

The Emergence of Higher Levels of Description

Chris Thornton

Cognitive and Computing Sciences
University of Sussex
Brighton
BN1 9QH
UK

Email: Christopher.Thornton@firenet.uk.com

Tel: (44)1273 678856

May 21, 2003

Abstract

The paper considers what is involved in the emergence of higher levels of description. It suggests that we can think of the entities making up a particular level of description as *virtual entities* and argues that concept learning involves forming representations of *degenerate* virtual entities. These are virtual entities which play no role in the emergence of higher levels of description. The paper also explores the way in which the formation of intermediate levels of description allows an agent to trade off computational costs against representational costs.

1 Introduction

The paper attempts to answer two questions about the things we call *levels of description*; namely, What are they? And, Where do they come from? There are two reasons for being interested in the emergence of higher levels of description.

- The process seems to be closely related to the acquisition of knowledge through concept learning. In learning some new set of concepts we move from a relatively concrete, object-based view of the world to a relatively abstract, category-based view; i.e. we move to a more abstract level of description. Thus we can think about concept learning in terms of the development of new levels of description.
- The process seems to play an important part in scientific discovery. The development of a new theory often allows us to understand some range of

phenomena in terms of a new set of concepts.¹ The set provides a new level of description; so, we can think of the development of the theory in terms of the elucidation of a new level of description.

In the first part of the paper (section 2) we try to bring out what is meant by the phrase **level of description**. In the second part of the paper (section 3) we consider the way in which higher levels of description might emerge in given cognitive agents.

2 What is a level of description?

Typically, the subject of a description (e.g. a state, mechanism, or object) is characterised (by the description) in terms of (1) a set of constituent entities and (2) a set of relationships that bind them together. In most cases a given subject can be characterised in terms of several different classes of entity. For instance, a meteorological state can be characterised in terms of relationships between high and low pressure systems; or it can be characterised in terms of relationships between troughs and fronts; or it can be characterised in terms of relationships between air masses. In principle, it can even be characterised in terms of relationships between particles.

It seems that when we say a characterisation is at a particular level of description we are noting that it is expressed in terms of entities drawn from a particular class. Moreover, when we say some characterisation is at a *higher* level of description to some other characterisation we appear to be asserting that it is expressed in terms of entities which can themselves be characterised at the lower level of description. For example, in stating that a characterisation of a meteorological state expressed in terms of pressure systems is at a higher level of description to one expressed in terms of troughs and fronts, we draw attention to the fact that pressure systems can themselves be characterised in terms of relationships between troughs and fronts.

The implication appears to be that, in general, any set of entities is associated with just one level in a hierarchy of *possible* levels of description. At any particular point in the hierarchy there are levels above and levels below, and an entity at a particular level can be always be understood in terms of some set of entities at a lower level.

2.1 Examples

Consider a complete set of playing cards (labelled D below). What entities can be understood in terms of this set? Firstly, of course, we have the individual playing cards. Each of these can be understood in terms of an entity from the set, namely itself. At a higher level we can have an entity like a black card. This can be understood in terms of some subset of D (the subset of all spades and

¹For instance, the development of Newtonian mechanics made it possible to understand physical systems in terms of concepts such as gravitational pull and inertia.

clubs). We can also have an entity like a poker hand. This can be understood in terms of a subset of subsets of D (the subset of all subsets of D whose elements form legal poker hands.)

Moving up another level we can have an entity like an unbeatable hand. This can be understood in terms of some subset of poker hands (i.e. the set of hands which have maximal values). We can also have an entity like a close game. This can be understood in terms of some subset *of subsets* of hands (the subset of all collections of hands such that one hand beats all the others by a small margin).²

The more primitive the basic entities, the easier it is, in general, to map out *higher* levels of description. But it may still be possible even in the case where we set the basis entities to be complex, **high-level** objects. For example, imagine that the set of primitive objects D is the set of all human beings. Particular human beings can be understood in terms of particular elements of this set. Particular social groups can be understood in terms of sets of human beings which collectively exhibit certain properties (i.e. relationships). Interactions between social groups (e.g. class wars) can be understood in terms of sets of social groups which collectively exhibit certain properties. And so on.

