

# How AI Would Work

By Paul Almond, 4 September 2006

**Website:** [www.paul-almond.com](http://www.paul-almond.com)  
**Email:** [info@paul-almond.com](mailto:info@paul-almond.com)

© Copyright Paul Almond, 2006. All Rights Reserved.

# How AI Would Work

By Paul Almond, 4 September 2006

## Introduction

This article will outline how I think artificial intelligence (AI) should work. In previous articles I have proposed an approach for making AI, but I have not yet outlined it in a single article. This article will give an overview of the proposed AI system. This could be met with the challenge that it is easy to specify things in general terms like this, but there are the previous, more detailed articles. This article is not a substitute for these, but it should make my approach more accessible.

This article will also provide a discussion of how the *carpet texture problem* can be resolved in this system. I have not previously discussed this.

Readers may wish to compare the concepts in this article, and preceding articles, with the hierarchical system proposed by Jeff Hawkins [1,2].

## The Basic Idea

The approach involves a *probabilistic hierarchy of meaning extraction* by partial model algorithms from inputs and outputs in which low level probability values imply higher level probability values. Meaning extraction is abstraction and is the finding of patterns. *Downward transfer of probabilities* does the reverse: it transmits information down the hierarchy, causing high level probability values to influence low level probability values and fill in details.

Planning uses the same probabilistic hierarchy of meaning extraction. There is no separate planning system, as in the Hawkins system, in which there is a separate action planning hierarchy “closely-coupled” with the meaning extraction hierarchy, in which actions are planned in an abstract way and then become more detailed during transmission down the hierarchy: in this system there is only one hierarchy that does both modelling and planning. Planning is not a distinct process from modelling. This approach is based on the idea that *planning of behaviour is prediction of behaviour*: a system that makes a predictive model of the world, when allowed to observe its own behaviour as just another part of reality, will automatically make models that incorporate planning of its future actions. The system’s future behaviour is based on predictive modelling from its past behaviour, with the addition of a separate system, the *output vectoring system*, constrained by its predictions of its own future behaviour, giving it a “direction”(but not doing the *real* planning, which is done in the probabilistic modelling hierarchy).

The use of computing resources in the modelling hierarchy needs to be prioritized; otherwise the system would expend resources on extracting irrelevant patterns from reality. This is dealt with by the designation of some of the system’s own outputs as special control outputs which, instead of being directed at the outside world, are turned

inwards and used to control the prioritization of computing resources in the probabilistic hierarchy.

Now that I have summarized the approach I will provide a short discussion of each of the concepts on which the AI system is based. I will then expand on this in the rest of the article.

## **A Short Overview of the AI System**

### **Conceptual Hierarchy**

The conceptual hierarchy is an idealized structure used to represent reality with hierarchical extraction of meaning by partial model algorithms. Each partial model algorithm can be applied for different indices. Each meaning is a binary digit (bit). The conceptual hierarchy is based on input and output events that are fed into the bottom level an AI system and involves hierarchical extraction of meaning. It should be noted that outputs are fed into the bottom level of the system, together with inputs, so that the system's own actions are part of what it observes to make its model of reality: this is important with regard to how the system plans. A single input or output in the hierarchy is actually an input or output *event* - an instance of a conventional input or output occurring at an instant in time. That is to say, a conventional input changing over time is represented in the hierarchy by a number of inputs – each one corresponding to its state at a single instant. Because the conceptual hierarchy is based on inputs and outputs at different instants of time it contains temporal patterns. The conceptual hierarchy cannot actually be used in an AI system.

### **Functional Hierarchy**

The functional hierarchy is a probabilistic version of the conceptual hierarchy that is actually used in an AI system. Each meaning bit in the conceptual hierarchy has a probability in the functional hierarchy. The functional hierarchy uses a probabilistic version of the meaning extraction implied by partial models in the conceptual hierarchy to transmit information up the hierarchy, producing higher levels of abstraction. Information is transmitted down the hierarchy by downward transfer of probabilities, a process by which probability values at high levels of the hierarchy affect those at low levels, equivalent to “filling in the details”. Some of the input and output values in the bottom level of the functional hierarchy will relate to input and output events that have not occurred yet, but downward transfer of probabilities will influence their probability values. This is equivalent to the hierarchy making predictions of future inputs and outputs.

### **Planning of Actions**

There is no division between planning and modelling. *Planning of behaviour is prediction of behaviour*. Planning of actions occurs in the functional hierarchy itself: there is no separate planning system. As downward transfer of probabilities will affect the

probabilities for future input and output events in the bottom level of the hierarchy, the natural behaviour of the system involves predicting its own future outputs along with events in reality. The system's previous behaviour establishes patterns that direct its future behaviour. For this to work the system needs to be set on the path of desirable behaviour: it needs a direction. This is provided by the *output vectoring system*, which performs a tree search of possible future outputs to determine the outputs to make now which put the system in the best situation for later obtaining a high score from a *situational evaluation function*. The search by the output vectoring system for optimum, immediate behaviour is constrained by the probability values for the output events in the bottom level of the functional hierarchy. This is how the functional hierarchy's probabilistic *predictions* of the system's future behaviour are used to direct the *planning* of future behaviour.

