

Representational Redescription for Sea Slugs

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 discusses the Representational Redescription Hypothesis of Clark and Karmiloff-Smith [1]. This hypothesis — the ‘RRH’ for short — has two main parts to it. Part one suggests that human cognitive development is a process involving the redescription of knowledge at increasing levels of abstraction. Part two is a claim about the true nature of thought. It states that representational redescription is the *distinctive mark* of genuine thinkers. As such it should enable us to distinguish ‘the nature of our thoughts from the information processing states of a sea slug or a VAX mainframe’ [Clark and Karmiloff-Smith, Forthcoming, p. 1]. My paper attempts to flesh-out part one of the hypothesis by giving it a computational model based on a process called ‘representational abstraction’. It then attempts to question the status of part two. It argues that if representational abstraction is a good model for representational redescription, then part two of the hypothesis is suspect since the process in question is one that we would expect even very primitive organisms to engage in.

1 Introduction to the RRH

The RRH is conveniently introduced via the psychological work which underpins it. This has largely been carried out by Karmiloff-Smith [2,3,4] One of her studies [4] looked at children’s ability to draw pictures of ‘funny’ or ‘pretend’ objects. Karmiloff-Smith found that younger children (i.e. those between four and six years old) were more limited in the strategies they were able to bring to bear in developing such pictures. Very young children would typically employ

standard routines for drawing an ordinary version of the object in question and would then append some extraordinary feature to the drawing. For example, in drawing a picture of a man with two heads, the younger children would typically draw an ordinary man first and then add an extra head.

Older children (e.g. children between eight and ten years old) appeared to have access to more sophisticated procedures for producing such drawings. They appeared able to create drawings of anomalous objects without needing to add a ‘wrong’ feature to an essentially correct drawing. They also appeared much more adept at introducing more complex and abstruse types of permutation. For example, they were more able to mix-and-match cross-category elements and to change the overall component structure of objects. (See [5] for an accessible introduction to this work.)

The RRH, as presented by Clark and Karmiloff-Smith, attempts to explain these and related developmental effects. It suggests that by creating redescrptions of their own knowledge, agents are able to explore the possibility of more abstract manipulations and are thus able to achieve more flexibility in action. With respect to the drawing effects, Clark and Karmiloff-Smith suggest that drawing ability in very young children is based on inflexible, ‘hardwired’ routines that must be run through from start to finish. At a later stage the child converts the implicit knowledge captured in these routines into a more explicit or declarative form. This makes it possible to explore variations and manipulations that could not be produced by any one of the original hardwired routines.

While the RRH provides a plausible theory of development in children’s drawing abilities, this is far from being its only application. It purports to be a general theory which applies to all aspects of development.¹ The introduction given above merely scratches the surface of the hypothesis. However, it should provide enough of the flavour of the hypothesis to place the rest of the paper in context. For further details consult the original sources [2,3,4,Clark and Karmiloff-Smith,Forthcoming].

2 The role of a computational model of RR

The RRH provides an essentially philosophical and psychological model of development. But there are good arguments for attempting to extend it ‘downwards’ to a more detailed level of description. If the model says something important about development it may have an important role to play in guiding computational studies of learning. For this to be a realistic possibility, however, it is essential to provide a computational interpretation of the model. This is the main aim of the paper. Ideally, we would like a model that explains in detail why agents engage in redescription. The model will tell us what the computational point of redescription is and also, *exactly* how it works.

To focus ideas let us concentrate on a particular scenario. We imagine an ‘agent’ who inhabits a very simple ‘environment’. We assume that the agent’s only goal is to survive although this assumption could be relaxed without any

¹According to Boden, it may also help to explain the phenomenon of *creativity* [5].

adverse effects. The agent has certain sensors and certain effectors. In order to survive in the environment the agent must react contingently, producing specific types of behaviour in reaction to particular collections of sensory input.

In behaviourist models contingent action is typically achieved via a network of stimulus-response associations, each of which enables a particular response to be ‘triggered’ by a particular collection of sensory inputs.² The cognitivist position, on the other hand, is that the production of contingent behaviour involves the use of structured knowledge representations. The RRH is essentially a story about how such representations come into existence in humans. Our goal is to understand the computational foundations of this story.

Let us construe the agent’s sensors as ‘input variables’ and label them X_1 , X_2 , X_3 ... X_n . The sensory inputs received via the sensors can now be thought of as ‘instantiations’ of the variables. All the sensory inputs occurring at a particular time can be thought of as an ‘instantiation vector’ for the input variables. In order to achieve the goal of survival the agent must respond to particular instantiation vectors in particular ways. However, on the assumption that the sensors produce continuous values, the number of instantiation vectors will almost certainly outnumber the number of contingent behaviours by several orders of magnitude.³ Thus we can say that the agent’s task in general is to respond contingently to particular *classes* of instantiation vector.

