Inductive Learning with External Representations

Mark Wexler
Laboratoire de la Physiologie de la Perception et de l'Action
11, pl. Marcelin Berthelot, 75005 Paris, France
Email: wexler@ccr.jussieu.fr

Abstract

External representation is the use of the physical world for cognitive ends, the enlargement of the mechanisms of representation to include the action-perception cycle. It has recently been observed that such representation is pervasive in human activity in both pragmatic and more abstract tasks. It is argued here that by forcing an artificial learning system to off-load all of its representation onto a (simulated) external world, we may obtain a model that is biased in a very natural way to represent functional relations in ways similar to those used by people. After learning a function from examples, such a model should therefore generalize to unseen instances in ways that we would consider correct. These ideas are tested by developing two machine learning systems, in which representation relies on the sensorimotor control of simulated robotic agents. These systems are able to represent a variety of functional relations by means of their action and perception, and they learn to spontaneously do so from examples. Moreover, they generalize extremely well to unseen problems even after a small number of examples, including on functions such as *n*-parity that are notoriously difficult to generalize for machine learning algorithms. It is argued that despite these systems' simplicity, the external representations that they evolve are similar to those used by people on similar tasks.

1. Introduction: From external representation to learning and back again

Internal and external representation. Any ambitious learning machine must possess some form of internal representation, it has long been acknowledged, for the reason that interesting things to learn—functions, concepts, etc.—are seldom directly expressible as simple input–output mappings, but rather seem to require re-representation of the raw input data.[1] The trouble with representation is that once we admit internal degrees of freedom—the architecture and weights of hidden layers in a neural network, for example, or the structure of a decision tree that is built up, etc.—we have a great deal of freedom in deciding the general form and the specific parameters of the our system's innards, which embody its internal representations. In all interesting cases, these hidden degrees of freedom are severely underconstrained by the data used to train the system. It might be profitable to consider what we know about representation in natural systems, for example in humans.

Internal representation in biological systems is neural-based. The appropriate unit of analysis for the understanding of the representation of high-level concepts (rather than, say, the representation of the fact that there is a bar of a certain orientation in the receptive field of a particular cell in visual cortex) may be anywhere from one neuron up to an ensemble of billions of neurons. The problem with neural representation is a practical one: we hardly know how it works. Many of the details of single-neuron mechanisms are known, but we have little idea how, or whether, single cells represent high-level concepts. At the other extreme we have large ensembles of neurons, some of whose connections are known, but whose causal role in cognition is far from understood. At best, the data we have are correlational: a neuron in an animal brain may change its activation systematically following a particular

[1] In this paper we will use "learning" to mean "learning a function from one set of numbers (input) to another set of numbers (output), from examples." The examples are called the *training set*, and include both input and desired output; they are used to adjust the internal parameters of the system. Following this learning phase, the system is presented with another set of examples, the *test set*, in order to quantify generalization. The usual caveats apply about this being a strong abstraction of what "learning" might mean; it is nevertheless useful for discussing fundamental issues in inductive learning.

sensation or preceding a particular movement; or an area of a human brain may be activated during the performance of a certain type of cognitive task. But we are very far indeed from knowing the causal neural mechanisms behind how, say, the representations of "*p*" and of "if *p* then *q*" might (or might not) lead to the representation of "*q*". These kinds of questions are probably not unanswerable in principle; but, despite dramatic recent progress in neuroscience, we are still far from this goal.

Together with internal representation, there is another, often overlooked variety: external representation. External representations can be defined as *states of the physical world deliberately created as part of a cognitive process, rather than for pragmatic ends* (Kirsh 1995; Zhang and Norman 1994; Clark and Chalmers 1998). A common example is the rearrangement of cards in games: one rearranges the cards in one's hand so as to render the relations between the cards more perceptually salient, to offload at least a part of the planning process onto the external arrangement, thereby freeing cognitive resources for other tasks—rather than for any immediate pragmatic end in the game. Another example is counting a large pile of similar objects, such as coins, where the difficulty is to count each item, and only once. People typically use strategies such as pointing to the border separating the counted from the uncounted items, or sorting the items into a new pile as they are counted. These actions have no direct pragmatic relevance to the task at hand, but are executed for their subsequent facilitating effects on one's mental operations: namely, keeping track of the items that have already been counted. As Clark and Chalmers (1998) point out, there is no principled way to separate such external representations from the internal, neural variety, based purely on the work that the two types of representation do.

External representations need not play only a static, memory-like role; they can do cognitive work, as well. In other words, a physical action performed on an external representation may be isomorphic to a mental operation performed on the analogous internal representation. For instance, consider having to determine whether a large number (of objects or events) is even or odd. An external representation for even might be, say, having one's index finger extended, with the index finger flexed standing for odd. The relevant way to operate on this representation would be to change the state of the finger (to flex it if extended, and vice versa) for every object or event to be tallied. If at the end the finger is in the initial state, the number is even, otherwise it is odd. This is functionally isomorphic to, for example, to 'mentally uttering' alternatively "even" or "odd" after every event to be counted. To take another example from the 'digital' domain, consider deleting a file in a windowed computer interface. An icon, an external representation of some information (more properly, an external representation of an external representation—the data on the disk) is dragged into the trash can icon and thereby disappears, a process that parallels the mental forgetting (or labeling as unimportant) some internally represented information. External representations thus need not be just reminders or facilitators; if well designed, they can also participate in external transformations that do cognitive work, in a process that parallels internal, neural representations being (presumably) transformed in the central nervous system.

