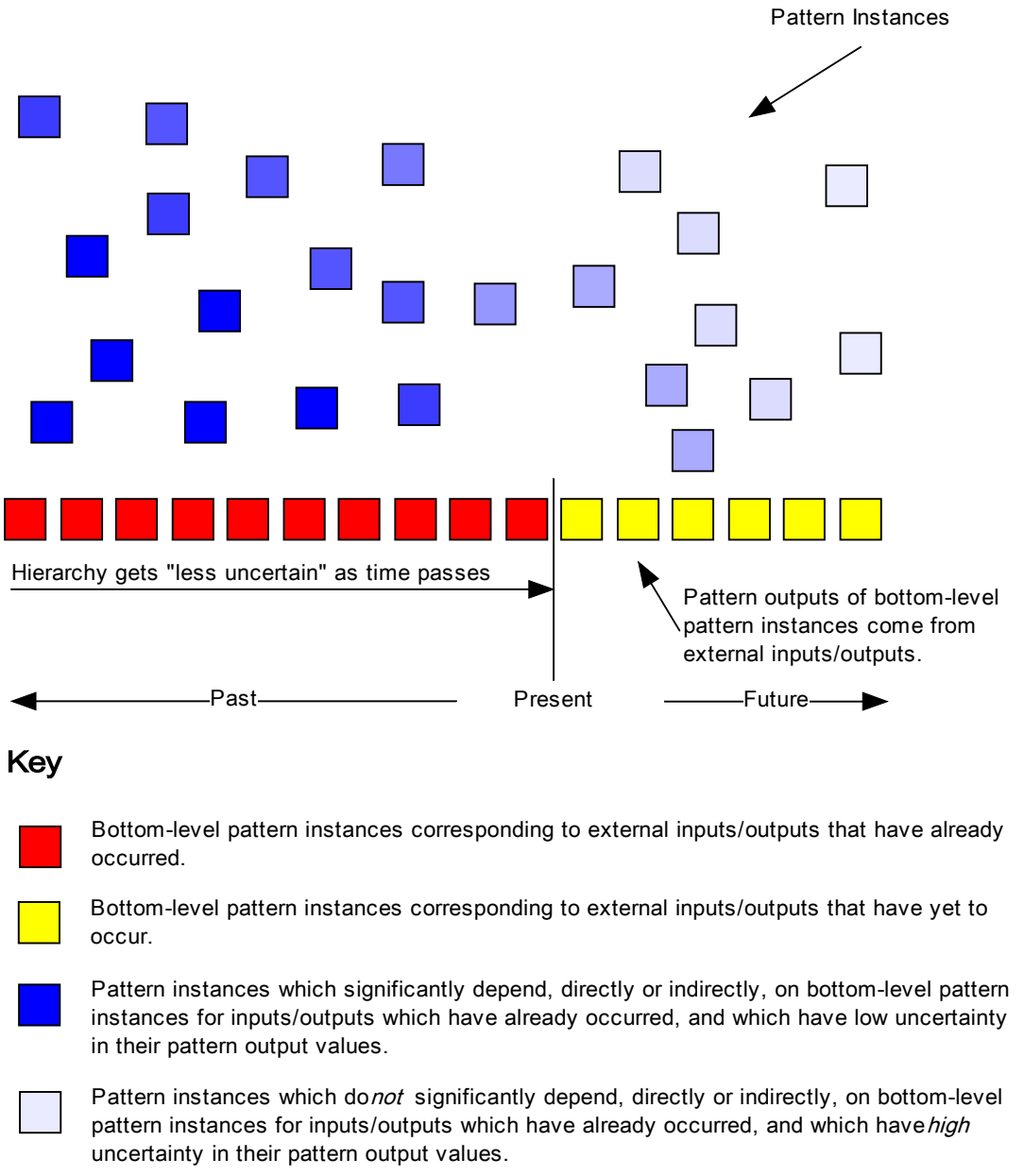


Figure 2: The Hierarchy

Pattern instances belong to *patterns*. A pattern is a set of related pattern instances. (See Figure 3: Patterns and the Hierarchy, on page 13.)

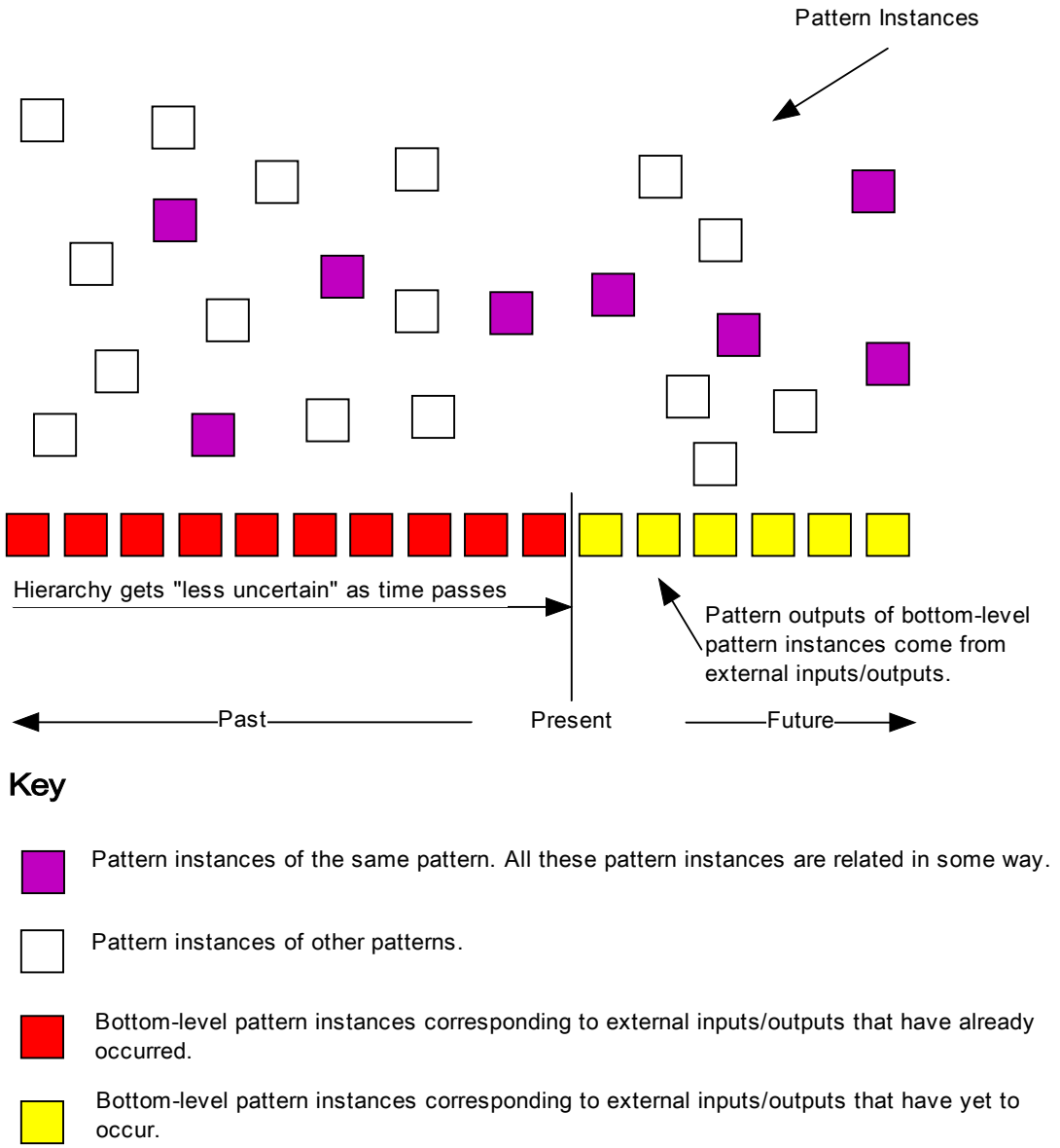


Figure 3: Patterns and the Hierarchy

2.2 Pattern Specifications

2.2.1 The General Idea of Pattern Specifications

Each pattern consists of a *pattern specification* and the associated set of pattern instances. The pattern specification describes the set of pattern instances and gives a set of rules about how they behave. Throughout most of this article, a placeholder view will be taken which is a *constructive* one in which the pattern specification generates the

pattern instances and effectively causes them to exist. (See Figure 4: The pattern instances of a pattern are generated and described by the pattern specification., below.)

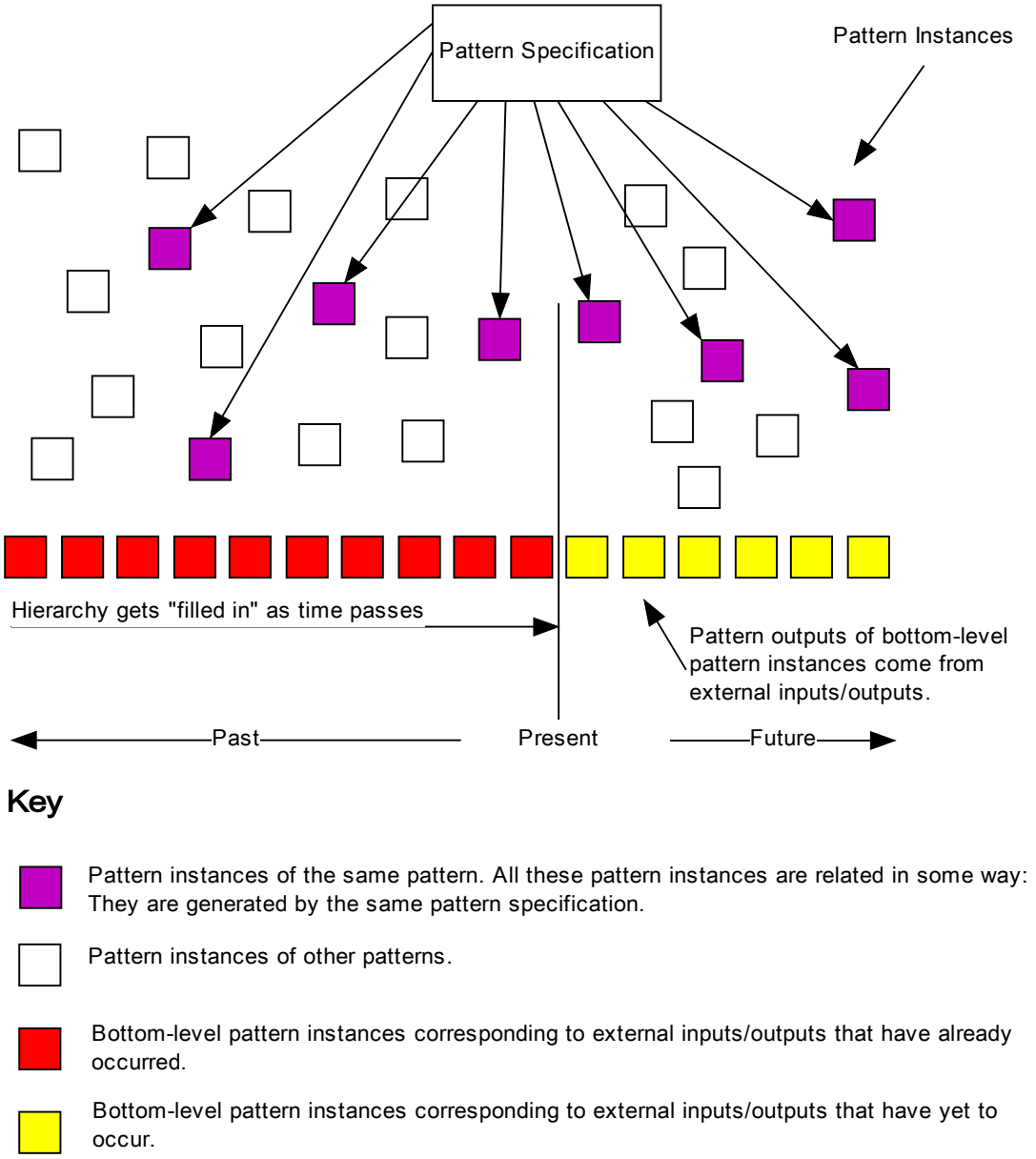


Figure 4: The pattern instances of a pattern are generated and described by the pattern specification.

2.2.2 The Logic Specification of a Pattern

The rules used by a pattern instance to generate its pattern outputs from its labeled pattern inputs are given in the *logic specification*, which is part of the pattern specification for that pattern. The rules relate pattern inputs, described using their labels, to a pattern output for a pattern instance and they are the same for all the pattern instances of that pattern. What is different for each pattern instance in a pattern is that the labeled inputs are being obtained from different pattern outputs in the hierarchy.

Example

Suppose that for some pattern, the pattern specification states that each pattern instance has three inputs, A, B and C. The logic specification gives rules describing how labelled pattern input values relate to a pattern output value. One of these rules is:

"If Pattern Input A = 0, Pattern Input B = 1 and Pattern Input C = 0 then make the pattern output = 1."

What this rule does not state is where pattern inputs A, B and C are coming from. Each of these will be obtained from the pattern output of another pattern instance, but it will be different for each pattern instance. Each of the pattern instances of the same pattern will get its pattern inputs A, B and C from the pattern outputs of different pattern instances in the hierarchy.

One way of thinking of this is as follows:

A pattern is a set of pattern instances. Each pattern instance is like a microchip. All the microchips (pattern instances) for a pattern are the same. Each microchip has a number of inputs (A, B, C, etc.) and uses some logic (such as a truth table) to generate an output. The logic is the same for all the microchips of a pattern. Each microchip has its inputs connected to the outputs of other microchips (which can be those of different patterns). Although the microchips for a pattern all contain the same logic (such as the same truth table), what is different is the way their inputs are connected to the hierarchy: Each microchip has its inputs connected to different outputs of other microchips, so although any two microchips use the same logic to determine what effect their pattern inputs A and B have on their outputs, their inputs A and B are wired to the outputs of completely different microchips.

The pattern consists then of a set of pattern instances, all following the same logic to generate a pattern output value from pattern inputs, but each having its inputs connected differently to other pattern output values in the hierarchy. (See Figure 5: Pattern instances of the same pattern can be connected differently., on page 16.)

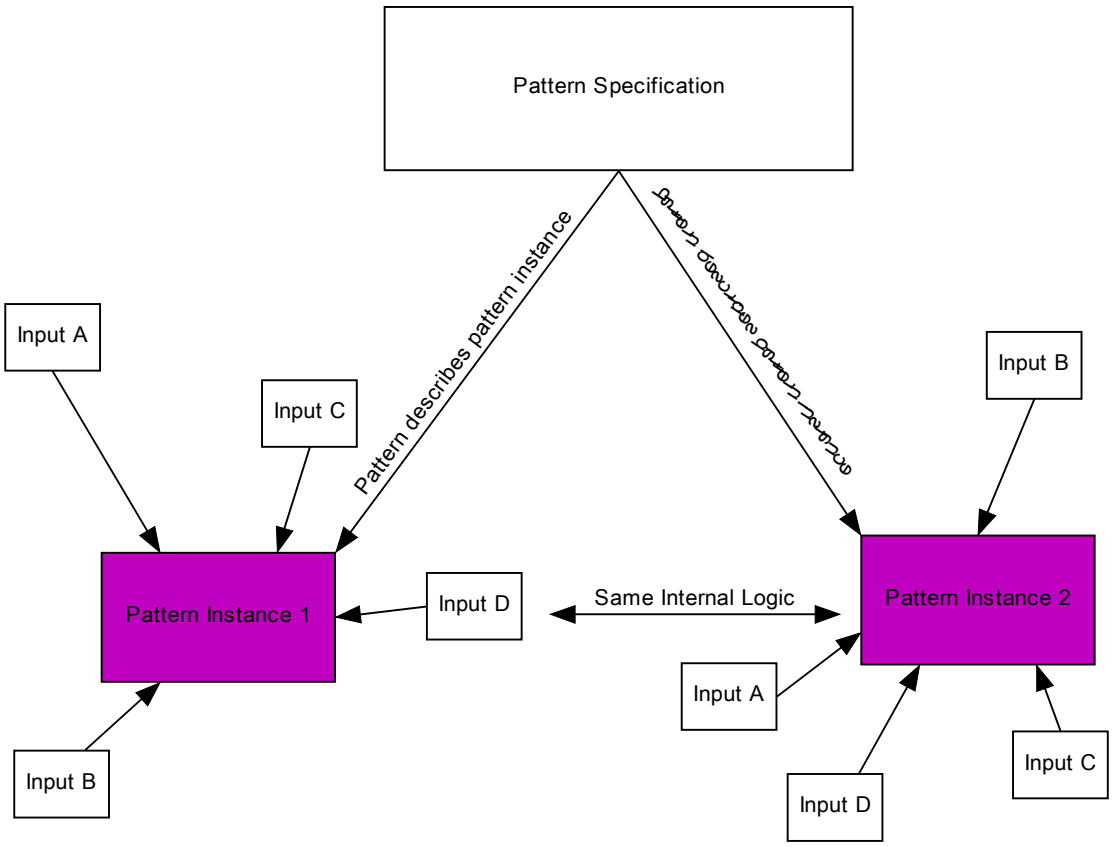


Figure 5: Pattern instances of the same pattern can be connected differently.

2.2.3 The Construction Specification of a Pattern

What needs to be dealt with now is how this set of pattern instances is generated for a pattern. How is it decided what pattern instances connect their inputs to? This is also controlled by the pattern specification. The pattern specification contains a *construction specification*. The construction specification for a pattern *describes the set of pattern instances*. In other words, *the construction specification for a pattern specifies the criteria that a pattern instance has to meet, in terms of how it is "wired" into the hierarchy, to be a member of that pattern*.

This could work in a variety of ways for actually creating the pattern instances and connecting them to the hierarchy. As stated earlier in Section 2.2.1: The General Idea of Pattern Specifications, on page 13, for now, as a placeholder idea, we will take a view in which the relationship between patterns and pattern instances is a *constructive* one.

In this kind of view, each pattern *generates* its set of pattern instances. For each pattern instance, the construction specification determines the pattern outputs to which its labeled pattern inputs correspond. (See Figure 6: Roles of the logic specification and construction specification, on page 17.)

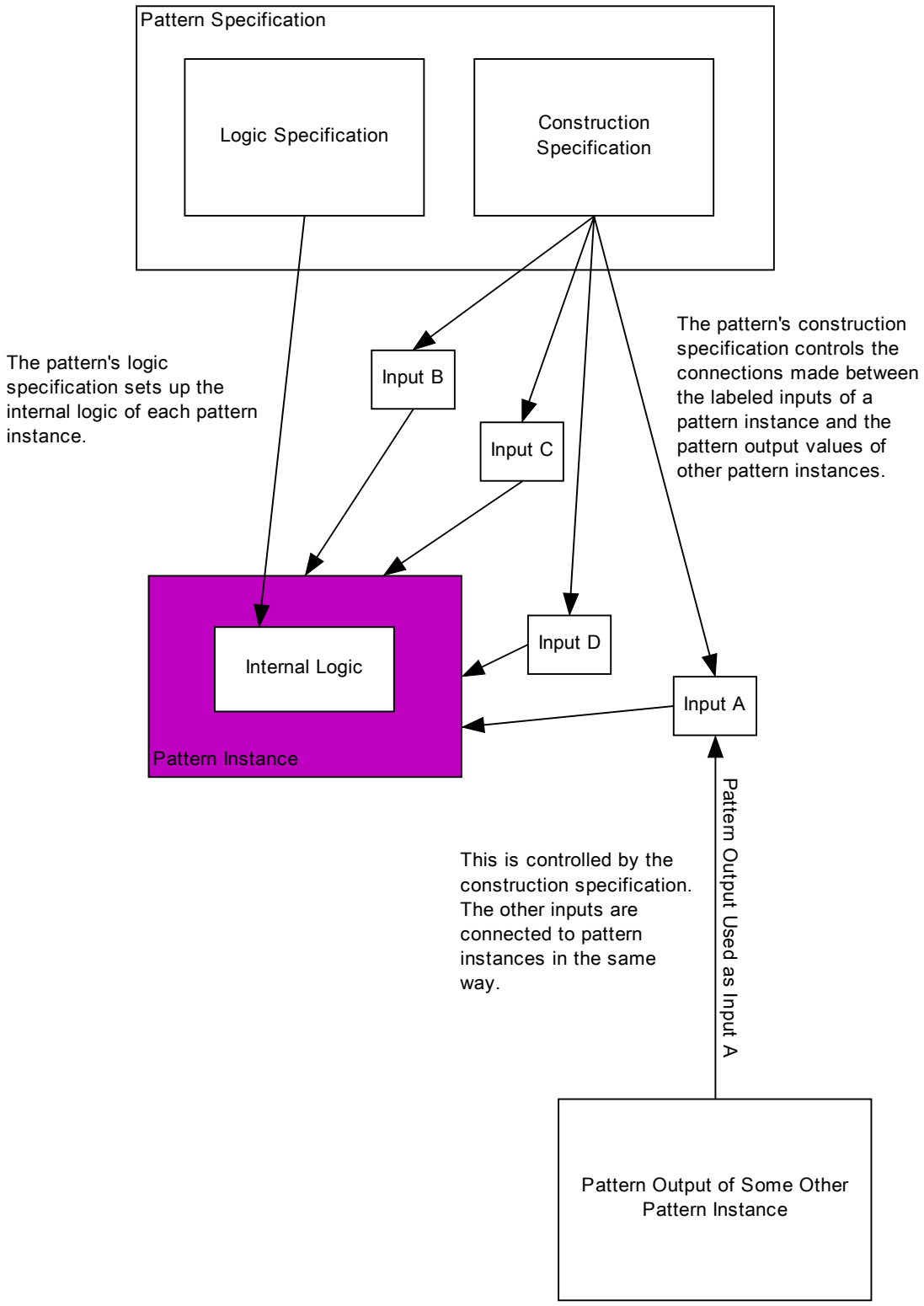


Figure 6: Roles of the logic specification and construction specification

To get an idea of how the construction specification works, imagine it "moving around" in the hierarchy, saying, "Make a pattern instance. Connect Input A to here. Connect Input B to here. Connect Input C to here. Make another pattern instance. Connect Input A to here. Connect Input B to here. Connect Input C to here…" and so on. Although the pattern instances are all connected in different ways, there is a statistical link between them all because the same thing is doing the connecting.

The whole point of this proposal is that the ontology should be as general as possible. The construction specification therefore needs encoding in a very general way. It needs to be able to distribute the pattern instances for a pattern throughout the hierarchy in very general ways. The only connection between the different pattern instances of a pattern is the logic specification and the fact they have all been set up by the same construction specification, but if the construction specification can be defined very generally, this connection could be very abstract.

