TINA: A PROBABILISTIC SYNTACTIC PARSER FOR SPEECH UNDERSTANDING SYSTEMS*

Stephanie Seneff

Spoken Language Systems Group
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

ABSTRACT

A new natural language system, TINA, has been developed for applications involving speech understanding tasks, which integrates key ideas from context free grammars, Augmented Transition Networks (ATN's) [1], and Lexical Functional Grammars (LFG's) [2]. The parser uses a best-first search strategy, with probability assignments on all arcs obtained automatically from a set of example sentences. An initial context-free grammar, derived from the example sentences, is first converted to a probabilistic network structure. Control includes both top-down and bottom-up cycles, and key parameters are passed among nodes to deal with long-distance movement and agreement constraints. The probabilities provide a natural mechanism for exploring more common grammatical constructions first. Arc probabilities also reduced test-set perplexity by nearly an order of magnitude. Included is a new strategy for dealing with movement, which can handle efficiently nested and chained gaps, and rejects crossed gaps.

INTRODUCTION

Most syntactic parsers have been designed with the assumption that the input word stream is deterministic: i.e., at any given point in the parse tree it is known with certainty what the next word is. As a consequence, these parsers generally cannot be used effectively, if at all, to provide syntax-directed constraint in the speech recognition component of a spoken language system. In a fully integrated system, the recognizer component should only be allowed to propose partial word sequences that the natural language component can interpret. Any word sequences that are syntactically or semantically anomalous should probably be pruned prior to the acoustic match, rather than examined for approval in a verification mode. To operate in such a fully integrated mode, a parser should have the capability of considering a multitude of hypotheses simultaneously. The control strategy should have a sense of which of these hypotheses, considering both linguistic and acoustic evidence, is most likely to be correct at any given instant in time, and to pursue that hypothesis only incrementally before reexamining the evidence. The linguistic evidence should include probability assignments on proposed hypotheses; otherwise the perplexity of the task is much too high for practical recognition applications.

This paper describes a natural language system, TINA, which addresses many of these issues. The grammar is constructed by converting a set of context-free rewrite rules to a form that merges common elements on the right-hand side (RHS) of all rules sharing the same left-hand side (LHS). Elements on the LHS become parent nodes in a family tree. Through example sentences, they acquire knowledge of who their children are and how they can interconnect. Such a transformation permits considerable structure sharing among the rules, as is done in typical shift-reduce parsers [3]. Probabilities are established on arcs connecting pairs of right siblings rather than on rule productions. We believe this is a more reasonable

*This research was supported by DARPA under Contract N00039-85-C-0254, monitored through Naval Electronic Systems Command.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **1989**	2. REPORT TYPE	3. DATES COVERED **00-00-1989 to 00-00-1989**

4. TITLE AND SUBTITLE
TINA: A Probabilistic Syntactic Parser for Speech Understanding Systems

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Massachusetts Institute of Technology,Laboratory for Computer Science,Spoken Language Systems Group,Cambridge,MA,02139

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**11**	

way to use probabilities in a grammar. Context-dependent constraints to deal with agreement and gaps are realized by passing features and flags among immediate relatives.

GENERAL DESCRIPTION

TINA is basically a context-free grammar, implemented by expansion at run-time into a network structure, and augmented with flags/parameters that activate certain filtering operations. The grammar is built from a set of training sentences, using a bootstrapping procedure. Initially, each sentence is translated by hand into a list of the rules invoked to parse it. After the grammar has built up a substantial knowledge of the language, many new sentences can be parsed automatically, or with minimal intervention to add a few new rules incrementally. The arc probabilities can be incrementally updated after the successful parse of each new sentence.

Three Context-free Rewrite Rules:

NP ==> ARTICLE NOUN
NP ==> ARTICLE ADJECTIVE NOUN
NP ==> ARTICLE ADJECTIVE ADJECTIVE NOUN

The Resulting Probabilistic Network:

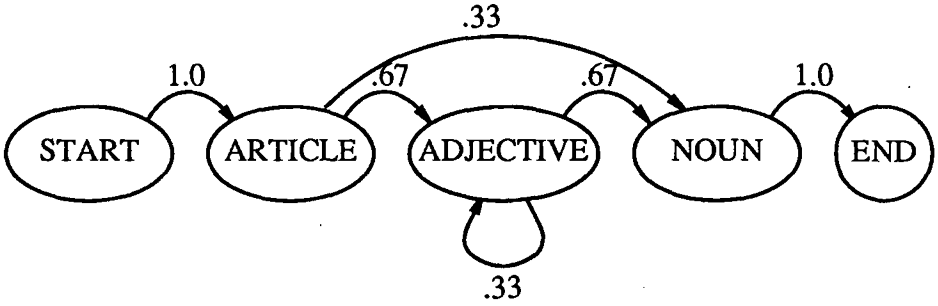

Figure 1: Illustration of Process to Convert Context-free Rules to probabilistic Network Form.