Inductive machine learning. We now return to machine learning. In artificial inductive learning, the machine is trained on only part of the set of possible input–output pairs; once it reaches some criterion of success on this training set, the machine is tested for 'correct' generalization on the remaining test set. The number of generalizations consistent with the training set is usually very large. The trouble is: in many, probably most, natural cases of learning, no purely formal, procedural criterion can systematically pick out that (those) generalization(s) which we would consider 'correct,' out of this large field of possibilities. The fact that past experience cannot *formally* constrain future experience was famously pointed out by Hume; Goodman (1983) demonstrated with his elegant "grue" argument that without inductive bias, one cannot formally decide which predicates are projectible (i.e., can be generalized over). Mitchell (1980) and Wolpert (1996), among others, have discussed these ideas in the context of machine learning, showing, for instance, that, for a given learning algorithm, for every problem set that is generalized correctly there is one that is generalized incorrectly.

From a strictly unbiased, 'objective' point of view, any generalization that is consistent with the training set is as good as any other. Nevertheless, if all generalizations are equal, some generalizations strike *us*

as more clever or perspicacious. If we want to construct an artificial system that reasons in a human-like fashion, or if we want to model human inductive reasoning, it is these latter generalizations that we want our system to prefer.

The usual solution is to endow the learning system with systematic inductive bias. No formal system can be entirely free of bias, as it is obliged to formulate its hypotheses in some language; any such language excludes some hypotheses, and of the ones it does include makes some hypotheses easier and more natural to express than others. Another type of bias is due to the learning algorithm. This results in a norm, "badness", on the hypothesis language—usually the error committed by the hypothesis on the training set, for example, often combined with a factor to bias the system towards parsimony. Induction is then seen as a form of optimization or search to reduce the badness of the hypothesis.

In cognitive science, the traditional cure for the various sticky problems of learning has been innateness, or, less crudely, strong developmental constraints on the *contents* of eventual mental representations. This solution, however attractive, cannot be the whole story, however, for a number of reasons. First, shifting the burden from ontogeny to phylogeny begs the cognitive question, for phylogenetic learning is at least as difficult as its ontogenetic counterpart. Second, hard developmental constraints seem to be at odds with the real plasticity of central nervous system development (Quartz and Sejnowski 1997). Third, learning occurs on many time scales; humans are capable of fast learning, from very few examples, in rather artificial domains for which no direct innate bias could have evolved.

The problem of induction has recently been discussed from the point of view of intermediate representations (Kirsh 1992, Clark and Thornton 1997). In the inductive learning of patterns, categories or rules, the distinction is made between low-order regularities that are present directly in the training data (such as conditional probabilities between individual input and output bits that are close to 0 or 1), and more subtle higher-order regularities that can only be expressed in a richer language and in terms of the lower-order regularities (such as relational properties, etc.). The lower-order regularities can certainly be picked up by formal, fairly general and unbiased techniques. As for the higher-order regularities, if the learner is to succeed it must first re-represent the raw input in order to make the more subtle pattern manifest. (The learner can of course choose to treat the higher-order regularity as if it were lower-order, by, e.g., simply memorizing the input–output pairs, or by learning bit–by–bit associations. But then it would fail to generalize 'correctly': it will certainly have learned something, but not what we were trying to teach it.) The problem of perspicacious re-representation is thus seen as the critical component of higher-order learning.

Multi-layer neural networks have become popular mainly due to the assumption that the intermediate layers will somehow carry out this re-representation, and the magic is supposed to be that this is induced by means of a completely mindless procedure, hill-climbing. Clark and Thornton (1997) have shown that this is not so for a number of difficult learning problems; the one I will discuss is *n*-parity, as I also use this problem in my toy models (see below). The *n*-parity mapping receives *n* bits as input and outputs one bit, the sum of the input modulo 2. This mapping is hard to learn from examples because they provide no raw statistical information: all the conditional probabilities between input and output bits are 0.5. It is well known that multi-layer feed-forward networks can 'learn' parity by means of hill-climbing—but this is only when they are trained on *all* the 2^n input–output pairs. No simple associationistic learning will do for this rule: changing any bit of the input flips the answer. Reproducing the training set is necessary but not sufficient for having a concept or a rule; simple memorization will lead to the same result, and we would not then say that the memorizer has learned a rule, because (among other reasons) no re-representation of the input has taken place. A sharper test for having learned a rule is of course correct generalization to previously unseen cases.

Clark and Thornton have found that *no* training algorithm on *any* network architecture leads to networks that generalize correctly to previously unseen cases of parity; in fact, even in the best cases it suffices to withhold a very small fraction of the problems from the training set for generalization to fail completely.[2] This is very bad news for neural networks. Practically, it shows how bad general network methods are at generalization: in problems of

[2] The closest that any neural network—or any general learning system—has come to generalizing parity is Pollack's (1992) cascaded network, which, given *all* input strings of length 1 and 2, and some longer strings as training, generalizes correctly to *some* longer input strings.

practical interest no system can be trained on *all* cases. More fundamentally, it shows that, at least in this case, that when a network learns to reproduce input–output pairs, what it has actually learned is entirely unlike the rule that *we* have used to arrive at the training set. Indeed, what reason do we have to suppose that a neural network, with its peculiar dynamics and bias, would generalize beyond the training set in the way *we* would? The only possible reason would be the biological verisimilitude of artificial neural networks; but this, apparently, is not enough. And lest one specifically blame neural networks, Clark and Thornton found that popular symbolic algorithms, such as decision trees, fare just as badly on generalization.

