10

Constraint Dependency Grammars: SuperARVs, Language Modeling, and Parsing

Mary P. Harper and Wen Wang

10.1 Introduction

Over the past ten years, we have been investigating the use of Constraint Dependency Grammar (CDG). This effort began shortly after Maruyama introduced the concept of a CDG (Maruyama, 1990b,c). In Maruyama's original vision, grammar constraints were used to limit dependencies in the space of potential parses of a sentence to those that are valid. This vision of CDG was adapted by the speech group at Purdue University in order to address speech processing applications and by Menzel and his students at the University of Hamburg (Heinecke et al., 1998; Menzel, 1994, 1995, 1996; Schröder et al., 2000) for applications requiring grammar robustness, such as the diagnosis of grammar errors by second language learners. These two foci have had deep impacts on the way each of these labs has utilized CDG. This chapter focuses on our spoken language processing path.

Our goal was to build effective and efficient linguistically motivated language models (LMs) for large vocabulary continuous speech recognition (LVCSR) tasks. CDG was selected as the underlying grammar formalism for this language modeling work for several reasons. CDG is able to utilize a variety of knowledge sources in the constraints used in parsing, and these constraints can be ordered for efficiency, withheld, or even relaxed. Hence, CDG provides a flexible framework for combining multiple knowledge sources. A parse of a sentence in CDG is represented distributively as a set of values assigned to variables associated with each word in the sentence; hence, a CDG model can be highly lexicalized. Also, CDG can model languages with crossing dependencies and free word ordering; hence, the results of our efforts can be applied to languages with these phenomena.

This chapter first provides a brief overview of CDG. It then describes how a CDG can be extracted from a treebank of CDG annotated sentences. Next is a discussion of our statistical CDG parser in which we evaluate some of the factors contributing to its performance. Finally, two types of CDG-based language models are presented and evaluated.

10.2 Parsing with Constraint Dependency Grammar

Maruyama (1990b,c) defined a CDG as a four-tuple, $\langle \Sigma, R, L, C \rangle$, where $\Sigma = \{\sigma_1, \ldots, \sigma_c\}$ is a finite set of lexical categories (e.g., determiner), $R = \{r_1, \ldots, r_p\}$ is a finite set of uniquely named roles (e.g., governor, need1, need2), $L = \{l_1, \ldots, l_q\}$ is a finite set of labels (e.g., subject), and C is a constraint formula. The number of roles (per word) p in a CDG is the *degree* of the grammar. In CDG, a *sentence* $s = w_1 w_2 w_3 \ldots w_n$ has a length $n \geq 1$ and is an element of Σ^* (that is, the parser operates over lexical categories). Given a grammar of degree p, each $w_i \in \Sigma$ of a sentence s has p different roles, yielding $n * p$ roles for the entire sentence. Each *role* is a variable that is assigned a *role value*, which is a tuple consisting of a label $l \in L$ and a modifiee m (a position of a word in the sentence) and is depicted in parsing examples as *l-m*. The label l indicates a syntactic function for the word when its role is assigned that role value, and m specifies the position that the word is *modifying* when it takes on the function specified by l.

Given that $L(G)$ is the language generated by the CDG G, a sentence of length n, $s = w_1 w_2 w_3 \ldots w_n$, is in $L(G)$ if for every w_i there is at least one assignment of role values to each of the roles of w_i such that the constraints in C are satisfied. Hence, a parse is represented by an assignment of role values to all the roles in s that records the syntactic functions of the words and their dependents. Consider, for example, the parse for *clear the screen* depicted in figure 10.1. Each word has a single lexical category in Σ and a set of role value assignments that are used to represent the parse. Maruyama introduced two types of roles: governor and need roles. The governor role of a word is assigned a role value such that the word's modifiee is the position of that word's governor or head. For example, the role value assigned to the governor role (denoted G in examples) of *the* is det-3, where the label det indicates its grammatical function and the modifiee 3 is the position of its head, *screen*. The need roles (denoted N1, N2, and N3 in this example) are used to ensure that the grammatical requirements of a word are met. The use of need roles enables CDG to distinguish between complements and adjuncts. In the example shown in figure 10.1, the verb *clear* needs an object (and so the role value assigned to N2, S-3, has a modifiee that points to the object *screen*). Maruyama uses the label blank and a nil modifiee to indicate that a role value is not used for a particular role. Because the verb *clear* does not require a subject or another complement, the role values of its other need roles are set to blank-nil. In our CDG parser implementation, we have found that it is more parsimonious to indicate that a role value has no modifiee by using the word's position rather than nil (e.g., the N1 role value for *clear* in figure 10.1 would be blank-1 rather than blank-nil). This convention will be used henceforth in this chapter.

Maruyama framed the CDG parsing algorithm as a constraint satisfaction problem: the rules are the constraints and the solutions are the parses. During parsing, the grammaticality of a sentence is determined by applying C to all possible role value assignments, applying node and arc consistency so that assignments and assignment pairs are consistent with C, and then extracting a consistent parse assignment. Murayama defined C as a first-order predicate calculus formula over all roles; a valid assignment of role values to roles must be consistent with C. Each subformula P_i in C is a predicate involving $=$, $<$, or $>$, or predicates joined by the logical connectives and, or, if, or not. A subformula is called a *unary constraint* if it contains only a single variable (by convention x_1) and a *binary constraint* if it contains two variables (by convention x_1 and x_2). A unary constraint restricts assignments of role values to

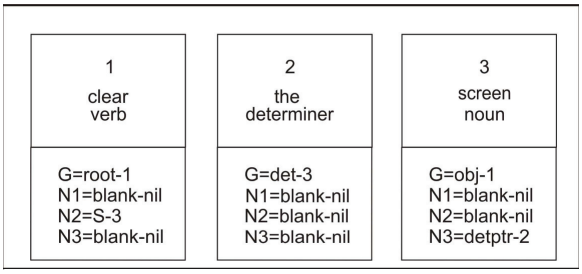


FIGURE 10.1 A CDG parse for *clear the screen* is represented by the assignment of role values to roles associated with each word in the sentence.

roles. A binary constraint limits pairs of assignments. See figure 10.2 for a parsing example with $\Sigma = \{\text{det, noun, verb}\}$, $R = \{\text{governor}\}$, $L = \{\text{DET, SUBJ, ROOT}\}$. The figure shows the steps used by a traditional CDG parser using the unary and binary constraints of C depicted in the figure. After arc consistency, parses are extracted from the remaining role values (see Maruyama, 1990c for more detail). A sentence s is said to be generated by the grammar G if there exists an assignment $\mathbf{A}$ that maps a role value to each of the roles for s such that C is satisfied. There may be more than one assignment of role values to the roles of a sentence that satisfies C, in which case there is ambiguity. A CDG has an *arity* parameter a, which indicates the maximum number of variables in the subformulas of C (e.g., the grammar in figure 10.2 has an arity of two).

Because Maruyama represents the problem of parsing a sentence in CDG as a constraint satisfaction problem (CSP), his parsing algorithm operates on sentences for which each word is assigned a single lexical class. Harper and Helzerman (1995a) expanded the scope of CDG to enable the analysis of sentences containing lexically ambiguous words, to support the inclusion of lexical features (e.g., subcategorization, number, case), and to operate on word graphs containing multiple alternative sentence hypotheses. To support this, they extended the constraint satisfaction problem; a MUSE CSP (*MU*ltiply *SE*gmented *C*onstraint *S*atisfaction *P*roblem) (Helzerman and Harper, 1996) is able to support problems that arise naturally in applications for which it can be quite difficult to segment the data in only one way (e.g., computer vision, speech processing, and handwriting recognition). To parse efficiently, algorithms for MUSE arc and path consistency were developed (Helzerman and Harper, 1996) and refined (Harper et al., 1999b). Methods for staging in feature constraints were also developed to control parse forest size and reduce parsing time (Harper et al., 1999a).

Given these modifications to support lexical ambiguity and features, a CDG is defined as a five tuple, $\langle \Sigma, R, L, C, T \rangle$, where Σ, R, L, and C are defined as previously, and T is a table that specifies which roles are used for each lexical category (most lexical classes need only one or two (Harper et al., 1999a), e.g., determiners use only the governor role, but others use additional roles, e.g., verbs), the set of labels that are supported for each role and lexical category, the domain of feature values for each feature type (if there are k feature types, the domains are denoted as $F_1, F_2, \ldots, F_k$), the feature types that are defined for each category in Σ, and the subset of feature values allowed by each category and feature type combination. A dictionary of lexical entries for words is also used for parsing. A lexical entry is made up of one lexical category $\sigma \in \Sigma$ and a single feature value for each feature supported