Figure 1 illustrates the process of converting a set of context-free rules to a probabilistic network structure of *grammar nodes*. All rules with the same LHS are combined to form a structure describing possible interconnections among children of a parent node associated with the left-hand category. A probability matrix connecting each possible child with each other child is constructed by counting the number of times a particular sequence of two siblings occurred in the RHS's of the common rule set, and normalizing by counting all pairs from the particular left-sibling to *any* right sibling. Two distinguished nodes, a START node and an END node, are included among the children of every grammar node. A subset of the grammar nodes are terminal nodes whose children are a list of vocabulary words.

A functional block diagram of the control strategy is given in Figure 2. At any given time, a distinguished subset of "active" *parse nodes* are arranged on a priority queue. Each parse node contains a pointer to a grammar node of the same name, and has access to all the information needed to pursue its partial theory. The top node is popped from the queue, and it then creates a number of new nodes (either children or right siblings depending on its state), and inserts them into the queue according to their probabilities. If the node is an END node, it collects up all subparses from its sequence of left siblings, back to the START node, and passes the information up to the parent node, giving that node a completed subparse. The process can terminate on the first successful completion of a sentence, or the Nth successful completion if more than one hypothesis is desired.

A parse in TINA is begun by creating a single parse node linked to the grammar node SENTENCE, and entering it on the queue with probability 1.0. This node creates new parse nodes with categories like STATEMENT, QUESTION, and REQUEST, and places them on the queue, prioritized. If STATEMENT is the most likely child, it gets popped from the queue, and returns nodes indicating SUBJECT, IT, etc., to the queue. When SUBJECT reaches the top of the queue, it activates units such as NOUN-GROUP (for noun phrases and associated post-modifiers), GERUND, and NOUN-CLAUSE. Each node, after instantiating first-children, becomes inactive, pending the return of a successful subparse from a sequence of children. Eventually, the cascade of first-children reaches the terminal-node ARTICLE, which proposes the words "the," "a," and "an," testing these hypotheses against the input stream. If a match with "the" is found, then the ARTICLE node fills its subparse slot with the entry (ARTICLE "the"), and activates all of its possible right-siblings.

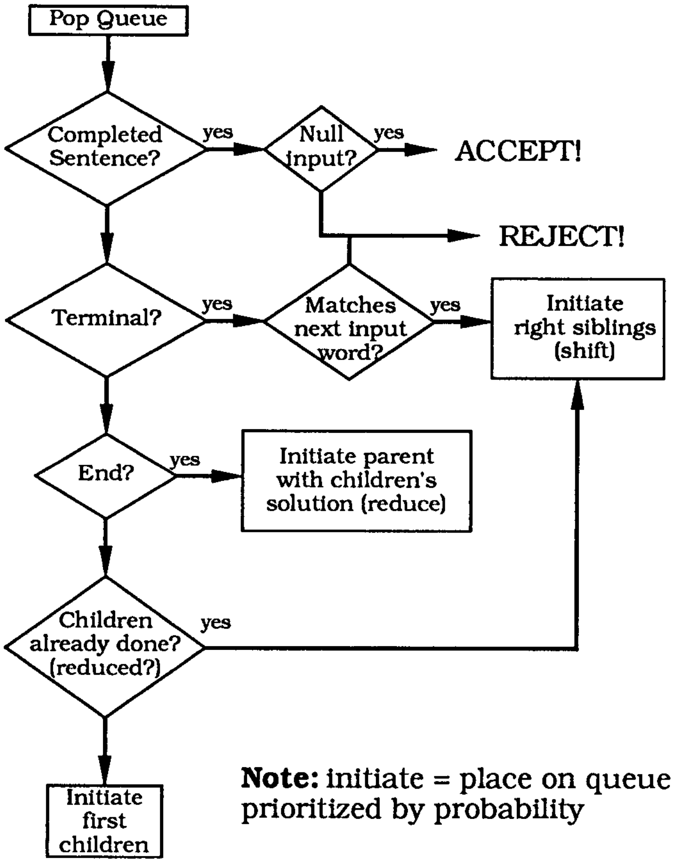
Pop Queue
Completed Sentence?
yes
Null Input?
yes
ACCEPT!
REJECT!
Terminal?
yes
Matches next input word?
yes
Initiate right siblings (shift)
End?
yes
Initiate parent with children's solution (reduce)
Children already done? (reduced?)
yes
Initiate first children
Note: initiate = place on queue prioritized by probability

Figure 2: Functional Block Diagram of Control Strategy.