The construction specification must be able to examine the "wiring" of the hierarchy when it is setting up a pattern instance. For example, when it has determined that some pattern instance that it is setting up is going to use a particular pattern output as Input A, it might follow the "wiring" from this pattern output to see what inputs that pattern instance uses, and it may look at one of these pattern instances to see what pattern instances use its output and so on.

The construction specification for a pattern might be considered being applied locally in the hierarchy, setting up pattern instances dependent on the local "wiring" of the hierarchy. An approach like this would have some similarities with the way in which the DNA of a cell controls how a cell is made, taking account of other nearby cells and signaling.[10] This idea might be extended if the construction specification is applied "locally" in the parts of a hierarchy near an existing pattern instance, so that a new pattern instance is generated from an existing one, and "wired" to the hierarchy in a way that takes account of the local "wiring" of the hierarchy.[11]

At this stage, I am unsure about how to implement the construction specification, but we need some way of thinking about it for now. The need for generality suggests that we think of the construction specification for a pattern as being a computer program, in some general purpose programming language. The language in which the construction specification is expressed should make it possible for the construction specification

[10] I am not suggesting a strong association here though. This is just an analogy. I am not suggesting that DNA serves as the construction specification in humans.

[11] My use of the word "local" might be questioned, given that I have said that we are trying to get away from geometry; however, we are trying to get away from a "strong" geometrical analogy, in which things have rigidly defined coordinates. There is a sense in which a "weak" kind of geometry can be useful and this is discussed later, in Section 7.2: Weak Geometry, on page 45.

program to “read” the structure of the hierarchy, determine what pattern instances are connected to what, and make decisions based on this.

As I stated earlier, this “constructive” view of the relationship between patterns and pattern instances is only a placeholder for now. The main requirement is that patterns specify the criteria that pattern instances have to meet, in terms of how they are connected into the rest of the hierarchy, to be members of them. There are other possibilities for the relationship between patterns and pattern instances, and some of these are discussed later in Section 12: Pattern Instance Construction Alternatives, on page 91.

2.2.4 Generation of Pattern Specifications

I have not discussed, in detail, how the pattern specifications are generated. From where do the logic specifications and construction specifications come? For now I will say that trial and error will play a big part in this – and there could also be some kind of Darwinian aspect. This may seem to be asking a lot of a trial and error process, given that the patterns are supposed to provide a system with intelligence. It is important to realize, however, that patterns are not expected to be very intelligent by themselves. A pattern is not required to solve any complex problem. A pattern is only required to expose a statistically interesting relationship within its set of pattern instances.

3 The Conceptual and Actual Hierarchies and the Atemporal Nature of the Conceptual Hierarchy

The hierarchy can be considered in two different ways, as the *conceptual hierarchy* and the *actual hierarchy*.[12]

3.1 The Conceptual Hierarchy

The conceptual hierarchy consists of all possible pattern instances: It is therefore infinite and cannot be realized in a real computer.

The conceptual hierarchy should be viewed from the imaginary perspective of an observer who knows its entire history. This means that every external input and output that occurs throughout its history is known. It is important to understand that, viewed this way, *the conceptual hierarchy is atemporal*. Although each pattern instance has logic that controls its state, that state is determined completely by specific inputs and outputs occurring at particular times. In the case of a bottom-level pattern instance, the state is the value of the relevant input/output occurring at some time. In the case of a higher-level pattern instance, the state will ultimately be dependent on bottom-level pattern instances, and so will still be completely determined by pattern inputs and outputs occurring at particular times. Although a pattern instance contains logic to set its state, its state never changes over time: The logic simply defines its state in terms of inputs/outputs occurring at specific times, or in terms of the states of other pattern instances which are themselves dependent on inputs/outputs occurring at specific times. Each pattern instance has only a single state. Pattern instances are, therefore, basically unchanging and *atemporal*.

3.2 The Actual Hierarchy

The actual hierarchy is a reduced version of the conceptual hierarchy that actually exists in a computer. The actual hierarchy only contains *some* of the possible pattern instances of the conceptual hierarchy and, ideally, these are selected according to *relevance*, as discussed later in Section 8: Relevance, on page 47.

Whereas the conceptual hierarchy can be viewed as timeless, from the perspective of an observer who can see its entire history, with the actual hierarchy in a computer there is

[12] Almond, P., 2010. *An Attempt to Generalize AI - Part 4: Modeling Efficiency*. [Online] paul-almond.com. http://www.paul-almond.com/AI04.pdf or http://www.paul-almond.com/AI04.doc. p.13.
Almond, P., 2010. *An Attempt to Generalize AI - Part 5: A Completely Probabilistic Hierarchy*. [Online] paul-almond.com. http://www.paul-almond.com/AI05.pdf or http://www.paul-almond.com/AI05.doc. pp.8-9.

only full information about the inputs and outputs that have actually occurred at any time, and any pattern instances that depend completely on them. Pattern instances corresponding to future inputs/outputs, or pattern instances dependent on them, can only be described *probabilistically*, so each pattern instance will have a probability value. As time passes, and inputs and outputs occur, the probability values of pattern instances will change. It is important to note that this does not really mean that the hierarchy itself is changing. Rather, each pattern instance just has a pattern output value that we do not necessarily know and it is the information that we have about it that is changing.

Probability values are assigned to the pattern instances in the actual hierarchy by probabilistic propagation processes, which will now be described.

4 Probabilistic Propagation in the Actual Hierarchy

4.1 The General Idea of Probabilistic Propagation in the Actual Hierarchy

Each pattern instance has a pattern output value which is generated from the pattern logic for that pattern and the pattern instance's pattern inputs. The pattern output value for a pattern instance never changes; however it is not known with certainty until all the inputs and outputs on which it depends have occurred. Pattern instances are assigned probabilities indicating the degree of knowledge about their pattern output values. Probabilistic information is propagated through the hierarchy from the pattern instances corresponding to inputs/outputs which have already occurred.[13]

Propagation of probabilistic information involves the following operations being performed on the hierarchy of pattern instances.

- Logic application
- Statistics generation
- Statistics application

Each of these will now be described.

4.2 Logic Application

This is the most obvious process. Logic application can be applied upwards and downwards, and we will first consider the more important case: logic application working upwards.

Each pattern has a logic specification describing how the pattern output value of any its pattern instances is generated, based on its pattern inputs. Upward logic application is the process of using what we know about the pattern inputs of each pattern instance, together with the logic specification (of the pattern to which that pattern instance belongs) to generate a probabilistically described pattern output for that pattern instance. Upward logic application can be thought of as applying truth tables with probabilistically described inputs to obtain probabilistically described outputs.[14]

---

[13] Almond, P., 2010. *An Attempt to Generalize AI - Part 5: A Completely Probabilistic Hierarchy*. [Online] paul-almond.com. http://www.paul-almond.com/AI05.pdf or http://www.paul-almond.com/AI05.doc. pp.21-29.

[14] It should be noted, however, that logic specifications do not *have* to use truth tables.

If nothing is known about the pattern inputs of a pattern instance, then upward logic application will not add any information about it; however there are the external inputs/outputs that have already occurred. Upward logic application will therefore start at the bottom level of the hierarchy. Each element in the arrays of inputs/outputs to/from the AI system is treated as a pattern instance. The pattern instances corresponding directly to past inputs/outputs to/from the AI system *are* known about with certainty. This allows pattern outputs of pattern instances which receive their pattern inputs solely from the past inputs/outputs to be computed. This in turn makes the pattern outputs of more pattern instances known, allowing more pattern outputs to be calculated, and so on. Pattern outputs are calculated for pattern instances working "up" the hierarchy, increasing abstraction. It should be easy to imagine this working when the pattern inputs of a pattern are known with certainty, but an important point is that *the process continues even if there is not full information about the pattern instances of patterns*. If the pattern inputs of a pattern instance are only known about probabilistically, this information is still used to generate a probabilistic pattern output value for that pattern instance. This can work even if there are some pattern inputs of a pattern instance about which nothing is known – if they have probabilities of 0.5.

Example

To see how pattern logic can be applied probabilistically, suppose we knew the values of the pattern inputs to a pattern instance with complete certainty, and they were as follows.

Pattern Input A=1, Pattern Input B=0, Pattern Input C=1

The logic specification for this pattern tells us that the pattern output for this set of pattern inputs is 1, so we would assign a pattern output value of 1 to this pattern instance.

Now, suppose we did not have complete certainty, but were *almost* sure that the pattern outputs were as above. We have a probability of each pattern input being 1, as follows.

P(A=1)=0.97, P(B=1)=0.03, P(C=1)=0.98

This situation is scarcely different from the first one, in which we determined that the pattern output was 1. We can still tell, from this, that there is a *high probability* of the pattern output being 1. Furthermore, there is no cut-off point beyond which we cannot do this. If the probability values contained less information – if they were closer to 0.5 – they would still give us some probabilistic information about the pattern output.

Assigning a probability to the pattern output of a pattern instance, based on the pattern's logic and probabilities for the inputs, should be quite simple. We might do it just by determining the probability of each possible combination of inputs, and looking at the associated output.[15]

Example

Suppose a pattern instance has two inputs, A and B, and the pattern's logic takes the form of the following truth table.

Pattern Input A	Pattern Input B	Pattern Output
0	0	1
0	1	0
1	0	1
1	1	0

We have probabilities for the pattern inputs as follows:

P(A=1)=0.37, P(B=1)=0.65

We can use this to calculate that:

P(A=0,B=0)	=(1-0.37)(1-0.65)	=0.4095 (and Pattern Output=1)
P(A=0,B=1)	=(1-0.37)(0.65)	=0.2205 (and Pattern Output=0)
P(A=1,B=0)	=(1-0.37)(1-0.65)	=0.1295 (and Pattern Output=1)
P(A=1,B=1)	=(1-0.37)(1-0.65)	=0.2405 (and Pattern Output=0)
So, P(Pattern Output=1)		=0.4095 + 0.1295
		=**0.539**

If a pattern instance is ever assigned a probability of 0 or 1, then this means that it has become completely dependent, directly or indirectly, on previous inputs/outputs and the probability will not change again.

As well as upward logic application, downward logic application can also be used: Probabilistic information about the pattern output of a pattern instance can be used with the pattern's pattern logic to generate information about its pattern inputs.

[15] If a pattern instance has many pattern inputs, we may need to cut corners here.

Logic application will not provide much information about pattern instances that are mainly dependent on future inputs/outputs. Logic application is not the main operation in propagation: It is a means to an end and its main purpose is to prepare data to be analyzed in *statistics generation*, described in 4.3, below.

4.3 Statistics Generation

Statistics generation involves acquiring statistical information about the frequency with which each of the possible combinations of labeled inputs is expected to occur with the pattern inputs for each pattern. Importantly, reality, and its effects on the AI system, has an influence here. For the pattern instances of a particular pattern, some combinations of pattern inputs may be more common than others, because of the particular patterns of external inputs/outputs which occur and their interaction with the way that the pattern instances of that pattern are distributed throughout the hierarchy.

Statistics generation treats all the pattern instances of a pattern as belonging to the same statistical set and generates statistics for that pattern, based on the information about pattern inputs and pattern outputs which has previously been generated by logic application, as previously described in 4.2.

For each pattern, the frequency with which each particular combination of labeled pattern inputs is expected to occur is determined, based on the combinations of probabilities which have occurred so far for pattern instances of that pattern in logic application.

One way in which we can do this is to keep a running count of the relative frequency with which each combination of pattern inputs for the pattern instances of a particular is expected to occur, and to keep this updated based on the probability values which result during logic application.

Example

Suppose we had the particular pattern instance and combination of pattern input probabilities being considered in the previous example.

P(A=0,B=0) =(1-0.37)(1-0.65) =0.4095
P(A=0,B=1) =(1-0.37)(0.65) =0.2205
P(A=1,B=0) =(1-0.37)(1-0.65) =0.1295
P(A=1,B=1) =(1-0.37)(1-0.65) =0.2405

This is suggesting that A=0,B=0 might occur often for pattern instances of this pattern, whereas A=1,B=0 might occur rarely. Of course, this is only one pattern instance. We must take account of many pattern instances of this pattern, with pattern input probabilities that have been set by logic application, to obtain a full picture.

For this pattern's statistics (which will affect any pattern instances of this pattern), we would:

add 0.4095 to the relative frequency with which A=0,B=0 is expected to occur.
add 0.2205 to the relative frequency with which A=0,B=1 is expected to occur.
add 0.1295 to the relative frequency with which A=1,B=0 is expected to occur.
add 0.2405 to the relative frequency with which A=1,B=1 is expected to occur.

The purpose of statistics generation is to generate statistical data for further extending the hierarchy probabilistically in *statistics application*, as described in 4.4, below.

4.4 Statistics Application

Statistics application involves using the pattern instances in the hierarchy, together with the statistics from statistics generation (Section 4.3: Statistics Generation, on page 25), to fill in the hierarchy with probability values that reduce the uncertainty in the states of its pattern instances. The idea of statistics application is that it can fill in probability values in parts of the hierarchy about which relatively little is known, and on which logic application will not have had much effect: *those pattern instances with pattern outputs which are mainly determined by inputs/outputs that have yet to occur*.

Logic application (Section 4.2: Logic Application, on page 22) will have partially filled in the hierarchy with probability values, but logic application is not much use as a predictive process: It will only have made any significant difference to pattern instances that depend mainly on previous inputs/outputs with known values. Pattern instances that are mainly dependent, directly or indirectly, on future inputs/outputs will be mostly unaffected by logic application.

Some pattern instances will depend significantly on previous, known inputs/outputs and partly on future inputs/outputs. Statistics application allows these pattern instances to have probabilities assigned to their outputs, because the partial information we have about their pattern inputs from logic application, can be used, with the statistics for that pattern, to determine the probability of each possible combination of pattern inputs.

This in turn allows probabilities to be assigned to "higher up" pattern instances that are more dependent on future inputs/outputs, and so on.[16] The process works both upwards, generating probability values for high-level, "abstract" pattern instances in the hierarchy, and downwards, ultimately "filling in" probability values on the bottom level of the hierarchy, for the pattern instances corresponding to the arrays of future inputs/outputs. (See Figure 7: Statistics Application, on page 28.)

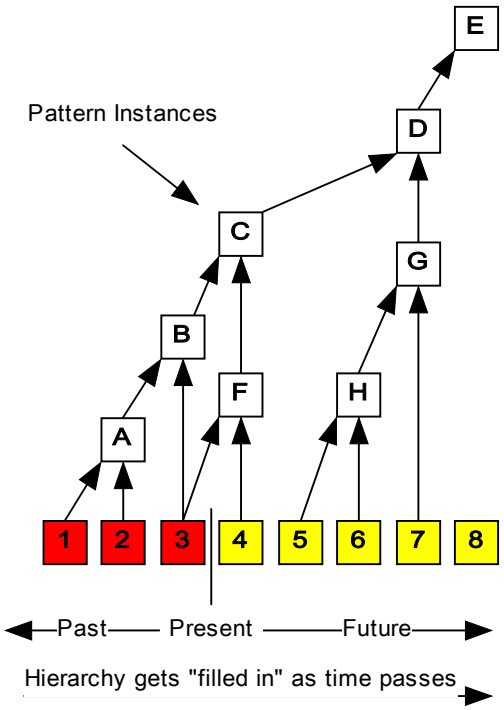
In Figure 7, pattern instances 1, 2 and 3 correspond to inputs/outputs that have already occurred, while 4, 5, 6, 7 and 8 have yet to occur, so the diagram represents a "snapshot" of part of the hierarchy, just after the input/output corresponding to Pattern Instance 3 has occurred. Pattern Instance A depends on pattern inputs 1 and 2, which are known, so its pattern output can be determined with a high degree of confidence: In fact, it could be determined with complete confidence, going off this, but this diagram is ignoring the possibility of pattern instances with missing pattern inputs. The output for B can similarly be determined by logic application with high confidence.

We have high confidence about B, but logic application will not tell us much about C or F, because they depend more on inputs/outputs that have yet to occur. However, we at least have some information about the pattern input being provided to C by B, and this means that we can use statistics application to make a reasonable, probabilistic prediction for C, based on the combinations of inputs that we expect, from experience of previous logic application, for pattern instances of that pattern. We can also use statistics application with our partial information about the pattern input to F to make a reasonable, probabilistic prediction for F, which in turn improves our ability to use statistics application with C.

When we have better probabilistic information about B, this in turn is telling us something about one of the pattern inputs for D, so statistics application enables us to make a prediction about D, which in turn allows us to make a prediction for E. Importantly, we are now making predictions for pattern instances which depend on inputs/outputs further in the future. Statistics application does not just work upwards: It

[16] In Almond, P., 2010. *An Attempt to Generalize AI - Part 1: The Modeling System*, I mentioned (page 9) assigning probabilities this way to pattern instances that are *only* dependent on future inputs/outputs. I should have actually said "more dependent", "almost completely dependent" or something similar. The idea is that as we work up the hierarchy we will encounter pattern instances with direct or indirect dependency that "spans" an increasing chronological range of inputs/outputs. When we work down the hierarchy, however, we can assign probabilities to pattern instances that are dependent only on future inputs/outputs.

works downwards, telling us about D, F, G and H, ultimately allowing us to make predictions about bottom-level pattern inputs 4, 5, 6 and 7, meaning that we are predicting external inputs/outputs.



Key

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.

Other pattern instances.

Figure 7: Statistics Application

4.5 Notes on Logic Application, Statistics Generation and Statistics Application

Logic application, statistics generation and statistics application are ongoing processes; however they should not be mixed together incorrectly. Statistics generation is only intended to use information that has been generated in logic application.

Logic application will only tend to work well with pattern instances that are strongly dependent on previous inputs/outputs. Statistics application will only tend to work

better than logic application with pattern instances that are more dependent on future, external inputs/outputs, or which are dependent on missing pattern inputs. This does not mean, however, that each operation needs to be restricted to a specific part of the hierarchy. Logic application can be performed, in principle, on the entire hierarchy; however for pattern instances which are not strongly dependent on previous inputs/outputs, it will not generate much more information than is already known about them.

Likewise statistics generation does not need to be applied to any particular group of pattern instances: Information for statistics generation can be obtained from any pattern instance in the hierarchy. Only pattern instances about which logic application has generated a significant amount of information will have any significant effect on statistics generation.

Statistics application can occur with any pattern instances, but when applied to ones which are strongly dependent on previous inputs/outputs it will be ineffective: Logic application will have already told us more in such cases. When either process is applied, it is only allowed to affect the information stored about a pattern instance if it increases that information.

There is the need to ensure that logic application and statistics application work as separate processes, even though they can be applied to the same pattern instances, so that logic application only involves information from external inputs/outputs and probabilities produced in logic application itself, and to ensure that statistics generation only uses information created by logic application.

One way of separating the information produced by logic application and statistics application is to store two types of information for each pattern instance: the information generated by logic application, and the information that represents our state of knowledge about the pattern instance after statistics application. For example, a pattern instance may have a probability for its pattern output which has been generated by logic application, and a second probability for the pattern output which represents what is known about the pattern output after any statistics application process. Before statistics application, or if statistics application has occurred, but has not succeed in generating a probability with less uncertainty, the two probability values would be the same.[17]

[17] With regard to statistics generation, it should be noted that the information from a pattern instance could be obtained once for any single process of logic application, as soon as logic application generates it, anyway.

Example

Suppose logic application indicates a probability of 0.35 that a particular pattern instance will have a pattern output of 1, and statistics application has not been applied yet. The following information might be stored for the pattern instance.

Logic Application Probability =0.35
Resultant Probability =0.35

If statistics application indicates that the probability is 0.87 then this indicates less uncertainty than the existing resultant probability of 0.35. The probabilities would be set as follows.

Logic Application Probability =0.35
Resultant Probability =0.87

and the pattern instance would be regarded as having a probability of 0.87. The probability generated by logic application, 0.35, is still retained, however, for use in any logic application or statistics generation process.

(There are other ways of doing this.)

4.6 Convective Delusion

Logic application and statistics application will together propagate probabilities through the hierarchy. In any real system, we must ensure that this is not done in a way that causes what I will call "convective delusion".[18] Convective delusion occurs when some low-level belief about reality reinforces a high-level belief, which in turn reinforces the original, low-level belief, and so on, so that there is a positive feedback loop, going up and down the hierarchy, in which beliefs are reinforced tautologically.

In simple, human terms, we might imagine convective delusion working as follows.

Suppose you walk into some poorly lit, outdoor place at night. You cannot see it clearly. You see some shapes that are hard to recognize, but you think they might be animals, though you are far from certain about this. This suggests, just a bit, that you may be in a zoo. The slight possibility that you may be in a zoo makes it just a bit more likely that the unknown shapes are animals. The increased chance that you are looking at animals makes it more likely that you are in a zoo, making it more likely that you are looking at animals, in turn making it more likely that you are in a zoo, and so on. The beliefs that

[18] Almond, P., 2010. *An Attempt to Generalize AI - Part 5: A Completely Probabilistic Hierarchy*. [Online] paul-almond.com. http://www.paul-almond.com/AI05.pdf or http://www.paul-almond.com/AI05.doc. pp.31-32.

you are seeing animals and that you are in a zoo are reinforcing each other. If this continues, you will eventually be sure that you are seeing animals and that you are in a zoo, and this conviction will have come from nowhere. (See Figure 8: Convective Delusion, below.)

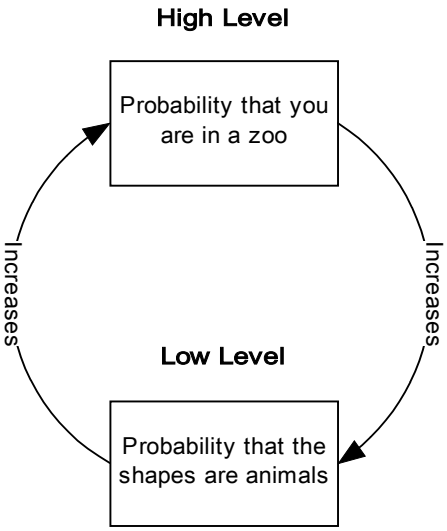


Figure 8: Convective Delusion

With the hierarchy discussed here, the equivalent of this would be a low-level probability contributing to the increase of a high-level probability which in turn increases the low-level probability, which increases the high-level probability, etc. We should not want this to happen.

A description of how the hierarchy can model reality has now been given, though we will have to return to this later to deal with the issue of relevance (in Section 8: Relevance, on page 47). I will now discuss how the hierarchical model can be used to plan actions.