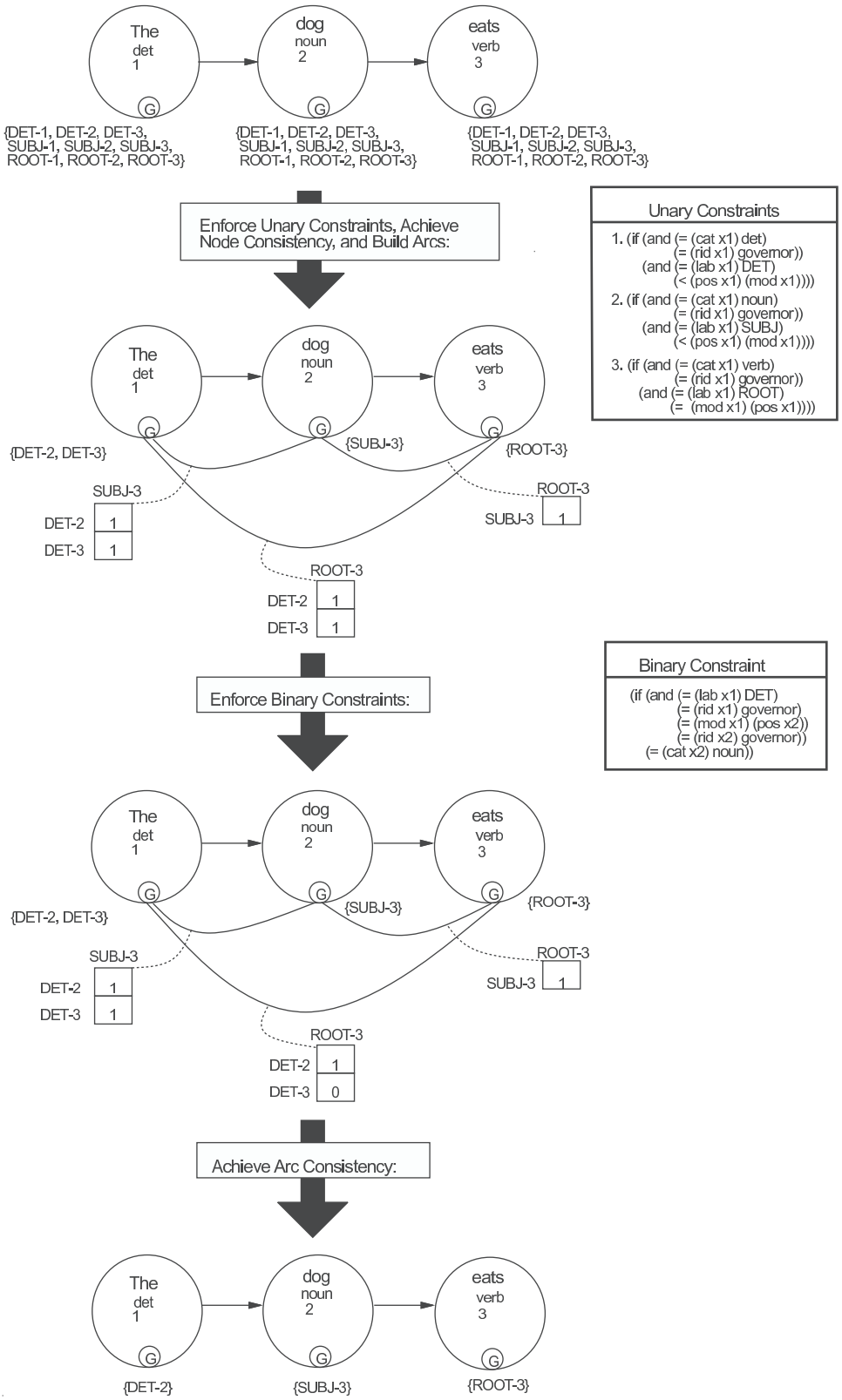


FIGURE 10.2 Using unary and binary constraints to parse the sentence: *The dog eats.*

by σ. Each word has one or more associated lexical entries. An example parse, given these modifications, appears in figure 10.3 for a sentence from the DARPA Naval Resource Management (RM) corpus (Price et al., 1988). This corpus has a vocabulary of around 1,000 different words drawn from queries in the form of *wh*-questions and yes/no-questions involving naval resources or commands for controlling an interactive database and graphics display. We constructed a conventional CDG for RM by hand with around 1,500 unary and binary constraints (that is, its arity is 2) that were designed to parse the sentences in this corpus. This CDG covers a wide variety of grammar constructs (including conjunctions and wh-movement) and has a fairly rich semantics. It uses 16 lexical categories, 4 roles (so its degree is 4), 24 labels, and 13 lexical feature types (agr, behavior [e.g., mass], case, conjtype, gap, inverted, mood, semtype, subcat, takesdet, type [e.g., interrogative, relative], voice, and vtype [e.g., progressive]). The parse shown at the top of figure 10.3 is an assignment of role values to roles that is consistent with the unary and binary constraints of this grammar.

In order to derive C directly from CDG annotated sentences, we then developed a method to extract grammar relations using information derived directly from annotated sentences (Harper et al., 2000). Using relative position information, unary constraints can be represented as a finite set of *abstract role values* (ARVs). Formally, an ARV for a particular grammar $G = \langle \Sigma, R, L, C, T \rangle$ is an element of the set: $\mathcal{A}_1 = \Sigma \times R \times L \times F_1 \times \ldots \times F_k \times \mathrm{UC}$, where k is the number of feature types defined in T, F_i represents the set of feature values for that type, and UC encodes the three possible positional relations between the position (denoted $\mathrm{P}x_1$) and modifiee (denoted $\mathrm{M}x_1$) of a role value assigned to the role of a word ($\mathrm{M}x_1$ and $\mathrm{P}x_1$ can be related by $<$, $>$, or $=$). The box under the parse in figure 10.3 shows an example of an ARV for the role value assigned to the governor role of the word *the* obtained from the parsed sentence. Note that an ARV can be extended to include information about the lexical category and features of the modifiee, that is, $\mathcal{A}'_1 = \Sigma \times R \times L \times F_1 \times \ldots \times F_k \times \mathrm{UC} \times (\Sigma \times F_1 \times \ldots \times F_k)$. This added information is equivalent to modifiee constraints investigated in Harper et al. (1999a); these constraints when included in unary constraints (and equivalently ARVs) do not change grammar coverage and help to improve parse times by eliminating ungrammatical role values during the less costly unary constraint enforcement stage of parsing. The ARV in figure 10.3 contains modifiee constraints that appear surrounded by parentheses.

Similarly, binary constraints are represented as a finite set of abstract role value pairs (ARVPs), which are members of the domain $\mathcal{A}_2 = \Sigma \times R \times L \times F_1 \times \ldots \times F_k \times \Sigma \times R \times L \times F_1 \times \ldots \times F_k \times \mathrm{BC}$, where BC encodes for a pair of role values the positional relations among their positions ($\mathrm{P}x_1$ and $\mathrm{P}x_2$) and modifiees ($\mathrm{M}x_1$ and $\mathrm{M}x_2$) (there are six ways to pair the position and modifiee of two role values, each of which can be related in three different ways; Harper et al., 2000; White, 2000). The box under the example ARV in figure 10.3 shows an example of an ARVP obtained from the parsed sentence. Modifiee constraints can also be added to the ARVPs, that is, $\mathcal{A}'_2 = \Sigma \times R \times L \times F_1 \times \ldots \times F_k \times \Sigma \times R \times L \times F_1 \times \ldots \times F_k \times \mathrm{BC} \times (\Sigma \times F_1 \times \ldots \times F_k) \times (\Sigma \times F_1 \times \ldots \times F_k)$. However, since the ARVP space is larger than the ARV space, using this information could generate a very large, overly specific grammar. The ARVP in figure 10.3 contains two sets of modifiee constraints, each surrounded by parentheses.

An enumeration of the positive ARV/ARVPs can be used to represent the CDG constraints, C, thus simplifying the process of writing constraints. A role value would be supported by the ARVs only if it appears in the set of ARVs for the grammar. Similarly, a role value pair would be supported by the ARVPs only if it appears in

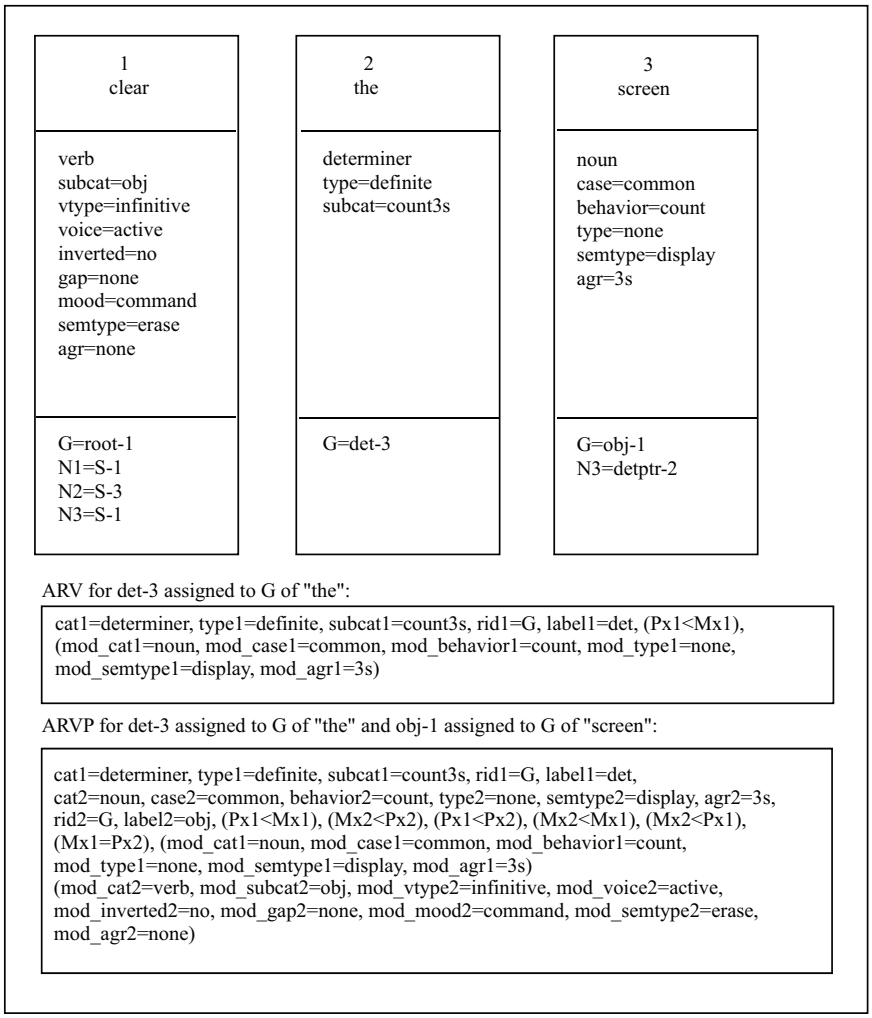


FIGURE 10.3 A CDG parse for *clear the screen*, shown in the top portion of this figure, is represented by the assignment of role values to roles associated with a word that has a specific lexical category and one feature value per feature. ARVs and ARVPs (see an example of each in the bottom two boxes) represent grammatical relations that can be extracted from a sentence's parse.

the set of ARVPs. A fast table lookup method was developed (Harper et al., 2000; White, 2000) to determine whether a role value (or role value pair) is allowed (rather than propagating thousands of constraints) to speed up parsing.

