Integrating Language Models with Speech Recognition

M. P. Harper, L. H. Jamieson, C. D. Mitchell, G. Ying, S. Potisuk,
P. N. Srinivasan, R. Chen, C. B. Zoltowski, L. L. McPheters, B. Pellom, and R. A. Helzerman
1285 The Electrical Engineering Building
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907-1285
{harper,lhj,cdm,sgsying,pong,pramila,ruxin,bauercar,mcpheter,pellom,helz}@ecn.purdue.edu

Abstract

The question of how to integrate language models with speech recognition systems is becoming more important as speech recognition technology matures. For the purposes of this paper, we have classified the level of integration of current and past approaches into three categories: tightly-coupled, loosely-coupled, or semi-coupled systems. We then argue that loose coupling is more appropriate given the current state of the art and given that it allows one to measure more precisely which components of the language model are most important. We will detail how the speech component in our approach interacts with the language model and discuss why we chose our language model.

1 Introduction

State of the art speech recognition systems achieve high recognition accuracies only on tasks that have low perplexities. The perplexity of a task is, roughly speaking, the average number of choices at any decision point. The perplexity of a task is at a minimum when the true language model is known and correctly modeled. A poor language model increases perplexity and lowers performance. To achieve higher recognition accuracy for a given perplexity, it is necessary to improve the acoustic model or utilize more high-level knowledge, such as prosody or semantics. Knowledge sources commonly used in speech understanding are shown in Figure 1. There is good evidence that there is an implicit ordering among these knowledge sources such that one type of information must be available before it makes sense to progress to the next level[1].

Although determining the true language model is often very difficult or impossible, approximate models can often be found which have sufficiently low perplexity so that accurate recognition is feasible. A second, potentially more difficult problem is how to integrate an acoustic model with a language model which includes syntactic, semantic, and pragmatic knowledge sources.

The question of how to integrate language models with speech recognition systems is becoming more important as speech recognition technology matures. The level of integration for current and past approaches can be classified into one of three categories: tightly-coupled, loosely-coupled, or semi-coupled systems. A *tightly-coupled system* is one which integrates all of the knowledge sources for speech into a highly interdependent set of processes which cannot be separated. If we apply software engineering principles to tightly-coupled systems, we observe the following:

1. The language and acoustic models are not separable.
2. It is difficult to evaluate the impact of each of the knowledge sources.
3. For complex domains, the systems tend to be intractable. For example, tightly integrating acoustic/phonetic processing with syntactic processing

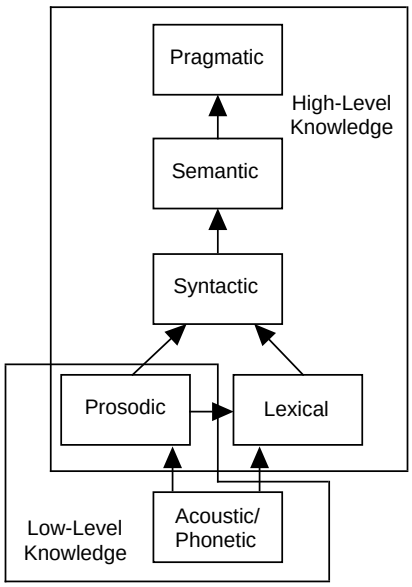

Figure 1: The order of precedence between spoken language knowledge sources.

[1] Prosody can help a word recognizer to rule out word candidates with unlikely stress and duration patterns, but it can also impact syntactic and semantic modules. Therefore, we depict the prosody module as both a high-level and low-level knowledge source.

can yield a system which is orders of magnitude slower than real time.
4. It is difficult to scale up tightly-coupled systems to realistic tasks because the integration of knowledge sources makes the system much larger and more difficult to understand.

A *loosely-coupled system* is one that isolates the knowledge sources into relatively independent modules which communicate with each other. Again applying software engineering principles, we observe the following properties of loosely-coupled systems:

1. They require the system designer to determine the best way for the modules to communicate. This has proven to be a difficult problem [6, 13, 23].
2. They use level-appropriate information, which should make them more tractable. This avoids the combinatorial explosion caused by making acoustic decisions in the syntactic module, for example.
3. Since the knowledge sources are independent, they should scale up to larger problems better than tightly-coupled systems.
4. It is easier to measure the impact of each of the knowledge sources (which is important given our current level of understanding).
5. The modularity should make them easier to understand, design, and debug. The individual modules can also be tested in a stand-alone fashion.
6. They can easily accommodate more than one task or language by replacing the individual modules.

