# How Much is Enough? A Connectionist Perspective on the Representation Debate

Chris Thornton
Cognitive and Computing Sciences
University of Sussex
Brighton BN1 9QN
Email: Chris.Thornton@cogs.susx.ac.uk
Tel: (44)273 606755 x 3239

May 19, 1994

**Abstract**

The paper looks at the tension between the classical assumption that representation is vital for effective cognition and the relatively recent 're-activist' movement which takes a contrary view. A statistical analysis of cognitive tasks is developed and used to support the argument that purely reactivist approaches cannot hope to deal with anything but the most primitive of domains.

## 1 Introduction

Time was, the position of *representation* in the cognitivist universe seemed unassailable. Most people working in the paradigm appeared fully convinced that good representation was one of the keys (maybe *the* key) to successful cognition. AI researchers threw themselves enthusiastically into the task of thinking up pithy epigrams which neatly encapsulated the essence of this faith.[1] But now the old, simple trust in the absolute primacy of representation is coming under attack.

[1]A recent example appears in [1]: 'there are three important aspects to any AI system: representation, representation, and representation.'

The name that is often associated with this new attack is that of Rodney Brooks. As a result of his relatively recent work with mobile robots, Brooks has come to feel that the use of abstract, representational formalisms can be, and often is, counter-productive. He stresses that complex, intelligent behaviour can be more easily and efficaciously produced by systems which have simple 'reactive' behaviours with regard to environmental events. He has lambasted the use of over-simplified micro-worlds in AI, noting that they may merely serve to finesse the difficult problems that real-world, cognitive systems must confront. In his own work he has concentrated efforts on the development of *situated* or *embedded* 'Creatures' — simple, insect-like robots which rely primarily on reactive behaviour to negotiate and interact with genuine (ie. non-simulated), physical environments [2, 3, 4, 5].

Of course, the line Brooks is taking is not entirely new. The key idea that 'the world is its own best model' [4] can be traced back to Gibson's work on vision (which stressed the ways in which the world provides information about itself) while the notion that 'Intelligence is determined by the dynamics of interaction with the world' [4] is perhaps best known to cognitivists through the writings of Simon and his well-known parable of 'the ant on the beach' [6]. But even if the reactivists position is not entirely novel, it is certainly not lacking in force as Brooks' recent publications amply testify.

In the context of this developing debate, connectionism finds itself 'pig-in-the-middle'. Clearly, the rise of the connectionist paradigm has involved a rejection of the complex declarative and procedural formalisms that earlier AI research so favoured.[2] In this sense, connectionism might be seen as reinforcing and confirming the reactivists' position. On the other hand, connectionist researchers continue to volubly stress the central importance of representation in all non-trivial computational tasks (cf. [8, 9]). In fact, much state-of-the-art work on constructive algorithms, eg. [10], is directly concerned with methods via which useful representational structures can be learned. Thus, the field might be seen as exerting pressure against the reactivist thesis.

Being the debate's pig-in-the-middle, connectionism might reasonably be expected to provide a reconciliation of the relatively extreme positions of classical AI on the one hand and Brooksian reactivism on the other. But if it was to attempt to do so, it would have a testing time. The problem, of course, is that virtually everything we 'know' about representation boils down to anecdote, gut-feeling and heresay. Though AI and computer science have both put in long-service in the support and advancement of the 'principle of good representation', neither discipline has been able to provide any sort of *theory* that might give the principle a rational basis.

[2]In fact hybrid approaches are more eclectic and retain the emphasis on structured representation [7].

When one considers the central role that representation has occupied in the thoughts and practices of AI researchers, the total absence of any consensual theory of representation is perplexing. Brooks has certainly done the field a great service in drawing attention to it. By disputing the necessity of representation he has put the onus on classical AI to *demonstrate* that necessity. And the sad fact is, that despite years of work, the demonstration is still very hard to come by.

The remainder of the paper attempts to remedy this deficiency to some degree by putting together a statistical analysis of a series of simple cognitive tasks. Under reasonable assumptions this analysis demonstrates that abstract representational structure is essential for all but the most trivial of cognitive tasks. It thus provides an argument which suggests that the extreme reactivist position ('representation is irrelevant') is almost certainly wrong.

