A Neuroidal Architecture for Cognitive Computation

LESLIE G. VALIANT

Harvard University, Cambridge, Massachusetts

Abstract. An architecture is described for designing systems that acquire and manipulate large amounts of unsystematized, or so-called commonsense, knowledge. Its aim is to exploit to the full those aspects of computational learning that are known to offer powerful solutions in the acquisition and maintenance of robust knowledge bases. The architecture makes explicit the requirements on the basic computational tasks that are to be performed and is designed to make these computationally tractable even for very large databases. The main claims are that (i) the basic learning and deduction tasks are provably tractable and (ii) tractable learning offers viable approaches to a range of issues that have been previously identified as problematic for artificial intelligence systems that are programmed. Among the issues that learning offers to resolve are robustness to inconsistencies, robustness to incomplete information and resolving among alternatives. Attribute-efficient learning algorithms, which allow learning from few examples in large dimensional systems, are fundamental to the approach. Underpinning the overall architecture is a new principled approach to manipulating relations in learning systems. This approach, of independently quantified arguments, allows propositional learning algorithms to be applied systematically to learning relational concepts in polynomial time and in a modular fashion.

Categories and Subject Descriptions: F1.1 [Computation by Abstract Devices]: Models of computation—neural networks; I2.0 [Artificial Intelligence]: General—cognitive simulation; I2.4 [Artificial Intelligence]: Knowledge Representation Formalisms and Methods; I2.6 [Artificial Intelligence]: Learning

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Cognitive computation, computational learning, learning relations, nonmonotonic reasoning, PAC learning, robust reasoning

1. Introduction

We take the view that intelligence is a large-scale computational phenomenon. It is associated with large amounts of knowledge, abilities to manipulate this knowledge to derive conclusions about situations not previously experienced, the

© 2000 ACM 0004-5411/00/0900-0854 \$05.00

Journal of the ACM, Vol. 47, No. 5, September 2000, pp. 854-882.

This research was supported in part by NSF grants CCR 95-04436 and CCR 98-77049, ONR grant N00014-96-1-0550, and ARO grant DAAL-03-92-G-0115.

A preliminary version of this paper appeared in Lecture Notes in Computer Science, vol. 1443. Springer-Verlag, New York, 1998, pp. 642–669.

Author's address: Division of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138.

Permission to make digital/hard copy of part or all of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery (ACM), Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee.

capability to acquire more knowledge, and the ability to learn and apply strategies of some complexity.

The large-scale nature of the phenomenon suggests two prerequisites for constructing artificial systems with these characteristics. First, some theoretical basis has to be found within which the various apparent impediments to this endeavor that have been identified can be addressed systematically. Second, some large-scale experiments need to be conducted to validate the suggested basis—it is possible that the fundamental phenomena do not scale *down* and that small-scale experiments do not shed light on them.

Much effort has been devoted to identifying such a theoretical basis. One major thrust has been to develop definitions of capabilities that are functionally adequate. Functionalities that, even if realized, would not go far towards achieving a significant level of performance are of little interest. Another thrust has been the search for capabilities that are demonstrably computationally feasible. Functionalities that are computationally intractable are again of little direct interest. A third viewpoint is that of biological plausibility. Perhaps an understanding of how cortex is constrained to perform these tasks would suggest specific mechanisms that the other viewpoints are not able to provide.

The hypothesis of this paper is that for intelligent systems to be realized the sought after theoretical basis has not only to be discovered, but needs to be embodied in an *architecture* that offers guidelines for constructing them. Composed as these systems will be of possibly numerous components, each performing a different function, and each connected to the others in a possibly complex overall design, there will need to be some unity of nature among the components, their interfaces, and the mechanisms they use.

Our purpose here is to describe a candidate for such an architecture. This candidate emerged from a study that attempted to look at the issues of functional adequacy, computational feasibility and biological constraints together [Valiant 1994]. We call the architecture *neuroidal* since it respects the most basic constraints imposed by that model of neural computation. One feature of that study was that it was a "whole systems" study. It addressed a range of issues simultaneously insisting on biological and computational feasibility, plausible and simple interactions with the outside world, and adequate explanations about the internal control of the system. It suggested that, while definitions of intelligence and thinking may continue to prove elusive, there may be a basic computational substrate that systems that realize these phenomena need to share, which may be identified more easily. We believe that the functions and mechanisms associated with this substrate characterize an area that transcends both natural and artificial computation and is appropriately described as *cognitive computation*.

In abstracting an architecture from that neural study, as we do here, we dispense with some of the most onerous constraints considered there to be imposed by biology. In the mammalian brain, these constraints include the sparsity of the interconnections among the components, the inaccessibility of the separate components to an external teacher or trainer, and certain bounds on the weights of individual synapses. In artificial systems, these constraints are not fundamental, and in order to maximize the computational power of the architecture we free ourselves of them here.

Our purpose in describing the architecture is to suggest it as a basis for large scale experiments. A first critical issue clearly, is whether the architecture is theoretically adequate: Can some of the obstacles that have been identified by researchers be solved, in principle, within the model? Are there further fundamental obstacles?

We observe that the McCulloch–Pitts model may turn out to be a valid architecture for intelligence if it turns out that threshold elements represent well the basic steps of cortical computation. Its shortcoming, clearly, is that it does not appear to offer useful guidelines for constructing systems or for understanding their capabilities.

Our architecture, in contrast, is designed to provide a design methodology. In particular, its main feature is that it comes with a set of associated algorithms and mechanisms. The two basic constituents of the architecture are classical enough, being *circuit units* consisting of linear threshold elements, and short-term memory devices called *image units*. The novelty is that for aggregates of such devices we can detail how the accompanying mechanisms can perform a list of tasks that address significant problems. The bulk of this paper is devoted to enumerating these problems and describing how they can be addressed. Our purpose here is to point out that this broad variety of mechanisms can be supported on this single unified architecture, and that together, they go some way toward addressing an impressive array of problems.

In particular, we shall provide mechanisms to address the following eight questions. We consider that any large scale system exhibiting intelligent behavior as we envision that here will have to take a position on each of them.

- (1) *Conflict Resolution*. What mechanism is provided to ensure that choices between plausible but inconsistent alternative actions or classifications are resolved in a principled manner?
- (2) *Learning from Few Examples*. What mechanism is provided to ensure that inductive learning can be performed even when few examples are available and the description of each one is highly complex?
- (3) *Multiple Objects and Relations*. When concepts need to be expressed in terms of relations among several objects rather than as a propositional combination of predicates about one object, what mechanism is provided to ensure that the learning of the concept and the recognition of instances of the concept, are computationally feasible?
- (4) *Learning Strategies*. How is the learning of sequential processes, such as strategies, handled?
- (5) *Robustness*. How is the system made resilient to errors or to inconsistent use of categories in the descriptions of the examples or the rules that are supplied to the system, and hence able to cope with a noisy world?
- (6) *Context*. What mechanisms are provided to help cope with the fact that individual rules usually apply only in restricted contexts?
- (7) *Reasoning*. What mechanisms are provided for reasoning, and to what extent are they sound and computationally tractable?
- (8) *Reasoning with Partial Information*. How are the issues of incomplete information and nonmontonic reasoning addressed?

The extent to which the architecture constitutes progress toward constructing intelligent systems remains to be determined. The performance of a system

would clearly depend on both the high-level design as well as the quality of the training data. The magnitude of the problems that each of these two components presents will become apparent from any large-scale experimentation.

The largest benefit that this architecture promises is a capability for realizing large *robust knowledge bases* by learning by a process that one might call massive *knowledge infusion*. We expect that in order that this potential be fully realized there will be needed a significant new kind of effort directed towards the *preparation of teaching materials*. It would be convenient if currently available resources such as dictionaries and annotated corpora of text were usable. However, since the system will not operate identically to humans, the teaching materials needed cannot be expected to be identical to those that are effective for humans. The system and teaching materials need to be chosen so as to fit each other, and we believe that the development of these in tandem is a major challenge for future research. Although, clearly, there is much data currently available in natural language form, aside from the issue of natural language understanding, there is the independent question of whether these materials make explicit all the commonsense and other information that one would wish a system to know.

To summarize, the main feature of the architecture is that it brings learning to the heart of the general problem of intelligence. Recent advances in machine learning help to distinguish those aspects of learning that have very effective computational solutions, from those for which none is known. For example, the problem of finding new compound features that are particularly effective in a specific learning task is not one to which general effective solutions are known. In contrast, we know that for some classes of functions the presence of irrelevant features can be tolerated in very large numbers without degrading learning performance. Our proposal, therefore, is that insights of this kind be incorporated into the design of systems that perform cognitive computations. We claim that this approach does offer a new view of some of the now traditional problems of artificial intelligence, and alleviates at least some of them.

On a historical note, it is interesting to recall Turing's [1950] paper on "Computing Machinery and Intelligence" in which he describes his "Test". While he considers the possibility that a programmed system might be built to solve it, he appears to come out in favor of a learning machine. In the subsequent history of research on general intelligent systems, the overwhelming emphasis has been on programmed systems. We believe that this was largely because there existed a mathematical basis for following this avenue, namely mathematical logic. However, in the intervening years there has been much progress on the theoretical foundations of machine learning, as well as success in the experimental study of learning algorithms. It is, therefore, particularly appropriate now to reexamine whether a synthesis can be found that resolves Turing's dilemma.

2. The Architecture

A system having the proposed architecture is composed of a number of *circuit units* and a number of *image units*. The image units can be thought of as the *short-term* or working memory of the system: In the course of an interaction with the world or of a complex reasoning act, this is where the intermediate results are stored. The circuit units are the *long-term* repositories of large amounts of

knowledge: their contents can be updated both by inductive learning as well as through explicit programming, but each individual update changes only a small part of the total long-term knowledge base.

The circuit units are defined so as to ensure that they have the most desirable learning capabilities, in particular error-resilience and attribute-efficiency. For this reason, we shall define them here to consist of one layer of linear threshold gates. The circuit units have one layer of inputs that are the inputs to these gates, and one layer of outputs that are the outputs of the gates themselves. The weights of the threshold elements can be either learned or programmed. In principle, any set of functions that is superior as far as expressiveness or learnability could be used instead of threshold functions to enhance the architecture.

