44764

National Library of Canada — Bibliothèque nationale du Canada

Canadian Theses Division — Division des thèses canadiennes

Ottawa, Canada
K1A 0N4

PERMISSION TO MICROFILM — AUTORISATION DE MICROFILMER

• Please print or type — Écrire en lettres moulées ou dactylographier

Full Name of Author — Nom complet de l'auteur

J. A. LAMONT

Date of Birth — Date de naissance

Feb. 11, 1952

Country of Birth — Lieu de naissance

Canada

Permanent Address — Résidence fixe

Apt. 4A, 66 Collier St. TORONTO, Ontario

Title of Thesis — Titre de la thèse

FUZZY LANGUAGE LEARNING

University — Université

U. of Alberta

Degree for which thesis was presented — Grade pour lequel cette thèse fut présentée

M. Sc.

Year this degree conferred — Année d'obtention de ce grade

Spring 1980

Name of Supervisor — Nom du directeur de thèse

Prof. L. K. Schubert

Permission is hereby granted to the NATIONAL LIBRARY OF CANADA to microfilm this thesis and to lend or sell copies of the film.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

L'autorisation est, par la présente, accordée à la BIBLIOTHÈQUE NATIONALE DU CANADA de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans l'autorisation écrite de l'auteur.

Date: Dec. 21, 1979

Signature: Jim Lamont

-91 (4/77)

National Library of Canada
Collections Development Branch

Canadian Theses on
Microfiche Service

Bibliothèque nationale du Canada
Direction du développement des collections

Service des thèses canadiennes
sur microfiche

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us a poor photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

THIS DISSERTATION HAS BEEN MICROFILMED EXACTLY AS RECEIVED

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de mauvaise qualité.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

LA THÈSE A ÉTÉ MICROFILMÉE TELLE QUE NOUS L'AVONS REÇUE

Ottawa, Canada
K1A 0N4

NL-339 (Rev. 8/80)

THE UNIVERSITY OF ALBERTA

FUZZY LANGUAGE LEARNING

by

© J. A. LAMONT

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTING SCIENCE

EDMONTON, ALBERTA

SPRING, 1980

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled .... FUZZY LANGUAGE LEARNING .... submitted by J.A. Lamont .... in partial fulfilment of the requirements for the degree of Master of Science.

....
Supervisor

....

....

Date. Nov. 26/79 ....

To Dale

Now it is fog, I walk
Contained within my coat;
No castle more cut off
By reason of its moat:
Only the sentries cough,
The mercenaries talk.

The street lamps, visible,
Drop no light on the ground,
But press beams painfully
In a yard of fog around.
I am condemned to be
An individual.

In the established border
There balances a mere
Pinpoint of conciousness.
I stay, or start from, here:
No fog makes more or less
The neighboring disorder.

Particular, I must
Find out the limitation
Of mind and universe,
To pick thought and sensation
And turn to my own use
Disordered hate or lust.

I seek, to break, my span.
I am my one touchstone.
This is a test more hard
Than any ever known.
And thus I keep my guard
On that which makes me man.

Much is unknowable.
No problem shall be faced
Until the problem is;
I, born to fog, to waste,
Walk through hypothesis,
An individual.

Thom Gunn

ABSTRACT

Language learning is an example of a task that is not, in general, recursively solvable, but for which non-algorithmic solutions exist that give insight into many areas of theoretical, and potentially practical, interest. A scenario for language learning is proposed that:

*parallels that used for the inductive inference of partial recursive functions,

*permits a precise description of the relationship between function and language learning,

*suggests how language learning might be rephrased in a fuzzy context

The possibilities for function and (non-fuzzy) language learning are surveyed. Several problems commonly confused with language learning are outlined and their relationship clarified.

Fuzzy formal languages result from the continued quest to make formal languages somewhat closer to natural language, by making membership in a formal language gradable. The various types of grammars for fuzzy languages are critically analyzed. Some comments are made on how the "generation problem" might be approached in the future.

The previous work dealing with the learning of fuzzy languages is critically examined. A similar solution involving only the assignment of grammaticalities to rules is given. It is noted that both solutions rest upon the

dubious assumption that a superset of some set of correct rules is known. A new outlook, arising from the unique presentation employed in earlier chapters, is suggested and a theorem is shown that establishes the equivalence of certain partial recursive functions and fuzzy grammars. This leads to a general method of solution when fuzzy grammars are used to name the target language.

Viewed philosophically, fuzzy languages seem to require a more approximate criterion for learning than any previously given, one that permits an infinite, yet bounded, number of discrepancies between target and hypothesis. Previous material on approximate learning is surveyed. A new criterion for learning implied by the previous work for fuzzy languages, "order matching", is defined and shown to be reducible to the usual concept of matching. A new notion extending the previous work for approximate function learning, "E-identification", is defined. It is shown that E-identifiers can learn very large classes of total recursive functions, and are more powerful on the total recursive functions than almost everywhere identifiers. E-identification bounds the overall proportion of differences between the target and hypothesis. In order to permit the overall proportion of differences for each range value of the target to be bounded, E_{range}-identification is defined, and, for finite ranged functions, shown to entail E-identification. The theorems for E-identification are restated for E_{range}-identification. Finally, the equivalent results are stated for the languages generated by fuzzy grammars.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Len Schubert, for many, many things. Without his suggestions and patience I would not have completed this.

Thanks are also due Jeff Pelletier, who listened and responded to some of my early ideas and was an observer in my exam, and my examining committee.

For assistance in the preparation of the final draft using the Anderson-Jacobson, I am indebted to Anne Brindle and Jim Achugbue.

Many people gave me emotional support during this time: Tekin and Meral Ozsoyoglu, Alan Covington, Randy Pawson, Jean Dumouchel, Jim Savage, Dale and my parents. I can merely say that I am very grateful.

This research was supported, in part, by the Natural Sciences and Engineering Research Council of Canada under Operating Grant A8818.

TABLE OF CONTENTS

CHAPTER PAGE

Chapter 1

INTRODUCTION

> The true guarantee of the validity of induction is that it is a method of reaching conclusions that, if it be persisted in long enough, will assuredly correct any error concerning future experience into which it may temporarily lead us.
>
> C. S. Pierce

> A method of solution is perfect if we can forsee from the start, and even prove, that following that method we shall attain our aim.
>
> Leibnitz

> The solution to a problem changes the problem.
>
> Peer's Law

1.1 The Problem, Intuitively

A problem that has received considerable attention[1] since its formulation by Chomsky <Chomsky,1957; 1965>, is, simply stated, that of discovering a name for a formal language L <Hopcroft and Ullman,1969>, given only a finite sample of L (and perhaps L complement) from which to make the inference. This is commonly known as "grammatical inference" since some form of grammar, often a Chomsky Type grammar, is what is conventionally meant by a "name" for a

[1] For the two major, albeit incomplete, surveys see <Biermann and Feldman,1972> and <Fu and Booth,1975a>. The latter stresses "practical solutions" that demonstrate a tendency to confuse this with "the good encoding problem" as discussed later in this chapter.

formal language.

The general problem dealt with by this thesis is the reinterpretation of Chomsky's original task within a fuzzy or vague context. First, a statement and analysis of language learning for fuzzy formal languages[1] is given along the lines Chomsky proposed for (nonfuzzy) formal languages. Second, a new approximate notion of language learning, consistent with the spirit of fuzzy languages, is analyzed.

The problem of fuzzy language learning manifestly depends upon developments in both the theory of formal language learning and the concept of fuzziness. The former suggest the problem's outline, and the latter its elaboration.

1.2 Chomsky's Problem in Context

Although the problem of natural language learning has a very long history <Chomsky,1975>, the study of the learning of formal languages from examples and, possibly, counterexamples, of course arose only after the creation of formal languages. These reflect not only a relatively primitive conception of language as simply a corpus of utterances; but also a uniquely Chomskian outlook that "language shall no longer be regarded as a corpus of utterances per se, but rather as the abstract system of rules that underlies these utterances" <Chomsky,1957>.

---

1 i.e. languages for which membership is gradable

Formal languages incorporate Chomsky's view that natural language possesses a non-trivial structural or syntactic component (i.e. one that is more than a mere list built up by repetition and "analogy") that is independent of any semantic considerations.

Our models of natural language have steadily become more semantically based and complex [1], while work on formal language learning has continued to be exclusively syntactic, despite the occasional, rather dubious, claim to the contrary <Crespi-Reghizzi,1971>. Although the recent work in formal semantics <Stoy,1977> may eventually provide an opportunity to rectify this, the divergence has meant that the relationship between natural and formal language learning studies has become somewhat tenuous. A subject that often appears to impinge upon both, namely the so-called "computational study of language acquisition" <Reeker,1976>, has had in fact no particular relevance for either. The motivations and questions in these fields are often similar, as a comparison of the studies of Gold <1967>, Shrier and Brown <1978>, Reeker <1976>, and Dale <1972> shows particularly well; and occasionally even the resultant research is similar, as the formal studies by the psycholinguists Hamburger and Wexler <1973a,1973b,1975>

---

1 See, for example, <Katz and Fodor,1963> for the desirability of a semantic emphasis in linguistics, and <Charniak and Wilks,1976> for some developments along these lines in Artificial Intelligence.

demonstrate. The distinguishing factor seems to be the relative importance assigned to certain features of natural language versus the tractability of formal language, which determines the role formal languages can play in understanding natural language phenomena[1].

It is the author's belief that, in the current absence of any markedly linguistic constraints, the learning of formal languages does not yet provide an adequate paradigm for even the syntactic component of natural language learning. It appears to lack many of the features that serve to distinguish natural language learning from more general inferencing, for example, uniformity, rapidity, comparative intellectual ease, and freedom from motivation and emotional state <Chomsky,1965; Miller,1967; Dale,1972>. However, this picture could change with some of Angluin's research <Angluin,1974; preprint> that consciously seeks to incorporate these qualities into a formal language situation, and with recent experiments <Reber,1977> that suggest the differences between natural and artificial language learning situations may not be nearly so great as previously believed.

The well known equivalence between grammars, machines, and programs or partial recursive functions <Hopcroft and Ullman,1969>, has resulted in the formulation of problems in

---

1 See Levelt <1974> for an excellent appraisal of this issue.

a variety of different terminologies that are very similar to that of learning a formal language. Studies on the "identification of a finite state machine from a sample of its input/output behavior", "the automatic programming of a task given examples", and the "inductive inference of a partial recursive function from some partial enumeration" are all extremely relevant to the task of learning a formal language. So relevant, indeed, that researchers in any one of these areas customarily cite results deriving from all.

Solutions to the fuzzy language learning problem hinge upon an explicit unification of the functional and linguistic approaches. The exact relationship between these two areas is not immediately apparent and indeed has been variously interpreted <Blum and Blum,1975; Feldman and Shields,1977; Wiehagen,1977>. It is elaborated upon considerably in Chapters Two and Three.

Caution is necessary in interpreting the results peculiar to the various notations, since each can encourage subtly different formulations, the differences of which may not be immediately apparent, as Gold's <1967> "Black Box Identification" illustrates. However, with this in mind, the problem of learning a formal language may be viewed in the light of any of the developing theories of inductive inference couched in "effective" terms.[1] So it is particularly surprising that the vast philosophical literature on induction <Barker,1957> seems to be almost wholly irrelevant, of use only in pointing out the difficulties that must beset any such enterprise.

Two philosophical conundrums challenge the very possibility of a solution. Goodman's Paradox states that for every predicate P and every finite set O of objects, there is another predicate P* equivalent to P on O and -P on O complement <Kutschera,1973>. This merely formalizes the commonplace observation that an infinite language cannot be characterized logically by any finite set of examples and counter-examples. Hume's Paradox then asks: If in inductive inference the hypothesis or theory P is not logically or deductively contained within the given data C, that is to say that in some worlds C does not arise from P but from some different premise P*, then what is there to guarantee that we are in a world where the inductive inference of P, and not P* say, is correct? Nothing. Consequently the current philosophical consensus appears to be that induction can not be justified deductively.

Of course induction does appear to work, and so

---

1 This equation of language learning to general theory formation is dependent upon the view, realized by formal languages, that a grammar is a theory for a language <Chomsky,1957>. As noted previously this equation may now appear somewhat perverse to those whose concern is natural language <Derwing,1973>.

philosophers have striven to provide rational, rather than purely logical, justifications for it <Black,1970b>. Depending upon the reader's tastes these may or may not prove satisfactory. Of more significance here however, are the attempts to construct an "inductive logic", in which rather than requiring deductive validity of an inductive argument, a degree of probability or "confirmation" is assigned to it. That is, an inductive logic would provide a mechanism whereby (possibly conflicting) hypotheses could be ranked in the light of the data currently available. According to Morgan <1971a>, philosophers have concentrated almost exclusively upon this Confirmatory problem. Unfortunately the very nature of "confirmation" is beset by paradoxes <Salmon,1973> which grow, if not worse, at least more explicit in the usual probabilistic treatments of induction <Gardiner,1978>.

Two of the main reasons for the curiously diminished relevance of philosophical studies to this thesis are apparent from the previous paragraph. First, the issue of confirmation (as distinct from validation) can wrongly focus attention upon the merits of <u>particular</u> inductive arguments: "Given the data, which of the available, deductively adequate hypotheses is more likely?" Although this approach appears in some (inconclusive) attempts at constructive grammatical inference <e.g.Cook and Rosenfeld,1974>, the key to the logical justification of induction, and the fundamental outlook of the material covered in this thesis, is that it is inductive strategies which must be evaluated: "Given a growing set of data, will a particular strategy eventually arrive at a correct hypothesis?" Herein is the avenue to a deductive treatment of induction, and a rigorous examination of its potential. And although, as this chapter's opening quotations indicate, several philosophers may have realized this, their insight remained undeveloped.

The second debilitating feature of philosophical studies, in so far as the interests of this thesis are concerned, also stems from their concentration upon Confirmation. The Discovery process surely is of fundamental significance to our problem, yet it has usually been shelved pending a full explication of Confirmation <Morgan,1971a>. It has even been declared extra-logical[1] and best studied by analyzing society, history or the creative genius <Toulmin,1973; Hadamard,1954>. This deficiency is not really remedied here. This thesis analyzes only one particular creative method, that of enumeration with possible tests (although many of the results are valid for all possible, effective, methods.) The failure to realize that these results provide few if any suggestions for non-enumerative learning methods is responsible for the voluminous, but to date ineffectual, studies on "constructive" schemes, about

[1] "There are... no generally applicable 'rules of induction', by which hypotheses or theories can be mechanically derived or inferred from empirical data. The transition from data to theory requires creative imagination." Hempel cited in <Derwing,1973>

which more is said towards the end of Chapter Three. So then, although new methods of Discovery would be extremely relevant here, the philosophers have not been disposed to look for them, believing with Popper that there is no "logic of discovery" to discover.

One final point about the philosophical studies on induction is that they have operated outside of the computational orientation which is central here. This, together with the other factors noted, means that explicitly philosophical material is of only peripheral relevance. Even Karl Popper's work on the "hypothetico-deductive" method <1968> fails to make specific, computational suggestions other than the basic one that successful strategies should employ a "generate and test" strategy operating upon a basis of falsification rather than confirmation. In fact the influence seems, if anything, to flow in the opposite direction, with Case and Smith <1978> and Kugel <1977>, to name but a few, explicitly detailing the import of their work for the philosophy of science. There appears to be particular relevance for Chomsky's LAD <Chomsky,1975> and the rationalist-empiricist debate <Derwing,1973>. However, scientific theories are judged on many more grounds than generative or predictive adequacy, simplicity, or indeed any of the criteria of the formal studies discussed in this thesis <Toulmin,1963,1973>.

Even computer scientists committed to the creation of a logic of discovery often fail to come to grips with the central dilemma, in the learning of formal languages. Men such as Hajek <1975>, Meltzer <1970>, Morgan <1971>, and Plotkin <1971>, trying to design explicit "logics of discovery", generate inductively complete sets of hypotheses for given finite sets of data. Trivial and contradictory hypotheses abound in such complete sets. Moreover, for any static set of data, such as the above researchers are concerned with, the philosophical conundrums mentioned earlier ensure the impossibility of picking out the correct hypothesis. Consequently their work will be of relevance only if it is incorporated into some strategy dealing with growing data sets.[1]

There are several other problems that, although often confused with the topic of this thesis <e.g. Gaines,1977>, really avoid the central difficulty of the language learning problem. Two of these are the "good encoding problem", and the "finite selection problem". The good encoding problem may be stated as: How, from a finite sample S of a formal language L (and perhaps its complement), can one discover "good" names for S (Note: for S not for L) <Daley,1977>. Solutions to the good encoding problem may be thought of as logics of discovery that filter hypotheses by a type of "confirmatory measure" for which intrinsic properties of the

---

1 A suggestion along this line <Schubert,personal communication> that appears capable of speeding up and perhaps making more practical some of the enumerative approaches is discussed later.

hypothesis (for example simplicity) rather than any relationship to L are what matter. The language learning and the good encoding problems coincide when L is finite. This, together with the overlap in terminology and the fact that an immediate concern with good encodings can go hand in hand with the larger problem of ultimately acquiring a correct name for L, makes it difficult to distinguish which problem is addressed by some authors. This is particularly true of most of the conceptual or structural learning programs in Artificial Intelligence <e.g. Winston, 1970>, but occurs even in adequately formalized theories such as the General Systems approach to the identification of "generative structures in observational data" <Klir,1976>[1] . Characteristic of such work is that it is "situation static", and is only shown to provide subjectively reasonable solutions.

Take the language learning problem and modify it by the provision of sufficient (usually a priori) additional information to enable all but a finite number of hypotheses to be discarded out of hand and one has the finite selection problem. Finite state machine identification provides perhaps the prime example of this <Moore,1956; Gaines,1975>. Here the usual ploy is to assume knowledge of the (maximum) number of states and input/output symbols. Conceptually the

---

1 <Zalecka-Melamed,1977> makes the point that these theories are really concerned with what is called later "identification in known time".

problem is then trivial. Since there are only a finite number of fsm's satisfying these bounds, the target machine can be identified by forming their direct sum and conducting a homing experiment <Kohavi,1978>. Such variations can lead to a host of very practical problems, for example in fsm fault detection <Kohavi,1978>.

Finally, there is a curious hybrid between the language learning and finite selection problems. This is obtained by the provision to the language learner of information additional to examples and counterexamples from the target language, yet insufficient necessarily to reduce the problem to the finite selection case. The provision of partial parsing information for the data by Crespi-Peghizzi <1971> is a good example of this. More often the studies on this problem involve examples of a program's input-output together with program traces <e.g.Barzdin and Freivald,1972; Biermann and Feldman,1972a; Siklossy and Sykes,1975>. Since the boundaries are not sharp between the categories of language learning, good encoding, finite selection, or this additional information situation, where a particular study should be placed is often problematic.

It should be clear that a solution to the language learning problem depends upon a very strong notion of "pattern" or "structure", one involving the existence of a mechanism for its generation. As the search for regularity in the environment, language learning studies are related to the countless other endeavours in this direction:

statistical analysis, classical pattern recognition, information theory, and so on. Yet beyond this commonality of interest there is little similarity apparent even when, as in <Watanabe,1969>, the subject is explicitly that of "induction".

However, certain deep connections are appearing that should be mentioned, if only briefly. First of all, the non-probabilistic inference outlined by this thesis is the simplest case of the more general probabilistic inference which since <Solomonoff,1964> has blossomed into a far reaching discipline of its own[1]. This in turn is related to classical probability theory as follows. Classical studies left the notion of "random" (and hence of "non-random" or, intuitively, patterned) as an undefined primitive, a characteristic of a process rather than a sequence. Over the last decade various researchers have striven to give a theory of randomness in terms of general models of computability, and thereby erect a new, constructive, theory of probability[2]. Developments in complexity theory have also been intimately involved with this <Schnorr,1973>. Recently a direct link has been shown to exist between the precise notions of "random" and "predictable" <Levin,1973; Schubert,1977>.

---

1 See <Solomonoff,1975> for some recent developments and a partial overview.

2 See <Schubert,1977> for some recent developments and an overview; <Humphreys,1977> for a philosophical discussion.

At a more superficial level the relationship between the language learning problem and these other theories is one of mutual borrowing. For instance, both classical and Bayesian statistical methods are used heavily in stochastic language learning <Fu and Booth,1975b>. A whole new field with numerous applications <Fu,1977>, Syntactic Pattern Recognition, has grown out of classical pattern recognition due to the influence of grammatical inference and the invention of picture grammars <Fu,1974>.

Although still primarily of only theoretical interest, as both the highly abstruse mathematics of <Lindner,1974> and the practical calculations in <Wharton,1977> attest, the study of language learning has been turned to several quite practical ends. It has been applied to the inference of biologically relevant L-systems[1], the design of programming languages <Crespi-Reghizzi,1973>, and to the automatic construction of transition network grammars <Chou and Fu,1972> popular in A.I. studies of natural language <Kaplan,1972>. Some researchers <Biermann and Smith,1977> are using it to study automatic programming[2] and, as

---

1 See <Herman and Walker,1972> for the first treatment, albeit one focussing more on the good encoding problem. <Coy and Pfluger,1979> gives many results and shows their relationship to the standard language and function learning material.

2 However the more customary approach is to attempt the inference of a program from some semi-formal description in another language. Strictly speaking, this is more of a problem in translation than of inductive inference <Biermann,1976>.

mentioned previously, pattern recognition <Evans, 1971>.

1.3 Fuzzy Language Learning in Context

The peculiarities inherent in human imprecision have long been known to the philosophers <Black,1970a>. With Zadeh's creation of "fuzzy" set theory in 1965, the issue of imprecision could be analyzed precisely. Humans use such vague concepts as "long", "old", "relevant", and so on, to great advantage. The hope is that "fuzziness" models this everyday phenomenon sufficiently well to be useful even if it is not unchallengeable <Stallings,1977>.

Although there has been some effort towards the creation of a fuzzy deductive logic <Zadeh,1977>, there have been no fuzzy inductive logics developed.

Learning a fuzzy language is related to previous work in fuzziness since a fuzzy formal language is defined to be a fuzzy set of sentences constructed from some finite vocabulary <Lee and Zadeh,1969>, and so the learning of fuzzy languages can be thought of as what Zadeh termed the problem of abstraction <Bellman et al.,1969>. This can be seen as an attempt to make the inference of formal languages closer to the natural language situation <Tamura and Tanaka,1973>, as a theoretically interesting extension of the usual studies on the acquisition of formal languages, or even as possibly leading to an often called for aid <Fu,1974> to those engaged in fuzzy syntactic pattern recognition <e.g. Thomason, 1973; Kickert and Koppela,1976;

De Palma and Yau, 1975>.