## 2 The simplest description of a cognitive task

At the most basic level, a cognitive task is simply a mapping between certain 'inputs' and certain 'outputs'. Thus finding an explanation for a cognitive task involves providing a viable computational description of how the mapping can be implemented.[3] In terms of the-Figure 1, it involves substituting an explanation for the question mark in the box.

Different models make different assumptions about the form taken by the inputs and the outputs. But perhaps the most common and general assumption is that both the inputs and the outputs take the form of 'vectors' of values.[4] This gives us a basic picture which states that a cognitive task is essentially a mapping from input vectors to output vectors.

Let us look at a simple example. Imagine that we have a task which involves a mapping whose initial pairs are as follows.

```
0 0 1 1 0   -->  1
0 0 0 0 1   -->  0
1 0 1 0 1   -->  1
0 1 1 1 0   -->  1
1 1 1 1 0   -->  1
0 0 1 0 0   -->  1
1 1 1 0 1   -->  1
```

[3]The question of how a particular mapping can be learned is usually even more difficult to answer [11].

[4]No generality is lost by making this assumption since more complex representational forms (eg. structured forms) can always be converted to/from 'flat', vector form.

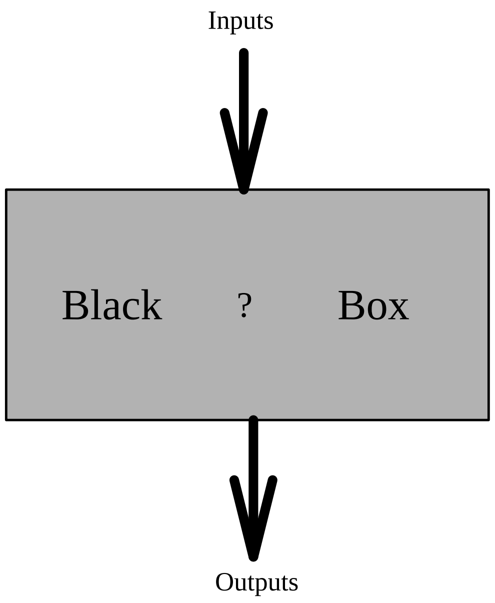

Figure 1:

```
0 1 0 0 1   -->  0
0 1 0 0 0   -->  0
0 1 1 0 1   -->  1
```

Each line here represents a particular entry in the mapping. The sequence of values before the arrow represents the input vector. The sequence after the arrow (a single digit in this case) represents the output vector. We might imagine that the input vector values represents simple sensory inputs to some cognitive agent (eg. pressure or absence of pressure at some particular site) and that the output value represents a motor-action signal of some sort (eg. move-hand-left).

A classical model of this task might involve the use of complex, representational constructs (eg. ISA hierarchies, frames, schemas, declarative and procedural constructs). A reactivist model might attempt to capture the mapping in a much more direct way, eg. by making the motor-action output directly dependent on the presence of particular input values, eg. by 'hard-wiring' the response of the agent.

## 3 The statistical analysis of mappings

Rather than ask which of these two models is right we will attempt to determine what sort of representational structure is necessary for this particular task. We will base our analysis on the assumption (commonplace in connectionism) that the role of a representation is to capture the statistics of the underlying task/mapping. Thus our aim will be to analyze the statistics of the mapping and to decide what is required for these statistical properties to be adequately captured. The underlying idea here is that the analysis of the statistics of a mapping tells us something about the structure of an optimal representation of that mapping.

The important statistical properties of a mapping are derived from the relative frequencies with which certain output values are associated with certain input values. The relative frequencies can be classified by 'order'. The *first-order statistics*[5] of a mapping are derived by observing the relative frequencies with which particular input values are associated (in the mapping) with particular output values. In the case of the mapping shown above, an initial input value of 0 is associated with an output value of 1 in exactly four of the ten cases. Thus the 'observed conditional probability' that the output is a 1 given that the initial input value is 0 is exactly 0.4. We can write this as follows:

```
P(1 | <0>@1 ) = 0.4
```

We use the normal notation for a conditional probability but we specify the condition in terms of a sequence beginning at a certain position in the input vector. As an illustration of the notation, the string

```
<0>@1
```

denotes a subsequence which begins at position 1 of the input vector and consists of the single digit 0. The string

```
<3 4 5>@2
```

denotes a subsequence beginning at position 2 of the input vector which consists of the digits 3, 4 and 5 (in that order).