Semi-coupled systems fall in between the previous two in that a knowledge source can be used to guide a lower level search in the system. In this capacity, the removal of the knowledge source from the system impacts the lower level search, and so the semi-coupled module is not completely independent. Semi-coupled systems tend to be intractable when they combine levels of information from the low-level and high-level categories. Table 1 summarizes this discussion.

In this paper, we argue that loose coupling is more appropriate given the current state of the art and given that it allows one to measure more precisely which components of the language model are the most important. We will begin with a review of the past literature and then detail our approach. We propose a loosely-coupled system which uses a uniform approach to integrate the high-level knowledge sources.

2 Review of Related Work

In a tightly-coupled system, the language and acoustic models are applied simultaneously. CSELT's system, based on finite-state language models [7], falls in this category[2]. Although easily incorporated into a hidden Markov model (HMM), a finite state grammar does not sufficiently constrain the utterances in a spoken language [30]. There are two ways to circumvent this shortcoming and still maintain tight coupling. The first is to modify the HMM to incorporate more powerful language models. For example, the inside-outside algorithm [21, 22] is an extension to HMMs which allows recursive embedding. Whereas a standard HMM can handle only regular grammars, the inside-outside algorithm can process a context-free grammar. The second is to utilize acoustic information in a syntactic processor. For instance, a CYK parser can be modified for acoustic input [28] by exhaustively finding all possible endpoints for every terminal. The modified CYK parser could be thought of as an extension to dynamic time warping speech recognition in much the same way that the inside-outside algorithm could be thought of as an extension to HMM speech recognition.

Although theoretically appealing, the modified CYK parser and the inside-outside algorithm are impractical due to their huge computational costs. Both are $O(n^3)$, where n is the number of input symbols. If these symbols are acoustic measurements (which are typically taken once every 10 ms), then the system is intractable. Lari and Young [22] require 64 transputers for very small problems. The inside-outside algorithm requires a large training set in order to train both acoustic and language probabilities.

Tightly-coupled systems are hampered by their degree of integration. The best acoustic models do not allow a detailed language model, and the best language models are not well suited for the low-level probabilistic pattern matching needed to accurately classify the acoustic patterns. Systems that work adequately for both acoustic and language processing are often intractable for all but simple examples.

A semi-coupled approach combines a language model with an acoustic model in such a way that they cannot be separated procedurally, even though some components can be removed from the system. In a top-down system, for example, the language model is invoked first at a particular decision point, and then the acoustic model is used to select the best of all candidates that are allowed by the language model. In [9, 17, 18, 19], Kita, Kawabata, and Saito use an LR parser to predict phones, which are then verified by a phone HMM. The phones that make up a word are specified by rules in the grammar. They use a stack splitting method to cope with ambiguity. The acoustic and language components are not entirely separable since the acoustic model receives its focus from the language model.

By far the most successful approach to integrating a language model with an acoustic model has been to embed an N-gram language model into an HMM [2, 29, 31]. The N-gram model assumes that the probability of the current word is a function of the previous N-1 words. This model can easily be integrated because of its simplicity and reduces the perplexity significantly compared to an HMM without a grammar. However, the approach has several disadvantages. Even for small N (i.e., 2 or 3), millions of words of text are required to estimate the N-grams for moderate to large vocabularies. Even so, many of the N-grams are undertrained and extensive smoothing is required. Another disadvantage of N-grams is their task depen-

[2] Their finite-state grammar model is applied as a postprocessor in their paper; however, their stated goal is to incorporate it into the HMM.

	Tightly-coupled	Semi-coupled	Loosely-coupled
Separability	One integrated model for all KS's	Some KS's can be removed but not isolated	Each KS is modeled by a stand-alone module
Inter-module Communication	NA	Typically one way communication	Designer specifies interaction
Easy to scale	No	No	Yes
Computation	Usually intractable for all but very small examples	Tractable for small problems or simple language models	May be tractable for large problems
Examples	Acoustic-level CYK parser, Inside-outside algorithm	LR parser with HMM phone verification, N-grams	Constraint-based system, Blackboard model as in Hearsay II

Table 1: Characteristics of spoken language systems.

dence. Also, they do not provide a parse or a semantic representation for a sentence, both of which are useful for speech understanding. We classify this method as a semi-coupled approach since the N-gram model is typically used to guide lower level search and is not a post-processor[3].