Given Maruyama's initial research and our extensions, CDG offers a flexible and powerful parsing framework. First, the set of languages accepted by a CDG is a superset of the set of languages that can be accepted by context-free grammars (CFGs). Maruyama (1990a,b) proved that any arbitrary CFG converted to Griebach Normal form can be converted into a CDG with a degree of two and an arity of two that accepts the same language as the CFG. In addition, CDG can recognize languages that CFGs cannot, for example, $a^n b^n c^n$ (where a, b, and c are terminal symbols) or ww (where w is some string of terminal symbols). Note that grammars recognizing these languages utilize crossing dependencies, not supported by a CFG, to represent the positional correspondences. Harbusch (1997) has shown also that every language recognized by a tree-adjoining grammar (TAG) can also be generated by a CDG, but not vice versa (e.g., $L_6 = a^n b^n c^n d^n e^n f^n$). Like other dependency grammars (Holan et al., 1998; Järvinen and Tapanainen, 1998; Polguère and Kahane, 1998; Tesnière, 1959), free-order languages can be handled without enumerating all permutations (Harper and Helzerman, 1995a). In addition, the CDG parser uses sets of constraints that operate on role values assigned to roles to determine whether or not a string of terminals is in the grammar. These constraints can make use of syntactic, prosodic, semantic, and context-dependent knowledge (Harper and Helzerman, 1995a,b; Menzel, 1995, 1996). Constraints can be ordered for efficiency, withheld, or even relaxed. The presence of ambiguity can trigger the use of stricter constraints to further refine the parse for a sentence (Harper and Helzerman, 1995b). This flexibility can be utilized to create a smart language processing system: one that decides when and how to use its constraints based on the state of the parse. Furthermore, a CDG parser can be parallelized to speed up parsing (Helzerman and Harper, 1992). Finally, grammar constraints can be derived by reading them from a corpus of CDG parses (Harper et al., 2000), as was described in this section.

10.3 Grammar Induction Experiments

Before developing a statistical parsing model for CDG, we carried out a series of grammar induction experiments. We did this work prior to building statistical models in order to evaluate the learnability of various types of CDG constraints and their importance for reducing parse ambiguity. For this effort, we developed a CDG treebank for the 2,844 sentences in the RM corpus. This corpus was chosen for several reasons: it provided a good source of both training and testing materials; we already developed a CDG by hand for the domain; the sentences have syntactic variety and a definable semantics; the scope of the problem was limited enough to enable systematic investigation of a wider variety of grammar development techniques than would be possible with larger corpora; it is a domain-specific task that enabled investigation of semantics; and the underlying grammar that we were trying to learn was well defined, which is important for evaluating the learnability of CDG from corpora.

Although there had been some previous work carried out on corpus-based learning of other grammar formalisms, in particular on CFGs, these prior results provided only limited guidance for our task of investigating whether a CDG could be learned from a corpus of sentences and what would be needed to successfully learn a high-quality grammar. CFGs appear to have a different learning bias than a CDG. For example, consider the CFG and CDG (only unary constraints are depicted) that would be

obtained from the parsed sentence shown in figure 10.4. Once the CFG has learned an NP rule, it can be used in any context, making the grammar more general than the CDG on the basis of this single example. By contrast, the CDG is unable to parse the new sentence given the rules obtained from the dependency parse. The CDG must see an example of a pronoun being governed by a verb to its left in order to allow that relation. Hence, it appears that CDGs learned from a corpus of sentences would need a larger training corpus or require more careful training to achieve the desired level of generality.

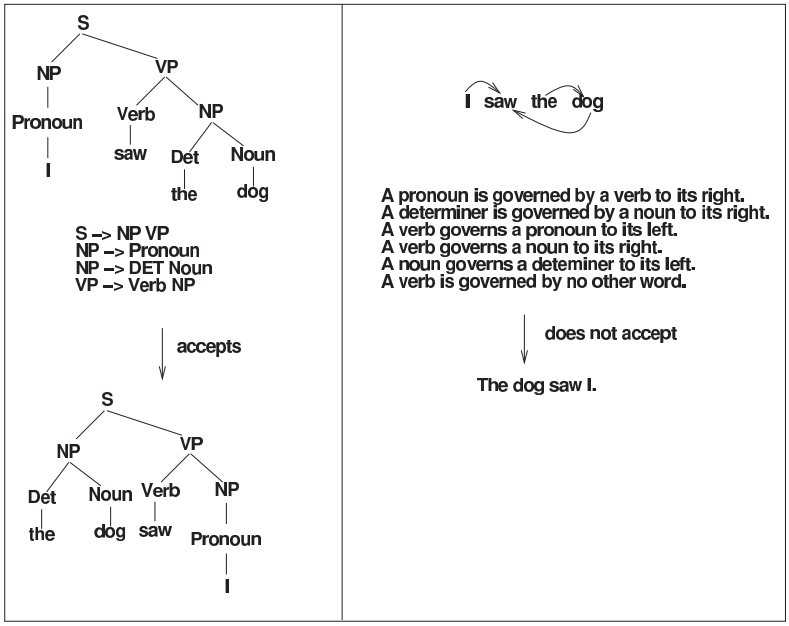


FIGURE 10.4 Comparing CFG and CDG generality on a simple example.

In our investigations, we considered two important characteristics of learned grammars: generality and selectivity (Allen, 1995). Ideally, a learned grammar should be general enough that it covers unseen sentences in the target language but restrictive enough that it does not generate a large number of spurious parses. There is clearly a trade-off between these two attributes, and yet both attributes are important for developing high-quality statistical CDG-based parsing and language models.

We have investigated a variety of methods for deriving C directly from sentences in the RM corpus annotated with CDG parse relations (Harper and Wang, 2001; Harper et al., 2000; Wang, 2003; White, 2000). Since the number of distinct ARVs tend to be more limited than the number of distinct ARVPs (see White, 2000), in all the grammars we have investigated, ARVs include all the information that can be extracted from the parse concerning a single role assignment: the role, the label of its assigned role value, the category and feature values of its word, a UC relation, and the modifiee's lexical category and feature values. To obtain ARVPs that are

closest in spirit to the binary constraints of Maruyama, they would include lexical category, role, label, modifiee, and feature information for all pairs of role values from the annotated sentences. When ARVPs are extracted in this way, they are called Full grammars. Modifiee constraints can also be added to the ARVPs, but since the ARVP space is larger than the ARV space, using this information could generate a very large, overly specific grammar. We examined Full grammars with and without modifiee constraints (Full Mod versus Full) in order to explore the impact of this added information. In a series of experiments described in several documents (Harper and Wang, 2001; Harper et al., 2000; Wang, 2003; White, 2000), we investigated the generality and selectivity of CDGs extracted from CDG-annotated treebanks. Two important factors impacting grammar generality were investigated: increasing the size of the corpus (either directly or by enhancing what can be learned from annotations) and selectively relaxing grammar constraints.

To examine the first factor, we constructed a treebank for the RM sentences in two different ways. The first treebank involved a set of sentences annotated using the CDG annotation tool SENATOR (Harper et al., 2000; White, 2000) together with an active learning, selective sampling procedure similar to Thompson et al. (1999). The second treebank was composed of the same sentences updated to include subgrammar invocations (e.g., dates, years, times, numbers, and latitude and longitude coordinates) in place of some of the word strings (Harper and Wang, 2001; Harper et al., 2000; Wang, 2003) in an attempt to enhance what could be learned from a sentence in the corpus. We compared the Full and Full Mod grammars extracted from the sentence and subgrammar-expanded treebanks on grammar size, learning rate, the generality and parse ambiguity on an independent set of 4,946 sentences generated from the underlying grammar model used to create the original Resource Management corpus sentences (called RM domain sentences) and the reranking of sentence recognition hypotheses output by a speech recognition system trained on the RM speech corpus (Harper and Wang, 2001; Harper et al., 2000; White, 2000). The coverage and parse ambiguity for the Full and Full Mod grammars extracted from the standard sentence treebank can be compared in the baseline curves in figure 10.5(a) and figure 10.5(b), respectively; the modifiee constraints reduce parse coverage for the test set significantly, while reducing ambiguity only slightly. The coverage improves dramatically for the Full and Full Mod grammars when using the subgrammar-expanded treebank as can be observed in the baseline curve in figure 10.6(a); whereas, ambiguity only increases slightly (see figure 10.6(b)). In general, the Full Mod grammars are larger than the Full grammars regardless of the treebank type, and the subgrammar-expansion process increases the grammar size for the Full Mod grammar (178.65% increase) more dramatically than the Full grammar (67.51% increase) (Harper and Wang, 2001; Wang, 2003). The Full and Full Mod grammars trained using the subgrammar-expanded treebank achieve a greater coverage with a lower percentage of the corpus observed and generalize better to unseen test sentences than their corresponding sentence grammars with only a minor increase in ambiguity (Harper and Wang, 2001; Wang, 2003). They also perform much better when reranking the sentence hypotheses output by the speech recognizer (Harper and Wang, 2001; Wang, 2003). This suggests that annotation of subgrammar-expanded sentences is an effective method for intelligently increasing the size of a training set for developing high-quality domain-specific CDGs (e.g., radiology reports).