The *second-order statistics* of a mapping are derived by observing the relative frequencies with which particular 2-tuples of input values are associated with

[5]We use the term 'statistics' interchangeably with 'statistical properties.'

particular outputs. In general, the $n$th-order statistics of a mapping are derived by observing the relative frequencies with which n-tuples of input values are associated with particular output values.

## 4 Capturing first-order statistics

Detailed examination of the mapping shown above reveals that the output value is directly dependent on the value of the third input value. In fact the output value is, in all the examples shown, identical to the third input value. The first-order statistics are thus very informative here: they yield an absolute certainty regarding the output value. There are just two cases to consider and in both, we have complete certainty as to what the output value should be. The situation is summarized in terms of the following probabilities:

```
P(1 | <1>@3 ) = 1
P(0 | <0>@3 ) = 1
```

Capturing this very simple statistical structure is computationally trivial. If we are thinking in terms of a network substrate with activation-carrying connections, then the solution is simply a hard-wired link which connects the stimulus corresponding to input value three with the main output. 'Captures' of this statistical structure in terms of other substrates are just as trivial. In effect all we need to do to capture this structure is to arrange for a 'copycat' response to input value three.

## 5 Capturing higher-order statistics

Unfortunately, it is not always this easy to capture the statistics of a particular mapping. Consider the following sample pairs from a mapping which we call 'centre-parity'.

```
1 1 0 1 0 1 1 1 0 1 0 0  --> 0
1 1 1 0 1 1 1 1 1 1 1 0  --> 1
0 0 1 1 0 1 0 0 1 0 1 0  --> 0
0 0 0 1 1 1 0 0 0 0 1 0  --> 1
1 1 0 0 1 0 1 0 0 1 0 0  --> 1
0 1 1 1 0 1 1 1 0 0 0 0  --> 0
0 0 1 0 1 1 0 0 0 0 1 1  --> 1
1 1 0 1 1 0 1 1 0 1 1 1  --> 0
```

```
1 0 1 1 0 1 1 0 1 1 0 1  --> 1
0 1 1 0 0 0 1 1 1 0 0 0  --> 1
```

The first-order statistics of the (total) mapping turn out to be rather uninformative. In particular there are no certainties derivable from the first-order observations. In fact, we find no certainties whatsoever at any level of analysis until we reach fourth-order. At this level, the mapping can be captured in terms of a small number of certainties, which relate to the parity of the second group of four input values. The certainties are:

```
P(1 | <0 0 0 0>@5 ) = 1
P(1 | <1 1 0 0>@5 ) = 1
P(1 | <1 0 1 0>@5 ) = 1
P(1 | <1 0 0 1>@5 ) = 1
P(1 | <0 1 1 0>@5 ) = 1
P(1 | <0 1 0 1>@5 ) = 1
P(1 | <0 0 1 1>@5 ) = 1
P(1 | <1 1 1 1>@5 ) = 1
P(0 | any-other-case ) = 1
```

These probabilities tell us that the 'rule' underlying the mapping is that the output value is 1 provided that the four input values starting at position 5 contain an even number of 1s. What representational implications does this statistical structure have?

Note that any agent able to perform this task (ie. implement the mapping) must be able to detect the existence of parity among the second group of four inputs. Depending on the computational substrate used, the detection of this property might be effected in different ways. However, there is nothing about the statistical structure which forces us to assume that explicit representational structures will be involved. An agent could implement the mapping solely by virtue of having hard-wired circuitry (eg. a network of AND and OR gates) which enabled it to respond contingently (and in the same way) whenever the four-bit input value sequence starting at position 5 contains zero, two or four 1s.

In case it is not obvious, it should be stressed at this point, that in the mappings (tasks) we have considered so far we have had no reason to assume the need for any kind of explicit, abstract representational structure. In all cases, a simple reactivist approach would have been quite sufficient.