5 Planning of Actions

5.1 The General Idea Behind Planning of Actions

If we can predict future inputs/outputs, based on previous ones, planning becomes simple. Suppose we have some output to make, and we need to decide whether to make an output of 0 or 1. We can try each output value in turn and tell the hierarchy the output has actually happened with that value by setting the appropriate pattern instance – even though it has not yet happened. We can then look at the probabilistic predictions for future inputs/outputs to see the kind of situation in which the system is in the future, and how desirable it is. We can then select the output value which causes the most desirable situation to be predicted, and actually make the output with that value.[19]

To find out the desirability of a situation, based on predictions of inputs/outputs to the system, we could look at some inputs that correspond to desirable things, such as inputs showing that the system is receiving energy, for example, and some inputs that correspond to undesirable things; for example, inputs corresponding to threats of damage, such as excessive temperature, or pressure. A good way of doing this is to use an evaluation function (EF). The evaluation function examines the inputs/outputs occurring at approximately some instant of time and inputs them into an algorithm to generate an evaluation function score (EFS). By looking at the predictions for inputs/outputs, we could predict what the EFS would be at some time in the future. However, we can do this in a more sophisticated way. If we continually compute the EFS, and provide it to the hierarchy as one or more inputs, by looking at a prediction for the inputs in the future corresponding to the EFS, we would be obtaining the predicted EFS.

Although an approach to planning like this does involve some processing for planning that is external to the system, that processing is minimal. Most of the processing for planning is actually taking place in the modeling system itself, where the system's future behavior will be modeled along with everything else. This has implications for the nature of consciousness and the "self" that are discussed later in Section 6: The "Self, on page 42.

[19] Almond, P., 2010. An Attempt to Generalize AI - Part 2: Planning and Actions. [Online] *paul-almond.com*. http://www.paul-almond.com/AI02.pdf or http://www.paul-almond.com/AI02.doc.

5.2 The Action Selection Process

5.2.1 Computation of Evaluation Function Score

The evaluation function (EF) is continually used to compute the evaluation function score (EFS). Each time the EFS is computed, this is based on inputs which have occurred around the time that the score is computed.

e.g.

At t=0.01s, the EFS is computed based on inputs occurring just before t=0.01s.
At t=0.02s, the EFS is computed based on inputs occurring just before t=0.02s.
At t=0.03s, the EFS is computed based on inputs occurring just before t=0.03s
and so on.

5.2.2 Input of Evaluation Function Score

Each time an EFS is computed, it is encoded as input data to the hierarchy and one or more pattern instances corresponding to external inputs at the bottom of the hierarchy are "fixed": They are permanently assigned pattern output values encoding the most recently computed EFS.[20]

5.2.3 Selecting Actions

Suppose some output, $Output_n$, has to be made at time, $t=t_1$. The output has two possible values: 0 or 1.

The expected effects of $Output_n=0$ are determined. To do this, the hierarchy is experimentally updated. The pattern instance corresponding to $Output_n$ is fixed such that $Output_n=0$. The effects of this are propagated through the hierarchy by logic application and statistics application. Probabilities for pattern instances dependent on future inputs/outputs will be affected and, ultimately, probabilities for pattern instances corresponding to future, external inputs will be affected. As the EFS values have been continually encoded as inputs, some of the pattern instances corresponding to future, external inputs will correspond to future input of the EFS. The probabilities for one or more inputs corresponding to future input of an EFS value are examined.

For some output, $Output_n$, occurring at time, $t=t_1$:

First, try $Output_n=0$:

[20] A way of imagining the input of the desirability score comes from the movie, *The Terminator*, and its sequels. In this, a robot had computer generated information overlaid on its visual field, so that it would see information, presumably generated by its own computer systems, superimposed on objects that it was seeing. This meant that information generated by the software in the system was going back into the system as an input.

Pretend that $Output_n$=0 has actually occurred. Set the pattern instance for $Output_n$ as if $Output_n$ has already occurred and $Output_n$=0.

Propagate the effects of this through the hierarchy using logical application and statistics application.

For one or more pattern instances corresponding to input of the EFS at some future time, $t=t_2$, obtain one or more probabilities from the bottom level of the hierarchy.

Use these probability values to obtain a predicted EFS at $t=t_2$. This expected EFS is an indication of the desirability of making $Output_n$ such that $Output_n$=0.

Reverse the changes just made to the hierarchy.

Then, try $Output_n$=1 using the same procedure as was just used for trying $Output_n$=0.

Compare the predicted EFS values for $Output_n$=0 and $Output_n$=1. The output value with the greater score is the more desirable output value.[21]

Actually make $Output_n$ occur with the most desirable output value. Assign the value of the pattern output for the pattern instance corresponding to $Output_n$ with the relevant value, to indicate that it has now occurred.

and that is it.[22]

5.3 How the Action Selection Process Works

5.3.1 The *real* planning process

Because the hierarchy predicts future inputs/outputs, it can be used to predict the system's future situation. We can use this to test the expected results of making an output with some value. It is as simple as it sounds: Tell the hierarchy that the output has occurred with some value (even though it has not yet occurred really), and then ask it how desirable the future situation seems.

The process as described above for selecting an output may seem to be the main process by which the system plans – and it may seem simplistic if it is supposed to capture all the complexity of human planning. Could it really reduce to something like that? What I will now say may seem strange. The process of action selection that I just described is not the "real" action selection process, but is just a means to an end. The

[21] This assumes that a higher score corresponds to greater desirability.

[22] Actually, there are some variations of this method, which I will not go into here, but what was just described is the main principle. I described this kind of approach to planning of actions previously, with more detail, in Almond, P., 2007. *Planning As Modelling in AI: A New Version*. [Online] paul-almond.com. http://www.paul-almond.com/PlanningAsModellingNew.pdf or http://www.paul-almond.com/PlanningAsModellingNew.doc.

real, deep action selection process actually occurs within the hierarchy itself – within the model. The system is using modeling for its really deep planning.

What are the grounds for thinking this?

First imagine that the system is controlling a car in a simple videogame: It has to drive the car along the road, moving left or right to stay on the road.

Imagine that the system has a history of playing this game competently. (Do not worry about how it got this history, but just pretend it has it.) Pattern instances corresponding to previous inputs/outputs in the hierarchy describe a history of moving the car correctly – going left when left was correct and right when right was correct.

Suppose we are *not* using the action selection process described previously, but instead we just look at the hierarchy's predictions of the value of an imminent output, and actually make the output with this value.

We want to know whether some imminent output, $Output_n$, should be "left" or "right". We look at the probability for the pattern instance corresponding to this output, and it tells us that it is more likely that the output will be "right". We select this as the output, so $Output_n$ actually occurs with $Output_n$=Right. We then tell the hierarchy that the output has occurred with this value, so the pattern instance for $Output_n$ has its pattern output value fixed with "Right".

What we just did would actually be a good way of selecting the output. The prediction for $Output_n$ was based on everything that has been happening in the past – on the history of inputs/outputs encoded in the hierarchy's fixed pattern instances – and this history is one of playing the game competently. The system's predictions will be based on the patterns relating all these inputs/outputs. Whatever patterns, relating inputs to outputs, correspond to playing the game competently, will determine the system's predictions of its own behavior. The system will predict future behavior for itself that continues the previous pattern of competent behavior relating inputs to outputs: The system will predict competent behavior for itself.

When inputs have indicated that the road is bending right, the system will have tended to have made outputs to move right, and when inputs have indicated that the road is bending left, the system will have tended to have made outputs to move left. Any predictions for future outputs will be based on a continuation of this pattern into the future. If inputs have just been received indicating that the road is bending right, the hierarchy will predict an output of "Right", because that is what has happened in the past. This example is only simple, but the same principle would apply if the pattern relating previous inputs and outputs were more complex.

All this means that, assuming a perfect modeling system and looking at things simplistically, if the system's pattern instances already contain a history of competent

behavior, basing the system's behavior just on predictions of imminent outputs, would simply continue this behavior.

5.3.2 Why the Action Selection Process is Needed

If what I just said is true, why is the action selection process I described needed at all? Why not just base all the system's behavior on predictions of what the hierarchy is going to do next? There are two problems with this idea.

5.3.2.1 Problem 1: Initially, there is no history of suitable behavior.

In the above discussion, I asked you to assume that the hierarchy contained a history of competent behavior, and not to worry about where this comes from. In reality, the system would not start with any such history: It would start with nothing. This would prevent it from behaving competently in the first place, with such a simplistic approach: It would need a history of competent behavior before it could behave competently to produce such a history. It would be in a catch-22 situation.[23]

5.3.2.2 Problem 2: Random drift would occur.

No probabilistic modeling system, like the one proposed for the hierarchy, will be perfect. The predictions of future behavior will have some error. The behavior that such a system produces would, at best, be *almost* as good as the behavior that would have been extrapolated from previous behavior by perfect modeling. This behavior would become part of the system's behavioral history, contributing to predictions of behavior that will be used to generate further behavior, causing it to deteriorate slightly, and so on. This feedback loop would continue, with nothing to halt the deterioration in behavior, until eventually the system is behaving randomly.

5.3.2.3 The Same Problem

Both of the above problems just described in 5.3.2.1 and 5.3.2.2 are special cases of the same problem: The driving of such a simplistic system is tautological, there being nothing to drive the system, beyond its own history, in the direction of improvement. Something outside the hierarchy is needed to give the system's behavior any preferred direction. This need is met by the action selection process previously discussed.

5.3.2.4 How the Action Selection Process Gets the System Started

The action selection process has the role of driving the system's behavior from outside. It does not need to drive the system to a great extent: The hierarchy is still doing the

[23] Also, as will be discussed shortly, modeling will produce most of the learning, but this will need a behavioral history with a pattern of improvement, and the action selection process is needed to establish this, unless such a history pre-exists in the system.

main work. All it needs to do is start the process off by causing the behavior of the system to improve initially, and then prevent random drift.

Suppose now that we have a system which uses the action selection process described previously.

Initially, the system has no behavioral history. The action selection process relies on predictions of inputs corresponding to future EFS values, but there is nothing on which to base these predictions. The system's initial behavior will be random.

After the system has been behaving randomly for a while, simple patterns will start to emerge between various actions and various effects and later EFS values: It will be apparent that doing certain things, in certain situations, causes low EFS values to occur later, and doing other things, in certain situations, causes high EFS values to occur later. The action selection process will now start to give meaningful predictions: Some output values will be determined to be better than others because they really are better, and these output values will be selected. The system is now behaving non-randomly, though its behavior is still simple.

The history of previous inputs/outputs provides a context, and that context is one of slightly non-random, slightly organized behavior. When any new output value is being assessed for desirability, it is being assessed within the context of all these previous actions: The hierarchy will assume that the previous behavior is a guide to what will happen in the future, and any specific output that is being considered will cause EFS predictions dependent on how it fits in with this previous behavior. (This issue of context is important: Outputs are not being assessed in isolation.) The fact that the action selection process is now selecting output values within the context of a history of slightly improved output values means that the outputs selected will become slightly better. This will in turn improve the behavioral history, improving the behavioral context for later action selection, and so on.

5.3.2.5 Learning as Modeling

The system as described above may seem to depend on the external action selection process for any improvement in its behavior, the modeling capabilities of the system just being used to determine the consequences of any action. If this were the case, the system's learning capabilities would be very limited, because the action selection process itself is very shallow. However, the action selection process is only there to impose, externally, a direction on the system. The real learning will come from the hierarchical modeling system itself.

How does this happen? I have already described how the action selection process, working with the hierarchy, can produce at least some improvement in behavior. This means that, after some time, the system will have a history of *improving* behavior: The pattern instances corresponding to previous inputs/outputs on the bottom level of the

hierarchy will describe behavior which has been gradually improving. *This improvement is itself a pattern*. When the hierarchy predicts the consequences of making a particular output, it will be looking at the output within the context of a pattern of improving behavior and outputs that best fit into such a pattern will result in the highest EFS values. The action selection process only has to cause a small amount of improvement in behavior, and the hierarchy will model an improving system on this basis. The improved behavior that results will become part of the system's behavioral history, in the bottom-level pattern instances, and this will contribute to modeling of a still more improved system – on top of the improvement that would have been modeled already. This cycle repeats, with improvement in behavior being added to the behavioral history where it leads to modeling of further improvement.

For an easy way of thinking about this, imagine the simplistic version of the system I have already described, in which the action selection process is removed and an output is selected merely by getting a prediction of it from the hierarchy, just before it is made. Assume that we somehow have a history of *improving* behavior. This improvement will have a pattern and the hierarchy will extrapolate this pattern into the future, predicting outputs which are slightly better than previous ones. These better outputs will become part of the behavioral history.

Approaches to planning of actions with some similarity to the one described in this article have been tried before in reinforcement learning, but I think that the potential power of such an approach tends to be overlooked. The approach gives a system with the power to extrapolate any small amount of learning you initially give it into continued, deeper learning, given a sufficiently general modeling system, which the first article was intended to describe. It may seem like an attempt to get something for nothing – to get the learning from nowhere – but the learning is really coming from the hierarchical model. Planning and learning have now been largely subsumed into the modeling process.

5.3.2.6 How the Action Selection Process Prevents Random Drift

From the above, it may seem that the action selection process is only needed to get the system going and establish an initial pattern of improvement, and that we could then dispense with it and just use the system's predictions of its own outputs to select them; however, there is also the problem of random drift. If an attempt were made to do this, small imperfections in the modeling by the hierarchy would affect outputs, which would in turn degrade the behavioral history, and in turn the system's performance, this cycle eventually causing random behavior.

As well as getting the system started off, the other purpose of the action selection process is to push the system, slightly, in the direction of improvement.

5.3.3 Why the Evaluation Function Score is Used as an Input

I have described treating the EFS as an input, so that each time it is computed, it is encoded as pattern outputs for one or more bottom-level pattern instances and stored in the hierarchy. To determine the desirability of making some output, the hierarchy is updated as if the output has occurred, and one or more pattern instances that will be used for future output of an EFS value is obtained to get an indication of desirability of the future situation.

Some readers may wonder why this is done. Why not just obtain predictions for conventional inputs (e.g. from vision or sound input devices) and apply the evaluation function to them to determine the desirability? The problem with this approach is that it requires the system's predictions of the future to be very specific. If the hierarchy predicts that, on average, some future situation will have a desirability of 20, this could happen in many ways. There could be a huge number of potential permutations of inputs, and trying to predict so specifically is much harder.

As an analogy, suppose that someone invents a pair of glasses which tell you how pleasant your current situation is. The idea is that the glasses tell you when good things seem to be happening to you and warn you when bad things seem to be happening to you. The glasses do this by looking at what is going on in front of you, using a camera, and applying some AI to this to generate a score, which is then superimposed on your field of view. If you find some money in the street, the glasses may display a high score. If someone punches you in the face the glasses may display a low score, warning you that you are having an unpleasant experience, so that you can deal with this.[24]

Suppose you are walking down the street wearing these glasses and a score of 80 is being displayed in your field of view. You trip and fall. As the ground rushes up to hit your face, you realize that the score displayed by the glasses is going to be a lot lower when your face hits the ground and the glasses realize that things are unpleasant. You guess that a score of about 15 is expected.

That score of 15 could result from a huge number of different ways of hitting the ground. To actually predict the details of the accident, in terms of inputs/outputs would be difficult – and maybe even impossible. However, that is not needed to predict how bad it will be. The predicted score is an average across this huge number of possible outcomes. You might not know *exactly* how you are going to experience hitting the ground, but you know it will be bad.

This is the kind of thing we are doing when we treat the EFS as an input.

[24] I am not expecting anyone to take such an invention seriously. It is just being used to illustrate a point.

5.4 Possible Variations on the Action Selection Process

5.4.1 The Possibility of "Giving the System More of a Start"

In Section 5.3.2.5: Learning as Modeling, on page 37, I described how the system can learn by modeling its own improving behavior. One role of the action selection process is to "give the system a start" by slightly improving its behavior. There is the possibility that more external systems could be added to give the system this initial start. The following might be used.

- A learning system, outside the main system, "shallower" than the hierarchical system being discussed in this article, and less powerful in the long-term, but able to produce more rapid improvement in the short-term, might initially be used to establish a history of improving behavior.
- "Pre-programmed" behavior, controlled by some system separate to the main system, might be used in the early stages.
- A false history of inputs/outputs might be fed into the system to establish a history of improvement.

In all these cases, the idea is the same: to establish an initial history, and therefore a pattern, of improving behavior, which the modeling system will then extrapolate into the future.

5.4.2 The Possibility of "Coming in Higher Up" in Action Selection

The approach just described, in Section 5.2: The Action Selection Process, on page 33, in which different output values are tried for an output, is effectively one of manipulating the bottom-level of the hierarchy. Some people may be critical of the idea that this would generate enough of an improvement in behavior to establish the pattern of improvement that is needed to lead to much more sophisticated behavior being modeled in the hierarchy. It is, perhaps, a difficult kind of objection to articulate, but it is basically one of accessibility and I should point out that I am aware of this possible concern.

If this turns out to be a problem, an alternative approach might be to "come in higher up" to make modifications to the hierarchy when trying to determine the merits of different courses of action: Instead of a bottom-level pattern instance being modified, a pattern instance just above the bottom level might be modified, or maybe one even higher up. An approach like this might be considered, although it would have complications: Pattern instances higher up in the hierarchy would tend to have values that depend on both inputs and outputs, and altering their probabilities would tend to imply making assumptions about future inputs as well as outputs, which could

compromise the system's accuracy. Nevertheless, it is possible that some approach involving changing probabilities higher up in the hierarchy could be devised.

If such an approach were used, the action selection process should remain essentially the same, except with the intervention in the hierarchy being higher up, unless a better alternative is available: the goal of maximizing future inputs of the evaluation function score, and therefore gaining the highest possible predicted values of such scores, remaining. An important reason for this is that the action selection process, by giving importance to specific pattern instances corresponding to future input of the EFS, allows those pattern instances to assigned relevance, ultimately allowing relevance to be determined for all the pattern instances in the actual hierarchy. (The importance of predicted EFS values to relevance is discussed later, in Section 8: Relevance, on page 47.) I suppose that an alternative might conceivably involve "coming in higher up" for the predictions as well as the manipulation of the hierarchy, involving the use of probabilities for higher-level pattern instances that are dependent on future inputs of the EFS. In this case, the general idea of basing outputs on predictions for specific pattern instances should remain, to allow the relevance processing to be described later to be used.

The action selection process, as just described, has implications for the "self", and these will now be discussed.

6 The "Self"

What I have been saying here is supposed to be a description of how we might make AI work and how *we* may work. Some readers may have noticed the lack of much reference to concepts such as minds, consciousness or the "self" in the discussion so far. Instead, I have taken a rather mechanistic approach, just trying to describe how a system may be made to behave intelligently. There is no explicit attempt to represent anything like "consciousness" or the "self" in all this, so what of the "self"? How does it relate to all this? [25]

The hierarchy contains patterns relating previous inputs and outputs. We might consider this to be representing how the "outside world" behaves, but any distinction between an "outside world" and an "inside world" is, from the point of view of the hierarchy's model, meaningless. The hierarchy makes no such distinction. It simply models inputs/outputs, based on any patterns relating them, whether these patterns result from events in the "outside world" or events in the system itself.

Some of the patterns in the modeling system will correspond to things in the outside world. It may have some pattern relating inputs, corresponding to "tree", or a pattern relating inputs over a period of time corresponding to "moving closer". Some of the patterns will relate outputs to later inputs, so that they link actions to the consequences of those actions.[26]

Some kinds of patterns should particularly interest us, in any consideration of the "self". These will be patterns that relate outputs to later outputs. Patterns like this are modeling the system's own behavior. The hierarchy will model the AI system itself. What we think of as the "self", "consciousness" or "intentionality" all come down to the system's representation of itself in it own hierarchical model. As well as applying to AI systems, I am saying that this applies to us. What this means (and I know it is a strange statement) is:

"You" are an object in a hierarchical model, generated by your brain's modeling system to explain your brain's previous behavior, and therefore predict your future actions, just as anything else is modeled to explain and predict things.

---

[25] Almond, P., 2010. *An Attempt to Generalize AI - Part 2: Planning and Actions*. [Online] paul-almond.com. http://www.paul-almond.com/AI02.pdf or http://www.paul-almond.com/AI02.doc. pp.23-24.

[26] This should explain why a simple evaluation function (EF) in the action process is not inconsistent with the complex, highly abstracted motivations that we experience: They are not the same thing. The motivations, desires, goals, etc. that we experience are part of the abstracted model of our behavior in the hierarchy, far above the level at which the EF works, which has emerged from the system's complex behavior.

In this view, the "self" has no *qualitatively* special status, though it would differ from other objects in the model in various matters of degree.

The idea that the "self" is merely an object in a model has been previously proposed by Thomas Metzinger, a philosopher.[27]

[27] Metzinger, T. (2003). *Being No One: The Self-Model Theory of Subjectivity*. Cambridge, MA: MIT Press. Metzinger, T. (2009). *The EGO Tunnel: The Science of the Mind and the Myth of the Self*. New York: Basic Books. While noting Metzinger's work, I will stay away, in this article at least, from the philosophical issue of whether this means that the self is not real. I have said what I think the "self" is, and how it gets caused in a system. There are technical issues to cover for now.

7 Geometry

7.1 Breaking the Strong, Geometrical Analogy

The hierarchy is intended to provide a general ontology which can have real-world relationships mapped onto it. The ontology is more general than that in my previous proposal.[28] In a previously proposed, and now obsolete, system, each pattern consisted of a set of pattern instances arranged in an array, each input of a pattern instance being the pattern instance of another pattern at some coordinate offset.[29] This required all the pattern instances of a pattern to be at the same level of the hierarchy, looking at the same data. Reality does not work like this. An abstract concept such as "circle" might be relevant at multiple levels of the hierarchy and describe a relationship between different kinds of object. The basic system could not, in itself, represent this kind of abstraction. The approach limited the relationships between the pattern instances in a pattern to being ones based on a strong, geometrical analogy. In reality, the inputs/outputs at the bottom level of the hierarchy may be related according to some simple, strong, geometrical analogy, but this might not be the case for more abstract pattern instances, higher in the hierarchy. The previously proposed system would impose this geometrical analogy all the way up though, with the geometry used closely matching that of the input/output data. As well as limiting the generality of the hierarchy's ontology, this could cause problems when the input/output data on the bottom level corresponds to pattern instances in different arrays. This will be particularly obvious if the arrays of input/output pattern instances have different dimensionality for different input/output devices.