Learning with external representations. We are faced, on one hand, with traditional machine learning methods based on intuitively appealing principles such as association (i.e., the solution of new problem should be close to that of a previously seen problem, if the two problems are close, in some metric) and parsimony (shorter descriptions are preferred to longer ones, in some language) that nevertheless fail to evolve appropriate internal representations and therefore fail to generalize 'correctly' to unseen training cases, on many interesting problems. On the other hand, Nature offers few clues about the shape and dynamics of appropriate internal representations, other than some general information about nerve cells and their connections and interactions, which by themselves do not seem to be sufficient constraints. In such circumstances, a prudent way to proceed would be to seek inspiration from the one type of high-level representation that we can observe in detail, namely external representation.

We will therefore study learning and generalization in systems that are obliged to make use of external representations. We will do this by developing a toy model that is built in such a way as to be forced to off-load all of its representation onto a (simulated) external world. What is meant by this is that its dynamical degrees of freedom have a passing resemblance to the sensorimotor links that biological creatures have with the outside world. This resemblance is cartoon-like, in the same way that artificial neural networks are an extremely abstract version of real neural systems. It is to be hoped that such a rough sketch will nevertheless capture something essential about the effect of external representation on learning.

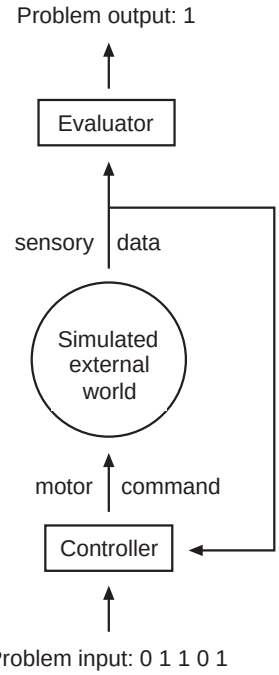


Figure 1: The general scheme for the sensorimotor learning systems discussed in this paper.

We will experiment with two variations on the general theme, in Sections 2 and 3, respectively, that have somewhat different sensorimotor embeddings. The first system is modeled after a simple animat: it lives in an analog two-dimensional world, its repertoire of actions being confined to self-rotation and forward motion. The second system is embedded in a discrete, one-dimensional world similar to Blocksworld, where it can move but also pick up and drop objects.

The general architecture of our sensorimotor learning systems is shown in Fig. 1. The problem is fed, one bit at a time, to the controller. The architecture of the controller is different for the two models to be presented. In the first case it is a perceptron neural network; in the second, a hierarchical LISP-like program. In both cases, the controller is 'representationally shallow', in the sense that it cannot itself store any state, that it cannot re-represent the input in any way.

The presentation of a problem always begins with the system in a canonical world state. The controller translates the input (and possibly the current sensory state) into a motor command, to be executed in the

(simulated) external world. The motor command is executed, sensory data collected, and, together with the next element in the problem sequence, fed back into the controller. This procedure is repeated for the length of the problem input. Finally, the evaluator (which has the same general architecture as the controller) translates the sensory data from the final world state into a solution or output to the current problem.

It may be argued that since the 'external' embedding of our systems is only simulated, it is not external at all; this type of representation is just a special case of internal representation. This is true, but beside the point. What is important is the way that the representational capacity of the system has been sliced: a shallow controller and evaluator that themselves cannot store or represent anything, forcing all representation to be offloaded onto the simulated external world. As we will see, this architecture, together with the sensorimotor details, strongly constrains what can be represented, what can be learned from examples, and how this learning generalizes.

The models to be presented in Sections 2 and 3 are trained using versions of genetic algorithms. The advantage of this choice is that it leaves the form of the representations completely free. We will devote most attention, however, not to how the systems learn—although their learning is quite efficient—but to how they generalize, once they reach a learning criterion. This is because learning performance depends strongly on the incidental details of the learning algorithm. Generalization performance, on the other hand, which is largely independent of the learning algorithm, answers the real question that we want to address: what the system really learns from the limited training data, given its external representation constraint.

2. External representation and learning in an animat[3]

We begin with a very simple creature with a highly simplified sensorimotor embedding in a simulated external world, similar to an animat. This particular creature lives on a 2-dimensional plane; its external 'world' state is simply its position on the plane and its heading. Time is taken as discrete. At each tick the creature issues a motor command, telling it the distance to travel forward and the angle by which to turn. The sensory data returned consist of the distance to a fixed landmark, and the angle between the animat's heading and the landmark (given as a sine and cosine, to avoid discontinuities).

The architecture of the system is a specialization of the general scheme given in Fig. 1. The controller and evaluator are perceptrons—i.e., feedforward neural networks with no hidden layer. The controller has one input (for the current problem bit) and two outputs (the motor commands); the evaluator has three inputs (the sensory data) and one output (the problem output). In this case, there is no feedback connection of sensory data to the controller (but there is such a connection in the system presented in Section 3).

The function that we will teach this system will be n-parity. Before the beginning of the presentation, the creature is put in a canonical world state, that is a given position and heading. As described in the Introduction, the input bits of the parity problem are fed one–by–one to the controller, which translates them into motor commands. After all the input bits are exhausted, the evaluator translates the final sensory data from the world state into the problem output. Having no other representational methods at its disposal (such internal recurrent connections), the system is obliged to represent the problem, and to keep track of the intermediate results, by means of its action.