This approach to planning can be shown by a simple example.

Suppose we have a conventional AI system, controlling some sort of robot, and its modelling system determines, based on modelling from previous data, that there is a 60% chance of rain tomorrow. In a conventional system, this information would be passed to a planning system, which may then decide to take an umbrella out. In this sort of system, however, things are different. As well as determining that there is a 60% chance of rain, the modelling system may also determine, based on the same kind of modelling of previous observations that gave rise to the prediction of rain, that there is an 80% chance that the system will take an umbrella out tomorrow. This prediction of the taking out of an umbrella is equivalent to the decision to do it! This prediction of future action is part of the information sent to the output vectoring system to constrain it. The output vectoring system has most of the work done for it when given predictions like this. This example is simplistic and, in reality, the predictions would not refer specifically to future actions like taking umbrellas: they would refer to future output bit values.

### **Use of Outputs to Control Internal Prioritization**

One issue needing addressing is the carpet texture problem – the problem of how to prioritize processing resources in the functional hierarchy. The approach taken is to designate some of the system's outputs as special *prioritization control outputs* which, instead of being used for motor outputs in the outside world, are used to control prioritization of computing resources in the functional hierarchy, determining which partial model algorithms are in use for particular indices. This approach deals with the carpet texture problem by using same planning capabilities that are used to manipulate the system's situation in the outside world.

See figure 1 for an overview of the AI system.

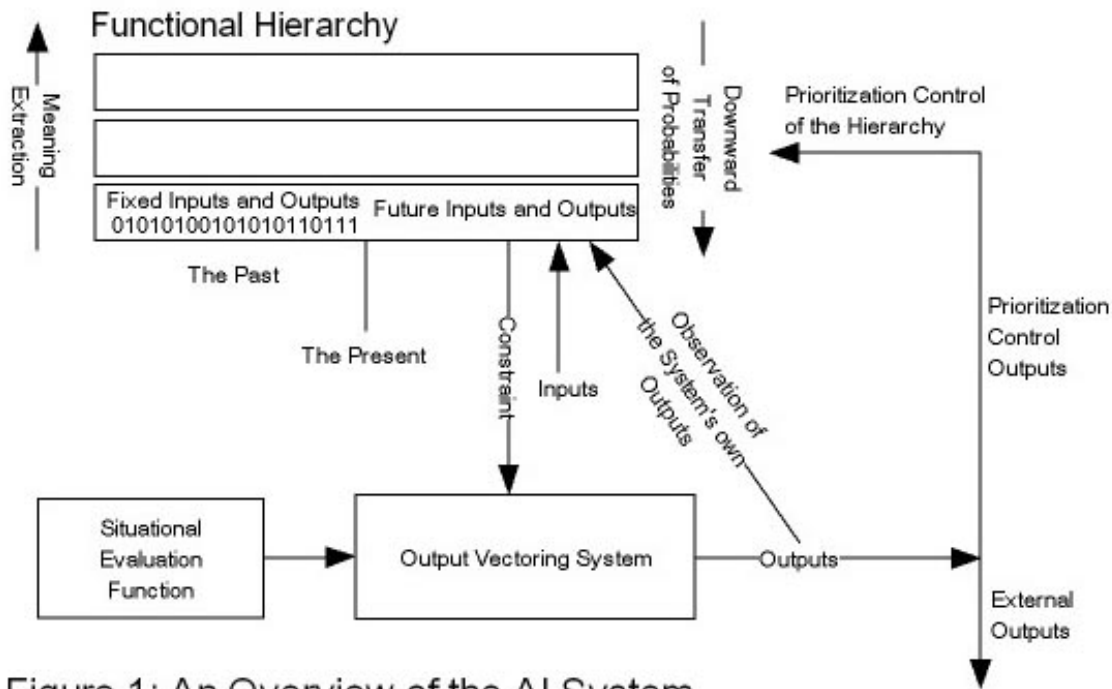


Figure 1: An Overview of the AI System

Now that I have provided a short discussion of each of the concepts in the AI system, I will provide a more detailed description – although it will still be an overview.

## The Conceptual Hierarchy

### What the Conceptual Hierarchy Is

The conceptual hierarchy is an idealized, layered structure used to describe reality. Each layer consists of meanings extracted from the layer below. Extraction of meaning is abstraction and pattern detection. Each extracted meaning is a single bit.

### Meaning Extraction Algorithms

The way in which meanings are extracted is specified by partial model algorithms (meaning extraction algorithms). A partial model algorithm processes bits in a layer and generates a value of “0” or “1” for a bit in the layer above. A single partial model algorithm may be used for a number of bits with different indices when the meaning for those bits is extracted in the same general way. Partial models algorithms are described in a previous article *Occam’s Razor Part 6: Partial Models as “Envelopes”* [3].

### Inputs and Outputs in the Conceptual Hierarchy

The bottom level of the conceptual hierarchy is the interface between the hierarchy and reality. It consists of input and output bits. Each input or output bit corresponds to an input or output *event*. An input or output event is not the same as a conventional input or

output in a computer: it is the state of one of the system's inputs or outputs at a single instant in time.