1.4 Formalisms for "Solvability"

In a 1962 paper Shamir remarked informally that since it is possible to have two distinct (infinite) languages coinciding upon any specified finite set of strings (Coodman's paradox revisited), it is impossible in general to discover a correct grammar from only a finite sample of an infinite language[1]. Moore <1956> proved that for an arbitrary finite state machine M, even if it is permissible to specify any finite input sequence S for M to respond to, there will be other non-equivalent machines that have the same output sequence for S and hence

> "it will never be possible to perform experiments on a completely unknown machine which will suffice to identify it from among the class of all sequential machines."

Wiehagen <1978> characterized the classes of languages for which Chomsky's problem is recursively solvable, showing them to be relatively trivial (cf. 2.2.1).

This perhaps explains the appeal of the variants of the language learning problem mentioned earlier. For if Church's Thesis was fully endorsed and any intuitively "solvable" problem was required to be solvable in the usual Turing Machine sense, then the quest for general solutions to the

---

[1] Although the point was not emphasized before, the problem assumes that the samples are not of some special sort such as the "representative samples" of Schubert <1974b>.

language learning problem would be futile. To paraphrase <Gold,1967>, given only a finite amount of information and no a priori basis for choosing among logically valid inferences, a learner cannot possibly avoid making mistakes. Consequently, all that a learner should be expected to do is employ a sound METHOD of making inductive inferences, not always to make the particular inference that is correct.

In the mid-sixties another conception of solvability arose that "overshoots the bounds of Church's Thesis" <Crisculo et al.,1975>. Putnam's "trial and error predicates" <1965> and Gold's "limiting recursion" <1965> mimic the activity of a successful scientist. Unlike Turing's calculator of arithmetic sums <Turing, 1950> which must halt and announce its final solution if it is to succeed, the scientist is not expected to ever cease the calculation of new and better theories as increasing amounts of data become available. Crudely put, the requirement for success is only that the sequence of hypotheses be convergent, somehow, to "The Truth". The distinction between these two kinds of solution has been compared to that between a procedure and an ongoing process <Crisculo et al.,1975>.

Gold's formulation considers a Turing Machine T to successfully compute the value of a function f at x if T gives an infinite sequence of outputs, only finitely many of which are different from f(x). A function f that can be so computed, by one Turing Machine, at every point of f's domain, is called "limiting recursive". More rigorously, "limit" is taken to be a functional operator that associates to each total function g of n+1 variables a partial function f of n variables such that:

$$f(x_1,\ldots,x_n)=\lim_m g(x_1,\ldots,x_n,m) \text{ if the limit}^1 \text{ exists}$$
$$\text{undefined} \quad \text{otherwise}$$

A (partial) function is said to be a (partial) limiting recursive function if it is expressible as the limit of a total recursive function. A set is said to be limiting recursively enumerable if it is the domain of a partial limiting recursive function. Terminology and results for limiting recursion, akin to those of recursive function theory, can be developed considerably further <Goetze and Klette, 1974; Criscuolo et al., 1975>.

Limiting recursion is more powerful than normal recursion. For instance, the classic "unsolvability" result, "The Halting Problem", is limiting recursively solvable.[2] All that is required is a modified Universal Turing Machine U which when fed a T.M. index i outputs "no" unless and until its simulation of i has halted, upon which it outputs "yes" thereafter. Notice that the strategy S of taking the

---

[1] This is the usual number theoretic "limit", namely $\lim_x f(x)=a$ iff $f(x)=a$ for almost all $x \in N$

[2] Actually Putnam's "2-trial predicates" suffice to solve membership in K (i.e. the set of programs that halt upon their own index). These are weaker than limiting recursive predicates since ($\exists$ k : P is a k-trial predicate) iff ($P \in \Sigma_1^*$) <Putnam, 1965>. A commonly known, still more "unsolvable" problem that is limiting recursively solvable is the "Busy Beaver" problem <Ausiello and Protasi, 1975>.

current answer provided by U as the correct one guarantees that only a finite number of mistakes will be made before S is operating upon the correct assumption. This is a general feature of limiting recursive solutions. A feature of this solution that is not characteristic of limiting recursive solutions in general is the knowledge that if a "yes" occurs all further computations are redundant since "yes" must then, by construction of U, be the correct answer. In general, although from some point on in the computation of a limiting recursive function f at x only f(x) is returned, it is not possible to determine when that point has been reached. In short, although the Turing Machine eventually "knows" the correct answer, it may never be able to know that it knows and so halt.

To illustrate the scope of limiting recursive techniques, it is necessary to outline something called the "Arithmetical Hierarchy" <Rogers,1969>. This serves as a standard means of ordering the different degrees of recursive unsolvability. An n-ary relation R is in the <u>arithmetical hierarchy</u> iff it is recursive or can be expressed as $\{(x_1,\ldots x_n): (Q_1 y_1),\ldots(Q_m y_m)\ S(x_1,\ldots x_n, y_1,\ldots y_m)\}$ where each Q_i is either $\forall$ or $\exists$, and is over numeric not functional coordinates (however the x_i may be function indices), and S is an (n+m)-ary recursive relation. The expression within the brackets is called a <u>predicate form for P.</u> . It can be shown that if a relation P can be stated within quantificational logic using recursive relations, then it is in the arithmetical hierarchy (the converse is also true, trivially). This is the case iff R is definable within elementary arithmetic, hence the name. It is a curious fact that it is the minimum number of quantifier alternations (i.e. number of adjacent but unlike quantifiers) in a relation's predicate form(s) that determines its maximal degree of unsolvability. Σ_n is defined to be the class of all relations expressible by predicate forms beginning with $\exists$ and having (n-1) quantifier alternations. Π_n is defined exactly as Σ_n except that the predicate forms must begin with $\forall$ rather than $\exists$. Often a superscript ^{0} is added to these symbols in order to indicate that the quantifications are over numeric rather than functional coordinates. The smallest n for which a relation belongs to Σ_n or Π_n indicates the relative recursive unsolvability of the relation, with higher n denoting "harder" problems.

This solvability hierarchy is a complicated affair, but for our purposes here a few examples and one key result by Kleene and Post should suffice. $\Sigma_0 = \Pi_0$ = the recursive sets. Σ_1 is the class of recursively enumerable sets. {i : domain of the ith Turing machine is finite} is in Σ_2 and {i : the domain of the ith Turing machine is infinite} is in

Π_2. In fact any sets in Σ_2 or Π_2 are "Turing reducible"[1] to these two sets respectively. A particularization of the Kleene-Post Theorem states that for any relation R, ($R \in \Sigma_2 \cap \Pi_2$) iff (R is Turing reducible to K). Also, $R \in \Sigma_2$ iff R is recursively enumerable in K.

The power of limiting recursion can now be sketched in terms of the Arithmetical Hierarchy. There are two[2] main results of concern here:

* A predicate R is limiting recursive iff $R \in \Sigma_2 \cap \Pi_2$ <Gold,1965; Putnam,1965>. So the question of whether or not R is true at a given point is limiting recursively solvable iff R is Turing reducible to K.
* A predicate R is limiting recursively enumerable iff $R \in \Sigma_2$ <Crisculo et al.,1975>. That is, the points for which R is true can be effectively enumerated given only some (arbitrary) enumeration of K.

---

1 Intuitively speaking, a set A is <u>Turing reducible</u> to a set B if the the provision of an oracle to decide questions of membership for set B allows the resolution by a Turing machine of membership questions for set A. A very similar notion is that of a set A being recursively enumerable in a set B. This means that there is a Turing machine that, given any enumeration of B, can then enumerate A. This is a slightly weaker notion than Turing reducibility.

2 See <Jeroslow,1975> for an analysis of the scope of limiting recursive methods in terms of the more usual logical notions of "consistency" and "completeness".

1.4 Formalisms for "Solvability"

1.5 More About Formalisms for "Solvability"

Very many of the investigations of language learning, from Gold's initial demonstration in 1967 that it was possible, through to Wiehagen's elaborate complexity and numbering theoretic characterizations in 1978, depend upon a limiting recursive functional to effect their solution. The investigation of any problem is inexorably determined by what is deemed to constitute an acceptable type of solution. So when a recursive solution is sought, Chomsky's problem is all but impossible, whereas when a limiting recursive solution is sought, it is solvable for distinctly non-trivial classes of languages. Much of this thesis is devoted to the explication of these words and their realization in a fuzzy context.

Notions of "solution" other than "recursive" and "limiting recursive" are used occasionally in language learning studies. The two most frequently occurring ones are generalizations of limiting recursion.

Schubert <1974a> defined a (partial) k-limiting recursive function by applying the limit operator k times (assuming the intermediate functions are total) to a total recursive function. Of course the 1-limiting recursive functions are just the limiting recursive functions. But the limit operator is enormously powerful - the entire Arithmetical Hierarchy can be characterized by repeated applications of it[1] <Schubert,1974a; Crisculo et al.,1975>. And for low values of k there are intuitive interpretations that still appear to preserve a degree of "effectiveness" <Schubert, 1974a>. For example, 2-limiting recursion may be modelled as an expanding community of processes of which only finitely many never settle upon the correct answer.

Probabilistic limiting recursion is the second major generalization of limiting recursion. If defined carefully, this is also a very powerful approach. For example a function f is "weak computable in the limit with probability>0" iff f $\in \Sigma_3$ <Freivald,1974>, where a function is defined to be weak computable in the limit with probability > p if there exists a Turing machine T with access to a Bernoulli generator (p=1/2) such that:

a. if f(x) is defined then the probability of printing an infinite output sequence with limit f(x) is > p

b. if y ≠ f(x) the probability of printing an infinite output sequence with limit y is ≤ p.

The need to draw the limit somewhere, together with the fact that one model has predominated to this date in the language learning problem (and the related problems in the other terminologies), means that this thesis deals almost exclusively with answers based upon the limiting recursive paradigm, and mentions the other generalizations only

---

1 Very briefly: A set R has a k-limiting recursive characteristic function iff R $\in \Sigma_{k+1} \cap \Pi_{k+1}$. A set R is k-limiting recursively enumerable iff R $\in \Sigma_{k+1}$.

occasionally to give some idea of the relationships.

Chapter Two outlines the topic of function learning, providing the framework for the third chapter and detailing the major theorems that constrain solutions to the formal language learning problem. Chapter Three describes at length the relationship between function and language learning studies and discusses certain features more characteristic of the latter area. There is a dual emphasis in Chapters Two and Three, namely the provision of a general basis for comprehension, comparison, and reformulation with respect to fuzzy language learning, and the description of the specific results relevant to approximate learning. Chapter Four introduces the notion of "fuzziness", and analyzes the various suggestions for naming fuzzy languages. And Chapter Five shows how the previous learning material can be rephrased in a fuzzy context.

Chapter 2

LEARNING FUNCTIONS

2.1 Introduction

Perhaps the most revealing view of Chomsky's problem stems from the realization that learning a formal language can be understood as the learning of either the language's characteristic or semi-characteristic function. This quite properly suggests that the greater number of results dealing with function learning be considered in any investigation of language learning. In fact the inescapable question is why there are two areas at all; why is there not a single unified development? To quote <Feldman and Shields,1977>: "there has been surprisingly little carryover from the one domain to the other ...[although] a common understanding of the issues seems to be emerging".

A dual presentation is maintained in this thesis for a number of reasons. Foremost is the fact that as informal terms such as "learning" are exchanged for their precise counterparts certain differences appear in what researchers in the two fields have been trying to do. For example, the acceptance of extensions to partial functions has been standard in the function learning problem whereas it is not usually acceptable when considering a function as the semi-characteristic function for some language. Furthermore, while the function enumerated and the target function are synonymous in functional studies, this is not always the case in the linguistic research. These potential differences are expanded upon in Chapter Three. The second reason for retaining the function/language dichotomy is that functional and linguistic terminologies encourage distinctive habits of thought and, by rendering certain questions, restrictions and modifications more natural, encourage the pursuit of different kinds of results. For example, the largely linguistically motivated distinction between examples and counter-examples plays an important role in the language learning results, but only rarely is the corresponding functional version mentioned.

For these reasons then, this chapter presents a separate outline of function learning.

The terminology and ideas dealing with function learning have the advantage of clarity and relative simplicity over those dealing specifically with language learning. So much so, in the author's opinion, that this thesis attempts to maintain the style of the functional material through into the linguistic studies. To avoid the confusion so easily engendered by premature generality, this chapter proceeds by adding to or modifying a basic model. In contrast to this, the next chapter starts with a general framework that, given the basic understanding developed here, should broaden the perspective.

2.1 Introduction

2.2 The Basic Models

2.2.1 Identification in the Limit

A few definitions are required to begin with. In general the notation of <Hopcroft and Ullman,1969> and <Rogers,1967> is used wherever appropriate. Functions are usually assumed to be mappings from N to N (N is sometimes identified with W), and a standard indexing of the partial recursive functions is assumed throughout. t_i stands for the ith partial recursive function, and T_i stands for a computational complexity measure for t_i of the type analyzed in the survey article of Hartmanis and Hopcroft <1971>. R is the class of total recursive functions. P is the class of partial recursive functions.

Definition: An (arbitrary) enumeration $\hat{f}$ of a partial recursive function f, is an infinite sequence of the form $(v_1,v_2,v_3,\ldots)$ where either $v_i=*$ or $v_i=(x_i,f(x_i))$ with $x_i \in$ Domain(f) and every $x_j \in$ Domain(f) appearing in some v_j.

Definition: $\hat{f}=(v_1,v_2,v_3,\ldots)$ is a primitive recursive [effective] enumeration of a partial recursive function f if $\hat{f}$ is an enumeration of f and $\exists$ a primitive recursive [recursive] function $p:N \rightarrow (N \times N) \cup \{*\}$ such that $p(n)=v_n$.

Definition: $\hat{f}=(v_1,v_2,v_3,\ldots)$ is an increasing [methodical] [request] enumeration of a partial recursive function f if $\hat{f}$ is an enumeration of f and v_n=* or (n,f(n)) [v_n=* if f is undefined at z(n) and =(z(n),f(z(n))) otherwise, where z ∈ P is prespecified] [to determine v_n, the inductive inference machine specifies x_n, and v_n is subsequently either * (if f is undefined at x_n) or (x_n,f(x_n))]

Definition: A partial enumeration $\hat{f}_n$ of a function f, is the finite sequence consisting of the first n elements of an enumeration of f.

Definition: A Codelization [$\hat{f}_n$], of a partial enumeration $\hat{f}_n$, is the natural number supplied by some 1-1 recursive mapping, from partial enumerations to N, operating upon $\hat{f}_n$.

Definition: An inductive inference machine is a total Turing Machine whose inputs and outputs (for the first two models) are to be interpreted as Godelized partial enumerations and Turing machine indices respectively.

Definition: An inductive inference machine M converges to i for an enumeration $\hat{f}$, if the sequence M([$\hat{f}_1$]), M([$\hat{f}_2$]), M([$\hat{f}_3$]), ... has limit i.

Definition: An inductive inference machine M identifies a function f in the limit [1] if for every enumeration of f ∃ i such that M converges to i, and i is an index for a program that computes some extension of f.

[1] The "in the limit" qualification is often omitted for convenience.

2.2.1 Identification in the Limit

Definition: An inductive inference machine M identifies a class C of functions in the limit, if $f \in C$ implies that M identifies f. A class C of functions is identifiable if ∃ an inductive inference machine that identifies C.

Definition: Program i is compatible with a partial enumeration $\hat{f}_n$ if t_i includes $\hat{f}_n$.[1]

Definition: The identifying power of an inductive inference machine M, is the largest class of partial recursive functions that M can identify in the limit.[2]

Definition: ID is the class of sets of total recursive functions that are identifiable.

Definition: A Popperian machine is an inductive inference machine that outputs only indices of total recursive functions.

Definition: A finite function f is a function such that Domain(f) is finite.

Definition: A total recursive function f is h-easy iff ∃ $t_i = f$ such that $T_i(x) \leq h(x)$ for almost all $x \in N$, where $h \in R$.

Definition: A partial recursive function is h-honest if

[1] Note: For notational convenience, a sequence, that is a function whose domain is N, is sometimes spoken of as if it were its range. For example the sequence (1,2,3,...) is spoken of as if it were {1,2,3,...} when terms such as inclusion or containment are used.

[2] A model of induction is loosely described as "more powerful" than another if its power is larger than or contains the other's. This apparently is contrary to standard usage in linguistics.

$\exists$ an extension t of f such that $T(x) \leq h(x,t(x))$ for almost all $x \in$ Domain(f), $h \in R^2$.

Definition: $f \in P$ is everywhere O-compressed for some general recursive operator[1] O, if $\exists$ a program i computing f such that for any other program j for f and $\forall x \in$ Domain(f), $T_i(x) \leq O(T_j)(\max(i,j,x))$. That is, "modulo O", t_i is the fastest program for f.

The problem of demonstrating the existence or construction of an inductive inference machine that identifies a given class of functions in the limit is the main formalization of the intuitive goal of learning a function from a finite set of input-output tuples. Before going on, it should be emphasized once more that this goal can be made rigorous in a variety of more or less plausible ways. Two other models, "matching" and "extrapolation", are discussed in the next two subsections since they fit into very much the same framework. However there are still other models dependent upon rather different frameworks that are omitted. Most significantly, the desirable addition of probabilistic considerations is not treated here. Thus the learning of stochastic languages, as in <Horning,1969,1972>, <Cook and Rosenfeld,1974>, <Booth and Maryanski,1977>, <Liou

---

1 See <Rogers,1967>. Loosely speaking, a general recursive operator O is a mapping from P to P such that: R $\subset$ Domain(O), O maps R to R, and O is an "enumeration operator". An enumeration operator is a mapping from sets to sets that formalizes the notion of enumeration reducibility.

and Dubes,1977>, <Shrier,1977>, or <Van der Mude,1978> for example, is not considered (see <Fu and Booth,1975b> for a good survey); nor is the effect of probabilistic inductive inference machines <Podnieks,1975>.

The subtle nature of identification is not immediately apparent. Since a machine M that identifies a function f has converged upon a correct name after seeing only finitely many input-output tuples, identification satisfies the requirements of the informal problem statement. Yet in general there is no effective method for judging when sufficient input-output pairs have been input to M for M to have ceased giving incorrect outputs. It is this which permits identification to escape the trivial confines of the earlier recursive interpretations of the problem. Setting an a priori bound on the number of distinct input-output pairs input before M outputs a correct index,[1] or requiring M to indicate in the course of its calculations when this point has been reached[2] only reintroduce the problems discussed in Chapter One.

<u>THEOREM</u> <Wiehagen,1978> A class C of total recursive functions is identifiable in known time iff C ⊆ some recursively enumerable class of partial recursive functions

---

[1] This is known as <u>identification in fixed time</u>
[2] This is usually known as "identification in finite time". However, since identification takes place in finite time even for identification in the limit, this is more accurately described here as <u>identification in known time</u>. "Time" here refers to the partial enumeration numbers.

such that the ith and jth functions (for i ≠ j) differ from one another for some argument ≤ r(i) where r ∈ R.

There are a number of variations of the definitions given for which the possibilities of identification in the limit remain unaltered, that is solutions using the definitions given can be effectively translated into solutions involving the following modifications (and vice versa).[1] Inductive inference machines may be taken to be primitive recursive <Barzdin and Freivald,1972> or partial recursive[2] <Minicozzi,1976> functions. For total functions the requirement of convergence by arbitrary enumerations is equivalent to requiring convergence by increasing <Blum and Blum,1975>, request <Gold,1967>, methodical <Gold,1967> and effective <Blum and Blum,1975> enumerations. Although the definitions given do not require that in order to identify a function an inductive inference machine M must converge to the SAME index i for f regardless of the particular enumeration $\hat{f}$ input to M, this can be required without altering the results <Blum and Blum,1975>. Finally, the same results hold even if an inductive inference machine is permitted to output a correct index only once for a given function while varying the remainder of its hypotheses

---

1 However these modifications may, for example, alter an analysis of the solution in terms of the complexity.

2 In this case the requirement is that at least one output be made, and that there is an algorithm index i for the function, such that there is some point in every enumeration of the function past which the last output is i.

between only finitely many alternatives <Case and Smith,1978>.

As stated, the problem of identification in the limit is enumeration independent. Certainly inductive inference machines must be required to work for any of some fairly general class of enumerations, in order to avoid the theory's trivialization through certain trick classes of enumerations that give away the answer, such as those for which the x-value of the first pair in the enumeration is a least upper bound for a program index of the function being enumerated. However perhaps it is an over-reaction to insist that inference machines must work for arbitrary enumerations, since a less stringent requirement might suffice[1] and it is intuitively the case that a good teaching sequence, or order of presentation, can be a valid aid to learning.

If in the definition of identification "every enumeration" is changed to "primitive recursive enumerations" then P is identifiable in the limit <Blum and Blum,1975>. This follows from the observation that every partial recursive function can be enumerated by a primitive recursive function (namely that resulting from the standard dovetailing enumerative procedure). The method is to go

---

[1] For example, classes of enumerations containing only partial enumerations that can be translated algorithmically into a correct program index could be declared insufficiently general.

through the list of the partial recursive functions provided by some (arbitrary) listing of the primitive recursive functions, until one compatible with the current partial enumeration is found, upon which the partial recursive function responsible for that particular primitive recursive function enumeration is output <Gold,1967>.

It is customary in functional studies to assume that an inductive inference machine may choose its hypotheses from all of P. Tampering with this assumption can alter the machine's power, as is shown by a comparison of the following result with the subsequent characterizations of ID.

<u>THEOREM</u> <Case and Smith,1978> Given any Popperian machine M ∃, uniformly in M, a recursive function that enumerates the class C of functions M identifies.

This follows from the creation of C by the extension, by means of M, of every "finite initial function" (i.e.functions whose domains are some finite initial portion of N), the class of which is recursively enumerable.

Mention should also be made here that although it is usually assumed that the inductive inference machine has access to the entire partial enumeration $\hat{f}_n$ to make its nth hypothesis, the effect of "memory constraints" is investigated in <Wiehagen,1975>. Call the class of sets of total recursive functions identifiable when an inductive inference machine is only permitted to see the next element in $\hat{f}_n$, or only one element of its own choice, ITERATE and

FEED-BACK respectively. Then CONSISTENT $\subset$ ITERATE $\subset$ FEED-BACK $\subset$ ID, where CONSISTENT is defined shortly, and the containments are strict.

It is worthwhile detailing the exact relationship of limiting recursion to identification in the limit.