Whenever a terminal node has successfully matched an input word, the path probability is reset to 1.0. Thus the probabilities that are used to prioritize the queue represent not the *total* path probability but rather the probability *given* the partial word sequence. Each path climbs up from a terminal node and back down to a next terminal node, with each new node adjusting the path probability by multiplying by a new conditional probability. The resulting conditional path probability for a next word represents the probability of that word in its syntactic role given all preceding words in *their* syntactic roles. With this strategy, a partial sentence does not become increasingly improbable as more and more words are added.[1].

NATURAL LANGUAGE ISSUES

This section describes how TINA handles agreement constraints and long distance movement, issues that are usually considered to be part of the task of a syntactic parser. Movement concerns a phenomenon of displacing a unit from its natural position in a phrase, usually to a preceding position. Such "gaps" occur commonly, for instance, in questions and passive voice, as in "(Which article)$_i$ do you think I should read (t_i)?".[2] TINA is particulary effective in handling gaps. Complex cases of nested or chained gaps are handled correctly, and most ill-formed gaps are rejected. The mechanism resembles the "hold" register idea of ATN's [1] and the treatment of bounded domination metavariables in LFG's ([2], p. 235 ff), but seems to be more straightforward than both of these.

Each parse node comes equipped with a number of slots for holding information that is relevant to the parse. Included are person and number, verb-form (*root*, *finite*, etc.) and two special slots, the current-focus and the float-object, that are concerned with long-distance movement. This information is passed along from node to node: from parent to child, child to parent, and left-sibling to right-sibling. Certain nodes have the power to adjust the values of these features. The adjustment may take the form of an unconditional override, or it may be a constraint that must have a non-null union with the value for that feature passed to the node from its relative, as will become clear in the next section.

VERB-FORM AND AGREEMENT

Certain nodes have special powers to set the verb-form either for their children or for their right-siblings. Thus, for example, HAVE as an auxilliary verb sets verb-form to *past-participle* for its *right-siblings*. The category GERUND sets the verb-form to *present-participle* for its *children*. Whenever a PREDICATE node is invoked, the verb-form has always been set by a predecessor.

Certain nodes specify person/number restrictions which then propagate up to higher levels and back down to later terminal nodes. Thus, for example, A NOUN-PL node sets the number to [*PL*], but only if the left sibling passes to it a description for number that includes [*PL*] as a possibility (otherwise it dies, as in "each boats"). This value then propagates up to the SUBJECT node, across to the PREDICATE node, and down to the verb, which then must agree with [*PL*], unless its verb-form is marked as non-finite. A more complex example is a compound noun phrase, as in "Both John and Mary have decided to go." Here, each individual noun is singular, but the subject expects a plural verb (*have* rather than *has*). TINA deals with this by making use of a node category AND-NOUN-PHRASE, which sets the number constraint to [*PL*] *for its parents*, and blocks the transfer of number information *to its children*. The OBJECT node blocks the transfer of any predecessor person/number information to its children, reflecting the fact that verbs agree in person/number with their subject but not their object.

GAPS

The mechanism to deal with gaps involves four special types of grammar nodes, identified as *generators*, *activators*, *blockers*, and *absorbers*. *Generators* are parse nodes whose grammatical category allows them to fill the current-focus slot with the subparse returned to them by their children. The current-focus is passed on to right-siblings and their descendents, but not to parents, and thus effectively reaches nodes that

[1] Some modification of this scheme will be necessary when the input stream is not deterministic. See [4] for a discussion of these very important issues regarding scoring in a best-first search.

[2] *Which article*, the object of the verb "read," has been pulled out of place and moved to the front of the question.

are *c-commanded* by the generator and its descendents [5,6]. *Activators* are nodes that move the current-focus into the float-object slot. They also require that the float-object be absorbed somewhere among their descendants. *Blockers* (such as SUBJECT) are nodes that block the transmission of the float-object to their children. Finally, *absorbers* are allowed to use the float-object as their subparse.

