*Electronic Notes in Theoretical Computer Science 53 (2001)*

*URL:* [*http://www.elsevier.nl/locate/entcs/volume53.html*](http://www.elsevier.nl/locate/entcs/volume53.html) *14 pages*

*Learnability of type-logical grammars Sean A. Fulop 1;2*

*Depts. of Linguistics and Computer Science The University of Chicago*

*Chicago, U.S.A.*

*Abstract*

*A procedure for learning a lexical assignment together with a system of syntactic and semantic categories given a xed type-logical grammar is brie y described. The logic underlying the grammar can be any cut-free decidable modally enriched extension of the Lambek calculus, but the correspondence between syntactic and semantic categories must be constrained so that no in nite set of categories is ultimately used to generate the language. It is shown that under these conditions various linguistically valuable subsets of the range of the algorithm are classes identi able in the limit from data consisting of sentences labeled by simply typed lambda calculus meaning terms in normal form. The entire range of the algorithm is shown to be not a learnable class, contrary to a mistaken result reported in a preliminary version of this paper. It is informally argued that, given the right type logic, the learnable classes of grammars include members which generate natural languages, and thus that natural languages are learnable in this way.*

*1 Introduction*

*Recent investigations (e.g. [12,13]) have shown the potential for the description* of natural language syntax promised by forms of Lambek's syntactic calculus

*[10] enriched by multiple combination modes and families of unary modalities.* Any version of Lambek's calculus in this landscape of logical systems will be called a type logic in keeping with current practice. A type-logical grammar is then a triple G = (VG; IG; RG) consisting of a vocabulary VG, a lexical assignment function IG, and a type logic RG.

*1 Thanks to my supervisor Ed Stabler, my de facto co-chair Michael Moortgat, and Makoto Kanazawa for his inspiring work. Special appreciation to Christophe Costa Florencio for helping me iron out the mistakes in the preliminary version. This work was supported in part by a UCLA Dissertation Year Fellowship, NSF LIS grant 9720410, an OTS-UCLA exchange grant awarded to Prof. Dr. M. Moortgat, and a Research Incentive grant from the University of Chicago Dept. of Computer Science.*

*2 Email:* [*sfulop@uchicago.edu*](mailto:sfulop@uchicago.edu)

*c 2001 Published by Elsevier Science B. V. Open access under* [*CC BY-NC-ND license.*](http://creativecommons.org/licenses/by-nc-nd/3.0/)

*This paper proceeds according to the following:*

*Type-logical grammar as a framework for syntactic description is brie y* summarized.

*The syntax-semantics connection between type logic and the lambda calcu-* lus is outlined, as a generalization of the Curry-Howard morphism.

*We outline a procedure OUTL for learning type-logical lexicons from sen-* tences plus lambda terms, which works for any type logic meeting certain conditions.

*It is shown that the entire class of languages generated by the grammars* discoverable using OUTL is not in general a learnable class, in Gold's [5] sense of identi ability in the limit.

*It is then shown how restricting to certain subclasses of the language class* L(Rng(OUTL)) can give rise to learnability.

*Finally, the rami cations for theoretical psycholinguistics and computa-* tional induction of grammars are mentioned.

*The focus here is on the learnability results; this paper is part of a larger* research project intended to show the feasibility of a particular implementa- tion within the type-logical framework of the psycholinguistic hypothesis that human languages can be learned via semantic bootstrapping [14].

# *2 Type-logical grammar*

*De nition 2.1 A type-logical grammar can be de ned as a triple G = (VG; IG; RG) such that:*

*(i) VG, the vocabulary of G, is a non-empty nite set;*

*(ii) IG, the lexicon of G, is a function which to each v 2 VG assigns a nite* set of types;

*(iii) RG, the calculus of G, is a type logic.*

*The non-associative Lambek calculus NL serves as a \base logic" for the* kinds of type logical grammars that the present paper applies to. The systems of inference rules shown below conform to the logical style of Gentzen's [4] sequent calculus.