<u>THEOREM</u> <Wiehagen,1978> For any class C of partial recursive functions, $\exists$ a limiting recursive functional F such that $F(f) \geq \min \{i : t_i = f\}$ for all $f \in C$, iff C can be identified in the limit <Wiehagen,1978>.

The limiting recursive functional of the theorem is defined by $F(f) = \lim_n M([\hat{f}_n])$ where M is an inductive inference machine that identifies C and $\hat{f}$ is any enumeration of f.

The immediate question is: Which classes of functions can be identified in the limit? A partial answer is provided by:

<u>THEOREM</u> <Gold,1967> Any class included in a recursively enumerable class C of total recursive functions can be identified in the limit. [1]

This is clear by the following argument. Given a partial enumeration $\hat{f}_n$, enumerate C, checking the algorithms one by one for compatibility with $\hat{f}_n$, and output the index of the first compatible algorithm. Since the functions in C are total recursive, these checks always terminate, and since an index for the function f will eventually be reached (as all

---

[1] In fact, any class of total recursive functions that is r.e. in K can be identified in the limit <Case and Smith,1978>.

previous programs that do not compute f must compute a value different from f at some point in f's enumeration and so be discarded) the sequence of hypotheses resulting from successive partial enumerations must converge as required. Notice that it is not decidable in general whether the current hypothesis is correct.

The "enumeration technique" described above is beguiling in its simplicity. It is not, however, a maximally powerful method even on R. Define NUM to be the class of sets of total recursive functions that can be identified by enumerating some class of partial recursive functions and choosing the first one found to be compatible with the current data.

<u>THEOREM</u> <Blum and Blum,1975> A set S of total recursive functions $\in$ NUM iff $\exists$ a total recursive function h such that every function f $\in$ S is h-honest.

This result renders enumeration ineffective, for example, for any classes containing "arbitrarily difficult to compute" <Hartmanis and Hopcroft,1971> total recursive functions. The "self-describing functions" [1] form such a class that nevertheless is trivially identifiable. <Barzdin and Freivald,1972> contains the first mention of the existence of classes of total recursive functions that,

[1] These are functions for which the least x such that f(x)=1 is an index for a program for f. Obviously, for any partial recursive function there is a self-describing function that is almost everywhere identical.

although identifiable in the limit, are not included in any recursively enumerable class of total recursive functions <Barzdin and Freivald,1972>, and for which enumeration is therefore clearly inapplicable.

NUM and IDknown[1] are incomparable <Wiehagen,1978>.

In <Blum and Blum,1975> a more general technique, called "A Posteriori Inference", is devised that depends upon a "running bound" on the target function's complexity by means of general recursive operators. Although the method is defined to work only for total recursive functions, on these it is extremely powerful.

<u>THEOREM</u> <Blum and Blum,1975> ∀ general recursive operators O, ∃ M uniformly in O, such that f ∈ R is everywhere O-compressed implies M identifies f in the limit.

The converse of this result is also true and is stated later.

However, no method, however clever, will ever work for all total recursive functions since:

<u>THEOREM</u> <Gold,1967> R is not identifiable in the limit.

Since the proof is instructive and appears only slightly modified for several other results, it will be sketched here. Suppose there is some inductive inference machine M that identifies R in the limit. A recursive function will be

---

[1] This is the class of sets of total recursive functions that are identifiable in known time. Classes corresponding to other restrictions are indicated similarly, by the concatenation of "ID" with the appropriate term.

(ineffectively) constructed that M does not identify in the limit:

Let f^1 be the function whose increasing enumeration is $(i_1, i_2, \ldots, i_n, 0inf)$ where 0inf is the infinite string 0,0,0,... , and each i_i is either 0 or 1 (arbitrary). M must identify all functions looking like this, so suppose M guesses correctly for $\hat{f}_{x1}{}^1$.

Let f^2 be the function whose increasing enumeration is $(i_1, i_2, \ldots, i_n, 0^{x1}, 1inf)$, where 0^{x1} is a string of x1 0's separated by commas. Suppose M guesses correctly for $\hat{f}_{x2}{}^2$.

Let f^3 be the function whose increasing enumeration is $(i_1, \ldots, i_n, 0^{x1}, 1^{x2}, 0inf)$. Suppose M guesses correctly for $\hat{f}_{x3}{}^3$.

And so on. Let $f^* = \text{limit}_n\ f^n$. f^* is total recursive since following the above procedure permits the calculation of $f^*(x)$ for any x. Yet when the increasing enumeration of f^* is fed to M, by the construction of f^* M does not converge to any index. M's failure demonstrates the contradiction inherent in the assertion that an inductive inference machine exists that identifies R.

ID has been characterized in several ways. One method uses a function's complexity.

<u>Definition</u>: For a general recursive operator O define $R_O\text{max} = \{t_i : \forall n\ \max T_i(x) \leq O(t_i)(n)$ where max is taken over all $x<n$, and $t_i \in R\}$

THEOREM <Wiehagen,1978> A class C of total recursive functions ∈ ID iff ∃ a general recursive operator O such C ⊂ $R_o max$.

The operator that establishes the necessity of this condition is simply: $Of(n)=max\{T_m(x)$, where $m=M[\hat{f}_n]$), such that $x<n\}$.

In passing it should be noted that this characterization gives the most powerful (on R) inductive inference machines or strategies possible.

Since very often there are no estimates of the computational complexity of the target function, other sorts of characterizations of ID have been derived. <Wiehagen,1978> identifies the relevant literature, almost all of which is east European, recent, and untranslated. A typical result shown in <Wiehagen,1978> is:

THEOREM A class C of total recursive functions ∈ ID iff C ⊂ some recursively enumerable class of partial recursive functions such that the ith and jth functions ($i \neq j$) differ from one another for some argument $\leq r(i,j)$ where $r \in P^2$.

The previous material should not suggest to the reader that only classes of total recursive functions can be identified. Quite the contrary. It is merely that the situation for R is easier to analyze, and provides an upper bound to identification results in the sense that no identifiable class of partial recursive functions can contain all of R.

However, unlike the situation with respect to R and ID, there is no characterization for P of the identifiable classes.[1] However, if the object is to identify a class containing any strictly partial recursive functions, then it seems there must be some way to bound the complexity of the functions wherever they are defined. A simple way to do this is via the notion of h-honesty.

<u>THEOREM</u> <Blum and Blum,1975> For every 2-place total recursive function h, ∃ M, uniformly in h, such that M identifies the class of h-honest functions. ▯

This is called "A Priori Inference" since the idea is to use the a priori bound that h provides on the complexity to disallow any hypothesis that "takes too long" to compute the current partial enumeration. The converse statement is also true and is stated later.

The methods used in the demonstration of the various results entail relatively enormous amounts of calculation. That is, they establish the possibility, not the feasibility, of identification for various classes. This is a feature of the subject at present <Coy,1979> and is discussed further in the "Implementations" section of the next chapter.

Whether there can EVER be efficient, practical inductive inference machines that identify non-trivial classes of functions is an issue that, to avoid too great a

---

1 Although Wiehagen <1978> claims that most of his results with respect to R can be duplicated for P.

digression, is only touched on very briefly here. Many of the good encoding references cited in Chapter One bear upon this question. An early result by Gold <1967> states:

<u>THEOREM</u> If M is an identification by enumeration machine (of the sort described earlier) that identifies a class of total recursive functions C, then there is no inductive inference machine M′ identifying C such that:

1) for all f ∈ C, if M converges to a proper index for f given some partial enumeration from some enumeration of f, then so does M′
2) For some g ∈ C M′, given some partial enumeration from an enumeration of g, converges to a proper index for g while M does not.

The maximum number of hypothesis changes involved in the identification of any function in a given class is investigated in <Barzdin and Freivald,1972; Barzdin,1974>. Since the worst case behavior approximates trying every function in the class Barzdin gloomily concludes that input/output listings do not suffice to design economical inductive inference machines, and goes on to suggest ways of improving efficiency through additional information such as program histories, or by changing the character of the inductive inference machine from a total to partial function (there is a class of functions for which the latter requires arbitrarily fewer changes to succesfully perform an identification <Barzdin and Freivald,1972>). That the problem of identification is likely to be insoluble in practical terms without some clever modification is also suggested by:

THEOREM <Gold,1978> The determination of whether there exists a deterministic finite state automaton of at most t states compatible with a given partial enumeration is NP-complete.

Angluin <1979> shows that analogous results hold even when constraints are placed upon the "density" of the partial enumeration, and surveys the general question. Although initially pessimistic about the possibilities of a practically worthwhile inductive inference machine, Angluin <personal communication> is currently hopeful that such machines may yet be designed for very special, yet useful, classes. Pudluk <1975> shows that NP-complete problems exist in the logics of discovery mentioned earlier.

Considerations with respect to a candidate solution's "complexity" often go hand in hand with analyses of the potential efficiency of discovery procedures <e.g. Kinber,1974>. Again the discussion here will be extremely brief. There are two distinct conceptions of program minimality employed, corresponding to the distinction between the "size" of programs in some representation versus their "efficiency" <Hartmanis and Hopcroft,1971>. For example, "size" might be measured by the number of symbols in a program's description or its position in some standard numbering of P. Size measures are customarily labelled "intrinsic". "Efficiency" is defined in terms of some computational complexity measure with respect to time or space. Efficiency measures are customarily labelled "derivational". "Total" measures are given by some function (usually linear) of both the intrinsic and derivational measures. The impact of minimality requirements on a solution is frequently investigated for intrinsic measures (e.g. the inductive inference machine must settle upon not only a correct index, but the least correct one in some ordering of P) <Schubert,1974; Freivald,1975>. <Feldman et al.,1969; Feldman,1972; Feldman and Shields,1977> consider minimal identification with respect to total complexity (specifically wrt size and run times). Such studies shade into those exclusively concerned with good encoding.

2.2.1.1 "Consistent" and "Reliable" Identification

Upon reflection, one realizes that the requirements for identification in the limit permit several possibly undesirable types of solution. Although an inductive inference machine must ultimately hypothesize a correct program index for any function that it identifies, the interim hypotheses may be totally bogus, as may be the responses to functions that M cannot identify. For example:

> The self describing functions can be identified by the, in a sense trivial, machine M that outputs anything until, if ever, a tuple of the form (x,1) appears in a partial enumeration, and thereafter outputs the smallest such x that appears in any of the partial enumerations.

2.2.1.1 "Consistent" and "Reliable" Identification

Until M reaches the correct value of x for a self-describing function, a program hypothesized by M may not even agree with the current partial enumeration, that is M's hypotheses may not even be compatible with the available data. Moreover M converges to incorrect hypotheses for many non-self-describing functions.

Machines that avoid these features are called "consistent"[1] and "reliable"[2] respectively.

<u>Definition</u>: An inductive inference machine M is <u>consistent</u> if (M identifies a partial recursive function f) implies (the program $M([\hat{f}_n])$ is compatible with $\hat{f}_n$ $\forall$ n).

<u>Definition</u>: An inductive inference machine M is <u>reliable</u> on a class of functions C, if, for every enumeration $\hat{f}$ of each $f \in C$, M converges iff M identifies f.

Reliability and consistency are intimately related:

<u>THEOREM</u> <Blum and Blum,1975> For any inductive inference machine M, if M is consistent then M is reliable on P, and if M is reliable on P then $\exists$ M', uniformly in M, such that M' is as powerful as M, and M' is consistent.

To obtain either consistency or reliability it is a necessary and sufficient condition that M identify the class of finite functions <Blum and Blum,1975>.

It is perhaps reasonable to hope a priori that some

---

[1] "Overkill" <Blum and Blum,1975>, "feasible" <Gold,1976>, and "regular" <Kinber,1974> are also used.

[2] "strong" <Minicozzi,1976> is also used.

effective method exists to ensure that an inductive inference machine is consistent (reliable on P), since for example, the large class of enumeration machines are consistent by construction. This is not possible however. The class of self-describing functions provides an example of a class that cannot be identified by a consistent machine (this follows immediately from the proof of the Non-Union Theorem, given in Section 2.2.1.2, and the previous equivalence to reliability).

<u>THEOREM</u> <Case and Smith,1978> There is no algorithm that, given an inductive inference machine M, specifies a function that M fails to identify.

A numbering theoretic characterization of the classes of functions that can be identified by a machine reliable on P (i.e. consistent) is to be had in <Wiehagen,1977>. The following characterization is given in terms of the complexities of the functions involved; its converse was stated previously as the method of "A Priori Inference".

<u>THEOREM</u> <Blum and Blum,1975> If a class C of functions can be identified by a machine M reliable on P then $\exists$,uniformly in M, h such that f $\in$ C implies that f is h-honest, for h a total recursive, 2-argument function.

This theorem operates in much the same manner as Wiehagen's result stated earlier. The key factor is that if a function is h-honest, then the complexity of some extension is bounded by the maximum of some constant (from the "almost everywhere" condition) and h(x,f(x)). By enumerating through tuples of the form (i,c), where i stands for a program index and c the sought after constant, and checking whether or not program i is compatible with the data or requires computational time exceeding the above allowable bound, one must eventually settle on a program for some extension of f as desired. And, of course, if a machine M identifies a class of functions C and M is reliable on P then the requisite h-honest function is given by h(x,y)= (the maximum complexity encountered by any machine hypothesized by M, given a partial enumeration each of whose elements is < (x,y), when working upon the input values of these partial enumeration elements).

This shows, for example, that reliable (on P) machines cannot identify arbitrarily complex 0-1 valued recursive functions.

Reliability on R is characterized by the formulation of the converse to the method of "A Posteriori Inference" mentioned previously, along exactly the same lines as the result just explicated except using general recursive operators rather than total recursive functions. Machines reliable on R can be much more powerful than those reliable on P or even the class T of total functions. And machines reliable on Pinf = P-{f : f is a finite function}, while more powerful than consistent machines, are nevertheless not as powerful as those reliable only on R (for which some arbitrarily difficult to compute functions may be identified) <Blum and Blum,1975>.

To summarize then: NUM ⊂ IDconsistent = IDreliableonP ⊂ IDreliableonPinf ⊂ IDreliableonR ⊂ ID; IDknown ⊂ IDconsistent; and IDreliableonT ⊂ IDreliableonR, where the containments are strict.

Consistency is evidently a rather strong requirement to place upon an inductive inference machine. A related but less stringent requirement that nevertheless prevents an inductive inference machine from "contradicting the evidence", is called "conformability" by Wiehagen <1978>. A machine M is "conformable" if M's hypotheses are always either compatible with each partial enumeration or are possibly undefined at some of the data points. The power of conformable machines is strictly between that of unrestricted and consistent machines.

2.2.1.2 Communal Identification

The previous material indicates that alone, any inductive inference machine has definite limitations. Yet such a lone machine may adequately model neither a scientific nor a linguistic community. The Non-Union Theorem states that:

<u>THEOREM</u> <Blum and Blum, 1975> {self-describing functions} U {finite functions} is not identifiable.

This is despite the obvious identifiability of these two sets individually. A simple way to see the truth of this is that such a machine would have to be consistent (since it identifies the finite functions) yet consistency is unattainable for any machine that identifies the self describing functions. There is, therefore, a difference in the power of an individual versus that of a collection of individuals.

This difference in power exists only for unreliable machines.

<u>THEOREM</u> <Minicozzi,1976> Given any recursively enumerable class M of inductive inference machines, each of which is reliable on a class C of functions, then ∃, uniformly in M, an inductive inference machine M′ that is reliable on C and is as powerful in C as any of the machines belonging to M. The idea behind the machine M′ implementing this "Union Theorem" is to gradually feed a function′s enumeration to more and more of the machines in M, and by checking to see whether or not a given machine′s last two hypotheses are the same, try to settle on a machine that is converging.

The informal idea of a group of inductive inference machines identifying a function has been made precise in two quite disparate ways: First by the requirement that some machine, the group′s "expert" for that function, identify the function. And second by the requirement that almost all of the machines identify the function in the limit. Case and Smith <1978> investigate the consequences of the first conception for static, finite groups of inductive inference machines; while Schubert <1974> and Kugel <1977>, via Schubert′s notion of 2-limiting recursive strategies, in effect utilize the second for expanding or potentially infinite groups.[1] In <Case and Smith,1978> a number of claims are made relating the classes of functions that can be (almost everywhere) identified with n machines restricted to at most m hypothesis changes (m possibly unbounded) and (almost everywhere) matched (see next subsection) with n machines. Of special relevance here are the claims that: 2n+2 machines allowed only m discrepancies (cf. 2.3) in the solution have greater identification power than n+1 machines allowed m+1 discrepancies; The identification power of n+2 machines not allowed any discrepancies is greater than the union, over d ∈ N, of the matching powers (cf. 2.2.2) of n machines allowed d discrepancies; and MATCH (cf.2.2.2) is larger than the union, over n ∈ N, of the identification powers of n machines allowed any (finite) number of discrepancies.

2.2.2 Matching

Identification in the limit requires that an inductive inference machine converge to a particular correct program for a target function. Matching [2] requires only that almost all the hypotheses are correct, i.e.

<u>Definition</u>: An inductive inference machine M <u>matches</u> a

---

[1] And k-limiting recursive strategies generally embody conceptions of communal or supra-communal identification.

[2] <Feldman et al.,1969> is the first use of both the notion and name. The idea appears elsewhere <e.g. Barzdin and Freivald,1972; Case and Smith,1978> although the exact definitions and names employed vary.

function f, if for any enumeration $\hat{f}$ only a finite number of $M([\hat{f}_n])$, for n=1,2,3..., are not program indices for f.

Definition: The matching-power of an inductive inference machine M is the class of all sets of functions that can be matched by M.

Definition: MATCH is the class of sets of total recursive functions that are matchable.

THEOREM <Barzdin,1974> MATCH strictly includes ID.
Any class of "almost everywhere identifiable" (defined later in this chapter) but not identifiable functions, provides an example of a class of functions that can be matched yet not identified in the limit.

Yet matching is curiously similar to identification in that, for example, a re-examination of the argument used to demonstrate the impossibility of any machine that identifies R, reveals that by virtually the same argument:

THEOREM <Feldman et al.,1969> R is not matchable.

Neither matching, nor the model to be outlined next, "extrapolation", is as fully developed as identification. Consequently, many of the issues in identification have not been investigated in these contexts. No concept of reliability exists. <Feldman and Shields,1977> is one of the few papers on matching in the presence of complexity constraints. Consistency , on the other hand, can always be guaranteed, trivially[1], and so is never mentioned.

2.2.3 Extrapolation

To be compatible with the original problem statement, an inductive inference machine must discover a name for f. Both identification and matching have assumed that the inductive inference machine must explicitly generate such names as hypotheses. However, extrapolation rests upon a subtler interpretation, namely that at some point in the enumeration the inductive inference machine must itself have become a name for a function that is almost everywhere equal to the target function. In intuitive terms, the difference is that between the linguist who constructs explicit grammars, and the child who is merely seen to obey some grammar.

<u>Definition</u>: A <u>total</u> <u>enumeration</u> of a function f is an enumeration of f for which every element is of the form $(x_i, f(x_i))$.

<u>Definition</u>: A <u>query</u> <u>partial</u> <u>enumeration</u>, $q\hat{f}_n$, of a function f, is the finite sequence consisting of (the first n-1 elements of a total enumeration of f) concatenated with $(x_n, ?)$ where $(x_n, f(x_n)$ is the nth

---

[1] Suppose M matches f. Define M′ by the following program description: Given f_n, calculate $i = M([f_n])$ and output an index for the program $t = \lambda x [\, f(x)$ if $(x, f(x)) \in f_n$; $t_i(x)$ otherwise].

element of the particular total enumeration of f in use.

Definition: An inductive inference machine M extrapolates a function f if for every total enumeration of f $\exists$ n such that $M([q\hat{f}_m])=f(x_m)\ \forall\ m>n$. [1]

Definition: EXTRAP is the class of sets of total recursive functions that are extrapolatable.

Extrapolation predates both identification and matching and has particularly close ties with the new computational models of randomness mentioned in Chapter One <Solomonoff,1964>.

The relationship of extrapolation to identification is simple.

THEOREM <Case and Smith,1975> A set S of functions can be extrapolated iff S can be identified in the limit by a Popperian machine.

EXTRAP can also be characterized in ways similar to those used for ID, namely:

THEOREM <Barzdin and Freivald,1972> A class C of functions $\in$ EXTRAP iff C is included in a recursively enumerable class of total recursive functions.

---

[1] This concept has been variously defined. For example, in <Blum and Blum,1975> it is defined with respect to increasing enumerations. <Barzdin and Freivald,1972> define it much as it is defined here.

<u>THEOREM</u> <Blum and Blum,1975>

i) If an inductive inference machine M extrapolates a class C of functions, then ∃ , uniformly in M, a total recursive function h such that $f \in C$ implies f is h-easy.

ii) If h is total recursive, then ∃, uniformly in h, an inductive inference machine M such that f is h-easy implies M extrapolates f.

To obtain the function h from M it suffices to note that from the recursive function that enumerates C, h can be defined as h(x)=(the maximum complexity involved in the computation at x by any of the first x machines enumerated). Conversely, the procedure for obtaining M from h follows the pattern seen several times before. That is, to compute $M([q\hat{f}_x])$, begin enumerating all tuples (i,n), where i stands for a program index and n for the complexity bound adjustment induced by its "almost everywhere" nature. Look for a combination for which both $T_i(y) \leq \max(n,h(y))$ $\forall y \leq x$ and t_i is compatible with $q\hat{f}x$. If and when found, output $t_i(x)$.

<u>Corollary</u> EXTRAP is strictly included in ID.

From the preceding characterization it can be seen that the class of "step-counting" functions, for example, cannot be extrapolated, yet is easily identifiable by listing P and calculating only for the "time" supplied by the partial enumeration.

Here perhaps it should be noted that the definitional variants said to have no effect upon the possibility of identification, may very well affect the other models.

<u>THEOREM</u> <Barzdin and Freivald,1972> EXTRAP includes ID for partial recursive inductive inference machines.

No proof accompanied this assertion. In fact, the inclusion is strict since:

<u>THEOREM</u> EXTRAP includes MATCH for partial recursive inductive inference machines.

<u>Proof</u>: Suppose C $\in$ MATCH, f $\in$ C, and M is a machine that matches C.

For $q\hat{f}_n$: Let $M([\hat{f}_{n-1}])=i$. (Wnlg M may be assumed total.)

Output $t_i(x_n)$ if the computation halts.

By the definition of matching $\exists$ N such that $n \geq N$ implies $t_i=f$, and so the extrapolated values past this point are both defined and correct.//

With the definition of almost everywhere identification and matching in the next subsection, it becomes clear that this containment is also strict, since the same method seems to work for showing containment of the classes corresponding to these approximate learning criteria in EXTRAP.