I knew this was an issue at the time, and to try to deal with it I proposed that "meta-patterns" should be used. A meta-pattern defines a set of patterns combining all of their statistics. A single meta-pattern could cause many patterns to exist at many levels in the hierarchy, allowing more abstract relationships to be represented. This solution is a crude attempt to provide generality. Any generality it provided would be limited according to whatever system was used for representing meta-patterns and the basis of it all would still be simple, geometrically based patterns. While they may give some capability for abstraction, the meta-patterns themselves would still impose a strong, geometrical analogy. The proposal in this article is intended to provide more generality, although with the cost of me being unable to say, right now, how some parts of the system, in particular, the construction specification, should work.

---

[28] Almond, P., 2006. *A Proposal for General AI Modeling*. [Online] paul-almond.com. http://www.paul-almond.com/Modeling.pdf or http://www.paul-almond.com/Modeling.doc.

[29] For example, if Pattern Instance (100,100) of Pattern 1 obtained Input A from Pattern Instance (103,104) of Pattern 2, then Pattern Instance (101,101) of Pattern 1 would obtain Input A from (104,105) of Pattern 2.

This issue of geometrical analogy, with the higher-level features of the model being expected to map onto the same kind of geometry as the inputs/outputs, is also present in the hierarchical system proposed by Hawkins.[30] The system being proposed here is an attempt to get beyond this by breaking the dependence of relationships between elements of the same kind in the hierarchy on an analogy with geometry and the way that the inputs/outputs are arranged.

With the system proposed here, "hierarchy" does not mean a system arranged in neat layers. All it really means is that some pattern instances get their inputs from other pattern instances and so can be considered to be on a "higher level" than them.

In the proposed system, the construction specifications of patterns would probably tend to be simple. This means that any individual pattern is not likely to use relationships vastly different from those of the pattern instances from which its own pattern instances tend to get pattern inputs. The bottom level of the hierarchy consists of arrays of pattern instances corresponding to external inputs/outputs. These pattern instances *are* arranged in a regular, geometrical way, so it is likely that pattern instances which obtain all their inputs from here will also tend to be related with a geometrical analogy, but to a slightly lesser degree. As we go higher in the hierarchy, the relationships between pattern instances will become progressively less like those between the bottom-level pattern instances. Things will become less describable in simple, geometric terms, or in some cases different geometries will start to apply. At the bottom level of the hierarchy, and with pattern instances not too far away from it, it will make sense to use coordinates to describe the "position" of a pattern instance, but as we go higher in the hierarchy this will become less meaningful and it will only make sense to describe "where" a pattern instance is in terms of the pattern instances it uses for inputs.

This "breaking of the strong, geometrical analogy" is an important part of what I am trying to do here.[31]

7.2 Weak Geometry

This article will refer to the idea of "regions" in the conceptual hierarchy, as well as related concepts such as "nearness" of pattern instances. When I refer to concepts that might seem to relate to geometry, such as "region" in this article, a spatial analogy is not being used.

Most of the hierarchy is not explicitly spatial: Pattern instances do not have coordinates in some space. Instead, a pattern instance is described in terms of its internal logic and

---

[30] Hawkins, J. & Blakeslee, S., 2004. *On Intelligence*. New York: Henry Holt.

[31] Almond, P., 2010. *An Attempt to Generalize AI - Part 1: The Modeling System*. [Online] paul-almond.com. http://www.paul-almond.com/AI01.pdf or http://www.paul-almond.com/AI01.doc. pp.27-28.

state and the pattern instances to which it is connected. This does not preclude the use of a weak, geometrical analogy: We can still apply some idea of "proximity" to the hierarchy, without space, using the concept of logical distance between pattern instances.[32] 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.

Some readers might ask how *any* information about geometry manages to enter the hierarchy in the first place. It should be noted that on the bottom level of the hierarchy, the pattern instances would often have some relationship to each other corresponding to the real world: For example, if two light sensitive elements are next to each other in a camera, this needs to be represented in the hierarchy. As an exception, the construction specifications for pattern instances would be able to access information for bottom-level pattern instances that was directly derived from the real-world arrangement of the corresponding input/output devices. Immediately above the bottom level, information derived from spatial relationships would not be explicitly available, but it should still be reflected in the *logical* structure of the hierarchy, as it has been derived directly from a structure with explicit spatial relationships. Further above the bottom level, things would get further away from the simple, spatial relationships on the bottom level, and relationships between pattern instances would become increasingly abstracted into ones that cannot be well-represented by simple, spatial coordinates.[33]

---

[32] Almond, P., 2010. *An Attempt to Generalize AI - Part 4: Modeling Efficiency*. [Online] paul-almond.com. http://www.paul-almond.com/AI04.pdf or http://www.paul-almond.com/AI04.doc. p.11.

[33] I discussed the issue of weak geometry online with Steve Grand OBE in April 2010 in the comments of a post on his blog (Grand, S., 2010. *Brainstorm #3 – cheating doesn't pay (sometimes)*. [Online] Steve Grand's Blog: Artificial Life in Real Life. http://stevegrand.wordpress.com/2010/04/22/brainstorm-3-cheating-doesnt-pay-sometimes/). Steve Grand disagreed with me, saying that there is evidence that brain cells involved in facial recognition are arranged in a way that corresponds with location of features in the "outside world". In response to this, I would say that we should expect this: Sufficiently close to the bottom of the hierarchy, the pattern instances would still be strongly influenced by the underlying,

8 Relevance

8.1 The General Idea Behind Processing for Relevance

Without some way of ensuring relevance, the hierarchy will be attempting to model every aspect of the world, which is clearly impossible. There needs to be a way of determining which patterns should be in use and which pattern instances should be included in the actual hierarchy. Patterns, and individual pattern instances, need to be selected according to their *relevance* to the hierarchy's function of making the predictions that will best guide its actions.

The approach to relevance will rely on the action selection process, described previously in Section 5: Planning of Actions, on page 32.[34] The action selection process uses evaluation function score (EFS) values which are continually computed and treated as inputs. A pattern instance corresponding to a future EFS input which is used in the action selection process can be regarded as having some relevance, and then a relevance measurement process (RMP) back-propagates this relevance to other pattern instances in the hierarchy, assigning relevance values to them. The relevance values for the hierarchy are used to decide which pattern instances to keep and which to remove, and the hierarchy is extended in an exploratory way, so that it tends to grow into high-relevance regions and retreat from low-relevance ones.

Relevance needs to be considered with regard to patterns and individual pattern instances. I will start by considering it with regard to individual pattern instances, so I will be assuming, for now, that we already know which patterns we want to use, but want to decide which pattern instances of those patterns should be in the hierarchy.

8.2 What parts of the hierarchy are relevant?

8.2.1 Analyzing the Actual Hierarchy

We need to look at how we might determine the usefulness of different parts of the hierarchy. We need a way of taking an actual hierarchy, in an AI system, and measuring the usefulness of each pattern instance in it. Before we start this, we should step back and consider what we mean by "useful". This all comes back to what the point of the hierarchy is.

---

geometrically arranged inputs/outputs, and would tend to be arranged in ways that are close to mapping onto a spatial geometry. This effect would get weaker as we went higher up in the hierarchy.

[34] Almond, P., 2010. *An Attempt to Generalize AI - Part 2: Planning and Actions*. [Online] paul-almond.com. http://www.paul-almond.com/AI02.pdf or http://www.paul-almond.com/AI02.doc.

8.2.2 The Purpose of the Hierarchy

The hierarchy's use is in the action selection process, previously described in Section 5: Planning of Actions, on page 32.[35] An EFS is continually computed, being treated each time as if it had been received as an external input to the AI system, so that EFS values are continually encoded as bottom-level pattern instances. When an output is required, each possible value is tried and the hierarchy updated as if the output had occurred with that value, the relevant changes to probabilities being propagated through the hierarchy. The pattern instance(s) corresponding to a future prediction of the EFS input is/are obtained from the hierarchy, and used to obtain an expected EFS value for each possible output value. This allows the better output value to be selected, and the output actually occurs with this value.

This process might be modified to some degree, and it is not the real planning process: The *real* planning process actually occurs as part of the hierarchy's modeling of its own behavior, as a natural part of the hierarchy's functioning. Nevertheless, this is the process by which the hierarchy is driven to produce a certain type of behavior. Even if it gets modified later to some extent, the important feature of it will still remain: *We are interested in predicting the values of pattern instances corresponding to specific future inputs (the EFS values) on the bottom level of the hierarchy.*

8.2.3 The Importance of the Action Selection Process

It is no accident that the action selection process works in this way: It gives us a well-defined requirement. We want the hierarchy to predict a specific, future input value and this means that we will want the uncertainty in the pattern instance(s) corresponding to that value to be minimal. We want the probability associated with that pattern instance to reflect as little uncertainty as possible – meaning that it should be as close as possible to 0 or 1. This gives us something measurable, and it suggests a possible opening for us to measure the performance of the hierarchy, or parts of the hierarchy, with reference to this objective, and optimize it to reduce the uncertainty in the predicted value.[36]

The action selection process provides a well-defined end-point, which can be regarded as relevant, and which can be used as a starting point for a process of back-propagation of relevance.

---

[35] Almond, P., 2010. *An Attempt to Generalize AI - Part 2: Planning and Actions*. [Online] paul-almond.com. http://www.paul-almond.com/AI02.pdf or http://www.paul-almond.com/AI02.doc.

[36] If some more sophisticated version of the action selection process were used, involving multiple EFS predictions, the general idea would be the same – as would the important measurability in this: We would just want to minimize the uncertainty in the predictions of whatever EFS values were involved.

8.2.4 Does the propagation of information *from* previous inputs/outputs tell us anything useful?

We might consider using the flow of information from the bottom-level pattern instances to tell us which pattern instances are relevant – viewing the pattern instances which get their probabilities strongly affected by the pattern inputs as particularly relevant, and therefore worthy of representation in the actual hierarchy – but the information spreads out exponentially to so many pattern instances that we could never represent them all.

The relationships themselves will tell us nothing useful, but what about the actual probability values? One pattern instance might affect another in a way that significantly reduces the uncertainty in its probability, while another pattern instance affects it in a way that only slightly reduces its uncertainty. Could we say that it is only interesting when pattern instances *strongly* affect others, and try to follow the "paths of strong effect" through the hierarchy?

This deals with part of the issue. The default probability value for a pattern instance is 0.5, and we know nothing of such a pattern instance. If we represented all of the hierarchy, much of it would remain like this, even after propagation of information through it, because the inputs/outputs would just not tell us much about a lot of the hierarchy. It would not be worth using these pattern instances for anything, which means it would not be worth computing probabilities for them in the first place: We should only be interested in parts of the hierarchy which have probability values that are significantly affected by occurrence of inputs/outputs. If we follow the paths through the hierarchy from the bottom-level pattern instances corresponding to previous inputs/outputs, through such pattern instances, we will be considering all of the hierarchy that could really interest us, and if we stray away from such paths we will just enter regions of the hierarchy where the probability values are all close to 0.5, and therefore useless.

This is only part of the issue, however. A pattern instance needs to be significantly affected by propagation of information from previous inputs/outputs to be of any interest, but being significantly affected does not mean that it *must* be of interest: It is merely the minimum requirement. Many paths through the hierarchy will lead to pattern instances that are strongly affected by previous inputs/outputs, but which have hardly anything to do with the probabilities of the pattern instances that interest us.

As an example, suppose that you are a detective investigating the particularly nasty murder of a celebrity in the USA. You may work out that the story of the murder will appear on the Internet, that people in Switzerland will read about the murder, and therefore that a fan of the celebrity *who works in the Swiss chocolate industry* will be horrified by the murder. You might be following the "flow of strong effect" here, but it would probably be leading you nowhere useful.

8.2.5 Does the propagation of information *into* pattern instances tell us anything useful?

8.2.5.1 Let's just consider the structure…

If it is useless to start at pattern instances corresponding to previous inputs/outputs and work our way forwards, what if we start at a pattern instance corresponding to a future input/output which we want to predict and work *backwards*, looking at what pattern instances affect it, and what pattern instances affect those, etc., all the way back to the pattern instances corresponding to previous inputs/outputs, the idea being that the pattern instances through which we passed are the important ones?

To start with, we will just consider the *structure* of the hierarchy – the way pattern instances are connected together – without considering any of the probability values allocated to pattern instances. In other words, we will look at what pattern instances a given pattern instance uses at its pattern inputs, but we will not look at any of the probability values themselves. As previously discussed, multiple pattern instances can directly affect a single pattern instance through logic application or statistics application. Suppose we start at a single, bottom-level pattern instance corresponding to a future pattern instance. We could find that a number of pattern instances directly affect it. For each of these, we could find a number of pattern instances affecting it in turn, and so on. As we work backwards, the number of pattern instances that we have to consider will, once again, increase exponentially. Eventually, we might be considering all of the hierarchy. We would eventually get back to the previous inputs/outputs along some paths, but other paths would lead us into regions of the hierarchy that are useless.

Maybe we should consider some approach that combines working forwards from previous inputs/outputs and backwards from future inputs/outputs? Such an approach will still have issues with exponential increase. Fortunately, there is a better way. I introduced this part of the discussion by saying that we would start by ignoring the probability values. What if we tried working backwards from a pattern instance corresponding to a future input/output, but this time we looked at the actual probability values describing individual pattern instances? We will now consider this kind of approach.

8.2.5.2 But what if we consider probabilities too?

Suppose we start at a bottom-level pattern instance corresponding to a future input/output which we want to predict, probabilistically. As before, we look at the pattern instances that directly affect the probability of this pattern instance by logic or statistics application. This time, however, we do not just look at whether or not a given pattern instance directly affects the one that interests us. We look at how strongly it affects it.

How can we determine how strongly one pattern instance affects another? This will be discussed in more detail; however for now we will just consider the probability of the pattern instance doing the affecting. All else being equal, if two pattern instances directly affect the pattern instance that interests us, and we know a lot about one of them – it has a probability value close to 1 or 0 – and we know little about the other one – it has a probability value close 0.5 – the one about which we know more will do more affecting.

Why this is the case should be obvious. As stated previously, the kind of transmission of "effect" that we are dealing with when we consider propagation of probability values is not a transmission of *real* effect. The probability values represent what we know about the hierarchy, and when one pattern instance's probability "affects" another's probability through logic application or statistics application, what we know about one pattern instance is being used to tell us about another pattern instance. If we know a lot about a pattern instance, we might expect this knowledge to have a big effect on what we know about another pattern instance – to alter its probability value so that we are much less uncertain about it. If, however, we know hardly anything about a pattern instance, we should hardly expect our knowledge to have much effect on anything else. Consider, for example, a pattern instance with a probability of 0.5. We know nothing at all about this pattern instance. Does it have a state of 0 or a state of 1? We have no idea: We do not even know which state is more likely. If we tried to use this pattern instance in logic application or statistics application, this would tell us nothing: It would have no effect.

We could, then, look at all the pattern instances directly affecting the one that interests us through logic application and statistics application, upwards or downwards. We would look at the probability of each such pattern instance and that would give us an idea of how much effect it would have, *all else being equal*. We could store a *relevance value* for each pattern instance indicating how much effect it had on the pattern instance that interested us. This would give us a list of pattern instances, and how relevant each one is. We could now work back to each of these pattern instances and look at the pattern instances that directly affect it, allocating each one a relevance value, but now, we would not just base the relevance values on the probability values of the pattern instances, but also on the importance of the pattern instance they were affecting. That is to say, a pattern instance gets a lot of relevance by having a high probability and affecting a relevant pattern instance. We could continue to work back like this, propagating relevance backwards. Eventually we would reach the bottom-level pattern instances corresponding to previous inputs/outputs.

We could not use such a process to construct the actual hierarchy by itself. A process like this only tells us how important a particular pattern instance in an existing representation of the hierarchy, so we need the hierarchy there before we can use it. However, a process like this would tell us about which paths through the hierarchy were relevant.

Suppose that no attempt had been made to limit the size of the actual hierarchy in the AI system. Applying the process just described would mean we would have to process all of these pattern instance values, but suppose we prioritized? We could start at a pattern instance corresponding to a future input/output which we wanted to predict and work backwards. As we traced the propagation of information backwards, the paths would branch out, but we could prune branches which led us into irrelevant parts of the hierarchy. Whenever we encountered a pattern instance of low relevance – meaning its relevance would be below some defined level – we would not follow any paths going back further from that pattern instance. We would only be following paths of high enough relevance through the hierarchy. A pattern instance on such a path would be important in affecting the probability of the pattern instance that interested us, or the path would have been pruned earlier on, but to have that much effect it would generally have a relatively high probability value: It would have been strongly affected by propagation of information from pattern instances corresponding to previous inputs/outputs. The pattern instances encountered on such paths would be relevant: They would be highly dependent on previous inputs/outputs *and* important in making predictions of future inputs/outputs. With such pruning, the paths followed back from the predicted pattern instance would automatically "seek out" the previous inputs/outputs, passing through regions of maximum relevance to get there.

Such a pruning process would leave us with the *relevant* parts of the hierarchy. It would be doing what we wanted: focusing the system on the *relevant* and removing the *irrelevant*.

Such a process could not be used, exactly as described, to generate the hierarchy. In this discussion we assumed that we started with all of the hierarchy being represented in the actual hierarchy, and then pruned the less relevant parts. In reality, we would never be able to construct or handle a hierarchy that large in the first place; however, this way of propagating relevance can form the basis of a practical process. The ability to measure the relevance of pattern instances in the hierarchy enables an exploratory process. It allows us to analyze a hierarchy and decide which parts to keep and which to remove, based on relevance, and where we should try connecting new pattern instances to the hierarchy. Such a process can start with a hierarchy of manageable size and grow and prune it, the process being informed by continual measurement of the relevance of pattern instances, so that a relevant hierarchy is grown.

We could use a process of pruning like the one just described, in which high-relevance paths are followed back and low-relevance ones are pruned from the relevance back-propagation process itself, but we do not necessarily have to do that. I described such a process to show how we could start with a hierarchy which was too large for us to process in its entirety, and how we could still prune it, but, if we stop the hierarchy getting too large in the first place, by controlling its growth, informed by continual measurement, we could always keep the actual hierarchy small enough to be managed in its entirety.

This article will not get very involved in a discussion of how we could use a relevance back-propagation process to direct the growth of an actual hierarchy in an AI system. The next article in this series will discuss how we could actually *use* relevance back-propagation in an exploratory process to control the growth of the hierarchy: This article will just discuss the relevance back-propagation process.

8.2.5.3 The amount of effect is not just determined by probability.

When I said that probability would indicate the extent to which one pattern instance affects another I said "all else being equal". This is a simplification, though it is not a bad one. In reality, when one pattern instance affects another, through logic application or statistics application, the amount of effect depends on its probability value and the way that it interacts. The pattern logic will be a factor, as will the probability values of other pattern instances which are affecting the same pattern instance. We might know a lot about a pattern instance – it may have a value very close to 0 or 1 – but this might not translate into it having a strong effect on another pattern instance, because the pattern logic of that pattern instance may use it as a relatively unimportant pattern input, or it may be made unimportant due to other pattern inputs. Probability matters, but so do other things. On the other hand, a pattern instance with a probability of 0.5 will never propagate any information to any other pattern instance: We know nothing about such a pattern instance, so it can tell us nothing about any other pattern instances.

The relevance measurement process (RMP), a process for assigning relevance to pattern instances will now be described.

8.3 The Relevance Measurement Process (RMP)

Relevance is assigned to pattern instances by the *relevance measurement process* (RMP), which is one of *back-propagation of relevance* through the hierarchy.[37]

Each pattern instance is going to be assigned a *relevance value*. The relevance values of all the pattern instances in the actual hierarchy are initially set to 0. An exception is the bottom-level pattern instances corresponding to the particular input/output values, required for predictions of future EFS values by the action selection process, that we want to predict, and which are assigned non-zero relevance values; for example, they might each be assigned a relevance of 1.

Relevance is propagated back from the pattern instances about which we want to make predictions, to the pattern instances which affected their predictions, depending on how much they affected them, and relevance values are propagated back still further to the ones which affected them and so on. Every incident of a pattern instance

[37] Almond, P., 2010. *An Attempt to Generalize AI - Part 6: Measuring Relevance*. [Online] paul-almond.com. http://www.paul-almond.com/AI06.pdf or http://www.paul-almond.com/AI06.doc.

propagating information to another one in setting probabilities now has a corresponding act of relevance being propagated back. In other words, a pattern instance gets relevance by being relevant to one that is already relevant.

The basis of the RMP is the back-propagation of relevance from a single pattern instance.

8.3.1 Description of the Relevance Measurement Process (RMP)

The Relevance Measurement Process

Suppose we have some pattern instance X, which has a relevance value of R_X. For each pattern instance P_1, P_2 ,…P_n which directly affected X's probability (such as in logic application or in upwards or downwards statistic application), determine the *amount of effect* that that pattern instance had on X's probability value. Sum all the amount of effect values, E_1, E_2, …E_n, giving a total amount of effect, E_T. For each pattern instance P_1, P_2, …P_n, whatever proportion of the total effect, E_T, its own amount of effect is, add that proportion (E_1/E_T, E_2/E_T, … E_n/E_T) of X's relevance value, R_X, to its existing relevance value, R_1, R_2, …R_n. e.g. $(E_1/E_T)R_X$ is added to R_1.

When a pattern instance is given relevance because of its effect on the probability of another pattern instance, that relevance is *added* to any relevance that it already has: It does not replace it. This means that a pattern instance can receive relevance from a number of pattern instances which it affects, all the relevance that it receives combining to give its total relevance.

When relevance is back-propagated from some pattern instance, X, to those pattern instances which have affected its probability, it does not mean that X's relevance is reduced: X still keeps its own relevance.

It is simplest to imagine this working with the basic action selection process discussed previously[38], with a single EFS value being required, and any pattern instance(s) which are to be used for a future input of this value being assigned an initial, non-zero relevance at the start. With different variations on the basic action selection process, however, a process like this could as easily work with multiple EFS predictions being required from the hierarchy: It would just mean that the corresponding pattern instances would be set with non-zero relevance values at the start of the relevance back-propagation process.

[38] Almond, P., 2010. *An Attempt to Generalize AI - Part 2: Planning and Actions*. [Online] paul-almond.com. http://www.paul-almond.com/AI02.pdf or http://www.paul-almond.com/AI02.doc.

Example

A pattern instance, X, has a relevance of 0.8. It was assigned some probability by being directly affected by three pattern instances, P_1, P_2 and P_3.

P_1 is determined to have had an effect of 3.5 on X's probability. P_2 is determined to have had an effect of 2.6 on X's probability. P_3 is determined to have had an effect of 7.2.

The effects are all summed to obtain E_T.

E_T =3.5+2.6+7.2 =13.3

P_1's proportion of the total effect is $^{3.5}/_{13.3}$ =0.263, so P_1 gets this proportion of X's relevance. Therefore, 0.263x0.8 = 0.2104 is added to P_1's relevance.