Even though higher quality grammars are obtained by taking the extra effort to identify and use subgrammars within sentences (Harper and Wang, 2001; Harper et al., 2000; Wang, 2003; White, 2000), this approach requires more effort than simply

learning grammars from sentences; hence, we also investigated the impact of selective constraint relaxation to balance generality and selectivity (Harper and Wang, 2001; Harper et al., 2000; Wang, 2003; White, 2000). Relaxing CDG constraints is a double-edged sword: it can improve the generality of the grammars, but at the cost of allowing the incorrect acceptance of ill-formed sentences or spuriously increasing the amount of parse ambiguity. The features used in our grammars are important for enhancing the selectivity of a CDG; however, in many cases the features associated with a pair of role values may be irrelevant to the pair (e.g., for many transitive verbs, the number of the object noun should not prevent it from acting as its direct object) making the grammar less general than it should be. We have investigated the impact of fairly straightforward global relaxation methods: reducing grammar degree (that is, the full grammar versus one that uses only the governor role), relaxing specific features (e.g., semantics), or relaxing both (Wang, 2003). We also investigated the impact of relaxing constraints on role value pairs based on whether or not they are linked by a modifiee relation (that is, one role value has a modifiee pointing to the other or they share a modifiee) (Harper and Wang, 2001; Harper et al., 2000; Wang, 2003; White, 2000). The rationale for this constraint relaxation method is that role value pairs that are not linked should be much less important to the parse and so their violation should not cause the parse to fail. In one variation, we ignore feature information when there is no modifiee relation (denoted Feature), and in the other (denoted Direct), we allow any role value pair to exist when they are not directly linked by a modifiee link, that is, the only pairs that can be eliminated by the grammar constraints are those directly linked by a modifiee (the others would need to be deleted by the arc consistency algorithm). We also examined a knowledge-based (KB) feature relaxation approach (Wang, 2003); the fundamental idea is that for a particular role value pair, only some features are relevant to the relation. See Wang (2003) for more details.

Each of these constraint relaxation methods was evaluated by parsing the RM domain sentences after learning the grammar based on our original CDG treebank or based on the subgrammar-expanded treebank. First, we examine the impact of relaxing constraints on role value pairs based on whether or not they are linked by a modifiee relation. As can be seen in the baseline curve in figures 10.5(a) and 10.6(a), the Full grammars are less general than Feature grammars, which are slightly less general than the Direct variant. The fact that the Direct and Feature grammars had a far greater generality than the Full grammar, with only a limited increase in parse ambiguity (see the baseline curve in figures 10.5(b) and 10.6(b)), suggests that focusing on learning specific feature information for role values that share a modifiee link is an effective strategy, a strategy that we exploited in developing our statistical parsing model. The grammars with modifiee constraints tend to be less general than their counterparts without modifiee constraints while being slightly more selective (less parse ambiguity) when extracted from the standard treebank; however, modifiee constraints offer a good balance between coverage and ambiguity containment when using the subgrammar-expanded treebank.

Considering the global relaxation methods of reducing the grammar degree to governor only, eliminating semantic constraints, and the combination of both, we find in figures 10.5 and 10.6 that these methods improve coverage, but at a cost of increased parse ambiguity. Degree relaxation has less impact on coverage and ambiguity than relaxation of semantics, although their combination results in a slight increase in coverage and a large increase in parse ambiguity compared to relaxation of semantics alone. Based on these results, it appears that semantics is difficult to

learn together with parse rules, and that the use of multiple roles helps to contain some of the ambiguity when the semantic information is ignored.

We next examine the effect of the more specific knowledge-based relaxation approach. As can be seen in figures 10.5 and 10.6, the global relaxation methods were less effective at containing parse ambiguity than the knowledge-based approach. The knowledge-based method relaxes the features for role value pairs to those that were identified as being important for selectivity (e.g., the agr feature is important for a subject and verb pair, but not an object and a preposition). Considering the grammar coverage and parse ambiguity plots in figures 10.5 and 10.6, we observe that KB Feature Relaxation improves generality of all grammar extraction variations while only increasing parse ambiguity slightly relative to the baseline grammar. Although the Direct and Feature grammars are much less general when they also encode modifiee information, if they are combined with KB feature relaxation, they achieve a good balance between generality and selectivity. Overall, KB Feature Relaxation produced grammars with a good trade-off between generality and selectivity for both sentence grammars and subgrammar-expanded sentence grammars.

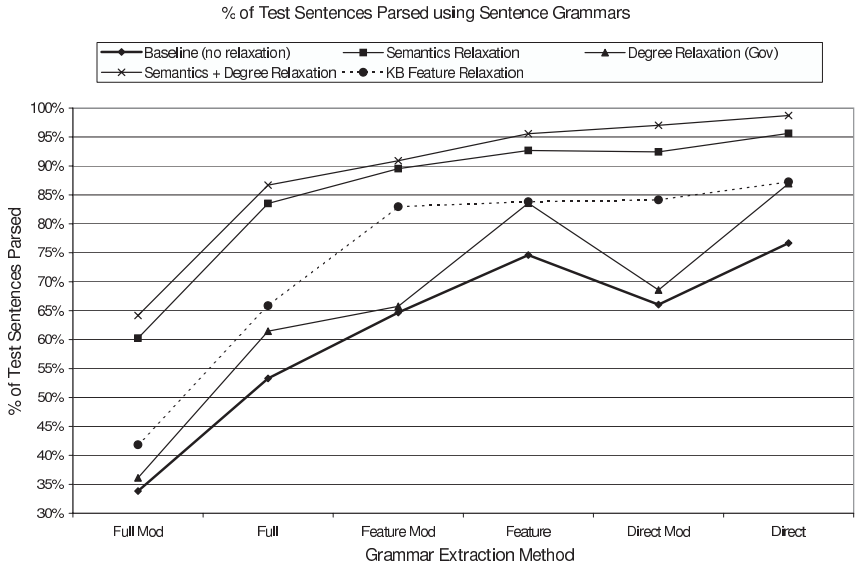
We have investigated the impact of corpus enrichment and selective constraint relaxation on improving grammar quality by balancing generality and selectivity (Harper and Wang, 2001; Harper et al., 2000; Wang, 2003; White, 2000). This research was important for several reasons. First, CDG has a different inductive bias than CFG, suggesting that we will need to work more effectively with and/or identify additional training data when constructing the statistical models. It is critical to identify materials that cover the phenomena we wish to model. Second, although features have proven to be an effective way to reduce the number of roles needed to represent grammar relations in CDG (White, 2000), the inclusion of feature information in ARVPs dramatically increases the size of the ARVP space (Harper and Helzerman, 1995b), making it more challenging to obtain training data with sufficient coverage. Some features are more relevant for some relationships among words than for others; exploiting this insight should be very important especially when modeling modifiee constraints in our statistical models. Third, we have found that focusing on ARVPs that involve modification links provides a good balance between selectivity and generality. It is infeasible to integrate all the ARVPs of a Full model into a statistical CDG parsing model; fortunately, the grammar induction results suggest that an effective statistical CDG parsing model can be constructed by focusing on those parsing relations involving direct modifiee links.

10.4 Statistical CDG: SuperARVs and Modifiees

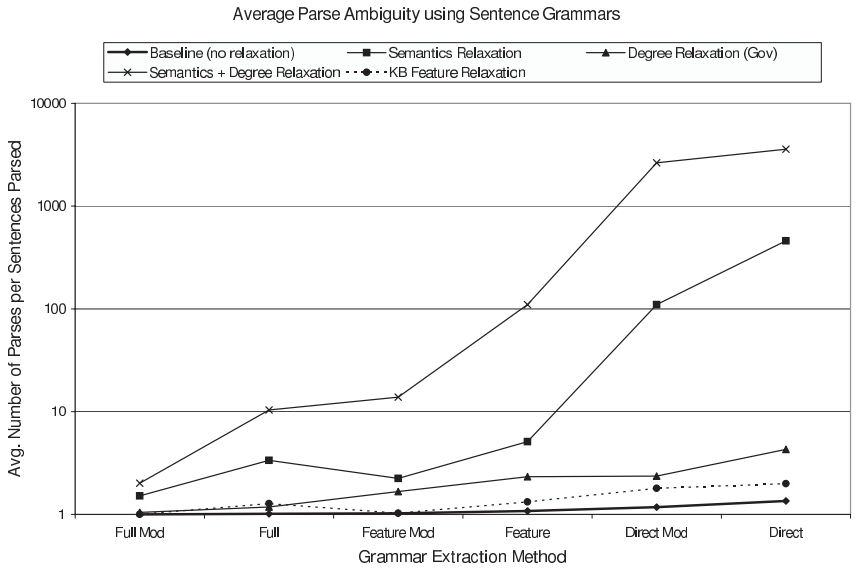
This section first describes the concept of the Super Abstract Role Value (SuperARV), the data structure central to our statistical modeling efforts. Next, we discuss our statistical CDG parser together with the factors contributing to its performance. Finally, two types of CDG-based language models are presented and evaluated.