The system is trained by means of a genetic algorithm. Each experiment proceeds as follows. A fraction f of the 2^n problems are assigned to the training set; the other $2^n (1 - f)$ problems are assigned to the test set and are never used in training (the training set always has an equal number of even and odd cases). In each generation of the GA each member of the (initially random) population is evaluated on the $2^n f$ training problems. (The weights and thresholds of the networks are coded as 10-bit strings in the genome.) A logical-0 input bit is coded as –1 for the neural networks, a logical-1 as +1. The neural networks have sigmoid activation functions between –1 and +1, with bias. The score on each problem is the absolute value between the output of the sensory-output network and the true answer; the score on the entire training set is the mean of the scores on each problem. The population (size 50, 10 bits/ weight) is evolved by both 2-point crossover and mutation, with rank-based fitness. The experiment was stopped as soon as at least one member of

[3] The results in this section have been reported in Wexler (1996).

the population reached criterion on the training set, a score of 0.001 or below. (Occasionally experiments ran over 200 generations without reaching criterion; these were discarded.) The best member of the population is then tested on the 2^n $(1 - f)$ problems in the test set, which the population had never seen during its training. The average score on the best population member on the generalization test is the score for the experiment. 100 experiments were run for each value of f, with new training and test sets were chosen for each experiment. (Further details available on request. All techniques used were very generic. The results were not sensitive to small variations in the parameters.)

In order to have a comparison for the generalization performance of these systems, the same task was run with ordinary, non-embedded neural networks. The idea was to make things as hard as possible for hypothesis by comparing the generalization performance of the embedded systems with that of the *best* generalizing ordinary networks. As shown by Clark and Thornton (1997), feed-forward networks for the non-temporal version of the problem are quite miserable at generalization. For the temporal version feed-forward networks won't do, as they do not preserve state, and therefore at least some recurrent connections are required. After experimenting with a number of architectures, it was found that simple recurrent nets generalize best. Within this class, 1-*a**-*b*-1 architectures are the best (* denotes a recurrent context layer), and as long as b is not too large the performance depends essentially on a; $b = a$ was found to be the best choice. The three best architectures are 1-2*-2-1, 1-3*-3-1, and 1-4*-4-1. These networks were trained by exactly the same method as the embedded sensorimotor systems. It should be noted that they got stuck in local minima much more often than the embedded systems.

The results for 4-parity are presented in Fig. 2, where the mean generalization performance is plotted against f, the fraction of the 2^4 problems that were withheld from the training set. Error of 0.5 corresponds to chance level. (There is a good but not very enlightening explanation for why the control nets actually perform worse than chance for small values of f.) The embedded, systems with external representation generalize almost perfectly down to $f = 0.25$. As for the control nets, with the marginal exception of the 1-2*-2-1 architecture (which is almost pre-engineered to become a parity-calculating flip-flop), they generalize very poorly (as do Clark and Thornton's models): omitting just two problems gives very high error, and at four problems they are close to chance level. Even the 1-2* -2-1 architecture has errors that are 50–100 times greater than those of the embodied systems for f above 0.25. The problem length can be increased without substantially changing the results: keeping f fixed one obtains similar generalization performance for 5-, 6-, and 7-parity.

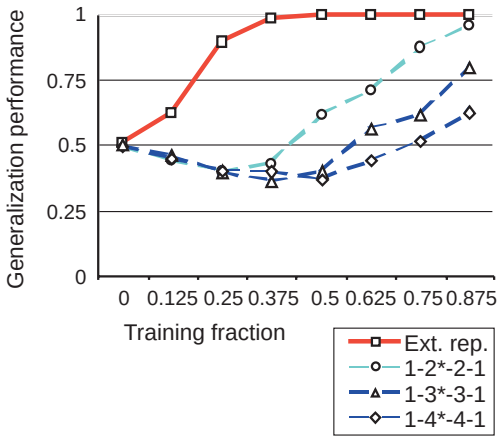


Figure 2: Generalization performance as a function of training fraction on the 4-parity problem for the system using external representation and the three best SRN controls. Chance level is at 0.5.

The interesting question of course is how the embodied systems managed to represent the parity function. As already discussed, these systems had no internal means to represent the problem, therefore they had to perform all 'computations' externally, i.e., by means of their movement. The systems that generalized successfully (which, as can be seen in Fig. 2, is most of the trained systems) adopted variations on the following controller strategy: do nothing if the input is 0, turn by 180˚ if the input 1. To then calculate parity, the evaluator simply had to give 0 if the creature oriented in its original direction, and 1 if it was oriented the other way. Many of the systems performed additional, spurious movement which had little or no effect on the final answer.

It should be emphasized that this toy system is not meant as a 'model' for any natural learning system. Instead, it should be seen as a metaphor and an illustration of how a naturalistically inspired sensorimotor embedding can lead to external representation that usefully channels learning.

3. A second example with a symbolic controller

The main proposal of this paper to use external representation as a means to channel appropriate generalization is independent of the controlling mechanism. Thus, in the systems described in the previous section, the crucial aspect was the systems' external embedding, not their controlling mechanism, which happened to be a neural network. Broadly speaking, a more flexible controlling mechanism will allow the system to represent more complex functional relations given the same external representational medium, but in itself is not a means to canalize generalization.

To illustrate this idea, we will now describe a new learning system that will be similarly embedded in a pseudo-physical world, and will rely on this embedding for its representational mechanisms, but will be controlled by a very different mechanism from neural networks: a LISP-like program, trained by means of a genetic algorithm.[4] Like the systems described in the previous section, the new system is composed of a controller and an evaluator. The system receives the bits of the binary training and test patterns serially; based on the current input and world state, the controller carries out a series of actions that may alter the world state; the input thus processed, the evaluator returns an output that depends on the final world state.

The world that the new systems are embedded in is a discrete half-line. Always starting at the origin, the system can move in either direction, provided it doesn't bump into the wall. Each position on the half-line may contain zero or more objects (with all objects cleared at the beginning of a training or test pattern). When receiving an input bit of 1, the system has an object placed in its 'hand'; otherwise its hand is empty. It can deposit this object at its current position, or, provided there is an object at its current position, can pick it up. Finally, the system may perform a number of tests, and base its subsequent actions on the outcome.