A simple example will help explain how this works. For the sentence "(How many pies)$_i$ did Mike buy (t$_i$)?" as illustrated by the parse tree in Figure 3, the Q-SUBJECT "how many pies" is a generator, so it fills the current-focus with its subparse. The DO-QUESTION is an activator; it moves the current-focus into the float-object position. Finally, the object of "buy," an absorber, takes the Q-SUBJECT, as its subparse. The DO-QUESTION refuses to accept any solutions from its children if the float-object has not been absorbed. Thus, the sentence "How many pies did Mike buy the pies?" would be rejected. Furthermore, the same DO-QUESTION node deals with the sentence "Did Mike buy the pies?," except in this case there is no current-focus and hence no gap.

More complicated sentences involving nested or chained traces, are handled staightforwardly by this scheme. For instance, the sentence "(Which hospital)$_j$ was (Jane)$_i$ taken (t$_i$) to (t$_j$)?" can be parsed correctly by TINA, identifying "Jane" as the object of "taken" and "which hospital" as the object of "to." This works because the VERB-PHRASE-P-O, an activator, writes over the float-object "Which hospital" with the new entry "Jane," but only for its children. The original float-object is still available to fill the OBJECT slot in the following prepositional phrase.

The example used to illustrate the power of ATN's [1], "John was believed to have been shot," also parses correctly, because the OBJECT node following the verb "believed" acts as both an absorber and a (re)generator. Cases of crossed traces, which are blocked by the Strict Cycle Condition and the Subjacency Condition in the Government/Binding rule system [7] are automatically blocked here because the second current-focus gets moved into the float-object position at the time of the second activator, overriding the preexisting float-object set up by the earlier activator. The wrong float-object is available at the position of the first trace, and the parse dies, as in the following agrammatical sentence:

*(Which books)$_i$ did you ask John (where)$_j$ Bill bought (t$_i$) (t$_j$)?

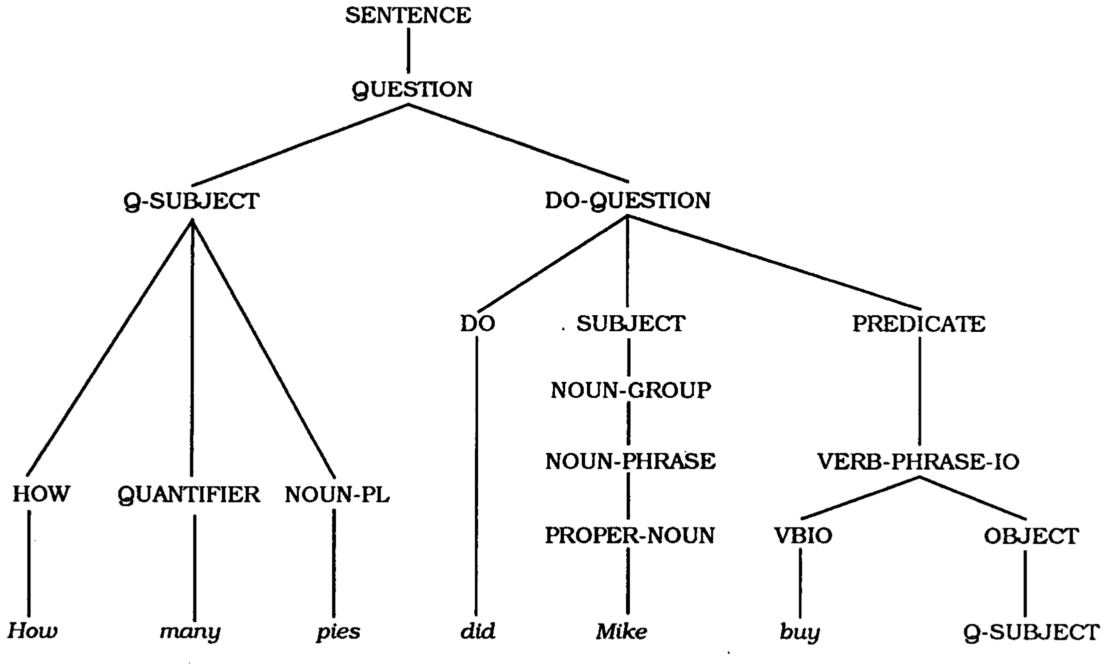

Figure 3: Example of a Parse Tree Illustrating a Gap.