10.4.1 The SuperARV

CDG parse information can be easily lexicalized at the word level by combining the information associated with a word in a CDG parse into a structure that we call a Super Abstract Role Value (SuperARV). A SuperARV combines the lexical category of a word, a rich set of lexical features, and the syntactic constraints represented by all its role value assignments and how their modifiees interrelate. It is super in the sense that it includes all of the information associated with the word (except its word

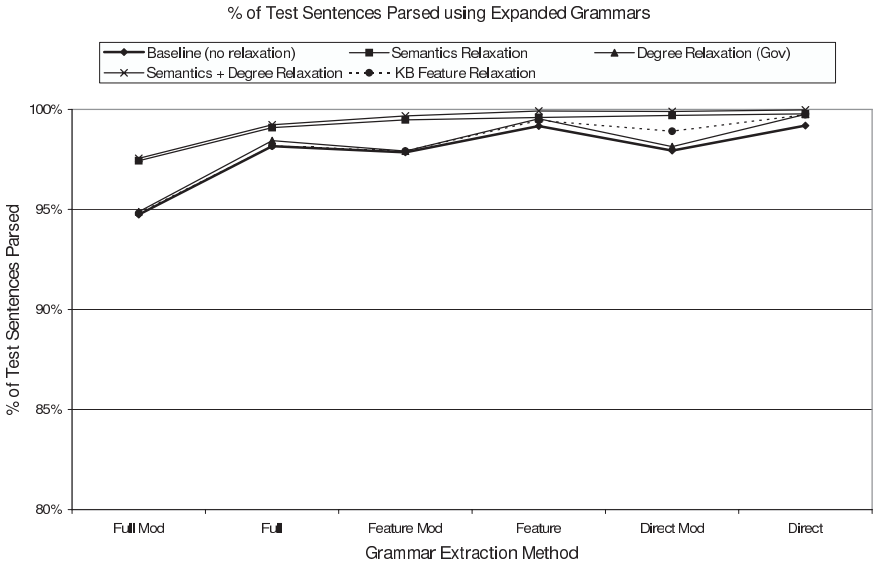


(a)

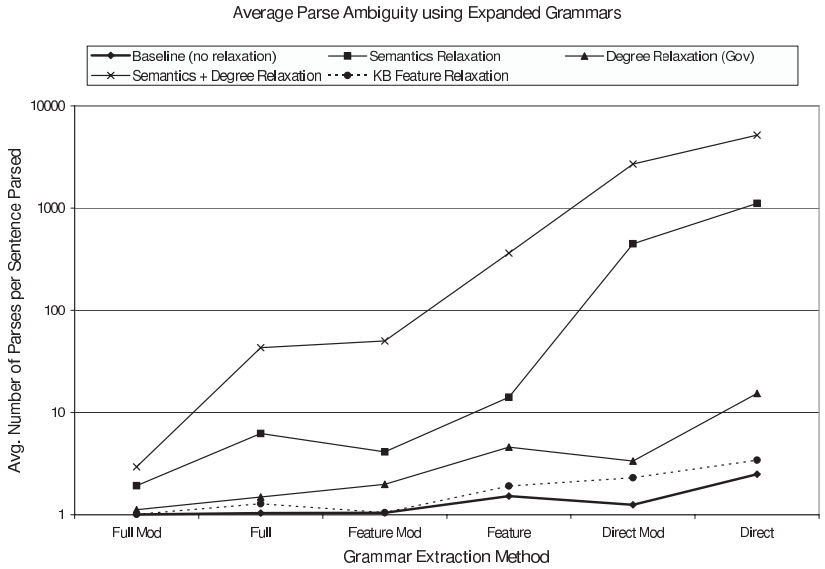


(b)

FIGURE 10.5 Percentage of RM domain sentences parsed for each grammar extraction method (shown in subplot a) and the corresponding parse ambiguity (shown in subplot b) given the indicated relaxation on sentence grammars.



(a)



(b)

FIGURE 10.6 Percentage of RM domain sentences parsed for each grammar extraction method (shown in subplot a) and the corresponding parse ambiguity (shown in subplot b) given the indicated relaxation on subgrammar-expanded grammars.

string) in a parse. The concept of a SuperARV was inspired by the supertag concept from Lexicalized Tree-Adjoining Grammar (LTAG) (Joshi et al., 1975; Schabes, 1992; Schabes et al., 1988). A supertag encodes lexical dependencies as well as syntactic and semantic constraints in a representation of supertag-based classes that are more fine-grained than part-of-speech (POS) based classes (Joshi and Srinivas, 1994; Srinivas, 1996).

The concept of a SuperARV is central to our development of statistical CDG-based parsing and language models. A SuperARV is formally defined as a four-tuple for a word, $\langle C, F, (R, L, UC, MC)+, DC\rangle$, where C is the lexical category of the word, $F = \{Fname_1 = Fvalue_1, \ldots, Fname_f = Fvalue_f\}$ is a feature vector representing the lexical feature values for the word (where $Fname_i$ is the name of a feature and $Fvalue_i$ is its corresponding value), (R, L, UC, MC)+ is a list of one or more four-tuples, each representing an abstraction of a role value assignment, where R is a role variable (e.g., governor), L is a label (e.g., np), UC represents the relative positional relation for a word and its modifiee dependent, MC encodes modifiee constraints for the dependent (e.g., the lexical category and/or lexical features of the modifiee), and DC represents binary constraint (ARVP) information concerning the relative ordering of the positions of the corresponding word and all its modifiees. Since modifiee constraints can make the grammar quite specific, they need to be added with care; hence, in initial studies, we included only the lexical category of the modifiee.

The gray box in figure 10.7 shows an example of a SuperARV for the word *clear* derived from the CDG parse for the sentence *clear the screen*. Note that the SuperARV structure provides an explicit way to organize all the information concerning one consistent set of dependency links for the word *clear* that can be directly derived from its parse assignments. The SuperARV aggregates the four ARVs (unary constraints) that would be extracted for that word in the parse in a factored form; the lexical category and feature information is shared across these ARVs in the SuperARV representation. In addition, the representation aggregates information from six ARVPs; these ARVPs would be extracted over the pairs of role value assignments for the word *clear* and essentially add additional information on the relative positions of the word and each of its dependents. Based on this, a SuperARV can be thought of as providing a set of admissibility constraints on syntactic and lexical environments in which the word may be used. It encodes lexical information as well as syntactic constraints in a uniform fine-grained representation.

Like ARVs and ARVPs, SuperARVs can be extracted from a corpus of CDG annotated sentences. Once the SuperARVs are extracted, it is common for words to have more than one SuperARV to indicate different uses. The average number of SuperARVs for words of different lexical categories varies, with verbs having the greatest SuperARV ambiguity. This is mostly due to the wide variety of feature values and variations on complement types and positions found for verbs. It should be noted that the set of lexical features used in our statistical models include agr, behavior, case, gap, inverted, mood, type, voice, and vtype. We eliminated semtype based on the grammar induction experiments; learning semantic distinctions while learning syntactic constraints requires greater resources than are currently available for CDG. We also eliminated features that were redundant given information encoded by the need roles (e.g., subcat).

As in Joshi and Srinivas (1994), a sentence can be tagged with SuperARVs to provide an almost-parse for the sentence; to produce a parse all that remains is to specify the precise position of each modifiee. SuperARVs inherit an important

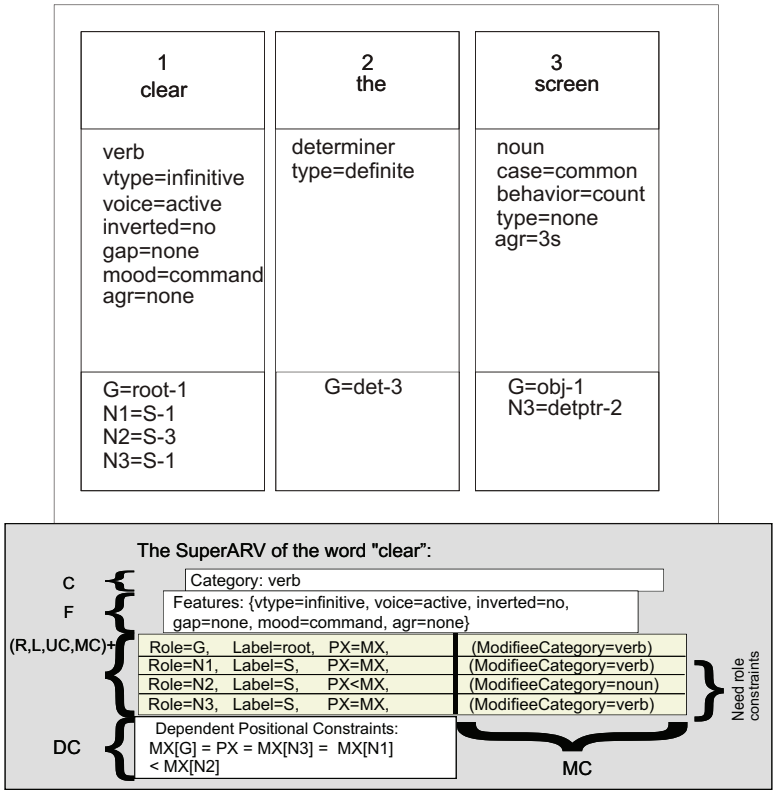


FIGURE 10.7 An example of a CDG parse and the SuperARV for the word "clear" from the sentence *clear the screen*. Note: *G* represents the governor role; the need roles, *N1*, *N2*, and *N3*, are used to ensure that the grammatical requirements of the word are met. PX and MX([R]) represent the position of a word and its modifiee (for role R), respectively.

characteristic from CDG that has proven useful for constructing probabilistic models: the addition of more knowledge sources tightens constraints but ignoring knowledge sources loosens them. Selective constraint relaxation can be implemented by eliminating one or more component from the SuperARV structure. This provides us with a very flexible framework not only for modeling natural language but also for supporting smoothing (that is, we can build backoff models for smoothing using SuperARVs that encode less specific information).