In a fashion similar to that for identification, the difficulty of extrapolation has been estimated by bounding the maximum number of erroneous answers given while extrapolating any function within the class <Barzdin and Freivald,1972>. But there has been much less work for extrapolation as compared to identification on the effect of definitional variants and the potential difficulty of the task.

2.3 Approximate Variations of the Main Models

The previous models are linked by the requirement that an inductive inference machine output nothing but completely correct hypotheses past some point in any enumeration. Approximate learning relaxes the "completely correct" prerequisite to varying degrees. This is desirable since, for example, R is not learnable with respect to any of the main models.

Perhaps the simplest yet least satisfactory thing to do is to select some "priveleged" finite subset S of the natural numbers and consider any function that agrees with the target function f on S to be a "suitable" name for f. Since this arises in the context of language identification <Wharton,1974> its discussion is deferred until Chapter Three.

Hypotheses that are guaranteed to agree with the target function only on some finite domain seem rather unsatisfactory. Hypotheses that disagree with the target function at only finitely many places perhaps have more appeal. This is what "almost everywhere" identification and matching[1] permit.

Definition: Given two functions f and g, the discrepancies between f and g are those x ∈ Domain(f) such that $f(x) \neq g(x)$.

Definition Given two functions f and g, $f =_n g$ for n ∈ N if ∃ at most n discrepancies between f and g. $f =_* g$ if ∃ n such that $f =_n g$.

Definition: An inductive inference machine M almost everywhere (d) identifies a function f in the limit if for every enumeration of f ∃ i such that M converges to i, and i is an index for a program that computes some extension of a function g such that $f =_* g$ ($f =_d g$). Almost everywhere matching is defined analogously.

Definition: ID_*, ID_d, $MATCH_*$, $MATCH_d$ are defined as are ID and MATCH except that the words "almost everywhere (d)" are inserted in the pertinent locations.

Permitting even a single discrepancy between the target and hypothesis results in more powerful inductive inference machines.

THEOREM <Case and Smith,1978> ID_{n+1} strictly includes ID_n ∀n ∈ N.

A set very similar to the self describing functions establishes this for n=1 by being almost everywhere (1)

---

1 These are also known as "sub-identification" <Minicozzi,1976>, "anomalous explanatory" and "behavioral" "identification mod ≤ n anomalies" <Case and Smith,1978> respectively.

identifiable but having at least one function, for every inductive inference machine M, that is not identifiable by M. This corroborating set, S, is the set of all recursive functions whose value on 0 is an index for a program that computes f at all save perhaps one point. S is trivially almost everywhere (1) identifiable. However given a machine M, a function f ∈ S that M does not identify can be (ineffectively) constructed:

This f is defined by the program that outputs its own index at 0 (via the recursion theorem) and is defined elsewhere by a program that constructs an ever larger input/output finite sequence containing a single "anomaly" (i.e. an x value for which no (x,y) value is given in the infinite sequence), moving it iff the definition of some y value at that x will cause M's last hypothesis to be incompatible with the new finite sequence, or if the new finite sequence causes M to output a new hypothesis. If the anomaly never settles, then by construction M never converges and so does not identify the function that the infinite sequence enumerates. However, if there is some anomaly at which no y-value definition can either force M to change its mind or be wrong in its last hypothesis, it must be because M's hypothesis is undefined at that point. In this case the function that is identical to the function defined by the infinite sequence constructed EXCEPT that it equals 0 at the anomaly, is a function that M misidentifies. Yet this function is still clearly almost everywhere (1) identifiable as before.

An extension of this method leads to:

<u>THEOREM</u> <Case and Smith,1978> ID_* strictly includes $\cup\ ID_n$, $n \in N$.

And the corresponding strict containments hold for matching also i.e.:

<u>THEOREM</u> <Case and Smith,1978> $MATCH_{n+1}$ strictly includes $MATCH_n$, for $n \in N$.

<u>THEOREM</u> <Case and Smith,1978> $MATCH_*$ strictly includes $\cup\ MATCH_n$, $n \in N$.

So great is the power conferred by the acceptability of a finite number of discrepancies between the target and hypothesis that strong constraints such as reliability[1] can be imposed and still permit powerful identification results.

<u>THEOREM</u> <Minicozzi,1976> $\exists$ S such that S is almost everywhere identifiable by a machine reliable on P, yet S is not identifiable.

Minicozzi infers the existence of such sets from:

<u>THEOREM</u> <Minicozzi,1976> If a set S of functions can be identified and $\exists$ a machine reliable on P that almost everywhere identifies precisely S, $\exists$ a machine reliable on P that identifies S.

Despite this however, it is not possible to almost everywhere identify R. In fact:

---

[1] i.e. convergence implies almost everywhere identification

2.3 Approximate Variations of the Main Models

<u>THEOREM</u> <Case and Smith, 1978> MATCH includes ID_*
This is easy to see since from a machine M that almost everywhere identifies a class C of functions, M′ can be defined which takes the output of M, splices in a table containing the current partial enumeration values, and outputs an index for the resulting function. Eventually the last of the discrepancies must have gone past in any enumeration, and past that point M′ outputs completely correct hypotheses.

<Blum and Blum, 1978> gives a general condition on the complexities of the functions in a class C which is sufficient for C to be almost everywhere identified by a machine reliable on R. ID_* is characterized in <Wiehagen, 1978> as follows:
It is assumed that the complexity measure used results in complexity classes R_t satisfying the condition that [(f $\in$ P_t iff g $\in$ R_t, holds $\forall$ f, g, t $\in$ P such that f(x) = g(x) for almost all x.]

<u>THEOREM</u> <Wiehagen, 1978> A class C of total recursive functions is almost everywhere identifiable iff $\exists$ an effective operator O such that C $\subset$ $\{t_i : T_i(n) \leq O(t_i)(n)$ for almost all n$\}$ $\cap$ R.

Case and Smith <1978> note that the power of almost everywhere (d) identification is a consequence of the following facts:

1) the permissible discrepancies between target and hypothesis include those where the hypothesized program

is not defined rather than merely defined differently from the target at some point

2) the exact number of discrepancies needed for any particular function is unknown.

Any definition of almost everywhere identification that denies either of these two conditions is reducible to identification.

Increased power is not the only advantage sought by settling for an approximation rather than a replica of the target function. Approximate learning may provide a means to escape from the explosive computational problems seemingly inherent in the implementation of inductive inference machines. Almost everywhere identification is "simpler" than identification with respect, for example, to the the maximum number of hypothesis changes necessary for a partial recursive inductive inference machine to learn any of the functions in a class C.

<u>THEOREM</u> <Case and Smith, 1978> {sets of functions that can be almost everywhere n+1 identified with 0 hypothesis changes} strictly includes {sets of functions that can be almost everywhere n identified}, for $n \in N$.

Note that partial recursive inductive inference machines are assumed here.

In general it is the case that the classes of functions which can be almost everywhere a identified by machines restricted to b hypothesis changes form a lattice under set inclusion.

2.3 Approximate Variations of the Main Models

THEOREM <Case and Smith,1978> If C_i is the class of functions which can be almost everywhere a_i identified by machines restricted to b_i hypothesis changes, then $C_i \subset C_j$ iff $(a_i \leq a_j)$ and $b_i \leq b_j)$, where a_i and $b_i \in N$.

Almost everywhere identification lends itself to the notion of learning variants of already learnable functions. The approach discussed below of beginning with an identifiable class and "blowing it up" appears later in Chapter Five for the fuzzy variants of identification.

Definition: f is a finite variant of a function g if g $=_*$ f.

Notice that finite variants of recursive functions are always recursive. This is not the case for the variants defined in Chapter Five.

THEOREM <Minicozzi,1975> If M (almost everywhere) identifies a class C of partial recursive functions, and M is reliable on P, then $\exists$, uniformly in M, M′ that (almost everywhere) identifies the finite variants of C, and is reliable on P.

Minicozzi <1975> also investigates the effect of "recursive variants" (i.e. derived by composition with some known, 1-1 recursive function). Her investigations in this area with respect to the "constant bounded functions" (i.e. those with finite range) may have special relevance for learning language variants since the functions involved there (cf. 3.1) are constant bounded.

At the time of writing, <Mellish,1978> is the only attempt to define a notion of approximate learning which permits an infinite number of discrepancies between the target and successful hypotheses. This is done in the context of the extrapolation of nonrecursive functions by total recursive functions. Before describing these results, a brief discussion of the problems associated with such functional approximations will be helpful.

Similarity between the number theoretic functions employed in this area is based upon the points of non-equivalence, and not, for example, the continuous measures to be found in numerical analysis <Isaacson,E.>. To judge how different two functions f,g are, the "size" of $\{x : f(x) \neq g(x)\}$ must somehow be measured. Yet as soon as f and g are permitted to differ for infinitely many points in their domains, formidable conceptual problems arise in trying to measure this. The problem is that of measuring the relative sizes of two countably infinite sets - those points where f, g coincide versus those points where they do not. The obvious method, that of taking the limiting percentage of the one set's members in arbitrary joint enumerations of the two sets, clearly does not work, different enumerations being capable of producing arbitrarily different answers. Thus if f, g agree on the even numbers and disagree on the odd, then the enumeration (1,2,3,4,5,...) gives 1/2 as the relative agreement of f and g, whereas (1,3,2,5,7,4,...) yields the answer 1/3.

The most pertinent source of solutions to these problems appears to be in the studies, notably <Rose and

Ullian,1963>, <Tsichritzis,1969; 1971> and <Lynch,1974>, which seek to weaken the concept of constructiveness by approximating (arbitrary) functions with recursive ones. <Ausiello and Protasi,1975> analyzes the different notions of approximation, showing their inter-relationships and relating them to the "global" approximations provided by limiting recursion. Informally stated the situation is that regardless of the exact definition of approximation used $\exists$ functions that are not approximable by recursive functions, and that the classes of approximable functions corresponding to the varying definitions are incomparable.

The example of the even and odd integers is revealing. The "standard enumeration" (1,2,3,...) of the possible domains leads to the intuitively acceptable measure of similarity. Perhaps this is why two of the three studies cited above consider the standard enumeration as the arbiter. The definitions used here also reflect this acceptance.

<u>Definition</u>: Given two functions f and g, $AGREE(f,g,n)=\{x : f(x) = g(x) \text{ and } x \leq n\}$.

<u>Definition</u>: A class C of total recursive functions is <u>continuous</u> if any n-tuple of integers forms the first n values of a least one $t \in C$.

Both of the following theorems by Mellish <1978> are stated with reference to a class C that is some (arbitrary) continuous recursively enumerable subset of R. For simplicity write $\lim_n \inf (AGREE(f,r,n)-pn)$ as $A(f,r,p)$ [1] and denote by m the function evaluated by an extrapolating inductive inference machine M.

<u>THEOREM</u> $\forall\ 0<p<1\ \exists$, uniformly in p, an inductive inference machine M such that if for a function f $A(f,r,p) > -\infty$ for some $r \in C$, then M can "approximately extrapolate" f in the sense that $A(f,m,p) > -\infty$, although it may be different from the earlier lim inf.

<u>THEOREM</u> $\forall\ p>0\ \exists$, uniformly in p, an inductive inference machine M such that given any function f M can "approximately extrapolate" f in the sense that $\lim_n \inf (AGREE(f,m,n)/n) \geq \lim_n (AGREE(f,r,n)/n) - p \quad \forall\ r \in C$ provided the limit exists.

---

[1] Mellish implicitly assumes that p is a computable real number. However, only minor alterations of his proofs are necessary in the general case since an initial (ineffective) construction of a p′ by a suitable inversion-truncation-inversion or truncation of p suffices to reduce the problem to one for computable p.

Chapter 3

LEARNING LANGUAGES

3.1 Introduction

A formal language, L, is defined to be any set of finite strings composed from some finite terminal vocabulary V_t, i.e. $L \subset V_t^*$ <Hopcroft and Ullman,1969>. While not departing from this definition of L, this thesis considers L as the extensional definition of its characteristic function. Although constituting a simple shift in perspective, this permits the clarification of the relationship of functional to linguistic learning, and the provision of a straightforward generalization of the standard language learning material to fuzzy languages.

The characteristic function ch of a formal language need *not* be recursive. However, it is customary to assume that the languages dealt with are at least generated by a Type 0 grammar[1] so that at worst there is a partial recursive function that is identical to ch for all strings s

---

[1] Until the development of limiting recursion this assumption was mandatory to even make sense of the problem statement since for non-Type 0 languages there was no finite encoding device or name to be discovered. The normal form indexing for limiting recursive functions <cf.Crisculo et al., 1975> may have changed this.

such that ch(s)=1, and is undefined elsewhere. In other words there is always a semi-characteristic function sch for L.

Both the characteristic and semi-characteristic functions can be used to name a formal language in Chomsky's problem. They correspond to Gold's <1967> "tester" and "generator" naming schemes. He proves that if identification in the limit is possible given naming scheme N_1, then it is possible given naming scheme N_2 if there is a limiting recursive translation from N_1 to N_2. The actual names used for both tester and generator naming schemes in <Gold,1967> are partial recursive function indices. There is then an obvious recursive translation of testers to generators, yet even for recursive languages there is no limiting recursive translation in the opposite direction.

Before proceeding, the basic scenario for language learning employed in this thesis should be sketched rather more precisely. Partial enumerations drawn from an enumeration of either the characteristic or semi-characteristic function of a language L are input to an inductive inference machine M. So, for example, if L = {ab, aa}, then M might receive an input sequence like ((ab,1),(a,0),(aaa,0),...). To identify L in the limit, say, M must converge to a name for either the characteristic or semi-characteristic function of L.

Thus far the language learning scenario should seem little different from the previous function learning material. Indeed it may appear to be so obvious an extension as to scarcely merit separate statement. If so, it can come as something of a shock to realize that it is not the one traditionally employed in language learning studies. The usual scenario <cf.Gold,1967; Wiehagen,1977> presents a language as a function of time. That is, the inductive inference machine is presented with sequences of the form $((1,\pm s_1),(2,\pm s_2),\ldots)$ where $+s$ implies that $s \in L$, and $-s$ that $s \notin L$, and the first variable is taken to refer to discrete time intervals. While not affecting the basic results, this can lead to some minor differences with respect to the various types of enumeration [1], and does not emphasize the parallels between the lingusitic and functional studies.

However, despite the basic unity between the two fields, differences arise because:

*Any extensions to partial functions are deemed permissible for function learning, but not for language learning.

*In function learning, the function that must be correctly named is the function that is enumerated. For languages there are two possible functions, i.e. the characteristic and semi-characteristic functions, either one of which the inductive inference machine can be

1 A few examples of this and a terminology difficult to disentangle from this traditional approach has meant that few citations of <Wiehagen,1977> appear here.

required to name on the basis of enumerations of the other.

I will elaborate.

Four situations determine the analysis of the task of learning a recursively enumerable language L on the basis of some enumeration E:

1) ch_L is recursive and E is of ch_L;

2) ch_L is recursive and E is of sch_L;

3) ch_L is not recursive and E is of ch_L;

4) ch_L is not recursive and E is of sch_L.

With respect to the <u>tester naming scheme</u>:

<u>Case 1</u> is exactly that of learning the function ch_L as in Chapter 2.

<u>Case 2</u> is a new problem, that of learning the function ch_L as in Chapter 2 with only partial information.

<u>Cases 3 and 4</u> are impossible to solve, by definition.

With respect to the <u>generator naming scheme</u>:

<u>Case 1</u> can be transformed to the task of performing Case 1 with respect to the tester naming scheme and subsequently deriving sch from ch.

<u>Case 2</u> is a new problem whose solution can sometimes, but not always, be obtained as in Case 1 above (i.e. Case 2 is easier for a generator than for a tester).

<u>Cases 3 and 4</u> are essentially problems of learning sch_L as in Chapter 2, but with one crucial difference. Chapter Two's acceptance of hypotheses that compute extensions to the target partial function, correspond

here to the acceptability of hypotheses generating super-sets of a target language. This is patently unsatisfactory (since a universal grammar generating all of V_t* would then be a general solution). Only some extensions of a language's generation are permissible, namely all 0-extensions.

The relationship between functional and linguistic studies has been further muddied by the fact that since Gold <1967> the studies in language learning have normally used Chomsky Type grammars, rather than program indices, to name languages. When Type 0 grammars are used, this has no effect on the analysis (and is operating within the generator naming scheme) since there is an obvious recursive translation from such grammars to the indices of partial recursive functions (where, of course, the partial recursive functions are now taken to be functions from V_t* to N rather than the usual N to N) and vice versa.

<u>THEOREM</u> <Hopcroft and Ullman,1969> If L is generated by a Type 0 grammar G then ∃, uniformly in G, a partial recursive function index i such that $t_i = sch_L$. Conversely given any (0-1 valued) partial recursive function index i ∃, uniformly in i, a Type 0 grammar G such that $t_i(s)=1$ iff $sch_L(s)=1$.

It is the use of Type 1,2,or 3 grammars as the sole permissible names for the language being learned that is initially so confusing. Of course, such a restriction of the "Hypothesis Space" (cf.3.2) trivially implies that only Type 1,2, or 3 languages, respectively, can be learned. More significantly it destroys the distinction between the generator and tester naming schemes since such grammars not only generate the language but also permit membership to be decided algorithmically.

<u>THEOREM</u> <Hopcroft and Ullman,1969> G is a context sensitive grammar implies ∃, uniformly in G, r ∈ R such that $ch_L(G) = r$.

So in other words, work dealing with such grammars appears to operate within the generator naming scheme while actually working within the tester naming scheme. Even this statement should be modified slightly however, since any class of Chomsky grammars is recursively enumerable and consequently there are total recursive characteristic functions that are not the characteristic function of any context sensitive language. Furthermore, each class of Chomsky grammars determines a certain "complexity class" much as those used in the previously surveyed material in <Blum and Blum,1975> <cf.Hopcroft and Ullman,1969>. In short then, such a Hypothesis Space automatically restricts attention to certain recursively enumerable complexity classes of total recursive functions.

The introduction of a different way of naming languages customarily entails a special investigation. So, for example, transformational grammars are treated in <Hamburger and Wexler,1973a; 1973b; 1975>, regular bilanguages in <Pair,1976>, transition network grammars in <Chou et al.,1976>, VL decision rules in <Larson et al.,1977>, and L-systems in <Coy and Pfluger,1979>.

This chapter in general continues Gold's <1967> use of partial recursive function indices as language names.

3.2 The General Framework

As just noted, the material of the previous chapter transfers directly to the more general problem of learning languages. Both Chapter Two's results and those peculiar to language learning studies are illuminated when viewed in terms of differing specifications along the seven dimensions of:

1. The Naming Scheme
2. The Hypothesis Space
3. The Sample Presentation
4. The Inductive Inference Process Allowed
5. The Fundamental Limiting Criterion
6. The Secondary Limiting Requirements
7. The Interim Constraints [1]

The <u>Naming Scheme</u> has already been discussed at some length for languages. In passing, it seems rather remarkable

[1] This is essentially the breakdown given in <Biermann et al., 1972>. That survey did not recognize the influence of #1, #4 or #6, and omitted aspects of #5 and #7 (e.g. extrapolation, consistency).

that the programs-for-extensions naming scheme should not have been challenged or even received the most cursory of examinations in the functional setting.

The *Hypothesis Space H* [1] is a given set of language names[2] from which the inference process must choose its hypotheses. In a sense it represents a minimal rationalist or Chomskian concession in what is otherwise an empiricist analysis. As indicated previously with respect to Chomsky grammars, the hypothesis space can affect the general problem profoundly by providing the general form of the target language (for example, indicating whether it is regular or what its complexity requirements are) and, by serving as an a priori framework, facilitating or impeding the search for a language's name. It is a concept that although relevant to functional learning, usually does not receive explicit treatment in that context, presumably being either P or a recursively enumerable subset of R. Since language learning studies conventionally take place under the aegis of "grammatical inference", a commonly employed hypothesis space is, for example, the set of all Context Free grammars. The most general hypothesis space is the set of Chomsky Type 0 grammars, or P.

---

[1] Note that this concept has not been applied to extrapolation.

[2] Names and names of names are deliberately conflated here. It is as if H contained subsets of P or R or the Chomsky grammars, rather than N. See the various papers by Barzdin for some consideration of the features obscured by this.

The hypothesis space within which an inductive process P is constrained to operate should be distinguished from P's power, although they are often the same by construction. So, for example, a Popperian machine has R as its hypothesis space but can only identify recursively enumerable subsets of R.

The possible solutions to the language learning problem intuitively appear to depend upon such things as whether the hypothesis space H contains a name for the target language, whether H is recursively enumerable, the decidability of the members of H (take for example the difference between an H containing only the strictly Type 0 grammars for the class of regular languages, versus that containing the Type 3 grammars), and so on. A very common assumption is that a Hypothesis Space H is <u>admissible</u>, i.e. that H is recursively enumerable, and the members of H are decidable.

The <u>Sample Presentation</u> refers to what, in the previous chapter, was called the "enumeration". As indicated in the previous section, the issue is slightly more complex here, since decisions must be made not only as to which class of enumerations the inductive process should be successful upon, but also as to whether to represent a language by its characteristic or semi-characteristic function. The latter decision leads to the central distinction for language learning, that between "text" and "informant" sample presentations.

<u>Definition</u>: A sample presentation S for a language L is

(arbitrary) <u>text</u> if S is an enumeration of the semi-characteristic function of L.

<u>Definition</u>: A sample presentation S is (arbitrary) <u>informant</u> if S is an enumeration of the characteristic function of L.

And things such as "primitive recursive text" and "increasing informant" may be defined in the obvious manner.

Sometimes the distinction is made between "complete" and incomplete text (informant). Only complete sample presentations, that is only sample presentations that are enumerations, without omissions, of a language's characteristic or semi-characteristic function, are considered here. Another kind of sample presentation that is not mentioned further, but which arises in language learning studies sufficiently frequently to warrant comment, is that of a "teacher" <cf.Knobe et al.,1976>. This has not yet been adequately formalized. Finally, some researchers have used sample presentations that embody the wish to specify the language in accordance with a certain schedule with respect to the lengths of the strings in the partial enumerations. Such text presentations have been called "effectively quasi-ordered by length" in <Wharton,1974>, and correspond to a methodical enumeration of sch_L.