## 6 Capturing covert statistics

We now turn attention to tasks in which there is 'covert' statistical structure in addition to the overt structure. The most easily described cases of this arise when the task involves detecting the presence of simple patterns on a grid of sensors (eg. a retina). Consider the input/output pairs shown in the-Figure 2. (Try to work out the input/output rule before reading on.)

```
4  6  2  6  4  1  2  1    -->  1
7  6  1  9  1  2  5  1    -->  0
7  2  3  6  1  8  8  5    -->  0
1  5  9  5  1  2  9  2    -->  1
7  2  1  2  7  6  1  6    -->  1
8  6  9  7  4  7  3  4    -->  0
4  3  1  3  4  6  1  6    -->  1
4  7  8  7  4  8  8  8    -->  1
5  5  9  3  1  1  3  9    -->  0
7  1  4  1  5  4  5  4    -->  0
7  1  1  1  7  3  1  3    -->  1
3  4  3  4  3  6  3  6    -->  1
4  3  7  8  5  4  2  4    -->  0
1  7  2  4  7  7  9  5    -->  0
3  6  6  4  9  5  1  7    -->  0
```

These pairs are taken from a mapping whose rule states that the output value is 1 only if the input values describe a rectangular pattern when construed as a sequence of four, 2-dimensional, cartesian coordinates.

So much for the rule; what of the statistics? If we continue to use the method we have employed above then we should try to write down a set of unit probabilities (ie. certainties) which mutually exhaust all the relevant input possibilities. The initial few cases might be taken from the 'positive' examples shown. We would write these down as follows.

```
P (1 | <4  6  2  6  4  1  2  1>@1 ) = 1
P (1 | <1  5  9  5  1  2  9  2>@1 ) = 1
P (1 | <7  2  1  2  7  6  1  6>@1 ) = 1
. . .
```

But, of course, if we continue this way we will end up writing down *all* the positive examples in the mapping. Given the assumption that coordinate values range from 1 to 9, the total number of input vectors is 9^9 (387420489). The number of input cases satisfying the rule is a substantial proportion of this (certainly well over a million) so we can see that our list of probabilities will be exceedingly long.

The problem here is that the statistical structure of the mapping is 'covert' rather than 'overt'. The structure has to do with abstract properties and relationships and not with the presence of particular, explicit values. Thus if we want to represent the statistical structure of the mapping we should do so with reference to the relevant abstract properties and relationships. A convenient way of doing this involves making use of an imaginary predicate called 'rectangle' which takes eight inputs and is satisfied only if the inputs satisfy the rectangle rule given above. This enables us to specify the statistics in terms of two unit-probabilities, the first of which simply has as its condition an application of the rectangle predicate to the eight input values starting at position 1; ie.

```
P (1 | rectangle@1 )     = 1
P (0 | any-other-case ) = 1
```

Using the rectangle predicate we are able to express the statistical structure of the mapping in terms of just two unit probabilities rather than the million or so that it would have taken with our usual method. This dramatic saving tells us something important about the structure of any implementation of this mapping. In particular, it tells us that, assuming limited resources, an agent will never be able to capture this mapping using the simple lookup methods described above. The number of cases that need to be considered in even this, relatively simple, task is absurdly large. Thus realistic cognitive agents could not possibly capture this mapping in terms of hard-wired reactions to explicit input patterns. To put it rather bluntly, any realistic agent *must compute the rectangle predicate*, ie. it must be capable of recognizing the abstract relationships which underpin the property of 'rectangleness'.

## 7 The need for structured representations

We have shown that implementing tasks in which output behaviour is related to relationships among the input values necessarily involves the need to computationally detect those relationships. Thus, cognitive agents who require to be sensitive to relationships between sensory inputs must 'represent' those relationships (at least in the sense that they must be able to compute them). However, showing that agents will typically represent abstract relationships does not show that they will typically use *structured* representations. To show this we need to explore the implications of agents having limited computational resources.

Consider the situation in which a particular agent needs to implement the 'rectangles' mapping but does not have the computational resources to directly compute the 'rectangle' predicate. This limitation might result from any number of causes. It may be that the agent cannot simultaneously carry out different types of test as is essential in this predicate. It may be that the agent cannot deal with anything other than binary-valued inputs. Or it might just be that the agent has a limited 'fan-in', ie. cannot compute any function which takes a large number of inputs.