10.4.2 Statistical CDG parsing

Our Statistical CDG (SCDG) parser is a probabilistic generative model described in detail in Wang (2003) and Wang and Harper (2004). For all sentences s and all parses D, the parser assigns a probability $P(s, D) = P(D)$ for all D with yield s. For s, the parser returns the parse D that maximizes its probability, as follows:

$$\begin{aligned} argmax_d P(D|s) &= argmax_D P(s, D) \\ &= argmax_D P(D) \end{aligned} \tag{10.1}$$

The parser can be viewed as consisting of two components: SuperARV tagging and modifiee determination. These two steps can be either loosely or tightly integrated. To simplify discussion, we describe the loosely integrated algorithm, but we have implemented and evaluated both strategies. The basic parsing algorithm for the loosely integrated case is summarized in figure 10.8, with the algorithm's symbols defined in table 10.1. Later in this section, we will describe additions to this basic model.

In the first step of parsing (item 1 in figure 10.8), the top n-best SuperARV assignments are generated for an input sentence $w_1, \ldots, w_n$ using token-passing (Young et al., 1997) on a Hidden Markov Model with trigram probabilistic estimations for both transition and emission probabilities. Each SuperARV sequence for the sentence is represented as a sequence of tuples: $\langle w_1, s_1\rangle, \ldots, \langle w_n, s_n\rangle$, where $\langle w_k, s_k\rangle$ represents the word w_k and its SuperARV assignment s_k. The parse probability for each sequence is initialized to the tag assignment probability. Each sequence is then stored on the stack ranked in non-increasing order by this probability.

During the second step of parsing (item 2 in figure 10.8), the modifiees are statistically specified in a left-to-right manner. Modifiee specification is conducted on each tag assignment in the stack, which maintains the record of all partial CDG parses determined for the sentence prefix. Note that the algorithm utilizes modifiee lexical category constraints to filter out candidates with mismatched lexical categories. When processing the word $w_k, k = 1, \ldots, n$, the algorithm first attempts to determine the left dependents of w_k from the closest to the farthest (see step a in item 2). The dependency assignment probability when choosing the $(c+1)^{\text{th}}$ left dependent (with its position denoted $dep(k, -(c+1))$) of w_k, s_k is defined as:

$$P(\text{link}(s_{dep(k,-(c+1))}, s_k, -(c+1)|syn, \mathcal{H})),$$

where $\mathcal{H} = \langle w, s\rangle_k, \langle w, s\rangle_{dep(k,-(c+1))}, \langle w, s\rangle_{dep(k,-1)}^{dep(k,-c)}$. The dependency assignment probability is conditioned on the word identity and SuperARV assignment to w_k and $w_{dep(k,-(c+1))}$, as well as all the c previously chosen left dependents $\langle w, s\rangle_{dep(k,-1)}^{dep(k,-c)}$ for w_k. After the algorithm statistically specifies the left dependents for w_k, it then

determines whether w_k could be the $(d+1)^{\text{th}}$ right dependent of a previously seen word $w_p, p = 1, \ldots, k-1$ (where d denotes the number of already assigned right dependents of w_p), as shown in step b in item 2 of figure 10.8. After processing word w_k in each partial parse on the stack, the partial parses are reranked according to their updated probabilities (see the last step under item 2). Step 2 is iterated until all the words in the sentence are processed. The top parse in the stack is then selected as the parse for the sentence.

TABLE 10.1 Definitions of symbols used in the basic parsing algorithm.

Term	**Denotes**
$\mathcal{L}(s_k)$, $\mathcal{R}(s_k)$	all dependents of s_k to the left and right of w_k, respectively
$N(\mathcal{L}(s_k))$, $N(\mathcal{R}(s_k))$	the number of left and right dependents of s_k, respectively
$dep(k,-c)$, $dep(k,c)$	c^{th} left dependent and right dependent of s_k, respectively
$dep(k,-1)$, $dep(k,1)$	the position of the closest left dependent and right dependent of s_k, respectively
$dep(k,-N(\mathcal{L}(s_k)))$ $dep(k,N(\mathcal{R}(s_k)))$	the position of the farthest left dependent and right dependent of s_k, respectively
$\text{Cat}(s_k)$	the lexical category of s_k
$\text{ModCat}(s_k,-c)$ $\text{ModCat}(s_k,c)$	the lexical category of s_k's c^{th} left and right dependent (encoded in the SuperARV structure), respectively
$\text{link}(s_i,s_j,k)$	the dependency relation between SuperARV s_i and s_j with w_i assigned as the k^{th} dependent of s_j, e.g., $\text{link}(s_{dep(k,-(c+1))},s_k,-(c+1))$ indicates that $w_{dep(k,-(c+1))}$ is the $(c+1)^{\text{th}}$ left dependent of s_k
$D(\mathcal{L}(s_k))$, $D(\mathcal{R}(s_k)))$	the number of left and right dependents of s_k already assigned, respectively
$\langle w,s\rangle_{dep(k,-1)}^{dep(k,-c)}$	words and SuperARVs of s_k's closest left dependent up to its c^{th} left dependent
$\langle w,s\rangle_{dep(k,1)}^{dep(k,c)}$	words and SuperARVs of s_k's closest right dependent up to its c^{th} right dependent
syn	a random variable denoting the synergistic relation between some dependents

CDG differs from traditional dependency grammars in that CDG utilizes multiple roles to specify symmetric grammatical dependencies that are necessary for grammaticality. For example, the verb-object relationship involves two role value assignments in CDG: a governor dependency from the object to the verb and a need dependency based on subcategorization from the verb to the object. Hence, if a noun w_i is dependent on a verb w_j as an object, then the expectation of the verb w_j for an object should be simultaneously satisfied by w_i. In our model, we use a Boolean random variable *syn* to capture the synergistic relationship between certain role value pairs. This mechanism allows us to elevate, for example, the probability that the subject of a sentence w_i is governed by a tensed verb w_j when the need role value of w_j points to w_i as its subject. Note that this is a much simpler mechanism

Basic Parsing Algorithm

1. Using SuperARV tagging on word sequence $w_1, \ldots, w_n$, obtain a set of n-best SuperARV sequences with each element consisting of n (word, SuperARV) tuples, denoted $\langle w_1, s_1\rangle, \ldots, \langle w_n, s_n\rangle$, which we will call an assignment. Initialize the stack of parse prefixes with these assignments, ranked in nonincreasing order by their probability.
2. /* From left-to-right, process each $\langle word, tag\rangle$ of the assignment and generate parse prefixes. */
   for $k := 1$ to n do
      for each item on the stack
         /* Step a: */
         /* Decide left dependents of $\langle w_k, s_k\rangle$ from the nearest to the farthest. */
         for $c := 0$ to $N(\mathcal{L}(s_k)) - 1$ do
            /* Choose a position for the $(c+1)^{\text{th}}{}^{left}$ dependent of $\langle w_k, s_k\rangle$ from the set possible positions $\mathcal{C} = \{1, \ldots, dep(k, -c) - 1\}$. The position choice is denoted $dep(k, -(c+1))$. */
            /* In the following equations, different left dependent assignments will generate different parse prefixes, each of which is stored in the stack. */
            for each $dep(k, -(c+1))$ from positions $\mathcal{C} = \{1, \ldots, dep(k, -c) - 1\}$
               /* Check whether the lexical category of the choice matches the modifiee lexical category of the $(c+1)^{\text{th}}$left dependent of $\langle w_k, s_k\rangle$. */
               if $\text{Cat}(s_{dep(k,-(c+1))}) == \text{ModCat}(s_k, -(c+1))$ then
               $P(D) := P(D) \times P(\text{link}(s_{dep(k,-(c+1))}, s_k, -(c+1)|syn, \mathcal{H}))$
               where $\mathcal{H} = \langle w, s\rangle_k, \langle w, s\rangle_{dep(k,-(c+1))}, \langle w, s\rangle_{dep(k,-1)}^{dep(k,-c)}$
         /* End of choosing left dependents of $\langle w_k, s_k\rangle$ for this parse prefix. */
         /* Step b: */
         /* For the word/tag pair $\langle w_k, s_k\rangle$, check whether it could be a right dependent of any previously seen word within a parse prefix of $\langle w_1, s_1\rangle, \ldots, \langle w_{k-1}, s_{k-1}\rangle$. */
         for $p := 1, k-1$ do
            /* If $\langle w_p, s_p\rangle$ still has right dependents left unspecified, then try out $\langle w_k, s_k\rangle$ as a right dependent. */
            if $D(\mathcal{R}(s_p)) \neq N(\mathcal{R}(s_p))$ then
               $d := D(\mathcal{R}(s_p))$
               /* If the lexical category of $\langle w_k, s_k\rangle$ matches the modifiee lexical category of the$(d+1)^{\text{th}}$ right dependent of $\langle w_p, s_p\rangle$, then s_k might be $\langle w_p, s_p\rangle$'s $(d+1)^{\text{th}}$right dependent. */
               if $\text{Cat}(s_k) == \text{ModCat}(s_p, d+1)$ then
               $P(D) := P(D) \times P(\text{link}(s_k, s_p, d+1)|syn, \mathcal{H})$,
               where $\mathcal{H} = \langle w, s\rangle_p, \langle w, s\rangle_k, \langle w, s\rangle_{dep(p,1)}^{dep(p,d)}$
         Sort the parse prefixes in the stack according to $logPr(D)$ and apply pruning using the thresholds.
3. After processing $w_1, \ldots, w_n$, pick the parse with the highest $logPr(T)$ in the stack as the parse for that sentence.