The intention is that each output gate in a circuit unit recognize some predicate. We say that it "fires" if such recognition has occurred in the sense that the output has taken value one or "true." The inputs may be Boolean or real valued. The latter is needed if, for example, some of the inputs are from input devices or image units that produce numerical values.

The image units are the main repositories for the transient information that is computed in a recognition, reasoning or planning task (cf. Miller [1956]). In this paper, we shall, for illustrative purposes, employ a particular instance of them designed to make them directly comparable to predicate calculus mechanisms. In this instance, the contents of the image at any time consists of a set of token objects, and a set of relational expressions that describe the properties of and relationships among the objects represented. The information in the image unit may be provided to the inputs of circuit units. The contents of the image units may be modified by the outputs of circuit units or by input devices that take input from the outside world. Thus, image units can be used both to store information received from the outside world through sensory devices, as well as to store "imaginary constructs" that are useful for the system's internal computational processes. It is possible to have a number of such constructs in the image simultaneously in a novel combination. The system can then bring to bear the power of the large knowledge base stored in its circuits on a representation of a complex situation not previously represented in it. In general, an image unit may be used to analyze an input, to perform deduction on it, or to perform planning by enumerating a sequence of imaginary depictions of possible sequences of future situations. Our use of predicate calculus notation is mainly for illustrative purposes. For processing visual information, for example, other representations may be preferable and image units might be formalized in other ways to incorporate that fact.

The circuit and image units will be composed together in a block diagram so that the outputs of some of the units are identified with the inputs of others. Typically, image units will interface directly with circuit units, rather than other image units. Some inputs of some units are identified with the outputs of input devices, and some outputs with the inputs of output devices. The block diagram may ultimately contain feedback or cycles. It will typically include a stack of circuit units that represent concepts that are more and more complex and further removed from the perceptual data inputs.

In any implementation of the architecture, mechanisms need to be specified for realizing a range of tasks, such as those enumerated in Section 4. In addition, some further mechanisms are needed to allow the acquisition of programmed knowledge. For example, new output gates may be added to a circuit unit and the parameters or weights of these gates assigned appropriate values so that they fire under the intended conditions. These added gates can then also serve as targets for inductive learning or as inputs to other units. One method of adding to the circuit is to add gates to represent various fixed functions of existing nodes so as to enrich the feature space. Thus, one may add nodes that compute the conjunction of all pairs from a certain set of existing nodes. Another method of programming a circuit is to change a weight between an existing pair of nodes. Thus, one can create a subcircuit to realize "grass \Rightarrow green" by making the weight from the node representing "grass" to that representing "green" larger than the threshold of the latter.

2.1. OVERVIEW. We shall outline here the properties that are required for a device to be an image unit. In the section to follow, we shall describe one particular realization of image units that is based on predicate calculus notation and highlights how our approach can be used to generalize the capabilities of certain logic-based AI systems.

An image unit will at any instant contain some data that we call a *scene S*. The scene itself consists of a number of *object tokens a*₁, \cdots , *a*_N as well as information about the properties of the objects and about the relationships amongst them. This information may be represented, in principle, in any number of ways, among which predicate calculus notation is one.

The computational processes that we describe are entirely in terms of interactions among the circuit units and the image units. The circuit units are the repositories of rules that may have been acquired by either learning or programming. These rules are applied in the image units to the scene in question. We limit the number N of *objects* in the image units to some moderate number, say 10, so as to limit ab initio the computational cost of the so-called binding problem, which is concerned with matching variables in the stored relational rules to objects in the scene.

Although we use standard logical notation to describe the contents of the image or to label circuit nodes, our interpretation of this notation is slightly different from the usual. In particular the objects a_1, \dots, a_N in our architecture are best viewed as tokens or internal artifacts of the system. They are not intended to refer to the external world in the same direct way as in more familiar uses of the predicate calculus. Further, the primary *semantics* that we ascribe to nodes in the system are the PAC semantics described in Section 3 and in Valiant [2000] rather than the standard semantics of predicate calculus.

Each node of a circuit unit is at any one instant in some state. Most simply, it is in one of two states that indicates whether or not the predicate associated with the node is true of the current scene in the associated image unit. However, it can, in principle, also contain further information, such as a real number that expresses a confidence level for the validity of the predicate (cf. neuroids in Valiant [1994]).

Each node can be thought of as representing a relation, for example,

859

$$R(x_1, x_2, x_3),$$

where the variables range over the objects a_1, \dots, a_N in the scene. We shall, for simplicity, emphasize in this paper the restricted case that all the quantified variables are quantified *existentially*. Thus, the threshold gate at this node will *fire*, in general, if $R(x_1, x_2, x_3)$ holds for the current scene for *some* binding of x_1, x_2, x_3 to a_1, \dots, a_N . As discussed in Valiant [2000], this case can be generalized without any substantial changes to allow arbitrary use of universal in addition to existential quantifications.

While no general assumptions are made about how R is represented, it is assumed that when an expression of the form $R(x_1, x_2, x_3)$ is recognized to hold for the current scene, a corresponding *binding*, or mapping θ from $\{x_1, x_2, x_3\} \rightarrow \{a_1, \dots, a_N\}$ is made explicit in the system. To be precise, therefore, when a node for R fires, *it fires for a particular binding* θ . For brevity, where this leads to no ambiguity, we shall use R variously to refer to the predicate computed, the value of the predicate or to the node itself that evaluates R.

An aggregate of circuit units is a directed graph in which each connection between a pair of nodes that is connected is associated with a *weight*. Each node has an update function that updates its state as a function of its previous state, the states of nodes from which there are directed edges to it as well as the weights of these edges. More particularly each circuit unit is a directed graph of depth one (i.e., one layer of input nodes, one layer of output nodes, and no "hidden layer") and the update functions are linear threshold functions.

A directed edge from a node representing R_1 to a node representing R_2 contains binding information called the *connection binding* $R_1 \rightarrow R_2$. If the nodes represent $R_1(x_1, x_2, x_3)$ and $R_2(y_1, y_2, y_3)$, then the binding information could specify, for example, that x_1 and y_2 have to represent the same object in the image but that x_2 , x_3 , y_1 and y_3 may be objects arbitrarily different from each other.

There may be several connections from R_1 to R_2 , each one corresponding to a different connection binding and having a different weight. For example, if R_2 represents the notion of grandparent and R_1 that of parent, then in the definition of the former one may wish to invoke the latter twice with different bindings.

Thus, if grandparent_via (x_1, x_2, x_3) is to represent that x_1 is the grandparent of x_3 via x_2 , and if parent (y_1, y_2) represents that y_1 is the parent of y_2 , then there would be two connections from parent to grandparent_via: In one, the binding would be $x_1 = y_1, x_2 = y_2$ while in the other $x_2 = y_1$ and $x_3 = y_2$. If the weight of each connection is 1 and the threshold at the node grandparent_via is 2, then the appropriate conjunction parent $(x_1, x_2) \land parent(x_2, x_3)$ would be computed.

The scene σ in an image unit contains information about which relations hold for which object sets. An aggregate of circuit units will be evaluated for any one scene S node by node. For a node that represents relation $R(x_1, \dots, x_i)$ the set of all bindings $\theta\{x_1, \dots, x_i\} \rightarrow \{a_1, \dots, a_N\}$ that make R hold will be evaluated. If the node R has inputs from nodes R_1 and R_2 , and the gate at R evaluates the threshold function $R_1 + R_2 \ge 2$, for example, then R will fire for a binding θ if *there exist* bindings θ_1 , θ_2 of the variables of R_1 and R_2 that are consistent with θ with respect to the connection bindings $R_1 \rightarrow R$ and $R_2 \rightarrow R$, and make both R_1 and R_2 true. We note that in the example of the previous paragraph, if $R = \text{grandparent}_{via}$ via and $R_1 = R_2 = \text{parent}$, and N = 5, then for $\theta(x_1) = a_5$, $\theta(x_2) = a_2$, $\theta(x_3) = a_4$, the node for R would fire for binding θ if the following conditions held: $\text{parent}(a_5, a_2) = 1$ and $\text{parent}(a_2, a_4) = 1$. The reason is that the bindings θ_1 and θ_2 , where $\theta_1(y_1) = a_5$, $\theta_1(y_2) = a_2$, $\theta_2(y_1) = a_2$ and $\theta_2(y_2) = a_4$ are consistent with θ with respect to the stated connection bindings for the two occurrences of $\text{parent}(y_1, y_2)$, and make these two occurrences true.

More generally, it may be that $R(x_1, x_2)$ is a function of $R_1(y_1, y_2)$ and $R_2(z_1, z_2)$ and arguments occur in R_1 and R_2 that are not bound to the arguments of R by the connection bindings. Suppose, for example, that $y_1 = x_1$, $z_1 = x_2$ and that y_2 and z_2 are not bound. Then, the architecture allows the unbound variables to be quantified, either universally or existentially, and this information to be encoded in the corresponding connection binding, as long as the quantification in R_1 is independent of that in R_2 . Thus

$$f(\exists y R_1(x_1, y), \forall z R_2(x_2, z)) \Rightarrow R(x_1, x_2)$$

$$(2.1)$$

is possible where f is a threshold function of two Boolean arguments. The existential quantification $\exists yR_1$ denotes that, for some object y in the scene, $R_1(x_1, y)$ is true. The universal quantification $\forall zR_2$ denotes that, for all objects z in the scene, $R_2(x_2, z)$ is true. The important constraint is that the quantification we call *independently quantified arguments* (IQA). This is the main single technical innovation that permits the architecture to successfully unify learning and reasoning in a relational setting while retaining computational tractability. It provides a direct method of learning in a relational domain by means of a propositional learning algorithm. This is described in more detail in Valiant [2000]. Certain alternative ways of treating relations systematically in a learning context appear to introduce intractability almost from the start [Haussler 1989].