Text and informant embody the distinction between examples and counter-examples so commonly employed in pattern recognition and linguistically oriented studies

<cf. Fu, 1974>. If a pattern or language is thought of as its characteristic function, then text presents the inductive inference machine M with all and only the members of the pattern or language, whereas informant presents M with both members and nonmembers labelled as such. Since text is just the enumeration of a partial recursive function f it is always possible to generate it algorithmically (by a standard dovetailing of the individual computations that f performs on each $s \in V_t^*$). Informant obviously cannot be generated algorithmically if ch_L is not recursive.

The <u>Inductive Process</u> is the "mechanism" that generates the hypotheses upon the input of a function's enumeration. As stated previously, this thesis deals exclusively with solitary inductive inference machines and the treatment expresses this. However, communities of inductive inference machines, communities of communities, and so on, as in <Schubert, 1974>, are much more powerful mechanisms for inductive inference.[1] A third possibility is to equip an inductive inference machine with a Bernoulli generator (p=1/2), which also results in a more powerful device <cf. Barzdin et al., 1972; Podnieks, 1975>.

The <u>Limiting Criterion</u> is concerned with the long term

---

1 These mechanisms rapidly exceed what is usually known as inductive inference. It has been suggested that "evolutionary" rather than "inductive" may better describe the process <Schubert, 1974; Fugel, 1977>. The issue is whether inductive inference necessarily is a solitary activity.

behavior of the sequence of outputs emitted by the inductive-inference machine when a language's enumeration is input to it via successive partial enumerations. As for function learning, extrapolation, matching, and identification in the limit, are the major categories for language learning. The choices of whether to use the generator or tester naming schemes, and whether to enumerate the characteristic or the semi-characteristic function, give each of the latter two terms 4 distinct meanings.

Definition: An inductive inference machine M *identifies* a language L in the limit, *with respect to the generator naming scheme*, *given text [informant]*, if for every enumeration of sch_L $[ch_L]$ $\exists$ i such that M converges to i and i is an index for a program that computes some 0-extension of sch_L.

Definition: An inductive inference machine M *identifies* a language L in the limit, *with respect to the tester naming scheme*, *given text [informant]*, if for every enumeration of sch_L $[ch_L]$ $\exists$ i such that M converges to i and $t_i = ch_L$.

The definitions for matching are exactly analogous. For convenience the qualifications ("with respect to..." and "given...") are often omitted when the context is clear.

Feldman <1969; 1972; 1977> investigates a concept he calls "strong approachability", which requires not only that the sequence of hypotheses contain infinitely many repetitions of a correct name, but also that any incorrect name for the language appear only finitely often in the hypothesis sequence.

The <u>Secondary Limiting Requirements</u> refer to features such as reliability, minimality, the number of permissible hypothesis changes, and so on. They too are requirements upon the hypothesis sequence as a whole, rather than upon any single hypothesis. Few of these have been investigated specifically within the linguistic context.

The <u>Interim Constraints</u> deal with such things as the compatibility of individual hypotheses with the current partial enumeration (i.e. consistency), good encodings of current partial enumerations, the decidability of individual hypotheses, and so on.

The precise formalizations dealing with the language learning problem can now be expressed schematically along these dimensions: For a language in some class C, a sequence of partial enumerations of L in accord with some Sample Presentation and Naming Scheme is input to an Inductive Inference Process P. The resulting sequence of P's outputs ("hypotheses"), must satisfy the Fundamental Limiting Criterion as understood in the light of the Naming Scheme, and possibly some Secondary Limiting Requirements. If the Fundamental Limiting Criterion is either matching or identification then each of P's outputs are required to belong to the Hypothesis Space. Moreover each of P's outputs may also be required to satisfy the Interim Constraints.

3.2 The General Framework

3.3 The Effect of the Naming Scheme

It is trivially the case that if a language is not recursive then the tester naming scheme dooms to failure any attempts at identification or matching. Not so obvious is the fact that even for recursive languages the tester naming scheme can have an adverse effect.

<u>THEOREM</u> <Gold,1967> If a class C is identifiable in the limit with respect to the tester naming scheme, given primitive recursive text, then either C does not contain all finite languages or C does not contain any infinite language.

This contrasts sharply with the fact that:

<u>THEOREM</u> <Gold,1967> The class of all recursively enumerable languages is identifiable in the limit with respect to the generator naming scheme, given primitive recursive text.

The first result follows from a proof much like that used to show the non-identifiability of R. That is, given an inductive inference machine M, a primitive recursive function is (ineffectively) constructed that enumerates an infinite language in such a way as to cause M to hypothesize ever larger finite languages, thereby never converging to a correct decision procedure for the entire infinite language. Intuitively this counter-example does not apply for the generator naming scheme since it suffices to enumerate the primitive recursive functions and by checking for compatibility, eventually output the index of the primitive recursive function that is enumerating sch_L (and hence which provides a legitimate name for L with respect to the generator naming scheme). This is in fact the idea used in the proof of the latter result (and has appeared before,in section 2.2.1, for P and primitive recursive enumerations).

Various paragraphs of Section 3.1 are relevant here also.

3.4 The Effect of the Hypothesis Space

Much of the discussion in Section 3.1 is directly relevant here.

The hypothesis space can provide a convenient notation for, and enumeration of, the potential solutions. This is one of the great benefits of hypothesis spaces of, say, context free grammars. Efficient generation of such grammars, while not trivial (cf.Wharton's tests in section 3.6), is facilitated by their comparatively simple structure.

Cleverly chosen hypothesis spaces may be able to circumvent the general limitations on inductive inference machines. Thus, using certain L-systems as hypothesis spaces, it is possible to identify DOL and DPOL given text <Coy and Pfluger,1979>. This does not deny the general results on the "poorness" of text described in section 3.5 since these classes do not contain all finite languages. Nevertheless they are "useful" classes. The search for such restricted yet interesting hypothesis spaces has long been a preoccupation of researchers hoping to cut across the boundaries of the conventionally learnable classes. Hypothesis spaces of certain subsets of the context free grammars <e.g. Crespi-Reghizzi,1971>, Type 3 grammars <e.g. Gold,1972>, certain kinds of Lisp programs <e.g. Biermann et al.,1977>, simple programs containing no loops <e.g. Treister et al.,1978>, and "1-pattern grammars" <cf.Angluin,preprint> have received attention.

In studies on (exact) identification and matching it is usually assumed that a name for the target language occurs within the Hypothesis Space. This is not always the case for the approximate variations. For ϵ -identification (cf.end of next section), given an admissible hypothesis space H generating a class of languages dense in the universal class of languages, then if a name for the target does not occur in H, as $\epsilon \Rightarrow 0$ the sequence of intrinsic complexities of the hypotheses diverges <Wharton,1974>.

3.5 The Effect of Text vs Informant

Despite the example of section 3.3 using primitive recursive text and the generator naming scheme, in general it is impossible to identify or match "large" sets of languages given text.

<u>THEOREM</u> <Gold,1967> If a class C is identifiable in the limit given effective text, then either C does not contain all finite languages or C does not contain any infinite language.

Unlike the situation for primitive recursive text, there is no enumeration of the class of enumerating functions (i.e. R), and so no obvious way to acquire a name for even the target's semi-characteristic function. An exactly analogous result is shown in <Feldman,1972> for matching given recursive text.

A result generalizing Gold's theorem requires several new concepts for its statement.

Definition: A chain of languages is any sequence $C=(L_1,L_2,\ldots,L_n)$ of languages such that $L_1 \subset L_2 \ldots \subset L_n$.

Definition: A chain C is infinite if C has infinitely many distinct members.

Definition: An infinite chain $C=(L_1,L_2,\ldots)$ has a fixpoint F if $\exists$ a language F such that $L_1 \subset L_2 \subset \ldots \subset L_i \subset \ldots \subset F$.

THEOREM <Coy and Pfluger,1979> If a class C of languages contains an infinite chain of languages and its fixpoint, then C is not identifiable given text.

The dilemma at the root of the problems with text is that it does not distinguish super-sets of the target language L from L itself. That is, nothing ever appears in a text of L that invalidates a super-set name.[1]

The simple way around this problem is to use either effectively quasi-ordered by length or increasing text. Both of these kinds of text implicitly supply the necessary counterexamples and so are equivalent to informant.

The previously cited non-identifiability results employ a vast number of repetitions to "fool" the inductive inference machine. This suggests another way of improving the performance of text, namely bounding the permissible number of repetitions of any element of sch_L. The results shown by Feldman et al <1969> indicate that while this enlarges the potentially learnable class, the difference is not terribly significant.

Pursuing the idea of bounding repetitions, it seems to the author that if the number of occurrences of every element in a text is known to follow some non-zero limiting frequency then that text is equivalent to informant. This constraint on the limiting frequencies is precisely what gives the probabilistic language learners their power on text <Horning,1969>, but no equivalent results exist for non-probabilistic language learning.

Another way of "improving" text is given by Crespi-

---

1 In passing it should be pointed out that this fact also invalidates the strategy of finding a minimal name for each partial text. This is suggested by the example of the language {a*} - {a}. Given any text presentation of this language, a correct hypothesis intuitively must be more complex than any hypothesis that generates {a*}.

Reghizzi <1971>. He constructs machines that identify non-trivial subsets of the Context Free languages when given parse trees rather than mere strings. However this is beginning to stray considerably from the statement given initially for Chomsky's problem, and so is not discussed further.

The previous material gives various ways to make machines more powerful given text. In a sense, a bottom line to this is given by:

<u>THEOREM</u>: <Gold,1967> If C is identifiable given recursive text, then C is identifiable given arbitrary text.

So then, the question remains: What CAN be learned given (arbitrary) text?

<u>THEOREM</u> <Gold,1967> Any class of finite languages is identifiable in the limit given text.

Clearly all that must be done is to always hypothesize exactly the current partial text.

IDtext has been variously characterized in <Hamburger and Wexler,1973b>, <Kugel,1977>, <Wiehagen,1977>, <Feldman et. al,1969>, <Angluin,1975b>, and <Coy and Pfluger,1979>.

<u>THEOREM</u> <Angluin,1979b> For C any class of recursive languages, C is identifiable given text if either:

1) Any finite set of strings is contained in only finitely many of the languages in C.

OR

2) a) The containment problem is solvable for languages within C.

b) For each language $L \in C$, there is a finite sublanguage of L for which no language belonging to C both contains the finite sublanguage and is itself included in L.

Conversely:

THEOREM <Angluin, 1979b> If a class C of recursive languages is identifiable given text then C satisfies condition 2b above.

There are several results that appear to suggest that text may not be so terribly limited for approximate identification. However close examination tends to destroy such optimism.

THEOREM <Wiehagen, 1977> $\exists$ an identifiable (by text) class C of languages such that (L is a recursively enumerable language) implies ($\exists L' \in C$ and L′ is almost everywhere identical to L).

Inspection reveals that this class corresponds to the class of self describing functions, and unfortunately there is no effective method to acquire a name for L′ when presented with the text of L.

A few definitions are required before presenting Wharton's seemingly powerful results on approximate identification.

Definition: $W=(w_1,w_2,\ldots)$ is a sequence of weights if $\forall i$ w_i is positive and $\sum w_i = 1$.

Definition: Given some finite terminal vocabulary V_t and some sequence of weights W, for $L \subseteq V_t^*$ $norm_W(L) = \sum ch_L(s_i)*w_i$ where the strings s_i are lexicographically ordered.

Definition: Given some finite terminal vocabulary V_t and some sequence of weights W, for L_1 and $L_2 \subset V_t^*$ $dist_w(L_1,L_2) = norm_W(L_1 \oplus L_2)$, where $\oplus$ stands for the symmetric difference.

Definition: ϵ -identification ,with respect to some sequence of weights W, is defined like identification except that the index i converged upon must satisfy $dist_w(L_i,L)<\epsilon$ where L_i is the language specified by t_i, L is the target language.

$Dist_w$ is a metric on the class of all languages within V_t^* (i.e. the "universal class" of languages), and so permits of such metric space concepts as denseness. The corresponding notion of ϵ -matching has not been defined, but appears to present no new problems. The following theorems are all stated assuming an admissible hypothesis space that generates a class C of languages dense in the universal class of languages.[1]

THEOREM <Wharton,1974> $\forall\ \epsilon > 0$, C is ϵ -identifiable given text.[2]

THEOREM <Wharton,1974> $\forall\ \epsilon > 0$, C is ϵ -identifiable in fixed time given effectively quasi-ordered by length text.

These results suggest that an approximate learner can

---

1 {finite languages} is an example of such a class.
2 Informant allows identification in known time.

be very powerful. However their force is vitiated somewhat by the fact that the measures are "weighted metrics", and as such have the feature that for any $\epsilon > 0$ and any language L, there is a language L′ that is almost everywhere distinct from L yet $dist_w(L,L') < \epsilon$. So languages are being identified by matching only some finite portion of them, in this case all strings up to a certain length. The justification given for this is that it is the short strings which are important in any practical sense.

In contrast to text, informant provides full information about both the characteristic and semi-characteristic functions. In language oriented studies the discussion is most often about recursive languages for which the results in chapter two apply directly.

3.6 About Implementations

Any discussion of the implementations that are tolerably efficient and provably valid on more than the handful of examples used by the designer must be extraordinarily brief. For the worthwhile results in language learning currently consist not of practical designs for inductive inference machines but of abstract specifications for the various possibilities. The insights thus far do not provide, nor even suggest, efficient general solutions. This should not be taken to indicate the subject's irrelevance, but rather its current focus. To papraphrase a quote given in <Forning,1969>: although proving the existence of a solution is only a first step toward finding a practical solution, in a subject replete with logically intractable problems it is worthwhile delineating even the possibilities for solution.

Nevertheless, some mention should be made of the various studies aimed at the creation of more or less "practical" language learners. The fundamental distinction made between the various implementations is that between inductive inference machines that are "enumerative" versus those that are "constructive".

As the term suggests, enumerative machines enumerate through a Hypothesis Space, testing each name (so far as any complexity restrictions permit) for compatibility with the current sample. Essentially, these are the machines employed in the results detailed in the last two chapters. Thus their power is relatively easy to characterize, and they can often be modified so as to infer minimal (intrinsic complexity) hypotheses. However they are, in effect, just the "Monkeys with Typewriters" prescription for inductive inference, i.e. try everything. Clearly, for many natural hypothesis spaces, this strategy results in an explosive number of candidates. Some estimates of the seriousness of this problem are given in <Bierman et al.,1972b>. An example is their calculation that $\exists$ about $2^{kn(1+n)}$ different Type 3 grammars with k terminals and n nonterminals. And Type 3 grammars produce only a very limited number of the (theoretically) identifiable classes of recursive languages.

Figures such as those above have ensured that very few "users" of enumerative inductive inference machines profess any practical ambitions for their machines. And one of the major efforts by Wharton <1974>, although not so intended, serves as a warning rather than a beacon for future practically oriented research. For even using "failure points" to eliminate future occurrences of any hypotheses that are known to cover failed hypotheses, "success points" to guide future hypotheses, and tests for complete equivalence, disconnected grammars, blocking grammars, merging nonterminals, direct substitutibility, circular non-terminals, left and right recursion ambiguity, missing terminals and so on, even after all of these refinements, resulting in a 10^7 decrease in the number of grammars examined, for one context free language requiring a grammar with only 6 rules and 2 and 5 terminals and non-terminals respectively, Wharton's machine counts through 355,576 (!) candidates, a figure that while admittedly better than the 2,225,706,812,694 necessary if the above refinements had not been employed, is nevertheless horrendous. And that is virtually the largest grammar that can be acquired by Wharton's machine in an even remotely practical sense.

Many of Wharton's tests are designed to eliminate "worthless" grammars before employing them. His grammar generating schema generates many undesirable types of grammars (those equivalent to previous ones, blocked and disconnected etc.) and very many grammars that do not even generate the current partial enumeration.

The latter problem is where the "Logics of Discovery" mentioned in Chapter One might conceivably be useful. An enumerative machine equipped with a Logic of Discovery "preprocessor" might be able to enumerate through only hypotheses that are at least compatible with the current partial enumeration, and thus (so the reasoning goes) obtain much greater efficiency. A minor quibble with this is that the results of Chapter Two show that such an approach limits the power of the inductive inference machine (since this approach clearly guarantees consistency). However any of the practical efforts suffers (?) from this. More serious are the indications that such an approach is still combinatorially explosive. <Pudlak,1975> details several NP-complete problems with respect to Hajek's Logic of Discovery. The use of "derived grammars" <Fu,1975> in the development of finite state grammars would seem to realize the best that such a preprocessor could do, since it results in an admissible hypothesis space each of whose grammars is compatible with the current partial enumeration and at least one of whose grammars is correct. Significantly, this technique is considered unmanageable when more than 10 nonterminals are needed. Other more efficient methods, starting from the "canonical derivative" or "k-tail" grammar for example, unfortunately do not guarantee the existence of a correct grammar in the enumerated class.

Constructive machines take a partial enumeration and, beginning with some "candidate hypothesis", often the ad hoc grammar, derive a "good" hypothesis for the current sample by modifying or adding to the rules. Proofs of limiting behavior rarely appear (<Crespi-Reghizzi,1971> is one of the few exceptions). The suggestions of <Solomonoff,1964>, <Feldman et al,1969>, <Klein and Kuppin,1970>, <Lee and Fu,1972>, <Knobe and Knobe,1976>, <Porter,1976>, and <Miclet,1976> for example, are therefore of purely heuristic value, and indeed are easily confused with solutions to the good encoding problem.

The many recursively unsolvable problems for Type i (i<3) languages <cf.Hopcroft and Ullman,1969> have meant that much less progress for these languages has been made. For these it might well be the case that a man-machine interactive system similar to that in <Klein and Kuppin,1970>, <Lee and Fu,1972> or <Guiho and Jouannaud,1977> offers the best hope for workable language learners in the near future.

Chapter 4

FUZZY LANGUAGES

4.1 Introduction

Fuzzy sets are defined by replacing the usual set-theoretic 0-1 valued characteristic function, with a "membership" function whose range is contained in [0,1].[1] The basic set operations are then defined in terms of the respective membership functions. The membership function of the union of two fuzzy sets with membership functions f_1 and f_2, is defined to be $\max(f_1,f_2)$, since intuitively something is in the union of two sets at least as much as it is in either one. The membership function of the intersection of two fuzzy sets with membership functions f_1 and f_2, is defined to be $\min(f_1,f_2)$, since intuitively nothing can be in both sets any more than it can be in either one. And finally, the membership function of the complement of a set with membership function f, is defined to be 1-f, since

[1] This is the usual way. Other suggestions have the membership function take as its range: an arbitrary ordered semi-ring, in order to remove the bounded nature of the degree of membership <Wechler,1975>; an arbitrary lattice, in order to allow incomparable degrees of membership <Kim et al.,1975>; the set of fuzzy sets, as Zadeh defined them, contained in [0,1], in order to incorporate vagueness into the very assignation of membership <Mizumoto et al.,1976>.

membership in the universe must be total. From these three definitions the full range of the usual set-theoretic operations can be defined if desired.

There is a certain logical necessity to the previous max-min definitions, which is worth realizing in the extension of fuzzy set theory into the realm of formal languages. Bellman <1973> showed the max-min definitions to be inescapeable given only the acceptance of five axioms[1] which do not obviously conflict with intuitive notions of vagueness, and the logical equivalence of the statements A U (∩) B, with the statements that for all x [x ∈ A] v (and) [x ∈ B].

Since a formal language is defined to be simply a set of finite strings constructed from some finite vocabulary, there is no immediate obstacle to the definition of fuzzy formal languages. A fuzzy formal language is just a fuzzy set defined on the finite strings constructed from some finite vocabulary. That is:

<u>Definition</u>: A <u>fuzzy</u> <u>formal</u> <u>language</u> <u>L</u> is {(x,m(x)) : x ∈ V_t* and m is some (arbitrary) real valued function mapping V_t*->[0,1] [2] }, where V_t is some finite set of

---

[1] 1. Union and Intersection are reflected in commutative, associative, binary, and mutually distributive operations and , v on [0,1].
2. x and y, x v y are continuous and nondecreasing in x.
3. x and x, x v x are strictly increasing in x.
4. x and y ≤ min(x,y), x v y ≥ max(x,y)
5. 1 and 1=1, 0 v 0=0

[2] Non-fuzzy languages are often conflated with fuzzy languages with 0-1 valued membership functions.

characters.

Definition: In the above definition, m is known as the membership function for L, and the semi-membership function sm for L is a function identical to m except that m(s)=0 implies sm(s)=undefined.

That the development and usefulness of formal languages has depended upon the invention of finite methods to encode the structure exhibited by infinite sets of strings, is so obvious as to scarcely deserve mention. However it is precisely at this point that the concept of a fuzzy formal language begins to experience difficulties. For non-fuzzy languages there are a variety of satisfactory ways to accomplish the requisite encoding. Chomsky grammars, the most common, permit the naming of all recursively enumerable languages. Unfortunately there is no comparably successful notion of "grammar" for fuzzy languages. Instead there are a number of suggestions, all flawed in one or more respects.

4.2 Grammars for Fuzzy Languages

4.2.1 Fuzzy Grammars

Non-fuzzy languages are so well generated by Chomsky grammars that the obvious method to try for the generation of fuzzy languages is the generalization of the powerful phrase structure type of grammar. This is done in <Lee and Zadeh,1969>. Their "fuzzy grammars" are the original grammars devised for fuzzy languages, and still receive such wide acceptance as to render all subsequent proposals of only peripheral practical significance.

Definition: A fuzzy grammar is a quadruple (V_t, V_n, S, P) where V_t, V_n are terminal and non-terminal alphabets and S is a sentence symbol, as for Chomsky Type grammars, and the production set P is a set of expressions of the form $(x \Rightarrow y,\ g)$ where $x, y \in (V_t \cup V_n)^*$ and $g \in (0,1]$.

So in essence, a fuzzy grammar is obtained from a Chomsky Type grammar by attaching "grammaticality coefficients" g to the production rules. However, thus far a fuzzy grammar is indistinguishable from, say, a probabilistic grammar. Clearly then, a grammar's right to the title of "fuzzy" resides primarily in it's computation of membership.

Definition: Given a fuzzy grammar $G=(V_t, V_n, S, P)$, the base grammar $Gbase=(V_t, V_n, S, P')$ where $P' = \{(x \Rightarrow y) : (x \Rightarrow y,\ g) \in P$, for some $g\}$

Definition: Given a two-tuple $(x \Rightarrow y, g)$ belonging to the production set of a fuzzy grammar, $x \Rightarrow y$ is the production rule and g is the production grammaticality.