The controllers are LISP-like programs, composed of a set of operands. Each controlling program is a *list*. A list is composed of a *head* (which are given in Table 1) followed by zero or more elements, each of which may either be an *atom* (which are given in Table 2) or another list.

Head	Description
`(PROG a b ...)`	Executes operands in order, returning the value of the last operand. The empty list `(PROG)` is the same as `(IFS)`.
`(NPROG a b ...)`	Same as `PROG`, with the return value inverted.
`(IFI a b ...)`	If the current input is 1, execute and return the value of `a`, otherwise `b`, if present. Subsequent elements are ignored.
`(IFS a b ...)`	If there are one or more objects at the current position, execute and return the value of `a`, otherwise `b`, if present.
`(IFO a b ...)`	If the current position is at the origin, execute and return the value of `a`, otherwise `b`, if present.

Table 1: Types of list heads.

Atom	Description
`LEFT`	Moves one unit of distance towards the origin, if possible.
`RIGHT`	Moves one unit of distance away from origin.
`PUT`	If the hand is holding an object, deposits it at the current position.
`TAKE`	If there is one or more object at the current position, moves one object into the hand.

Table 2: Types of atoms.

Atoms perform actions; atoms and lists return values. (Empty conditionals do nothing and return their value; `(IFS)` returns `1` if objects are present at the current position, and otherwise `0`.)

The systems are trained using a genetic algorithm similar to Koza's genetic programming (Koza 1992). The parameters of the training algorithm[5] are not very important, though, as they tend to bear on the efficiency with which systems that reproduce

[4] Somewhat paradoxically, the choice of learning algorithm will be seen as not very important for our purposes. A particular choice of algorithm might make the system learn some training sets faster, but, at least to a first approximation, has no effect on generalization performance. Thus we could instead have used a brute search through the space of LISP-like programs, resulting in much less efficient learning of the training sets, but very similar generalization performance.

the training set are evolved but not those systems' subsequent generalization scores. A brute-force search through tree space, though highly inefficient, gives very similar generalization scores as the genetic algorithm, once systems that perform perfectly on the training set are found.

This architecture has the capacity to represent and to learn a variety of 'intuitively simple' boolean functions. Consider, for instance, our old friend parity. A system whose controller is `(IFO (IFI RIGHT) (IFI LEFT))`, and whose evaluator is `(NPROG (IFO))` 'solves' the parity problem in a way reminiscent of the systems from the previous section, but using its position to keep track of state: `0` is represented by being at origin, `1` by being at the neighboring position. There are other solutions possible. A system that uses objects rather than positions has controller `(IFI (IFS TAKE PUT))` and evaluator `(PROG)`; `0` is represented by no objects in the stack, `1` by one object. Mixed space/object solutions also exist, for instance controller `(IFS (IFI RIGHT) PUT)` and evaluator `(PROG)`, which represents `1` by an object at the current position but zeroes the state by moving to an unoccupied adjacent spot. All three types of representations, and others besides, have been evolved by the system.

The architecture not only has the *capacity* to represent parity, its external representational mechanism makes parity simple to represent (as illustrated by the above examples) and therefore *canalizes* the system to learn parity and to generalize it correctly after very few examples. This is illustrated in the same way as in the previous section. We consider length-4 parity; of the 16 possible problems, the system is trained on a randomly chosen training subset of 16 *f* problems until its performance is perfect.[6] Its generalization is then tested on the remaining, never–before–seen 16 (1 *f*) problems. (Here, performance is defined as the fraction of problems solved correctly, so 0.5 is chance level.) Each training fraction *f* is re-run many times, with the training and test sets reshuffled each time. The result is shown in Figure 3.

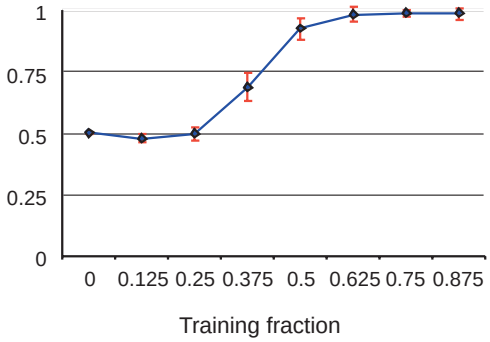


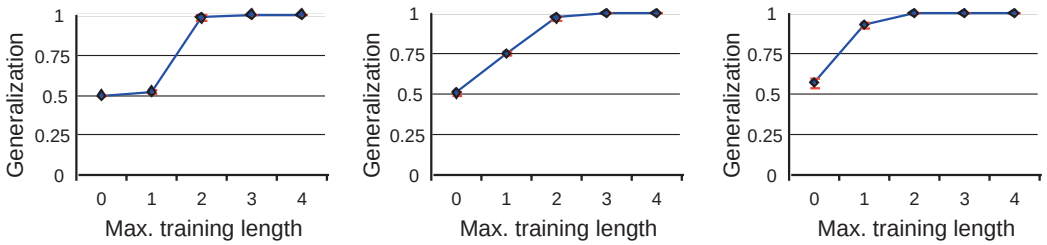
Figure 3: Generalization performance on the 4-parity problem as a function of the fraction of problems used for training. Error bars are standard deviations.