This means that any conventional input or output which changes over time – such as a light sensor which gives an input of “1” when light strikes it and “0” otherwise – is represented over a period of time in the hierarchy by a number of input or output bits, each of which corresponds to the state of the input or output at a single instant and has a value of “0” or “1”.

All the inputs that the system can receive and the outputs that it can make are represented in the system by input bits and output bits.

### **No Uncertainty in the Conceptual Hierarchy**

Because all of the input and output events that occur, at any time, are in the bottom level of the hierarchy it is incoherent to consider anything happening to change the information in the hierarchy later.

### **Temporal Nature of Meanings in the Hierarchy**

All bits at higher levels in the hierarchy are derived by partial model algorithms from the input and output bits at the bottom level and these describe a temporal situation – the change in the inputs and outputs over time – so the meanings corresponding to bits in the higher levels in the hierarchy are also temporal.

### **Inclusion of Outputs into the Hierarchy**

The bottom level of the hierarchy contains output bits as well as input bits. The hierarchy is therefore extracting temporal meanings from input events and output events. As far as the hierarchy is concerned, output events are treated in the same way as input events.

The bottom level of the hierarchy contains all the observations of reality that the system makes. This includes any inputs, but the system also observes its own outputs and these observations are just another form of input into the bottom level.

### **No Distinction Between Inputs and Outputs At Higher Levels**

Because the bottom level of the hierarchy contains bits corresponding to input events and output events, all meaning at higher levels is derived from input events and output events. Partial model algorithms do not distinguish between inputs and outputs.

A single extracted meaning bit at a high level could be derived from a number of input bits and a number of output bits and another high level extracted meaning bit could be extracted from a number of lower level extracted meanings, each being derived from a number of input bits and output bits. It is incoherent to talk about whether a particular high level meaning is extracted from inputs or outputs.

Above the bottom level of the hierarchy the idea of differentiation between inputs and outputs is incoherent: there are just extracted meanings.

### **Static Nature of the Conceptual Hierarchy**

The inputs and outputs in the bottom level of the hierarchy do not change as the inputs and outputs of a real AI system would change. This is because each input or output bit in the hierarchy corresponds to an input or output event and the bits corresponding to any future input or output events are already in the hierarchy. The bottom level of the hierarchy does not describe the states of the system's inputs or outputs at some instant of time, but at all instants of time.

The bottom level of the hierarchy is atemporal. Higher level meanings are just extracted from this so the entire conceptual hierarchy is hierarchical and does not change over time: the hierarchy is static.

Although the hierarchy as a whole is atemporal it does contain temporal information.

### **Idealism of the Conceptual Hierarchy**

The conceptual hierarchy does not exist in a computer: it is purely a conceptual structure. It assumes omniscience of the computer containing it and leaves no room for uncertainty and its static nature leaves no scope for a system containing it even to act.

Ideas like this are discussed in more detail in *Occam's Razor Part 7: Hierarchy and Ontology* [4].

## **The Functional Hierarchy**

### **What the Functional Hierarchy Is**

The functional hierarchy is a representation of the conceptual hierarchy which can be implemented in a computer. It forms the basis of AI systems. The purpose of the functional hierarchy is to provide a model of reality. As shall be discussed later, modelling encompasses planning so the functional hierarchy is the basis of everything in an AI system.

### **Probabilistic Nature of the Functional Hierarchy**

Whereas the conceptual hierarchy has no uncertainty, the functional hierarchy contains uncertainty. Where the conceptual hierarchy contains bits, each with a value of "0" or "1" to describe input or output events at the bottom level or extracted meanings at higher levels, the functional hierarchy contains a probability for each of these bits. Each probability value in the functional hierarchy is the probability that the corresponding bit in the conceptual hierarchy is "1".

## Where the Uncertainty Comes From

In the conceptual hierarchy, the values of all input and output bits are known, but this is impossible for the functional hierarchy: some of the input and output bits correspond to future input and output events and their values cannot be known yet.

Input and output bits for future input or output events can only be described by probabilities. Any meanings extracted from such bits by partial models must also be probabilistic. Because of this injection of uncertainty, all input, output and extracted meaning bits in the functional hierarchy are described by probability values rather than values of “0” and “1”.

If an input or output bit corresponds to a future input or output event then its default probability is 0.5. This can be changed, however, by *downward transfer of probabilities* – a process that will be discussed later.

## Probabilistic Interpretation of Partial Model Algorithms

Partial model algorithms do not have any probabilistic nature: a partial model algorithm simply processes the bits on one level and generates an extracted meaning bit at a higher level. With the functional hierarchy, however, we no longer have actual bit values, but just probabilities. This means that partial model algorithms cannot be executed: we do not know the bit values that they take as data.

Instead of executing partial model algorithms, we make a probabilistic interpretation of them. This means examining the probability values for the bits from which the partial model algorithm will extract its meaning and the code in the partial model algorithm itself and determining the probability that the partial model algorithm would return a value of “1” for the higher level extracted meaning bit. This probability is then placed in the functional hierarchy where the corresponding higher level, extracted meaning bit would be in the conceptual hierarchy.