Definition: The language generated by a fuzzy grammar G is defined by the membership function:

$m(s) = 0$ if $s \notin L(Gbase)$

sup min $(m_1, m_2, \ldots, m_n)$ otherwise,

where m_i is the production grammaticality associated with a production rule appearing in some derivation of s by Gbase, and sup is taken over all possible derivations of s by Gbase.

Some further definitions that are of use later are:

<u>Definition</u>: The <u>set of support</u> of a fuzzy language L(G) is L(Gbase).

<u>Definition</u>: Let S be a fuzzy set and lambda ∈ [0,1], then the <u>lambda-level set of S</u> is the non-fuzzy set Slambda={x : m(x) ≥ L}

<u>Definition</u>: Given a non-fuzzy set S, and lambda ∈ [0,1], lambdaS denotes the fuzzy set whose membership function is given by:

m(x)=lambda   if x ∈ A

0   otherwise

<u>Definition</u>: Given a fuzzy grammar G=(V_t,V_n,S,P), and 0 < lambda ≤ 1, Glambda=(V_t,V_n,S,P′) where P= {(x=>y) : (x=>y, m) ∈ P for m ≥ lambda}.

<u>Definition</u>: Given a (non-fuzzy) grammar G=(V_t,V_n,S,P), and 0 ≤ lambda ≤ 1, the <u>lambda-fuzzification</u> of G, lambdaG is (V_t,V_n,S,P′) where P′={(x=>y, lambda) : (x=>y) ∈ P}.[1]

A simple example of a fuzzy grammar is the following grammar for almost balanced parentheses. Define G=(V_t,V_n,S,P) where:

---

[1] Note that this is not a fuzzy grammar if lambda=0.

$V_t=\{l,r\}$, $V_n=\{S\}$

$P=\{r_1,r_2,r_3,r_4\}$

with $r_1=$ (S=>lr, 1)

$r_2=$ (S=>lSr, 1)

$r_3=$ (S=>lS, 1/2)

$r_4=$ (S=>Sr, 1/4)

Then G generates a fuzzy language L with a membership function m_L defined as $m_L(l^k r^x)=1$ if $k=x$; 1/2 if $k>x$; 1/4 if $x>k$.

Several defects with fuzzy grammars are apparent immediately. First, the languages generated by them have membership functions with only finite ranges since the ranges can be no larger than the set of production grammaticalities.[1] The second and related difficulty is that no significant interaction can occur between the grammaticalities of the relevant production rules during the assignation of a membership to a string. For example, in the above grammar the "unbalancing" production rule can only lower a membership to 1/2, regardless of how many times it is applied and how unbalanced the resulting string is. The philosophy underlying fuzzy grammars, namely the "weakest link in the chain" <Zadeh,1970> conception of derivational validity, violates the linguistic intuition that repeated

---

[1] This is one of the two motivations cited for the development of Fractionally Fuzzy Grammars in <De Palma and Yau,1975>.

application of a less than fully acceptable grammar rule yields ever less grammatical sentences; scarcely grammatical sentences may result from the collective employment of rules that individually can scarcely be faulted.

The above is not to say that fuzzy grammars have no good points. They are simple. Moreover, Type 1 fuzzy grammars [1] generate languages with recursive membership functions <Thomason and Marinos,1974>. Perhaps the most compelling argument for focussing on fuzzy grammars here however, is simply that they are the grammars that completely dominate the fuzzy literature and applications <cf.Kickert and Koppela,1976; Thomason,1973>.

4.2.2 N-fold Fuzzy Grammars

"N-fold fuzzy grammars" <Mizumoto et al.,1973> define "conditional grades of membership". Like all of the proposals, they too start with a Chomsky Type grammar and modify the form of the production set. The "N" refers to the number of rules taken into consideration when defining a given rule's grammaticality. In effect grammaticality is no longer a property of a rule, but that of a rule's application in conjunction with other rules. For example, an element of the production set of a 1-fold fuzzy grammar is

[1] Fuzzy grammars are classified as Type 0,1,2, or 3, depending upon where the corresponding base grammar lies in the Chomsky hierarchy. A body of results directly analogous to those for non-fuzzy formal languages exists <Lee and Zadeh,1969>.

of the form:

$(x \Rightarrow y;\ m_1$ if $rule_1$ occurs in the derivation
m_2 if $rule_2$ occurs in the derivation
....
m_n if $rule_n$ occurs in the derivation)

For N=2 a production rule's grammaticality is permitted to be conditional upon the occurrence in the derivation of any two given rules, and so on. A string's membership is then evaluated in the same max-min manner as for fuzzy grammars. A certain "context sensitiveness" may be achieved [1], but since N is always some fixed integer, the problems noted for fuzzy grammars are merely deferred not eliminated.

4.2.3 Fractionally Fuzzy Grammars

Fractionally fuzzy grammars <De Palma and Yau,1975> take a Chomsky Type grammar and attach the values of two rational functions g, h to each rule, requiring that $0 \leq g(r) \leq h(r) \leq 1$ and $h(r) \neq 0$. Given such a grammar, the membership function for the language is evaluated by taking the supremum of $\Sigma\, g(r) / \Sigma\, h(r)$ where Σ is over all the productions used in some derivation of s, and supremum is over all possible such derivations (and is assumed to be zero if none exist).

---

[1] For example, context free threshhold grammars of this type can generate context sensitive languages <Mizumoto et al.,1973>.

An example of these grammars is $G=(V_t,V_n,S,P)$ where

$V_t=\{l,r\}$, $V_n=\{S\}$

$P=\{r_1,r_2,r_3,r_4; *g,h\}$

and $r_1=$ S=>lr $g(r_1)=1$ $h(r_1)=1$

$r_2=$ S=>lSr $g(r_2)=1$ $h(r_2)=1$

$r_3=$ S=>lS $g(r_3)=0$ $h(r_3)=1$

$r_4=$ S=>Sr $g(r_4)=0$ $h(r_4)=1$

G generates a fuzzy language with a membership function $m_L(l^k r^x)=min(k,x)/max(k,x)$. It can be seen that this generates an "almost balanced" parenthesis language much more adequately than the example used for fuzzy grammars, with the membership of a string steadily declining as the string becomes more unbalanced.

Designed with future applications in mind, fractionally fuzzy grammars are easy to parse due to the built in convenience of backtracking. [1] Type 1 fractionally fuzzy grammars result in total recursive membership functions <De Palma and Yau,1975>. And the class of languages generated by fractionally fuzzy grammars properly includes the languages generated by fuzzy grammars (with rational production grammaticalities) <De Palma and Yau,1975>. Best of all, fractionally fuzzy grammars do not seem to suffer the defects noted for fuzzy grammars. The repeated application

[1] To back up the grammaticalities after an unsuccesful attempt at parsing a string, it suffices to perform the relevant subtractions of g(r) and h(r).

of an only vaguely grammatical rule (i.e. one for which h(r)-g(r) is large) drives the resultant membership towards zero, and there can be infinitely many levels of membership in fractionally fuzzy languages. But the converse of the first point is that the influence of any one rule no matter how ungrammatical may be swamped by the application of many others. Furthermore, like the suggestions that follow, fractionally fuzzy grammars suffer from an appearance of ad hocness. No justification in terms of any conception of fuzzy languages is provided for their novel calculation of memberships. The "weakest link principle" of fuzzy grammars may not be valid, yet it at least provides some sort of rationale for the max-min membership calculations.

Fractionally fuzzy grammars provide an opportunity to re-examine fuzzy grammars. Although originally postulated for the set-theoretic operations of union and intersection, Bellman's axioms are suggestive in the case of fuzzy grammars also. The assumption for fuzzy grammars is that derivations are sets of production rules such that the membership of an individual derivation in the set of grammatical derivations corresponds to the truth value of the statement about its constituent rules that: "r_1 is grammatical and r_2 is grammatical and ... and rn is grammatical", while the membership of the set of several alternate derivations d_i in the set of grammatical derivations corresponds to the truth value of the statement: "d_1 or d_2 or... d_n is grammatical". For fuzzy grammars these truth value calculations follow Bellman's axioms on the operations on [0,1]. And the max-min assignation of grammaticality necessarily follows. Fractionally fuzzy grammars assign each rule a definite grammaticality yet avoid this. The only point where the corresponding calculation of truth values differs with Bellman's axioms is the fourth axiom which states x and y $\leq$ min(x,y). For example, the use of two rules r_1 and r_2 to generate a string s could result in m(s)=2/5 for r_1 having grammaticality 1/2 and r_2 1/3.

4.2.4 The Grammars of Eugene Santos

Fractionally fuzzy grammars raise the suspician that there may be many legitimately "fuzzy" ways of assigning a string s a real number while generating s via a Chomsky type grammar. The work of Santos <1974> strengthens this suspician. Three general methods for realizing fuzzy languages are outlined there. The first and the last of these are just the standard stochastic (?!) and fuzzy grammars. The second is a curious hybrid: A normal fuzzy grammar is given a fuzzy set of sentence symbols, rather than the usual single sentence symbol, and the value of a given derivation is then computed by taking the product of the grammaticalities (as for stochastic grammars) together with the membership of the particular sentence symbol employed to begin the derivation. If there is more than one derivation for a string, then the string's membership is taken to be the supremum of these values. If there is no derivation for a string, its membership is zero. No rationale for the use of max-product grammars is provided by Santos, and the inclusion of stochastic grammars in the same scheme seems rather odd initially. However max-product grammars avoid the flaws noted for fuzzy grammars; also, with max-product grammars the effect of the application of a single bad production cannot be swamped by the subsequent application of fully grammatical productions, yet the repeated application of slightly ungrammatical productions can arbitrarily lower the final assessment of grammaticality. And the similarity of max-product grammars to stochastic grammars may not be so very unreasonable after all since an empirical definition of the "grammaticality" of a sentence might well be that it is the likelihood, not of being generated (as for probabilistic languages), "but of being judged acceptable by a member of the language community at a particular time" <Schubert,personal communication>.

4.3 Conclusions

These then have been the only attempts to define generative mechanisms for fuzzy languages. Fuzzy grammars with their max-min scheme fail to generate many apparently useful fuzzy languages, lacking the necessary flexibility of assignment, yet are used almost universally. N-fold fuzzy grammars ultimately have the same failings as fuzzy grammars. Fractionally fuzzy grammars seem rather arbitrary and still fail to model some grammatical intuitions. And max-product grammars are rarely, if ever, used.

Perhaps the main value of the latter types of grammars for fuzzy languages rests in their demonstration that the max-min principles of fuzzy set theory do not necessarily apply in any obvious way to the application of production rules. This opens the way to a general study of the ways of generating strings and attached coefficients simultaneously, with the goal of choosing one that is simultaneously powerful and yet true to the spirit motivating the creation of <u>fuzzy</u> languages. While this is beyond the scope of this thesis, some possible avenues for the first task will be mentioned.

The results of associating a "cost function" with the state transition function of a finite automaton have been investigated under the name of "sequential decision processes" <Ibaraki,1976; 1978>. For a given string abcd...z, the "cost" h(abcd...z) is determined as the result of the consecutive cost evaluations corresponding to the fsm state transitions yielding a, b, c, and so on. A sequential decision process accepts a string s if h(s) does not exceed some threshold value v. Not only do such machines subsume the fsm version of stochastic, max-product and fuzzy grammars, but they are capable of accepting any r.e. set in V_t*.

Perhaps a fuzzy language should be considered, not as a language per se to be generated by a grammar, but instead as a mapping or a "translation" from strings to a grammatical scale. Translations from one formal language to another, have been investigated under the name of "syntax-oriented translation" <Abramson,1973> or "generalized syntax-directed translation schemes" <Aho and Ullman,1973>. Essentially such translations are algorithms that analyze, and perform some transformation on, sentences from some class of languages. They do this by associating one or more transformations with each production rule and non-terminal symbol. Some common examples of their use include the translation of certain strings of zeros and ones representing the positive integers as sums of Fibonacci numbers into their decimal representation, and the differentiation of polynomial expressions. Although currently the theory refers to context free languages, Abramson <1973> believes that the scope will be extended eventually.

Chapter 5

FUZZY LANGUAGE LEARNING

5.1 Learning Fuzzy Languages

5.1.1 Previous Work

At the time of writing, <Tamura and Tanaka,1973> is the only paper ostensibly addressed to the problem of learning fuzzy formal languages. This is a surprising situation considering the very partial nature of their solution, particularly given the amount of material that exists on virtually ever other conceivable "fuzzy topic" <cf.Caines and Kahout,1977>, and the repeated expressions of interest in some method for learning fuzzy languages from a "training set" <cf.Thomason,1973; De Palma and Yau,1975>.

Despite the title -"Learning of Fuzzy Formal Languages"- and much of the intuitive motivation and explanation, Tamura and Tanaka's paper is in fact devoted to the approximation of a non-fuzzy formal language L by successively hypothesizing ever "better" fuzzy grammars. Informally stated, their goal is the development of a procedure that, given an initial fuzzy grammar whose base grammar includes a grammar for the target language L, outputs a sequence of fuzzy grammars whose languages have membership functions that approach L's characteristic function in the limit[1]. While this goal can be modified to accommodate fuzzy target languages, not only does their solution break down, but it is shown that such a goal is, in a certain sense, futile.

The problem is restricted to recursive target languages and the fuzzy grammar G initially provided is decidable (i.e. Type 1,2 or 3). Each string of a partial text is parsed, and although parsing ambiguities lead to some complications, essentially the procedure is to take the fuzzy grammar hypothesized for the previous partial text and apply a standard linear learning scheme to its production grammaticalities on the basis of a production rule's participation in the latest series of parses. i.e., If S is a set of rules necessary and sufficient for some parse of any string s in the current partial text, and $g_n(r)$ is the grammaticality of rule r in the previously hypothesized fuzzy grammar, then the updated grammaticality of rule r is:

$$k*g_n(r) + (1-k)*ch_S(r)$$

where $k \in (0,1)$ is arbitrary and ch_S is the characteristic function of S.

Using this method Tamura and Tanaka claim to be able to attain their previously stated goal of approaching the target language in the limit. What they prove is

---

[1] The analytical, not the number theoretic, limit.

considerably different.

<u>THEOREM</u> <Tamura and Tanaka,1973> Given some partial text T of a recursive language L, and an initial recursive fuzzy grammar G such that L $\subseteq$ L(Gbase), then:

1) $\forall$ lambda $\in$ (0,1) $\exists$ N such that n>N implies

$L(G_n)$lambda = $L(G_{pos})$

2) $L(G_i base)$ is constant $\forall$ i

3) {s such that (s,1) $\in$ T} $\subset$ $L(G_{pos})$ $\subset$ L(Gbase).

where $G_n$lambda is the lambda level set of the grammar hypothesized after T has been input n times, and G_{pos} is a subgrammar of Gbase, i.e. a rule r is in G_{pos} iff r participates in some parse of a string in T.

So their result concerns grammar convergence for repeated presentation of a finite sample of a language, rather than convergence for a presentation of the entire language.

5.1.2 A New Outlook

The functionally oriented presentation of language learning given in Chapter Three, extends in an obvious manner to fuzzy languages. A (non-fuzzy) language has a 0-1 valued characteristic function. A fuzzy language has a real valued membership function[1]. A (non-fuzzy) language has a

---

[1] This glosses over the fact that the functions are no longer number theoretic, but rather have real valued ranges in [0,1]. This transition poses no difficulties if (and only if) their ranges are restricted to the computable real numbers. This seems to be a very reasonable assumption.

semi-characteristic function. A fuzzy language has a semi-membership function. All the usual set theoretic relations and operations, such as equality, intersection and containment, have fuzzy equivalents. So then, much as for (non-fuzzy) languages, learning a fuzzy language L can be considered as learning either L's membership or semi-membership function, with the new definitions of identification, matching, informant, text[1] and so on being obvious extensions of the former ones. Consequently the functional results discussed in Chapter Two apply to fuzzy language learning just as they do to non-fuzzy language learning.

However, the acquisition of grammars, rather than programs, is such a standard requirement in language learning studies, that the problem is universally known as the "grammatical inference problem". Seen in this light the problem still exists for fuzzy languages. Because of their simplicity and wide acceptance, fuzzy grammars are used for the remainder of this section. The fuzzy languages corresponding to fuzzy grammars will occasionally be denoted as the fuzzy (grammar) languages.

---

[1] Note that (fuzzy) text in this sense includes exact grammaticalities for each string.

5.1.2 A New Outlook

5.1.3 Assigning Grammaticality to Known Rules

The obvious way to proceed is to modify the approach in <Tamura and Tanaka,1973> so as to accommodate non-trivially fuzzy languages while remaining within the framework outlined in the previous chapters. This leads to results like the next theorem, which may be viewed as facilitating the assignment of grammaticalities in practical situations where some grammar is already known that includes a base grammar for the target language.

<u>THEOREM 1</u> The class of Type 0 fuzzy (grammar) languages can be identified in the limit given text, assuming that the inductive inference machine is given a Type 0 unambiguous grammar G that includes a base grammar for the target language's set of support.

<u>Proof</u>: A procedure will be given and then shown to work. Call the text used for L, $\hat{L}$.

To begin with, 0-fuzzify G, calling the result GF.

For $\hat{L}_n$:

For an element (s,m(s)) appearing in $\hat{L}_n$ but not in $\hat{L}_{n-1}$, parse s by G. Given that a production rule r participates in the derivation of s, examine the current grammaticality g of r in GF. If g < m(s) then set g to m(s). Return as the hypothesis H_n the fuzzy grammar obtained by removing all pairs of the form (r,0) from the production set of GF.

This procedure works since:

*There can be only finitely many distinct values m(s) appearing in the sample presentation.

*The current grammaticality of any rule in H_n that confers its grammaticality to some string in L's set of support is $\leq$ its grammaticality in any grammar derived from G that generates the target language.

*There is a partial text $\hat{L}_n$ past which the rules in the production set of H_n are adequate to generate the set of support of the target language.

*There is a partial enumeration $\hat{L}_n$ past which H_n remains constant, since a rule's grammaticality can increase only a finite number of times and there are only a finite number of rules in G.

*F_n is by construction a fuzzy grammar.

Suppose there is no n past which H_n generates the target language. Let H_n be the grammar finally settled upon by the fourth observation. Let s be a string assigned different memberships in the target and hypothesized languages. If m(s)=0 then H_n has a production rule r that is not in any subgrammar G_t of G for the target language. Since s is never parsed, some other string s_0 in L's set of support must have been responsible for the introduction of a non-zero grammaticality for r in H_n. But this implies that there are two parses of s_0, one involving r and the other only rules in G_t. This contradicts the fact that the grammar G is unambiguous. Assume $m(s) \neq 0$. If $m(s) < m_{Hn}(s)$ then, since the rules are fixed and unambiguous, a contradiction arises to the second observation above. However, if $m_{H_n}(s) < m(s)$ then some rule r in H_n used in the derivation of s has a grammaticality < m(s). But s must appear in some partial enumeration, forcing all the rules appearing in its derivation to have grammaticalities of at least m(s) after this. Therefore H_n is not the final grammar hypothesized. This contradicts the assumption that H_n is the final grammar settled upon. Since this has exhausted the possibilities, H_n must generate a language equivalent to the target language. //

Note that this result seems very much stronger than any results cited for non-fuzzy languages. The provision of a grammar that contains a correct base grammar is responsible for this. The theorem clearly holds also for the Type 1,2,3 restatements.

The unambiguous restriction in the above result is inessential, however there is then no longer nearly such an efficient updating procedure due to the masking effect of the max operator. That is, the participation of a rule r in a derivation of some string s no longer implies that the grammaticality of $r \geq m(s)$. This forces what is essentially a trial and error assignment of grammaticalities. One plausible, albeit highly inefficient, solution might be to begin generating all possible parses for each string s while simultaneously (by dovetailing the operations) generating a tree of altered GFs by the previous method modified so that a check is made as to whether the current grammaticality of at least one rule in each parse is $\leq m(s)$, and the branch is eliminated if this is not the case. Actually, a restriction to unambiguous grammars is not uncommon in other language learning studies <cf.Horning,1969>.

5.1.4 Can Coefficient Assignment Methods be Extended?

While the previous method may aid in the construction of fuzzy grammars in certain instances, in general the a priori assumption of a grammar including a base grammar for the target language is unwarranted. The natural response is, as Tamura and Tanaka suggest, to attempt the addition of a "front end" that discovers this. That is, much as for stochastic languages, the problem is broken into two parts. The first involves the acquisition of a non-fuzzy grammar containing a base grammar that generates the target language's set of support; and the second involves the acquisition of the fuzzy coefficients. Unfortunately there is good reason to think that such an approach is not feasible. With the exception of <Crespi-Peghizzi,1971>, all language learning studies are concerned with finding grammars that generate merely the strings of the target language. This focus arises naturally from the definition of formal languages as mere sets of strings with no associated derivational histories nor "meaning". The possibility of assigning grammaticalities to the production rules is crucially affected by this. For example, if the target language has six distinct levels of membership, no grammar with only four production rules can possibly generate it, yet many such 4-rule grammars may generate the same set of support. While a solution may be had involving the interaction of the two sub-problems (e.g. When a conflict of this nature occurs, start solving problem 1 again.), such an approach seems very clumsy at best.

5.1.5 A General Solution

The previous discussion suggests that the acquisition of a fuzzy grammar should come about through the simultaneous acquisition of both rules and coefficients. An extension to a result for fuzzy sets states that:

<u>THEOREM</u> <Zadeh,1970> If G is any fuzzy Type i (i=0,1,2,3) grammar then L(G)=UNION lambda L(Glambda),

where UNION stands for the union, over lambda=the production grammaticalities appearing in G, of the fuzzy sets lambda L(Glambda);

and the Glambda are all of Type i <Zadeh,1970>.

This fact suggests a general solution, namely that of acquiring a separate grammar for each non-zero lambda-level set of the target language, by standard function (language) learning techniques, lambda-fuzzifying them, and then outputting the union of these fuzzy grammars. This approach works in fact, however the details establishing this for each learning criterion are very tedious. The next theorem provides a cleaner technique based upon essentially the same idea. A function's domain is now assumed to be included in some V_t* rather than N.

THEOREM 2 Given any partial recursive semi-membership function f with finite range R, ∃, uniformly in f, a fuzzy Type 0 grammar G such that $sm_{L(G)} = f$.

Proof: For each r ∈ R, assuming R ≠ ∅:

a) Construct a Type 0 grammar for $f^{-1}(r)$. This construction is uniformly effective in f since $f^{-1}(r)$ is recursively enumerable [1] and hence ~~is~~ the domain of a partial recursive function t_r effectively constructible from f <Rogers,1967> and hence is the language generated by a Type 0 grammar G_r that is effectively constructable from t_r <Hopcroft and Ullman,1969>.

b) Construct a fuzzy grammar F_r by r-fuzzifying G_r.