What strategies can the agent bring to bear in achieving contingent behaviour? A convenient way of answering the question involves using statistical analysis. By dividing input classes (i.e., instantiation vector classes) up into statistical types we quickly see that certain types of class can only be processed in certain ways. Furthermore, we see that there is a particular type which can only be processed by an agent who makes use of *constructive* processes. These processes are, I will argue, essentially redescriptive in character and therefore provide a good, low-level model for the RRH part one.

3 Statistical cases

Statistical theory tells us that any data set can exhibit properties of various types. [7] A basic distinction is between type-1 (type-1) and type-2 (higher-order) properties. The type-1 (statistical) properties of a particular data set are the relative frequencies (ie. probabilities) of variable values. A type-1 property of a set of instantiation vectors might be the fact that the relative frequency with which $X_1 = 0$ is 0.8. That is to say, within the complete set of instantiation vectors, X_1 has the value zero in exactly 80% of cases. A data set with a strong type-1 property is listed below. Instantiations of the input variables are listed in columns working left to right. Thus each line shows a particular instantiation vector. Note how that the instantiation $X_2 = 1$ occurs in 100% of cases. This

²Interestingly, this view is not incompatible with recent ‘reactivist’ approaches in artificial intelligence and robotics [6].

³If the agent has 10 sensors each of which returns a range of 1000 readings, then there are more than 10^{29} instantiation vectors in all.

deviates significantly from the ‘chance’ 50% value.⁴

1	1	1	1	0
0	1	1	0	1
0	1	0	1	0
0	1	1	1	1
1	1	1	1	1
0	1	0	0	1
1	1	1	0	0
1	1	1	1	1
0	1	0	0	0
0	1	1	1	1

The type-1 (higher-order) statistics of a data set are the correlations or relationships between the variables. A type-2 statistical property of a set of instantiation vectors might be the fact that the correlation coefficient between X₁ and X₂ is 0.9. Or it might be the fact that the value of X₁ always perfectly divides the value of X₂. In the illustrative data set listed below X₁ and X₂ have a strong, negative correlation.

0.7	0.3
0.9	0.1
0.5	0.5
1.0	0.0
0.7	0.3
0.3	0.7
0.4	0.6
0.2	0.8
0.0	1.0

Finally, we have type-3 properties. These are slightly more difficult to conceptualize. They are made up of the correlations or relationships between variable *values*. In a simple case we might have a type-3 property associated with the duplication of variable values. The presence of this particular property in a set of instantiation vectors means that in any particular vector we will always see one duplicate, i.e., a value instantiating two different variables. An example of this property is illustrated in the data set below.

0.1	0.1	1.0	0.6
0.4	0.1	1.0	1.0
0.6	0.6	1.0	0.9
0.4	1.0	0.4	0.4
0.9	0.9	0.8	0.4
0.6	0.4	0.0	0.4
0.1	0.1	0.5	0.8

⁴The chance value is 50% on the assumption that the instantiation variables are binary valued.

0.9	1.0	0.3	1.0
1.0	0.6	1.0	0.4
0.9	0.4	0.6	0.4

The division of statistical properties into types 1, 2 and 3 appears to cover the full range of possibilities.⁵ The following summarizes the essential characteristics of each statistical type.

- type-1 properties are to do with **frequencies**
- type-2 properties are to do with **variable relationships**
- type-3 properties are to do with **value relationships**

4 Which strategy for which class?

Let us now return to our imaginary scenario and consider the question of what strategies the agent can use in detecting classes in the three categories. Clearly, the strategies that a particular agent can use depend on the agent's computational abilities. Thus before we can say anything about how our agent can deal with input classes we must state clearly what we assume these abilities to be. For present purposes our assumption will be that the agent is able to construct and run recursive-function structures involving just one type of function, namely the semi-linear 'squashing' function commonly used in feed-forward neural networks [9]. Formally this means that the agent is able to construct and apply lattices in which each node is a semi-linear (e.g. sigmoidal) function taking inputs from its children and giving outputs to its parents. Informally it means that our agent is able to construct and run ordinary feed-forward neural networks.

Why have we chosen this particular assumption? Disallowing sophisticated processing possibilities enables us to make contact with research in the area of 'reactive systems' [6,10] which de-emphasizes the role of sophisticated computational mechanisms. Fixing on standard feed-forward networks allows us to make contact with those aspects of connectionist research which are concerned with feed-forward networks. In particular it allows us to make use of the large body of theoretical work relating to this type of architecture.