A fairly well-worn example concerns computers. A functional unit such as a CPU can be understood in terms of sets of primitive electronic components (logic gates etc.). A machine instruction (e.g. shift-right) can be understood in terms of (interactions between) a set of functional components (CPU, memory etc.). A program or process can be understood in terms of sets of instructions, and an operating system can be understood in terms of sets of programs.³

2.2 Virtual entities

The general principle underlying the implementation of levels of description seems to be fairly straightforward. Each level in a LOD hierarchy corresponds to a set of entities; and an entity associated with a particular level of description is seen as made up of collections of entities at lower levels of description. For Xs to be at a higher level of description than Ys it is necessary that a given X can be seen as consisting of some set of Ys that collectively exhibit certain properties; i.e. it seems to be necessary that Xs can be understood *in terms of* Ys.⁴

There are a number of possible ways in which we may be able to understand a given entity in terms of a given set of basic primitives and the *way* in which we understand a particular entity effectively tells us what level it is at with respect to the primitives. We may be able to understand the entity in terms of one

²We might also identify lower levels of description although this would involve **unpacking** the basic entities rather than glueing them together into subsets. For instance, each playing card can be understood in terms of sets of atoms and each atom can be understood in terms of sets of elementary particles.

³For further descriptions of elaborate LOD hierarchies the reader is referred to [1, Chapter 10] and [2].

⁴In fact, as we show below, in some cases it is only necessary that the X can be understood in terms of a singleton set of Ys; i.e. a single Y.

element of the set, or in terms of a subset (of subsets) of the set, or in terms of entities which can be understood in terms of the set, or in terms of entities which can be understood in terms of entities which can be understood in terms of the set. To see what level an entity is at (with respect to the primitives) we simply need to count the number of times we have to reiterate the phrase **in terms of entities** as we describe how the entity can be understood in terms of the primitives.

Given the assumption that the primitives form the level 0 entities. Then an entity which can be understood in terms of level 0 entities is at level 1. An entity which can be understood in terms of level 1 entities is at level 2. And so on. In computational terminology we should say that the entities at level n ($n > 0$) are *virtual entities* with respect to level $n-m$ ($m > 0$) entities. They do not exist in the fullest sense of the word. They are merely hypothetical compositions of lower level entities. With respect to the set of playing cards, the ace of spades is a level 0 entity, a poker hand is a virtual entity at level 1 and a close game is a virtual entity at level 2.

2.3 Degenerate entities

In mapping out the levels of description which seem to relate to some set of primitive objects we frequently come across *degenerate* cases. A simple example is the black card entity. Understood in terms of the set of playing cards, this is a level 1 entity. But whereas the black card can be understood in terms of a subset of level 0 entities other level 1 entities, such as the poker hand have to be understood in terms of subsets of level 0 entities. Why is this?

In accordance with the principle noted above, both the poker hand entity and the black card entity are made up of collections of level 0 entities (members of D) which collectively exhibit certain properties. But in the case of the black card the collections are all *singleton* sets. Thus although, in general, we can understand a level n entity in terms of a subset of subsets of level $n-1$ entities, in the case of things like the black card, the **of subsets** is redundant, since each one has only one element.⁵

Degenerate entities turn out to play a very limited role in the ‘implementation’ of higher levels of description. To see why, imagine that an X is a degenerate entity made up of a singleton set of Y s. And that a Y is a degenerate entity made up of a singleton set of black cards. (And remember that a black card is made up of a singleton set of cards.) In this case an X is, literally, the same thing as a Y which is the same thing as a black card. The X and Y are therefore redundant; they do not exist as entities separate from the black card.

Contrast this with the case where an X is made up of non-singleton set of Y s which collectively exhibit certain properties and where a Y is a poker hand. (And remember that a poker hand is made up of a set of cards which are related

⁵Compare the unbeatable hand. With respect to the set of playing cards, this is made up of a singleton set of level 1 entities.

together in a certain way). Here an X is definitely not the same thing as a Y; and a Y is not the same thing as a card. Thus both the X and the Y do exist as separate entities at higher levels of description.