With very few problems included in the training set, the system understandably performs at chance level. But when as few as 6 out of 16 problems are used for training, the average performance is significantly better; with 8 problems performance is above 90%, and with 10 or more it is very close to perfect. When the size of the problem is increased, the size of the training set needed to achieve a given generalization level grows very slowly; training on 10 problems, for example, leads to about the same generalization score for 5-parity as training on 8 problems does for 4-parity. For comparison, the reader should keep in mind that most machine learning methods, based on associativity rules, are obliged to represent a function like parity descriptively (such as dividing the *n*-dimensional hypercube of the problem space by a large number of hyperplanes) rather than procedurally. The resulting descriptions are complex compared to the simple programs evolved by our system (and grow rapidly more complex as the problem size increases), and are found to generalize very poorly omitting just *one* problem is usually sufficient to reduce the system to chance performance.

The system can learn and generalize a variety of other interesting functions. We illustrate with *length parity*, *boolean AND*, and *boolean OR*. Length parity is the function that returns the parity of the number of input bits, regardless of the value of the bits

[5] Population size 1000. Root of tree either `PROG` or `NPROG`, with equal probability. Subsequent elements chosen from Table 2 with equal probability; probability of branching a new list is 0.4, of terminating the list 0.3. Minimum length 5 for controller, 3 for evaluator. Top 10% of each generation copied unchanged into next generation; the rest of the population is 'sexual offspring' of previous generation with one tree swap (Koza 1992), with parent choice probability proportional to inverse rank.

[6] Care is taken to include the same number of problems in the training set whose answer is 0 as those whose answer is 1. If the training criterion is not reached after 50 generations, the system is restarted.



(a) length parity (b) boolean AND (c) boolean OR

Figure 4: Generalization performance on three functions, as a function of maximum length of training problems. Generalization was tested on all unseen problems through length 6.

(unlike ordinary parity); it can be thought of as a simple period-2 timer. One way our system can represent length parity is with controller `(IFO RIGHT LEFT)` and evaluator `(NPROG (IFO))`, making itself into a period-2 oscillator, cycling between the starting position and the next cell to the right on every bit of the input. Other variations on this theme can also be learned and generalized correctly. For example, the *length modulus 3* function returns 1 if the length of the input is a multiple of 3, and zero otherwise; it may be represented by the controller `(PROG (IFO (PROG RIGHT RIGHT RIGHT)) LEFT)`, and evaluator `(NPROG (IFO))`. This system again uses position for its representation; it moves two units to the right on the first bit of the input, then one unit to the left on each of the next two bits, and so on. The function can be learned, and generalizes as well as length parity. The systems implementing the length parity and length modulus 3 functions may be thought of period-2 and period-3 timers.

Boolean AND can be represented by controller `(IFI (PROG) RIGHT)` and evaluator `(IFO)`; boolean OR by controller `(PROG PUT)` and evaluator `(IFS)`. As with parity, there are many alternate ways that the system can represent these functions. Once again, however, the system's representational mechanism canalizes it into correct generalization with very few examples.

For the length parity, boolean AND, and boolean OR problems, the system was trained on all input patterns of a given length (the 'training length'), and tested on all longer problems, through length 6 bits. The generalization performance is shown in Fig. 4.

As can be seen in Fig. 4, excellent generalization can be achieved with very small training sets. When problems of length 0 to 2 are included, generalization is very close to perfect.

With the inclusion of an additional operand, `(IFHS a b ...)`, that evaluates `a` if the state of the hand is the same as the state of the current location (i.e., both have at least one object, or both are empty), and otherwise `b`, the system acquires a capacity to represent, learn and generalize a new set of functions, requiring short-term memory. Consider the function, which we will call $B_N = B_{N\,1}$, that returns 1 iff the last bit is equal to the second-to-last bit. It might at first seem that, lacking any internal state, our system would be incapable of representing such a function (and, *a forteriori*, of learning it from examples). But the system has another way to implement memory, through off-loading the information onto the external world. Consider, for instance, the controller `(PROG PUT RIGHT)`. Every time that this system receives an input bit, it 'writes' it on its current spatial location by leaving the space empty if the bit is 0, and placing an object there if the bit is 1; it then moves to the next spatial location on the right. The system thus turns its one-dimensional space into a transcript of its temporal input sequence, much as people do when we reason about time in spatial terms. Now if the evaluator were to be `(PROG LEFT TAKE LEFT (IFHS))`, the system will represent precisely the $B_N = B_{N\,1}$ function: it backs up, 'picks up' the final bit, backs up once more and compares the contents of that spatial location with the contents of its hand. Similarly, to compare the third–to–last bit with the fifth–to–last, the $B_{N\,2} = B_{N\,4}$ function, the system would need the evaluator `(PROG LEFT LEFT LEFT TAKE LEFT LEFT (IFHS))`, or one that is equivalent.

This type of short-term memory not only can be represented by the system, but can also be learned from (a small number of) examples, and then generalized correctly. This is shown in Fig. 5, the gener-

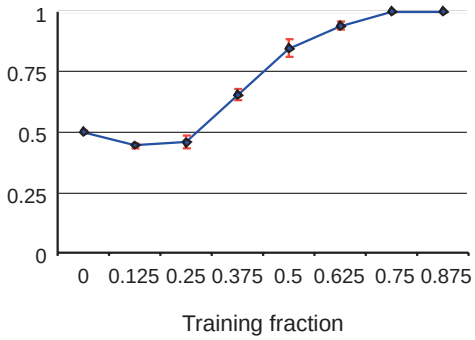


Figure 5: Generalization curve for the short-term memory function $B_N = B_{N\,1}$ with $N = 4$.

alization curve for the $B_N = B_{N\,1}$ function for $N = 4$. All conditions were identical to those used to test parity (see Fig. 3), except for the inclusion of the `IFHS` conditional. The $B_N = B_{N\,2}$ function had a virtually identical generalization curve.