The current-focus slot is not restricted to nodes that represent nouns. Some of the generators are adverbial or adjectival parts-of-speech (POS). An absorber checks for agreement in POS before it can accept the float-object as its subparse. As an example, the question, "(How oily)$_i$ do you like your salad dressing (t_i)?" contains a Q-SUBJECT "how oily" that is an adjective. The absorber PRED-ADJECTIVE accepts the available float-object as its subparse, but only after confirming that POS is adjective, as shown in the parse tree in Figure 4.

The current-focus has a number of other uses besides its role in movement. It plays an important part in identifying the subject of verbs and in establishing the references for pronouns. For a complete description of these and other topics, the interested reader is referred to [4].

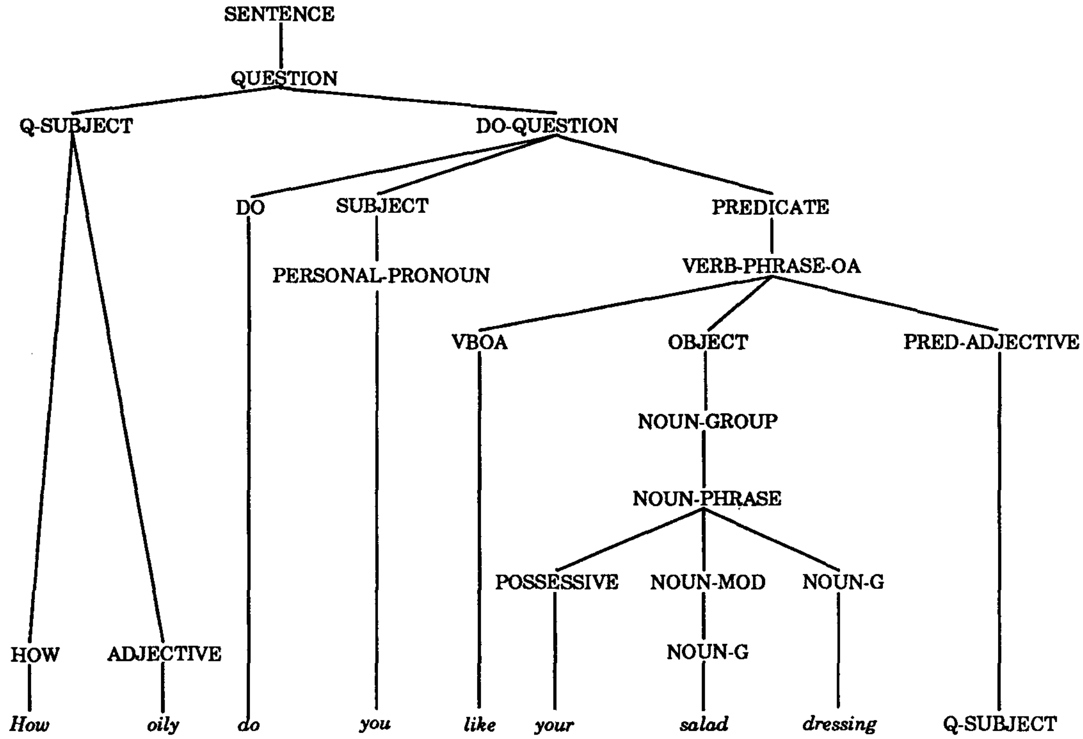

Figure 4: Parse Tree for the sentence, "How oily do you like your salad dressing?"

EVALUATION MEASURES

This section addresses several distinct performance measures for a grammar, including coverage, overgeneration, portability, perplexity and trainability. Coverage/overgeneration are concerned with the degree to which the grammar is able to capture appropriate generalities while rejecting ill-formed sentences. Perplexity, roughly defined as the geometric mean of the number of alternative word hypotheses that may follow each word in the sentence, is of particular concern in spoken language tasks. Portability and trainability concern the ease with which an existing grammar can be ported to a new task, as well as the amount of training data necessary before the grammar is able to generalize well to unseen data.

To address these issues, we used two sets of sentences. The first set is the 450 *compact acoustic-phonetic* sentences of the TIMIT database [8]. These sentences represent a fairly complex syntax, including questions, passive voice, compound and complex sentences, relative clauses, subjunctive form, comparatives, etc. They represent a semantically unrestricted space, which makes it hard to use them for tests of constraint reduction due to semantic filtering. The second set of sentences has become popular in the DARPA speech research community for both speech recognition and natural language processing applications. They concern a naval Resource Management (RM) task, and are fairly restrictive semantically. A particular subset of 791 designated training sentences and 200 designated test sentences from this task have been selected by researchers at Bolt Beranek and Newman, Inc. for studies in natural language. We have used these two sets for testing portability, perplexity and coverage.