*De nition 2.2*

*(Axiom)*

*A ) A*

*(Cut)*

*) A [A] ) C*

*[ ] ) C*

*(= L)*

*) B [A] ) C*

*[(A=B )] ) C*

*(n L)*

*) B [A] ) C*

*[( BnA)] ) C*

*(= R)*

*( B) ) A*

*) A=B*

*(n R)*

*(B ) ) A*

*) BnA*

*Extensions of NL have been developed in the literature [11,13] which em-* ploy unary operators that are analogous to those found in modal logics. Here are the sequent rules of inference governing the unary \modal" operators, using their corresponding structural operator.

*(3L)*

*(3R)*

*) A*

*[hAi] ) B*

*[3A] ) B*

*( #L)*

*[A] ) B*

*[h #Ai] ) B*

*h i ) 3A*

*( #R)*

*h i ) A*

*) #A*

*A major purpose of introducing modal operators is to license the use of* structural rules, in e ect restricting their applicability to certain contexts. Typically, extended type logics also employ more than one \family" of slash operators, and these are indexed using subscripts. The di erent families of slashes are sometimes called modes of combination.

*A simple example illustrating the use of the modal enrichments is provided* by the treatment of English wh-extraction to form a relative clause, taken from Puite and Moot [15]. Take the following lexicon, in which the wh-word has unary modal operators in its assigned type.

*(1) IG(agent) = n*

*IG(Mulder) = np*

*IG(liked) = (npns)=np*

*IG(which) = (nnn)=(s=3a #anp)*

*The following structural rule permits associativity only in the presence of* the appropriate structural modal environment.

*[(*

*) h ia] ) C*

*[*

*( h ia)] ) C*

*(Assoc)*

*1 2 3*

*1 2 3*

*[ 1 ( 2 h 3ia)] ) C [( 1 2) h 3ia] ) C*

*The gure shows a proof of the complex noun agent which Mulder liked, which has a desired constituent structure.*

*np : Mulder ) np*

*np : which ) np s : [Mulder, [liked, which]] ) s Ln np npns : [liked, which] ) s*

*np ((npns)=np np): [Mulder, [liked, which]] ) s*

*L=*

*L #*

*np ((npns)=np h # npia ): [Mulder, [liked, which]] ) s*

*a*

*a*

*(np (npns)=np) h # npia : [[Mulder, liked], which] ) s Assoc*

*a*

*(np (npns)=np) #*

*a*

*L*

*R= agent [agent, [which, [Mulder, liked]]] Ln*

*np : [[Mulder, liked], which] ) s n : ) n n : ) n*

*np (npns)=np : [Mulder, liked] ) s= # np n nnn : [agent, [which, [Mulder, liked]]] ) n*

*a*

*n ((nnn)=(s= #*

*a*

*L=*

*np) (np (npns)=np)) : [agent, [which, [Mulder, liked]]] ) n*

*Fulop*

*4*

## *3 Grammar discovery from semantics*

*3.1 The syntax-semantics connection*

*A general algorithm has been developed [3] for learning a syntactic category* system for a natural language together with a lexical assignment that de- termines a completely descriptive grammar. The syntax can in principle be handled using any cut-free decidable multimodal type-logical grammar [11]. The learning data consists of term-labeled strings, i.e. sentences annotated by typed lambda calculus meaning recipes with either variable semantic types on the subterms, or with no types on the subterms. The lambda calculus is used in a standard fashion to model the compositional meaning structures of natural language; an example of a term-labeled string is:

*(2) ((loves(s ) (Mary ))(John ))s : hJohn; loves; Maryi*

*alternatively, without subterm types*

*(3) ((loves(Mary))(John))s : hJohn; loves; Maryi*

*The meaning recipes show the basic compositional construction of the sen-* tence meaning in terms of application and abstraction, and can be used as directionally non-speci c recipes for the construction of type-logical proofs of the labeled sentence which correspond via a generalized Curry-Howard homo- morphism.

*In more detail, let us recall that the Curry-Howard formulae-as-types in-* terpretation induces a morphism from lambda terms to proofs, such that a lambda term can be used as a recipe to construct a proof in a logic whose operators correspond to the lambda type operators. Considering the simply typed lambda calculus, the terms can be used to construct valid proofs in the positive implicational propositional logic, whose only logical operator is the implication.