4. Discussion

I first summarize the main lessons of the simulations.

- Using the two models developed here, I have shown that simulated robots with physical-like degrees of freedom can use action on their simulated worlds to represent a variety of abstract functional relations.
- The systems learn these external representations from small training sets of examples, and, most importantly, generalize to previously unseen cases 'correctly'—in ways that people would find reasonable.
- No prescription is given to the systems as to *how* to use their action to represent the functions, and indeed, for a particular system and particular training set, a variety of representational strategies are observed to arise spontaneously.
- These results hold true for at least two different kinds of simulated robot, one moving on a continuous two-dimensional world, the other on a discrete one-dimensional world. The second robot can, in addition to moving, pick up and drop objects. The second robot seems to be able to represent a richer variety of functional relations in virtue of its expanded sensorimotor repertoire.
- The architecture of the sensorimotor controller and evaluator does not seem to be crucial, as long as it allows sufficient flexibility.

An interesting question to ask is: is there anything in common between the our systems' representations and ours? People do, of course, make frequent use of external representations, most obviously in practical tasks (as in separating a pile of objects in two when counting, or laying out objects in a spatial order that 'represents' the temporal order of their use; for further examples, see Kirsh 1995). But does external representation play a role in less pragmatic and, at least notionally, more abstract reasoning, such as the learning and representation of relations studied in the previous two sections?

Consider the way the model in Section 3 learns the short-term memory functions that test the equality of the last bit in the sequence with the second- or third–to–last bits. Having no 'internal' memory, the model is obliged to off-load the information onto the external, spatial configuration of objects. By dropping an object at its current position if the current input is 1, and moving to the next position, the system converts the temporal input sequence into an ordered spatial representation. This reminds us of the well-known propensity that people have to represent and reason about temporal events in terms of spatial structures and metaphors.

Another comparison can be made between our models' representation of the parity function, and typical human representations of the same relation. It can be revealing to ask people to learn, for instance, the parity function from examples. (This is of course more interesting with those who are mathematically naive, and do not possess the ready-made notion of parity.) Start off by saying, "0 gives 0 and 1 gives 1," and let the subject pose longer sequences as questions. Most people guess after very few examples, and guess 'correctly' (i.e., they guess parity), often after learning that "1 1 gives 0" but that "1 1 1 gives 1." Observing carefully what the subject does and says during the evaluation of a problem, as well as his description of the algorithm, provides important clues about the representation employed. Some people actually use physical movement. For instance, when asked "what does 1 1 1 1 0 1 0 1 give," a subject of this type might hold out a finger and, every time there is a "1" in the sequence, change the finger's 'state' by flexing or unflexing it; if, at the end of the sequence, the finger is in the original state, the subject will answer "0,"

and otherwise "1." It is important to note that such strategies (that have obvious connections to, and possible reinforcement from, counting on one's fingers) are evolved spontaneously, without any directions, other than the solutions given to the posed questions.

Other subjects do not produce overt movement, but employ similar strategies acting on *imaginary movement*. A typical example is imagining a coin (starting on the "heads" side, say), and imagining flipping it for every "1" in the sequence; if it is "heads" at the end the answer is "0," otherwise "1." Although there is no *overt* movement involved in this type of imagination strategy, it has been shown that the imagination of self-movement in the brain is closely related to the production of movement in the activation of cortical areas, and in the close matching of spatio-temporal characteristics (Jeannerod 1997). What is more, it has also been shown recently (Wexler, Kosslyn and Berthoz 1998) that manual movement (such as rotation) interferes in a very selective way with the cognate transformation of mental images even of abstract objects (such as mental rotation). Thus, a strategy in which one imagines transforming an object may be very closely related to the fully external strategy in which one physically and overtly transforms an external object. Instead of overtly acting on the world and letting the world perform the memorization and calculation, one covertly simulates this action and internally predicts its outcome.[7]

A precise developmental sequence that interpolates between external and internal representation, and between overt and covert action, has been established by V. Pouthas (1985). In her experiments, children aged 4–8 years had to produce, and therefore to represent, specific time delays. In a paradigm borrowed from animal studies, the child had to wait a precise amount of time after a signal (usually 10 or 20 seconds), and then press a button. Accurate performance was rewarded (with an interesting display), but the child received no other indications of what to do. Those of the youngest subjects (4 years old) who succeeded all employed external represenation. For instance, one child would jump out of his chair as soon as the signal was given, run to the door of the room, slap the door twice, run back to the chair, and then press the button. A strategy such as this can work, and work systematically, because of the biomechanical and neurological constraints on movement that insure the high repeatability of the time needed to execute the same chain of actions; the trick is, of course, to find a chain of actions that 'represent' a given target delay. The actions themselves, entirely constructed by the subject, differed widely. Another child would get up out of her chair, turn the chair (a light cube) upside-down and back, sit back down and then press the button. Another child would sway her entire body sideways, like an inverted pendulum, a number of times, and then press the button. None of the youngest children could systematically produce the required delays while remaining still. These children's only way to 'represent' a time delay is to execute a chain of actions that, for biomechanical reasons, systematically takes about the same time.