One of the unique aspects of TINA is that a grammar can be acquired automatically from a set of parsed sentences. A typical procedure is to gradually build up the rule system by parsing new sentences one by one, introducing new arcs as needed. Once a full set of sentences has been parsed in this fashion, the parse trees from the sentences are automatically converted to the set of rules used to generate each sentence. The training of both the rule set and the probability assignments is established directly from the provided set of parsed sentences; i.e. the parsed sentences *are* the grammar.

We took advantage of this feature to test the system's capability of generalizing to unseen data from a small set of sentence examples. Since there were only 450 sentences in the TIMIT task, we were unwilling to set aside a portion of these as designated test sentences. Instead, we built a grammar that could parse all of the sentences, and then generated a subset grammar from 449 of the sentences, testing this grammar for coverage on the remaining one. We cycled through all 450 sentences in this fashion.

Our experiments were conducted as follows. We first built a grammar from the 450 TIMIT sentences. We tested coverage on these sentences using the above jackknifing strategy. We assessed overgeneration by generating sentences at random from the TIMIT grammar, checking whether these sentences were well-formed syntactically. We then tested portability by beginning with the grammar established from the TIMIT task and then deriving a grammar for the 791 designated training sentences of the RM task. Within this task it was possible to make use of semantic categories, particularly within noun phrases, in order to reduce perplexity. We measured both perplexity and coverage on the remaining 200 test sentences of this task, using a grammar built automatically from the 791 parsed training sentences.

COVERAGE WITHIN TIMIT

The result of the jackknifing experiment was that 75% of the unseen sentences were successfully parsed based on structures seen in the remaining 449 sentences. In most cases where the system failed, a single unique form occurred somewhere in the unseen sentence that had not appeared in any of the other sentences, as illustrated in Table 1. We do not mean to suggest that a grammar should not be able to handle such forms; however, we are encouraged that three quarters of the sentences could parse based on such a small amount of training data. It suggests that the system can learn generalizations fairly quickly.

Table 1: Some examples of sentences that failed in the jackknife experiment.

as-as construction:	Withdraw only *as* much money *as* you need.
rather x than y:	I'd *rather* not buy these shoes *than* be overcharged.
why predicate:	*Why* buy oil when you always use mine?
triple verb:	*Laugh, dance, and sing,* if fortune smiles on you.
"*are both*":	The patient and the surgeon *are both* recuperating from the lengthy operation.
adjunct as subject:	*Right now* may not be the best time for business mergers.

An example of a sentence that succeeded is given in Table 2, along with a list of sentences that could be used to build the portions of the parse tree necessary for the successful parse. The parse of this sentence is given in Figure 4. Because of the sharing of structures among rules with the same LHS, the system is capable of synthesizing new rules from pieces of other rules, which allows it to learn more quickly.

Table 2: Example of generalization capability of parser.

New Parsable Sentence:
"How oily do you like your salad dresssing?"

Useful sentences in training set:

Q-subject as Adjective:	"*How permanent* are their records?"
Q-subject Do-question:	"*How do oysters make pearls*?"
Verb Object Predicate-adjective:	"Calcium *makes bones and teeth strong*."
Possessive Noun-mod Noun:	Gwen planted green beans in *her vegetable garden*.

OVERGENERATION

The issue of overgeneration is extremely important for spoken language tasks, because of the need to keep perplexity as low as possible. TINA can be run in generation mode, where, instead of proposing all alternatives at a decision point, a random number generator is used to select a particular decision. Generation mode is an extremely useful tool for discovering errors in the grammar. Randomly generated sentences are typically syntactically correct but semantically anomalous. Occasionally an ill-formed sentence is generated, due to inappropriate generalities in the rules. Usually the situation can be corrected through rule modification.

Since all of the arcs have assigned probabilities, the parse tree is traversed by generating a random number at each node and deciding which arc to take based on the outcome, using the arc probabilities to weight the alternatives. Some examples of sentences generated in this way are given in Table 3. While many of these sentences are clearly nonsense, due to the complete absence of semantic constraint, they all appear to be syntactically representative of the language. It is clear that proper semantic constraint would greatly decrease the perplexity, although, given the rich semantic base of the 450 sentences, it is not reasonable to expect to build a suitable semantic component for them. Applying semantic constraint appropriate in a sublanguage that a natural language system can interpret, however, is a much more feasible undertaking.

Table 3: Some sample sentences generated by the parser.