*Now, for type logics based on the non-associative Lambek calculus, there* is no longer a direct correspondence with the simply typed lambda calculus, because there are more logical operators in the logic than just the implication found in the lambda calculus types. The key di erence is that type logics in general will be sensitive to the directional (left-right) relationship among the formulae, and there may also be modal operators and structural rules invoked that have no counterpart in the simply typed lambda calculus. In fact, however, any type logic RG with the typical pair of slash operators can be shown to induce a fragment RG of the ordinary typed lambda calculus such that for every sequent ) sG in RG provable by some proof there will be a term M sG 2 RG such that M is a construction modulo direction of by a generalization of the Curry-Howard isomorphism [7] to a homomorphism. The initial idea for this is found in van Benthem [18].

*De nition 3.1 The mapping from syntactic types in the type logic to se-* mantic types in the lambda calculus is de ned as follows. The semantic types that are mapped to in this way are said to be equivalent modulo direction of application (emda) to the correponding syntactic types, and vice versa. The function M ( ) is used as a schematic, in which M stands for any string of unary modal operators. The subscript i signi es the possibility of having multiple families of slash operators.

*(M (c)) = c for corresponding primitive types c; c ;*

*0 0*

*(M (A=iB)) = (M (BniA)) = (A) (B)*

*Notice that, in general, a single semantic type is said to be equivalent modulo* direction to an in nite number of possible syntactic types.

*3.2 A discovery procedure*

*In order to rst discover a general form lexicon as an intermediate step, which* assigns syntactic types whose primitive types are distinct variables except for the principal type constant `s' which is assigned to sentences, our algorithm is intended to learn from a sample of term-labeled strings such as the above ex- ample, whose labels either contain no explicit semantic types on the subterms (they are unsubtyped ), or are subtyped but contain only such semantic types in which the primitive types are all variables except for the principal type `s' of a sentence. The complete algorithm is charged with learning the system of categories (other than `s') together with the lexical assignment function, for a

*xed vocabulary and type logic.*

*The broad outline of the procedure, called Optimal Uni cation for Type-* Logical grammars (OUTL), is as follows:

*(i) Given a sample D of unsubtyped term-labeled strings, compute a counter-* part sample D0 of subtyped term-labeled strings whose terms are typed in a most general way. This can be accomplished using some kind of principle type algorithm, such as the one which is discussed at length in [6].

*(ii) Compute the set of general form type-logical lexicons GFTL(D0 ), in each* of which distinct variable primitive types will each occur atomically only once, and such lexicons will generate only the sample D0 and not an in nite language. This is accomplished by taking the following steps:

*(a) For each term-labeled string in the sample, determine all proofs in the*

*type logic at hand which can be constructed by using the subtyped* lambda term as a construction term modulo direction, and which are also compatible with the word order that is evident in the sentence.

*(b) Non-deterministically select one proof for each term-labeled string in*

*the entire sample; a general form lexicon can then be read o from the*

*types labeling the words. Repeat this step until all di erent ways of* selecting one proof for each term-labeled string have been exhausted. This will provide all general form lexicons that could generate the learning sample.

*(iii) Find all of the optimal uni cations [1] of each of the lexicons in GFTL(D0 ).*

*The language class generated by the entire range of OUTL (which assumes* a xed decidable cut-free type logic) is not learnable in Gold's [5] sense, but suÆciently large subclasses are learnable provided that the generalized Curry- Howard correspondence is also nitary in the following sense.

*De nition 3.2 The relation of equivalence modulo direction of application* (emda) between semantic and syntactic types is said to be nitary just when for each semantic type , the set fT j emda(T; )g of syntactic types equivalent to it is nite, and for each syntactic type T , the set f j emda(T; )g of semantic types equivalent to it is nite.

*No class of type-logical languages that does not meet this condition is* learnable by the above discovery procedure.