This changed in children who were older. There was still movement, but its amplitude decreased and it became more erratic. One child, for instance, sat still but moved her finger back and forth—compare this with the whole-body movement seen in one of the younger subjects. Finally, by age 8 children are able to produce the delay without any overt movement whatsoever; most likely, they counted to themselves, as adults would typically do. An interpretation of this developmental sequence, in light of the arguments given here, is as a progressive internalization of an external representation. The youngest subjects do not have access to an internal timer, or at the very least find it more natural to offload the task onto the world through self-movement and/or object manipulation. The progressive diminution of the reliance on external representation can of course be interpreted as the its gradual replacement by a disembodied internal clock. But it would certainly be more parsimonious to interpret it as an internalization of the previously utilized external representations. Instead of performing an action in order to use its duration as a timer, the subject *simulates* the action (such imagining performing an oscillatory movement, or subvocalic speech); the imaginary action can work just as well as a timer, since its tem-

[7] This assumes, of course, that there are specific mechanisms in the brain to predict the outcome of about–to–be–executed action, or, short-circuiting the action-perception cycle, of imagined action. There is much evidence for such a mechanism, probably located in posterior parietal cortex. See Clark and Grush (1998) for a summary of some of the evidence, as well as an interesting discussion of its link to mental representation.

poral parameters closely match those of the overt action (Jeannerod 1997).

The external representations evolved by our toy system closely resemble those actually used by people to represent the same relations, as discussed in the preceding paragraphs. In the case of parity, both the analog and digital artificial systems typically adopt an external representation of two states (usually by means of self-rotation, for the analog system; through self-translation or object manipulation for the digital system), a rule that amounts to "do nothing if input is zero; change state if input is 1"; and a final evaluation rule, to the effect of "if we're in the same state as initially, the answer is 0, and otherwise 1."[8] This is functionally identical to the typical spontaneous human representations that involve either overt movement (i.e., using a finger to keep track of state), or imagined movement (i.e., imagining flipping a coin). The case of length parity and length modulus 3 functions can be considered as a period-2 and period-3 timers, respectively. Our system learns to represent these functions through cyclic movement: the period-2 by moving right then left then right then left, etc., the period-3 by moving right two spaces then left then left, etc. This is qualitatively very similar to the self-movement strategies spontaneously adopted by 4-year-old children when faced with the task of reproducing 10- or 20-second intervals, as discussed above.

Finally, we come back to the theme of generalization. The generalization of a given training set to previously unseen cases is always an ill-posed problem. Formally, without inductive bias any generalization is as good as any other, by some set of principles. Given this, if we want artificial systems that generalize as we do (and a large part of reasoning involves different types of induction), it makes sense to constrain their representational mechanisms to be similar to ours, for, as we have seen, constraints on representation are an effective way to introduce inductive bias. Having little information about the detailed dynamics of internal, neural representations, we turn to external representations as a form easy to study and imitate, at least on the level of a toy model. Implementing a system (in two versions) that has no choice but to represent problem state through very simple sensorimotor dynamics, we construct devices that not only can learn a variety of simple functional relations, but, due to their representational constraints, generalize these relations in the way most people would consider 'correct'—a non-trivial accomplishment, at least for the parity function, which most general-purpose learning systems fail miserably to generalize. Moreover, the representations evolved by our systems resemble closely those observed in people learning the same relations. This canalization of 'correct' generalization suggests that perhaps external representation plays a non-negligible role in human inductive reasoning. Indeed, the abundance of examples of how we employ action in our reasoning suggest that this might be the case, even when we perform no overt action, but rather internally simulate its effects.

[8] This representation is far from being the only one possible. One could, for instance, add up the input bits, and at the end subtract by two's until reaching 1 or 0. Or one could represent the input string as a vertex of a hypercube and divide up the space in the appropriate way with a large number of hyperplanes.

References

Brooks, R. A. and Stein, L. A. (1993) Building brains for bodies. MIT AI Memo 1439.

Clark, A. and Chalmers, D. (1996) The external mind. Manuscript, Dept. of Philosophy, Washington University.

Clark, A. and Grush, R. (1997) Towards a cognitive robotics. Manuscript, Dept. of Philosophy, Washington University.

Clark, A. and Thornton, C. (1997) Trading spaces: Computation, representation and the limits of uninformed learning. *Beh. Brain Sci.* 20: 57–90.

Goodman, N. (1983) *Fact, fiction, and forecast*. Cambridge, Mass.: Harvard Univ. Press.

Harnad, S. (1990) The symbol grounding problem. *Physica* D 42: 335–46.

Kirsh, D. (1992) From connectionist theory to practice. In M. Davis (ed.) *Connectionism: Theory and practice*. New York: Oxford Univ. Press.

Kirsh, D. (1995) The intelligent use of space. *Art. Intell.* 73: 31–68.

Koza, J. R. (1992) *Genetic programming: On the programming of computers by natural selection*. Cambridge, Mass.: MIT Press.

Mitchell, T. (1980) The need for biases in learning generalizations. Technical Report, Department of Computer Science, Rutgers University, No. CBM-TR-117.

Pollack, J. (1992) The induction of dynamical rec-

ognizers. *Machine Learning* 7: 227–252.

Pouthas, V. (1995). Timing behavior in young children: A development approach to conditioned spaced responding. In: Michon, J. A. and Jackson, J. L. (eds.) *Time, mind, and behavior.* Berlin: Springer-Verlag.

Quartz, S. and Sejnowski, T. (1997) The neural basis of cognitive development: A constructivist manifesto. *Beh. Brain Sci.* 20: 537–596.

Wexler, M. (1996) Embodied induction: learning external representations. *AAAI 1996 Fall Symposium.*

Wexler, M., Kosslyn, S. M. and Berthoz, A. (1997) Motor processes in mental rotation. *Cognition* 68: 77–94.

Wolpert, D. (1996) The lack of a priori distinctions between learning algorithms. *Neural Computation* 8: 1341–1390.

Zhang, J. and Norman, D. A. (1994) Representations in distributed cognitive tasks. *Cognitive Science* 18: 87–122.