Wash, of course, but puree the high hats and article colleges straight ahead.
How did the income of gold open the algebraic hit?
A child of execution stole the previous attitude.
Which tunafish would lots of the muscles smash?
Make a scholastic marriage; then only get ski fangs under a medical film.
Whenever a simple perfume must diminish near stew, enter.
It is fun to pledge to break down.
Hyenas might be used to eat every coach.
The screen is surely blistered occasionally.

PORTABILITY

We tested ease of portability for TINA by beginning with a grammar bulit from the 450 TIMIT sentences and then deriving a grammar for the RM task. These two tasks represent very different sentence types. For instance, the overwhelming majority of the TIMIT sentences are statements, whereas there are no statements in the RM task, which is made up exclusively of questions and requests. The process of conversion to a new grammar involves parsing the new sentences one by one, and adding context-free rules whenever a parse fails. The person entering the rules must be very familiar with the grammar structure, but for the most part it is straightforward to identify and incrementally add missing arcs. The parser identifies where in the sentence it fails, and also maintains a record of the successful partial parses. These pieces of information usually are adequate to pinpoint the missing arcs. It required less than one person-month to convert the grammar from TIMIT to the RM task.

PERPLEXITY AND COVERAGE WITHIN RM TASK

We built a subset grammar from the 791 parsed RM training sentences, and then used this grammar to test coverage and perplexity on the unseen 200 test sentences. The grammar could parse all of the training sentences and 78.5% of the test sentences. We are unwilling to examine the test sentences, as we may be using them for further evaluations in the future. Therefore, we cannot yet assess why a particular sentence failed, or whether the parse found by the grammar was actually the correct parse.

A formula for the test set perplexity is [9]:

$$Perplexity = 2^{-\frac{1}{N}\sum_{i=1}^{N} log_2 P(w_i|w_{i-1},...w_1)}$$

where the w_i are the sequence of all words in all sentences, N is the total number of words, including an "end" word after each sentence, and $P(w_i|w_{i-1},...w_1)$ is the probability of the ith word given all preceding words.[3] If all words are assumed equally likely, then $P(w_i|w_{i-1},...w_1)$ can be determined by counting all the words that could follow each word in the sentence, along all workable partial theories. If the grammar contains probability estimates, then these can be used in place of the equally-likely assumption. If the grammar's estimates reflect reality, the estimated probabilities will result in a reduction in the total perplexity.

An average perplexity for the 157 test sentences that were parsable was computed for the two conditions, without (Case 1) and with (Case 2) the estimated probabilities. The result was a perplexity of 374 for Case 1, but only 41.7 for Case 2. This is with a total vocabulary size of 985 words, and with a grammar that included several semantically restricted classes such as SHIP-NAME and READINESS-CATEGORY. The incorporation of arc probabilities reduced the perplexity by a factor of nine, a clear indicator that a proper mechanism for utilizing probabilities in a grammar can help significantly.

DISCUSSION

This paper describes a new grammar formalism that addresses issues of concern in building a fully integrated speech understanding system. The grammar includes arc probabilities reflecting the frequency of occurrence of the syntactic structures within the language. These probabilities are used to control the order in which hypotheses are considered, and are trained automatically from a set of parsed sentences, which makes it straightforward to tailor the grammar to a particular need. Ultimately, one could imagine the existence of a very large grammar that could parse almost anything, which would be subsetted for a particular task by simply providing it with a set of example sentences within that task.

The issue of semantic analysis has not yet been dealt with properly within TINA. We were able to achieve significant perplexity reduction by defining some semantically restricted classes, particularly within noun phrases, but this is probably not the appropriate way to represent semantic constraint. I would rather see semantic filters introduced through co-occurrence data on syntactic pairs like Adjective-Noun and Subject-Verb, adopting methods similar to those proposed in [10,11].

[3] in the case of TINA, all words up to the current word within each sentence are relevant.

At the time a set of word candidates is proposed to the acoustic matcher of a recognizer, all of the constraint available from the restrictive influence of syntax, semantics, and phonology should have already been applied. The syntactic parse tree of TINA can be used to express other constraints ranging from acoustic-phonetic to semantic and pragmatic. The syntactic node would contain slots for various kinds of constraint information – syntactic filters such as person/number and verb-form, semantic filters such as the permissible semantic categories for the subject/object of the hypothesized verb, and acoustic-phonetic filters (for instance, restricting the word to begin with a vowel if the preceding word ended in a flap). As the parse tree advances, it accumulates additional constraint filters that further restrict the number of possible next-word candidates. Thus the task of the predictive component is formulated as follows: given a sequence of words that has been interpreted to the fullest capability of the syntactic/semantic/phonological components, what are the likely words to follow, and what are their associated a priori probabilities?