FIGURE 10.8 The basic loosely coupled parsing algorithm.

than the *trigger* approach used by Rosenfeld (2000) to model synergistic features. The values of *syn* for a dependency relation were determined heuristically based on the lexical category, role name, and label information of the two dependent words (Wang, 2003; Wang and Harper, 2004).

For the tightly coupled parser, the first step of SuperARV tagging is removed, and the SuperARV assignment for each word and the corresponding decisions on specifying its modifiees are integrated into a single step that updates the conditional probabilities for modifiee specification and adds each SuperARV possibility to the conditions for dependency assignment probability estimations. For both versions, the parsing algorithm is implemented as a simple best-first search using a stack to store partial parses.

To control time and memory complexity, we used two pruning thresholds: maximum stack depth (maximum number of partial parses allowed in the stack) and maximum difference between the log probabilities of the top and bottom partial parses in the stack. The latter is defined as a percentage, that is, the bottom partial parse cannot have its probability lower than $\sigma\%$ of that of the top partial parse (a negative value due to the use of log probabilities). These two pruning thresholds were tuned based on the trade-off between time and memory complexity and parsing accuracy on a heldout set, and they both had hard limits.

Note that the maximum likelihood estimation of the dependency assignment probabilities in the basic loosely coupled parsing algorithm presented in figure 10.8 is likely to suffer from data sparsity, and the estimates for the tightly coupled algorithm are likely to suffer even more so. Hence, we smooth the probabilities using Jelinek-Mercer smoothing (Jelinek, 1997). To simplify the presentation, we describe the smoothing procedure for the loosely coupled case, where the + prefix represents items related to right dependents and the − prefix, left dependents:

$$
\begin{aligned}
&P(\text{link}(s_{dep(k,\pm(c+1))}, s_k, \pm(c+1))|syn, \mathcal{H}) \\
&= \sum_{i=1}^{c} \eta_i \cdot P(\text{link}(s_{dep(k,\pm(c+1))}, s_k, \pm(c+1))|syn, \delta_i) \\
&+ \lambda_1 \cdot P(\text{link}(s_{dep(k,\pm(c+1))}, s_k, \pm(c+1))|syn, \mathcal{H}_1) \\
&+ \lambda_2 \cdot P(\text{link}(s_{dep(k,\pm(c+1))}, s_k, \pm(c+1))|syn, \mathcal{H}_2) \\
&+ \lambda_3 \cdot P(\text{link}(s_{dep(k,\pm(c+1))}, s_k, \pm(c+1))|syn, \mathcal{H}_3) \\
&+ \lambda_4 \cdot P(\text{link}(s_{dep(k,\pm(c+1))}, s_k, \pm(c+1))|syn, \mathcal{H}_4) \\
&+ \lambda_5 \cdot P(\text{link}(s_{dep(k,\pm(c+1))}, s_k, \pm(c+1))|syn, \mathcal{H}_5) \\
&+ \lambda_6 \cdot P(\text{link}(s_{dep(k,\pm(c+1))}, s_k, \pm(c+1))|syn, \mathcal{H}_6),
\end{aligned}
$$

where:

$$\mathcal{H} = \langle w, s\rangle_k, \langle w, s\rangle_{dep(k,\pm(c+1))}, \langle w, s\rangle_{dep(k,\pm1)}^{dep(k,\pm c)}$$
$$\delta_i = \langle w, s\rangle_k, \langle w, s\rangle_{dep(k,\pm(c+1))}, \langle w, s\rangle_{\pm1}^{\pm i}$$
$$\mathcal{H}_1 = \langle w, s\rangle_k, w_{dep(k,\pm(c+1))}, s_{dep(k,\pm1)}^{dep(k,\pm c)}$$
$$\mathcal{H}_2 = \langle w, s\rangle_k, s_{dep(k,\pm(c+1))}, s_{dep(k,\pm1)}^{dep(k,\pm c)}$$
$$\mathcal{H}_3 = s_k, \langle w, s\rangle_{dep(k,\pm(c+1))}, s_{dep(k,\pm1)}^{dep(k,\pm c)}$$
$$\mathcal{H}_4 = s_k, s_{dep(k,\pm(c+1))}, s_{dep(k,\pm1)}^{dep(k,\pm c)}$$
$$\mathcal{H}_5 = s_k, s_{dep(k,\pm(c+1))}, POS_{dep(k,\pm1)}^{dep(k,\pm c)}$$
$$\mathcal{H}_6 = POS_k, POS_{dep(k,\pm(c+1))}, POS_{dep(k,\pm1)}^{dep(k,\pm c)}.$$

Note that POS_k denotes the part of speech of word w_k. The η_i and λ values are determined using the EM algorithm described by Jelinek (1990) on a held-out data set.

Additions to the basic model

Next, we describe a set of additions to the basic model presented in section 10.4.2 that should help improve parsing accuracy.

- **Modeling crossing dependencies:** The basic parsing algorithm was implemented to preclude crossing dependencies; however, it is important to allow them in order to model certain *wh*-movement structures in our model. Hence, we investigated the impact of modeling crossing dependencies during parsing, relying on the dependency statistics of the SuperARVs to determine when they are allowed.
- **Distance between dependents:** Distance between two dependent words (e.g., the distance value 0 means the two words are adjacent) is an important factor in determining the modifiees of a word. Collins found that for the Wall Street Journal Penn Treebank (WSJ PTB) corpus, 74.2% of the dependent words of a particular word abut that word, 86.3% of the dependencies are within a trigram window, and 95.6% of the dependencies are within a 5-gram window (Collins, 1996). Since distance plays an important role in statistically determining the modifiees of a word, we evaluated an alternative model that includes distance by adding $\Delta_{dep(k,\pm(c+1)),k}$ to $\mathcal{H}$. To avoid data sparsity problems, distance was bucketed as: [0], [1], [2, 4], [5, 9], [10, ∞), that is, a discrete random variable with five possible ranges was used to model distance.
- **Barriers between dependents:** Punctuation and verbs provide useful information for restricting dependencies. For example, words do not tend to modify words to the left of a verb they follow unless there is *wh*-movement or to modify words on the other side of a punctuation mark unless there is an apposition or a parenthetical comment. Hence, we added several boolean random variables to our model representing questions related to these barriers as conditions to the conditional dependency assignment probability estimations (as in Collins, 1996). This allows us to capture some of the binary constraints that were part of our hand-developed CDG.
  1. Is there a verb between the dependencies?

2. Are there 0, 1, 2, or more commas or colons between the dependents?
3. Is there a comma immediately following the first of two dependents, or is there a comma immediately preceding the second of the two dependents?

- **Additional modifiee constraints:** In the basic model, the modifiee constraints of the SuperARV were limited to the lexical category of the dependent. In our previous work (Harper and Wang, 2001), we found that modifiee lexical features play an important role in strengthening selectivity of a CDG; however, for many dependencies, only a subset of the possible lexical features are relevant to the dependency relation. For example, a noun's number feature is often irrelevant in many of its grammatical roles (e.g., as a possessive noun) and yet it is a very important constraint when the noun acts as a subject of a sentence with a present-tensed verb. Since the knowledge-based feature relaxation approach discussed in section 10.3 provided a good trade-off between grammar generality and selectivity, we have utilized a knowledge-based method to incorporate selective relevant lexical features into modifiee constraints of a SuperARV structure (Wang, 2003).

Parsing evaluation

In order to train an SCDG parser, we had to either identify or create corpora annotated with CDG parse relations. Because of lack of availability of CDG annotated treebanks, we chose to convert an existing CFG annotated corpus, namely, the Wall Street Journal Penn Treebank (WSJ PTB) into CDG annotations. Dependency structures were generated by headword percolation together with a rule-based method to determine lexical features and need role values for words; details appear in Wang (2003). Note that the soundness of the CFG-to-CDG transformer was evaluated by examining a subset of the CDG parses generated from the transformer on the WSJ PTB development set to ensure that they were correct. The SCDG parser was evaluated using this converted treebank with the traditional data setup (that is, sections 02–21 for training, section 23 for testing, and section 24 for development). As in Charniak (2000), Collins (1999), and Ratnaparkhi (1999), it was evaluated on all sentences with length ≤ 40 words (2,245 sentences) and length ≤ 100 words (2,416 sentences).