*The logics found in linguistic applications, including of course the basic* nonassociative Lambek calculus but also various extended systems such as the Dutch system presented in [12], either already satisfy the above restrictions, or can be made to by some ad hoc means to force a nitary syntax-semantics correspondence. Note, however, that no type logic enriched with unary modal- ities can have a nitary emda relation with the simply typed lambda calculus in general. The sequent proof corresponding to a given lambda term might have any number of modal operators tacked onto its consequent type; a indef- inite combinatorial explosion of available sequences of modal and structural rules would then have to be explored to nd the ones that eventually get rid of all of the modals so that the logical rule that corresponds to some appli- cation or abstraction in the lambda term can nally be invoked. To try to quell this explosion, we can stipulate a nitary correspondence by saying that only syntactic types with fewer than some small number k modal operators will be considered in the search for corresponding proofs. This way of meeting the nitary correspondence requirement is mathematically ad hoc, but does not seem linguistically unrealistic. Do we really wish to countenance an in-

*nite number of syntactic categories for a given semantic category? There*

*is probably some de nite upper bound on the number of distinct syntactic* behaviors that a member of a particular semantic category will be found to have in natural language, and in practice, some rather sophisticated linguistic phenomena can already be handled with a maximum of two modal operators adorning the types.

*To demonstrate how the learning turns out, a simple example shows the* results of applying the OUTL procedure to two di erent samples of four anno- tated sentences, which have the same vocabulary. The procedure settles on a

*grammar for the same language in each case (the same grammar too, in fact).*

*(4) (sings(John))s : hJohn; singsi*

*((loves(Mary))(John))s : hJohn; loves; Maryi ((loves(a(man)))(Mary))s : hMary; loves; a; mani*

*((sees(John))(a(man)))s : ha; man; sees; Johni*

*(5) (sings(Mary))s : hMary; singsi*

*((loves(John))(Mary))s : hMary; loves; Johni ((loves(Mary))(a(man)))s : ha; man; loves; Maryi*

*((sees(a(man)))(John))s : hJohn; sees; a; mani*

*The general form lexicons discovered for the above two samples by exploit-* ing the Curry-Howard homomorphism optimally unify to the same lexicon, presented as an assignment function IG:

*(6) IG(Mary) =*

*IG(sings) = ns*

*IG(loves) = ( ns)= IG(John) =*

*IG(a) = = IG(man) =*

*IG(sees) = ( ns)=*

*It should be noted that this implementation of semantic bootstrapping* uses much less information than has frequently been considered in such ef- forts. The basic requirements of Pinker's [14] proposal for a bootstrapping procedure include the Canonical Structure Realization, through which the syntactic realization of known semantic categories is provided as a part of in- nate Universal Grammar. This requires the would-be semantic bootstrapper to already know the system of semantic categories as well as their correspond- ing syntactic ones. An implementation which follows this tack is presented by [16]. Our implementation, in contrast, does not provide complete information about syntactic structure, and it provides no speci c information about the particular semantic or syntactic categories which should be used to generate the language.

## *4 Learnability of optimally uni ed lexicons*

*De nition 4.1 [8] Let h ; S; Li be a grammar frame, consisting of a set of grammars, a set S of expressions (sentences), and a function L which maps* from the grammars to sets of expressions (i.e. languages). A learning function is a partial function ' that maps non-empty nite sequences of sentences to

*grammars:*

*' : [ Sk ! :*

*k 1*

*Sk denotes the set of k-ary sequences of sentences. A learning algorithm is* one that computes a learning function.

*De nition 4.2 Let a grammar frame h ; S; Li be given, let G . A learn-* ing function ' is said to learn G if the following condition holds:

*for every language ` in L(G),*

*for every in nite sequence hsiii2N that enumerates the elements of ` (i.e.* fsi j i 2 N g = `),

*there exists some G in G such that L(G) = ` and ' converges to G on hsiii2N .* [8]

*4.1 A negative result*

*De nition 4.3 A class L of languages is said to have a limit point if there* exists an in nite sequence hLnin2N of languages in L such that

*L0 L1 Ln*

*(we call this an in nite ascending chain) and there is another language L in* L such that

*1*

*L = [ Ln:*

*n=0*

*The language L is said to be a limit point of L. [8]*

*Lemma 4.4 ([8]) If L(G) has a limit point then G is not learnable.*

*Thus, we can show a class of languages is not learnable by showing it has* a limit point.