2.2. AN IMPLEMENTATION OF IMAGE UNITS. We shall describe an implementation that is oriented towards predicate calculus notation, but extends it in the directions advocated in this paper to allow for effective learning with attributeefficiency and error-resilience. We consider the following restriction of the language of predicate calculus (e.g., Russell and Norvig [1995, p. 186]). As constants, we shall choose the base objects $\mathcal{A}_B = \{a_1, \dots, a_N\}$ of the image unit. We allow a set \Re of base relations of arities that are arbitrary but upper bounded by a constant α . We use no function symbols. We represent variable names by $\{x_1, \dots, x_n\}$ and relations will be defined in terms of these. For illustrative purposes, we shall observe that any programmed system in which knowledge is described as Horn clauses, without function symbols can be embedded into this framework. The central role in AI systems of this language of description is discussed by Russell and Norvig [1995, pp. 265–277] and this is also the datalog framework in databases [Ullman 1984]. We are observing, therefore, that in *at least* this one possible implementation of our architecture, it is the case that a significant class of existing programmed systems can be embedded. Our main claim, however, is that the additional benefits of being more expressive and of having a powerful learning capability make our system superior in kind over Horn representations. Also, while we are showing here that the function computed at a circuit gate can be interpreted as a logical rule, it is the case that

such a function can be associated with probabilistic information in addition, such as a probability of being correct.

The two features of predicate calculus that we exclude, therefore, are constants that refer directly to individuals in the world, for example, Shakespeare, and functions that can be applied to them to refer to other individuals, for example, mother_of(Shakespeare). In our system, the only constants are the predefined base objects of the image, which are internal constructs of the system. In order to refer to an individual like Shakespeare, we shall use a unary predicate that can be applied to a base object. Thus, Shakespeare (a_3) would express the fact that a_3 has the attributes of Shakespeare. Also, we dispense with function symbols by referring to the resulting object as a base object, and by expressing the necessary relationship by an appropriate relation symbol. Thus instead of saying $x_1 =$ mother_of(x_2) we would say mother_of(x_1, x_2), where the latter is a binary relation. In these senses, we ensure that the two linguistic restrictions, on constants and functions symbols, do not constrain absolutely what can be expressed. The aim of these restrictions is to impose some upper bound, namely N, on the number of token objects that the system needs to represent in an image at any one time. Note that by using appropriate encodings one can have such a token object in the image represent a set of many objects in the world.

A term will be therefore a base object a_i , and an *atomic sentence* will be a single relation such as mother_of $\in \Re$ applied to the appropriate number of base objects (e.g., mother_of(a_3, a_7). A Horn *rule* will be an implication

$$\begin{aligned} R_{i_1}(x_{i_1,1},\cdots,x_{i_1,\alpha(i_1)}) \wedge \cdots \wedge R_{i_r}(x_{i_r,1},\cdots,x_{i_r,\alpha(i_r)}) \\ \Rightarrow R_{i_0}(x_{i_0,1},\cdots,x_{i_0,\alpha(i_0)}) \,. \end{aligned}$$

where $R_{i_j} \in \Re$ and $\alpha(k)$ is the arity of $R_k \in \Re$. We note that \Re may include the zero arity relation False which signifies logical impossibility and may be used meaningfully for R_{i_0} on the right-hand side of an implication. Horn rules may have some of the x arguments instantiated by fixed values from \mathcal{A} , but all the other x variables are to be interpreted as quantified universally over \mathcal{A} .

A binding θ is a mapping from $\{x_1, \dots, x_n\}$ to $\{a_1, \dots, a_N\}$. We say that a relation $R(x_1, x_2, x_3)$ is made true by θ if $R(\theta(x_1), \theta(x_2), \theta(x_3))$, or $R(\theta(\underline{x}))$ for short, holds. The derivation rule *modus ponens* is the following:

"if $R_{i_1}(\underline{x}) \wedge \cdots \wedge R_{i_r}(\underline{x}) \Rightarrow R_{i_0}(\underline{x})$ " is a rule, and if, for some binding θ , $R_{i_j}(\theta(\underline{x}))$ holds for $1 \le j \le r$, then $R_{i_0}(\theta(\underline{x}))$ also holds.

In a logic-based system, we would program a set of rules of the form (2.1) and consider the input to be a set of atomic sentences. We could then consider the output to be the set of all atomic sentences that can be deduced from the input by applying the rules using modus ponens in all possible ways. This output could be derived in the logical framework by means of forward chaining [Russell and Norvig 1995, p. 273].

Let us now describe the implementation of the neuroidal architecture into which this process embeds naturally. In this implementation, the contents of the image will be simply the atomic sentences of the input. The circuits will implement the rules as follows: For each relation $R \in \mathcal{R}$, there will be a gate in the circuit. We shall assume that each relation R occurs on the right-hand side in just one rule – otherwise, we replace multiple occurrences of R on the right-hand side by different names, say R^1, R^2, \dots, R^m and add a new rule $R^1 \vee R^2 \vee \dots \vee R^m \Rightarrow R$. For such rules, we can use the identity connection binding from each R^i to R.

For each Horn rule $R_{i_1} \wedge R_{i_2} \wedge \cdots \wedge R_{i_r} \Rightarrow R_{i_0}$, we shall make a *connection* to R_{i_0} from each R_{i_j} $(1 \le j \le r)$. This connection will have the appropriate *connection binding*, defined below. At the R_{i_0} node, we shall implement the equivalent of an AND gate in terms of a threshold gate with threshold *r*. In other words, we regard the values of the R_{i_j} as Boolean $\{0, 1\}$, and have a gate that realizes

$$R_{i_0} = 1$$
 if and only if $\sum_{j=1}^r R_{i_j} \ge r$.

Executing this threshold gate will correspond, therefore, to performing one application of modus ponens.

To simulate OR gates, which are used as shown above if multiple occurrences of the same relation occurs on the right-hand side, we also use a threshold gate but have 1 as the threshold instead of r, that is, $\sum R_{i_i} \ge 1$.

In this implementation we consider the network to evaluate for each gate all *possible* bindings θ : $\{x_1, \dots, x_n\} \rightarrow \{a_1, \dots, a_N\}$ so as to find all the ones, if any, that make the gate evaluate to one. In the case that the aggregate of circuit units is acyclic, we can form a topological sort of their nodes (e.g., Knuth [1973]) and for one such topologically sorted order evaluate each node in succession. The evaluation of each node is for all the N^d bindings θ , where $d \leq \alpha$ is the arity of the relation R at that node. More generally the circuits will contain cycles, which allow recursive rules to be expressed. This makes the evaluation mechanisms slightly more complex, though still polynomial time for constant α . Full details of this can be found in Valiant [2000].

Our architecture is more expressive than is required for evaluating these Horn clauses in several respects. The ability to quantify arguments is one such respect. We shall therefore now clarify the nature and meaning of the connection bindings and independent argument quantifications: Each rule with r relations on the left corresponds to a threshold function with r arguments, where each argument has its own connection binding. Consider the following rule with r = 2:

$$\exists x_1 R_1(x_1, x_2, x_3) \land \exists x_4 R_2(x_3, x_4, x_5) \Rightarrow R_3(x_2, x_3, x_5).$$
(2.2)

This notation expresses the connection bindings implicitly. The binding of the connection from R_1 to R_3 says simply that the second parameter of R_1 binds with the first parameter of R_3 , the third parameter of R_1 binds with the second of R_3 , and that x_1 is quantified existentially. Similarly the binding of the connection from R_2 to R_3 identifies the first parameter of R_2 with the second parameter of R_3 , and the third parameter of R_2 with the second parameter of R_3 , and the third parameter of R_2 with the third of R_3 . The naming of the variables in each rule or gate can express this precisely. Note that in this example the implication is that any binding of x_1 that makes R_1 true, can be combined with any binding for x_4 that makes R_2 true, in order to make R_3 true. For

example, if elsewhere in the rule set, we have the rules $R_4(x_7, x_8, x_9) \Rightarrow R_1(x_7, x_8, x_9)$ and $R_4(x_7, x_8, x_9) \Rightarrow R_2(x_8, x_7, x_9)$, then it will be sufficient for these two rules to be satisfied for two *different* values of $\theta(x_7)$ in order to make R_3 true, since (2.2) did not require otherwise.

If from some R_i there is more than one connection, then for conceptual purposes it is simplest to think about an expanded circuit in which the R_i node is replicated so that each node now has just one connection directed away from it and hence the circuit is a tree. This is because connection bindings impose joint constraints on the several inputs to a node but not to the several outputs.

Note also that any correspondence between two variables occurring in two relations on the left hand side (i.e., x_3 in the example (2.2)) can be enforced only by having x_3 as an explicit variable in the right hand side relation, that is, R_3 in the example. This can be circumvented, however, in the following sense: We could create a gate called $R_6(x_2, x_5)$ for computing $\exists x_1 \exists x_3 \exists x_4 (R_1(x_1, x_2, x_3) \land R_2(x_3, x_4, x_5))$ by first evaluating $R_3(x_2, x_3, x_5)$ as in (2.2) and having an additional rule $\exists x_3 R_3(x_2, x_3, x_5) \Rightarrow R_6(x_2, x_5)$.

The evaluation process can be viewed as being performed on a graph structure. In this graph, the input nodes with no predecessors will each represent a relational expression, such as $R(a_3, a_4, a_7)$. To evaluate a gate at R, we simply enumerate all the N^d bindings of the *d* variables that appear in it. First, we scan all the atomic sentences in the image that contain R and see which bindings make R true before any rules are applied. Then, for each binding, and for each predecessor node, if any, we determine whether that binding can make true the quantified expression, for example, $\exists x_1 R_1(x_1, x_2, x_3)$ that is the corresponding argument in the rule for R. Having done this for each predecessor, we can, for each binding compute the value of the gate at the current node, whether it is a disjunction, conjunction, or more generally, an arbitrary linear threshold gate. Note that the complexity of this task that is contributed by the binding problem is N^{d} . It is exponential in the *maximum arity* of the relations, and not in the number of base objects or the number of relations in a rule! This fact encapsulates the computational benefits of IQA. (Note, however, that the arity may be made larger than in some equivalent predicate calculus representations by the restriction that all binding information is in the connections. This necessitates that if some correspondences among the variables need to be enforced on the left-hand side of a rule, they must be made explicit on the right-hand side. For example, if we wish to represent father(x, z) \land mother(z, y) \Rightarrow grandfather(x, y), then in analogy with how R_6 above is represented by means of R_3 , our representation of grandfather will need to be in terms of grandfather_via that has a third argument, say t, that is to be identified with the two occurrences of z by connection bindings.)

It is easy to verify that such an evaluation algorithm will compute all satisfying bindings for each relation that is represented at the nodes, in exactly the same way as would applying modus ponens in all possible ways to the rules.