2.4 Properties

We have said that level n entities appear to be made up of sets of level n-1 entities which collectively exhibit certain properties; but we have not considered what might be meant by **made up of**. Let us think about an entity which is at level 3 with respect to the set of playing cards D; namely an **exciting match**. Let us assume that an exciting match is a sequence of (poker) games in which roughly half of the games are close and half are easy wins.

A poker hand, in this context, is an entity in terms of which it is possible to understand the close game entity. And a close game is an entity in terms of which it is possible to understand the exciting match entity. But when we understand a close game in terms of poker hands we pay no attention to the cards which make up the hands in question. We only pay attention to the overall *values* of the hands. Similarly, when we understand the exciting match entity we pay no attention to the games themselves, only to how close the games were.

Thus, when we say that the level 2 entity **close game** is made up of level 1 entities called poker hands, what we are really trying to say is that a set of subsets of cards forms a close game if each of the subsets forms a hand with a certain *property*, namely a value near to the average value. Similarly, when we say that the level 3 entity **exciting match** is made up of level 2 entities we mean that a set of subsets of hands forms an exciting match if each of the subsets forms a game with a certain property, namely a non-typical closeness value. A level n entity, then, exists only if certain subsets of level n-1 entities exist all of which have certain properties.

We can summarise as follows. On the face of it, it seems as if each level in a LOD hierarchy is made up of a set of (virtual) entities. But when we come to look more closely, it seems that, in general, the link between a level n entity and some level n-1 entities is a mapping between subsets of level n-1 properties and the values of a level n property. Rather than understanding higher level entities in terms of lower level entities we seem to understand properties of higher level entities in terms of properties of lower level entities.

On this view we should see a given LOD hierarchy as based on properties of primitive objects rather than on the objects themselves. For instance, rather than thinking in terms of a hierarchy based on a complete set of playing cards we think in terms of a hierarchy based on a set of properties of playing cards. From this perspective, the ace of spades is seen not as an entity but as a property (ace-of-spadeness?) of some hypothetical but unknown entity.

In fact, in what follows, we will continue to think of a LOD hierarchy as an ordered sequence of sets of *entities* based on a set of primitive objects. This makes life much easier from the conceptual point of view. However, it is important to bear in mind our observation that understanding an X in terms of Ys actually involves understanding a property of an X in terms of properties of Ys.

2.5 Representations

Now, on the present view, LOD hierarchies are fairly simple structures computationally speaking: they are just hierarchies of mappings. This means that if we know how some given X can be understood in terms of Y s we can easily produce a symbolic (i.e. computational) representation of X in terms of symbolic Y s. That is to say we can construct an implementation which can decide whether an X exists given a set of symbolic Y s as inputs.

How would the representation be implemented? In general, in a LOD hierarchy a set of level $n-1$ properties has a level n property. This situation can be captured computationally as a many-to-1 mapping. If the level $n-1$ properties are hand values and the level n property is game-closeness, then the mapping can be implemented as an m -place function which accepts m hand values and returns a single output denoting the game-closeness property of the corresponding game. A complete LOD hierarchy is just a hierarchy of such mappings and can therefore be captured as a hierarchy of m -place functions (with the assumption being that lower level functions are computed before higher level functions).

The highest level in a LOD hierarchy might be constituted of a number of entities (properties). But for each entity we have, in the computational representation, one function which returns a property of that entity. The inputs for this function are provided by functions at the level below. The inputs to those functions are provided by the functions at the level below that, and so on. Thus, each top level function forms the root of a tree structure and a complete LOD hierarchy is implemented as a *set* of trees.

An individual tree forms a *representation* for a given level n virtual entity X in the sense that it is a mechanism which effectively reproduces the process by which we understand X in terms of Y s (and Y s in terms of Z s etc.) only *in reverse*. That is to say, if X (an exciting-match) is an entity which we understand in terms of Y s (games), and Y s are entities which we understand in terms of Z s (hand values), then a computational representation of X (of the form described above) is a mechanism which accepts symbols standing for Z s (hand-values) and, ultimately, returns X (an exciting-match) if and only if the Z s form an X .

2.6 Generalised concepts

It may seem as if this sort of **function hierarchy** is a strange and abstruse mechanism. But in fact, from the computational point of view, it is not strange at all: it is just a *generalised concept*. Note that for some sets of level $n-1$ properties, the output of a level n function will be undefined. This is because there will be some subsets of level $n-1$ entities which do not map onto any particular level n entity. We can take account of this by assuming that the functions forming the representation return either a level n property or some special object as output (called <undefined> below).

