1 Introduction

Bias is *everything* that influences the choice of one hypothesis over infinitely many others. As Utgoff states it,

> given a set of training instances, bias is the set of all factors that collectively influence hypothesis selection. These factors include the definition of the space of hypotheses and definition of the algorithm that searches the space of concept descriptions. [1986]

Simply put, machine learning is biased search. Yet we do not understand how bias should operate, and a generative theory for bias does not exist.

One conclusion drawn at last year's AAAI conference is that we must work towards a deeper understanding of the functional components in bias so that we reason about and choose an appropriate bias for any particular application [Buntine 1990]. Clearly, any system which learns in our real world must be strongly biased. It is intractable to search any reasonably sized real world description space, it is improbable that all teach examples contain only and precisely the task relevant features, and one search strategy will not suffice to learn every task. People depend on a sophisticated dynamic attention mechanism. When playing chess, for instance, they consider far fewer moves than similarly competent computer players. Furthermore, different moves are considered at each turn. In order to be successful, artificial learners must imitate humans in changing their partiality over time.

As yet, a formal predictive theory for bias is too much to ask for. Not that we do not understand bias in an artificial setting. Whenever a new problem domain is encountered, researchers commonly develop a new independent algorithm manually, rather than modifying the old one to learn in both cases. Second, we do not have an accurate model of human bias. Our innate characterization cannot be isolated from our learned components, so we do not have a complete learning theory. Third, we do not have concrete examples of a "perfect" bias — even humans are fallible. Furthermore, there is no general

measure for "best" in human learners. The science of bias is premature and, as yet even a taxonomic classification is not agreed upon, let alone a theory that supplies design rules for creating learners.

This paper develops a descriptive framework of the kind that must precede a generative theory of bias. First, it establishes a set of criteria, or relative measures, for comparing inductive learning systems and their biases. Then, a framework is derived by separating bias into two components: static and dynamic. Dynamic bias is that part of the learner which changes while a concept is being discovered. In order to function within our world, systems must have dynamic bias, and this is where research must concentrate. Examples from Mitchell's version space learning system [Mitchell 1977] dominate the third section of the paper, and illustrate each distinguished form of bias. STABB [Utgoff 1986] adds a type of dynamic bias, enabling it to increase the description language of version space. The final section elaborates dynamic bias by explaining ETAR [Heise 1989, Heise & MacDonald 1989a, Heise & MacDonald 1989b], a complete system which learns robot assembly tasks as it operates in the real world. ETAR uses a focus mechanism to determine important information from an example and to significantly reduce the concept choices during generalization. It is able to learn tasks such as stacking blocks and sorting or selecting objects (all in the real world) from sequences of robot motions recorded as a user physically leads the robot to complete the task. Dynamic bias makes learning possible. All real world learners would benefit from the ability to shift attention dynamically.

2 What is a Good Bias?

Comparison between learners is difficult. Many "local" relative measures are used, such as the inability to learn particular concepts or the speed in learning common concepts. This paper uses three evaluation criteria to compare

inductive learners and their biases:[1]

Computational complexity measures the time and space required to compute an hypothesis that is consistent with a set of observations [Russell & Grosof 1990]. Since every learner searches through its hypothesis space either directly or implicitly, complexity is chiefly related to the size of the hypothesis space. It also depends on factors such as the organization of the hypothesis space, how often the hypotheses must be generated, and the traversing mechanism. Practical algorithms require time and space to be polynomial in all the parameters that influence the outcome (such as the number of features describing an example, the number of examples, and the size of the learned concept description) [Valiant 1984].

Example complexity measures the number of observations required to induce an appropriate hypothesis. The human teacher or user is often responsible for generating the examples. As the number of necessary examples increases, so does the difficulty of the teacher's job. Certainly any system with the ambition of helping users[2] must keep the required examples at a minimum. To learn a goal from fewer examples, a system must make a larger inductive "leap" or generalization, and is a better learner than a system which requires additional examples to derive the same conclusion. Again, one aim for the number of required examples is that it be polynomial in all parameters [Shvaytser 1990], though we expect that it should be better. Additional restrictions on the examples, such as the need for a specific order (*e.g.* [Andreae 1984]), may also be considered as increasing the example complexity.