*Theorem 4.5 The language class L(Rng(OUTL)) generated by the range of the OUTL algorithm is not a learnable class of (term-labeled) languages for any type logic.*

*Proof. In a fashion analogous to the proof of Kanazawa's [8] Theorem 7.20,* we prove that the term-labeled language class L(Rng(OUTL)) has a limit point, even when using just the AB classical type logic (equivalent to the binary combination version of classical categorial grammar) restricted to the forward slash.

*(i) Consider the following lexical assignment involving a single vocabulary* item:

*(7) IG(a) = fx; x=x; (s=(x=x))=xg*

*This assignment is optimally uni ed. The term-labeled language gener-* ated using AB and IG consists of the in nite set ftlsng of term-labeled strings, where the items Tn are de ned as follows:

*(a(s (x x)) x (ax x ( (ax) )x)s (x x)ax x : ha*

*| n t{imz es }*

*n+3*

*in 0*

*in which the types are shown for convenience, so that the manner in* which they are derived is easier to see.

*(ii) Now, for each n 0, de ne the type An by the following recurrence:*

*A d=ef x*

*0*

*(8) A d=ef (s=x)=x*

*1*

*A d=ef (s=A )=(s=A ):*

*n+2 n+1 n*

*Next, let IGn name the following assignment for all n 0:*

*(9) IGn(a) = fA0; : : : ; An+1g:*

*All lexicons IGn are optimally uni ed, and L(Gn) = ftls0; : : : ; tlsng for* all n 2 N , so the grammars Gn de ne an in nite ascending chain of languages.

*(iii) Finally, it can be veri ed (most easily by running the software) that IG* is in Rng(OUTL), as are all of the IGn. In fact, IG 2 OUTL(ftls0; tls1g), and IGn 2 OUTL(ftls0; : : : ; tlsng) for each n 0.

*(iv) Notice that L(G) = S1*

*n=0*

*L(Gn*

*), making L(G) a limit point in the*

*range of OUTL.*

*2*

*4.2 Positive results*

*The following notion, though de ned below as in [9], has its roots in Wexler* and Hamburger [19].

*De nition 4.6 The language L 2 L is said to be an accumulation point just when there exists a sequence of nite sets S0; S1; : : : such that*

*(i) 8i 2 N Si Si+1;*

*(ii) S1*

*i=0*

*Si = L;*

*(iii) 8i 2 N 9L0 2 L(Si L0 & L0 L).*

*The existence of an accumulation point in a language class is both neces-* sary and suÆcient for the class to be unlearnable. Thus, we can prove a class learnable by showing it cannot have an accumulation point.

*De nition 4.7 A syntactic type A which is a subtype of some type assigned by the lexicon IG of a type-logical grammar G is said to be useless just when there is no proof available in G that uses A other than as a proper subtype.*

*In fact, it is apparent from the discovery procedure that the algorithm OUTL can never output a lexicon containing useless types.*

*De nition 4.8 A type-logical grammar is k-valued just when its lexicon IG assigns no more than k types to any one vocabulary element.*

*Lemma 4.9 Given any decidable type logic RG meeting the nitarity condi- tion of Def. 3.2, for any learning sample of term-labeled strings which exhausts the vocabulary VG, there is a bounded number of distinct (modulo alphabetic variant) k-valued lexicons IG without useless types for any k.*

*Proof. The argument is combinatorial. Any sample of term-labeled strings determines a lexicon of semantic type schemata|a general form semantic lexicon. Clearly from the condition of Def. 3.2, there is some maximum number of distinct 1-valued lexicons corresponding to any such semantic lexicon, so long as there are no useless types permitted. The precise number can in principle be determined in any case, but it will depend on the nature of the semantic lexicon (which varies from sample to sample) and the speci cs of the syntax-semantics correspondence (which varies from logic to logic). The same will be true of the 2-valued, 3-valued, etc. lexicons for any k. 2*

*Theorem 4.10 Given any decidable type logic RG meeting the nitarity con- dition of Def. 3.2, no language class generated by a k-valued class of grammars in Rng(OUTL) can have an accumulation point.*