Now, a function which represents a degenerate level n entity in terms of some subset C of level $n-1$ entities is, technically, a concept covering C . That is to say, it is a function which returns a level n property if the input is in C , and

<undefined> otherwise. The function effectively partitions the set of all level n-1 entities into two subsets—and this is precisely, the behaviour of a computational concept (cf. [3,4]).

An example may help here. Consider the black card entity again. In terms of our framework, this is a (singleton) collection of playing cards which exhibit certain properties; i.e. it is a card which has certain properties (namely, spade-ness or club-ness). In a hierarchical representation the link between the level 0 entities (playing cards) and this particular level 1 entity is a function which accepts (a singleton set containing) one level 0 property and returns the level n property in question (black-card-ness) if the corresponding level 0 entity is black, and <undefined> otherwise. The function counts as a concept covering the set of all black cards because it effectively partitions the set of all cards into two subsets: the subset of black cards and the subset of non-black cards.

Thus a concept is, in the current framework, a representation of a *degenerate* virtual entity; i.e. a level n property which is made up of singleton sets of level n-1 properties. Now, in general, level n entities are made up from non-singleton sets of level n-1 entities. The implication is that function-hierarchy representations can be thought of as *generalised* concepts. This result is satisfying from the intuitive point of view, since (1) higher levels of description are typically thought of as being made up of sets of concepts, and since (2) concepts are typically thought of as identifying abstract entities.

2.7 Generalised extensions

The entities covered by a concept form its *extension*. But generalised concepts have extensions at various levels of description. Moreover, each part of the extension is made up from *subsets* of lower level entities which collectively exhibit certain properties. Let us illustrate this using the black card example again. (From here on we will denote the symbol for X as <X>, a set of symbols for Xs as <Xs> and a representation of X in terms of Ys as <X>/<Ys>.)

If we view <black card>/<playing cards> as an ordinary concept it has an extension and this is just the set of all black cards. If we view <black card>/<playing cards> as a representation of a level 1 (with respect to the set of playing cards) virtual entity it has no extension as such; however when we enumerate the set of all (singleton) subsets of level 0 entities which make up a <black card> we end up with precisely the set of all black cards. Thus, it makes no difference whether we view a given function hierarchy as a concept and derive its extension, or whether we view it as a representation of a virtual level n entity and derive the set of (subsets of) level n-1 entities which can go to make it up. Either way we end up with the same set.

By generalising in a straightforward fashion from this degenerate case we arrive at the notion that the extension of a level n concept at level m is the set of subsets of level m entities which when combined together in a certain way, form an instance of the concept. Thus the level 0 extension of <poker hand>/<playing cards> (a representation of a non-degenerate level 1 entity) is made up from *non*-singleton subsets of level 0 entities. The extension of <close

game>/<playing cards> at level 0 is the set of subsets of cards which when combined together appropriately form an instance of this concept. Its extension at level 1 is the set of subsets of <poker hands> which can combine together in the required manner.

We will denote extensions using the bar notation. Thus the level 0 extension of <X>/<Ys>, for instance, is denoted $|\langle X \rangle / \langle Ys \rangle|$. To denote a higher level extension we simply change the specification of the *representation*. For instance, if Zs are entities at level 1 and X is an entity at level 2 then we would denote the level 1 extension of <X>/<Ys> as $|\langle X \rangle / \langle Zs \rangle|$.

2.8 Tree diagrams

Representations of higher level virtual entities are just tree structures. As such they can be visualised very easily. In this section, a number of example representations are presented and these are drawn out as trees in the conventional fashion with the root(s) at the top and the leaves at the bottom. The leaves in these diagrams are always functions which accept (symbols for) level 0 properties and produce level 1 properties; the roots are the functions which produce the top-level properties.