One might suppose that the limitations facing this agent mean that it necessarily cannot implement the given mapping (given the assumption that it is also incapable of implementing a lookup table containing over a million elements). But this is certainly *not* the case. The agent may well be able to construct a composition of its basic computational resources which has the required power. For example, let us assume that the agent is limited to computing a 2-input equality test (`eq`) and 2-input, boolean AND. The agent can construct a composition of these predicates which will satisfactorily compute 'rectangle'. The composition effectively takes the AND of four coordinate-equality tests. For illustration, we restate the probability analysis in terms of this specific computational construction:

```
P(1 | AND ( eq@1,5 eq@2,4 eq@3,7 eq@6,8 ) ) = 1
P(0 | any-other-case ) = 1
```

Here, the string 'eq@1,5' denotes an application of the equality tester to the first and fifth input values, 'eq@2,4' denotes the application of the equality test to the second and fourth input values, and so on. Note that the AND test is applied to the results of the four equality tests. The general point here is that with finite and limited computational resources, a cognitive agent must resort to computational constructions for the purposes of computing relevant relationships and properties. These constructions necessarily implement representations of properties at various levels of abstraction and thus constitute what are, in effect, structured representations.

## 8 The feature-detector rule

In the previous example, the computationally limited agent we envisaged necessarily uses a computational construction to fulfil the given task. However, viewed as a tree, this structure is quite shallow (it is of depth 2). Can we make any statements about the *degree* of representational structure that will be necessitated in general?

There are two points to be made. First of all we should note that in all cases where the statistics of the task mapping are based on relationships between input values, it is likely that an agent implementing that task will have to compute the relationship rather than capture it in terms of hard-wired reactions to explicit input patterns. Lookup tables (which avoid computation) can certainly be used where the total number of possibilities is small. But where we assume (as we almost certainly should do) that inputs may take on any one of a range of values, the number of combinatorial possibilities is likely to be very large.

The second point to note is that the situation in which the agent 'just happens' to have access to a computational primitive which computes a significant relationship of the relevant task domain is unlikely to be very common. In fact, assuming that cognitive agents will have special detection hardware for all types of abstract phenomena is tantamount to believing in the existence 'grandmother cells' — ie. brain cells whose sole role is to signal the presence (eg. in the visual field) of a particular individual.

The first point leads directly to the notion that detecting a particular relationship in a given domain requires a special-purpose computation. This leads us to the feature-detector rule which is stated as follows.

**The Feature-Detector Rule**
An agent dealing effectively with domain D must have a
distinct feature-detector (ie. a distinct computational
mechanism or procedure) for each significant relationship in D.

Given the assumption that a computational procedure for detecting a particular relationship counts as an abstract representation of that relationships, the feature-detector rule implies that an agent's representational complexity will be roughly proportional to the number of significant relationships of the relevant domain, ie. the number of relationships that must be taken into account in order to successfully 'negotiate' within that domain.

## 9 Discussion

The paper has shown how statistical analysis of simple cognitive tasks leads us into a fairly ineluctable series of deductions. When we consider simple tasks whose overt, low-order statistics are very informative, there is no difficulty in envisaging how reactivist implementations might succeed very well. However, as soon as we begin to consider tasks with high-order and/or covert statistics we are forced to see that a pure reactivist approach seems to necessitate the use of (the equivalent of) astronomically large lookup tables. A far more plausible assumption is that agents will compute abstract relationships so as to preempt the need to explicitly represent millions of special cases.

Moreover, even a very brief consideration of the ramifications (to the agent) of computational-resource limitations suggests that agents will tend to use computational *constructions* for the detection of abstract properties of the world (eg. relationships). This, in turn, leads to the feature-detector rule. From this rule we infer that agents relying on purely reactivist behaviour will not be able detect anything other than basic, concrete properties of the world and thus will not be able to negotiate anything but the most primitive of environments.

But how can we account for the experimental success of Brooks' approach? In particular, how can we account for the fact that reactivist robots have been built which successfully negotiate simple worlds? The answer seems to be that Brooks' has (to date) confined his efforts to situations in which the exploitation of low-order statistical relationships is sufficient for the accomplishment of purposeful behaviour. He has described the way in which one of his robots explores the office environment at MIT 'searching' for soda cans. On finding a can it removes it [5]. Brooks characterizes the reflexes involved as follows.