From what has been said, it should be clear that our architecture is expressive enough that programmed systems based on modus ponens and the Horn rules we have described can be embedded into it. What the architecture adds to such systems is more expressivity and a capability for learning. The gates we allow are not just Boolean conjunctions and disjunctions, but linear threshold functions. This allows a host of learning algorithms, discussed in later sections, that provide provable learning capabilities that are not known to be available in strictly logical representations.

Learned rules can be incorporated in the architecture in a number of ways. A learned rule takes the form of an equivalence, rather than an implication in the first instance [Valiant 2000]. Even then it is natural to use it in just one direction—as if it were an implication—in order to justify the deduction that a certain base relation R holds for a set of base objects when a certain complex set of conditions is satisfied.

With regard to learnability, we note that there are theoretical results that show for certain classes of functions that extending the representation of the learner may make the original class more easily learned, while restricting the learner to the minimal Boolean representation needed for expressing the functions would make the task NP-complete [Pitt and Valiant 1988]. While the quoted results refer to the polynomial time criterion, as we shall discuss below at length, we also desire and seek here the very significant further advantages that learning be achieved with both attribute-efficiency and error-resilience. The expressiveness of our architecture reflects these considerations.

3. Semantics

We cannot expect to develop a set of robust mechanisms for processing representations of knowledge without a robust semantics for the representation. The emphasis here is both on the necessity of *semantics*, that relates the representation to some reality beyond itself, as well as *robustness* to the many uncertainties, changes and errors in the world or in communications with the world, with which the system will need to cope.

The need for semantics explains the attractiveness of formal logic and probability theory in AI research. It is the need for robustness that forces us to look beyond these, to notions centered on learning. The semantics we shall describe here, *PAC semantics* is based on the notion of computationally feasible learning of functions that are probably approximately correct [Valiant 1984].

To explain the contrast in viewpoints consider the situation calculus described in McCarthy and Hayes [1969]. There, a situation is "the complete state of the world", and general facts are relations among situations. Thus, $P \Rightarrow Q$ means that for all situations for which P holds Q holds also. This is an all embracing claim about some universe that is difficult to grade gracefully and becomes problematic in the real world where authoritative definitions of P and Q themselves may be difficult to identify.

In contrast, PAC semantics makes qualified behavioral assertions about the PAC behavior of a particular system in a particular environment. In PAC semantics, P and Q would be defined as functions computed by fixed algorithms or circuits within a system that takes input through a fixed feature set from a fixed but arbitrarily complex world. The inputs range over a set X that consists of all possible combinations of feature values that the input feature detectors can detect. There is a probability distribution D over this set X that characterizes the world in all the aspects that are discernible to the feature detectors of the system. The P and Q could correspond to nodes in a circuit. The relationship between P and Q would be typically of the following form: If a random example is taken from D that makes the P node fire, then it also makes the Q node fire with a

certain probability. The latter probability would, at best, be known to be in some range with a certain confidence. Thus, the semantics is relative to both the computational and sensory abilities of the system, and refers to an outside world about which direct observations can be made only one observation at a time. The claims that the semantics makes about a rule do not go beyond what is empirically verifiable in a computationally feasible way by the system operating in the world.

In addition to making observations from D, the system can also acquire rules by being told them. It can then use these as working hypotheses, subject to subsequent empirical testing by the system against observations from D, and make deductions using them. The general goal of the system is to accumulate knowledge about the world in the form of rules that hold in D with a probability that is estimatable by the system.

Rules are intended to capture the time-invariant aspects of the world. Their subject matter may, nevertheless, be how things change with time. A rule may, for example, express the later consequence of an action. Such rules, which relate different points in time, can be invariants of the world and hence PAC learnable.

In this paper, PAC semantics has the following more particular aspect. The domain X of observations refer to the possible contents of an image, or the system's "mind's eye" rather than the external world directly. A person's knowledge of physics, kings or dinosaurs is rarely based on direct observations. The computational significance of this approach is that it restricts the domain of discourse to the fixed set of internal token objects a_1, \dots, a_N . The philosophical significance is that, unlike the conventional semantics of predicate calculus when applied to unaxiomatized knowledge, it makes the domain of discourse specific.

In constructing an artificial system, we envisage that one would take advantage of the possibility of training each circuit unit separately. Typically a unit will take as inputs the outputs of input devices or other circuit or image units to which it is connected. It would see the world through a set of features that are themselves filtered through the input devices and circuits of the system. In contrast, the outputs of the unit being trained will be directly accessible to the trainer, who can venture to train each such output to behave as desired. If the system consists of a chain of circuit units trained in sequence in the above manner, then the errors in one circuit need not propagate to the next. Each circuit will aim to be accurate in the PAC sense as a function of the external inputs-the fact that intermediate levels of gates only approximate the functions that the trainer intended is not necessarily harmful as long as each layer relates to the approximations at the previous layer in a robustly learnable manner. At each internal level, these internal feature sets may nevertheless permit accurate PAC learning at that next stage. That this is indeed possible for natural data sets remains to be proved. Some analysis of this issue of hierarchical learning has been attempted [Rivest and Sloan 1994].

We note that by iteratively modifying the contents of image units by means of functions computed by circuit units our architecture can perform computations that are more dynamic than what the conventional static view of circuits might suggest. All the computational functions of the system, including those that use the image for reasoning as outlined in Section 4.7, for example, need to be effective in the PAC sense. The key advantage of PAC semantics is that it gives an intellectual basis for understanding learning and thereby validates empiricism and procedural views of knowledge. Inductive learning will be the key to overcoming the impediments that are to be enumerated in the next section. These will include defaults, nonmonotonic reasoning, inconsistent data, and resolving among alternatives. Intelligent systems will inevitably meet the dilemmas that these issues raise but they will have a learned response to at least the more common manifestations of them. For knowledge that can be systematized and expressed consistently by formal logics, the traditional declarative semantics offer the important advantages of compositionality, and easier comprehensibility. For other kinds of knowledge, where systematizations are not known, these benefits of traditional logic may not be attainable in the same sense.

We mention that PAC learning, when offered as a basis for intelligent systems or cognitive science, suggests the following view. The world is arbitrarily complex and there is little hope that any system will ever be able to describe it in detail. Nevertheless by learning some simple computational circuits such systems can learn to cope, even in a world as complex as it is. In the first instance, we define these circuits to act in a deterministic way. It is the complexities of the world and the uncertainties in the system's interactions with it that force the framework of the semantics to be probabilistic.

The circuits can be represented as a set of rules [Valiant 2000] and each rule can be associated with probabilistic information, such as an estimate of the probability that the rule holds and a confidence level that is justified for this estimate. If such extra information is available then a variety of probabilistic reasoning mechanisms (e.g., [Pearl 1988] beyond those described in this paper can be brought to bear in addition. We note, however, that probabilistic rules may make the learning task less tractable [Kearns and Schapire 1994].

Besides the question of whether the rules correspond to probabilistic statements, there is the independent question of whether the circuits are allowed randomization in their behavior, so that under identical conditions they produce different results at different times as a result of internal random choices.

Reasoning with probabilistic rules, and randomization in the circuits could be both incorporated within the scope of PAC semantics and the neuroidal architecture. This paper, however, emphasizes that deterministic circuits that represent rules that are correct with high probability may form an adequate basis for constructing systems for cognitive computations.

4. Some Algorithmic Mechanisms

In this section, we shall enumerate a series of algorithmic techniques for the described architecture. We claim that these mechanisms address issues that are fundamental to large-scale systems that are to perform cognitive computations. Our observation here is that they can be brought together to provide a powerful overall methodology within a single framework.

4.1. CONFLICT RESOLUTION. The circuit units will contain large numbers of nodes. In general, each node corresponds to a concept or action and has some reference to the world or to the internal constructs of an image unit. Let us suppose that each one, when regarded separately, is highly accurate, correct say

on 99% of the natural input distribution. The problem that arises is that because of the sheer number of nodes, perhaps in the hundreds of thousands, a large number of the nodes will typically behave incorrectly on almost any one input.

In a natural scene, we expect a certain moderate number of the predicates represented in the circuits to hold and the corresponding nodes to fire. However, among the large numbers of remaining nodes a certain number that should not fire will do so also. Further, some of these will be in semantic conflict with the correct ones. They may recommend inconsistent actions (e.g., "go left" as opposed to "go right") or inconsistent classifications (e.g., "horse" versus "dog").

This is a fundamental and inescapable issue for which a technical solution is needed. A conventional approach might be to suggest that each node be given a numerical "strength," and that in situations where several nodes fire in conflict the one with highest strength be chosen to be the operative one. However, before this can be offered as a technical solution, one would need to exhibit mechanisms for deriving and updating these strengths. Further, this approach makes the assumption, without justification, that a single totally ordered set of numerical strengths is sufficient for the overall resolving process.

Our approach is the following: we have a large number of circuit nodes, computing functions u_1, \dots, u_n , say, of the scene S. We assume that each u_i is correct in the PAC sense with high probability. The proposed resolving mechanism is to have another set v_1, \dots, v_n of nodes where v_i corresponds to u_i . The purpose of v_i is the following: When v_i fires this will be taken to assert that u_i is firing, and further that u_i is *preferred* over all the other u_j that may be firing. The implementation will have a circuit unit with u_j $(1 \le j \le n)$ as inputs and v_i $(1 \le i \le n)$ as outputs, and with a connection from each u_j to each v_i . Each v_i will be a linear threshold function of the u_j . Thus, if u_7 , when firing, is to be preferred to all the other u_j except for u_2, u_5 and u_8 , any of which are preferred over u_7 , then the appropriate linear inequality to be computed at v_7 will be

$$u_7 - u_2 - u_5 - u_8 \ge 1 . (4.1)$$

This will have the effect that v_7 will fire if and only if u_7 fires and none of u_2 , u_5 or u_8 fires.

The force of this approach is two-fold. First, it is more expressive than the regime in which there is a single global ordering on strengths. For each u_i , one can individually specify which u_j dominate it. Second, the representation needed for expressing the preferences, namely linear threshold functions, are learnable attribute-efficiently and with error-resilience as further discussed in subsequent subsections.

We are therefore suggesting that the conflict resolution problem when formulated in this reasonably expressive sense has a simple and effective technical solution: the correct resolutions can be learned from examples of past behavior.