Each node in a representation is a function which returns a level n symbol (i.e. a symbol for a level n property) when given a set of level n-1 symbols (or <undefined>). Thus, each function has n input parameters. We draw an n-place function as follows.

$$\mathbf{f1}(, ,)$$

Here **f1** is just an arbitrary label for a function. The empty slots between the parentheses represent its input parameters. So as not to be misled into thinking that the ordering of the inputs is somehow significant, it is best to think of the function as a black-box and the empty slots as input terminals. Actual inputs are obtained from the functions (black boxes) representing level n-1 entities and we denote this input/output link by drawing arcs from the functions in question to the corresponding input terminals; e.g.

$$\begin{array}{c} \mathbf{f1}(, ,) \\ | \\ | \\ \mathbf{f2}(, ,) \end{array}$$

Here the 1st input to **f1** is the output of **f2** (which is a 2-place function).

To make things a little clearer we can place the label of a given level n entity on the output arc of the function which generates the corresponding property. Also, in the case where a function takes its inputs in some unspecified way from some set of properties we can denote this by putting the label of the corresponding set of entities immediately below the function(s) in question. Thus we might draw out the hierarchy forming a representation of <black card> in terms of <playing cards> as follows.

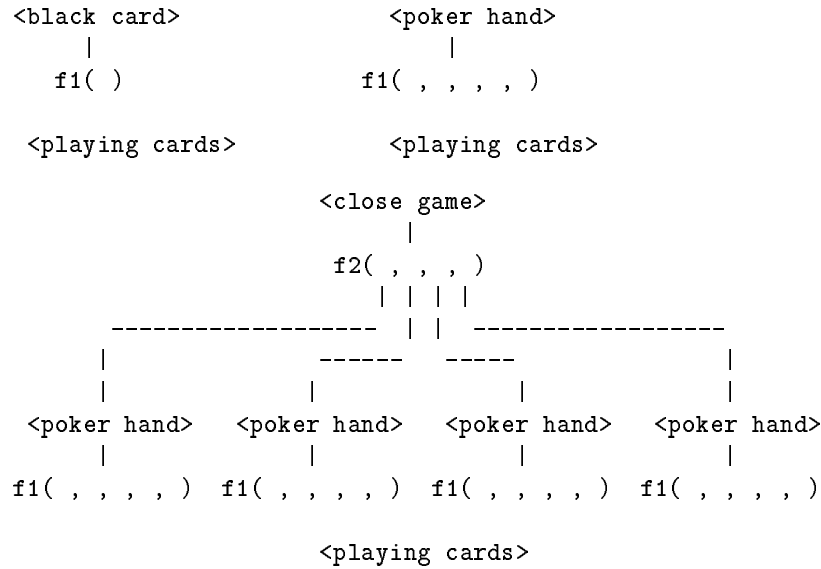
```

<black card>
  |
  f1( )

```

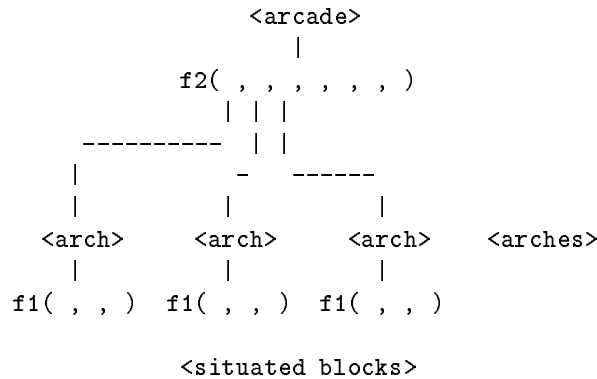
<playing cards>

Several representations using the <playing cards> primitives (including <black card>/<playing cards>) are shown in the figure below. Note that the function which produces properties of <poker hands> is 5-place. The function which produces properties of <close games> is 4 place—the assumption being that any poker game involves only 4 players.

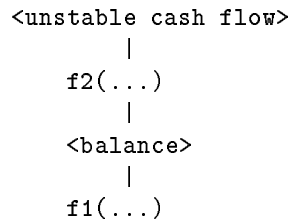


Representations for a collection of entities all of which can be understood in terms of situated blocks (i.e. particular blocks at particular locations in a particular blocksworld) are shown in the figure below. These include representations for the level 1 entity <arch> and the level 2 entity <arcade> (cf. [5,6]). The function that produces properties of <arches> is 3-place since an arch is made up of a set of three blocks which are related together in a certain way. The root of the representation of <arcade> is a 7-place function, the assumption being that any arcade is made up of a collection of seven arches. Only some of the representations of level 1 entities (<arches>) are shown.