8.3.2 Determining the Amount of Effect of a Pattern Instance

The RMP requires us to have the ability to determine the amount of effect that a given pattern instance has on the prediction for another pattern instance, X. What we are interested in is how much the given pattern instance seems to reduce the uncertainty in prediction of X. Any "effect" means moving the probability value of X away from 0.5 and towards 0 or 1, so that X is known about with less uncertainty.[39]

All else being equal, a pattern instance's probability should indicate the amount of effect it can have on other pattern instances. We can see this by considering the case of a pattern instance with a probability of 0.5. Nothing is known about such a pattern instance. In logic application or statistics application, it will be useless for telling us anything about other pattern instances. As the probability of a pattern instance gets closer to 0 or 1, and we know more about it, the more it should tell us about other pattern instances, and the greater the effect that it should have on their probabilities. A crude way of measuring the amount of effect of a pattern instance on X would simply be to look at how far away its probability value is from 0.5.

This was "all else being equal", however. A pattern instance's effect on some other pattern instance, X, is not just determined by its probability. The way in which the two pattern instances are interacting also matters. For example, a pattern instance may be known about with almost complete certainty (say it has a probability of 0.99) and it may affect X by downward statistics application because X serves as one of its pattern inputs, but the pattern instance's logic may mean that this does not tell us much about X. That

[39] The same would apply if there were more that one probability value associated with a pattern instance: Any "effect" that a pattern instance had on the prediction for some pattern instance, X, would mean moving its probability values towards 0 or 1 to indicate less uncertainty.

said, probability is still important: A probability of 0.5 will always have no effect at all, regardless of the interaction between two pattern instances – and the default probability of a pattern instance is 0.5.

We need to determine the effect of a given pattern instance on X in a way that takes account of the actual effect, without just looking at probability. This could be done in a number of ways. A simple method is to look at how much uncertainty was removed from X's probability value when it was affected by the given pattern instance, and how much would have been removed if the given pattern instance had a probability of 0.5. The more effect the given pattern instance is having, the greater should be the difference between the uncertainty removed when it has its actual probability and the uncertainty removed when we pretend it has a probability of 0.5.

Example

A pattern instance, X, is affected by a number of pattern instances in upward statistics application, as these pattern instances serve as its pattern instances. One of these pattern instances is A, which has a probability of 0.8. After statistics application, X has a probability of 0.65, but if we had set A to 0.5, X's probability would have been 0.75. The effect of the information in A was therefore to increase X's probability from 0.65 to 0.75, because 0.75 was achieved with the process working normally and only 0.65 when A was effectively disabled. This means that A can be considered to be removing 0.75-0.65 =0.1 of "uncertainty" from X, and this can be considered an indication of its amount of effect on X.

(Note: A change in uncertainty is not necessarily the same as a change in the probability value. For example, reducing a probability value from 0.7 to 0.4 is a reduction of 0.3 in the probability value, but this is actually an *increase* of 0.1 in uncertainty, because 0.7 is 0.2 away from 0.5 and 0.4 is only 0.1 away from 0.5.)

We might repeat the process by which pattern instances initially affected X, but with the probability value of the given pattern instance set to 0.5, to see what the effect is on the probability value assigned to X. This might involve "rewinding" the process by which probability values were assigned as we go along, or it might be done at the time that they were assigned in the first place, so that, for each pattern instance, the effect on it of other pattern instances is stored for later relevance propagation. We may need to be careful about how we organize such a process, but the basic idea will always tend to be the same: The effect of a given pattern instance on X is defined in terms of how much more uncertainty it removes from X with it working normally, than when it has a probability of 0.5.

8.4 The Basic, Exploratory Relevance Process (BERP)

8.4.1 Starting with a Basic, Exploratory Process

A process which controls the development of the hierarchy could take a number of forms, and different degrees of sophistication are possible. I want the main idea to be clear, so to start with I will just consider a basic process, which I will call the *basic, exploratory relevance process* (BERP) – about the simplest process that illustrates the main idea of what is going on.[40] Possible ways of extending this into a more sophisticated process are described in Section 9: Possible Improvements to the Basic, Exploratory Relevance Process, on page 70.

8.4.2 What Would Happen With an Unrestricted Hierarchy

The hierarchy is based on patterns. Each pattern has a set of pattern instances, and a pattern can cause its pattern instances to be "wired into" the hierarchy. If pattern instances were wired into the hierarchy in an unrestricted way, then the hierarchy would develop without any regard for relevance. The result would be a hierarchy consisting almost completely of very low-relevance pattern instances. A huge amount of computation would be required to compute the hierarchy, just to get useful results from the small number of pattern instances that happen to be relevant.

8.4.3 The Objective of the Basic, Exploratory Relevance Process

The objective of the basic, exploratory relevance process (BERP) is to ensure that the scenario just described, in 8.4.2, does not happen: that the hierarchy tends to consists of *relevant* pattern instances, rather than irrelevant ones.

Consideration should be given to how far this is taken. We might adopt an approach of only allowing the most relevant pattern instances in the hierarchy, so that if the hierarchy contains n pattern instances then the only way any pattern instance can remain in the hierarchy is by being one of the n most relevant pattern instances known to the system at any time and the system not encountering any pattern instance with higher relevance. A possible issue with such an approach is one of *accessibility*. There could be regions of the hierarchy with very high relevancy, but which can only be reached via intermediate pattern instances with slightly lower relevancy. If such pattern instances are not allowed, such regions could never be reached.[41]

For this reason, a better approach may be one that does not always select the absolute "best" pattern instances, but instead *favors* those with higher relevance, so that the number of pattern instances allowed in the hierarchy at any time with a given degree of

---

[40] Almond, P., 2010. *An Attempt to Generalize AI - Part 7: A Basic, Exploratory Relevance Process*. [Online] paul-almond.com. http://www.paul-almond.com/AI07.pdf or http://www.paul-almond.com/AI07.doc.

[41] Some readers may see a similarity with the issue of accessibility in Darwinian evolution, here.

relevance is related to that degree of relevance, increasing as the degree of relevance increases: We would have a lot of pattern instances with high relevance and fewer pattern instances with low relevance. It should be noted that an approach of selecting only the most relevant pattern instances is really only a special case of this, in which the distribution of pattern instances over a range of relevance is made as narrow as possible.

8.4.4 What the Basic, Exploratory Relevance Process Will Do

The BERP will attain a relevant hierarchy in two ways, as follows.

8.4.4.1 Removal of Pattern Instances

The BERP will remove pattern instances from the hierarchy, with the decision to remove a pattern instance based on its relevance value. As previously stated, in 8.4.3, a good way of doing this may be to allow greater numbers of more relevant pattern instances, so that the chance that a pattern instance is removed is dependent on its relevance, with the chance increasing as relevance decreases.

8.4.4.2 Addition of Pattern Instances

The BERP will control the addition of new pattern instances to the hierarchy by patterns. Addition of pattern instances should be allowed more in regions of the hierarchy where the new pattern instances are expected to have high relevance. A good start is to assume that higher-relevance regions of the hierarchy are the ones where more new pattern instances should be added. As with the removal of pattern instances, this should be done in a continuous way, so that new pattern instances are connected more often to higher-relevance regions of the hierarchy than to lower-relevance ones.

We do not need to go to any trouble to achieve this: The removal of pattern instances by the BERP, as just discussed, above, will actually ensure that this happens anyway, if we just allow new pattern instances to be connected to the hierarchy with equal frequency everywhere, or randomly. This is because the BERP's actions in removing pattern instances based on lack of relevancy will mean that the distribution of pattern instances in the hierarchy will be weighted towards higher-relevancy ones. If we just connect a pattern instance to the hierarchy in some randomly chosen region of the hierarchy, the distribution of pattern instances over relevancy throughout the hierarchy, caused by the removal of pattern instances, will automatically mean that it is more likely to be connected to a higher-relevance region. To put it another way, by taking out low relevance pattern instances, we are removing opportunities to connect new pattern instances to them, and therefore making new pattern instances more likely to be connected to high-relevance regions of the hierarchy.

8.4.5 Why add more pattern instances in higher-relevance regions?

The BERP will try to get relevancy by allowing more pattern instances to be connected to parts of the hierarchy where relevancy is already high, and as just explained, in 8.4.4, the BERP's removal process will tend to ensure that this happens anyway by manipulating the distribution of pattern instances in the hierarchy. Having new pattern instances connected to higher-relevance regions of the hierarchy is supposed to make it more likely that they too are highly relevant. This may seem to be assuming a lot. If a part of the hierarchy has high relevancy, why should this mean that any new pattern instances we add there will also have high relevancy? It is not being assumed that this is *always* the case. The idea is merely to increase the chances of new pattern instances having high relevancy.

Any pattern instances being connected "logically close" to ones that are already relevant will be related in some way to ones that are known to be relevant. This should make them more likely to be relevant themselves, and the more direct the relationship, the greater the chance of high relevancy.

As an example, imagine you are a detective investigating a crime. At the crime scene, a receipt was found for purchase of a book about western movies. Two new leads are offered to you, but for some reason you do not have full information about what the leads are, except they are both supposed to relate to the case in some way. You do know that one lead involves someone who rented a western movie from a store nearby, and another lead involves someone who bought a cheeseburger nearby. When you choose which lead you want to follow, you will be given full details.

All else being equal, the lead involving the person renting the western movie is more likely to *lead* somewhere useful. We might justify this by saying that it clearly involves someone who likes western movies, and we think the criminal may like western movies, but the real justification is more general to this: The better lead has the stronger relationship with what we know to be relevant – with our pre-existing structure of knowledge about the case.

This will be more obvious if we consider a more abstract example, one in which we may not have any really good, specific reasons for preferring one lead over another, but in which one lead is clearly more directly related to what we know to be relevant.

Suppose you are a detective investigating a murder, and the victim was known to be a scholar of Latin. In the course of your investigations, you find two suspicious looking, handwritten notes. One is in Greek. The other is in Esperanto. You have to choose which note to get translated first. All else being equal, you should prefer the note in Greek. Greek is a "classical" language, as is Latin, and so has a stronger relationship with what you know to be relevant. How could it be linked to what is going on? We do not know yet. Maybe the victim wrote it: As a Latin scholar it is reasonable to think he/she may have known another classical language. Maybe someone close to the victim wrote it? A

Latin scholar is likely to have associates who know Greek, maybe including personal enemies. Maybe the murderer wrote it? The point here is that we may not know what the specific significance of this note is, but most of you, reading this, would not have needed to know that. You would have immediately decided it was likely to be the more relevant piece of evidence based on its close relationship to what is already known to be relevant. All else being equal, the Latin note is more likely to be relevant because it is in a "region" of our model of the world that we already know to be relevant.

When we encourage addition of pattern instances in high-relevance regions of the hierarchy (and we can do this just by using the removal process to manipulate the distribution of pattern instances over relevancy within the hierarchy, we are doing the same thing: We are expecting information that relates to what we already know to be relevant to have a higher chance of relevancy than information that does not.

8.4.6 How the Hierarchy Should Develop

8.4.6.1 How the Structure of the Hierarchy Changes

The BERP modifies the hierarchy to make it more relevant by removing pattern instances. Pattern instances are selected for removal according to their relevance values, so that greater numbers of higher-relevance pattern instances are allowed to remain. The BERP also allows the addition of new pattern instances by patterns, with more new pattern instances being connected to the hierarchy where it is more relevant. The BERP does allow lower-relevance pattern instances to exist in the hierarchy: It just allows fewer of them.

The idea of this process is that the structure of the hierarchy will be continually changing. Some of this change will be due to the pattern instances themselves moving from the future to the past. A particular, future pattern instance might correspond to future input of an EFS value, so may have high relevance due to its use in the action selection process. However, this relevance comes from it being at some point in the future. As time passes, the instant to which that pattern instance corresponds will get closer to the present and some other pattern instance which is still far enough in the future will have to take its place. It will therefore lose its relevance while another pattern instance gains relevance. The BERP will be continually changing the hierarchy to "keep up", as well as to deal with changes in the kind of situation in which the AI system finds itself. All the time, the BERP is reducing it where it has low-relevance and extending it where it has most relevance.

We might also think of this in terms of *local density* of the hierarchy. The hierarchy is not absolutely prevented from going into regions that give it low-relevance, but the BERP will tend to make its density low in low-relevance regions, and high in high-relevance regions. Low-relevance regions of the hierarchy will tend to be sparse and skeletal, whereas high-relevance regions will be densely populated by a complex network of pattern instances.

8.4.6.2 Exploration is not random.

An important feature of all this is that the exploratory nature is not random. The hierarchy is not just being randomly extended in the hope that it finds some relevance. The hierarchy gets extended most from where it is already relevant, and this means that high relevance in the hierarchy causes more local exploration there. If that exploration finds more high-relevance pattern instances then the local density of the hierarchy will increase, in turn causing more exploration there.

More exploration, looking for high relevance, occurs "near" high-relevance pattern instances, because the hierarchy is arranged to have more of them in it. A pattern gets high relevance by being on a "high-relevance path" (actually, probably by being on many such paths) between the pattern instances corresponding to previous inputs/outputs and the important, predicted future pattern instances. When the exploratory process finds high-relevance pattern instances it is finding such high-relevance paths. When it finds such paths, the increase in local hierarchy density will cause more exploration locally, so finding high relevance paths will tend to cause more exploration where they are found. That exploration in turn may find further high-relevance paths, and so on. The whole point of this is that finding high relevancy, by affecting the local density of the hierarchy, guides the BERP to extend the hierarchy still further where it found it.

This has some similarity with the exploratory way in which the streamer of a lightning bolt obtains a high conductivity path to the ground. When the streamer encounters air with higher conductivity, it flows into there, which in turn increases the exploration for a path to the ground in that region.

One way of thinking about the low-relevance pattern instances is in terms of "tripwires". The hierarchy will spread out a network of paths featuring low density, low relevance pattern instances, and these will play little part in any of its predictions. Occasionally, something may happen that causes the relevance of some of these pattern instances to increase. This causes the density of the hierarchy to increase locally, with more exploration occurring and more paths being generated locally. This low density, low-relevance part of the hierarchy has acted as a tripwire, ready to detect something interesting impinging on that part of the world view.

8.4.7 Use of the Relevance Measurement Process by the Basic, Exploratory Relevance Process

The BERP is an exploratory process that extends or reduces the hierarchy based on the relevance of pattern instances in it, but it requires the pattern instances in the hierarchy to be assigned relevance values. This is done by the relevance measurement process (RMP), previously discussed in Section 8.3: The Relevance Measurement Process, on

page 53.[42] The RMP started with particular, bottom-level pattern instances corresponding to future inputs/outputs, and which had been assigned relevance with reference to something outside the hierarchy. Relevance was propagated back from these pattern instances through the hierarchy, tracing back along the flow of information which determined their probability values.

For this to work, the pattern instances at the start of the back-propagation process, from which all the relevance propagates, need to be assigned relevance. This must be done with reference to something outside the hierarchy. This is made possible by the way in which the system plans actions. The system's actions are selected by the action selection process discussed previously in Section 5: Planning of Actions, on page 32.[43] In this process, an evaluation function score (EFS) is continually computed, based on recent inputs. Each EFS value is treated as if it were an external input, in that it is encoded as one of more bottom-level pattern instances: It is therefore incorporated into the historical data on which the predictions in the hierarchy are based. When an output is required, the different possible output values (e.g. 0/1) are tried by experimentally updating the corresponding pattern instance(s) as if it had occurred with each value and propagating the effects on probabilities through the hierarchy. The probabilities for one or more pattern instances that will be used for a future input of the EFS are examined, and this gives a probabilistic prediction of the EFS. This allows the merits of each possible output value to be compared, and the output made with the appropriate value.

As was explained in Section 5.3.1: The *real* planning process, on page 34, this is not the real planning process: *That* occurs in the hierarchy itself, when the system models its own behavior and starts to predict a pattern of improving future behavior, which becomes the context in which any individual output is assessed. The purpose of the action selection process described here is to drive the system in a given direction, by establishing a historical pattern of improvement, and to protect against random drift. Nevertheless, the action selection process provides an external reference for assigning relevance to pattern instances. We can say that those pattern instances examined in the action selection process, corresponding to predictions of future EFS inputs are the pattern instances that are relevant, and the system will be functioning best when it is able to produce probabilistic predictions for them which have minimal uncertainty. This would still apply if some more sophisticated version of the action selection process were used; for example, one using multiple EFS predictions. With different versions of the action selection process, the pattern instances to be assigned relevance externally may

[42] Almond, P., 2010. *An Attempt to Generalize AI - Part 6: Measuring Relevance*. [Online] paul-almond.com. http://www.paul-almond.com/AI06.pdf or http://www.paul-almond.com/AI06.doc.
[43] Almond, P., 2010. *An Attempt to Generalize AI - Part 2: Planning and Actions*. [Online] paul-almond.com. http://www.paul-almond.com/AI02.pdf or http://www.paul-almond.com/AI02.doc.

be selected in different ways – in some versions of it maybe even corresponding to near-future outputs – but the general principle would remain the same.

The action selection process is therefore important to the RMP in defining which pattern instances are relevant at any time. The BERP, which in turn uses the RMP, can then continually modify the structure of the hierarchy to minimize uncertainty in prediction of whatever pattern instances are considered relevant by external standards at the time.

8.5 Description of the Basic, Exploratory Relevance Process

8.5.1 Addition of Pattern Instances

Patterns are allowed to extend the actual hierarchy by continually connecting new pattern instances to it. No particular part of the hierarchy is favored. Each pattern is expected to follow its construction specification when adding new pattern instances, but apart from this requirement, a pattern can connect a new pattern instance so that its pattern inputs come from any part of the actual hierarchy.[44]

A single pattern instance may be required to have all its pattern inputs coming from the same "region" of the hierarchy; for example, the pattern inputs for a pattern instance may all be required to be within some given logical distance of each other. This, or some similar rule (possibly some statistical version of it in which pattern instances with all their pattern inputs near each other, in logical distance terms, would be more likely) would be to reduce the number of options available for connecting a pattern instance to something manageable.

8.5.2 Removal of Pattern Instances

Pattern instances are continually removed from the hierarchy. This is in a continuous way, based on the *relevance values* of pattern instances. The lower some relevance value, the less the representation we should want of it in the hierarchy.

There is no cutoff relevance at which pattern instances are not allowed in the hierarchy: A pattern instance's relevance value just determines the likelihood that it will be allowed to remain when the hierarchy is being "pruned". The higher some amount of relevance is, the greater the proportion of the population we will want with that relevance value. The idea is to select pattern instances for high relevance, by removing pattern instances so that the pattern instances with the highest relevance values are the most common, followed by those with slightly less relevance, followed by those with

[44] This is assuming a "constructive" view of patterns, which we are doing for most of this article.

slightly less relevance, and so on, until, ultimately, we reach the pattern instances with the very lowest relevance values and these are the least common in the hierarchy.[45]

8.5.3 Ghost Pattern Instances

When a pattern instance is removed, it is not necessarily removed completely, immediately. Instead, a pattern instance, on "removal" can become a *ghost pattern instance*. A ghost pattern instance has its probability value retained. Probabilistic information can no longer be propagated into a ghost pattern instance: A ghost pattern instance's probability value can never be changed. It can no longer propagate any relevance. However, it can still affect other pattern instances through propagation of probabilistic information in logic application or statistics application.

If the BERP, or some other ERP, makes the decision to remove a pattern instance, and the pattern instance is not directly affecting any other pattern instance through propagation of probabilistic information (and other ghost pattern instances do not count – they cannot be affected, anyway), the pattern instance is removed conventionally, and all information about it is removed from the hierarchy.

If the BERP, or some other ERP, makes the decision to remove a pattern instance, and the pattern instance *is* directly affecting at least one other pattern instance (and it needs to be one that is not a ghost pattern instance), then a full removal does not occur immediately. Instead, the pattern instance becomes a ghost pattern instance, with its probability value remaining in the hierarchy, still able to influence other pattern instances through propagation.

If this situation changes, later, so that the ghost pattern instance is no longer directly affecting another pattern instance through propagation, the removal is completed and the probability value is removed as well: Nothing now remains of the pattern instance.

It may seem that the removal of pattern instances will be made much less effective, as ghost pattern instances would still need dealing with as probability values, meaning that computing resources have not been completely freed up by the removal. This would not cause severe problems. The BERP, or ERP, would be removing huge numbers of pattern instances. A pattern instance may persist as a ghost for a while, because it is directly affecting some pattern instance that is still active in the hierarchy, but it would not be propagating any relevance back to *other* pattern instances that are not directly affecting active pattern instances, making it more likely that they *will* be removed. Also, because a ghost pattern instance cannot change, another pattern instance cannot linger as a ghost by affecting it. A pattern instance can only linger as a ghost, after being removed,

---

[45] Almond, P., 2010. *An Attempt to Generalize AI - Part 7: A Basic, Exploratory Relevance Process*. [Online] paul-almond.com. http://www.paul-almond.com/AI07.pdf or http://www.paul-almond.com/AI07.doc.

if it is logically "near" the actual hierarchy, so a relatively small number of pattern instances may be lingering after removal near the edges of the actual hierarchy at any time, but the ones further out will tend to have been removed *properly* due to lack of relevance.

Ghost pattern instances will be particularly important in forgetting – discussed later in Section 10: Forgetting, on page 77.

8.6 Relevance and Patterns

8.6.1 The General Idea

The way that ERPs work, as just described in 8.5, deals with putting *pattern instances* into the hierarchy and removing them, so that the hierarchy grows into the regions where it should be most dense, but pattern instances are grouped into sets known as *patterns*, and there is also the issue of deciding what patterns to use. This will now be discussed.[46]

As a placeholder, it has generally been assumed in this article that patterns would be "constructive", in the sense that each pattern would control the connection of new pattern instances belonging to it into the hierarchy, as described in Section 2.2.3: The Construction Specification of a Pattern, on page 16. With patterns working like this, when pattern instances are added in the ERP, they would have to be added by patterns. The issue then is one of *which* patterns, out of the infinite set of possible patterns, should get to add pattern instances in the ERP.

The method used to generate the patterns will be involved with this. In Section 2.2.4: Generation of Pattern Specifications, on page 19, I stated that trial and error would play a big part in the generation of pattern specifications, that some Darwinian element may be involved, and that, while this may seem to be asking a lot of a trial and error process, not very much intelligence is being demanded from patterns: Each pattern is only supposed to expose a statistically interesting relationship between its pattern instances.