In a bottom-up system, the acoustical scores are found first, and then the language model is applied to reduce the number of acoustic candidates. This duo of acoustical modeling followed by language modeling can be done at each decision point, or just once, where all acoustical information is extracted prior to utilizing any part of the language model. In the first case, the two modules are semi-coupled; whereas in the second case, the language model is invoked as a postprocessor and is loosely-coupled. For a loosely-coupled bottom-up system to work correctly, all relevant acoustic information must be preserved by the acoustic processor. To be tractable, most superfluous information must be discarded.

Several modern systems utilize the language model as a postprocessing step, and so these systems are loosely coupled. In Bates [3], the authors first find the N-best [34, 35, 36] sentences with an HMM, and then apply syntactic and semantic rules using a chart parser. CMU's Phoenix uses frame based parsing and semantic phrase grammars [39] on single sentences. Individually processing each sentence hypothesis provided by a speech recognizer simplifies the task of the language model, but is inefficient since many sentence hypotheses can be generated with a high degree of similarity. MIT's Voyager uses LR parsing [43] as a postprocessing step with N-best input[4]. N-best input is simple to process, but at the cost of much repeated work. Seneff's robust parser [37, 38] operates on the most likely sentences and is a postprocessor for the speech recognizer.

The most dramatic example of the loosely-coupled systems is the blackboard model employed by Hearsay-II [6, 13, 23]. The blackboard model represents each knowledge source as an independent process which is gathering useful information from and dispensing new information to the blackboard. The blackboard consists of a uniform multi-level network, permitting generation and linkage between alternative hypotheses at all levels. Despite the complete modularity of this approach, it has not been as successful as current approaches which use bigram or trigram models. This is partially because acoustic processing has improved with the advent of new techniques. One reason this approach has not come back into favor may be that the blackboard approach is too loosely coupled. When a system is divided into many independent, cooperating processes (as in parallel processing), it is often more difficult to understand, coordinate, and debug the complex interactions among modules.

3 Our Approach

In this section, we describe how the speech component in our system interacts with the language model and discuss why we chose our language model. Figure 2 shows the modules in our speech system.

We have divided our system into three loosely-coupled modules. The first module corresponds to the low-level knowledge sources. The acoustic processing component consists of an HMM which selects word candidates and produces a word graph. The prosody processing component interacts with the word-level system to rule out word candidates with unlikely stress and duration patterns before they are placed in the word network. Additionally, the prosodic module annotates the word graph with information that can be used in conjunction with syntax and semantics to further prune the word graph in the second module.

A word graph is a directed acyclic graph which provides a very compact and expressive way to capture all possible sentence hypotheses given the ambiguity inherent in the task of recognizing the words in a continuously spoken sentence. Our goal for the speech recognizer is to produce a word graph with as few word nodes as possible without eliminating the target sentence hypothesis from the word graph. This is similar to passing a list of N-best sentences to the language

[3]For N = 2, however, it may be possible to incorporate the N-gram directly into the HMM topology. In this case, the acoustic and language models are tightly coupled.

[4]They also use their parser in a generative mode to create sample correct sentences to help estimate a word pair grammar. The word pair grammar is embedded in the HMM.

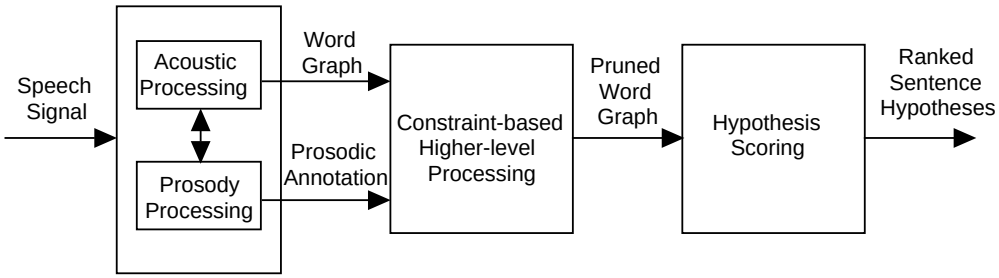

Figure 2: The modules in our loosely-coupled speech system.

model. However, an N-best list of sentence hypotheses limits the information passed between the acoustic model and the language model. In contrast, a word graph of word candidates is typically more compact and more expressive than a list of the most likely sentences. To compare N-best and word graph representations [10, 12, 42], we have constructed word graphs from sets of sentence hypotheses. The word graphs provided an 83% reduction in storage, and in all cases they encoded more possible sentence hypotheses than were in the original list of hypotheses. In some cases, the target sentence did not appear in the N-best list but did appear in the word graph. Prosodic information can also be stored in the word graph for higher-level processing.