> The hand had a grasp reflex that operated whenever something broke an infrared beam between the fingers. When the arm located a soda can with its local sensors it simply drove the hand so that the two fingers lined up on either side of the can. The hand then independently grasped the can. [4]

It is fairly clear from this and other extracts that Brooks' robots do not attempt to take any cognisance of *relationships* in the domain that inhabit. It is perhaps, quite remarkable, how much apparently purposeful behaviour Brooks has managed to achieve in agents which have no mechanism for understanding relationships. But it would be dangerous to infer that relationships are therefore insignificant for the purposes of intelligent behaviour.

Consider a slight variant of Brooks' can-retrieving task. Imagine that we wish to have a robot which takes raincoats draped carelessly over the backs of chairs and hangs them up. If we take it as read that the robot will have the sort of insect-like sensing capabilities of Brooks' early robots then we might assume that the robot will detect the presence of a raincoat by, perhaps, driving under the relevant chair and feeling the raincoat drag over its upper sensors. The robot would have to make a sensible decision about which way to drag the coat of the chair: trying to pull on the highest side might overturn the chair. But to make this decision the robot will have to detect relationships between the two drag patterns: the drag pattern created when the robot drove under the forwards edge of the raincoat and the drag pattern created when the robot drove under the backwards edge. In fact to distinguish a raincoat drag pattern at all it will be necessary to detect relationships between sequences of sensor readings.

Suffice it to say, it is not hard to think of Brooksian robotic tasks which would appear to *necessitate* the ability to detect and measure relationships. Taking into account our previous conclusions the general implication seems to be that Brooks has committed the traditional AI error of making the hard problem 'go away' by concentrating on an ultra-simple domain.

The general aim of our argument has been to suggest that the 'pure' reactivist view, which suggests that abstract representation can be largely dispensed with, is utterly wrong.[6] There is, unfortunately, a great deal of progress still to be made, before this point can be put on truly firm, theoretical footing.

## References

[1] Anderson, J., Speohr, K. and Bennett, D. (1991). A study in numerical perversity: teaching arithmetic to a neural network. Technical Report 91-3, Department of Cognitive and Linguistic Systems, Brown University, Providence, RI 02912.

[2] Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation, RA-2*, No. 1 (pp. 14-23).

[3] Brooks, R. (1986). Achieving artificial intelligence through building robots. MIT A.I. Memo 899, Masachusetts Institute of Technology.

[4] Brooks, R. (1991). Intelligence without reason. *Proceedings of the Twelth International Joint Conference on Artificial Intelligence* (pp. 569-595). San Mateo, California: Morgan Kaufman.

[5] Brooks, R. (1991). Intelligence without representation. *Artificial Intelligence, 47* (pp. 139-159).

[6] Simon, H. (1969). *The Sciences of the Artificial.* Cambridge, Mass.: MIT Press.

[7] Torrance, S. and Thornton, C. (Eds.) (1991). Special issue on hybrid models. *AISB Quarterly*, No. 78, AISB society.

[6] There is a serious question concerning the extent to which Brooks actually holds this view. Sometimes he seems to take the extreme line ('we believe representations are not necessary and appear only in the eye or mind of the beholder [5, p. 154]). Other times he appears much more moderate and in-line with the current AI orthodoxy ('... abstraction is the essence of intelligence and the hard part of the problems being solved [5, p. 143]).

[8] Hinton, G. (1989). Connectionist learning procedures. *Artificial Intelligence, 40* (pp. 185-234).

[9] Fahlman, S. and Lebiere, C. (1990). *The Cascade-Correlation Learning Architecture.* CMU-CS-90-100, School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213.

[10] Frean, M. (1989). *The Upstart Algorithm: A Method for Constructing and Training Feed-Forward Neural Networks.* Edinburgh Physics Preprint 89/479, Dept. of Physics, University of Edinburgh.

[11] Thornton, C. (1990). The complexity of constructive induction. DAI Research Paper No. 463, University of Edinburgh, Dept. of Artificial Intelligence.