I will not be trying to describe exactly how patterns are generated in this article. That would depend on how the pattern construction works. Having each pattern directing the wiring of its pattern instances into the hierarchy, as previously described in Section 2.2.3: The Construction Specification of a Pattern, on page 16, is only one way of doing it. Different ways in which pattern construction specifications could work are discussed later, in Section 12.3: The Possibility of Different Construction Methods, on page 92.[47]

---

[46] Almond, P., 2010. *An Attempt to Generalize AI - Part 12: Pattern Relevance*. [Online] paul-almond.com. http://www.paul-almond.com/AI12.pdf or http://www.paul-almond.com/AI12.doc. pp.9-11.

[47] Almond, P., 2010. *An Attempt to Generalize AI - Part 10: Alternatives for Pattern Instance Construction*. [Online] paul-almond.com. http://www.paul-almond.com/AI0.pdf or

We do not need to know exactly how patterns would work, or how they would be generated, to have some idea of how we would select patterns for use once they have been generated: We just need to know how to assess the relevance of a pattern. Fitting a way of doing this in with the ERP, discussed previously, is not very complicated.

The ERP works on the basis of relevance being assigned externally to particular pattern instances corresponding to future inputs: pattern instances which will be used in the action selection process for future input of the evaluation function score (EFS). A pattern can be assessed according to how it affects the uncertainty of the same pattern instances. If a pattern is not is not in use – meaning no pattern instances associated with that pattern are active – and the pattern is then activated, the change in the uncertainties of some pattern instances corresponding to future input of the EFS can be examined, and this would indicate the benefit that the pattern was providing. All else being equal, a pattern which caused the uncertainty in the pattern instances of interest to decrease a lot – meaning that their probability values moved away from 0.5 – would be considered very useful. This might not be the only thing that would be considered, however: The amount of computing power consumed by the pattern might be taken into account and weighed against the reduction in uncertainty, to give an overall indication of usefulness of the pattern.

When a new pattern is generated in some trial and error process, its desirability, in terms of reducing uncertainty in the bottom-level pattern instances involved in the action selection process, would be assessed. A limited pool of the most desirable patterns would be maintained, and these would be available for use in the ERP, with their pattern instances continually being added to the system, and removed from it, in the ERP. If the new pattern's desirability were high enough, it would be added to the pool, displacing a pattern already in it, and it would remain there until it was itself displaced by more desirable patterns.

An approach like this could be used if pattern instances were being generated randomly, but it could also be used if Darwinian evolution were used to generate patterns: Instead of generating patterns completely randomly, new patterns could be generated by variation, meaning mutation or crossover, of those already in the pool, and the desirability of these patterns would be assessed in the same way.

This kind of approach should be considered part of the ERP.

8.6.2 More Sophistication in Achieving Pattern Relevance

A more sophisticated version of the above approach might involve patterns having variable levels of activity. The patterns considered most effective, in terms of reducing uncertainty in pattern instances corresponding to future EFS input, would be kept in a

pool, but they would not have equal status. The more effective a pattern, in terms of reducing uncertainty, the more pattern instances it could construct. The approach may also involve varying the activity of pattern instances over time. The best-performing patterns may be kept in the pool, but the effectiveness of each may vary over time, and as this occurs the extent to which it is used will vary. The ERP might also vary the level of activity of different patterns in different regions of the hierarchy. If pattern instances of a particular pattern, in a particular region of the hierarchy, seem to significantly decrease uncertainty in the pattern instances corresponding to future EFS input, that pattern may be allowed to insert more pattern instances in that region. It should be noted that the removal of pattern instances in the EFS achieves this ultimately anyway: If the pattern instances of a pattern are effective in some region of the hierarchy, they will tend to stay there, and if not they will tend to be removed.

8.7 A General View of the Exploratory Relevance Process

I will now give some consideration to what is *really* going on in the BERP or some other ERP.

8.7.1 The ERP is exploratory.

When new pattern instances are added to the hierarchy, they are connected to pattern instances that have already survived previous pruning, so they are being added to parts of the hierarchy that are already particularly relevant. It does not *guarantee* that they will be relevant too, but the chances are increased a lot. The process is an exploratory one.

This might be thought of as being similar to growth of plant roots, as they seek out water and nutrients for a plant. Once a root has grown into some region and found suitable amounts of these, it makes sense to extend more roots out to explore from there. The process is therefore not totally random: It builds on what has been found. However, this analogy does not give a complete picture, as will now be described.

8.7.2 The ERP is *directed* by the hierarchy.

The above analogy of the growth of plant roots omits an important feature of the BERP or some other ERP. With plant roots, whether the tip of a given root will obtain water is mainly based just on where it is: The rest of the structure does not matter much. The actual hierarchy is not like this. In the actual hierarchy, the amount of relevance that a pattern instance gets does not depend just on "where" it is. Relevance is assigned to a pattern instance by back propagation in the RMP, and this measures the effects that a pattern instance has on reduction of uncertainty of the bottom-level pattern instances that are of interest in the action selection process. A pattern's relevance only makes sense, however, within the context of the rest of the actual hierarchy. For a pattern instance to have high relevance, the following are needed.

1. The rest of the hierarchy, which propagates probabilistic information into the pattern instance in logical application or statistics application, propagates a significant amount of information into the pattern instance.
2. The rest of the hierarchy, into which the pattern instance propagates probabilistic information in logical application or statistics application, causes the uncertainty of the bottom-level pattern instances, corresponding to future EFS input, that are of interest in the action selection process to decrease significantly, as a result of this information being propagated into it.

The first condition applies because, if hardly anything is known about a pattern instance, what we know about it cannot significantly affect anything else. It is a prerequisite for a pattern instance to affect other pattern instances significantly that its own uncertainty should be reduced significantly.

The second condition applies because the information propagated into the hierarchy by a pattern instance does not *directly* affect the predictions that are to be made. Instead, a pattern instance affects other pattern instances, which affect others and so on. Ultimately, this chain of propagation affects the bottom-level pattern instances corresponding to future EFS predictions. Whether or not the information put into the hierarchy by a pattern instance is relevant depends on what the rest of the hierarchy does with this information – what happens in the rest of the chain.

The relevance of a pattern instance comes from *context*. The relevance of a pattern instance depends on the pattern instance and what is going on around it – what information is being propagated into it and what is happening to the information propagated out of it. This means that in the exploratory growth of the hierarchy in the EFS, the rest of the hierarchy imposes requirements that must be met for any single pattern instance to have high relevance, or for the pattern instances in any given region to have high relevance. A pattern instance can only have high relevance if it "fits in" with the rest of the hierarchy – if it receives significant information from it and puts information into it which causes the hierarchy as a whole to make useful predictions. Suppose that a new pattern instance is connected into the hierarchy as part of the ERP. If the pattern instance "fits in" with the rest of the hierarchy it will have high relevance, and will be likely to survive the later pruning of the hierarchy, but if it does not "fit in" with the rest of the hierarchy it will have low relevance and will be likely to be removed very quickly. The hierarchy is determining the properties that new pattern instances need to survive in it for any significant period of time, and it is directing its own growth along particular paths – quickly filtering out growth in directions that are not along these paths. The hierarchy can be viewed as *directing* its own growth.

This article has discussed dealing with patterns in the BERP or some other ERP, so that the ERP includes processes to select a number of preferred patterns from all the patterns that have been tried, and possibly to give different degrees of preference to different patterns in some way. This view of the hierarchy as directing its own growth is

relevant here as well. Suppose a new pattern is generated and experimentally applied in the hierarchy. If it fits in well with the rest of the hierarchy, and achieves a significant reduction in the uncertainty of the hierarchy's predictions of future EFS values, then the pattern will remain in the hierarchy: otherwise it will be removed soon.

It is not all about pattern instances and patterns having requirements imposed on them by the rest of the hierarchy. A new pattern could be added to the hierarchy that generates a large number of high-level pattern instances that dictate the requirements for relevance of the pattern instances between them, so the new pattern itself can be imposing what we might think of as hills and valleys in the landscape of the hierarchy beneath it, directing the addition of the pattern instances that are propagating information upwards.

8.7.3 High relevance is not everything.

In the BERP, or some other ERP, the continual pruning could remove all but the most relevant pattern instances, but this would be inadvisable: There might be parts of the hierarchy of lower relevance that could become relevant later, and growing the hierarchy step by step into these regions when they start to become relevant could become an impractically slow process of crossing low-relevance gulfs. I have previously discussed this issue in Section 8.4.3: The Objective of the Basic, Exploratory Relevance Process, on page 57, and suggested dealing with it by merely *favoring* pattern instances with high relevance to some (limited) degree, so that each pattern instance's "life expectancy" in the hierarchy depends on its relevance.[48]

The result is a hierarchy that varies in density, rather than one which leaves entire regions entirely deserted. The hierarchy will extend lower-relevance "tendrils" out and these can be considered as being like tripwires, as previously described in Section 8.4.6.2: Exploration is not random., on page 61.[49]

[48] Almond, P., 2010. *An Attempt to Generalize AI - Part 7: A Basic, Exploratory Relevance Process.* [Online] paul-almond.com. http://www.paul-almond.com/AI07.pdf or http://www.paul-almond.com/AI07.doc. pp.6-7.

[49] Almond, P., 2010. *An Attempt to Generalize AI - Part 7: A Basic, Exploratory Relevance Process.* [Online] paul-almond.com. http://www.paul-almond.com/AI07.pdf or http://www.paul-almond.com/AI07.doc. p.10.

9 Possible Improvements to the Basic, Exploratory Relevance Process

9.1 The General Idea of Possible Improvements to the Basic, Exploratory Relevance Process

The basic, exploratory relevance process (BERP), previously discussed in Section 8.4: The Basic, Exploratory Relevance Process (BERP), on page 57, is intended to be the simplest viable method of ensuring relevance in the hierarchy.[50] There are various ways in which the BERP might be improved to give a more sophisticated exploratory relevance process (ERP), and some of these will now be discussed.[51]

9.2 Update Frequency Dependent on Relevance

9.2.1 The General Idea of Update Frequency Dependence on Relevance

The BERP achieves relevance by removing pattern instances, with the relevance of a pattern instance determining its likelihood of remaining in the hierarchy: The lower the relevance of a pattern instance, the more likely it is that it is removed at any time.

A more sophisticated ERP might treat pattern instances differently with regard to propagation of information, with propagation occurring less frequently for low-relevance pattern instances. The idea of this would be that a lower-relevance pattern instance is less important, so we should not be spending as much computing power on it. This could apply to probabilistic propagation through the hierarchy and relevance back-propagation.

9.2.2 Update Frequency for Probabilistic Propagation Dependent on Relevance

Probabilistic information is propagated through the hierarchy by the processes of logic application and statistics application.[52] Provided that a pattern instance exists in the actual hierarchy, it is updated as often as any other pattern instance. In a more

---

[50] Almond, P., 2010. *An Attempt to Generalize AI - Part 7: A Basic, Exploratory Relevance Process*. [Online] paul-almond.com. http://www.paul-almond.com/AI07.pdf or http://www.paul-almond.com/AI07.doc. pp.13-14.

[51] Almond, P., 2010. *An Attempt to Generalize AI - Part 9: Improving the Exploratory Relevance Process*. [Online] paul-almond.com. http://www.paul-almond.com/AI09.pdf or http://www.paul-almond.com/AI09.doc.

[52] Almond, P., 2010. *An Attempt to Generalize AI - Part 5: A Completely Probabilistic Hierarchy*. [Online] paul-almond.com. http://www.paul-almond.com/AI05.pdf or http://www.paul-almond.com/AI05.doc. pp.21-29.

sophisticated ERP, the frequency with which pattern instances are updated in probabilistic propagation may take account of relevance, with frequency of update decreasing as relevance decreases. High-relevance regions of the hierarchy would be updated often, but low-relevance regions might be updated infrequently.

This could mean that probabilistic information is propagated *from* a pattern instance with a frequency dependent on its relevance, so that low-relevance pattern instances affect others by probabilistic propagation less often, or it could mean that probabilistic information is propagated *into* a pattern instance with a frequency dependent on its relevance, so that low-relevance pattern instances are affected by probabilistic propagation from others less often.

9.2.3 Update Frequency for Relevance Back-Propagation Dependent on Relevance

This is similar to what was just discussed for probabilistic propagation, in 9.2.2, but it is applied to relevance. The frequency with which pattern instances are updated in relevance back-propagation may depend on relevance, with frequency of update decreasing as relevance decreases. High-relevance regions of the hierarchy would be updated with new relevance values very often, but low-relevance regions might be updated very infrequently.

This could mean that relevance is back-propagated *from* a pattern instance with a frequency dependent on its relevance, so that low-relevance pattern instances affect others by relevance back-propagation less often, or it could mean that relevance is back-propagated *into* a pattern instance with a frequency dependent on its relevance, so that low-relevance pattern instances are affected by relevance back-propagation from others less often.

9.2.4 Long and Short-Term Relevance

An obvious way to use approaches like those just described, in 9.2.1, to 9.2.3, is to consider the relevance for different timescales when removing pattern instances in the ERP and when determining what their update frequency in probabilistic/relevance propagation is.

- The ERP, when it prunes the hierarchy, may base its decisions on relevance over a *long* period of time, so that the chance that any pattern instance is removed is based on the average of its relevance over a long period of time: The lower this value is, the more likely a pattern instance would be to be removed at any time.
- The frequency with which pattern instances are updated, in probabilistic propagation or relevance back-propagation, may be based on relevance values averaged over a *short* period of time – or even on single relevance values: The lower the average of the recent values of a pattern instance's relevance is, the

lower would be the frequency with which that pattern instance is updated or updates others.[53]

9.2.5 Will this be of any use?

Whether it is *useful* to vary the update frequencies for pattern instances depends on how the computing hardware is put together.

- In some systems, all the storage and processing associated with a single pattern instance might be provided by hardware dedicated to that pattern instance, so that none of the computing resources used by a pattern instance, while it still exists, can be made available to any other pattern instance. That is to say, "storing" a pattern instance would use as much computing power as "running" it. In such systems, approaches like the ones just discussed, in 9.2.1, to 9.2.4, would be useless, because the computing resources freed when a pattern instance was not being updated, or updating others, would not be available for any other pattern instances.
- In other systems, some of the computing resources used in propagation by a pattern instance might be made available to others when the pattern instance was not being updated or updating others. That is to say, just "storing" a pattern instance would use less computing power than was needed to "run" it. In such a system, approaches like those just discussed could be more useful, with pattern instances having computing resources gradually withdrawn as they become less relevant, ultimately being removed completely if the ERP makes the decision to do this.

At one extreme, with a massively parallel computer, in which a computational element was assigned to each pattern instance, and specific hardware was dedicated to each connection between pattern instances, with pattern instances not existing in any other way, approaches like those discussed would have no benefit.

At another extreme, with a computer built according to the Von Neumann architecture, with information about the pattern instances and their connections being stored in memory, separate from processing, and propagation through the hierarchy being done by sequential processing, approaches like those just discussed would significantly reduce the processing time needed for infrequently updated pattern instances, although the same amount of memory would still be required to store the information about them.

A real computer might be somewhere between these two extremes. It may have many components working in parallel, but there may be some way in which a less frequently updated pattern instance is using fewer resources. For example, pattern instances might have dedicated computational units to store their states, while the processing to

[53] This could be done for one or both of probabilistic propagation and relevance back-propagation. Different timescales might be used in each case.

propagate information between pattern instances is performed on hardware that is shared among "local" pattern instances.

9.3 Pattern Instance Addition *Explicitly* Dependent on Relevance

9.3.1 The Basic Approach

The idea of the BERP is that the density of the actual hierarchy is reduced where it seems insufficiently relevant to justify a high density, and increased where it seems particularly relevant. As part of achieving this, new pattern instances should generally be added in higher-relevance regions of the actual hierarchy, to cause it to "grow" into high-relevance regions. The BERP already does this, because the pruning by the BERP already selects the pattern instances of the actual hierarchy for high relevance, and pattern instances are connected to the actual hierarchy in random regions. As the higher-relevance regions of the actual hierarchy will be denser than low-relevance regions, it is will be more likely that new pattern instances get connected here.

9.3.2 Clarification

Some clarification about the previous discussion on pattern instance addition may be useful here. In a previous article, I stated the following about addition of new pattern instances.

"As with the removal of pattern instances, this should be done in a continuous way, so that new pattern instances are connected more often to higher-relevance regions of the hierarchy than to lower-relevance ones.

We do not need to go to any trouble to achieve this: The removal of pattern instances by the BERP … will actually ensure that this happens anyway, if we just allow new pattern instances to be connected to the hierarchy with equal frequency everywhere, or randomly. This is because the BERP's actions in removing pattern instances based on lack of relevancy will mean that the distribution of pattern instances in the hierarchy will be weighted towards higher-relevancy ones." [54]

There does not need to be any preference about where in the actual hierarchy new pattern instances are connected, because the hierarchy's pattern instances have already been selected for high relevance: If a region of the hierarchy was really bad for adding new pattern instances, the hierarchy would probably have withdrawn from it, or would be of such low density there that new pattern instances would be rarely added there. For that reason, we can add pattern instances at random "places" in the hierarchy.

---

[54] Almond, P., 2010. *An Attempt to Generalize AI - Part 7: A Basic, Exploratory Relevance Process*. [Online] paul-almond.com. http://www.paul-almond.com/AI07.pdf or http://www.paul-almond.com/AI07.doc. p.7.

Some qualification of this is needed. It might be taken as implying that all the labeled pattern inputs of a new pattern instance are connected randomly, so that there is no organization at all in how a pattern instance is connected. This would not be correct: The way in which the labeled pattern inputs of a pattern instance are connected is controlled by the construction specification. It would be more accurate to understand this as meaning that there is no preferred part of the hierarchy in which to connect pattern instances, even though the use of particular pattern inputs for some pattern inputs implies that others will be used for other pattern inputs.

This might be done in a number of ways. One way would involve randomly selecting a pattern instance in the actual hierarchy at random. The construction specification for the relevant pattern would then select a pattern instance as near as possible to it as the first labeled pattern input. Once the first labeled pattern input had been connected, the construction specification would then control selection of the pattern instances to be used for the other pattern inputs. The result would be a pattern instance connected in a random region of the hierarchy, but with the construction specification still controlling individual selection of pattern instances for pattern inputs. The pattern instances serving as pattern inputs of a single pattern instance might be required to be logically "near" each other most of the time.[55]

9.3.3 A Possible Change

As just discussed, in 9.3.1 and 9.3.2, the BERP will automatically tend to add pattern instances in high-relevance regions of the hierarchy, because these are the ones that will be most represented in the actual hierarchy. It may be desirable, however, to exaggerate this, with the ERP *explicitly* taking account of relevance when adding new pattern instances.

If such an approach is used, when the ERP is adding a new pattern instance, the relevance values of pattern instances will be examined, and the probability that any single pattern instance is used as the point where the new pattern instance is connected increases as its relevance increases. Exactly how this works will depend on how the construction specification functions, but the general idea should be that preference is given to high-relevance pattern instances when it comes to connecting new ones. This could involve an approach similar to that just discussed, in 9.3.2, with some modification to prioritize pattern instances according to relevance.

This could also be combined with approaches in which update frequency is based on relevance, as previously discussed in Section 9.2: Update Frequency Dependent on Relevance, on page 70: As well as the update frequency for probabilistic propagation or

[55] using a non-spatial understanding of the word "near", as previously discussed in Section 7.2: Weak Geometry, on page 45.

relevance back-propagation depending on relevance, the frequency with which pattern instances are added in a given region of the hierarchy could also depend on relevance.

It is also possible to examine relevance averaged over different periods of time when adding and removing pattern instances, with the removal of pattern instances being based on relevance averaged over a longer period of time than it is for pattern instance addition. Whether this would be useful is debatable.

9.4 Relevance Computed for *Regions* of the Hierarchy

The BERP examines relevance values for single pattern instances only, but some of the processes occurring as part of the BERP might be more effective if they take account of the relevance for a *region* of the hierarchy, instead. As previously stated in Section 7: Geometry, on page 44, the hierarchy is not spatial, but the idea of a "region" could still be applied to it by considering the pattern instances which are logically "near" a particular pattern instance. For example, to obtain the relevance of a region we might obtain the relevance of some pattern instance, then take less account of the relevance of those logically closest to it, and then still less account of the relevance of those next closest to it, and so on, combining all this information to obtain a single relevance value.

This could be done when determining update frequency for probabilistic propagation or relevance back-propagation, when removing pattern instances or when adding pattern instances; however, the argument for doing it may not be very strong in all these cases.

The argument for taking account of regional relevance might be strongest in relation to addition of new pattern instances. The idea, when adding a new pattern instance, should be for it to have a high relevance. After it has been added, the ERP can examine it and remove it (or make it more likely that it will be removed) if its relevance is too low, but by then, in one respect, it is too late: Computing resources have already been spent on adding the pattern. Adding patterns randomly this way, and hoping to find high-relevance ones occasionally, would be too wasteful. The exploratory nature of the process already deals with this: The previous removal of low-relevance pattern instances means that any addition of a pattern instance is more likely to occur in a higher-relevance region of the hierarchy. I have just stated that we may want to give more explicit consideration to relevance when adding pattern instances, in 9.3, but while the relevance of a single pattern instance may be a useful guide, it will not be a perfect indicator of the relevance that a new pattern instance will have if we connect it logically nearby. Relevance will vary from pattern instance to pattern instance, and we may see that the relevance of a pattern instance is high, and connect a new pattern instance logically nearby, expecting that to have high-relevance too, and find out that it has low-relevance. We do not need to get this right every time, as the ERP's pruning will remove failures, but we should want to get it right as often as possible. One way of improving this may be to look at the relevance of a region when adding new pattern

instances, to obtain a more reliable indication of the likely relevance of local pattern instances than would be obtained by examining a single pattern instance.

10 Forgetting

10.1 The General Idea Behind Forgetting

The relevance measurement process (RMP) previously described in Section 8.3: The Relevance Measurement Process, on page 53, removes pattern instances that do not seem to be relevant to reducing uncertainty in the particular, bottom-level pattern instances that are of interest in the action selection process.[56] However, there is also the issue of *obsolescence* of pattern instances: A pattern instance that was useful some time ago may no longer be playing an active role in reducing relevance. This is most obvious for bottom-level pattern instances corresponding to the occurrence of inputs/outputs: It is obvious that it serves no purpose to retain a record of every input/output permanently, and the same can be said for many of the pattern instances with values derived from them. Forgetting will not require a special process. Instead, the RMP is modified so that obsolete pattern instances will receive low relevance scores, so that the basic, exploratory relevance process (BERP), previously discussed in Section 8.4: The Basic, Exploratory Relevance Process (BERP), on page 57, or some other exploratory relevance process (ERP), will tend to remove them.[57]

We will now look at the rationale behind such a modification to the RMP.