[1] [Angluin & Smith 1983] uses various efficiency measures (alternative measures of complexity, similar to those in this paper) and an inferability criterion (corresponding to this paper's learnability criterion).

[2] Or learning in a dangerous world, where mistakes have dire consequences, and every example is potentially destructive to the learner.

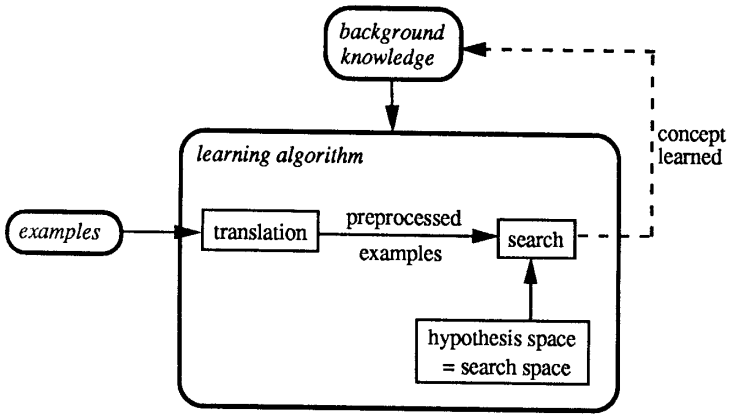

Figure 1: Components of an inductive learner

Learnability is the last, and most difficult, of the comparison measures. In simple terms, it is the number of concepts that a system can learn. But, when has learning occurred, is it complete, and has the correct concept been identified? We evaluate human learning with performance tests, conversation, and analysis of further examples. Artificial learners have not been subject to the same assessment, rather four (theoretical) models are used: identification in the limit [Angluin & Smith 1983, Gold 1967], probably approximately correct (PAC) learning [Valiant 1984], frequently approximately correct (FAC) learning [Dietterich 1989], and exact match.

These three criteria — computational complexity, example complexity, and learnability — are generally trade–offs. For instance, when few examples are available the computational complexity may increase, for equivalent learnability.

3 A Framework for Bias

This section separates the bias functionality of a typical learner model to distinguish two forms: static and dynamic. An inductive learner has three

parts, as shown in figure 1:

Examples. This includes all that is input to the learner from the external world. Generally, the examples are explicitly supplied by the user or observed by the learner's sensory mechanism. Each example contains features from an example vocabulary. These features are bound together by the example formalism to make up a complete example. *E.g.* if the features are "diamond" and "10" with the conjunction formalism, then the example is the 10 of diamonds ("10 $\wedge$ diamond").

Background knowledge. This information is present in the system before learning begins. It consists of a vocabulary and formalism, as well as a list of relations on the vocabulary. An example of a relation might be that "diamond" is an instance of "suit". This knowledge may be used to generate the hypothesis space for the learner, and has been termed *knowledge bias* [Hirsh 1990].

Learning algorithm. This is the heart of the learner and contains three components:

1. Hypothesis space: a (possibly infinite) set of all potential concepts which the system may learn. It is formed by taking all concept vocabulary primitives (conceptual bias [Genesereth & Nilsson 1987], concept language bias [Russell & Grosof 1990]) and combining them in all possible ways according to the allowed hypothesis formalism (logical bias [Genesereth & Nilsson 1987], or composition bias [MacDonald & Witten 1988]). Preference ordering bias [Genesereth & Nilsson 1987] may force a fixed order on the elements in the hypothesis space.

2. Translation: a process which transforms the input examples into a form suitable for the search algorithm

3. Search: the technique for traversing the hypothesis space.

As in Utgoff's definition (see the introduction), it is precisely the background knowledge and the learning algorithm that are the bias and determine the concept that is learned. Most researchers explicitly code their inductive learners, and whenever the system is to work in a different domain, the background knowledge and/or the learning algorithm are re–designed. This is the first category of bias. A bias is *static* if it does not change while the system attempts to learn a concept. A learner is entirely static if its behaviour, apart from the input examples, is determined before viewing the examples. In other words, in a static system one can always predict the search space, the search algorithm, and the features present in the preprocessed examples, regardless of the actual input examples. The input examples determine the path and termination of the search. Many of today's artificial learners possess only static bias.