5 Dealing with type-1 classes

The strategies that our agent can use to detect the three different input cases can now be worked out with a fair degree of detail. Type-1 classes can be dealt with most straightforwardly and, in some cases, trivially. In the simplest situation a type-1 class is defined in terms of a fixed constraint over a particular variable. For example, we might have the constraint that the value of X₁ is always 0. In this case sensor X₁ is literally a detector for the input class. Thus

⁵Contemporary statistical theory does not deal extensively with type-3 properties, cf. [8]

the agent already possesses the computational structure required to detect the class.

In a slightly less straightforward situation the agent may need to integrate two inputs. For example if the distinguishing characteristic of the class is that the sum of the values of X_2 and X_3 always exceeds some threshold, then the agent must make use of a single semi-linear function (SLF) whose weights are set such that a certain output is returned just in case the net input exceeds the relevant threshold. In general, if single variable values have a particular significance for contingent action (which they necessarily do in the type-1 case), then that significance must translate into a strong correlation between a particular variable value and the presence of a member of the input class. Positive and negative correlations of this type can generally be dealt with by a SLF configured with weights whose sign and magnitude reflect the relevant correlations.

To make this more concrete, imagine that our agent is a very primitive land-based animal. It has no visual system but negotiates its environment using simple tactile sensors. It has a sensor buffer via which it can get access to all recent sensory inputs. It has a relatively limited degree of mobility and for this reason requires behaviours which enable it to detect and move away from difficult or overgrown terrain. Consider, for example, the animal's behaviour with respect to large boulders. If the agent comes up against a boulder then contingent behaviour is necessitated that will achieve a change, and in extreme cases a reversal of direction.

Given the assumption that the agent has a series of pressure sensitive sensors arranged around its leading edge we expect that running into the boulder will create a pattern of input in which certain sensor readings are high. This situation is easily detected by a SLF which sums and then thresholds inputs from the relevant sensors. To ensure that the SLF only produces a positive output when a boulder is encountered it is only necessary to ensure that the threshold of the SLF is set appropriately high. To summarize, the usual architecture required for the detection of type-1 classes is a single-level hierarchy of SLFs; ie. a single layer of SLFs. The characteristic form of this architecture is shown in Figure 1. Not only are type-1 input classes easily dealt with from the detection point of view. They are also easily dealt with from the developmental point of view. There are many robust SLF-tuning algorithms which can be used either with or without environmental feedback (ie. positive or negative reinforcement), cf the Perceptron algorithm [11] the LMS algorithm [12], competitive learning [13] or the Kohonen net [14]. Provided a given type-1 class is evidenced in the type-1 properties of the input environment then any one of these algorithms might be used to tune the weights (coefficients) of a SLF so as to detect that class. Learning a detection architecture for a type-1 class thus involves first assembling a suitable number of SLFs which accept input from sensors and then applying one of the standard tuning algorithms.

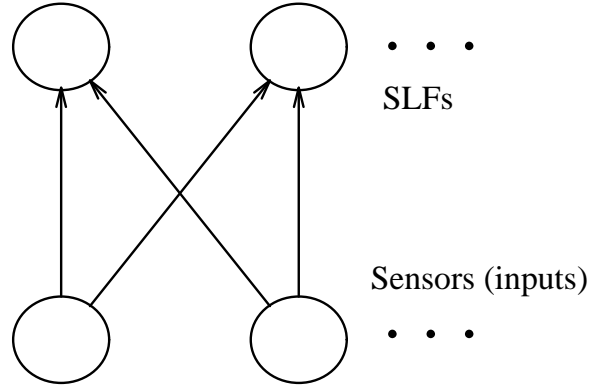


Figure 1:

6 Type-2 strategy

To deal with type-2 input classes the agent must be able to detect the presence of relationships (eg. correlations) between input variables. Typically, this *cannot* be done using a single layer of SLFs. The simplest demonstration of this is the well-known argument which shows that SLFs cannot compute the boolean relationship XOR [15].⁶ A more intuitive explanation notes that SLFs cannot be used to detect relationships because they effectively merge all inputs together and thus ‘forget’ which input is which. Where we want to compute a relationship between variables it is generally essential to retain this information. However, using small hierarchies of SLFs we *can* capture simple relationships. In particular we can capture the ‘equality’ relationships (which is simply the inverse of the XOR relationship) using a two-level hierarchy of SLFs [15] such as the one shown in Figure 2. In general, we assume that a two-level hierarchy is the least requirement for the detection of any type-2 class. Let us assume that due to mobility limitations the agent/animal has to avoid areas of long, loose grass. Thus, if moving into an area of grassland in which the average length and looseness of the grass increases, the agent must change direction. The question is: how can the agent detect the fact that it is moving into an area of this type? The simplest (type-1) strategy of testing for a high degree of pressure over the front sensors will not work in general since it cannot discriminate between short, tough grass (which can be negotiated) and long, loose grass (which cannot). Thus the agent must somehow discover the length and looseness of the grass through which it is passing.