If R = ∅ then let G be the null grammar. Otherwise, as the final step union[2] the F_r to obtain G.

The above procedure clearly works for f=the everywhere divergent function.

---

1 Begin calculating $f(s_1)$, $f(s_2)$, $f(s_3)$, ... dovetailing the calculations, and whenever a calculation of $f(s_i)$ terminates and yields r, output s_i.

2 This is done exactly as described by Hopcroft and Ullman <1969> for Chomsky Type grammars. Informally stated, the operation consists of ensuring that each F_r has a unique non-terminal vocabulary (in order to avoid "derivational cross-overs") save for a common sentence symbol, and then unioning the individual terminal and non-terminal vocabularies and production sets.

Suppose $f(s)=r$. Then $s \in f^{-1}(r)$ and so s is generated by G_r and has membership r in $L(F_r)$. Consequently the membership of s in L(G) is at least r. But if this membership is greater than r, then s is generated as well by some $G_u \neq G_r$, which in turn implies that $f(s)=u \neq r$ which is a contradiction. Therefore the membership of s in L(G) is exactly r.

Suppose f(s)=undefined. Then s will not be generated by any G_r and hence will have an undefined semi-membership in L(G).//

The converse of this theorem is immediate. Moreover, although the theorem has been stated with reference to Type 0 fuzzy grammars and partial recursive semi-membership functions, only slight modifications are needed for the other types of Chomsky grammars and corresponding functions. A function f is said to be "computable by a finite [pushdown] [linear bounded] automaton" M if M accepts precisely $\{(s,f(s)): s \in V_t^* \text{ and } f(s) \text{ is defined}\}$.

<u>COROLLARY</u>: Given any semi-membership function f, with finite range R, such that f is computable by a finite [non-deterministic pushdown] [linear bounded] automaton $\exists$, uniformly in f, a fuzzy Type 3 [2] [1] grammar G such that $sm_{L(G)} = f$.

<u>Proof</u>: By analogy with the proof of the theorem, replacing Turing machines by finite, pushdown or linear bounded automata respectively. In somewhat more detail, for finite automata, the altered lines of the former proof are as

follows. Let M be a finite automaton that accepts f. Construct a Type 3 grammar for $f^{-1}(r)$. This can be done effectively since $f^{-1}(r)$ is accepted by a finite automaton <Hopcroft and Ullman,1969>, namely the one that when given s, gives (s,r) to M. This step, for the three different types of machines, rests upon the fact that if fs is a finite state transducer that adds a fixed symbol to its input and g a finite [push down] [linear bounded] automaton, then g composed with fs is a finite [push down] [linear bounded] automaton.//

The method is now simplicity itself. To identify [match] a class of fuzzy languages using a hypothesis space containing fuzzy grammars, construct an inductive inference machine that identifies [matches] (in the linguistic sense, i.e. extensions are not permissible) the corresponding class of partial recursive (semi-membership) functions, and pass its hypotheses to a Turing machine that translates the hypothesized program index into the corresponding fuzzy grammar via the procedure outlined above. This permits most of the (non-fuzzy) language Learning results to be restated easily for fuzzy (grammar) languages using hypothesis spaces containing only fuzzy grammars. For example, call the class of sets of fuzzy languages generated by fuzzy grammars of type i $FGRAM_i$, then:

<u>COROLLARY 1</u>: $FGRAM_0$ is identifiable in the limit given primitive recursive text.

<u>COROLLARY 2</u>: $FGRAM_1$ is identifiable given informant.

COROLLARY 3: The total recursive subclass of $FGRAM_0$ that can be matched given informant, is strictly larger than the total recursive subclass of $FGRAM_0$ that can be identified given informant.

A possibly undesirable feature of this solution technique is that the grammars hypothesized may be ambiguous despite the fact that there is an unambiguous grammar for the target language. Suppose the target language L has three non-zero levels of grammaticality, $g_1 > g_2 > g_3$ and an unambiguous grammar G generating L has in its production set three rules p_1, p_2, p_3 corresponding to these grammaticalities. Suppose also that p_1 and p_2 together generate a string s. The grammars G_2, G_3 created for g_2, g_3 respectively, could very well both contain p_1 and p_2. And in the final fuzzy union this would create two derivation paths for s.

5.2 "Very Approximate" Learning Criteria

Chapters Two and Three reviewed the various ways that have been suggested to permit a language learner to hypothesize languages that are almost, but not quite, the same as the target language. In each case this simplified the task and permitted the learning of larger classes of languages. This is desirable due to the limitations noted for exact limiting criteria. Just as there is a need for theories of fuzzy or approximate deductive reasoning <cf.Zadeh,1977>, so too is there a need for theories of fuzzy inductive reasoning.

5.2.1 Order-Identification

The approximation of a (possibly fuzzy) language by fuzzy grammars, in the manner stated at the beginning of section 5.1.1 as the goal of Tamura and Tanaka, appears to provide a promising new limiting criterion for approximate learning. This is an illusion however. It confers no advantages over the standard exact limiting criteria even for non-fuzzy languages, as is apparent from the next theorem.

Definition: An inductive inference machine M order-matches a fuzzy language L in the limit if :

1) M's hypothesis space contains only fuzzy Type 1 grammars

2) for every text [informant] of L $\exists$ N such that if $[m_L(s_1) > m_L(s_2)]$ then $[m_{Hn}(s_1) > m_{Hn}(s_2)]$ $\forall$ $n>N$; where (H_i) is the sequence of M's hypotheses.

3) $(H_i$base) stabilizes

THEOREM 3 If an inductive inference machine M order-matches a class C of (non-fuzzy) languages given text then $\exists$, uniformly in M, a machine M' that matches C in the limit.

Proof: Suppose M order-matches L. Let (H_i) be the sequence of M's hypotheses given a text for L. Call the nth partial text $\hat{L}_n$. Define M' by the following program description.

For $\hat{L}_n$:

For all (s,1) ∈ $\hat{L}_n$, compute all parses of s, and thereby the membership assigned to s by H_n. Output H_n' defined to be the 1-fuzzification of $H_n$lambda where lambda=the minimum value obtained from the above membership calculations.

∀n, let lambda_n be the least grammaticality assigned by H_n to any member of L. Since M order-matches L, ∃ a partial text number N and a base grammar BG such that ∀n>N $H_n$base=BG, and all strings not in L have memberships < lambda_n in $L(H_n)$.

Claim: The value of lambda used by M′ to compute H_n' is lambda_n for n sufficiently large.

For any derivation d of a string, let r_d denote the set of rules used in d. For any s∈ L, let R_s denote the group of rule sets {r_d: d is a derivation of s using grammar BG}; i.e. R_s contains every set of rules in BG that can be used for deriving s. Finally, let R_L be the collection of all such groups of rule sets for strings in L; i.e. R_L={R_s: s∈ L}. R_L is finite since the set of rules in BG is finite. Hence ∃ some finite set of strings S ⊂ L such that R_L={R_s:s∈ S}. The strings in S all appear in $\hat{L}_n$ for n sufficiently large, say n>N′>N. Hence ∀n>N′, and every t∈ L, some s appears in $\hat{L}_n$ with R_t=R_s. Consequently, if t receives the minimal grammaticality for L, so does s, and therefore the lambda used by M′ to compute H_n' is lambda_n as required.//

Tamura and Tanaka apparently wished to assign grammaticalities to the production rules of an essentially stable Type 1 base grammar in such a way as to force the membership values of the resultant languages to approach the target language's values in the limit. They were concerned only with text sample presentations since the class of languages obtainable from Type 1 grammars is already identifiable in the limit given informant (since they form a r.e. class of recursive languages). In short, their hope presumably was to enhance the currently rather dismal performance of language learners given text. However, a machine embodying this goal would order-match the target language and so, by the last theorem, could be replaced by an (exact) matching algorithm. Consequently such a machine would still be extremely limited in its power with respect to text, as the results cited in section 3.5 demonstrate.

5.2.2 E and E_{range}-identification

Conceptually, fuzzy languages seem to demand a notion of equality that permits an infinite number of differences between target and hypothesis as long as the overall proportion is not "too large". In another context, Tsichritzis <1971> notes that such "fuzzy"[1] functional

[1] This term receives various interpretations. Tsichritzis <1971> and Santos <1974>, for example, use it essentially to indicate an assignment of coefficients free from the constraints of the axioms of probability. The term is used only informally here, with any technical usage being reserved for situations deriving more obviously from Zadeh's max-min membership definitions.

approximations can significantly simplify many problems. The expectation that this should be true for language learning, is strengthened by the results cited in Chapters 2 and 3. Consequently two criteria of such "very approximate" learning, E and E_{range}-identification are proposed below.

In the interests of simplicity, the following analysis is given initially in terms of functions rather than languages, with the implications for language learning discussed later in the chapter. The following definitions apply only to total functions.

Definition: Given two functions f and g, $DIF(f,g,n) = \{x: f(x) \neq g(x) \text{ and } x \leq n\}$.

Definition: Given two functions f and g, $DENSDIF(f,g) = \lim_n \sup \#DIF(f,g,n)\#/n$

Definition: A function f_v is an E-variant of a function f if $DENSDIF(f,f_v) \leq E$.

Definition: An inductive inference machine M E-identifies ($0 \leq E \leq 1$) a function f in the limit if for every enumeration of f, $\exists$ i such that M converges to i and t_i is an E-variant of f.

Definition: E-ID is the class of E-identifiable sets of total recursive functions.

Remarks

1) E-variants of E-variants of a function f are not necessarily E-variants of f, for $E \in (0,1)$; however, 0-variants of 0-variants of f are 0-variants of f.

This suggests that 0-variants are better behaved than E-variants in general, and so should be stressed.

2) If a total function f almost everywhere equals a total function g then f is a 0-variant of g.

3) 0-variants of a total function f are not necessarily almost everywhere equal f. For example, given f, define the function f_v by:

$$f_v(x) = f(x) \text{ if } x \neq 2^n,\ n \in N$$
$$\qquad f(x)+1 \text{ otherwise}$$

4) Whereas finite variants of recursive functions are again recursive functions, 0-variants of recursive functions are not necessarily recursive. And whereas the finite variants of a total recursive function are recursively enumerable,

<u>Proposition</u> Given any total recursive function f the set of total recursive 0-variants is not recursively enumerable.

<u>Proof</u>: By contradiction. Let $f_{v1}, f_{v2}, \ldots$ be some such effective listing. Given any total recursive function r, $\exists$ f_{vj} such that:

$$f_{vj}(x) = r(k) \text{ for } x=2^k,\ k=0,1,2,\ldots$$
$$\qquad f(x) \text{ otherwise}$$

since such a function is a total recursive 0-variant of f by construction. Define the new sequence of functions n_i by:

$n_i(x)=f_{vi}(2^x)$

By construction then, (n_i) is an effective listing of R. This contradicts the well-known fact that R is not recursively enumerable.//

<u>THEOREM 4</u> 0-ID strictly includes ID_*.

<u>Proof</u>: Containment is immediate from the previous remarks. Let C be a singleton set containing one (arbitrary) total recursive function f. Define C′ as follows.

$$C' = \{f_r : f_r(x) = r(k) \text{ for } x=2^k,\ k=0,1,2,\ldots$$
$$f(x) \text{ otherwise}$$
$$\text{where } f \in C,\ r \in R\}.$$

Intuitively, C′ contains all the total recursive functions obtainable from f by inserting the values of other recursive functions at intervals of exponentially growing length. This construction is not effective, there being no effective listing of R, but this does not matter.

By construction, C′ is a set of total recursive 0-variants of f.

C′ is trivially 0-identifiable by the inductive inference machine that always returns an index for f.

Suppose C′ is almost everywhere (*) identifiable by some machine M. Define the new machine M′ by the following program description:

Given $\hat{g}_n$ (wnlg assume increasing enumerations):

Define SpecialEnum = (f(0), g(0), g(1), f(3), g(2), f(5), f(6), f(7), g(3), f(9), ..., g(n))

Intuitively, if g ∈ R then SpecialEnum is a partial enumeration of $f_g \in C'$. Its definition is clearly uniform in f and $\hat{g}_n$.

Let M([SpecialEnum])=i,

and define t_j=lambda x[$t_i(2^x)$].

Output j.

Intuitively, t_j is a program for a finite variant of g whenever (if ever) t_i is a program for a finite variant of f_g. Consequently M' almost everywhere (*) identifies R. This contradicts the previously cited results that ID_* is included in MATCH which is strictly included in R. Hence C' $\sim\in ID_*$ and so 0-ID strictly includes ID_*.//

<u>Remark</u>: The same argument works for the corresponding definition of 0-matching and $MATCH_*$.

<u>THEOREM 5</u> For every $h \in R$, $\epsilon > 0$, $0 \le E < 1$, $\exists$ M, uniformly in h, such that M reliably[1] $E+\epsilon$ -identifies the class of E-variants of the h-easy functions.

<u>Proof</u>: Wnlg we assume increasing enumerations. The proof proceeds via two lemmas.

<u>Lemma1</u> For every $h \in R$, $\exists$ M', uniformly in h, such that for all g [$\exists$f such that (f is h-easy) and ($DIF(f,g,n)/n \le E+\epsilon$ $\forall n$] implies (M reliably $E+\epsilon$ -identifies g)

<u>Proof</u>: Define M by the following program description.

Set FLAG = FAILURE, and begin enumerating NxN.

[1] The definition for E-identification is analogous to that for almost everywhere identification.

On $\hat{g}_n$:

If FLAG = FAILURE, then:

Find the next pair (i,m) in the enumeration of NxN. Output i. For all x such that $(x,g(x)) \in \hat{g}_n$, check whether or not $T_i(x) \leq \max\{m, h(x)\}$. If the check is satisfied, then check whether or not, for these x, $\#\{x : g(x) \neq t_i(x)\}\# / n \leq E+\epsilon$. If either of these checks fail, set FLAG to FAILURE; otherwise set FLAG to SUCCESS.

Else if FLAG = SUCCESS then:

output the index hypothesized for $\hat{g}_{n-1}$, and do the checks and flag assignments as described above for this old hypothesis. //

Intuitively, the partial recursive functions are being enumerated and checked as to whether or not they are "almost compatible" with the target. The complexity check using h ensures that the inductive inference machine knows when to stop calculating with any particular partial recursive function. The second element, m, of the enumerated pairs permits functions to have complexities that are only almost everywhere, rather than everywhere, bounded by h.

<u>Lemma 2</u> For any class C of functions, if M reliably E-identifies C then $\exists$ M′, uniformly in M, such that M′ E-identifies C′ = the class of finite variants of functions in C.

<u>Proof</u>: Let $S = (S_1, S_2, S_3, \ldots)$ be an effective enumeration of all finite sequences of "functional pairs" $(x,y) \in N \times N$, such that for (x_i,y_i), (x_j,y_j) $[x_i = x_j] \Rightarrow [y_i = y_j]$.

For any finite functional pair sequence S, and partial enumeration $\hat{f}_n$, define $S*\hat{f}_n$ to be S "concatenated" with the partial enumeration in the sense that $[(x_i,y_i) \in S*\hat{f}_n]$ iff $[(x_i,y_i) \in S$ or $(x_i,y_i) \in \hat{f}_n$ and $x_i > \max\{x : (x,y) \in S\}]$. Intuitively, S is just an initial "trial sequence" followed by the inputted partial enumeration that has been doctored so as not to contradict any of the trial sequence pairs (i.e. to preserve the functional character of the enumeration). Let f_v be a finite variant of a function $f \in C$.

On $\hat{f}_{v1}$: $c(1) := 1$

$M([S_1 * \hat{f}_{v1}])$ is returned.

On $\hat{f}_{vn}$:

If $M([S_{c(n-1)} * \hat{f}_{vn-1}]) = M([S_{c(n-1)} * \hat{f}_{vn}])$ then

COMMENT: M appears to be stabilizing so perhaps the current trial sequence is one that alters the enumeration of f_v to that of some f that M can identify.

$c(n) := c(n-1)$

the common output is returned.

otherwise:

COMMENT: M is not stabilizing so things must be arranged to try a new trial sequence for the next partial enumeration, and a result must be output that is unquestionably different from the previous output (to ensure reliability).

If c(n-1)=1 then

c(n):=n

the previous output+1 is returned.

Otherwise

c(n):=c(n-1)-1

the previous output+1 is returned.//

The method of this proof is essentially that given for an analogous result for almost everywhere identification by Minicozzi <1976>. Intuitively, the initial portions of the enumeration of f_v are replaced by increasingly long trial sequences, and the altered partial enumerations of f_v are fed to M. The goal is to stumble upon a trial sequence that alters the enumeration of f_v to that of some $f \in C$. Its achievement is detected by M's stabilization, first suspected by M's agreement upon two consecutive partial enumerations.

The proof of Theorem 5 now follows by noting that given a function f, for any E-variant f_v and arbitrary $\epsilon > 0$, $\exists$ g such that [(g is a finite variant of f_v) and $\forall n$ $DIF(f,g,n) \leq E+\epsilon$].//

The notion of E-variants employed thus far has a feature that may or may not be acceptable, depending upon the situation and the reader's inclinations. It is this: the discrepancies between a function and an E-variant may be bounded satisfactorily in an overall sense, while being overwhelming for some particular range value. For example, it is possible for a 0-1 valued 0-variant f_v of a 0-1 valued function f to be "wrong" at every point where f assumes the value 1, if only f(x)=1 implies $x=2^k$, k=0,1,2,... . This may seem appropriate since f is, in a sense, close to the almost everywhere 0 function f_v. On the other hand, viewing f as a characteristic function, it can be argued that variants should allow neither too many additions to the set nor too many omissions. E-variants simply bound the proportion of additions and omissions. Just as statistics distinguishes between Type 0 and Type 1 errors, perhaps here also each kind of error (i.e.inclusions, ommissions) should be separately bounded.

The following discussion is again in terms of total functions.

<u>Definition</u>: $DIF_r(f,g,n) = \{x: r=f(x)\neq g(x) \text{ for } x\leq n\}$

<u>Definition</u>: $DENSDIF_r = \lim_n \sup \#DIF_r(f,g,n)\#/P*\#\{x: f(x)=r \text{ for } x\leq n\}\# + (1-P)*n$, where P is the predicate: f(n)=r.

<u>Definition</u>: A function f_v is an E_{range}-variant of a function f if sup $DENSDIF_r(f,f_v) \leq E_{range}$ where max is over all $r \in$ Range(f).

The definitions for E_{range}-identification and E_{range}-ID follow those given in terms of E-variants, substituting E_{range} for E.

A development very similar to that for E-variants seems possible. There are corresponding versions of both Theorem 4 and 5.

<u>THEOREM 6</u> 0_{range}-ID strictly includes ID_*.

<u>Proof</u>: By analogy with the proof of Theorem 4. Let C contain a single recursive characteristic function c. Insert the values of recursive functions r at the 2^kth points where c(x)=1, and the 2^kth points where c(x)=0, for k=0,1,2,... . Pad the given values of $\hat{g}_n$ accordingly.//

The calculation of $DENSDIF_r$ is materially affected by whether or not the range of a function is finite. The infinite case appears to pose many new problems. Since non-fuzzy and fuzzy (grammar) languages correspond to functions with finite range, Theorem 7 will be stated in terms of characteristic functions (the extension to finite valued membership functions is obvious).

THEOREM 7 For every $h \in R$, $\theta > 0$, $0 \leq E_{range} < 1$, $\exists$ M, uniformly in h, such that M reliably $E+\theta_{range}$-identifies the class of E_{range}-variants of the h-easy characteristic functions.

<u>Proof</u>: By analogy with Theorem 5. In lemma 1 the checks performed after the complexity checks, are altered to: If g(n)=0 then check whether $\#DIF_0(t_i,g,n)\#/\#\{x: t_i(x)=0 \text{ for } x \leq n\}\# \leq E+\theta$ AND $DIF_1(t_i,g,n)/n \leq (E+\theta)$. Otherwise, if g(n)=1 then do the checks obviously corresponding to those

just listed.//

How these two notions of fuzzy variants are related is stated in the next proposition. As might be expected, bounding the number of discrepancies for each member of the target's range results in the overall number of discrepancies being bounded also, i.e.

<u>Proposition</u> Given that Range(f) is finite, f_v is an E_{range}-variant of f implies that f_v is an E-variant of f.

<u>Proof</u>:

<u>Lemma</u> $\sum_{i=1}^{n} m_i / \sum_{i=1}^{n} n_i \leq \max_{1 \leq i \leq n} (m_i/n_i)$, where $m_i \leq n_i \neq 0$, and $m_i, n_i \in N$.

Proof: By induction on n.

The lemma is trivially true for n=1.

Suppose n=k.

First of all,

$(a+b)/(c+d) \leq b/d$ if $a/c \leq b/d$, for $a,b,c,d \in N$; $c,d \neq 0$.

$\sum^{k} m_i / \sum^{k} n_i = (\sum^{k-1} m_i + M) / (\sum^{k-1} n_i + N)$

where $M/N = \max_{1 \leq i \leq k} (m_i/n_i)$

By the inductive assumption $\sum^{k-1} m_i / \sum^{k-1} n_i \leq M/N$.

Therefore, by the introductory observation,

$(\sum^{k-1} m_i + M) / (\sum^{k-1} n_i + N) \leq M/N$.

i.e. $\sum^{k} m_i / \sum^{k} n_i \leq \max_{1 \leq i \leq k} (m_i/n_i)$.//

Let $M_{nr} = DIF_r(f, f_v, n)$

and $N_{nr} = \{x : f(x) = r \text{ and } x \leq n\}$.

Then $DENSDIF(f, f_v) = \lim_n \sup (\sum_{r \in R} M_{nr} / \sum_{r \in R} N_{nr})$;

and $\max DENSDIF_r(f, f_v) = \lim_n \sup \max_{r \in R} (M_{nr}/N_{nr})$

So to establish the proposition it suffices to show that:

$\sum_{r \in R} M_{nr} / \sum_{r \in R} N_{nr} \leq \max_{r \in R} (M_{nr}/N_{nr}) \ \forall n.$

But this, given that Range(f) is finite, is precisely what the lemma shows.//

The discussion thus far in this subsection, has been in terms of functions and functional learning. However, since it has dealt with total recursive functions, the translation into a linguistic context is relatively easy following the analysis given in section 3.1. For languages with recursive membership functions (and this includes most of the commonly used types), the fuzzy models of identification presented permit the learning, given informant, of languages by approximating them with (fuzzy) languages that, while infinitely different, are sufficiently similar.