## Fixing of Input and Output Bits

Input and output bits corresponding to future input or output events have uncertainty, but there is no uncertainty about input or output bits corresponding to past input or output events: the value that the bit had in the event is obviously known as the event has happened. In the functional hierarchy, input or output bits corresponding to input or output events in the past are *fixed*, meaning that they can only have one of two probability values: 0 if the bit is known to have a value of “0” or 1 if the bit is known to have a value of “1”.

When a bit has been fixed its probability value cannot change again: its actual value is now known. A probability value of 0 or 1 leaves no scope for further change, as it implies that there is no certainty about the bit’s value.

## **Downward Transfer of Probabilities**

Low level probabilities imply high level probabilities and probabilistic interpretation of partial model algorithms deals with this, but high level probabilities also imply low level probabilities: information about probability values needs to be transmitted down the hierarchy and this is downward transfer of probabilities.

While transfer of probability information up the hierarchy generates abstract, high level meanings, these high level meanings imply the filling in of blanks at a low level.

As an example, if we have the sequence 0,1,0,1,0,? Then a high level, abstract meaning would be that this is a sequence of bits that alternates: 01010101010101010, etc. Such a high level meaning simply describes the entire sequence as an object and makes no reference to individual numbers in the sequence. This higher level meaning implies that the next number in the sequence is “1”: the high level meaning influences what we know about things at a lower level.

If we did not know for certain that this was the sequence, but had only a probability of this, then our assertion that the next digit is “1” would be probabilistic, and if we already had a probability for the next digit being “1” then the probability that this is a sequence of alternating bit values would influence this probability. This is what is going on with downward transfer of probabilities.

## **Downward Transfer of Probabilities as Prediction**

Because the hierarchy contains temporal meanings then downward transfer of probabilities also has a temporal aspect: some of the probability values affected by downward transfer of probabilities will be input and output bits corresponding to future input and output events. This is equivalent to making predictions about the future inputs that will be received and the outputs that will be made.

## **The system is trying to predict its own outputs**

Downward transfer of probabilities, when it affects the bottom level of the hierarchy, is equivalent to the prediction of future inputs and outputs. It may seem normal for the system to try to predict its future inputs as this just means predicting what is going to happen in the world, but what about the system trying to predict its own outputs? This may seem strange, because the system is in control of its future outputs: it is trying to predict its own behaviour.

It is natural for the system to try to predict its own behaviour. This is an inevitable result of feeding the system’s outputs into the bottom level of the hierarchy so that it can observe them. The system needs to predict what will happen in the world, and its own behaviour is part of the world. If we declared it invalid for the system to do this then this would be a contrivance. A lot of the future inputs that the system will receive will be

contingent on its future behaviour anyway, so trying to predict these would really involve indirect prediction of the system's own behaviour.

Rather than try to dismiss the issue of the system predicting its own behaviour in some contrived way, the explicit inclusion of the system's own behaviour into its modelling of the future, by inclusion of output events in the bottom level of the hierarchy, is necessary to allow planning of behaviour to work: there is a close link between *prediction* of behaviour and *planning* of behaviour and this will be discussed later in this article.

### **Relationship Between the Conceptual and Functional Hierarchies**

The conceptual and functional hierarchies are basically the same hierarchy. The functional hierarchy is the conceptual hierarchy expressed probabilistically because of limited knowledge about it. This limitation in knowledge comes from some of the bottom-level input and output bits relating to input and output events that will occur in the future.

When input and output bits in the functional hierarchy become fixed they always agree with the conceptual hierarchy. This is the closest we can get to actually seeing the conceptual hierarchy.

An aim of probabilistic interpretation of partial model algorithms and downward transfer of probabilities is to reduce the uncertainty in the functional hierarchy as much as possible to make it as similar as possible to the conceptual hierarchy.

### **Differences Between the Functional Hierarchy and Bayesian Networks**

In Bayesian networks the relationships between variables are probabilistic. The conceptual hierarchy does not feature probability: abstract meanings are generated by partial model algorithms in a deterministic way. Probability is introduced in the functional hierarchy, but there is still no explicitly stated probabilistic relationship between variables.

Uncertainty is only introduced into the system in input and output bits corresponding to future input and output events and this uncertainty is propagated through the system through the processes of probabilistic interpretation of partial model algorithms and downward transfer of probabilities.

Use of this sort of probabilistic hierarchy is described in *Occam's Razor Part 8: Modelling in Artificial Intelligence* [5].

## **Planning of Actions**

### **How Planning Works**

Planning is the process by which the system determines what outputs it should make. An important part of the planning process is the hierarchy's natural tendency to predict the system's own outputs, which was mentioned previously and is a consequence of including output bits into the bottom level of the hierarchy from which higher level meanings are derived.

What I will now describe is only one approach to planning. Other methods may be possible, but they should all have the same general characteristics of using the statistical predictions by the functional hierarchy of the system's own behaviour to provide statistical constraint in some kind of secondary system which searches for optimum behaviour.

Planning in this system involves four main elements:

- the functional hierarchy
- the output vectoring system
- constraint of the output vectoring system
- the situational evaluation function

The functional hierarchy has already been described. Descriptions of the other elements will now follow:

### **Output Vectoring System**

The output vectoring system performs a tree search involving possible future outputs. This tree search has some similarity with the tree searches performed by chess algorithms [6,7].