While TINA's terminal nodes are lexical words, I believe that the nodes should continue down below the word level. Prefixes and suffixes alter the meaning/part-of-speech in predictable ways, and therefore should be represented as separate subword grammar units that can take certain specified actions. Below this level would be syllabic units, whose children are subsyllabic units such as onset and rhyme, finally terminating in phoneme-like atomic constituents. Acoustic evidence would enter at several stages. Important acoustic matches would take place at the terminal constituents, but duration and intonation patterns would contribute to scores in nodes at many higher levels of the hierarchy.

One issue that has not been addressed here but which I feel is very important is the notion of a *fully* automatic training capability. TINA trains automatically from a set of parsable sentences, but when a sentence fails to parse there is no recourse except human intervention to provide new rules. One possibility is to generate some new rules automatically by applying meta-rules similar to those used in the Unification framework, as in the generation of passive voice forms for verbs. Berwick's research in the area of automatic language acquisition [12] represents an important effort along these lines as well, and it is likely that some of his ideas could be extended to apply in TINA's framework.

We plan to integrate TINA with the SUMMIT speech recognition system [13] shortly. Two important issues are 1) how to combine the scores for the recognition component and the predictive component of the grammar, and 2) how to take advantage of appropriate pruning strategies to prevent an explosive search problem. In the case of the first problem, we would need to modify the mechanism for using path probabilities in TINA, in order to deal with the nondeterministic nature of the acoustic evidence. With regard to the second problem, each parse node in the tree can prune all but the best-scoring equivalent path by maintaining a record of all solutions returned to it by its children, and checking against this list whenever a new solution comes in. With this chart in place, there would not be a duplication of effort for paths beyond that point.

ACKNOWLEDGEMENTS

This research has benefitted from interactions with Lynette Hirschman, Mark Johnston, and Victor Zue. Christine Pao implemented a preliminary version of this parser, and Jim Glass developed the user interface. I am also grateful to Jim Glass, Mike Phillips, and Victor Zue for critically reading an early draft and offering perceptive comments.

References

[1] Woods, W.A., "Transition Network Grammars for Natural Language Analysis," Commun. of the ACM 13, 591-606, 1970.

[2] Bresnan, J., ed., *The Mental Representation of Grammatical Relations*, MIT Press, 1982.

[3] Tomita, M., *Efficient Parsing for Natural Language,* Kluwer Academic Publishers, Boston, MA, 1986.

[4] Seneff, S., "TINA: A Probabilistic Syntactic Parser for Speech Understanding Systems," LCS Technical Report, Laboratory for Computer Science, MIT, Cambridge, MA, forthcoming.

[5] Chomsky, Noam, *Lectures on Government and Binding*, Second Revised Edition, Foris Publications, Dordrecht, the Netherlands, 1982.

[6] Reinhart, T., *Anaphora and Semantic Interpretation*," The University of Chicago Press, Chicago, Il, 1983.

[7] Chomsky, Noam, "On wh-movement" in P. Culicover, T. Wasow, and A. Akmajian, eds. *Formal Syntax* Academic Press, New York, 1977.

[8] Lamel, L., R.H. Kassel, and S. Seneff, "Speech Database Development: Design and Analysis of the Acoustic-Phonetic Corpus," DARPA Speech Recognition Workshop Proceedings, Palo Alto, CA, Feb 19-20, 1986.

[9] Lee, K.F., *Automatic Speech Recognition: The Developemnt of the Sphinx System,* Appendix I, Kluwer Academic Publishers, Boston, 1989.

[10] Grishman, R., L.Hirschman, and NT. Nhan, "Discovery Procedures for Sublanguage Selectional Patterns: Initial Experiments," Computational Linguistics, Vol. 12, No. 3, pp. 205-215, 1986.

[11] Hirschman, L., R. Grishman and N. Sager, "Grammatically-based Automatic Word Class Formation," Information Processing and Management, Vol. 11, pp. 39-57, 1975

[12] Berwick, R., *The Acquisition of Syntactic Knowledge,* The MIT Press, Cambridge, MA 1985.

[13] Zue, V., J. Glass, M. Phillips, and S. Seneff, "The MIT SUMMIT Speech Recognition System: A Progress Report," *These Proceedings,* Feb., 1989.