In contrast, *dynamic* bias changes the learning algorithm while a concept is being acquired, so the system's behaviour cannot be predicted until the examples are known. The example transformation, the search space, and the search strategy may change as examples are processed. Figure 2 shows how dynamic bias extends the inductive learner model. Two dynamic biasing techniques can be identified, and both are apparent in the machine learning literature. Dynamic bias can either expand (magnify) or shrink (focus) what is considered by the learner.

Focus. The attention of the learner may be directed to specific features in the feature space, structures within the hypotheses, or strategies of the potential search algorithms. The most common use is focusing in the search space, where certain hypotheses are preferred (for example ETAR

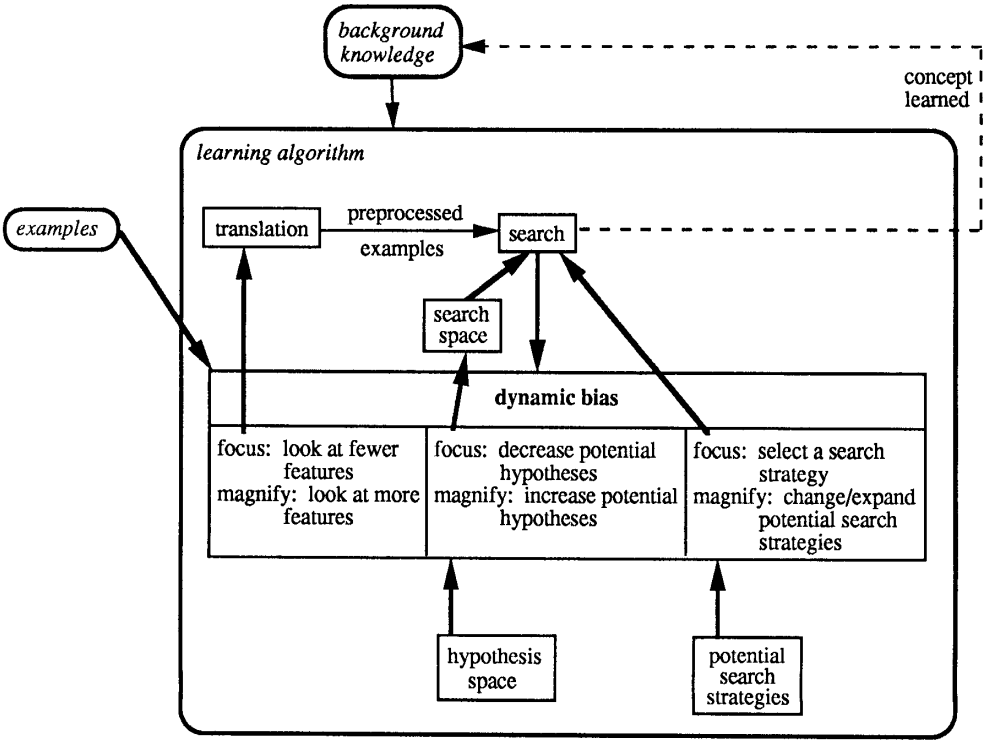

Figure 2: Dynamic bias added to an inductive learner

prefers to form loops and branches at the points in a task where nearby objects' roles change), effectively causing the space to shrink with each example presentation. The aim is to decrease the computational and example complexities, while maintaining the same learnability as the corresponding static system.