The purpose of the tree search is to determine the consequences of different sequences of outputs. This is not so that a particular sequence of outputs can be selected, but so that the best value for the next output to be made can be determined. Each output alters the situation in reality and alters the affects of any subsequent outputs, so we have to model future sequences of outputs, just as chess algorithms have to model future sequences of moves.

The tree search run by the output vectoring system involves making outputs in chronological order. This means that the output vectoring system must consider the output bit corresponding to the next output that the system will make, followed by the output bit corresponding to the next output event that the system will make after this and so on.

Each output bit corresponds to a node in the search tree and each decision about whether the output should be “0” or “1” corresponds to a branch. The output bit corresponding to the next output event to occur is the top node in the search tree. From this node, branches for “0” and “1” values being selected for that output go to other nodes. At the end of each of these branches is the next node, corresponding to the next output bit in chronological order.

Each path down through the search tree terminates at some node. In the simplest search trees, the criterion for termination is search depth; for example, a simple chess algorithm may explore hypothetical futures three moves ahead. The criterion for termination of this search tree is not simply based on search depth, but involves a form of constraint of the output vectoring system that depends on the functional hierarchy.

### **Simulation and Propagation**

The above planning process is one of *simulation* of the effects on the hierarchy. At each node there is a choice of two values for a particular output event – “0” or “1”. When a particular output is chosen – that is to say, when the decision has been taken to follow a branch in the search tree – the effects of following that branch are simulated in the functional hierarchy. This means that:

- The probability value of the relevant output bit in the functional hierarchy is set to one of two values – 0 if the selected output is “0” or 1 if the selected output is “1”.
- The effects of this probability change are propagated up the hierarchy by meaning extraction by partial models and down the hierarchy by downward transfer of probabilities. This will automatically cause probability values to change in the bottom level for future input and output events – equivalent to predictions of the future experiences and behaviour of the system being made based on the decision to make that output.

All of these changes are simulated: they are not permanent in the functional hierarchy. Any change to an output value, and the changes propagated up and down the hierarchy as a result, only apply below the node in the search tree for which the changes were made. This is the situation that exists in chess algorithms: hypothetical moves made at some node in a search tree are only in effect below that node.

One way of dealing with this is to make the changes due to selection of an output value and its propagation in the original functional hierarchy, but also to store the previous probability values, so the changes can be undone when reverting to a previous node in the search tree, a type of method widely used in chess algorithms.

### **Constraint of the Output Vectoring System**

Constraint of the search tree involves allowing it to reach greater depths on those paths that are more likely to occur, based on the predictions of the system’s own future outputs that have been made in the functional hierarchy.

A value known as the *cut-off probability* is defined. The cut-off probability has a real value between 0 and 1.

A value known as the *running probability* is maintained during the search. The running probability is initially set to 1.

Each time the search tree goes down a new branch (that is to say, each time we add a new output to the sequence), we multiply the running probability by the probability of the corresponding output bit having that value, according to the bottom level of the hierarchy, as follows:

If a search tree branch involves making an output of “1” (assigning a probability of 1 to the relevant output bit) then the running probability is multiplied by the probability stored for this output bit in the functional hierarchy. If the search tree branch involves making an output of “0” (assigning a probability of “0” to the relevant output bit) then the running probability is multiplied by the probability of the output having a value of “0” – which is 1 minus the probability stored for this output bit in the functional hierarchy. The resultant value becomes the new running probability.

Any path through the search tree terminates when the running probability is less than the cut-off probability. Following any termination of the search tree the desirability of the hierarchy at the terminating node is assessed by the situational evaluation function.

### **Situational Evaluation Function**

The situational evaluation function is an algorithm that examines the probabilities in input values at the bottom level of the hierarchy and returns a score indicating the desirability of the situation. A similar concept is used in chess algorithms [8]. The situational evaluation function can examine the probabilities for input bits relating to input events in the past, present or future.

The situational evaluation function is similar to the evaluation functions used in chess algorithms, except that instead of being applied to a chess position it is applied to the input bits in the bottom level of the hierarchy.

The purpose of the situational evaluation function is to allow the desirability of the system’s situation to be assessed after the making of hypothetical sequences of outputs.

The score returned by the situational evaluation function at a terminal node is “backed-up” the search tree. This means that it is sent up to the node above. This is an approach common in chess. There will have been two branches leading down from that node – one for the “0” output being selected and one for the “1” output – so two backed-up scores will be received by the node. The higher of these two scores is selected as the score for that node and this score in turn is backed up so that, ultimately, scores are available for

values of “0” and “1” being selected for the earliest (next) output event bit in the search tree.

### **Where Planning Occurs**

It may seem that the output vectoring system is functioning as a planning system. In fact, in another article [9] I also referred to it as “the planning system”, but most of the system’s planning is occurring outside the output vectoring system. The system’s predictions of its own future behaviour, and the use of these predictions to constrain the output vectoring system’s search for optimum behaviour, mean that the predictions of behaviour in the hierarchy are driving planning. Planning and prediction are closely linked and we can say, for an AI system:

*Planning of behaviour is prediction of behaviour.*

### **Justification of the Planning Approach**

Why should this approach cause desirable behaviour? The hierarchical system automatically generates predictions of the system’s future observations. If the system were observing some other intelligent agent in the environment then the hierarchy would automatically predict the future intelligent behaviour of this other agent. There is nothing special about agents in the environment, so a predictive modelling system like the functional hierarchy should model their future behaviour along with anything else in reality. This would be done simply by modelling, without recourse to any special planning system. This is significant because prediction of the future behaviour of another agent’s planning means that the modelling system has the capability to plan. All that the system needs to do to exploit this is to do it for its own behaviour. This is done by treating the system’s outputs as inputs and feeding them into the bottom level of the hierarchy, so that the system can observe its behaviour and will automatically predict its future behaviour – which is equivalent to planning its future behaviour.