10.2 Relevance Propagation and Degree of Uncertainty

10.2.1 The Relevance Measurement Process

In the RMP, particular pattern instances corresponding to future inputs/outputs are regarded as relevant due to their use in the action selection process. Relevance is back-propagated from these pattern instances. A pattern instance propagates a total amount of relevance equal to its own relevance back to those pattern instances that have directly affected its probability value, and it is shared out amongst them in proportion to the amount of effect that they have had on reducing its uncertainty.

We need to think about what all this is supposed to achieve. Whenever the RMP computes relevance, this is based on the probability values that exist in the hierarchy at the time. This means that when a relevance value is obtained for a pattern instance it is not necessarily the relevance that the pattern instance will have *in future*, but is actually

---

[56] Almond, P., 2010. *An Attempt to Generalize AI - Part 6: Measuring Relevance*. [Online] paul-almond.com. http://www.paul-almond.com/AI06.pdf or http://www.paul-almond.com/AI06.doc.
[57] Almond, P., 2010. *An Attempt to Generalize AI - Part 8: Forgetting as Part of the Exploratory Relevance Process*. [Online] paul-almond.com. http://www.paul-almond.com/AI08.pdf or http://www.paul-almond.com/AI08.doc

the relevance that it has *already had* in the propagation of probability values that has already occurred through the hierarchy. This can be justified in two ways, as follows.

- We should expect the relevance of pattern instances to change gradually. Therefore, a pattern instance's relevance to the current situation should be a good indication of its likely relevance in the immediate future.
- When the BERP, or some other ERP, removes pattern instances from the hierarchy, those remaining are not just being left because they may be directly useful. New pattern instances are connected to existing ones in the hierarchy, so the pattern instances selected to remain in the hierarchy are also providing a structure from which the hierarchy will grow, and the new pattern instances are more likely to be highly relevant in the future if the ones to which they are being connected are already known to have high relevance – even if these get removed shortly afterwards.

Both of these show why relevance measurements based on propagation that has already occurred can be a useful guide to which pattern instances will be useful in the future, even if the RMP does not provide a completely accurate prediction of future relevance, but this does not change the fact that we should still want the relevance values to indicate future relevance *as accurately as possible*.

We will now consider the RMP in relation to the special case of a pattern instance for which the value is known with certainty – what was previously known as a "fixed" pattern instance.

10.2.2 The Special Case of a Pattern Instance with No Uncertainty

Suppose in a computer's actual hierarchy there is a single pattern instance, X, in the hierarchy for which, due to the probabilistic information being propagated to it directly by logic application or statistics application from other pattern instances, the probability value is 0 or 1. This pattern instance is known about with certainty. It is now known that the corresponding pattern instance in the conceptual hierarchy has an actual pattern output value (not a probability) of 1. X's probability can never change again: If it did change again, this would imply that the hierarchy's assignment of a probability value indicating certainty to it had been incorrect.

X would have been assigned this probability due to propagation of probabilistic information directly from other pattern instances. In the RMP, X will back-propagate a total amount of relevance equal to its own relevance to these pattern instances, and it will be shared out among them according to the contribution each has made to reducing its uncertainty. A contribution to reducing X's uncertainty has clearly been made, because X's probability has gone to 0 or 1. X will therefore propagate relevance to the pattern instances which directly affected its probability and reduced the uncertainty in it.

The reasoning behind this should be that the pattern instances receiving the relevance, because they are already known to have an important role in reducing uncertainty in X, should be expected to reduce it in future. We might also take the view that pattern instances that we might connect to them in future could reasonably be expected to have similar relevance. However we view this, we should still want to propagate an amount of relevance to each pattern instance, from X, that reflects the future expectations of relevance as accurately as possible.

So, what are our future expectations of relevance here? Relevance is being propagated from X to other pattern instances, but X's probability of 0 or 1 means that X is already known about with certainty. These other pattern instances can never tell us anything useful about X in future because we already know everything! The pattern instances which have received relevance from X will have no role to play in reducing X's uncertainty in future: X has no uncertainty left to reduce. In reality, we could just set X's probability to 0 or 1, and *back-propagate no relevance from X*.

The approach just described is the same as the basic forgetting procedure from the third article, except that it is now just part of the processing to provide relevance in the hierarchy: It is now part of the BERP or some other ERP. The basic forgetting procedure cannot be used in its current form, with the completely probabilistic hierarchy, because hardly any pattern instances will be known about with certainty. This does not change the fact that the approach just described, although impractical due to being usable only in uncommon special cases, would still be valid. This suggests that a generalization of this could be used, and to see how this would work we will now consider back-propagation of relevance from pattern instances which are known about with *almost* complete certainty.

10.2.3 A Pattern Instance with *Almost* Complete Certainty

Suppose there is a situation *almost* the same as the one just discussed, in 10.2.2. In the actual hierarchy there is a single pattern instance, X, in the hierarchy for which the probability value is now *almost* 0 or 1; for example, it may be 0.001 or 0.999. This pattern instance is known about with almost complete certainty. In the previous situation, in 10.2.2, the BERP or other ERP was back-propagating relevance to the pattern instances that had influenced X, but really no relevance should have been back-propagated, because the certainty in X made the other pattern instances redundant. Here, the situation is *almost* the same. If the pattern instances in the previous situation should have been receiving no relevance at all from X, then it makes no sense to say that the pattern instances in this almost identical situation should receive the normal amount of relevance: They will be telling us about a pattern instance about which we already know *almost* everything anyway, so they should be receiving *hardly any* relevance.

For the previous situation, in 10.2.2, I mentioned that when we knew about a pattern instance with certainty we could just permanently set its probability to 0 or 1 and not rely on any further propagation. Here, things need a bit more consideration, but I will return to this issue shortly.

Now that we have considered the case of pattern instances which are known about with almost complete certainty, we can go to the general case of a pattern instance known about with *any* degree of certainty.

10.2.4 A Pattern Instance with *Any* Degree of Certainty

As just discussed, in 10.2.3, when a pattern instance is known about with *almost* complete certainty, it should propagate hardly any relevance back because the value of any more probabilistic information that it receives in the future is reduced. There is no dividing line, however, at which the degree of certainty becomes "almost complete". If we can make a case for reducing relevance back-propagation from a pattern instance known about with almost complete certainty, as just discussed in 10.2.3, it follows that the total amount of reference propagated back from some pattern instance, X, should be affected by the degree of certainty for any probability that X could have, the full amount of X's relevance being available for propagation only when nothing is known about X (when its probability value is 0.5), and the total amount of relevance being propagated being reduced as the degree of certainty about X increases: As X's probability gets closer to 0 or 1, the total amount of relevance that it propagates back should be reduced.

This now gives us the basis for an approach to forgetting which is just part of the processing that the BERP, or some other ERP, performs to maintain relevance in the hierarchy.

10.3 Modifying the Relevance Measurement Process to Take Account of Degree of Certainty

10.3.1 The Basic Modification

To provide forgetting, the RMP, previously described in Section 8.3: The Relevance Measurement Process, on page 53 [58] is modified as follows.

10.3.1.1 Total Amount of Relevance

When relevance is propagated from some pattern instance, X, to those pattern instances which have directly affected its probability value,[59] the *maximum* total

[58] Almond, P., 2010. *An Attempt to Generalize AI - Part 6: Measuring Relevance*. [Online] paul-almond.com. http://www.paul-almond.com/AI06.pdf or http://www.paul-almond.com/AI06.doc. pp.16-19.

amount of relevance that will be propagated is the same as X's own relevance; however this is adjusted to take account of the degree of certainty in X. The closer X is to being known about with complete certainty – the closer its probability value is to 0 or 1 – then the lower is the total amount of relevance that X back-propagates.

- For a pattern instance with a **probability of 0.5**, the total amount of relevance that is back-propagated is equal to X's relevance.
- For a pattern instance with a **probability of 0 or 1**, no relevance is back-propagated.
- For a pattern instance with **some other probability value between these values**, the amount of relevance back-propagated is some fraction of X's relevance, the amount decreasing as the probability gets closer to 0 or 1.

10.3.1.2 Allocation of Relevance

The relevance that is being back propagated is shared out among the individual pattern instances which are considered relevant to X, each receiving a proportion of the relevance in accordance with the extent to which it reduces uncertainty in X. This is done as in the previous description of the RMP: The only difference is that the *total amount* of relevance available for propagation is modified as just described.

10.3.2 The Role of Ghost Pattern Instances in Preventing Loss of Information

The purpose of the modification to the RMP just described, in 10.3.1, is to reduce the relevance values of pattern instances that are obsolete, so that the BERP, or whatever ERP is in use, will tend to remove them from the hierarchy. If a pattern instance is known about with a high degree of certainty, little relevance will be back-propagated from it, and this is likely to cause the BERP, or some other ERP, to remove the pattern instances that are causing the high degree of certainty in the first place. With these pattern instances gone, in the absence of ghost pattern instances, if we propagated all the probabilistic information through from the bottom-level again, this would mean that the previous degree of certainty would not be achieved. By removing obsolete pattern instances, we would be removing the very thing that was providing the degree of certainty that justified their removal, and we would lose that degree of certainty without them. This issue is dealt with by the ghost pattern instances.

[59] or degree of uncertainty in the case of more than one such value

10.4 Issues with Forgetting

10.4.1 What if we just remember the probability?

Some readers may wonder why a pattern instance should be back-propagating *any* relevance to the pattern instances that have determined its probability, unless they are *continually* reducing its uncertainty. Could we not just as well store the previous probability that the pattern instances propagated, and does this not mean that any further propagation that failed to reduce uncertainty was redundant, and therefore irrelevant?

For example, suppose after probabilistic propagation through the hierarchy, a pattern instance, X, has a probability of 0.2. The next time any propagation occurs through the hierarchy, X's probability is still 0.2. We already knew that X's probability was 0.2: In the absence of any information from propagation, we would have just assigned it the previous probability value anyway. Given this, should we not say that nothing, beyond its own probability is relevant to X – unless some pattern instances can provide a reduction in uncertainty?

I would not agree with this. The first thing to note is that the hierarchy, although describing a temporal situation, is not a temporal object. In the conceptual hierarchy, which the actual hierarchy is modeling, each pattern instance has a pattern output value and there is no change: The concept of time does not even mean anything. In the actual, hierarchy, in a computer, the values of some of the pattern instances are unknown, because they depend on future inputs/outputs that have not yet occurred, and so have to be described probabilistically. As inputs/outputs occur, more information about the hierarchy becomes available, and probabilities in the actual hierarchy change to reflect this.

This means that if some pattern instances recently helped to reduce the uncertainty in X, but the last time that propagation occurred there was no change, it does not follow that those pattern instances are irrelevant to X: The probability that X has was still determined by those pattern instances. If we took such an approach it would be hard for any pattern instance to have any relevance for any appreciable time: As soon as some pattern instances generated some information in another pattern instance, we could simply refer to that information and rule that, unless these pattern instances had anything else to offer *immediately*, they were irrelevant. This would be ignoring the fact that those pattern instances, by being responsible for X's current probability, might be expected to be particularly relevant to any change in this probability in future – and also that they indicate regions of the hierarchy where we should think it is particularly likely that new, high relevance pattern instances can be added.

It should be noted that the rationale for limiting the amount of relevance back-propagated from a pattern instance based on the degree of certainty, in Section 10.2.4: A Pattern Instance with *Any* Degree of Certainty, on page 80, is entirely different to the

kind of reasoning being discussed here. When relevance is limited like that, it is because the scope for pattern instances to do anything useful is limited in future by the degree of certainty that already exists. A situation in which pattern instances have made some useful contribution and in which there is still scope for them to offer more is not the same.

10.4.2 Isn't this inconsistent?

10.4.2.1 Two Different Standards

There may *seem* to be inconsistency in the way relevance is back-propagated from a pattern instance, as described in Section 10.3.1: The Basic Modification, on page 80.

1. The total relevance back-propagated from a pattern instance, X, decreases as the certainty in the pattern instance increases, implying that the pattern instances receiving the relevance are penalized for achieving a high degree of certainty in X, and this may seem to imply that we *do not want* a high degree of certainty in X.

2. The proportion of the total relevance back-propagated from X to an individual pattern instance increases as the contribution of that pattern instance to achieving a high degree of certainty in X increases, implying that the pattern instances receiving the relevance are rewarded for achieving a high degree of certainty in X, and this seems to imply that we *want* a high degree of certainty in X.

10.4.2.2 Why there is No Inconsistency

There is no inconsistency here, because each of the cases just described, in 10.4.2.1, is actually referring to a different thing. One refers to a group of individual pattern instances and the other refers to the group.

- In (1) it is the *group* of pattern instances directly affecting X that has its reward reduced for achieving a high degree of certainty in X, by having the total amount of relevance reduced as this degree of certainty increases.
- In (2) it is *individual* pattern instances that are rewarded for helping to achieve a high degree of certainty in X, by giving each a proportion of the total relevance given to the group that increases as the degree of certainty that it achieves increases.

There is nothing inconsistent about this. As the certainty in X increases, the benefit of any propagation into X is reduced, so less total relevance is back-propagated; however, we still want to share out what relevance is back-propagated according to merit, with each pattern instance receiving a share according to the contribution that it makes.

10.4.2.3 An Analogy

As an analogy, suppose we are manufacturing some product that we want to improve, so that it will be more commercially successful. Maybe we want more people to buy it. Maybe we want it to be more reliable, so we have fewer returns. Maybe we want it cheaper to manufacture. Whatever the specific objective, the general idea is that we want the product to earn us more money.

We offer reward money to our employees for thinking of ways to improve the product. Each month, a prize fund is made available, and the money is given to employees who made useful suggestions for improving the product that month. The proportion of that month's prize money that any single employee receives depends on how good his/her idea is: Employees whose improvements to the product earn us a lot of money receive a lot of that month's prize money as a reward.

As time passes, the product becomes harder to improve. The easier improvements have been made earlier on, and it is hard for anyone to think up really significant, new ones. The product design is becoming *optimized*. The returns from making further improvements start to diminish: The total value of the ideas we receive each month tends to be less than it was when we started. We reduce the prize fund for each month, so less money will be paid out each month. What we are prepared to give to the group is reduced to reflect the reduced scope for improving the product. However, this does not change the fact that we are still going to share that money among individual employees according to what each has contributed.

10.4.2.4 Relevance is about allocation of computing resources.

The analogy just given should have shown that, just as there is no inconsistency in what we are *doing* here, there is no inconsistency in what we *want* either. What we want is *always* a high degree of certainty in pattern instances. When we reduce the total amount of relevance back-propagated from some pattern instance, dependent on its degree of certainty, this does not mean that we do not want a high degree of certainty. All it means is that the degree of certainty already there limits the amount of extra certainty that we can get in future, so we are limiting the computing resources that will be spent on trying to get that certainty so that more are available elsewhere. The amount of relevance being provided is what ultimately determines resource allocation, by controlling the density of the hierarchy.

10.5 Bottom-Level Pattern Instances and Forgetting

The issue of bottom-level pattern instances still needs consideration with regard to any ERP that could cause their removal, as was stated in a previous article as follows. Bottom-level pattern instances will tend to be removed by any forgetting process, but some will tend to be removed in the existing pruning by the BERP. Normally, when pattern instances are removed, they can be added again if needed: The hierarchy can be

reduced in some region where it is insufficiently relevant, and later regrown if that region becomes relevant again. The information in the hierarchy is based on bottom-level pattern instances corresponding to previous inputs/outputs. If these are removed, the information in them is lost from the hierarchy. It may be advisable to make the conditions for removal of such pattern instances more demanding than for others.[60]

[60] Almond, P., 2010. *An Attempt to Generalize AI - Part 7: A Basic, Exploratory Relevance Process*. [Online] paul-almond.com. http://www.paul-almond.com/AI07.pdf or http://www.paul-almond.com/AI07.doc. p.17.

11 Reflexive Outputs as a Way of Providing Relevance

11.1 The Idea of Reflexive Outputs

The ways in which the relevance measurement process (RMP) and the basic, exploratory relevance process (BERP) or some other exploratory relevance process (ERP) could provide relevance in the hierarchy have been discussed. Reflexive outputs are a further way in which relevance might be provided in the hierarchy.[61] Reflexive outputs are like normal outputs except that, *instead of affecting things outside the modeling system, they affect the modeling system itself*, controlling the actual hierarchy. Reflexive outputs can focus the actual hierarchy on particular features of the world, causing the actual hierarchy to represent or not represent a given part of the conceptual hierarchy. The idea is that the right combination of reflexive outputs could provide relevance in the hierarchy, by making it represent those parts of the conceptual hierarchy most important for making the predictions that are needed.

11.2 Why Reflexive Outputs Would Work

Reflexive outputs are special in terms of *what they do* – affecting the system itself – but they are not special in terms of *how they are made*. They are made in the same way as other outputs.

No special mechanism is required for the making of reflexive outputs. With the action selection process, as previously described, the system automatically learns to make the outputs that best improve its situation.[62] Reflexive outputs would just be a special case of this, and the system would just learn to make suitable reflexive outputs as part of learning to make all its other outputs. If the system learned that making reflexive outputs in a certain way improved its situation, then it would tend to continue doing this.

Reflexive outputs make sense if we realize that, at a basic level, any distinction between the system itself and the "outside world" is artificial. Parts of the model may make such a distinction, giving rise to the "self": the part of the model that relates the system's future outputs to previous inputs/outputs, and is therefore describable in terms of concepts such as "intentionality", but this is an emergent property of the hierarchy. The basic workings of the hierarchy – the way that pattern instances work and propagate information – and the workings of the action selection process make no such distinction.

---

[61] Almond, P., 2010. *An Attempt to Generalize AI - Part 13: Reflexive Outputs*. [Online] paul-almond.com. http://www.paul-almond.com/AI13.pdf or http://www.paul-almond.com/AI13.doc.

[62] Almond, P., 2010. *An Attempt to Generalize AI - Part 2: Planning and Actions*. [Online] paul-almond.com. http://www.paul-almond.com/AI02.pdf or http://www.paul-almond.com/AI02.doc.

At a basic level, there is no difference between the system and the outside world. The model just represents the world, which includes the system itself, and the system's outputs are selected to manipulate the world in such a way as to obtain the most favorable evaluation function scores, and whether such manipulation of the world happens to affect the system itself is, at a basic level, irrelevant.

If there are special outputs which can affect the hierarchy itself, then the hierarchy can learn to make suitable outputs of this kind which cause prediction of high EFS values, just as it would do with any other output. An output of this kind would affect a predicted EFS by reducing the uncertainty in the system's predictions of the future, allowing it to be better at planning its actions, so that it has more information to select navigate between them to achieve a better outcome.

11.3 Analogies for Reflexive Outputs

11.3.1 Analogy 1: Indirect Manipulation of the System

Suppose the AI system were installed in a robot which had dials on the exterior case. These dials can be used to change the actual hierarchy in the modeling system. A human engineer could come along and use those dials to adjust the modeling system, so as to adapt it to some situation. The dials might somehow alter the settings of the actual hierarchy inside the machine, either causing it to become denser or less dense in particular regions, or altering some more general settings in the ERP. A useful adjustment of the dials would be one which caused the uncertainty in the hierarchy's predictions of pattern instances corresponding to future EFS input – the pattern instances involved in the action selection process – to be reduced.

Now, suppose the robot has arms and hands and can manipulate its own dials by itself. It could do exactly the same as the human engineer, and it would not need any special learning process to do so: The robot is only using its outputs to manipulate reality, as usual, and the fact that the reality being manipulated now includes a dial on its case is irrelevant.

The idea of a robot using arms and hands to manipulate dials on its own exterior may seem slightly comical, and it is unwieldy and inefficient. If we wanted the robot to be able to do this, instead of doing this, we could link some of the robot's outputs directly to servomotors that move the dials. We could go still further, however. Ultimately, we could just link some of the robot's outputs directly to the systems they were supposed to be affecting in the hierarchy. These outputs would now be reflexive outputs.

11.3.2 Analogy 2: Obtaining a Better View

For another idea of what would be going on with reflexive outputs, we can imagine a system that manipulates its environment to get a better view. Suppose a robot needs to make some decision, and what is down a corridor in front of it is relevant to this

decision. However, the robot cannot see down the corridor because some crates are stacked up in the way. If we considered the robot intelligent, we would not be surprised if it moved the crates to obtain a better view down the corridor: to reduce the uncertainty in the information it has about what is down there, and ultimately, the future. We would regard an intelligent system manipulating the world to obtain a "better view" as something expected.

Now suppose that the part of the world being manipulated to obtain a better view is the AI system itself, that instead of moving crates to see what is down the corridor, the system can adjust parts of itself to see what is in the future. We should regard this as normal behavior, no different from any other kind, and requiring no special learning process or justification.

11.4 A Philosophical View of Reflexive Outputs

We will now look at what is happening when reflexive outputs are made. This is a bit more philosophical, because it relates to the more general issue of the "self", and things can be viewed in different ways.

11.4.1 View 1: Reflexive Outputs are the "Self" Controlling the Hierarchy

We normally think of the "self" as controlling the making of outputs – as initiating action in the world. For example, with the sentence "John kicked the ball" most people would have some idea that there is some person, some "self", called John, in a brain somewhere and this self decided to kick the ball. If we are taking this view, we could view reflexive outputs as the "self" manipulating the hierarchy that is running it. This seems to fit in well with our experience. Just as we can decide what to do in the "outside world" – kicking balls, picking things up, etc – we seem to be able to decide what to focus our minds on. With such a view, the experience of deciding to do something in the outside world and the experience of deciding to concentrate on something are very similar, differing only in terms of whether the outputs are directed inwards.

One issue with such a view might be that some actions seem to take place subconsciously, raising the whole issue of what the "self" means here, but that issue already exists when we apply this view to conventional actions anyway. We might say that some actions can be made subconsciously, or that there is some subconscious component to planning, but it seems intuitively meaningful to us to think of the "self" as *doing* things, even with this issue.

I have said that we might associate our experience of deciding to concentrate on something with reflexive outputs. This is not automatically the case. In the kind of system being discussed in this series of articles, there is no "self" separate to the modeling system: The "self" is merely part of the representation of the world in the

hierarchy, being the representation of that part of the world that very strongly relates to future behavior, which is similar to the kind of view previously proposed by Metzinger.[63] This means that the "self" is itself derived from what the rest of the hierarchy is doing. The "self" might be having an experience of deciding to concentrate on something, and this might be caused merely by the ERP arranging the hierarchy in a particular way, the experience then being derived from it. On the other hand, if reflexive outputs are in use it would make sense to say that at least some experience of deciding to concentrate on something, or some component of such experience, is explainable in terms of the "self" acting with the same degree of validity with which we would say that the "self" does anything. What I am really saying here is that we need to be careful not to run away and start making over-generalized statements relating experience to lower level processing.