The justification of our architecture that we further elaborate in subsequent sections can be viewed as the conjunction of a variety of facets of the same fundamental idea: learning can resolve many otherwise problematic issues, and the neuroidal architecture can support an appropriate learning mechanism in each case. We chose to discuss this conflict resolution problem first as it is a particularly simple and convincing instance of this idea. 4.2. LEARNING FROM FEW CASES. Systems based on massive knowledge bases need to have effective mechanisms for coping with the issues of scale. In the previous section, for example, we suggested that a learning mechanism for linear threshold functions can address the issue of conflict resolution. In a similar fashion, we shall invoke learning as the solution to a variety of other questions also. It is, therefore, necessary to address the problem of how the learning process itself faces the issue of scalability.

The basic problem is fundamental and widely recognized. If there are n functions represented in the system, and each one can depend potentially on a large fraction of the n - 1 others, as a linear threshold (or some other) function, then there are potentially of the order of n parameters to set when learning any one of these functions. It is a remarkable fact that biological learning systems appear to be able to learn from relatively few examples, certainly many fewer than n for which reasonable estimates exceed 10^5 . Some mechanism needs to be present to enable very high dimensional systems to learn from a number of interactions with the world that are very small compared with this dimension. The most generic statistical considerations suggest that to learn a function of n variables, of the order of n examples are needed [Ehrenfencht et al. 1989]. In cognition, many fewer seem to be sufficient.

The theory that has been developed to shed light on this phenomenon is that of *attribute-efficient learnability*. The remarkable fact is that for certain function classes of n variables one can *prove* that certain learning algorithms converge to a good hypothesis after a number of examples that depends on n not linearly, but much more slowly, sometimes *logarithmically*.

The phenomenon of attribute-efficient learning was first demonstrated by Haussler [1988]. A striking embodiment of this idea followed in the form of Littlestone's Winnow algorithm for learning linear threshold functions [Littlestone 1988]. The algorithm is similar in form to the classical perceptron algorithm except that the updates to the weights are multiplicative rather than additive. Thus, algorithm WINNOW2 maintains as its hypothesis a linear threshold function $\sum w_i v_i > \Theta$, where Θ is fixed and each coefficient $w_i \ge 0$. On each input vector \underline{v} , the algorithm multiplies each coefficient w_i by $(1 + \beta)$ if $v_i = 1$, the example is positive and <u>v</u> does not satisfy the hypothesis. It multiples w_i by $1/(1 + \beta)$ if $v_i = 1$, the example is negative and \underline{v} does satisfy the inequality. In all the other cases, w_i is unchanged. The modification gives the algorithm the remarkable property that when learning a monotone k-variable Boolean disjunction over $\{u_1, \dots, u_n\}$ the number of examples needed for convergence, whether in the PAC or mistake-bounded sense, is upper bounded by $ck \log_2 n$, where c is a small constant [Littlestone 1988; 1989]. Thus, the sample complexity is linear in k, the number of relevant variables, and logarithmic in the number of irrelevant ones.

Littlestone's Theorem 9 [Littlestone 1988] adapted to the case where coefficients can be both positive and negative (his Example 6) has the following more general statement; For $X \subseteq \{0, 1\}^n$ suppose that, for the function $g: X \rightarrow \{0, 1\}$, there exist δ ($0 < \delta < 1$), $\nu_1, \nu_2, \cdots, \nu_n \ge 0$ and $\bar{\nu}_1, \bar{\nu}_2, \cdots, \bar{\nu}_n \ge 0$ such that for all $(u_1, \cdots, u_n) \in X$

$$\sum_{i=1}^{n} (\nu_{i}u_{i} + \bar{\nu}_{i}(1 - u_{i})) \ge 1 \quad \text{if} \quad g(u_{1}, \cdots, u_{n}) = 1$$

$$\sum_{i=1}^{n} (\nu_{i}u_{i} + \bar{\nu}_{i}(1 - u_{i})) \leq 1 - \delta \quad \text{if} \quad g(u_{1}, \cdots, u_{n}) = 0.$$

Then the algorithm WINNOW2 with $\Theta = 2n$ and $\beta = 1 + \delta/2$ applied to the variable set $(u_1, \dots, u_n, 1 - u_1, \dots, 1 - u_n)$ makes at most a fixed constant times the following number of mistakes:

$$n\,\delta^{-2}\Theta^{-1} + (\delta^{-1} + \delta^{-2}\ln\Theta)\sum_{i=1}^{n} (\nu_i + \bar{\nu}_i).$$
(4.2)

Here, Θ and β are parameters of the algorithm and δ , which quantifies the margin by which positive and negative examples are separated, is a parameter of the space of examples. For a monotone disjunction of k out of n variables, we can have $v_i = 1$ for the k variables in the disjunction, with all the other $v_i = 0$, and all $\bar{v}_i = 0$. Then, clearly $\delta = 1$. Hence, Eq. (4.2) becomes $O(k \log n)$. For linear inequalities with (positive and negative) integer coefficients whose magnitudes sum to Z, (4.2) becomes $O(Z^2 \log n)$ (e.g., Valiant [2000]). In all these cases the algorithm can be adapted so that it has similar bounds in the PAC model [Littlestone 1989].

For linear inequalities of the form (4.1), we see that the particular example given is equivalent to 1/4 $(u_7 + (1 - u_2) + (1 - u_5) + (1 - u_8)) \ge 1$ so that $\delta = 1/4$. In general, if there were k negative terms, then $\delta = 1/(k + 1)$. In order to make the margin larger, it is better to learn the negation of (4.1), namely $(1 - u_7) + u_2 + u_5 + u_8 \ge 1$ so that $\delta = 1$. The generalization of this to k terms would also give $\delta = 1$ and hence the O $(k \log n)$ bound.

It appears that some mechanism for attribute-efficiency is essential to any large scale learning system. The effectiveness of Winnow itself has been demonstrated in a variety of experiments. A striking example in the cognitive domain is offered in the work of Golding and Roth on spelling correction [Golding and Roth 1999]. Even in the presence of tens of thousands of variables, Winnow is able to learn accurately from few examples, sometimes fewer than 100. The perceptron algorithm also performs well if the examples have few nonzero attributes.

The question arises whether attribute-efficient learning is possible for more expressive knowledge representations. Recently some positive evidence for this has been found. Under a certain projection operation attribute-efficient learning algorithms can be composed to yield algorithms for more expressive representations that are still attribute-efficient. In Section 4.4, we shall discuss this further.

Finally, we note that attribute-efficient learning is closely related to the issue of *relevance*, which has been widely discussed in the statistics and AI literature. Conventionally, in statistical analysis, for example, one would expect to preprocess a database containing historical observations in order to identify the attributes that are relevant to the classification or action in question. One would then eliminate the irrelevant attributes, apply a learning algorithm to the database after the irrelevant features have been removed, and then apply the

and

learned hypothesis to new cases. There is, however, no evidence that biological systems have such an explicit preprocessing stage. Winnow achieves the same overall effect implicitly, without explicitly identifying which variables are irrelevant in a preprocessing stage. What it offers therefore seems novel and important. (Note that A. Beimel has observed that given an implicit method, such as Winnow, one can explicitly identify the relevant variables by a binary search technique that needs about $k \log n$ applications of the implicit method.)

4.3. LEARNING AND REASONING ABOUT MULTIPLE OBJECTS. It is well known that even in the propositional context both learning and reasoning need to be carefully formulated if exponential computational complexity is to be avoided. It is further known that if this is to be achieved in a multi-object rather than a propositional framework, then further orthogonal sources of computational intractability present themselves [Haussler 1989]. The main technical achievement of the neuroidal architecture is that it brings together learning, reasoning, and multi-objects domains within a computationally tractable formulation. The technique that is central to this is that of IQA – independently quantified arguments – which was defined in Section 2 and is analyzed extensively in Valiant [2000].

Most basically, if there are relations R_0, R_1, \dots, R_m represented as nodes then the circuit can represent, evaluate and learn rules of a form of which the following is an instance:

$$f(\exists y_1 \exists y_2 R_1(y_1, x_1, y_2), \forall y_3 \exists y_4 R_2(x_1, x_2, y_3, y_4)) \equiv R_0(x_1, x_2).$$
(4.3)

Here, f is a two argument function from an easily learned class, such as linear threshold functions. Such a rule is to be evaluated for a particular scene σ in an image. Evaluation means that the truth value of f is computed for a fixed binding θ : $\{x_1, x_2\} \rightarrow \{a_1, \dots, a_N\}$, and this value is assigned as the value of R_0 (x_1, x_2) . The quantification over the y variables refers to quantifications over the objects in one particular scene. Thus, for a scene where R_2 is defined for every 4-tuple of objects and x_1 , x_2 are fixed by θ , the quantified expression $\forall y_3 \exists y_4 R_2(x_1, x_2, y_3, y_4)$ is either true or false. Hence, for any one scene, any fixed function f acquires a Boolean value, as will also therefore $R_0(x_1, x_2)$. For this reason, if *relational* expressions such as 4.3 are to be learned then this can be done by an arbitrary *propositional* learning algorithm for f. Learning can then be viewed as taking place from a set of scenes taken randomly from a distribution D in the PAC sense. All the algorithmic techniques of propositional learning can then be applied directly. The only restriction needed is that the variables quantified over in the different arguments of f be disjoint. This is satisfied if the y variables that are quantified over in the respective R_i relations are disjoint for distinct values of *i*.

To ensure that both learning and evaluation are polynomial time in the relevant parameters, including particularly the number of object tokens N in the image, the only substantial restriction that is needed is that the relations all have arities bounded by a fixed constant α . We therefore need to assume that knowledge can be represented in terms of relations of bounded arity. This is discussed further in Valiant [2000].

4.4. LEARNING MORE COMPLEX FUNCTIONS AND STRATEGIES. While in our current state of knowledge there is compelling evidence that linear threshold

functions offer an attractive building block because of their learning properties, it remains unresolved whether a better choice exists. Clearly, the intention of our architecture is that each new function that is learned or programmed be expressible in terms of old ones already represented in the system. When discussing learning in general the crucial issue always is *how far removed* the new function can be allowed to be from the old ones already represented, without necessitating a learning capability that is computationally intractable. In other words, the issue is one of the *granularity* of the *relative* complexity of the successive functions that can be learned.