### **The Role of the Output Vectoring System in Improving Behaviour**

An obvious question about this is why planning of behaviour based on prediction of future behaviour should be of any use. If the system were already acting intelligently then modelling of its future behaviour by the hierarchy should predict (and therefore plan) future intelligent behaviour, but what if the system is initially behaving stupidly? Would modelling of its future behaviour not just produce more stupid behaviour?

This is where the output vectoring system comes in. Even if the functional hierarchy is not making any useful predictions about the future behaviour of the system, the output vectoring system will still attempt to make the best outputs: it just will not have the benefit of any effective constraint. This will still be better than nothing and after this behaviour has occurred for a while the functional hierarchy will be predicting it and probability values for the outputs in the bottom level of the hierarchy will be constraining the output vectoring system’s search appropriately. This means that behaviour that was

previously the best behaviour that the output vectoring system could find is now the “default” behaviour for the system, meaning that there is an inbuilt preference in the output vectoring system for it. The output vectoring system will still search for the optimum behaviour, but the type of behaviour previously found to be optimum is now readily available to it so it can use its processing resources to find an improvement in this behaviour. This means that the behaviour can get slightly better.

This does not need to stop here. When the system has been managing to make improvements for a while then *this improvement itself is part of the behaviour of the system and the functional model will start predicting future outputs in which this improvement continues!* The functional hierarchy will eventually start improving the system’s behaviour by itself, because the best model of what it is going to observe next, based on what it has observed previously, is a smarter system. It should be clear from this that, although vital, the role of the output vectoring system in what is going on is actually quite shallow.

### **Prediction of behaviour is not simplistic**

What I have said here could be misunderstood as saying that the system stores past behaviour and repeats it in a simple way, and that the system can only continue past behaviour and not originate any new behaviour.

This would involve a misunderstanding of what it would mean to say that the future behaviour is a continuation of past behaviour. It simply means that the future outputs and past outputs are related to each other by being described by the same predictive model in the functional hierarchy. There is no reason why this model should be simple with the future outputs being easily discernible from past outputs. The model could actually be complicated.

Some people would still argue with this, saying that, even if the future behaviour and past behaviour are linked by a complicated model it is still the same model, so the system can never really originate anything new, but this is just semantics with the meaning of “originate anything new”. Such people should probably reject the idea of computational intelligence in general. In *any* computer system, future behaviour must be linked to past behaviour by being generated by the same algorithm and there is no reason to think that our own behaviour is not similarly limited. Some people may talk about “algorithms that can change themselves” or “computers that can alter their own hardware” but thinking that such things are profound is a fallacy, as in any such system an invariant algorithm could be used to describe the process of such an algorithm changing itself. There is no reason to think that “original” human thoughts do not occur within the same limitations, a subject discussed by Hofstadter [10].

## **No Separate Planning System**

There is no distinction between modelling and planning. Planning has been absorbed into the modelling system and the functional hierarchy is used for both prediction and planning.

This is different to other approaches in which there are separate planning and modelling systems. It should not be confused with approaches in which the planning system and modelling system use the same principles and are closely linked together. As an example, the approach suggested by Hawkins uses a “sensory” hierarchy based on extraction of meanings from inputs to produce a model, with successive layers of the hierarchy providing more abstraction. A similar “action” hierarchy deals with actions and generates actions in an abstract form. These are then passed down the hierarchy and become less abstract, eventually reaching the system’s outputs where they are realized as the system’s actions. The two hierarchies are supposed to be “closely-coupled” so that there is lot of involvement between meanings abstracted from inputs at a given level of the hierarchy, on the way up, and actions on the same level getting “unabstracted” on the way down. With the approach that I propose there is no such need for any “close-coupling” between input and output hierarchies as the distinction between modelling and planning is removed, leaving only one hierarchy. What is done in the Hawkins hierarchy by the separate planning system is done in this system by downward transfer of probabilities in the hierarchy of extracted meanings – and there is nothing particularly special about planning: the probabilistic predictions of the system’s behaviour, used in constraint of the output vectoring system, are just one aspect of the system’s predictions of future observations (in this case, its predictions of its future observations of its own inputs) and the hierarchy is not doing anything specifically to make these predictions.

*Planning is merely an incidental function of a predictive, probabilistic modelling system in which the system’s observations include its own outputs.*

Some readers may point out that planning does not just happen in the functional hierarchy because the output vectoring system is needed and this is a separate system, but the output vectoring system does not really do much compared to the hierarchy. It exists to give a direction to the system’s behaviour by relating it to the criteria for desirability and by the time probability values have been sent down to the bottom level of the hierarchy the system has really already decided what it is going to do.