Clearly, pruning the word graph is important, but in some cases higher level knowledge is more accurate at pruning the word graph. The pruning that is done by the recognizer can be done based on extremely low acoustic scores (i.e., a poor match to a word candidate), a very simple embedded language model (e.g., N-gram), and word-level prosodic information. The more pruning that can be done before or during word graph construction, the less work the language model has to do. Since the language model typically uses some sort of parsing algorithm, the running time for the algorithm will be at least $O(n^3)$, where n is the number of word nodes in the graph.

The second module in our system consists of a constraint-based processing component [10, 12, 42] which is based on an extension to Constraint Dependency Grammar (CDG) parsing as defined by Maruyama [24, 25, 26]. This component employs constraint propagation to prune word graphs. This system is capable of propagating a wide variety of constraints, including syntactic, lexical, semantic, prosodic, and pragmatic constraints. The central data structure for this module is a word graph augmented with parse related information, called a spoken language constraint network (SLCN). An SLCN represents all possible parses for the represented sentence hypotheses in a compact form, and is operated on by constraints.

The third module in our system represents our method for merging the influence of the previous two modules in order to select the best sentence hypothesis. By annotating the word graph with likelihood information from the first module and then pruning it with constraints representing higher level knowledge from the second module, we are using the appropriate information from both to select the best sentence

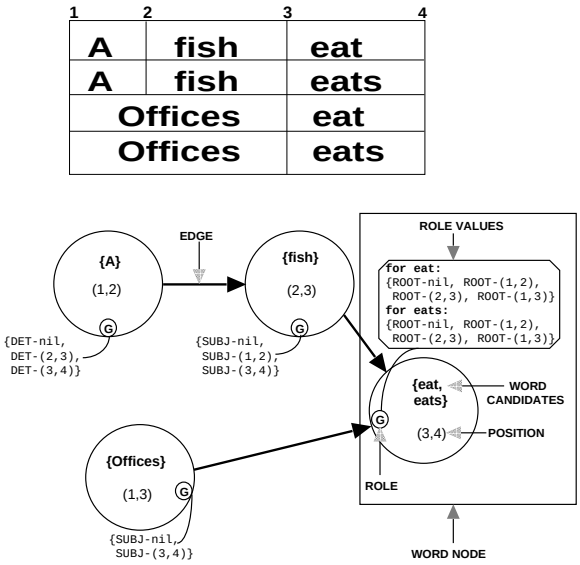

Figure 3: An SLCN with four sentence hypotheses.

candidates.

3.1 Properties of SLCNs

We have extended CDG parsing to process a word graph rather than single sentences, calling the new constraint network an SLCN. This approach allows us to efficiently process multiple sentence hypotheses while maintaining the flexibility of the CDG parsing method. Figure 3 depicts an example of an SLCN constructed from a four-best list of sentences. *Word nodes* contain information on the *position* of a word's utterance (represented as an integer tuple (b, e), where b and e represent the time at which the word begins and ends, respectively), a set of *roles* (which indicate the various functions the word fills in a sentence), and a list of all *word candidates* with the same *position*. Also, *edges* join word nodes that can be adjacent in a sentence hypothesis. Though two roles are required to write a grammar that is at least as expressive as a CFG [25], our example depicts a single role to simplify the discussion. The depicted role is the *governor* role, which represents the function that a word fills given that it modifies its head. For example, the word *the* has the function of a determiner (i.e., DET) when it modifies the head noun.

To parse an SLCN, each role for every word is initially assigned all *role values* allowed by each of the word candidates for each of its parts of speech. A role value consists of a label (e.g., DET) and the position of

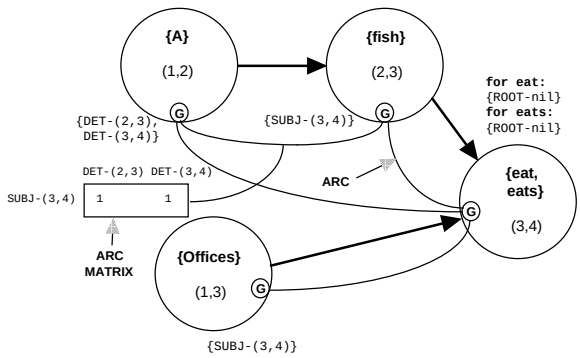

Figure 4: An SLCN prior to binary constraints.