11.4.2 View 2: Actions just seem to be initiated by the "self", but the "self" is just part of the model.

As previously discussed in Section 6: The "Self, on page 42, the view of the mind being taken in this cognitive model is one in which what we think of as the "self" is actually part of the model in the hierarchy. It could be argued that when we think that the "self" is initiating action, this is not really the case: that the "self" participates in this process only as an object in the model, and that the model itself initiates any action, the "self" being put together just to make sense of what is happening.

It should be noted, however, that even in this view, the "self" should not be viewed as completely passive. It would have to be having some effect on the outputs to justify its inclusion in the model at all. The issue here may be that the way in which it is having this effect is not necessarily what people think of when they talk about the "self" doing things. If the "self" is regarded as "doing things" when it influences a modeling system to produce a particular output, then the model of an umbrella in your brain is similarly "doing things" when it influences you to reach out and pick it up before going outside. The entire model could be said to be directing your actions and doing things. We might, however, regard part of the model, the "self" as different by a matter of degree, in its extreme effects on what you do.

If we take such a view, the idea of the "self" directing reflexive outputs fades away, and we no longer think in terms of the "self" deciding to manipulate the modeling system running it. This, however, is no different to the way we would be treating any other claim of the "self" acting.

---

[63] Metzinger, T., 2003. *Being No One: The Self-Model Theory of Subjectivity*. Cambridge (MA): MIT Press. Metzinger, T., 2009. *The EGO Tunnel: The Science of the Mind and the Myth of the Self*. New York: Basic Books.

Another issue here is that some people might take a particularly hard-line version of this view in which they say that the "self" does not exist.

11.4.3 Which view of reflexive outputs is correct?

How do we deal with the difference between these two views, and which one should we prefer? I suggest that neither view is the right one or the wrong one. Instead, each view is at a different level. The "self" in the first view may seem to fade away when examined closely, but practically anything seems to fade away when examined closely, so this should hardly concern us. Whatever view we take of reflexive outputs at any time should be consistent with the view that we take of outputs in general, and that view can be decided on according to what kind of discussion we are having. Sometimes we may want to discuss this at the level of the "self" doing things, and in those situations, it would be consistent to think of reflexive outputs as the "self" manipulating the system. At other times, we may be looking more deeply at the system, and just viewing things in terms of the hierarchy of pattern instances, and in those situations the "self" may not be relevant to an explanation of outputs.

Some people may say that the second view is right, and that the first view is wrong. Even those people, I think, when they are not doing cognitive science or philosophy, would use the same language as everyone else in discussing their actions.

11.5 Complications with Implementation of Reflexive Outputs

I have given a description of the general idea of reflexive outputs, here, but I have not discussed in any detail how they would work. I think it is going to be difficult, at this stage, to get very far with that. The obvious difficulty with getting reflexive outputs to work is how you map specific reflexive outputs onto the hierarchy of pattern instances. It would be absurdly impractical to have each reflexive output to control addition or removal of a single pattern instance.

One approach may be to associate reflexive outputs with specific, high-level patterns, so that they can make these patterns more or less influential on what is happening with the hierarchy. The idea here is that control of a small number of high-level patterns could have a significant effect on the hierarchy. The exact way in which this works would depend on the way the hierarchy is implemented and the relationship between patterns and pattern instances.

Another approach might be for reflexive outputs to control general settings in the ERP. For example, they might globally control the rate at which pattern instances are pruned or added.

12 Pattern Instance Construction Alternatives

12.1 The General Idea

Pattern instances belong to *patterns*, and this is important for doing statistics with the hierarchy: The members of a pattern about which a lot is known can be used to provide information about members about which less is known. The way that pattern instances are created and become members of patterns needs considering. A particular approach has been assumed in Section 2: The Basic Structure of the Hierarchy, on page 10; [64] however other approaches might also be considered.[65]

12.2 The Previously Described Approach to Pattern Instance Construction

A specific way of creating pattern instances which are members of patterns was proposed in Section 2.2.3: The Construction Specification of a Pattern, on page 16. This is as follows.

The approach involves patterns constructing pattern instances. That is to say, each pattern acts as a machine, examining different parts of the hierarchy and connecting new pattern instances of that pattern into it.

A pattern consists of a *pattern specification* and a set of pattern instances. The pattern specification consists of a *construction specification* and a *logic specification*.

The logic specification describes how each pattern instance generates a pattern output from its labeled pattern inputs. The logic specification is the same for every pattern instance in the pattern.

The construction specification controls the connection of new pattern instances for this pattern into the hierarchy. It is able to examine the hierarchy and follow connections between pattern instances. For example, it could examine a pattern instance and determine the pattern to which it belonged, the pattern instances it was using as pattern inputs and so on. I have not specified exactly how the construction specification will work, but a placeholder idea, for now, is that it is a computer program capable of

---

[64] Almond, P., 2010. *An Attempt to Generalize AI - Part 5: A Completely Probabilistic Hierarchy*. [Online] paul-almond.com. http://www.paul-almond.com/AI05.pdf or http://www.paul-almond.com/AI05.doc. pp.11.-20.

[65] Almond, P., 2010. *An Attempt to Generalize AI - Part 10: Alternatives for Pattern Instance Construction*. [Online] paul-almond.com. http://www.paul-almond.com/AI10.pdf or http://www.paul-almond.com/AI10.doc.

"reading" the hierarchy's structure and directing the "wiring" of new pattern instances into it.

12.3 The Possibility of Different Construction Methods

Inclusion of pattern instances in a pattern is determined by the pattern's construction specification. Previously, in Section 2.2.3: The Construction Specification of a Pattern, on page 16, I described the construction specification as actually making new pattern instances that are going to belong to the pattern, by examining the hierarchy and defining new pattern instances to be connected into it.[66]

This is only one way of constructing patterns, however. The most important features of a pattern are that the pattern instances belonging to it are determined by the construction specification and that the construction specification can examine the "wiring" in the hierarchy. In a previous article, I suggested that the construction specification might work differently.[67] Patterns could be wired into the hierarchy randomly, and the construction specifications of patterns, instead of *making* the pattern instances, could *select* existing pattern instances for inclusion in the pattern, but still by the same general method of examining the hierarchy. With such a process, a pattern instance might belong to multiple patterns, and membership of patterns might also be made "fuzzy", so that pattern instances could have varying degrees of membership of patterns.

Different approaches for "construction" of pattern instances will now be discussed.

12.4 Different Ways in Which the Construction Specification Could Work

12.4.1 Pattern instances are *made* by patterns.

This is the approach previously described in 2.2.3: The Construction Specification of a Pattern, on page 16.[68] A pattern has a *construction specification*. The pattern adds new pattern instances to the hierarchy, the construction specification controlling how their labeled pattern inputs connect to it, and they belong to the pattern that added them.

---

[66] Almond, P., 2010. *An Attempt to Generalize AI - Part 5: A Completely Probabilistic Hierarchy*. [Online] paul-almond.com. http://www.paul-almond.com/AI05.pdf or http://www.paul-almond.com/AI05.doc. pp.11.-20.

[67] Almond, P., 2010. *An Attempt to Generalize AI - Part 3: Forgetting*. [Online] paul-almond.com. http://www.paul-almond.com/AI03.pdf or http://www.paul-almond.com/AI03.doc. p.10. Note: The actual approach to forgetting described in that article is now obsolete.

[68] Almond, P., 2010. *An Attempt to Generalize AI - Part 5: A Completely Probabilistic Hierarchy*. [Online] paul-almond.com. http://www.paul-almond.com/AI05.pdf or http://www.paul-almond.com/AI05.doc. pp.11.-20.

When the construction specification is adding pattern instances, it is able to examine the structure of the hierarchy; for example, it can read the type of pattern to which a pattern instance belongs, or examine its pattern inputs.

12.4.2 Pattern instances are made *randomly* and *selected* by patterns.

When pattern instances are added to the hierarchy, they are connected randomly to it, with the possible limitation of some limit on the logic distance between pattern inputs of the same pattern instance. The pattern's specification examines the pattern instance's internal logic – how it relates its labeled pattern inputs to its pattern output – and its "local situation" – how it is connected to the hierarchy and the structure of the hierarchy around it, giving it some degree of membership of the pattern. The greater a pattern instance's degree of membership of some pattern, the greater the effect it would have on the pattern's statistics in statistics generation and the greater the effect those statistics would have on the probabilistic information for the pattern instance in statistics application. A pattern instance could belong to multiple patterns: In fact, it could have some degree of membership of *all* patterns.[69]

12.4.3 Pattern instances are made by some *exploratory process* and *selected* by patterns.

This is similar to the approach just described, in 12.4.2, except that a pattern instance is not made *completely* randomly. Instead it changes, and as it changes its degree of membership of different patterns changes. The pattern instance would change to seek relevance. This would mean altering the basic, exploratory relevance process (BERP), or any other exploratory relevance process (ERP) derived from it, a bit: The removal of pattern instances would now become pattern instances being required to change. The general idea of the BERP would remain however: Changing a pattern instance into another one is nothing more than a *gradual* removal of it, and the gradual replacement of it with another pattern instance.

12.4.4 The connection of pattern instances is *influenced* by patterns.

This has some similarity with the approach of allowing a pattern to construct pattern instances, in 12.4.1, except that a pattern instance's behavior and connection to the hierarchy is not controlled by a single pattern. Instead a pattern instance is influenced by different patterns over a period of time, each having some ability to change its behavior and the way it is connected to the hierarchy.

Other approaches might combine features of these.

---

[69] This approach was previously described in Almond, P., 2010. *An Attempt to Generalize AI - Part 3: Forgetting*. [Online] paul-almond.com. http://www.paul-almond.com/AI03.pdf or http://www.paul-almond.com/AI03.doc. p.10.

12.5 The general idea is the same for the different pattern instance construction methods.

A number of ways of relating patterns to pattern instances have just been described. The exact approach used does not affect the basic principles on which the system works. If we select a particular approach, it would have some affect on the *details* of how the various processes in the system work, but not on the general principles. For example, if a pattern instance could have varying degrees of membership of multiple patterns, then with statistics generation/application this would just mean that account would need taking of the degree of membership of some pattern, and therefore the extent to which a pattern instance affects the statistics or the statistics affect that pattern, or others, when generating or applying statistics. With some approaches to connecting pattern instances to the hierarchy, what is currently described as the "removal" of pattern instances from the hierarchy in the BERP may instead involve *changing* pattern instances.

For now, I will not make a final decision on this issue. The "constructive" approach described in Section 2.2.3: The Construction Specification of a Pattern, on page 16,[70] and the description of the ERP for such a system,[71] are adequate for now.

[70] Almond, P., 2010. *An Attempt to Generalize AI - Part 5: A Completely Probabilistic Hierarchy*. [Online] paul-almond.com. http://www.paul-almond.com/AI05.pdf or http://www.paul-almond.com/AI05.doc. pp.11.-20.

[71] Almond, P., 2010. *An Attempt to Generalize AI - Part 6: Measuring Relevance*. [Online] paul-almond.com. http://www.paul-almond.com/AI06.pdf or http://www.paul-almond.com/AI06.doc.
Almond, P., 2010. *An Attempt to Generalize AI - Part 7: A Basic, Exploratory Relevance Process.* [Online] paul-almond.com. http://www.paul-almond.com/AI07.pdf or http://www.paul-almond.com/AI07.doc.
Almond, P., 2010. *An Attempt to Generalize AI - Part 8: Forgetting as Part of the Exploratory Relevance Process*. [Online] paul-almond.com. http://www.paul-almond.com/AI08.pdf or http://www.paul-almond.com/AI08.doc.
Almond, P., 2010. *An Attempt to Generalize AI - Part 9: Improving the Exploratory Relevance Process*. [Online] paul-almond.com. http://www.paul-almond.com/AI09.pdf or http://www.paul-almond.com/AI09.doc.

13 Conclusion

The previous articles in this series have developed a model of cognition and approach to artificial intelligence (AI), and discussed some general philosophical issues that relate to this. This article has provided a complete description of the cognitive model developed so far, so that it can be understood by reading a single article. The writing of this article does not mean that the approach is fully developed and needs no more work: It merely describes what has been developed at the time of writing, and there may be one or more other complete descriptions, later, to encompass later developments.

The cognitive model is one of a hierarchy based on *patterns*, which are sets of *pattern instances*.[72] A pattern has a specification of the criteria that a pattern instance must meet to be a member of it. Pattern instances on the bottom level of the hierarchy correspond to external inputs/outputs occurring at specific times. Pattern instances at higher levels of the hierarchy have pattern outputs that are abstracted from these by the internal logic of the pattern instances. Only pattern instances corresponding to inputs/outputs that have already occurred, or pattern instances dependent only on them, can be known about with certainty. Other pattern instances – the ones dependent on future inputs/outputs – can only be known about probabilistically.

Probabilistic information is propagated through the hierarchy by logic application, statistics generation and statistics application.[73] Logic application involves using information about the pattern inputs of a pattern instance, together with its internal logic, to obtain probabilistic knowledge about its pattern output. This can be used to obtain probabilistic information about pattern instances that are strongly dependent on inputs/outputs that have already occurred. Logic application can also occur downwards, with information about a pattern instance providing information about its pattern inputs. Statistics generation involves using the probabilistic information that has been obtained in logic application for the pattern instances of a pattern to determine the probabilities of different combinations of pattern inputs occurring. Statistics application involves using the information about patterns obtained in statistics generation to obtain information about individual pattern instances. Statistics application can occur upwards or downwards.

Planning of actions is provided by the *action selection process*, which involves using the modeling system to model the system itself to plan its behavior.[74] The action selection process uses a prediction of a continually computed and input evaluation function score (EFS).

---

[72] Section 2.1: Patterns and Pattern Instances, on page 10.
[73] Section 4: Probabilistic Propagation in the Actual Hierarchy, on page 22.
[74] Section 5: Planning of Actions, on page 32.

This view of the planning of actions has implications for our view of the "self".[75] In this view, there is no separate system, outside the hierarchical model, that is planning all the actions. Planning is something that is done mainly within the model. The system has no special "self-modeling" process. Instead, what we think of as the "self" is part of the model within the hierarchy, generated in much the same way as the parts of the model that deal with any other aspects of reality. A view of cognition in which the "self" is part of the model has been described by Metzinger.[76]

The hierarchy needs to be *relevant*, including the pattern instances that represent those features of the world which are most useful in planning actions and excluding others. This requires a way of measuring the relevance of pattern instances, and this is provided by the relevance measurement process (RMP).[77] The RMP starts with particular pattern instances, corresponding to future input of the EFS and used in the action selection process, being assigned relevance, and this relevance is then back-propagated so that other pattern instances are assigned relevance. The relevance values are used in a *basic, exploratory relevance process* (BERP), which involves continual removal of low-relevance pattern instances, while the hierarchy is extended from the pattern instances that remain.[78] "Forgetting" – the removal of obsolete pattern instances – is also provided by the BERP.[79]

Various ways in which the sophistication of the BERP might be increased, giving an improved exploratory relevance process (ERP), have been discussed.[80]

Relevance can also be provided by *reflexive outputs*: special outputs which are directed inwards, acting on the hierarchy itself. [81]

The relationship between patterns and pattern instances is an issue that still needs to be resolved. Most of this article has assumed that patterns are "constructive" and control the generation of their pattern instances, but other possibilities have been discussed.[82] The issue of how to represent and generate pattern specifications needs some consideration. A trial and error approach is likely, and it may be a Darwinian one. The challenge here will be achieving sufficient ontological *generality* with something that responds well to a trial and error, and possibly Darwinian, approach.

---

[75] Section 5: Planning of Actions, on page 32.
[76] Metzinger, T. (2003). *Being No One: The Self-Model Theory of Subjectivity*. Cambridge, MA: MIT Press. Metzinger, T. (2009). *The EGO Tunnel: The Science of the Mind and the Myth of the Self*. New York: Basic Books.
[77] Section 8.3: The Relevance Measurement Process, on page 53.
[78] Section 8.4: The Basic, Exploratory Relevance Process, on page 57.
[79] Section 10: Forgetting, on page 77.
[80] Section 9: Possible Improvements to the Basic, Exploratory Relevance Process, on page 70.
[81] Section 11: Reflexive Outputs as a Way of Providing Relevance, on page 86.
[82] Section 12: Pattern Instance Construction Alternatives, on page 91.

In a previous article, I have suggested that dreaming might be explained in terms of a partially functioning hierarchy, the idea being that waking experience is a special case of a dream which happens to be caused by a particularly high-functioning hierarchy which is particularly strongly influenced by the external world.[83] I have also suggested that a the cognitive model suggests a mechanism for mind control, in which the brain is "lied to" about outputs that it has made, so that the "self" generated in the model is based on misleading records of previous behavior.[84]

[83] Almond, P., 2010. *An Attempt to Generalize AI - Part 11: Explaining Dreaming*. [Online] paul-almond.com. http://www.paul-almond.com/AI11.pdf or http://www.paul-almond.com/AI11.doc.
[84] Almond, P., 2010. *An Attempt to Generalize AI - Part 14: Mind Control Speculation*. [Online] paul-almond.com. http://www.paul-almond.com/AI14.pdf or http://www.paul-almond.com/AI14.doc.

14 Bibliography

Almond, P., 2006. *A Proposal for General AI Modeling*. [Online] paul-almond.com. Available at: http://www.paul-almond.com/Modeling.pdf or http://www.paul-almond.com/Modeling.doc. [Accessed 10 July 2010].

Almond, P., 2007. *Planning As Modelling in AI: A New Version*. [Online] paul-almond.com. Available at: http://www.paul-almond.com/PlanningAsModellingNew.pdf or http://www.paul-almond.com/PlanningAsModellingNew.doc. [Accessed 10 July 2010].

Almond, P., 2010. *An Attempt to Generalize AI - Part 1: The Modeling System*. [Online] paul-almond.com. Available at: http://www.paul-almond.com/AI01.pdf or http://www.paul-almond.com/AI01.doc [Accessed 10 July 2010].

Ibid. p.9, pp.27-28.

Almond, P., 2010. *An Attempt to Generalize AI - Part 2: Planning and Actions*. [Online] paul-almond.com. Available at: http://www.paul-almond.com/AI02.pdf or http://www.paul-almond.com/AI02.doc [Accessed 10 July 2010].

Ibid. pp.23-24.

Almond, P., 2010. *An Attempt to Generalize AI - Part 3: Forgetting*. [Online] paul-almond.com. Available at: http://www.paul-almond.com/AI03.pdf or http://www.paul-almond.com/AI03.doc [Accessed 10 July 2010].

Ibid. p.10.

Almond, P., 2010. *An Attempt to Generalize AI - Part 4: Modeling Efficiency*. [Online] paul-almond.com. Available at: http://www.paul-almond.com/AI04.pdf or http://www.paul-almond.com/AI04.doc [Accessed 10 July 2010].

Ibid. p.11, p.13.

Almond, P., 2010. *An Attempt to Generalize AI - Part 5: A Completely Probabilistic Hierarchy*. [Online] paul-almond.com. Available at: http://www.paul-almond.com/AI05.pdf or http://www.paul-almond.com/AI05.doc [Accessed 10 July 2010].

Ibid. pp.8-9, pp.11-20, pp.21-29, pp.31-32.

Almond, P., 2010. *An Attempt to Generalize AI - Part 6: Measuring Relevance*. [Online] paul-almond.com. Available at: http://www.paul-almond.com/AI06.pdf or http://www.paul-almond.com/AI06.doc [Accessed 10 July 2010].

Ibid. 16-19.

Almond, P., 2010. *An Attempt to Generalize AI - Part 7: A Basic, Exploratory Relevance Process*. [Online] paul-almond.com. Available at: http://www.paul-almond.com/AI07.pdf or http://www.paul-almond.com/AI07.doc [Accessed 10 July 2010].

Ibid. pp.6-7, p.10, pp.13-14, p.17.

Almond, P., 2010. *An Attempt to Generalize AI - Part 8: Forgetting as Part of the Exploratory Relevance Process*. [Online] paul-almond.com. Available at: http://www.paul-almond.com/AI08.pdf or http://www.paul-almond.com/AI08.doc [Accessed 10 July 2010].

Almond, P., 2010. *An Attempt to Generalize AI - Part 9: Improving the Exploratory Relevance Process*. [Online] paul-almond.com. Available at: http://www.paul-almond.com/AI09.pdf or http://www.paul-almond.com/AI09.doc [Accessed 10 July 2010].

Almond, P., 2010. *An Attempt to Generalize AI - Part 10: Alternatives for Pattern Instance Construction*. [Online] paul-almond.com. Available at: http://www.paul-almond.com/AI10.pdf or http://www.paul-almond.com/AI10.doc [Accessed 10 July 2010].

Almond, P., 2010. *An Attempt to Generalize AI - Part 11: Explaining Dreaming*. [Online] paul-almond.com. Available at: http://www.paul-almond.com/AI11.pdf or http://www.paul-almond.com/AI11.doc [Accessed 10 July 2010].

Almond, P., 2010. *An Attempt to Generalize AI - Part 12: Pattern Relevance*. [Online] paul-almond.com. Available at: http://www.paul-almond.com/AI12.pdf or http://www.paul-almond.com/AI12.doc [Accessed 10 July 2010].

Ibid. pp.9-11.

Almond, P., 2010. *An Attempt to Generalize AI - Part 13: Reflexive Outputs*. [Online] paul-almond.com. Available at: http://www.paul-almond.com/AI13.pdf or http://www.paul-almond.com/AI13.doc [Accessed 10 July 2010].

Almond, P., 2010. *An Attempt to Generalize AI - Part 14: Mind Control Speculation*. [Online] paul-almond.com. Available at: http://www.paul-almond.com/AI14.pdf or http://www.paul-almond.com/AI14.doc [Accessed 10 July 2010].

Grand, S., 2010. *Brainstorm #3 – cheating doesn't pay (sometimes)*. [Online] Steve Grand's Blog: Artificial Life in Real Life. Available at: http://stevegrand.wordpress.com/2010/04/22/brainstorm-3-cheating-doesnt-pay-sometimes/ [Accessed 10 July 2010].

Hawkins, J. & Blakeslee, S., 2004. *On Intelligence*. New York: Henry Holt.

Metzinger, T., 2003. *Being No One: The Self-Model Theory of Subjectivity*. Cambridge (MA): MIT Press.

Metzinger, T., 2009. *The EGO Tunnel: The Science of the Mind and the Myth of the Self*. New York: Basic Books.