From what we have said, linear threshold functions offer a level of granularity that is computationally attractive. The two questions raised therefore are: (1) is this level of granularity sufficiently large to offer a convincing approach to building cognitive systems and (2) can this granularity be enlarged (i.e., to richer knowledge representations) while retaining attribute-efficient learning and error-resilience.

Even when a function class is expressible as linear inequalities, this representation may be impractically large if many compound features need to be created. For example, for Boolean variables $\{u_1, \dots, u_n\}$ each of the 2^{2^n} Boolean functions can be expressed as a linear disjunction of monomials over $\{u_1, \dots, u_n, \overline{u}_1, \dots, \overline{u}_n\}$ and hence as a linear inequality, but this requires a new variable for each potential monomial, and there are 3^n of these (since each u_i can be present positively, present negatively, or absent.)

Examples of function classes that can be expressed as linear inequalities over the n base variables are: disjunctions, conjunctions, and threshold-k functions. In the last case we would have

$$\sum_{i=1}^{n} w_i \ge k \; ,$$

where w_i is a variable over the reals that is given value 1 if $u_i = 1$, and zero otherwise. A further class that can be so expressed is that of 1-decision lists [Rivest 1987]. These test for a sequence of literals v_{i_1}, \dots, v_{i_m} , where $v_{i_i} \in \{u_1, \dots, v_{i_m}\}$ $\dots, u_n, \bar{u}_1, \dots, \bar{u}_n$. If the literal v_{i_i} is true, then the function is defined to be a constant c_i for $c_i \in \{0, 1\}$; otherwise, the next literal $v_{i_{i+1}}$ is tested. Decision lists can be expressed as linear inequalities over n variables w_1, \dots, w_n corresponding to u_1, \dots, u_n . The size of the coefficients grows exponentially, however, with n. In the special case that the c_i sequence alternates between 0 and 1 a small number of times, this growth is more limited [Valiant 1999]. For example, if c_i has value 0 for all except d of the m values of j, then the decision list can be expressed as a linear inequality with integer coefficients, where the magnitudes of the coefficients, and also their sum, is upper bounded by $(2m/d)^d$. If $d \ll m$ then this is much better than the general upper bound which is exponential in m. Also, we see that this inequality can be expressed so as to fit the requirements of expression 4.2 for Winnow. We then have $\delta = (2m/d)^{-d}$ and the sum of the magnitudes of the coefficients bounded by 1, so that the mistake bound is quadratic in $(2m/d)^d$ and logarithmic in n.

The question of characterizing the classes of linear inequalities that are learnable attribute-efficiently remains unsettled. The possibility that unrestricted 1-decision lists can be so learned has not been excluded.

Another question is whether attribute-efficient learning can be achieved by classes of functions beyond linear inequalities. A positive indication on this is provided by projection learning, which is a technique that extends the scope of attribute-efficient learnability. It allows algorithms that are attribute-efficient to be composed so as to obtain learning algorithms for more expressive representations that are still attribute-efficient. These representations can be viewed as restricted multilayer networks, as contrasted with the single layer linear threshold functions. Since Winnow is the paradigmatic attribute-efficient algorithm currently known, the present applications of projection learning are based on composing Winnow with itself.

The basic idea is that for Boolean variables $u_1, \dots u_n$ we define a set $J = \{\rho_1, \dots, \rho_r\}$ of *projections* that each map $\{0, 1\}^n$ to $\{0, 1\}$. For example, we could have r = 2n and for each literal $\ell \in \{u_1, \dots, u_n, \bar{u}_1, \dots, \bar{u}_n\}$ we have a projection ρ_ℓ defined on a vector \underline{u} of Boolean values as $\rho_\ell(\underline{u}) = 1$ iff $\ell = 1$ on \underline{u} . This is the class of single variable projections. An alternative class has $r = 2^k$ with each member of J being a conjunction $l_1 l_2 \dots l_k$ where $l_i \in \{u_i, \bar{u}_i\}$, that is, if k = 3 and $\rho = u_1 \bar{u}_2 \bar{u}_3$ then $\rho(\underline{u}) = 1$ iff $u_1 \bar{u}_2 \bar{u}_3 = 1$.

For each $\rho \in J$ we consider the function $f(\underline{u})\rho(\underline{u})$. Clearly this equals zero for \underline{u} such that $\rho(\underline{u}) = 0$, and it equals $f(\underline{u})$ otherwise. The hope is that for some of the restrictions ρ , the function $f(\underline{u})\rho(\underline{u})$ can be learned more accurately than can $f(\underline{u})$ directly, and that the $f(\underline{u})\rho(\underline{u})$ that are learned for the various ρ 's cover the whole domain $\{0, 1\}^n$ of $f(\underline{u})$.

Suppose that, for the various choices of $\rho \in J$, the function learned that approximates $f(\underline{u})$ on $\rho(\underline{u}) = 1$ is $f'_{\rho}(\underline{u})$. Then $\sum f'_{\rho}(\underline{u})\rho(\underline{u})$ is taken as the approximation of f that has been learned. The main result in Valiant [1999] states that if the $f(\underline{u})\rho(\underline{u})$ belong to a class that is learnable attribute-efficiently on the restricted domain $\{\underline{u} \mid \rho(\underline{u}) = 1\}$ by an algorithm A say, and if a disjunction can be learned attribute-efficiently by an algorithm B that shares certain specified properties with Winnow, then provided $f(\underline{u}) = \sum f_{\rho}(\underline{u})\rho(\underline{u})$, $f(\underline{u})$ can be learned attribute-efficiently, in the sense that the needed sample complexity depends linearly on the number of relevant ρ 's and the number of relevant variables in all the f_{ρ} , but only logarithmically on the total number r of projections and the total number n of variables.

The motivation of projection learning is to learn more effectively in cases in which a representation more complex than linear thresholds is needed. In any one application domain, we may have no a priori reason to believe that such a representation is necessary. What we expect to find is that projection learning will always yield at least as good results as simple Winnow, (but at the expense of requiring more examples) and may yield better results when linear representations are insufficient.

Projection learning can be viewed in the first instance as an intuitive approach to the problem of distinct varieties or contexts. For example, to develop a system that distinguishes apes from other animals it may be useful to have projections for the four varieties of apes, and in that way learn a recognizer for each one separately. An area in which complex representations may need to be learned is that of strategies and plans. Here production systems are widely believed to have useful expressivity [Newell and Simon 1972]. Khardon has shown that a rich class of these can be learned using decision list algorithms [Khardon 1988]. The dilemma here is that no attribute-efficient learning algorithm is known for decision lists, unless the number of alternations is small, as explained in the previous section. Hence, we may have to look at the flatter disjunctive structures that arise in projection learning to find representations that are learnable attribute-efficiently.

4.5. LEARNING AS AN APPROACH TO ROBUSTNESS. Learning appears to be an inevitable and central component to any large scale intelligent system because of its unique role in ensuring robustness and avoiding brittleness. Robustness has many aspects, and we believe that no alternative approach is currently known for addressing these on a broad front.

One aspect of the robustness problem is that different parts of a large system need to be compatible with each other. For example, in a large system consisting of many programmed rules some mechanism is needed to ensure that the basic predicates in terms of which the rules are expressed are used consistently across the system. This compatibility has to be maintained even for systems that are continuously modified and enlarged. This issue is addressed directly by PAC semantics, as previously discussed.

A second important aspect of robustness is resilience to errors. Systems will acquire knowledge both by inductive learning as well as by explicit programming. Errors and inconsistencies may occur in either kind of input, and mechanisms are needed for coping with these.

In the case of inductive learning, the issue of noise has been studied extensively. On the theoretical side a range of noise models have been considered, ranging from a malicious adversarial model [Kearns and Li 1993; Valiant 1985] to the more benign random classification noise model, where the only noise is in the classification of the examples and this is random [Angluin and Laird 1988]. At least for the more benign models, there are some powerful general techniques for making learning algorithms resilient to noise in some generality [Kearns 1993].

For the problem of learning linear separators, there exist theoretical results that show that there is no fundamental computational impediment to overcoming random classification noise [Blum et al. 1996; Cohen 1997]. Currently somewhat complex algorithms are needed to establish this rigorously. In practice, fortunately, natural algorithms such as the perceptron algorithm and Winnow behave well on natural data sets (e.g., Roth [1998]) even where there is no a priori reason to believe that linear thresholds should work at all, or evidence that the noise is generated by any one fixed process. This empirical evidence lends credence to the use of linear threshold algorithms for complex cognitive data sets.

The issue of coping with noise also arises when knowledge is acquired by programming. The view we take here is that we use the same PAC semantics for programmed rules as for inductively learned rules. Thus, the system would have high confidence in a rule that agreed with many examples. A programmed rule is therefore one which is treated as a working hypothesis, and discarded if evidence builds up against it. In addition to the opportunity to discard rules there is also one for refining them. Thus we may have programmed rules $P \Rightarrow R$ and $Q \Rightarrow \neg R$ as working hypotheses, but discover that they do not hold in all cases. In particular, it may happen that in some relatively rare cases both P and Q hold and therefore the rules contradict each other. It may be that after sufficiently many observations it is found that $P \land Q \Rightarrow \neg R$ is a reliable rule. In that case, we would refine $P \Rightarrow$ R to $P \land \neg Q \Rightarrow R$. The main point is that the dilemma that arises from two potentially contradictory rules is resolved by learning, even in the case that the original rules themselves are programmed rather than learned. The refined rule may be derived by any one of the several mechanisms we have discussed, as the reader may verify: $P \land \neg Q$ may be generated automatically as a compound feature, a linear conflict resolution mechanism may establish a priority, or, thirdly, projection learning may be invoked.