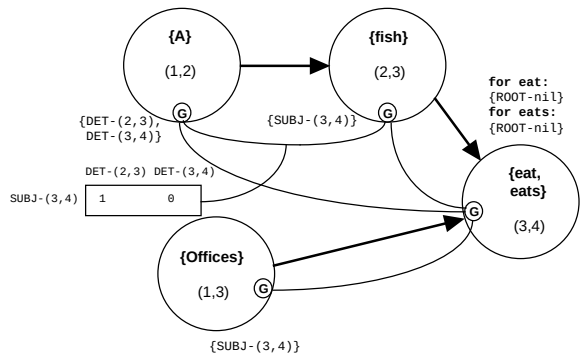

Figure 5: An SLCN after binary constraints.

the word it modifies. Figure 3 shows the initialization of the SLCN, when *a* is a determiner with the label DET, *fish* and *offices* are nouns with the label SUBJ, and *eat* and *eats* are verbs with the label ROOT.

Once the role values are enumerated, constraints are applied to eliminate the ungrammatical role values. A constraint is an *if-then* rule which must be satisfied by the role values. First, *unary constraints* (i.e., constraints with a single variable) are applied to each role value for every role. A role value violates a constraint iff it causes the antecedent of the constraint to evaluate to TRUE and the consequent to evaluate to FALSE. A role value which violates a unary constraint is eliminated. For example, DET-nil would be eliminated from the SLCN in Figure 3 given the following constraint[5]:

```
;; A DET must modify something to its right.
(if (equal (label x) DET)
    (< (position x) (modifiee x)))
```

Figure 4 depicts the SLCN after two additional unary constraints are applied requiring ROOTs to have a nil modifiee and SUBJs to modify words after them.

Figure 4 also illustrates how the SLCN is prepared for the propagation of *binary constraints*. Binary constraints contain two variables and determine which pairs of role values can legally coexist. *Arcs* are added to the SLCN to keep track of which pairs of role values can legally coexist given the binary constraints. Roles are joined with an arc iff they can be members of at least one common sentence hypothesis. For example, there is an arc between the roles for *a* and *fish*, but there is not an arc between the roles for *a* and *offices* in Figure 4.

Each of the arcs has an associated *arc matrix*, whose row and column indices are the role values from its two roles. Initially, all entries in the matrices are set to **1**. Only one of the arc matrices is depicted in Figure 4. After binary constraints are applied, a **0** in the (i, j)th entry of the matrix associated with the arc between $node_l$ and $node_m$ indicates that the ith role value of $node_l$ is incompatible with the jth role value of $node_m$. Figure 5 shows the state of the SLCN from Figure 4 after the following constraint is propagated:

```
;; A DET must modify a SUBJ.
(if (and (equal (label x) DET)
         (equal (label y) SUBJ))
    (equal (modifiee x) (position y)))
```

Following the propagation of binary constraints, the network could contain role values that would never be legal for any sentence hypothesis in the SLCN. For example, in Figure 5, DET-(3,4) would be eliminated by a process called filtering. Filtering removes those role values that are not supported by any sentence hypothesis (see [10, 16]).

In the next section, we will compare the operation of our parser with context-free grammar parsers. We then compare the power and flexibility of the two approaches.

3.1.1 Run-time Properties

1. CFG parsers use production rules (or equivalently, recursive transition networks) to determine whether or not a string of terminals is in the language. Our constraint-based parser, on the other hand, uses a set of constraints. Any type of constraint that can be formulated as an appropriate if-then rule can be used to constrain an SLCN.
2. The best serial running time for a practical CFG parser operating with an ambiguous grammar is $O(G^2 * n^3)$ [5], where n is the number of words in a sentence and G is the size of the CFG grammar. On the other hand, the serial running time for the SLCN parser is $O(k * n^4)$, where k is the number of constraints in the grammar and n is the number of word nodes. In practice, we have found that k is comparable in size to G for grammars with the same coverage, and $G >> n$.
3. CFG parsing has been parallelized by several researchers. For example, Kosaraju's method [20] using cellular automata can parse CFGs in $O(n)$ time using $O(n^2)$ processors. However, achieving CFG parsing times of less than $O(n)$ has required more powerful and less implementable models of parallel computation than used by [20], as well as significantly more processors (e.g., [33] requires $O(n^6)$ processors to achieve logarithmic time). In contrast, we have devised a parallelization for the single sentence CDG parser [15, 14] which uses $O(n^4)$ processors to parse in $O(k)$ time for a CRCW P-RAM model (Concurrent Read,

[5] The less-than predicate, $(< (b1, e1) (b2, e2))$, returns true iff $e1 < b2$.