## **Conversion and Limits on Performance**

I have described the way that planning works as allowing the system to incorporate previous patterns of improvement in its behaviour into the functional hierarchy, so that continuation of the improvement becomes automatic without the need for the output vectoring system to provide it. This may sound suspiciously like a claim for a “free lunch” situation involving an unlimited runaway process of improvement, but this is not the case: the system is ultimately dependent on the functional hierarchy.

An AI machine must construct a model and *convert* this modelling information into planning and actions. When there is any distinction between planning and modelling, conversion is an explicit process and is a bottleneck through which all the information must go. The sophistication of the modelling system is irrelevant if the only affect that it can have on planning is that which is permitted by an inefficient conversion process.

We can therefore think of this in terms of *modelling-planning conversion efficiency*. Modelling-planning conversion efficiency is a measure of the amount of sophistication in the model that gets through to be expressed in terms of planning and acting.

Conversion efficiency is total when planning is completely integrated into modelling, as has been attempted with this system. There is a small loss of conversion efficiency due to the need for the output vectoring system, but I do not see any way of avoiding at least some minimal system like this. The extent of the integration does, however, mean that conversion efficiency is near total.

Total conversion efficiency, combined with a functional hierarchy *with unlimited performance*, would lead to unlimited performance in planning, but this should not surprise us. That is what “total conversion efficiency” would have to mean. It is a situation that cannot arise in reality.

In this system we have *near* total conversion efficiency, with a modelling system (the functional hierarchy) that has limited of processing power. The system’s planning capability will be limited by the computing resources available to the functional hierarchy and, to a lesser degree, the output vectoring system.

This approach to planning is discussed in *Occam’s Razor Part 9: Representation and Planning of Actions in Artificial Intelligence* [9].

## **Use of Outputs to Control Internal Prioritization**

### **The Carpet Texture Problem**

The *carpet texture problem* is the problem of how an AI system should prioritize use of its computing resources. It gets its name from the idea of a machine analysing the pattern in a carpet and finding many interesting, but irrelevant, things about it. Just because an AI system can analyze things, this does not mean that it should analyse everything. There are two obvious questions which a solution to the carpet texture problem needs to address in this system:

- **What partial model algorithms should be made?** There is an infinite set of partial model algorithms available and it is only feasible to generate some of them. The use of a hierarchical system allows us to rely on shorter partial models, but we still have the problem of deciding which ones to make.
- **What partial model algorithms should be used?** It will not be appropriate for every partial model algorithm to be used in every situation. The same partial

model algorithm can be applied to the data which it processes for more than one index, so a partial model algorithm may only be relevant for some indices at various times.

It could be argued that both of the first of these questions are really contained in the second. Instead of thinking about “making” partial model algorithms we can imagine the entire infinite set of partial model algorithms being available to the computer, and the decision to use one of them in a particular situation and at a particular index, and select it from the set of all possible partial model algorithms, being really equivalent to the decision to “make it”. The general question is this: how should use of computing resources be prioritized in the functional hierarchy?

### **Equivalence of the Carpet Texture Problem and Planning of External Behaviour**

If the system does not use adequate computing resources where it should then there will be much uncertainty in areas critical to its behaviour and it will fail to find behaviour that leads to high situational evaluation function scores. If the system uses inadequate resources for trivial purposes then this will reduce the resources available for more important uses, again reducing its score. The criterion for optimum use of computing resources is that it would allow the system to achieve optimum scores in the output vectoring system’s situational evaluation function, because this is all that the system is intended to do.

The same thing can be said about the system’s behaviour in general! Optimum making of outputs by the system maximizes situational evaluation function scores and optimum prioritization of use of computing resources in the functional hierarchy also maximizes situational evaluation function scores.

*The criteria for optimum behaviour by the system and optimum prioritization of use of computing resources in the functional hierarchy are the same.*

and therefore:

*The same process should be used to optimize the system’s behaviour as is used to prioritize the use computing resources in the functional hierarchy.*

which means:

*Prioritization of the use of computing resources in the functional hierarchy should be by means of outputs designated for this purpose.*

### **Use of Outputs**

As well as outputs that affect things outside the system, some of the outputs should be set aside for controlling prioritization of computing resources in the functional hierarchy. These outputs would control express decisions about the prioritization of the use of

different partial model algorithms, with different indices, in the functional hierarchy. The functional hierarchy would use its limited computing resources to do as much processing as it can, with the prioritization set by these outputs and changing continuously as the outputs change.

The advantage of this is that no separate method is required for the system to determine the prioritization to be used. The existing methods of planning will automatically deal with it. The system does not even need to “know” which of its outputs are conventional and which control prioritization in the functional hierarchy: as far as the system is concerned, outputs relating to prioritization may as well be external outputs.

Such an approach would need some work. The way in which the prioritization control outputs actually control the prioritization would need to be specified. For it to be efficient, a single control output would need to be able to control a number of partial model algorithms. There are some issues to be resolved here, so at this stage I will just take the general position that I have outlined.

### **Turning Complexity Inwards**

The carpet texture problem is a serious issue and any general purpose AI proposal that does not address it will fail. Saying that the carpet texture problem is about increasing efficiency trivializes the role that prioritization of computing resources has in generating a lot of the complexity in an AI system.