Magnify. A search space may not contain a description for a given example set. In this case the bias is too strong, and the components of the learning algorithm must be expanded. Most often knowledge of the underlying structure and the domain are needed, enabling the system to add new concepts to the hypothesis space, modify the search algorithm, or look for more information in the examples. For instance, additional internal nodes might be added to an object type hierarchy, such as "polygon"

	static	magnify	focus	focus + magnify
Initial measures				
knowledge	29	29	29	29
learnability	12	12 +	12	12 +
Learning "circle" (in hierarchy)				
computational complexity	29	29	7 ↓	7 ↓
example complexity	22.5	22.5	2 ↓	2 ↓
concept learned	yes	yes	yes	yes
Learning "polygon" (not in hierarchy)				
computational complexity	29	33 ↑	7 ↓	11 ↓
example complexity	22.5	45 ↑	2 ↓	3 ↓
concept learned	no	yes/no ↑	no	yes ↑

Figure 4: Comparison of various biases under version space.

with respect to the search space, obtaining the main result in figure 4 — a comparative table of complexities and learnability for the biases.

First, we examine the three measures for static bias in the version space system. In figure 3, the vocabulary bias is a fixed set of size and shape features while the hypothesis formalism is ordered pairs of these. Concepts are related to each other by a general–to–specific partial order. The size of the hypothesis space, simply measured as the sum of the number of concepts plus the number of relations, is a good indication of the approximate computational complexity of the learner. There are initially 12 concepts and 17 relations, giving a total of 29. The search algorithm ensures that each of the concepts in the hierarchy may be learned if proper examples are given. Since no additional concepts can be learned, the learnability is 12. We will limit the example complexity to indicate the minimum number of examples required to isolate a concept and the difficulty with which these examples would be chosen. A simple formula divides the minimum number of examples by the probability of choosing a feasible set of such examples. In version space, two correctly chosen positive examples are always necessary and sufficient in determining the S set, as long

as the desired concept is not one of the leaves.[3] Moreover, each parent of the concept must be eliminated for the G set to meet S. Sometimes one negative example will eliminate more than one parent. The minimum number required is then the size of a smallest set (disjoint from S) so that each parent covers at least one member. Thus, for a tree–structured hierarchy, version space can always learn with three (two positive and one negative) properly chosen examples. The difficulty of choosing proper concepts is a monotonically increasing function dependent on the total number of possible examples and the complexity of the hierarchy. We simply calculate this as the inverse of the probability of chosing the correct examples, assuming that no example may be repeated. For figure 3, we begin with one positive example (2 possibilities out of 6), followed by either another positive example (1 in 5) and then a negative (4 in 4), or a negative example (4 in 5) and then a positive (1 in 4) to yield $\frac{2}{6}[\frac{1}{5} \cdot \frac{4}{4} + \frac{4}{5} \cdot \frac{1}{4}] = \frac{2}{15}$. Hence, the example complexity is $3 \cdot \frac{15}{2} = 22.5$.

STABB extends version space with magnification to increase learnability by expanding the applicable vocabulary, with a small cost in computational complexity. It assumes the initial system bias is strong so that the hypothesis space is always overly restricted. When it is apparent that a desired concept cannot be learned, built–in rules augment the hypothesis space, thereby weakening the bias and hopefully including the goal concept. The "magnify" column of figure 4 summarizes the complexities for STABB. When a concept is already present in the hierarchy (e.g. "circle"), STABB behaves identically to version space.

Traditional version space cannot learn the concept "polygon" since two positive examples ("large square" and "small triangle") cause the S set to cross the G set if a negative example ("small circle") has been seen. At this

[3] When the concept is one of the leaves, i.e. an explicit example, only one positive example exists.