Concurrent Write, Parallel Random Access Machine), where n is the number of words in the sentence and k, the number of constraints, is a grammatical constant, and have simulated it on the MasPar MP-1, a massively parallel SIMD computer with 16K processing elements (achieving an $O(k + log(n))$ running time by using $O(n^4)$ processors).

4. CFG parsers often build trees for each of the possible parses for a sentence. Because the enumeration of the trees for highly ambiguous sentences can require exponential time, some CFG parsers construct a parse forest (or similar structure). The SLCN constructed by our parser is a forest of parse graphs which is pruned during parsing. It differs from a CFG parse forest in that there are no nonterminals in the graph, only links between terminals, which are assigned sets of labels.

5. The smallest practical size for a CFG parse forest is $O(G^2 * n^3)$ [5], regardless of how many parses there are for an ambiguous sentence. In contrast, an SLCN has $O(n^4)$ size[6], and there is no stack or agenda to maintain.

3.1.2 Power and Flexibility

1. The set of languages accepted by a CDG grammar is a superset of the set of languages which can be accepted by CFGs. In fact, Maruyama [24, 25] is able to construct CDG grammars with two roles (degree = 2) and two variable constraints (arity = 2) which accept the same language as an arbitrary CFG converted to Griebach normal form. We have also devised an algorithm to map a set of CFG production rules to a CDG grammar. This algorithm does not assume that the rules are in normal form, and the number of constraints created is $O(G)$. In addition, CDG can accept languages that CFGs cannot, for example, $a^n b^n c^n$ and ww (where w is some string of terminal symbols).

2. To parse a free-order language like Latin, CFGs require that additional rules containing the permutations of the right-hand side of a production be explicitly included in the grammar [27]. Unordered CFGs do not have this combinatorial explosion of rules, but the recognition problem for this class of grammars is NP-complete. A free-order language can easily be handled by a CDG parser because order between constituents is not a requirement of the grammatical formalism. Furthermore, CDG is capable of efficiently analyzing free-order languages because it is does not have to test for all possible word orders.

3. When a CFG parser generates a set of ambiguous parses for a sentence, it cannot invoke additional production rules to further prune the parses. In contrast, in CDG parsing, the presence of ambiguity can trigger the propagation of additional constraints to further refine the parses for a sentence. A core set of constraints that holds universally can be propagated first, and then additional, possibly context dependent, constraints can be used if ambiguity remains. This type of flexibility is easy to achieve, since constraints do not require precompilation into a table or network. This flexibility allows us to experiment with the best way to order constraints. It also allows us to model context by applying constraints in one case, but not another.

4. Tight coupling of prosodic [4] and semantic rules with CFG grammar rules typically increases the size and complexity of the grammar and reduces its understandability. Semantic grammars have been effective for limited domains, but they do not scale up well to larger systems [1]. The constraint-based approach represents a loosely-coupled approach for combining a variety of knowledge sources. It differs from a blackboard approach in that all constraints are applied using the uniform mechanism of constraint propagation. Hence, the designer does not need to create a set of functionally different modules and worry about their interface with the other modules. Constraint propagation is a uniform method which allows us to focus on the best way to order the sources of information impacting the word graph.

5. There are currently a number of probabilistic CFG approaches [8, 40, 41]. We believe that the constraint-based approach must be extended to allow for probabilistic constraints. However, we also believe that context matters a lot for determining the semantic and pragmatic appropriateness of an utterance, hence stochastic methods may not be appropriate at this level. Contextual information may be difficult to capture with stochastic methods because of limited training data.