There are two seemingly troublesome points with this translation for these "very approximate" models of identification. First, the results are very enumeration dependent and some enumeration of V_t*, corresponding to the increasing enumeration of N, must be specified. The standard lexicographical order seems a reasonable choice here. The second point is that the functions corresponding to the (non-fuzzy) and fuzzy (grammar) languages have finite ranges, yet since an infinite number of discrepancies between target and hypothesis are allowed by the previous "very approximate" learning criteria, the functions hypothesized may no longer have finite ranges. This is a more serious difficulty than the enumeration dependence, but can be overcome by enumerating partial recursive functions with finite ranges (these form a "recursively enumerable class" <Rogers,1967>) rather than P wherever appropriate. For non-fuzzy languages 0-1 valued partial recursive functions must be enumerated.

More precisely, in terms of fuzzy (grammar) recursive languages, the previous definitions can be altered as follows.

<u>Definition</u>: Given two fuzzy languages L and H (assumed wnlg to share terminal vocabulary V_t), $DIF_r(L,H,n) = \{s: r=m_L(s) \neq m_H(s)$ and s is one of the first n strings in the lexicographical ordering of $V_t^*\}$.

<u>Definition</u>: A language L_v is an E_{range}-variant of a language L if $\max DENSDIF_r(m_L, m_{Lv}) \leq E_{range}$, for $r \in Range(m_L)$.

The other definitions can be similarly altered.

Theorems 6 and 7 can then be reformulated as follows.

<u>COROLLARY to THEOREM 6</u>

The class of sets of recursive languages that can be 0_{range}-identified given informant, strictly includes that which can be almost everywhere identified.

Call a recursive language with an h-easy membership function an "h-easy language".

<u>COROLLARY to THEOREM 7</u>

$\forall h \in R, e > 0, 0 \leq E_{range} \leq 1, \exists M$, uniformly in h, such that M reliably $E+e_{range}$-identifies the class of E_{range}-variants of the h-easy languages.

5.2.2 E and E_{range}-identification

Subjects for future research are: *The definition of more general types of equivalence of functions and their use in defining alternative notions of "fuzzy" identification. Briefly, such tests might permit h(x) to be within some (specifiable) neighborhood of t(x), for hypothesis h and target t. Whereas currently it is the proportion of points where the hypothesis does not equal (in a non-fuzzy sense) the target function that determines the acceptability of the hypothesis, technically fuzzy notions of point equality appear to be both possible and desirable here.
*The elimination of the current dependence upon the standard enumeration as arbiter in determining the acceptable error. This might be done by generalizing either to error relative to some (arbitrary) fixed recursive enumeration of domains, or to error relative to the particular (arbitrary) enumeration which is presented to the inductive inference machine.

In both cases, the modifications required to get results corresponding to Theorems 6 and 7 are likely to be minor.

REFERENCES

ABRAMSON, H. (1973), Theory and Application of a Bottom-up Syntax-Directed Translator

AHO, A. and ULLMAN, J. (1973), The Theory of Parsing, and Compiling Vol.2

ANGLUIN, D. (1974), Easily Inferred Sequences, Memo #ERL-M499, Elec. Res. Lab., Univ. of Cal.

ANGLUIN, D. (1979), On the Complexity of Minimum Inference of Regular Sets, Information and Control , preprint

ANGLUIN, D., Finding Patterns Common to a Set of Strings, SIGACT, preprint

AUSIELLO, G. and PROTASI, M. (1975), On the Comparison of Notions of Approximation, Lecture Notes in Computer Science 32, 172-178

BARKER, S. (1957), Induction and Hypothesis

BARZDIN, J. (1974), On Synthesizing Programs given by Examples, Lecture Notes in Computer Science 5, 56-63

BARZDIN, J. and FREIVALD, R. (1972), On the Prediction of General Recursive Functions, Soviet Math. Doklady 13:5, 1224-1228

BARZDIN, YA.M. and TRAKHTENBROT, B. (1973), Finite Automata: Behavior and Synthesis,

BASKIN, A. (1974), A Comparative Discussion of Variable-

Valued Logic and Grammatical Inference, Dept. of Comput. Sci. Rep., Univ. Illinois at Urbana-Champaign

BELLMAN, R. (1973), On the Analytic Formalism of the Theory of Fuzzy Sets , Information Sciences 5, 149-156

BELLMAN, R., KALABA, R., ZADEH, L. (1966), Abstraction and pattern classification , J. Math. Analysis Applic. 13, 1-7

BERWICK, R. (1979), Learning Structural Descriptions of Grammar Rules from Examples, IJCAI-79, 56-58

BIERMANN, A. (1972), On the Inference of Turing Machines from Sample Computations , Artificial Intelligence 3, 181-198

BIERMANN, A. (1976), Approaches to Automatic Programming, Advances in Computers 15, 1-63

BIERMANN, A. and FELDMAN, J. A. (1972a), On the Synthesis of Finite-State Machines from Samples of their Behavior, IEEE Trans. Computers C-21, 592-597

BIERMANN, A. and FELDMAN, J. A. (1972b), A Survey of Grammatical Inference, Frontiers of Pattern Recognition, ed. Watanabe, S., 31-54

BIERMANN, A. and SMITH, D. (1977), The Hierarchical Synthesis of Lisp Scanning Programs, IFIP,

BLACK, M. (1970a), Reasoning with Loose Concepts, Margins of Precision, 1-14

BLACK, M. (1970b), Induction, Margins of Precision, 57-91

BLUM, L. and BLUM, M. (1975), Toward a Mathematical Theory of Inductive Inference, Information and Control 28, 125-155

BOOTH, T. and MARYANSKI, F. (1977), Inference of Finite State Probabilistic Grammars, IEEE Trans. on Computers C-26:6 521-536

CASE, J. and SMITH, C.(1978), Anomaly Hierarchies of Mechanized Inductive Inference, Proc. 10th Annual Symp. Theory of Computing, preprint

CHARNIAK, E. and WILKS, Y. (1976), Computational Semantics,

CHOMSKY, N. (1957), Syntactic Structures

CHOMSKY, N. (1965), Aspects of the Theory of Syntax,

CHOMSKY, N. (1975), Reflections on Language,

CHOU, S. and FU, K. (1976), Inference for Transition Network Grammars, Dept. Comput. Sci. Rep. 8, Purdue Univ.

COOK, C. and ROSENFELD, A. (1974), Some Experiments in Grammatical Inference, Dept. Comput. Sci. Rep., Univ. of Maryland

COY, W. and PFLUGER, J. (1979), Identification of L-Systems in the Limit, Abteilung Informatik, Universitat Dortmund

CRESPI-REGHIZZI, S., MELKANOFF, M., LICHTEN, L. (1973), A Proposal for the Use of Grammar Inference as a Tool for Designing Programming Languages, Comm. ACM 16:2, 83-90

CRESPI-REGHIZZI, S. (1971), Reduction of Enumeration in Grammar Aquisition, Proc. 2nd Int. Conf. on Artificial Intelligence, 546-552

CRISCULO, G., MINICOZZI, E., TRATTEUR, G. (1975), Limiting Recursion and the Arithmetic Hierarchy, Revue Francaise d'Automatique Informatique Recherche Operationelle December, 5-12

DALE, P. (1972), Language Development

DALEY, R. (1977), On the Inference of Optimal Descriptions, Theoret. Comput. Sci. 4, 301-319

DE PALMA, G. and YAU, S. (1975), Fractionally Fuzzy Grammars With Applications to Pattern Recognition, Fuzzy Sets and Their Applications to Cognitive and Decision Processes, ed. Zadeh, Fu, Tanaka, Shimura, 329-352

DERWING, B. (1973), Transformational Grammar as a Theory of Language Acquisition

EVANS, T. (1971), Grammatical Inference Techniques in Pattern Analysis, Software Engineering, ed. Tou, J.

FELDMAN, J. A., GIPS, J., HORNING, J., REDER, S. (1969), Grammatical Complexity and Inference, Stanford A.I. Proj. Rep. CS125, Stanford Univ.

FELDMAN, J. A. (1972), Some Decidability Results on Grammatical Inference and Complexity, Information and Control 20, 244-262

FELDMAN, J. A. and SHIELDS, P. C. (1977), Total Complexity and the Inference of Best Programs, Math. Systems Theory 10, 181-191

FREIVALD, R. (1974), Functions Computable in the Limit by Probabilistic Machines, Lecture Notes in Computer Science 28, 77-87

FREIVALD, R. (1975), Minimal Godel Numbers and their identification in the Limit, Lecture Notes in Computer Science 32, 219-225

FU, K. (1974), Syntactic Methods in Pattern Recognition,

FU, K. ed. (1977), Syntactic Pattern Recognition: Applications

FU, K. and BOOTH, T. (1975a), Grammatical Inference: Introduction and Survey - Part 1, IEEE Trans. on Systems, Man and Cybernetics SMC- 5, 95-111

FU, K. and BOOTH, T. (1975b), Grammatical Inference: Introduction and Survey - Part 2, IEEE Trans. on Systems, Man and Cybernetics SMC-5, 409-423

GAINES, B. (1975), Approximate Identification of Automata, Elec. Letters 11:18, 444-445

GAINES, B. (1977), System Identification, Approximation, and Complexity, Int. J. of Gen. Systems 3, 145-174

GAINES, B. and KAHOUT, L. (1977), The Fuzzy Decade: A Bibliography of Fuzzy Systems and Closely Related Topics, Int. J. of Man- Machine Studies 9, 1-68

GARDINER, M. (1976), On the fabric of inductive logic, and some probability paradoxes, Scientific American - Mathematical Games: March, 119-122

GILL, A. (1966), Realization of input-output relations by Sequential Machines, J. Assoc. Comp. Mach. 13, 33-42

GOETZE, B., KLETTE, R. (1974), Some Properties of Limit Recursive Functions, Lecture Notes in Computer Science 28, 88-90

GOLD, M. (1965), Limiting Recursion, J. Symb. Logic 30, 28-48

GOLD, M. (1967), Language Identification in the Limit, Information and Control 10, 447-474

GOLD, M. (1972), System Identification via State Characterization, Automatica 8, 621-636

GOLD, M. (1978), Complexity of Automaton Identification from Given Data, Information and Control, preprint

GUIHO, G. and JOUANNAUD, J. (1977), Inference of Functions with an Interactive System, Machine Intelligence 9

HADAMARD, J. (1954), An Essay on the Psychology of Invention in the Mathematical Field

HAJEK, P. (1975), On Logics of Discovery, Lecture Notes in Computer Science 32, 30-45

HAMBURGER, H. and WEXLER, K. (1973a), Identifiability of a class of Transformational Grammars, Approaches to Natural Language, ed. Hintikka et al., 153-166

HAMBURGER, H. and WEXLER, K. (1973b), On the Insufficiency of Surface Data for the Learning of Transformational Languages, Approaches to Natural Language, ed. Hintikka et al., 167-179

HAMBURGER, H. and WEXLER, K. (1975), A Mathematical Theory of Learning Transformational Grammar, J. Math. Psych. 12, 137-177

HARTMANIS, J. and HOPCROFT J. (1971), An Overview of the Theory of Computational Complexity, J. Ass. Comp. Mach. 18:3, 444-475

HENDEL, R. (1979), Mathematical Learning Theory: A Formalized, Axiomatic, Abstract Approach, Information and Control 41, 67-117

HERMAN, G. and WALKER, A. (1972), The Syntactic Inference Problem applied to Biological Systems, Machine Intelligence 7, 341-356

HOPCROFT, J. and ULLMAN, J. (1969), Formal Languages and Their Relation to Automata

HORNING, J. (1969), A Study of Grammatical Inference, Ph.D. diss., Dept. of Comput. Sci., Stanford Univ.

HORNING, J. (1972), A Procedure for Grammatical Inference, Information Processing 71, 519-523

HUMPHREYS, P. (1977), Randomness, Independence, and Hypotheses, Synthese 36, 415-426

IBARAKI, T. (1976), Finite Automata Having Cost Functions, Information and Control 31, 153-176

IBARAKI, T. (1978), Finite Automata Having Cost Functions: Nondeterministic Models, Information and Control 37, 40-69

ISAACSON, E. and KELLER, H. (1966), Numerical Methods

JEROSLOW, R. (1975), Experimental Logics and TRIANG2-Theories, J. Phil. Logic 4, 253-267

JOUANNAUD, J. and KODRATOFF, Y. (1979), Characterization of a Class of Functions Synthesized from Examples by a Summers-Like Method Using a B.M.W. Matching Technique, IJCAI-79, 440-447.

KAPLAN, R. (1972), Augmented Transition Networks as Psychological Models of Sentence Comprehension, Artificial Intelligence 3, 77-100

KATZ, J., FODOR, J. (1963), The Structure of a Semantic Theory, Language 39, 170-210

KICKERT, W. and KOPPELA, H. (1976), Application of Fuzzy Set Theory to Syntactic Pattern Recognition of Handwritten Capitals, IEEE Trans. on Systems, Man and Cybernetics SMC-6, 148-151

KIM, H., MIZUMOTO, M., TOYODA, J., TANAKA, K. (1975), L-Fuzzy Grammars, Information Sciences 8, 123-140

KINBER, E. (1974), On a Theory of Inductive Inference, Lecture Notes in Computer Science 56, 435-440

KLEIN, S. and KUPPIN, M. (1970), An Interactive Heuristic Program for Learning Transformational Grammars, Tech. Rep., Dept. of Computing Science, University of Wisconsin

KLIR, G. (1976), Identification of Generative Structures in Empirical Data, Int. J. Gen. Systems 3, 89-104

KNOBE, B. and KNOBE, K. (1976), A Method for Inferring Context-Free Grammars, Information and Control 31, 129-146

KOHAVI, Z. (1978), Switching and Finite Automata Theory

KUGEL, P. (1977), Induction, Pure and Simple, Information and Control 35, 276-336

KUTSCHERA, VON F. (1973), Induction and the Empiricist Model of Knowledge, Studies in Logic and the Foundations of Mathematics 74, Proc. of 4th Int. Cong. for Logic, Methodology, and Philosophy of Science, Bucharest 1971, 345-356

LARSON, J. and MICHALSKI, R. (1977), Inductive Inference of VL Decision Rules, SIGART June

LEE, E. and ZADEH, L. (1969), Note on Fuzzy Languages, Information Sciences 1, 421-434

LEE, H. and FU, K. (1972), A Syntactic Pattern Recognition System with Learning Capability, Proc. Int. Symp. Comput. and Inf. Sci., 14-16

LEVELT, W. (1974), Formal Grammars in Linguistics and Psycholinguistics Vol.1, 2, 3

LEVIN (1973), On the Notion of a Random Sequence, Soviet Math. Docklady 14, 1413-1416

LINDNER, R. (1974), On the Theory of Inference Operators, Proc. Int. Congress of Mathematics Vancouver Vol. 2, 471-475

LIOU, J. and DUBES, R. (1977), A Constructive Method For Grammatical Inference Based on Clustering, Dept. Comput. Sci. Report, Michigan State Univ.

LYNCH, N. (1974), Recursive Approximations to the Halting Problem, J.C.S.S.

MADDEN, E. (ed.) (1960), Theories of Scientific Method: The Renaissance through the 19th Century

MELLISH, M. (1978), Some Prediction Algorithms for

Nonrecursive Sequences, Information and Control 37, 234-239

MELTZER, B. (1970), Generation of hypotheses and theories, Nature 225, 972

MICLET,L. (1976), Inference of Regular Expressions, 3rd I.J.C.P.R., 100-105

MILLER, G. (1967), Project Grammarama, The Psychology of Communication, 125-187

MINICOZZI, E. (1976), Some Natural Properties of Strong-Identification in Inductive Inference, Theoret. Comput. Sci. 2, 345-360

MIZUMOTO,M., TANAKA,K. (1976), Some Properties of Fuzzy Sets of Type 2, Information and Control 31, 312-340

MIZUMOTO, M., TANAKA,K. and TOYODA,J. (1973), N-fold Fuzzy Grammars, Information Sciences 5, 25-43

MOORE, E. (1956), Gedanken Experiments on Sequential Machines, Automata Studies, ed. Shannon, C. and McCarthy, J.

MORGAN, C. (1971a), Hypothesis Generation by Machine, Artificial Intelligence 2, 179-187

MORGAN, C. (1971b), On the Algorithmic Generation of Hypotheses, Dept. Phil. Paper, Univ. Alberta

MORGAN, C., Inductive Resolution, M.Sc. Thesis, Dept. of Computing Science, Univerisity of Alberta, 1972

PAIR, C. (1976), Inference for Regular Bilanguages, Formal Languages and Programming, ed. Anguilar, R., 15-30

PAO, T. (1969), A Solution of the Syntactical Induction-Inference Problem for a Non-Trivial Subset of Context Free Languages, Rep.#70-19, The Moore School of Electrical Engineering, Univ. of Penn.

PLOTKIN, G. (1971), A further note on inductive generalization, Machine Intelligence 6, 101-124

PODNIEKS, K. (1975), Probabilistic synthesis of enumerable classes of functions, Doklady. Akad. Nauk. SSSR 223, 1071-1074

POPPER, K. (1968), The Logic of Scientific Discovery,

PORTER, G. (1976), Grammatical Inference based on Pattern Recognition, Third I.J.C.P.R., 90-93

PUDLAK, P. (1975), Polynomially Complete Problems in the Logic of Automated Discovery, Lecture Notes in Computer Science 32, 358-361

PUTNAM, H. (1965), Trial and Error Predicates and the Solution to a Problem of Mostowski, J. Symb. Logic 20, 49-55

REBER, A. (1976), Implicit Learning of Synthetic Language: The Role of Instructional Set, J. Exp. Psych.: Human Learning and Memory 2(1), 88-94

REEKER, L. (1976), The Computational Study of Language Acquisition, Advances in Computers 15, 181-235

ROGERS, H. (1967), Theory of Recursive Functions and Effective Computability

ROSE, G. and ULLIAN, J. (1963), Approximation of Functions on the Integers, Pacific J. of Math., 693-701

SALMON, W. (1973), Confirmation, Scientific American, May

SANTOS, E. (1974), Context-Free Fuzzy Languages, Information and Control 26, 1-11

SCHUBERT, L. (1974a), Iterated Limiting Recursion and the Program Minimization Problem, J. Assoc. Comput. Mach. 21, 436-445

THOMASON, M. (1973), Finite Fuzzy Automata, Regular Fuzzy Languages, and Pattern Recognition, Pattern Recognition 5, 383-390

THOMASON, M., and MARINOS, P. (1974), Deterministic Acceptors of Regular Fuzzy Languages, IEEE Trans. on Systems, Man, and Cybernetics SMC-4, 228-230

TOULMIN, S. (1963), Foresight and Understanding

TOULMIN, S. (1973), Rationality and the Changing Aims of Inquiry

TREISTER, R., CARR, W. (1978), Example Induced Programming Using a Desk Calculator, CIPS, 1978

TSICHRITZIS, D. (1969), Measures on Countable Sets, Dept. Comput. Sci. Rep. 8, Univ. Toronto

TSICHRITZIS, D. (1971), Approximation and Complexity of functions on the integers, Information Sciences 3, 77-86

TURING, A. (1950), Computing Machinery and Intelligence, Mind 59, 433-460

VAN DER MUDE, A. and WALDER, A. (1978), On the Inference of Stochastic Regular Grammars, Information and Control 38 310-329

WATANABE, S. (1969), Knowing and Guessing

WATANABE, S. (1971), Ungrammatical Grammar in Pattern Recognition, Pattern Recognition 3, 385-408

WECHLER, W. (1975), R-Fuzzy Grammars, Lecture Notes in Computer Science 32, 450-456

WHARTON, R. (1974), Approximate Language Identification, Information and Control 26, 236-255

SCHUBERT, L. (1974b), Representative Samples of Programmable Functions, Information and Control 25, 30-44

SCHUBERT, L. (1977), Predictability and Randomness, Tech. Rep., Dept. of Comput. Sci., Univ. of Alberta

SHAMIR, E. (1962), A Remark on Discovery Algorithms for Grammars, Information and Control 5, 246-251

SHRIER, S. and BROWN (1978), Abduction Algorithms for Grammar Discovery, Div. Applied Math. Rep., Brown Univ.

SIKLOSSY, L. and SYKES, D. (1975), Automatic Program Synthesis from Example Problems, Advance Papers of the 4th IJCAI, 268-273

SCHNORR, C. (1973), Process Complexity and Effective Random Tests, J. Computer and Systems Sciences 7, 376-388

SOLOMONOFF, R. (1959), A New Method for Discovering the Grammars of Phrase Structure Languages, Information Processing

SOLOMONOFF, R. (1964), A Formal Theory of Inductive Inference, Information and Control 7, 1-22, 224-254

SOLOMONOFF, R. (1975), Inductive Inference Theory - A Unified Approach to Problems in Pattern Recognition and Artificial Intelligence, Advance Papers of the 4th IJCAI, 274-280

STALLINGS, W. (1977), Fuzzy Set Theory Versus Bayesian Statistics, IEEE Transactions on Systems, Man and Cybernetics SMC-7, 216-219

STOY, J. (1977), Denotational Semantics

TAMURA, S. and TANAKA, K. (1973), Learning of Fuzzy Formal Language, IEEE Transactions on Systems, Man and Cybernetics SMC-3, 98-102

WHARTON, R. (1977), Grammar Enumeration and Inference, Information and Control 33, 253-272

WIEHAGEN, R., (1975), Inductive Inference of Recursive Functions, Lecture Notes in Computer Science 32, 462-464

WIEHAGEN, R. (1977), Identification of Formal Languages, Lecture Notes in Computer Science 53, 571-579

WIEHAGEN, R. (1978), Characterization Problems in the Theory of Inductive Inference, Lecture Notes in Computer Science 62, 494-508

WILLIS, D. (1970), Computational Complexity and Probability Constructions, J.A.C.M. 17, 24-259

WINSTON, P. (1970), Learning Structural Descriptions from Examples, MAC-TR-76, M.I.T. Proj. MAC

ZADEH,L. (1965), Fuzzy Sets, Information and Control 8, 338-353

ZADEH,L. (1970), Fuzzy Languages and their Relation to Human and Machine Intelligence, Man and Computer Proc. Int. Conf., Bordeaux, 130-165

ZADEH,L. (1977), A Theory of Approximate Reasoning, Memo.#UCB/ERL M77/58, Elec. Res. Lab., Univ. of Calif., Berkley

ZALECKA-MELAMED,A. (1977), Structural Inference and Identification of Discrete Time Systems, Logic of Computers Group Rep. 200, Computer and Communication Sciences Dep., Univ. of Michigan