*Proof. Suppose for some k, there were a class of grammars in Rng(OUTL) whose class of generated term-labeled languages has an accumulation point*

*L. By de nition, L = Si=0 Si is the in nite union of a weakly ordered*

*1*

*chain of sets of term-labeled strings. Using the Si as learning samples, no*

*discovered IGi can have the full generating power of IG which generates L. This is just what it means to be an accumulation point.*

*Now, let us see that the actual situation contradicts this desideratum. Since the full IG generating L must be k-valued for some k, L may use just a bounded number of semantic type schemata for each word. This bound will depend on k and on the nature of the type logic RG. By Lemma 4.9, given any semantic lexicon of type schemata assigned to the vocabulary, there is a determinable number of distinct optimally uni ed k-valued lexicons IGi without useless types. For the complete semantic lexicon, which is bounded in extent, one of these IGi must actually generate L using the assumed type logic RG. Since the chain hSii converges to L, all the languages Li which contain the sets must equal L after a certain point m in the chain. This point m is a function of the number of distinct k-valued lexicons available, which in turn is a function of the assumed type logic RG and of the semantic*

*lexicon evident from the chain of samples. This proves that the hypothesized* accumulation point L cannot actually be one. 2

*The following is an immediate corollary from the facts about an accumu-* lation point.

*Corollary 4.11 Given any decidable type logic RG meeting the nitarity con- dition of Def. 3.2, the k-valued classes of grammars are learnable for all k.*

*The preceding proof ultimately shows that the unlearnability of any class of* type-logical lexicons depends on the possibility of unbounded lexical ambiguity within the class. That is why learnability is lost when the entire class of lexicons in the range of OUTL is considered, rather than just those with a maximum k types assigned to each word.

*De nition 4.12 If D is a nite set of term-labeled strings, let*

*def*

*0*

*LCT L(D) = fG 2 OU T L(D) j 8G*

*2 OU T L(D) (jIGj jIG0 j)g:*

*LCT L(D) picks those optimal lexicons that have smallest cardinality. The* following now seems to follow as a corollary of Theorem 4.10, though the argument is as yet informal.

*Conjecture 4.13 Given any decidable type logic RG meeting the nitarity condition of Def. 3.2, the language class L(Rng(LCTL)) is learnable.*

*The argument here is provided in part by the above remark to the ef-* fect that unlearnability of any class of lexicons depends on the possibility of unbounded lexical ambiguity. This is because the accumulation point of the generated language class must ultimately be generated in two ways|once by a possibly bounded lexicon, and again by an \in nite" lexicon that is the limit of the lexicons generating the chain of sets hSii. By selecting just those optimally uni ed lexicons with least cardinality, the algorithm LCTL ensures that these two ways of generating any language cannot exist, since only one of them will have least cardinality.

*In terms of recognizing power, the exact class of languages generated by* the class of type-logical grammars covered by these results is not known. Be- cause several type-logical fragments that are covered by these results generate non-context-free natural language fragments (the Dutch fragment from [12,3], for example), one might conjecture that a subclass of the (properly) mildly context-sensitive languages is covered. It is clear, however, that the recur- sively enumerable languages outside the context-sensitive class are not cov- ered in general, notwithstanding Carpenter's result [2] that the whole class of general multimodal type-logical grammars generate the entire Chomsky Type

*0 language class. The reason for this is that Carpenter's result depends cru-*

*cially on allowing the underlying type logics to be undecidable in general, with* the ability to add or delete arbitrary structure during the course of a proof.

*Since our type logics are required to be decidable, they do not seem to form a Turing-complete class.*

# *5 Concluding remarks*

*It has been shown that wide classes of optimally uni ed type-logical lexicons are identi able in the limit from term-labeled strings. We conjecture that these classes contain grammars strongly adequate for the description of natural languages. This is justi ed by noticing that, rstly, the learnable classes include grammars which generate languages beyond the context-free class, and secondly, the optimal uni cation procedure produces lexicons in which the assignment of distinct categories invariably re ects positive evidence in the learning sample of distinct syntactic behavior. The lexicons learned should thus in the limit assign all and only those syntactic and semantic categories which function properly in the language, a term we use to mean category di erences exactly re ect syntactic di erences. Familiar syntactic categories in linguistic theory such as \Verb" do not function properly in this sense, which is why subcategorization is required.*