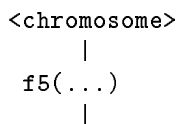
The figure below shows a representation of an <unstable cash flow> entity in terms of a set of <financial transactions> primitives. (Note that here and below we denote an n-place function where n is unknown as $f(\dots)$.) The thinking behind this representation is as follows. A particular daily balance for a firm is a property of a set of (daily) financial transactions. An unstable cash flow is a property of a set of daily balances. Hence, <unstable cash flow>/<financial transactions> is a level 2 representation.

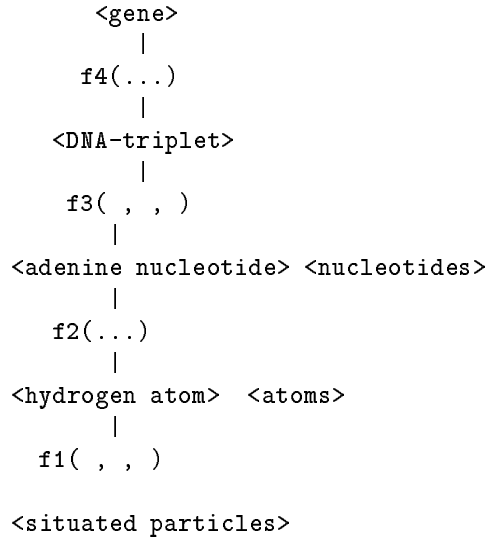


<financial transactions>

In the figure below we see a rather more complex representation. The set of primitive objects (level 0 entities) is here assumed to be the set of situated physical particles; i.e. the set of all possible particle/location pairings. The top level symbol is <chromosome>. The structure is derived by looking at how higher level entities consist of sets of lower level entities which collectively exhibit certain properties (i.e. by doing top-down analysis).

In this representation, a <chromosome> is made up of sets of <genes>. A <gene> is made up of sets of <DNA-triplets>. A <DNA-triplet> is made up of <nucleotides>. A <nucleotide> is made up of atoms and an <atom> is made up of sets of particles. Thus <chromosome>/<situated particles> is a level 5 entity.





3 How do higher levels of description emerge?

In the first part of the paper we have tried to characterise what a level of description is *in general*. In the second part of the paper, we will get to grips with the main issue; namely, how do higher levels of description emerge in particular cases? In tackling this problem we will make a number of assumptions and assume special meanings for certain phrases. For instance, we will say that an agent **has access to** a particular level of description if the agent has access to a representation of an entity at that level. And we will assume that having access to a representation of an entity X involves having access to the output of a function which represents X in the manner described above.

We will further assume that a cognitive agent is necessarily embedded in a particular world and always receives some inputs directly from that world. We will think of these inputs as a set of primitive entities; but to maintain uniformity we will assume that receiving inputs from a world implies having access to the outputs of producer functions which take no input and return (properties of) arbitrary primitive entities (i.e. inputs). We will continue to denote the set of primitive entities as D.

3.1 What levels of description are accessible to an agent?

Before considering how higher levels of description might emerge, it is worthwhile considering what levels of description are going to be *accessible* in any particular case; i.e. which levels might emerge in principle. Imagine that F is the set of functions involved in the hierarchies representing Xs in terms of Ys. If the agent is not able to compute every function in F and/or is not able to assemble members of F into the required tree structures, then the agent does

not have access to the level of description made up of Xs *in principle*. It is only if the agent *can* compute all of F, and *does* assemble the required tree structure, that access to this level is obtained.

Thus, the levels of description which are accessible to a given agent depend on the agent's computational abilities and behaviour. We cannot speak, in absolute terms, about *the* LOD hierarchy for some set of primitives. We can only speak about the LOD hierarchy accessible to some particular agent with some particular range of computational abilities.

3.2 Incremental v. one-shot emergence

New levels of description can be implemented either one at a time or all in one go. Moreover, new levels can be implemented in a random fashion or in an ordered sequence. For example, we can imagine a process in which the lower levels in a hierarchy are implemented before the higher levels. Or we can imagine the reverse case—where the higher levels are implemented before the lower levels.