We believe that our system overcomes many of the problems associated with loosely-coupled systems. Constraint propagation provides a uniform method for applying higher-level knowledge sources to prune a word graph. Because the constraints for each knowledge source can be developed independently, it is not as difficult to add another knowledge source to our parser. In an initial experiment, we constructed a set of syntactic constraints [42] for sentences in the Resource Management corpus [32]. We then determined the effectiveness of these constraints for pruning word graphs constructed from N-best lists of sentence hypotheses (supplied by BBN). The syntactic constraints were effective at pruning many ungrammatical sentence hypotheses from a word graph and limiting the possible parses for the remaining sentences. In a second experiment [12], we demonstrated the flexibility of constraint-based parsing for utilizing a variety of knowledge sources. We developed a second grammar for sentences in the ATIS corpus (Air Travel Information System), which was chosen because of its well-defined semantics. First,

[6] In natural language processing, n is typically much smaller than G. English sentences usually contain fewer than 30 words; whereas, hundreds or even thousands of production rules are not uncommon for broad coverage English grammars.

syntactic constraints were added to the Resource Management grammar to demonstrate the ease of adding additional grammar constraints to a previously developed grammar. Then semantic constraints were constructed to further limit ambiguity [12]. An example of a semantic constraint for our parsing example follows:

```
;; The SUBJ of a ROOT with a semantic feature of +action
;; must be +animate.
(if (and (equal (label x) SUBJ)
         (equal (label y) ROOT)
         (equal (modifiee x) (position y))
         (equal (sem_type y) +action))
    (equal (sem_type x) +animate))
```

Semantic constraints were relatively easy to create and incorporate into our parser. In fact they were added to the grammar without modifying a single syntactic rule. Preliminary work with prosodic constraints also suggests that prosodic constraints should be as simple to add to our grammar. Finally, we have developed a mechanism for handling pragmatic constraints [11].

In conclusion, our system separates low-level and high-level information sources into loosely-coupled modules. This requires that we provide an effective method to pass appropriate information between the two modules. A word graph provides that link. By separating the system into loosely-coupled modules, we are able to construct and test the modules independently. Our language module uses a uniform mechanism, constraint propagation, to apply a variety of high-level knowledge sources to prune the word graph. Our constraint-based system allows us to develop constraints for different knowledge sources and add them incrementally. The independence of the constraints also allows us to explore the interactions between these knowledge sources.

4 Acknowledgments

This work was supported in part by Purdue Research Foundation and NSF grant number IRI-9011179. I would also like to thank Carl Mitchell for his critical reading of this paper and the referees for their helpful comments.

References

[1] J. Allen. *Natural Language Understanding.* The Benjamin/Cummings Publishing Company, Menlo Park, CA, 1987.

[2] L. R. Bahl, F. Jelinek, and R. L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (PAMI-5):179–190, 1983.

[3] M. Bates, R. Bobrow, P. Fung, R. Ingria, F. Kubala, J. Makhoul, L. Nguyen, R. Schwartz, and D. Stallard. The BBN/HARC spoken language understanding system. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, pages 111–114, 1993.

[4] J. Bear and P. Price. Prosody, syntax, and parsing. In *Proceedings of the 28th annual ACL*, 1990.

[5] J. Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13:94–102, 1970.

[6] L. D. Erman and V. R. Lesser. The Hearsay-II speech understanding system: A tutorial. In W. A. Lea, editor, *Trends in Speech Recognition*, pages 361–381. Speech Science Publications, Apple Valley, MN, 1986.

[7] E. P. Giachin. Automatic training of stochastic finite-state language models for speech understanding. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, March 1992.

[8] D. Goddeau and V. Zue. Integrating probabilistic LR parsing into speech understanding systems. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, pages 181–184, 1992.

[9] T. Hanazawa, K. Kita, S. Nakamura, T. Kawabata, and K. Shikano. ATR HMM-LR continuous speech recognition system. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, pages 53–56, 1990.

[10] M. P. Harper and R. A. Helzerman. PARSEC: A constraint-based parser for spoken language parsing. Technical Report EE-93-28, Purdue University, School of Electrical Engineering, West Lafayette, IN, 1993.

[11] M. P. Harper and R. A. Helzerman. Managing multiple knowledge sources in constraint-based parsing of spoken language. Technical Report EE-94-16, Purdue University, School of Electrical Engineering, West Lafayette, IN, 1994.

[12] M. P. Harper, L. H. Jamieson, C. B. Zoltowski, and R. A. Helzerman. Semantics and constraint parsing of word graphs. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, volume II, pages 63–66, April 1992.

[13] F. Hayes-Roth. Focus of attention in the Hearsay-II speech understanding system. Technical report, Carnegie-Mellon University, Department of Computer Science, 1977.

[14] R. A. Helzerman. PARSEC: A framework for parallel natural language understanding. Master's thesis, Purdue University, School of Electrical Engineering, West Lafayette, IN, 1993.