A blade of grass making contact with a pressure sensor will trigger a hill-shaped pattern of readings. Initially there will be low readings (light pressure). These will gradually increase and then fall off as the blade is dragged over

⁶This function takes two binary inputs and returns true just in case only one of the inputs is true.

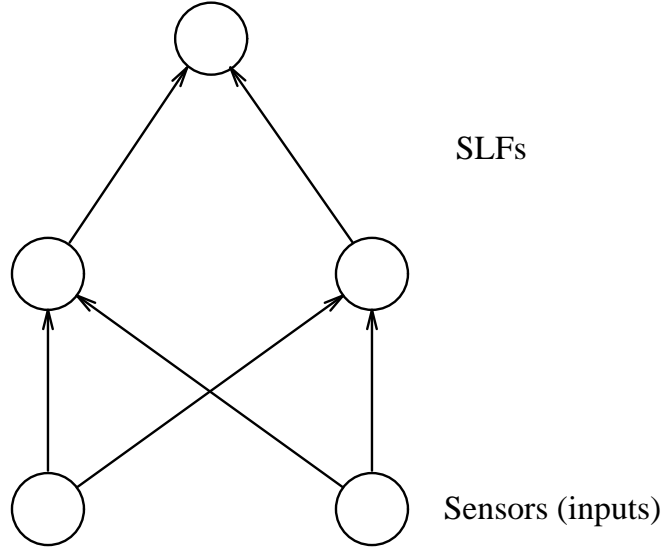


Figure 2:

and off the sensor pad. A long, loose blade will create a particular ‘signature’ of sensor readings in the buffer which differs from the signature created by a short, tough blade. Individual readings are completely meaningless taken out of context. A high reading could correspond to the presence of a negotiable but tough, short blade. A low reading could correspond to the presence of a non-negotiable but loose blade. Thus to detect the presence of a loose blade it is essential to consider the *relationships* between the readings.

Given the computational characteristics of the agent, the relationship-testing function can only be achieved using at least a two-level hierarchy of SLFs. Moreover, since the amount of time it takes for a blade to be fully pulled over a sensor pad is likely to be a large multiple of the sensor firing time, the relationship will involve readings over a large number of time periods. Thus we can expect the characteristic form of a minimal SLF architecture for the detection of this particular input class to be roughly that shown in Figure 3. For the purposes of *learning* to detect a type-2 class type-1 strategies will no longer suffice. The task of discovering the intrinsic relationships exhibited by a data set is a very difficult problem, currently the subject of a great deal of research; cf. [16, 17, 18, 17, Muggleton Inductive Logic Programming, 1992,19]. Whereas type-1 classes can typically be learned using simple gradient descent processes over a relatively small surface, type-2 classes require a much more complex search for which there is typically no reliable, problem-independent heuristic. Current methods tend to rely on the use of some form of background knowledge in order to guide and constrain the search. However, this is not relevant for present purposes since we are concerned with general rather than special-purpose strategies.

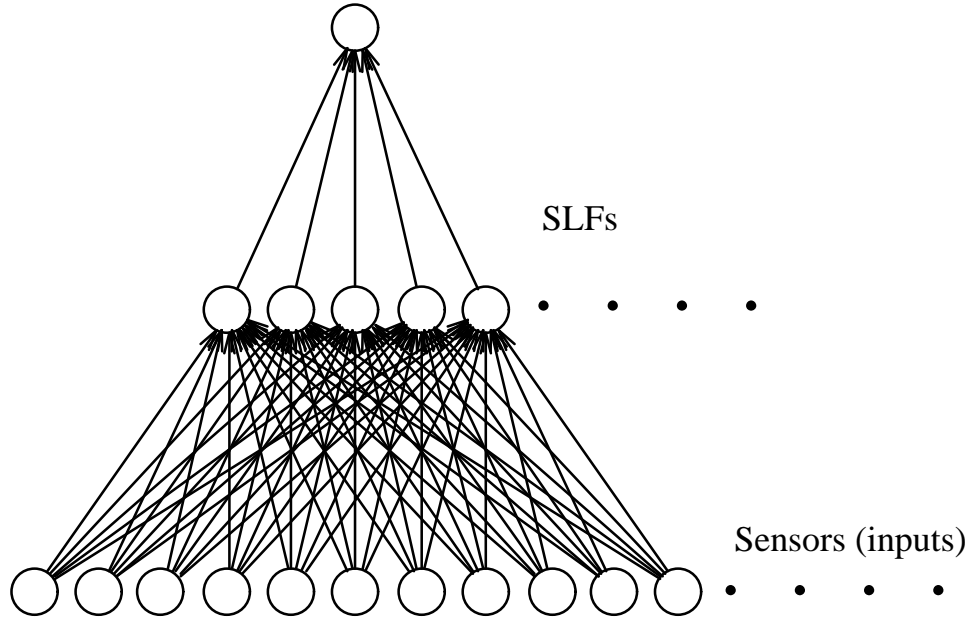


Figure 3:

The general implication is that the development (by learning) of detection abilities for type-2 classes will require substantial time resources. In some cases it seems that these requirements effectively rule out the development ‘from scratch’ of detection abilities in a realistic lifespan. However this need not necessitate the unappealing conclusion that type-2 detection abilities are impossible to acquire. However it should draw our attention to the possibility that the acquisition of type-2 detection abilities may be primarily the responsibility of evolutionary processes rather than developmental ones.