The case of incremental construction is interesting because we can give a fairly precise account of what is involved when a higher (lower) level is implemented on the basis of an existing lower (higher) level. Recall that if Xs are entities at a higher level of description than Ys it must be the case that Xs can be understood in terms of Ys. That is to say, it must be the case that an X can be seen as a set of Ys collectively exhibiting certain properties. This has clear implications for the way in which new levels of description emerge.

3.3 Bottom-up v. top-down emergence

A new level of description for characterising some given phenomena must either be higher or lower than an existing level. Thus, the implementation of a new level must either be a **bottom-up** process (a higher level emerging from a lower level) or a **top-down** process (a lower level emerging from a higher level). In either case, the process by which the new level emerges is fairly straightforward.

The implementation of a higher level of description involves first identifying sets of entities at an existing level of description which collectively exhibit certain properties. The higher level is then implemented in the form of a set of functions which map the sets in question onto the values of one of the properties. The outputs of the functions are (properties of) the entities at the new (higher) level of description.

A classic example of this process was the construction of the level of description commonly referred to as the **genetic code** (cf. [7]). This maps triplets of DNA nucleotides onto symbols standing for one of twenty amino acids (cf. [1, p. 520]). Following the emergence of this new level, a given strand of DNA could be characterised at either the (higher) genetic code level or at the (lower) nucleotide level.

Top-down implementation is a complementary process which involves identifying some set of Ys and some property function(s) such that a set of Xs can be seen as the outputs of the functions when given sets of Ys as inputs. The classic

example of this process was the so-called **reduction** of chemistry to physics. This involved the demonstration that chemical phenomena (and periodicity in particular) could be understood in terms of relationships between the numbers of electrons occupying specific shells of given atoms. Following the emergence of this new level, a given chemical phenomenon could be characterised at either the (higher) chemical level or, in principle, at the (lower) physical level.

3.4 What levels of description will be implemented?

We have noted that a given agent with particular computational abilities will have access to certain levels of description. However, it is fairly easy to show that, in general, an agent will have access to far more levels of description than could possibly be implemented. Even in the very restricted scenario where the agent can compute just one, m -place function, the number of possible level n entities for some given set of primitives (level 0 entities) grows exponentially with the size of n .⁶

Any level 1 entity consists of a set of level 0 entities. Thus a level 1 entity corresponds to a subset of subsets of the level 0 entities. There are therefore as many potential level 1 entities as there are subsets of subsets of level 0 entities. If there are just 10 level 0 entities The maximum number of level 1 entities is $2^{2^{10}}$ ($> 10^{300}$) possible level 1 entities. When we come to consider higher level entities the number becomes even more absurdly large. We can safely assume that no cognitive agent will be able to represent all possible higher level entities for some given set of primitive inputs and must therefore implement some tiny fraction of the total set of possible levels of description. But which levels of description *will* be implemented?

3.5 Useful levels of description

A plausible answer is that the agent will tend to implement the levels of description which are useful; i.e. which help the agent to achieve its goals. A given entity (concept) will be useful if it allows the agent to represent a particular state of affairs—and to discriminate between this state of affairs and other states.

In the present scenario, a particular state of affairs is just some particular class of input sets (i.e. subsets of D). An agent will be able to discriminate between a given state C and other states if it has access to a representation of C ; i.e. to a concept (entity) whose extension at level 0 is the set of subsets in question. Thus, if some class of input states corresponds to a significant state of affairs, then it will be useful, from the point of view of the agent, to construct the concept which represents it, and thereby implement (or at least initialise) the associated level of description.

But, given the assumption that the agent forms a representation for a particular set of subsets, is there any way of telling what its internal structure will be?

⁶We assume that any function returns a unique output for each unique input.

Let us imagine that $|\langle X \rangle / \langle Ys \rangle|$ is the set of subsets of level 0 entities which an agent needs to be able to discriminate from other sets. With unrestricted computational abilities, the agent could, in principle, represent C using a single n -place function, where n is the size of the subsets in question. The output of this function is a property of an entity but there is no straightforward way of telling what level the entity is at with respect to the set of $\langle Ys \rangle$. The computation performed by the function might involve the application of subroutines which do more or less the same job as the $\langle \text{poker hand} \rangle$ concepts in the level 1 representation shown above. In this case, the entity represented is effectively at level 2. On the other hand the computation performed might involve some kind of trivial lookup process. In this case the entity is definitely at level 1.