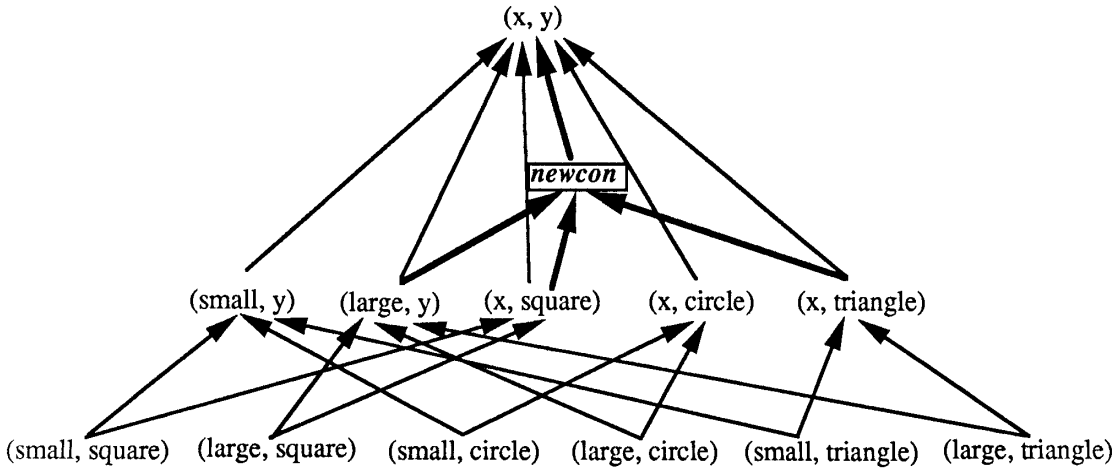

Figure 5: STABB constructs new concept

point, STABB fires its heuristics to discover the gap in the knowledge hierarchy and add a new concept above "square" and "triangle". Unfortunately, many of STABB's rules are domain specific and hard to transfer. Generally, STABB would execute the domain specific rules first, and if unsuccessful, then a domain independent least disjunction heuristic adds concepts into the hierarchy. In this paper we use only the latter. Adding least disjunctions may result in problems, since the heuristic assumes that the unseen examples are positive and often builds a description that is too encompassing. The search space can become cluttered with useless knowledge, so the computational complexity increases with no gain in learnability. For example, consider three examples: + small square, − small circle, + large triangle. STABB generates a new concept, *newcon*, shown in figure 5. The system incorrectly assumes this is the concept it is searching for, though it contains an unnecessary condition (large, y). If (large, circle) was stated negative as the third example, then STABB would build the new polygon concept as (x, square) ∨ (x, triangle). When properly done, the hypothesis space increases by four (one concept plus three relations) and learnability increases. The example complexity increases

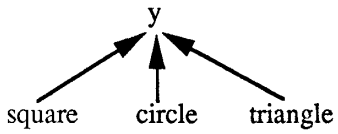

Figure 6: Focused version space hierarchy

to 45 since an additional example was required to place the polygon concept.

Since the color feature is not important when learning about shapes, a teacher might only give examples with the shape feature. In this case, a suitable focus bias would quickly reduce the hierarchy to that of figure 6. Now the computational complexity is only 7, while learnability remains the same. Furthermore, any two distinct examples enable any concept to be learned, hence the example complexity is 2. Focused version space, like the static one, still cannot learn concepts not in the original hierarchy.

A combination of focus and magnify provides an excellent bias. This learner can generate and converge on the polygon concept with any set of three distinct examples. Computational complexity is reduced (11). Example complexity is also lower (3), while the learnability increases.

5 Dynamic Attention in ETAR

This section examines the real world robot learner that inspired this paper. ETAR (Example–based Task Acquisition in Robots) reads examples in the form of sampled numerical joint data, as the user controls a six–axis robot arm to demonstrate the task, and constructs a simple assembly task procedure. The main problem was to reduce the potential search space when looking for relevant features and program structures, and this was achieved by focusing, ETAR's primary form of bias.

ETAR explicitly depends on the examples to dynamically and continuously limit the size of the search space. This is essential since ETAR operates

in a real robot domain, hence its search space is enormous. Furthermore, its hypotheses must describe space and time dependencies so that loops, branches, sequences, and object relations are formed. ETAR's focus of attention mechanism limits the relevant vocabulary and chooses the possible structures compatible with each observation. Thus, the focus of attention biases both the feature space and the hypothesis space.

ETAR's initial bias is a knowledge base of information about potential objects in its environment. Each new example forces the system to choose a subset of the applicable features. In this way the focus of attention mechanism, currently a fixed locality constraint, highlights the important features in an example. As user traces are recorded, so are the objects that are close to the robot. The features of these nearby objects are crucial to generalizing each step of the task. When more examples of the same task are noted, the vocabulary is further circumscribed by the intersection of the features from corresponding nearby objects in each example. Suppose that part of the task is to pick up an object. The first example grasps a cylinder, while a second grasps a cuboid. Clearly, *radius* is not a suitable attribute on which to base the generalization since it is not a descriptor of cuboids. On the otherhand, *height* should be considered.