7 Strategy for type-3 classes

When we turn attention to type-3 classes the situation becomes rather problematic. Every type-3 problem has a type-2 problem contained within it. This problem is reified by ‘normalizing’ the data so as to attach to specific variables those values which form the roles of the relevant relationship. This effectively reduces the problem from type-3 to type-2. From the engineering point of view it seems likely that any feasible solution to a type-3 detection problem will decompose the problem into its two parts: the reduction (‘normalization’) problem and the type-2 problem. Certainly there are strong developmental arguments which suggest that any SLF-using agent will use this approach.

The overall search-complexity of the task of learning to detect a type-3 class

can be derived by combining the complexities of the two subproblems. However, there are two quite different ways of doing this. If we assume that a single search is executed of the complete space of possibilities then the complexity (ie. the size of the search space) is derived by taking the product of the sizes of the two spaces. If, on the other hand, we assume that the two searches are performed in serial, one after the other, then the overall complexity is simply the sum of the two search-space sizes. In all non-trivial cases, the former complexity will exceed the latter by at least an order of magnitude.⁷ Thus we can conclude that any resource-limited agent seeking to acquire an ability to detect a type-3 class will almost certainly decompose the problem into its two halves. That is to say, the agent must acquire both an ability to reduce the problem from type-3 to type-2 and the ability to solve the type-2 problem.

The architectural implications of this can be illustrated by a further extension of the grass-avoidance example. Consider the situation in which the animal/agent collides obliquely with some fairly flexible object — the root of a plant, say. Rather than invoking its usual grass-avoiding behaviour the animal should simply swerve to the correct side so as to move around the obstacle. To achieve this effect the animal needs to detect that particular signature of sensor inputs which is created when a fairly robust object collides obliquely with, and is then dragged along the frontal sensors. The point at which the object strikes the frontal sensors determines the ‘location’ of the signature in sensor space. The object might strike the rightmost sensor first. Or a sensor slightly to the left of this. Or a sensor still further to the left, and so on. However, in all these cases the object is striking from the right. Thus a swerve to the left is called for.

In detecting this sort of situation the agent must recognize a particular signature of sensor readings. However, this signature is not localized or grounded in particular sensors. It might manifest itself across the sensor readings in a number of ways. The input class requiring detection is thus based on ‘floating relationships’ and is therefore a type-3 class. Given the arguments put forward relating to the necessity of decomposing any type-3 problem into a reduction component and a type-2 component, we can assume that the swerving behaviour of the agent must involve at least two interconnected, two-level hierarchies of SLFs. The first level will undertake the reduction task, i.e., it will normalize the sensor readings in a fixed frame. The second level will undertake the type-2 task. The characteristic form of the network will thus be broadly that shown in Figure 4.

8 Discussion

We have now worked through an analysis of the three types of input class and looked at how a particular type of agent (restricted to using feed-forward networks of SLFs) could reliably detect classes of each type. One of the surprising

⁷For example, if both spaces have 1000 points, then the ‘serial’ search space has 2000 points and the non-serial space has one million.

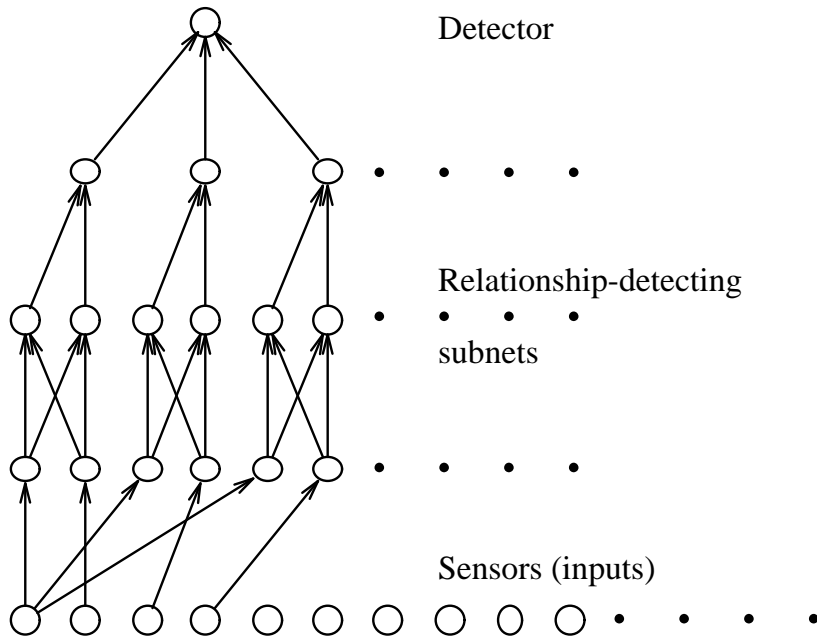


Figure 4:

features of the analysis is the fact that, in the scenario we have considered, even very trivial behaviours seem to necessitate quite substantial amounts of architecture. The network envisaged in figure-above is only partially sketched in. However, one can see that it is already rather complex. This is surprising when one considers that its intended purpose is merely the recognition of situations in which swerving is appropriate.

The rapid growth in the required architectures is partly due to the fact that we have restricted the agent to the use of constructions on a fairly simple primitive function, namely the SLF. However, it is not at all clear that much architecture would have been ‘saved’ by using a different function basis. Had we used, for example, a more sophisticated type of function, e.g., one which tests directly for a type-2 property (ie. a relationship), then we might have been able to solve the type-2 detection problems more easily but we would have had difficulty in deriving a solution to type-1 problems. Had we assumed that the agent had access to a more complex, generic computing device (eg. a Turing machine) then clearly we would have buried the difficulty of the problem inside our assumptions. My guess, then, is that any assumption regarding the computational limitations of the agent which (1) enables solutions to all types of detection problem to be envisaged and which (2) does not simply finesse the whole problem will lead to results of the same basic character as the ones derived above.

Having said that, it is important to emphasize that the architectural growth that we witness in the examples above merely scratches the surface of the growth that we would expect in a *realistic* situation. We have explored the ramifications of input-class detection through one level of abstraction. But agents operating in realistic environments can be expected to be faced with detection problems through many levels.

In the envisaged scenario the basic objects of our agent’s sensory world were the input variable instantiations. However, solving the envisaged type-2 detection task produces a ‘blade detector’ in the agent. This detector effectively signals the presence of blades in the environment. Thus ‘blade’ became an object in the agent’s sensory world. Ditto for ‘long, loose blade’. Quite feasibly the agent might face the problem of detecting a type-2 class involving relationships between such derived objects. For example, if the agent also has detectors for a range of heavy objects then it could form a detector for the class of ‘grass-flattenings’ (situations in which a heavy object flattens a patch of grass). This would be a type-2 class with respect to the set of objects which play roles in the grass-flattening relationship.

Once the agent has acquired detectors for ‘objects’ such as glass-flattening situations, further possibilities are opened up. For example, the agent might need to form a detector for mobile objects which are able to perform glass-flattening (ie. the class of all glass-flattening organisms). This class would be type-1 with respect to ‘glass-flattenings’. Moving on up, the agent might form a detector for ‘stable situation’. This might be a class of situations in which glass-flattening organisms are living peacefully with some other types of organism. This latter class would be type-2 with respect to objects such as ‘glass-flattening organism’.

This upwards cascade of objects can be continued still further. For example the agent might form a detector for ‘settled societies’ which are societies in which subgroups tend to have the ‘stable situation’ attribute. This new class would be type-1 with respect to the latter type of object. Once a detector for the ‘settled society’ is in place then we have the possibility of detectors for still higher level detectors (‘international destabilizing influence’ etc.) And so on.⁸

Starting with any set of basic objects, we have the possibility of first, type-2 and type-3 classes formed over those objects. Each such class potentially corresponds to a new object. Thus when we take the entire set of classes we effectively have a new level of objects in terms of which we can have a further layer of type-1, type-2 and type-3 classes. The growth pattern here is essentially an upright tree which branches into three at each level (i.e., a ternary tree); cf. Figure 5. Each node in the tree corresponds to a set of objects. The objects

⁸At each level, there is a type-1, type-2 and type-3 leg-up to the level above. However, there are reasons for thinking that type-3 leg-ups will primarily be found in the initial stages of the upwards cascade. A type-3 class is to do with floating relationships. These occur when we have a set of basic objects which effectively forms a framework in which higher level phenomena can manifest themselves in a range of ways. The most natural type-3 scenario is one in which the agent has an array of sensors (e.g. something like a visual system) upon which environmental phenomena are *projected*. Variations in projection cause variations of location and these in turn produce the floating relationships that give us a type-3 class.

at the root node are primitive. In the scenario we have considered they are basic sensory inputs. Objects at other nodes are non-primitive. Each non-primitive object picks out something in the environment. Thus each non-root node effectively provides a representation space. Other, higher-level aspects of the world can be represented in terms of the objects in question, via immediate first, type-2 or type-3 classes or by higher levels of such classes.