4.6. LEARNING AS AN APPROACH TO THE PROBLEM OF CONTEXT. It has been widely observed that rules that express useful commonsense knowledge often need qualification – they hold only in certain contexts. Thus, a rule $P \Rightarrow R$ or $P \equiv R$ may hold if context Q is true, but not necessarily otherwise. Frames in the sense of Minsky [1975] can be viewed as contexts that have a rich set of associated rules. The PAC semantics of such a rule is that on the subdomain in which Q holds, $P \Rightarrow R$ is the case, at least with high probability. In our architecture, the simplest view to take is that it can cope easily with a context Qif it has a node for recognizing whether an input satisfies Q. If there is a node in a circuit unit that recognizes Q, then a subcircuit that implements $P \land Q \Rightarrow R$ will implement exactly what is wanted, the rule $P \Rightarrow R$ applying in the subdomain in which Q holds. The question remains as to how domain sensitive knowledge can be learned. One answer is suggested immediately by projection learning; each concept R that is learned is also learned for the projections defined by every other concept Q that has been previously learned or programmed. Thus, if we have nodes for Q and R, then a complex set of conditions P that guarantee R in the context of Q can be learned from examples, for any Q and R. This would be, of course, computationally onerous unless the choice of Q is restricted somehow.

4.7. REASONING. Reasoning is concerned with arriving at a deduction about a combination of circumstances that has been encountered rarely before or not at all. It, therefore, complements inductive learning. In the neuroidal architecture, the rare or novel combination of circumstances is represented in an image, and the reasoning process is carried out, in the most basic case, by the circuit units that evaluate the contents of this image. In more complex cases, the results of the circuit evaluation will modify the contents of the image so that one or more further circuit evaluations can be applied to it in succession. For example, an instance of planning would be to run through several stages of the expected future sequence of events that would follow from a set of circumstances. Each circuit evaluation would update the contents of the image so as to describe the circumstances at the next stage in time.

What is the general nature of the reasoning process that can be expected to be achieved in one stage of circuit evaluation?

Much past effort has been put into seeing whether the various formalisms that have been suggested for reasoning in AI, at least those outside the probabilistic realm, can be formulated within predicate calculus. The general answer found to this question is in the affirmative – many of the various alternative formalisms, including those based on graph models, can be so expressed. Some of these alternative formalisms, and the necessary transformations are described by Russell and Norvig [1995].

Instead of reformulating this body of work so as to fit our architecture, we shall be content here to claim that a substantial part of it can be supported directly without change. In particular, we described an implementation of our architecture that can support generalized modus ponens on function free Horn clauses by performing an appropriate circuit evaluation. It follows that our architecture can do reasoning in all these frameworks whenever the translation results in Horn clauses. Further, the computational costs of this reasoning process can be quantified, as can the accuracy of the deductions made, as in described more fully in Valiant [2000].

In addition, our architecture has capabilities for reasoning beyond those provided by circuit evaluations alone. In particular, the circuits and the image units can be used together more proactively within the reasoning process. As previously mentioned, the evaluation of a circuit may result in various actions on the image, such as the addition of further objects, the addition of statements of further relationships, or the deletion of an object. Thus, a circuit may cause the execution of a step of a more complex reasoning strategy. In evaluating a scene, a circuit may add an object to the image to represent something already in the scene but at a later time, and add a relationship that expresses the future condition of the object. In this way, the circuits evaluate a representation of a likely scenario following on from the one currently depicted. In other words, we need circuits that are able to make useful modifications to the scene in a highly content dependent manner.

The upshot is that in addition to information about the external world, as implied in previous sections, the circuits will contain further information concerning the strategies that are to be used for the internal deliberations of the system. These deliberations will be seriously restricted, of course, by computational constraints, such as limits on the number N of objects that are allowed in a scene.

4.8. LEARNING AS AN APPROACH TO INCOMPLETE INFORMATION AND NONMONO-TONIC PHENOMENA. A fundamental issue in the design of any system that reasons is to define what the reasoning process is to accomplish in cases in which the available information is incomplete. Systems that attempt to describe the world declaratively appear to encounter methodological problems here. The difficulty is that such systems need to take a generic view of how to treat incomplete information – they need a uniform theory, such as circumscription or the closed world assumption that takes positions on how to resolve the unknown [Ginsberg 1989; McCarthy 1980; McDermott and Doyle 1980].

PAC semantics offers an advantage here – it resolves the unknown separately in each case by using information learned from past experience of cases where similar features to the case in hand were similarly unknown. Consider the paradigm of nonmonotonic reasoning exemplified by the widely discussed example of the bird called Tweety [Ginsberg 1989]. The system is invited to take positions on whether Tweety flies both before and after it is revealed to it that Tweety is, in fact, a penguin. The observation that motivates our approach here is that gates in circuits take values at *all* times. Suppose that in a brain, for example, there is a two-valued gate intended to recognize penguins. This would take value 1 say, when a penguin is identified, in the PAC sense, and 0 when what is seen is identified in the PAC sense as not being a penguin. However, this gate must also take some value in cases where conventionally one would say that the system has not taken a position on whether the object in question is a penguin or not. Suppose that in these cases the gate takes the value 0. Then, we could say that a 1 value means {penguin} and a 0 means {not penguin, undetermined}. In this sense, circuits represent internally the three cases {yes, no, undetermined} even if no explicit provision has been made for the third case. This means that learning and rule evaluations in the system are carried out with a semantics in which the undetermined value of each variable is represented. This is true both when a predicate is represented by a single gate having just two possible values (whether representing {yes, undetermined}/{no} or {yes}/{no, undetermined} by the two values) and also when there is a more explicit representation, say by a pair of binary gates whose four possible value combinations do distinguish among the three cases {ves}, {no}, and {undetermined}. Hence, once the system has reached stability in learning it will cope with instances of incomplete information exactly as it does with ones with complete information [Valiant 1994; 1995]. One could say further that in natural learning systems instances of incomplete information are the natural ones, and usually the only ones available.

Pursuing this example further, suppose that the system has a rule "bird = 1 and penguin = $0 \Rightarrow$ flies = 1". Suppose also that the gates "bird", "penguin", and "flies" all have value 0 if the truth of the corresponding predicate is undetermined in the system. Suppose further that this rule has been found to be consistent with most *natural cases* experienced by the system in accordance with the probability distribution D. It will then follow that for instances in which birdhood is confirmed but penguinhood is undetermined, it will be a reasonable conclusion that flies will be true and that the corresponding gate should indeed have value 1 to indicate that. This will be a valid inference since the same was true for past cases in the PAC sense. Thus, the undefined is resolved here by learning from the distribution D, as seen through the feature set of the system. In this example, D captures the fact that in the particular source of examples to which this system was exposed, in the majority of cases where a bird was identified but nothing was said about penguinhood, the bird did indeed fly.

In artificial systems, of course, the undefined value * may be treated more explicitly. For example, as previously mentioned, gates may take three values $\{0, 1, *\}$, or, closer to the spirit of our architecture, the three values may be encoded by a pair of binary gates.

Defaults may be viewed from this same perspective. Ignorance of the value of a predicate is rightly interpreted as not relevant to the value of another predicate if in natural past cases of ignorance of the former, a certain consequence was true with high probability for the latter. We regard defaults as rules where certain predicates are assumed to have the undetermined value in the precondition. Their validity arises from the fact that they had proved accurate in the past, or that they could be deduced from those that had. Further examples are given in Roth [1995] and Valiant [1994; 1995].

5. Systems Design

We shall be concerned here with the issue of constructing systems using the neuroidal architecture that have superior capabilities over existing systems in some useful task. Considerable experimentation in this direction will be needed before the potential and limitations of the general approach can be evaluated.

It has been asked whether in our architecture one still has to design systems, exactly as one would in a programmed approach to artificial intelligence systems, and all that is offered are some mechanisms for the automatic adjustment of weights. While in some sense the answer to this is yes, the important point is that the additional learning capability appears to offer a new level of functionality. Above all, it offers a robust unified approach to infusing knowledge into a system in bulk. Further, it offers a unified methodology for dealing with a broad range of issues, including conflict resolution, nonmonotonic phenomena, incomplete information, and the problem of context among others.

Because of the learning-intensive nature of the architecture, it is an inherent characteristic of it that any discussion of the design of a system cannot be divorced from that of the nature of the training data. Indeed, the primary impediment to experimentation at present is the scarcity of suitable training data. Natural language and vision are potentially sources of ample data. The characteristics that make a data set appropriate are that the data is labelled, implicitly or explicitly, that the knowledge it contains is self-contained and that the information can be extracted using known techniques. Thus, the system design has to bypass unsolved problems in natural language processing or computer vision.

There is already ample evidence that the basic learning mechanism used by the architecture is appropriate. It has been demonstrated on a wide range of natural language tasks, and one from computer vision, that attribute-efficient learning of linear separators is effective on existing data sets [Golding and Roth 1999; Roth 1998; 2000]. This basic learning mechanism can be contrasted with alternative approaches: learning with hidden units as in back-propagation, reinforcement learning, learning using feature sets that are hand-crafted separately for each application, and learning with heuristics for constructing new features. Attribute-efficient learning with linear separators appears to offer a successful unified and mechanized approach to the inductive component of the problem.

Conventional design principles offer a start in designing systems in this architecture. In a programmed system, one expects to construct various modules for various functionalities, and have these interface with each other. In particular, one may have a hierarchy of functions, lower-level modules processing the inputs directly, and higher level modules processing the outputs of the lower level ones. Thus, an acyclic graph of circuit units, in which some low level modules mimic the modules that would be used in, say, a language system, is one possible starting point.

There are also some design choices that are unique to the architecture. For creating new compound features, there are various possibilities. Because of the centrality of attribute-efficient learning algorithms, an obvious method is to generate large numbers of combinations of attributes in a systematic simple way. Any relevant ones so discovered will be valuable, and the irrelevant ones will do little harm. One approach is to generate for any variable set the set of



FIG. 1. Block diagram of a simplified example of a potential neuroidal system. The thesaurus is an example of an available programmed resource. Layers 2–5 consist of circuit units that contain all the learned knowledge, and may be trained separately in succession. The text categorizer is specific to the application, while the other circuit units may store more generic knowledge that might be shared by many applications.

conjunctions of all pairs of them. Another choice is to create conjunctions of just those pairs that occur with high correlation. More generally one can generate some set of *polynomially generable* combinations [Valiant 1985]. The intention is that large numbers of variables, even if most are irrelevant, will not degrade performance.

Figure 1 gives a block diagram of a simple example of a system that extracts knowledge from natural languages text inputs. The first layer performs some shallow parsing, determining, for example, subject-object classifications. The second layer extracts the atomic relationships found. The goal is to extract whatever knowledge can be reliably extracted, ignoring any parts of the text that cannot be analysed with confidence. The third layer contains rules among relationships, and hence may already capture complex facts about the world. The second and third layers are both connected to an image unit that records the relations that have been established as holding for the text being analysed. Where accurate programmed knowledge is available, such as dictionaries and thesauri, these would be used where possible, for example, to recognize synonyms or the "is-a" relation.