The focus of attention also limits the structure of the learned tasks. ETAR allows three control structures, all of whose production is governed by focusing:

Sequencing. Each task example is recorded as a joint sequence while the user moves the robot, hence, sequencing is explicitly presented. However, many steps in this sequence may be useless or may contain unnecessary precision. For instance, when the robot is moved to grasp a block, the exact path travelled does not matter during the few seconds when it moves through uncluttered space before approaching the block. Focusing distinguishes the important points along the path (where the robot

is close to an object and where the user slows down). The remaining points are partitioned into a sequence of primitive motions (*translate, rotate, translate-rotate, moveto,* and *grip*) augmented with the appropriate position arguments. These motions are grouped according to the objects in the robot's focus at the time. Each motion group is called a focus of attention group, and may be viewed as a subtask. The result is a number of traces of the task, which ETAR goes on to merge into one procedural description.

Loop. The search for loops is limited to repetition in the primitive motions about points in the task where the focal objects play the same role. The role is determined by the user who describes a robot's task in a functional format. Suppose one task example is *stack(block1, block2)* and another is *stack(block3, block4)*, then *block1* and *block3* have the same role (as do *block2* and *block4*). Objects not explicitly specified have the same role as each other — extra objects, non–essential to the task.

The first step in generating a loop is to search for a focus of attention group which contains a motion which begins or stops affecting other objects during its task, *i.e.* a *grip* motion. Once determined, this group is the head of the potential loop and the search continues for another group in which the same action occurs on another object with the same role. The groups between these two form the body of the loop. Search continues until the entire loop is generated.

Branch. All the examples of a task are matched together to form a generalized task procedure. Matching occurs between corresponding steps (primitive motions) in each example. Branches are a side–effect; the result of an impossible match. Special care is taken so that focus of attention groups only match to other groups that contain objects with the same role.

Thus, matching (and branching) is determined by one pass through each example.

Sequences, loops, and branches form the outer structure of a task. The inner structure is induced after the loop and branch conditions, and the primitive motion arguments are learned. Loop and branch conditionals are found by a simple discrimination operation[4] on the features and values of the objects in the focus of attention. Since primitive motion arguments are numerical positions, ETAR introduces another formalism: function composition of addition, multiplication, subtraction, and division. Every primitive motion argument is discovered by inducing a composite numerical expression of the features in the focus of attention. Focusing controls the entire structure of the learned task, reducing the example complexity, and drastically decreasing the computational complexity by cutting down the potential branches and loops that may be formed to make the final procedure. Learnability is somewhat reduced, but the simple focus on nearby objects seems powerful since — in a static environment — it brings all manipulatable objects to the learner's attention, as the arm moves.

ETAR demonstrates the potential of, and need for, focus as a form of dynamic bias. The focusing mechanism is effective but simple, and suggests several important extensions for investigation in table 1. These factors and others will ultimately combine to form a theory of focus.

Note that the model of figure 2 allows for both the input and the learning algorithm to modify the dynamic bias. It is interesting that one theory of human attention allows both unexpected or interesting input events to interrupt and redirect the sensory system, and allows higher "executive" level processes to shift attention so that learning and/or performance are improved [Glass & Holyoak 1986].

[4]Similar to A^q.

The goal of a task	Its characteristics may affect task motions.
Variable focus distance	The robot should also look around itself, in the neighborhood of the objects it is manipulating.
Speed	Objects moving quickly, especially towards the robot, must be in the focus.
Exceptions	Falsified pre–conceptions must be focused on.
New arrivals	Objects entering a scene should be noted.
Instruction	Someone may point to an object to indicate that it is important.
Previous errors	Objects which may have caused errors in the past deserve special attention.

Table 1: Possible extensions for ETAR.