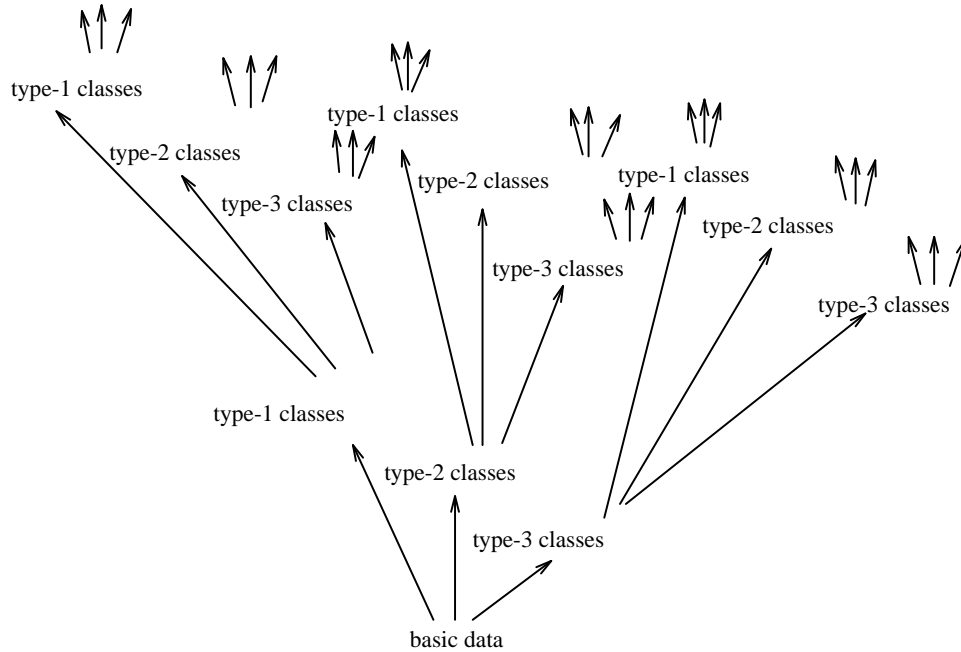


Figure 5:

9 Comparison with Representational Redescription

An agent that gradually explores the upwards cascade of classes and objects is effectively forming a hierarchical structure of representation spaces. Representation spaces are constructed one upon the other in a process which produces increasingly abstract sensing of the environment. Does this ‘representational abstraction’ process (RA as I will call it below) equate to the ‘representation redescription’ (RR) of Clark and Karmiloff-Smith? There are certainly points of correspondence. At the most general level we have a broad resemblance between two processes whose aim is the derivation of higher-level representations from lower-level ones. However, although RR is specifically stated to be conservative there is a clear suggestion that higher-level representations in RR somehow

improve on and supercede the lower-level ones. This is certainly not the case with RA. In RA the lower-level representations pick out objects which have a valid and independent existence. Thus higher-level representations in no sense supercede the lower level ones.

A further minor difference is related to the fact that the RRH emphasizes the way in which RR converts procedural representations into declarative ones. This is not an obvious feature of RA. However, since procedural transitions may constitute ‘relationships in action’ one might hypothesize that RA would lead to a RR-style procedural-to-declarative redescription just in case the relationships (type-2 and type-3) underpinning the objects at the initial level of description are of the procedural type.

There is an interesting correspondence in the RRH’s claim that humans ‘constantly go beyond goal attainment to build theories that are not actually necessary for the successful problem-solving procedures per se.’ [Clark and Karmiloff-Smith, Forthcoming]. Certainly, the advantages of spontaneously exploring higher levels of description stand out clearly in the RA process. Each new level in the process puts the agent in touch with a whole range of ‘new’ environmental phenomena. Unless the agent spontaneously explores higher levels there are environmental phenomena that may never be discovered. There may be evolutionary pressures pushing the agent towards the discovery of significant classes which are type-1, type-2 and type-3 with respect to currently known objects. But it is difficult to believe that there could be any evolutionary pressure on the agent to discover classes which are several levels of abstraction above a currently represented level.⁹

However, alongside the correspondences there are various differences to be noted. For example there seems to be no obvious echo in RA of the idea (important in the RRH) that redescription cannot take place until ‘behavioural mastery’ has been achieved at the existing level.¹⁰ Lacking also is any rejoinder to the idea that ‘the lowest level “languages” or procedures will be broadly *connectionist* while the higher level ones will be broadly *symbolic*’. [Clark and Karmiloff-Smith, Forthcoming].¹¹

Overall, we can say that there are interesting and suggestive correspondences between the RA process sketched out above and RR. Deciding whether RA forms a good, computational level model for RR will require further fleshing out of both processes. However, if it should turn out that RA is a good model for RR, then RRH part two — which says that RR is the distinctive mark of genuine thinkers — is almost certainly suspect. As the ‘grass’ example clearly shows, RA is a process that we should expect even very primitive organisms — maybe even *sea-slugs* — to engage in. Thus either we will have to face the prospect of

⁹The effect would be like the influence that a pin exerts on an arbitrary piece of straw in the proverbial haystack.