The construction of such a system would then involve designing the feature set and interfaces among the modules, and providing the necessary training data for each module.

We note that Roth's evolving SNOW system incorporates some of the mechanisms described in this paper, and this is discussed further in Khardon et al. [1999].

In order to construct significant systems along these lines, corpora suitably annotated with the appropriate levels of knowledge may have to be created. In other words, substantial infrastructure will need to be manufactured to provide the teaching materials for these systems. This would compensate for any shortcoming in the range of preprogrammed features that we are able to devise. Although large amounts of reliable knowledge are already available in linguistic format, there are many obstacles to preparing or selecting useful linguistic teaching materials. Much commonsense knowledge is nowhere encoded, and much text produced by humans contains linguistic constructs that are currently difficult for machines to analyze. These facts make the preparation of the teaching materials challenging, but, we believe, not insurmountable.

6. Conclusion

We have described a series of arguments that suggest that many of the problems that have been identified as obstacles to artificial intelligence systems at a conceptual level, can be solved in principle if one gives inductive learning a suitable central role. In this respect the proposed architecture differs from other general approaches to cognitive architectures that have been described, such as Anderson [1983]; Newell [1990], and Newell and Simon [1972], in which inductive learning plays a much smaller role.

Central to the proposed architecture is the notion of an image unit that relates to the concepts of short-term and working memory in psychology and cognitive science. Its computational role can perhaps be best understood by considering the notion of registers in computer systems. Registers are the most active storage elements in a digital computer – it is there that new values are evaluated. However, they make up only a negligible fraction of the overall storage capacity. Their constrained and localized nature appears to be a consequence of the fact that the circuits that are needed to act on them are complex and require a substantial investment. We believe that in an architecture for performing cognitive computations the active storage area that represents general relational information needs to be constrained and localized for broadly similar reasons, namely the complexity of the circuits that need to act on this area. It is this active storage area that we seek to capture with the notion of an image unit.

Also central to the development is the use of PAC semantics. Its role can be viewed in terms of a modification of the Turing Test that it suggests. Turing's criterion for whether a machine could think was that the performance of the machine should be indistinguishable in a specified sense from that of a human, to an interrogator communicating via a teleprinter [Turing 1950]. The significance of this informally stated criterion is that it is a purely *behavioral* one. What PAC semantics offers is a precise way of formulating such behavioral criteria. In particular it insists that both the task being learned, and the distribution of inputs over which learning is performed and performance measured, need to be identified. Any pairing of a specific task and a specific distribution then defines a different instance of this modified Turing Test.

The tasks in which we are interested here are those involving a large amount of knowledge that has not been systematized into an exact science. A possible limited area in which one might hope to test the performance of a system at such a task is that of intelligent word processing. One can imagine a word processor that not only detects misspelled words, but does a succession of more and more intelligent tasks, such as suggesting alternative words and phrasing, or noting confusions or inconsistencies, much as a teacher might when correcting an essay. There appears to be a continuum of tasks of increasing difficulty in this area. Systems could invoke more and more world knowledge in their suggestions, and their perceived performance would increase correspondingly. One can imagine experiments in which commentary to writers is provided variously by humans and machines, and the writers' task is to distinguish which one was the source. Such an instance of a modified Turing Test would therefore refer to a concrete real world distribution of cases generated, for example, by twelve-year-old students, and created independently of the context of the test. Each such distribution would define a different task and therefore a different test. For an empirical validation of our architecture, one would need to show that systems can be built that pass such specific tests of ever higher levels of difficulty.

ACKNOWLEDGMENTS. I am grateful to Roni Khardon, Dan Roth, and Rocco Servedio and several referees for their helpful comments on various versions of this paper.

REFERENCES

- ANDERSON, J. R. 1983. *The Architecture of Cognition*. Harvard University Press, Cambridge, Mass. ANGLUIN, D., AND LAIRD, P. 1988. Learning from noisy examples. *Mach. Learn.* 2, 4, 343–370.
- BLUM, A., FRIEZE, A., KANNAN, R., AND VEMPALA, S. 1996. A polynomial time algorithm for learning noisy linear threshold functions. In *Proceedings of the 37th IEEE Symposium on Theory of Computing*. IEEE Computer Society Press, Los Alamitos, Calif., pp. 330–338.
- CESA-BIANCHI, N., FREUND, Y., HAUSSLER, D., SCHAPIRE, R., AND WARMUTH, M. 1993. How to use expert advice. In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing* (San Diego, Calif., May 16–18). ACM, New York, pp. 382–392.
- COHEN, E. 1997. Learning noisy perceptrons by a perceptron in polynomial time. In *Proceedings of the 38th IEEE Symposium on Foundation of Computer Science*. IEEE Computer Society Press, Los Alamitos, Calif., pp. 514–523.
- EHRENFEUCHT, A., HAUSSLER, D., KEARNS, M., AND VALIANT, L. 1989. A general lower bound on the number of examples needed for learning. *Inf. Comput.* 82, 3, 247–266.
- GINSBERG, M. L. 1989. Readings in Nonmonotonic Reasoning. Morgan-Kaufmann, Los Altos, Calif.
- GOLDING, A. R., AND ROTH, D. 1999. A Winnow-based approach to context-sensitive spelling correction. *Mach. Learn.* 34, 107–130.
- HAUSSLER, D. 1988. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artif. Int.* 36, 177–221.
- HAUSSLER, D. 1989. Learning conjunctive concepts in structural domains. Mach. Learn. 4, 7-40.
- KEARNS, M. 1993. Efficient noise-tolerant learning from statistical queries. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing* (San Diego, Calif., May 16–18). ACM, New York, pp. 392–401.
- KEARNS, M., AND LI, M. 1993. Learning in the presence of malicious errors. *SIAM J. Comput.* 22, 4, 807–837.
- KEARNS, M. J., AND SCHAPIRE, R. E. 1994. Efficient distribution-free probabilistic concepts. J. Comput. Syst. Sci. 48, 3, 464.
- KHARDON, R. 1996. Learning to take actions. In *Proceedings of the National Conference on Artificial Intelligence*. AAAI, Reston, Va., pp. 787–792.
- KHARDON, R., ROTH, D., AND VALIANT, L. G. 1999. Relational learning for NLP using linear threshold elements. In *Proceedings of the International Joint Conferences on Artificial Intelligence* (Stockholm, Sweden, July). Morgan-Kaufmann, San Francisco, Calif., pp. 911–917.
- KNUTH, D. 1973. The Art of Computer Programming, volume 1. Addison-Wesley, Reading, Mass.
- LITTLESTONE, N. 1988. Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm. *Mach. Learn.* 2, 285–318.

LITTLESTONE, N. 1989. From on-line to batch learning. In *Proceedings of the 2nd Workshop on Computational Learning Theory*. Morgan-Kaufmann, San Mateo, Calif., pp. 269–284.

MCCARTHY, J. 1980. Circumscription-A form of non-monotonic reasoning. Artif. Int. 13, 27-39.

MCCARTHY, J., AND HAYES, P. J. 1969. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence*, vol. 4. D. Michie, ed. American Elsevier, New York.

MCDERMOTT, D., AND DOYLE, J. 1980. Nonmonotonic logic I. Artif. Int. 13, 1, 41-72.

MILLER, G. A. 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psych. Rev.* 63, 81–97.

- MINSKY, M. 1975. A framework for representing knowledge. In *The Psychology of Computer Vision*, P. H. Winston, ed. McGraw-Hill, New York.
- NEWELL, A. 1990. Unified Theories of Cognition. Harvard University Press, Cambridge, Mass.
- NEWELL, A., AND SIMON, H. A. 1972. *Human Problem Solving*. Prentice-Hall, Englewood Cliffs, N.J.
- PEARL, J. 1988. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan-Kaufmann, Los Altos, Calif.
- PITT, L., AND VALIANT, L. G. 1988. Computational limits on learning from examples. J. ACM 35, 965–984.

RIVEST, R. L. 1987. Learning decision lists. Mach. Learn. 2, 3, 229-246.

- RIVEST, R. L., AND SLOAN, R. 1994. A formal model of hierarchical concept learning. *Inf. Comput.* 114, 1, 88–114.
- ROTH, D. 1995. Learning to reason: The non-monotonic case. In *Proceedings of the International Joint Conference on Artificial Intelligence*. Morgan-Kaufmann, San Francisco, Calif., pp. 1178–118.
- ROTH, D. 1998. Learning to resolve natural language ambiguities: A unified approach. In *Proceedings of the National Conference on Artificial Intelligence*, AAAI, Reston, Va., pp. 806–813.
- ROTH, D., YANG, M.-H., AND AHUJA, N. 2000. A SNOW-based face detector. NIPS, To appear.
- RUSSELL, S., AND NORVIG, P. 1995. Artificial Intelligence. Prentice-Hall, Upper Saddle River, N.J.
- TURING, A. M. 1950. Computing machinery and intelligence. Mind 59, 433–460. (Reprinted in Collected Works of A. M. Turing: Mechanical Intelligence, (D.C. Ince, ed.), North-Holland, 1992).
- ULLMAN, J. D. 1989. *Principles of Database and Knowledgebase Systems*. Computer Science Press, Rockville, Md.
- VALIANT, L. G. 1984. A theory of the learnable. Commun. ACM 27, 11 (Nov.), 1134-1142.
- VALIANT, L. G. 1985. Learning disjunctions of conjunctions. In *Proceedings of the International Joint Conference on Artificial Intelligence* (Los Angeles, Calif.). Morgan-Kaufmann, San Francisco, Calif.
- VALIANT, L. G. 1994. Circuits of the Mind. Oxford University Press, Oxford, England.
- VALIANT, L. G. 1995. Rationality. In Proceedings of the 8th Annual Conference on Computational Learning Theory (Santa Cruz, Calif., July 5–8). ACM, New York, pp. 3–14.
- VALIANT, L. G. 1999. Projection learning. Mach. Learn. 37, 2, 115-130.
- VALIANT, L. G. 2000. Robust logics. Artif. Int. 117, 231-253.

Received June 1998; revised January 2000; accepted february 2000