If the agent is only able to compute the functions in some set F then we know that each level of the implementation is made up from elements of F . If $|\langle X \rangle / \langle Ys \rangle|$ *can* be represented using the functions in F but *not* using any single function in F , then the agent must construct a hierarchy of functions.⁷ But, again, we want to know what the structure of this hierarchy will be.

It seems reasonable to assume that the agent will face some kind of space/speed tradeoff here. That is to say, there will be a tradeoff between the costs of implementing complex LOD hierarchies and the costs of performing complex computations. If the costs involved in performing computations are high relative to the costs of implementing hierarchies, then the agent will optimise by implementing any $\langle X \rangle / \langle Ys \rangle$ using minimally complex functions and a large hierarchy. Conversely, if the costs involved in representing large hierarchies are prohibitive relative to the costs of computation then clearly the agent will optimise by constructing minimal hierarchies.

Thus the emergence of particular levels of description in particular cognitive context depends on the computational properties of the agent(s) in question. But this is not the whole story. It may be the case that there is some implementation $\langle Z \rangle / \langle Ys \rangle$ which (1) forms a representation of a significant state of affairs and (2) forms an entity in terms of which $\langle X \rangle$ can be implemented. In this case, the agent will have an independent reason for implementing $\langle Z \rangle / \langle Ys \rangle$ and may find it advantageous, from the point of view of efficiency, to represent $\langle X \rangle$ in terms of $\langle Zs \rangle$ rather than in terms of $\langle Ys \rangle$. This will certainly be the case, for example, if implementing $\langle X \rangle$ in terms of $\langle Zs \rangle$ allows $\langle X \rangle$ to be implemented using a function of lower arity.

4 Concluding comments

The paper has characterised the emergence of higher levels of description in terms of the implementation of hierarchical data structures. Nodes in these hierarchies are functions which implement (properties of) entities at some particular level of description in terms of subsets of (properties of) entities at a

⁷If F is a singleton set, then we know that every function in the representation is of the same form and for any fixed number of producers, there is exactly one tree structure which the agent can implement.

level below. Concepts were shown to correspond to degenerate entities and the emergence of higher levels of description in particular cases was shown to depend on the computational abilities of the agent, the existence of significant states of affairs and the relationship between computational and representational costs.

Can we draw any conclusions about the way in which human beings construct higher levels of description? The fact that the human brain is (at a low level of description) made up from large numbers of very elementary computational devices (i.e. neurons) might suggest that the costs, to the human brain, of computing arbitrarily complex functions are likely to be high relative to the costs of implementing large data structures. This idea leads one to speculate that the human brain is likely to favour complex, computation-poor LOD hierarchies over simple, computation-rich ones. And that human beings are therefore likely to imbue the world(s) they inhabit with rich and highly structured phenomena.

5 Acknowledgements

The approach taken in this paper arose as a result of feedback generated from an early verbal presentation of the ideas expressed. I am particular indebted to Richard Caley for suggesting that generalised concepts can be seen as computing higher-level entities.

References

- [1] Hofstadter, D. (1979). *Godel, Escher, Bach: An Eternal Golden Braid*. Penguin.
- [2] Hofstadter, D. (1984). The copycat project. A.I. Memo 755, Massachusetts Institute of Technology.
- [3] Valiant, L. (1984). A theory of the learnable. *Communications of the ACM*, 27 (pp. 1134-42).
- [4] Utgoff, P. (1986). Shift of bias for inductive concept learning. In R. Michalski, J. Carbonell and T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach: Vol II* (pp. 107-148). Los Altos: Morgan Kaufmann.
- [5] Boden, M. (1977). *Artificial Intelligence and Natural Man* (1st edition). Hassocks: Harvester Press.
- [6] Winston, P. (1984). *Artificial Intelligence* (2nd edn). Reading, Mass.: Addison-Wesley.
- [7] Watson, J. (1970). *The Double Helix*. Harmondsworth, Middlesex: Penguin.