*The above characterization of the present approach to learning grammars, if correct, has rami cations for the computational induction of grammars as well as for psycholinguistic ideas about language learning. It would be possible in principle to induce a precisely adequate grammar (provided one got the universal type logic correct in the rst place) from term-labeled strings, which are easier to create based upon modern semantic theories of language than are direct syntactic training databases for grammar induction such as the Penn Treebank. The Treebank induction approach also has the major disadvantage that the parts of speech are already assigned to the words, which stretches even a charitable interpretation of language learning on the psycholinguistic side, and which may propagate damaging assumptions for natural language processing e orts.*

# *References*

*[1] Buszkowski, W. and G. Penn, Categorial grammars determined from linguistic data by uni cation, Studia Logica 49 (1990), pp. 431{454.*

*[2] Carpenter, B., The Turing-completeness of multimodal categorial grammars, in: J. Gerbrandy, M. Marx, M. de Rijke and Y. Venema, editors, JFAK: Essays dedicated to Johan van Benthem on the occasion of his 50th birthday, Institute for Logic, Language, and Computation, University of Amsterdam, 1999 Available on CD-ROM at* [*http://turing.wins.uva.nl.*](http://turing.wins.uva.nl/)

*[3] Fulop, S. A., \On the Logic and Learning of Language," Kluwer, Forthcoming.*

*[4] Gentzen, G., Untersuchungen uber das logische Schliessen, Math. Zeitschrift 39 (1934), pp. 176{210, 405{431, english translation in [17].*

*[5] Gold, E. M., Language identi cation in the limit, Information and Control 10 (1967), pp. 447{474.*

*[6] Hindley, J. R., \Basic Simple Type Theory," Cambridge University Press, 1997.*

*[7] Howard, W. A., The formulas-as-types notion of construction, in: J. P. Seldin and J. R. Hindley, editors, To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism, Academic Press, New York, 1980 pp. 479{ 490.*

*[8] Kanazawa, M., \Learnable Classes of Categorial Grammars," Ph.D. thesis, Stanford University (1994).*

*[9] Kapur, S., \Computational Learning of Languages," Ph.D. thesis, Cornell University (1991).*

*[10] Lambek, J., The mathematics of sentence structure, American Mathematical Monthly 65 (1958), pp. 154{170.*

*[11] Moortgat, M., Categorial type logics, in: J. van Benthem and A. ter Meulen, editors, Handbook of Logic and Language, Elsevier, 1997 .*

*[12] Moortgat, M., Meaningful patterns, in: J. Gerbrandy, M. Marx, M. de Rijke and Y. Venema, editors, JFAK: Essays dedicated to Johan van Benthem on the occasion of his 50th birthday, Institute for Logic, Language, and Computation, University of Amsterdam, 1999 Available on CD-ROM at* [*http://turing.wins.uva.nl.*](http://turing.wins.uva.nl/)

*[13] Morrill, G. V., \Type Logical Grammar: Categorial Logic of Signs," Kluwer, Dordrecht, 1994.*

*[14] Pinker, S., \Language Learnability and Language Development," Harvard University Press, Cambridge, MA, 1984.*

*[15] Puite, Q. and R. Moot, Proof nets for the multimodal lambek calculus, Technical Report 1096, University of Utrecht, Dept. of Mathematics (1999).*

*[16] Siskind, J., Dispelling myths about language bootstrapping (1991), manuscript, MIT AI Laboratory.*

*[17] Szabo, M., \The Collected Papers of Gerhard Gentzen," North-Holland, Amsterdam, 1969.*

*[18] van Benthem, J., \Language in Action," North-Holland, Amsterdam, 1991.*

*[19] Wexler, K. and H. Hamburger, On the insuÆciency of surface data for the learning of transformational language, in: K. J. J. Hintikka, J. M. E. Moravcsik and P. Suppes, editors, Approaches to Natural Language, D. Reidel, Dordrecht, 1973 .*

*14*