¹⁰Unless we assume that behavioural mastery simply corresponds to having all the relevant objects represented.

¹¹If anything, the fact (noted above) that type-3 classes are likely to be more common in the lower levels of the hierarchy seems to undermine this idea. Type-3 classes are particularly hard to detect via purely ‘connectionist’ regimes.

allowing very primitive organisms to be ‘genuine thinkers’. Or we will have to face the fact that genuine thinkers are not distinctively marked by RR.

References

- [1] Karmiloff-Smith, A. (Forthcoming). In D.M. Peterson (Ed.), *Internal Representations and External Notations: A Developmental Perspective*. Intellect.
- [2] Karmiloff-Smith, A. (1979). *A Functional Approach to Child Language*. London: Cambridge University Press.
- [3] Karmiloff-Smith, A. (1979). Micro- and macro-developmental changes in language acquisition and other representational systems. *Cognitive Science*, 3, No. 2 (pp. 81-118).
- [4] Karmiloff-Smith, A. (1990). Constraints on representational change: evidence from children’s drawing. *Cognition*, 34 (pp. 57-83).
- [5] Boden, M. (1990). *The Creative Mind: Myths and Mechanisms*. London: Weidenfeld and Nicolson.
- [6] Brooks, R. (1991). Intelligence without reason. *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence* (pp. 569-595). San Mateo, California: Morgan Kaufman.
- [7] Hinton, G. and Sejnowski, T. (1986). Learning and relearning in boltzmann machines. In D. Rumelhart, J. McClelland and the PDP Research Group (Eds.), *Parallel Distributed Processing: Explorations in the Microstructures of Cognition. Vols I and II* (pp. 282-317). Cambridge, Mass.: MIT Press.
- [8] Krzanowski, W. (1988). *Principles of Multivariate Analysis: A User’s Perspective*. Oxford Statistical Science Series - 3, Oxford: Clarenddon Press.
- [9] Sejnowski, T. and Rosenberg, C. (1987). Parallel networks that learn to pronounce english text. *Complex Systems*, 1 (pp. 145-68).
- [10] Brooks, R. (1991). Intelligence without representation. *Artificial Intelligence*, 47 (pp. 139-159).
- [11] Rosenblatt, F. (1962). *Principles of Neurodynamics*. New York: Spartan Books.
- [12] Hinton, G. (1989). Connectionist learning procedures. *Artificial Intelligence*, 40 (pp. 185-234).

- [13] Rumelhart, D. and Zipser, D. (1986). Feature discovery by competitive learning. In D. Rumelhart, J. McClelland and the PDP Research Group (Eds.), *Parallel Distributed Processing: Explorations in the Microstructures of Cognition. Vol I* (pp. 151-193). Cambridge, Mass.: MIT Press.
- [14] Kohonen, T. (1984). *Self-organization and Associative Memory*. Berlin: Springer-Verlag.
- [15] Minsky, M. and Papert, S. (1988). *Perceptrons: An Introduction to Computational Geometry* (expanded edn). Cambridge, Mass.: MIT Press.
- [16] Feng, C. and Muggleton, S. (1992). Towards inductive generalization in higher-order logic. In D. Sleeman and P. Edwards (Eds.), *Proceedings of the Ninth International Workshop on Machine Learning (ML92)* (pp. 154-162). San Mateo, California: Morgan Kaufmann.
- [17] Muggleton, S., Srinivasan, A. and Bain, M. (1992). Compression, significance, and accuracy. In D. Sleeman and P. Edwards (Eds.), *Proceedings of the Ninth International Workshop on Machine Learning (ML92)* (pp. 338-347). San Mateo, California: Morgan Kaufmann.
- [18] Kira, K. and Rendell, L. (1992). A practical approach to feature selection. In D. Sleeman and P. Edwards (Eds.), *Proceedings of the Ninth International Workshop on Machine Learning (ML92)* (pp. 249-256). San Mateo, California: Morgan Kaufmann.
- [19] Fahlman, S. and Lebiere, C. (1990). The cascade-correlation learning architecture. In D.S. Touretzky (Ed.), *Advances in Neural Information Processing Systems 2* (pp. 524-532.). Morgan Kaufmann Publishers, Los Altos CA.