6 Conclusion

We must work towards a deeper understanding of bias so that we reason about and choose appropriate biases for learning systems. Humans direct their attention very specifically, and as they appear to be good learners, our intuition is to emphasize a similar bias mechanism. We must abandon the notion of never changing, preprogrammed hypothesis spaces and concentrate on active filtering processes, which generate potential concepts and modify search "on the fly".

For the moment, a formal predictive theory is a premature goal. This paper develops a descriptive framework of the kind that must precede a generative theory of bias. We examine computation and example complexities, and learnability as measures to compare learning. A learner comprises examples, background knowledge and a learning algorithm, where examples are translated to suit a search strategy over an hypothesis space. With this model we propose an important distinction for bias — *static* versus *dynamic* — then decompose dynamic bias into techniques that *focus* or *magnify* components of the learning algorithm.

The version space strategy is applied to a simple hierarchy, contrasting

static, magnify, and focus bias with respect to the search space, and obtaining a comparative set of results for the complexities and learnability. ETAR — an implemented real world robot task learner — demonstrates the potential of, and need for, focus as a form of dynamic bias. Focusing controls the structure of the learned task, reducing the example complexity, and drastically decreasing the computational complexity by constraining the potential branches and loops. Learnability is somewhat reduced, but the simple focus on nearby objects seems powerful since — in a static environment — it brings all manipulatable objects to the learner's attention, as the arm moves.

Learning systems must rely heavily on dynamic biasing mechanisms to focus and magnify the learning process, if they are to learn a variety of real world tasks within an acceptable complexity.

References

Andreae, P. M. (1984) Constraint limited generalization: acquiring procedures from examples. In *Proc. American Association on Artificial Intelligence*, Austin, TX, August.

Angluin, D. and Smith, C. H. (1983) Inductive inference: theory and methods. *Computing Surveys*, 15(3):237–269, September.

Buntine, W. (1990) Myths and legends in learning classification rules. In *Proceedings Eighth National Conference on Artificial Intelligence*, pages 736–742. AAAI, The MIT Press.

Cohen, P. R. and Feigenbaum, E. A. (1982), editors. *The handbook of artificial intelligence*, volume III. William Kaufmann, Los Altos, California.

Dietterich, T. G. (1989) Limitations on inductive learning. In *Proceedings of the Sixth Machine Learning Workshop.*

Genesereth, M. R. and Nilsson, N. J. (1987) *Logical foundations of artificial intelligence.* Morgan Kaufmann.

Glass, A. L. and Holyoak, K. J. (1986) *Cognition.* Random House.

Gold, E. M. (1967) Language identification in the limit. *Information and Control*, 10:447–474.

Heise, R. (1989) Demonstration instead of programming: Focussing attention in robot task acquisition. Master's thesis, Department of Computer Science, The University of Calgary.

Heise, R. and MacDonald, B. A. (1989a) Robot program construction from examples. In *Proceedings of AI / CS '89*, Dublin, Ireland, September.

Heise, R. and MacDonald, B. A. (1989b) Robots acquiring tasks from examples. In *Proceedings of the 2nd International Symposium on Artificial Intelligence*, Monterrey, Mexico, October.

Hirsh, H. (1990) Knowledge as bias. In Benjamin, D. P., editor, *Change of Representation and Inductive Bias*, pages 209–221. Kluwer Academic Publishers.

MacDonald, B. A. and Witten, I. H. (1988) A framework for knowledge acquisition through techniques of concept learning. Research Report 88/303/15, Department of Computer Science, University of Calgary, Calgary, Alberta, March.

Mitchell, T. M. (1977) Version spaces: a candidate elimination approach to rule learning. In *Proceedings of the Fifth IJCAI.*

Russell, S. J. and Grosof, B. N. (1990) Declarative bias: an overview. In Benjamin, D. P., editor, *Change of Representation and Inductive Bias*, pages 267–308. Kluwer Academic Publishers.

Shvaytser, H. (1990) A necessary condition for learning from positive examples. *Machine Learning*, 5(1):101–113, March.

Utgoff, P. E. (1986) *Machine learning of inductive bias.* Kluwer Academic Publishers.

Valiant, L. G. (1984) A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November.