[15] R. A. Helzerman and M. P. Harper. Log time parsing on the MasPar MP-1. In *Proceedings of the Sixth International Conference on Parallel Processing*, August 1992.

[16] R. A. Helzerman and M. P. Harper. An approach to multiply segmented constraint satisfaction problems. In *Proceedings of the National Conference on Artificial Intelligence*, July 1994.

[17] K. Kita, T. Kawabata, and T. Hanazawa. HMM continuous speech recognition using stochastic language models. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, pages 581–584, 1990.

[18] K. Kita, T. Kawabata, and H. Saito. HMM continuous speech recognition using predictive LR parsing. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, pages 703–706, 1989.

[19] K. Kita and W. H. Ward. Incorporating LR parsing into SPHINX. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, pages 269–272, 1991.

[20] S. R. Kosaraju. Speed of recognition of context-free languages by array automata. *SIAM Journal of Computing*, 4(3) :331–340, September 1975.

[21] J. Kupiec. Hidden Markov estimation for unrestricted stochastic context-free grammars. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, pages 177–180, 1992.

[22] K. Lari and S. J. Young. Applications of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech & Language*, 5(3):237–257, July 1991.

[23] V. R. Lesser, R. D. Fennell, L. D. Erman, and D. R. Reddy. Organization of the Hearsay-II speech understanding system. *IEEE Trans. Acoust., Speech, Signal Processing*, ASSP-23:11–23, 1975.

[24] H. Maruyama. Constraint dependency grammar. Technical Report #RT0044, IBM, Tokyo, Japan, 1990.

[25] H. Maruyama. Constraint dependency grammar and its weak generative capacity. *Computer Software*, 1990.

[26] H. Maruyama. Structural disambiguation with constraint propagation. In *The Proceedings of the Annual Meeting of ACL*, 1990.

[27] M. D. Moshier and W. C. Rounds. On the succinctness properties of unordered context-free grammars. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, 1987.

[28] H. Ney. Dynamic programming parsing for context-free grammars in continuous speech recognition. *IEEE Trans. Signal Processing*, 39(2), February 1991.

[29] D. B. Paul. An efficient A* stack decoder algorithm for continuous speech recognition with a stochastic language model. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, March 1992.

[30] F. Pereira and D. Roe. Empirical properties of finite state approximations for phrase structure grammars. In *Proceedings of the International Conference on Spoken Language Understanding*, October 1992.

[31] P. Placeway, R. Schwartz, P. Fung, and L. Nguyen. The estimation of powerful language models from small and large corpora. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, April 1993.

[32] P. J. Price, W. Fischer, J. Bernstein, and D. Pallett. A database for continuous speech recognition in a 1000-word domain. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, 1988.

[33] W. Ruzzo. Tree-size bounded alternation. *Journal of Computers and System Sciences*, 21:218–235, 1980.

[34] R. Schwartz and S. Austin. A comparison of several approximate algorithms for finding multiple N-best sentence hypotheses. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, May 1991.

[35] R. Schwartz, S. Austin, F. Kubala, J. Makhoul, L. Nguyen, and P. Placeway. New uses for the N-best sentence hypotheses within the Byblos speech recognition system. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, April 1993.

[36] R. Schwartz and Y-L. Chow. The N-best algorithm: An efficient and exact procedure for finding the N most likely sentence hypotheses. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, April 1990.

[37] S. Seneff. Robust parsing for spoken language systems. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, March 1992.

[38] S. Seneff. TINA: A natural language system for spoken language applications. *American Journal of Computational Linguistics*, 18:61–86, 1992.

[39] W. Ward. Understanding spontaneous speech: The Phoenix system. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, May 1991.

[40] C. S. Wetherell. Probabilistic languages: A review and some open questions. *Computing Surveys*, 12(4):361–379, July 1980.

[41] J. H. Wright. LR parsing of probabilistic grammars with input uncertainty for speech recognition. *Computer Speech and Language*, 4:298–323, 1990.

[42] C. B. Zoltowski, M. P. Harper, L. H. Jamieson, and R. A. Helzerman. PARSEC: A constraint-based framework for spoken language understanding. In *Proceedings of the International Conference on Spoken Language Understanding*, October 1992.

[43] V. Zue, J. Glass, D. Goodine, H. Leung, M. Phillips, J. Polifroni, and S. Seneff. Integration of speech recognition and natural language processing in the MIT Voyager system. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, May 1991.