The planning capability intrinsic to the functional hierarchy of the AI system already directs complexity outwards by imposing some of the information in the hierarchical model on the environment in the form of actions. The suggested approach to the carpet texture problem is one of turning some of this complexity inwards and back into the functional hierarchy itself, by diverting some of the system’s “outputs” into control of internal prioritization – which in turn contributes to the more efficient planning of these outputs.

### **Analogy**

This approach of turning complexity inwards has a broad analogy in Paulo Soleri’s “arcology” philosophy of urban planning [11] and the emphasis on miniaturization and inward focusing of complexity in nature [12] that Soleri points out. I am not saying that too much should be read into this – just that people familiar with these ideas will recognize some general similarity.

### **Conclusion**

I have outlined how AI could work. Previous articles, to which I have referred, have described these concepts in more detail.

Here are some of the main characteristics of the AI system that have been discussed in this article:

- The system is based on the functional hierarchy – a probabilistic hierarchy of extracted meanings.
- The functional hierarchy is based on the conceptual hierarchy, which has no probabilistic nature.
- Meanings extracted in the functional and conceptual hierarchies are temporal.
- No separate system is used to plan outputs. This is done in the functional hierarchy itself as part of the system’s modelling. There is a separate system needed – the output vectoring system – to give the system’s behaviour a direction and relate it to the concept of desirability.
- Planning of behaviour is prediction of behaviour.
- Planning is merely an incidental function of a predictive, probabilistic modelling system in which the system’s observations include its own outputs.
- When the system has been managing to make improvements for a while then *this improvement itself is part of the behaviour of the system and the functional model will start predicting future outputs in which this improvement continues!* The functional hierarchy will eventually start improving the system’s behaviour by itself, because the best model of what it is going to observe next, based on what it has observed previously, is a smarter system.
- The carpet texture problem is solved by designating some of the system’s outputs as prioritization control outputs. These outputs are used to control prioritization of the use of computing resources in the functional hierarchy.

There are issues still to be resolved:

- The way in which the hierarchy and the partial models works, as described in the previous articles [4,5], seems too rigid to me, in the sense of relying on matrices with fixed numbers of dimensions, so further consideration of the ontology is required. This is really an issue of indexing used to differentiate between different applications of, and meanings produced by, the same partial model algorithm. The current indexing system lacks flexibility.
- I have said in previous articles that the way in which partial model algorithms are constructed needs to be considered. In this article I have discussed the carpet texture problem and the two issues are related. If we consider the set of all partial model algorithms being available to an AI system then the consideration of which partial model algorithms to use deals with the consideration of which partial model algorithms to make. The issue of how prioritization of computing resources in the functional hierarchy will be controlled by the system’s own outputs needs further consideration.

## References

[1] Hawkins, J., Blakeslee, S. (2004). *On Intelligence*. New York: Henry Holt.

- [2] Web Reference: George, D., Hawkins, J. (?). *Belief Propagation and Wiring Length Optimization as Organizing Principles for Cortical Microcircuits*. Retrieved 24 April 2006 from <http://www.stanford.edu/~dil/invariance/Download/CorticalCircuits.pdf>.
- [3] Web Reference: Almond, P. (2006). *Occam's Razor Part 6: Partial Models as "Envelopes"*. Retrieved 1 March 2006 from <http://www.paul-almond.com/OccamsRazorPart06.htm>.
- [4] Web Reference: Almond, P. (2006). *Occam's Razor Part 7: Hierarchy and Ontology*. Retrieved 30 April 2006 from <http://www.paul-almond.com/OccamsRazorPart07.htm>.
- [5] Web Reference: Almond, P. (2006). *Occam's Razor Part 8: Modelling in Artificial Intelligence*. Retrieved 9 June 2006 from <http://www.paul-almond.com/OccamsRazorPart08.pdf>.
- [6] Levy, D.N.L. (1984). *The Chess Computer Handbook*. London: Batsford. Chapter 3, pp38-52.
- [7] Heinz, E, A. (2000). *Scalable Search in Computer Chess: Algorithmic Enhancements and Experiments at High Search Depths*. Vieweg Verlag. Chapter 0, pp11-18.
- [8] Levy, D.N.L. (1984). *The Chess Computer Handbook*. London: Batsford. Chapter 2, pp7-37.
- [9] Web Reference: Almond, P. (2006). *Occam's Razor Part 9: Representation and Planning of Actions in Artificial Intelligence*. Retrieved 29 July 2006 from <http://www.paul-almond.com/OccamsRazorPart09.pdf>.
- [10] Hofstadter, D. R. (2000). *Godel, Escher, Bach: an Eternal Golden Braid*. London: Penguin Books. Chapter 15, pp465-479. (Originally published:1980. New York: Vintage)
- [11] Soleri, P., Strohmeir, J. (2001). *The Urban Ideal: conversations with Paolo Soleri*. Berkeley: Berkeley Hills Books. pp94-95.
- [12] Soleri, P. (1973). *The Bridge Between Matter and Spirit is Matter Becoming Spirit: The Arcology of Paolo Soleri*. New York: Anchor Press/Doubleday. pp22-23.