Since our parser was trained using a CFG-to-CDG transformer (Wang, 2003), which maps a CFG parse tree to a unique CDG parse, we chose to evaluate our parser's accuracy using "gold standard" CDG parse relations. The alternative would be to map back to a CFG tree; however, this mapping would add noise to our evaluation. In addition, when there are crossing dependencies, then no tree can be generated for that set of dependencies. Consequently, we defined a dependency-based metric adapted from Eisner (1996), namely role value labeled precision (RLP) and role value labeled recall (RLR):

$$RLP = \frac{\text{correct modifiee assignments}}{\text{number of modifiees our parser found}}$$
$$RLR = \frac{\text{correct modifiee assignments}}{\text{number of modifiees in the gold test set parses}},$$

where a modifiee assignment for a word w_i in the test parse for a sentence is correct only when the three-tuple $\langle$role id, role label, modifiee word position$\rangle$ (that is, a role

TABLE 10.2 Results on section 23 of the WSJ PTB for four loosely coupled model variations. The evaluation metrics, RLR and RLP, are our dependency-based role value labeled recall and precision.

Models	**Configuration**	**≤ 40 words (2,245 sentences)**				
		Tagging	**governor only**		**all roles**	
		Acc.	**RLP**	**RLR**	**RLP**	**RLR**
(1)	basic model	94.7	90.6	90.3	86.8	86.2
(2)	(1)+crossing dependencies	95.0	90.7	90.5	87.0	86.5
(3)	(2)+distance and barrier model	95.7	91.1	90.9	87.4	87.0
(4)	(3)+modifiee lexical features	96.2	91.5	91.2	88.0	87.4
Models	**Configuration**	**≤ 100 words (2,416 sentences)**				
		Tagging	**governor only**		**all roles**	
		Acc.	**RLP**	**RLR**	**RLP**	**RLR**
(1)	basic model	94.0	89.7	89.3	86.0	85.5
(2)	(1)+crossing dependencies	94.2	89.9	89.6	86.2	85.8
(3)	(2)+distance and barrier model	94.7	90.4	90.2	86.8	86.3
(4)	(3)+modifiee lexical features	95.4	90.9	90.5	87.5	86.8

value) for w_i is the same as the three-tuple role value for the corresponding role id of w_i in the gold test parse.

In a first experiment (Wang, 2003; Wang and Harper, 2004), we evaluated the basic loosely coupled model that uses a trigram SuperARV tagger to generate 40-best SuperARV sequences prior to modifiee specification and examined the impact of the model improvements described previously on this loosely coupled model. Table 10.2 shows the results for each of the models, including SuperARV tagging accuracy (%) and parsing accuracy (%) based on RLP and RLR. Allowing crossing dependencies improves the overall parsing accuracy, but using distance information together with barrier heuristics produces an even greater improvement especially on the longer sentences. The accuracy is further improved by adding additional modifiee lexical feature constraints to the SuperARVs to enforce feature constraints on dependencies, highlighting the importance of selective enhancement of grammar specificity. Overall, using a distance model and modifiee lexical features provides the greatest improvement. Note that RLR is lower than RLP in these investigations; this is possibly due to errors in SuperARV tagging and the use of a tight stack pruning threshold.

In a second experiment (Wang, 2003; Wang and Harper, 2004), we evaluated the impact of increasing the context of the SuperARV tagger (3-gram to 4-gram) and increasing the length of the n-best list (40-best to 100-best) passed from the tagger to the modifiee specification step of our best loosely coupled model (that is, with all of the model enhancements). We also evaluated a model that tightly integrates SuperARV prediction with modifiee specification. As can be seen in Table 10.3, a stronger SuperARV tagger and a larger search space of SuperARV sequences

TABLE 10.3 Results on Section 23 of the WSJ PTB comparing models that utilize different SuperARV taggers and n-best sizes with the tightly coupled implementation.

Models		**≤ 40 words (2,245 sentences)**				
		Tagging	**governor only**		**all roles**	
		Acc.	**RLP**	**RLR**	**RLP**	**RLR**
Loose Basic	trigram, 40-best	94.7	90.6	90.3	86.8	86.2
Loose Plus	trigram, 40-best	96.2	91.5	91.2	88.0	87.4
	trigram, 100-best	96.7	91.9	91.5	88.3	87.7
	4-gram, 40-best	96.9	92.2	91.7	88.6	88.1
	4-gram, 100-best	97.2	92.4	92.3	89.1	88.6
Tight		97.4	93.2	92.9	89.8	89.2
Models		**≤ 100 words (2,416 sentences)**				
		Tagging	**governor only**		**all roles**	
		Acc.	**RLP**	**RLR**	**RLP**	**RLR**
Loose Basic	trigram, 40-best	94.0	89.7	89.3	86.0	85.5
Loose Plus	trigram, 40-best	95.4	90.9	90.5	87.5	86.8
	trigram, 100-best	95.8	91.3	90.8	87.7	87.0
	4-gram, 40-best	96.0	91.7	91.2	88.0	87.4
	4-gram, 100-best	96.3	91.8	91.5	88.5	87.8
Tight		96.6	92.6	92.2	89.1	88.5

produces improvements in parse accuracy; however, the best results were achieved by the tightly integrated model. Note that SuperARV tagging accuracy and parse accuracy improve in tandem, consistent with the observations of Collins (1999) and Eisner (1996).

Given this framework of tightly integrated, multiple knowledge sources, with special attention to modeling distance, barriers, and modifiee constraints, we have achieved a solid parsing accuracy on the WSJ PTB corpus. We have also compared the SCDG parser to other statistical parsing algorithms (Wang, 2003; Wang and Harper, 2004) and found that it performed similarly to and in some cases better than those models using our dependency metric. We have also evaluated the parser on the Switchboard corpus (Wang, 2003) and obtained substantially lower recall and precision results. It should be noted that the Switchboard Treebank is quite small for training an SCDG parsing model, and the corpus contains many typos that could seriously degrade the quality of the training data produced by the CFG-to-CDG conversion process. In addition, the SCDG parser that was used did not directly model some of the more challenging phenomena that occur in spontaneous speech (e.g., speech repairs).

10.4.3 Language modeling

We have also developed and evaluated two types of CDG-based language models (LMs) that we will summarize here, an almost-parsing LM and a full parser-based LM. It should be noted that the design of these LMs gained significantly from the insight obtained from our initial CDG grammar induction experiments. The SuperARV almost-parsing LM is trained, like the SCDG parser, using information derived from a corpus of CDG parses, but unlike the parser, it does not construct a full parse, which should help make the model more robust and more efficient than a full parser

model. The full CDG parser-based LM utilizes complete parse information obtained by adding the modifiee links to the SuperARVs assigned to each word in a sentence in order to capture important long-distance dependency constraints. Clearly there is a synergy between these two language models and our SCDG parser. Indeed, we have found that the insights that improve one usually positively impact the others.

The SuperARV LM

The SuperARV language model is implemented using a joint probabilistic model framework so that word form information is tightly integrated at the model level. Without loss of generality, given a word sequence w_1^N and their SuperARV tags t_1^N, a trigram version of the SuperARV language model is:

$$\begin{aligned} P(w_1^N t_1^N) &= \prod_{i=1}^{N} P(w_i t_i \mid w_1^{i-1} t_1^{i-1}) \\ &= \prod_{i=1}^{N} P(t_i \mid w_1^{i-1} t_1^{i-1}) \cdot P(w_i \mid w_1^{i-1} t_1^{i}) \\ &\approx \prod_{i=1}^{N} P(t_i \mid w_{i-2}^{i-1} t_{i-2}^{i-1}) \cdot P(w_i \mid w_{i-2}^{i-1} t_{i-2}^{i}) \end{aligned} \tag{10.2}$$

Note that the model does not encode word identity directly at the data structure level since this could dramatically increase the number of parameters and cause serious data sparsity problems. Instead, the SuperARV language model combines the knowledge of word identity with SuperARVs at the model level by the joint prediction shown in equation (10.2). A SuperARV class-based language model leads to a linguistically compact model that not only makes the use of higher order n-grams feasible but also enables the joint modeling of lexical features and syntactic constraints. Because a word can have multiple SuperARVs, this class-based language model uses soft class membership.

SuperARVs are accumulated from a corpus annotated with CDG relations and associated words, so we can learn their joint frequency of occurrence. As discussed previously, the CDG treebanks used in this work were derived from CFG trees (Wang, 2003). To estimate the probability distributions in equation (10.2), we used recursive linear interpolation among probability estimations of different orders as discussed in Wang (2003) and Wang and Harper (2002), with the best performance among smoothing algorithms achieved using the modified Kneser-Ney smoothing algorithm (Chen and Goodman, 1998). Table 10.4 enumerates the n-grams utilized and their interpolation order for smoothing the two distributions. The SuperARV LM hypothesizes categories for out-of-vocabulary words using the leave-one-out technique (Niesler and Woodland, 1996). The SuperARV LM is most closely related to the almost-parsing-based LM developed by Srinivas (1997). By comparison, the SuperARV LM incorporates direct word dependencies, more lexical feature information than the supertag LM, uses joint instead of conditional probability estimations, and uses modified Kneser-Ney rather than Katz smoothing.

We conducted a variety of experiments using this model, and some of the most interesting results are summarized as follows:

- We compared the perplexity of the joint probabilistic trigram SuperARV LM (as in Heeman, 1998) on the WSJ PTB task (Wang, 2003; Wang and

TABLE 10.4 The enumeration and order of n-grams for smoothing the distributions in equation (10.2).

n-**grams**	$\hat{P}(t_i \mid w_{i-2}^{i-1} t_{i-2}^{i-1})$	$\hat{P}(w_i \mid w_{i-2}^{i-1} t_{i-2}^{i})$
highest	$\hat{P}(t_i \mid w_{i-2}^{i-1} t_{i-2}^{i-1})$	$\hat{P}(w_i \mid w_{i-2}^{i-1} t_{i-2}^{i})$
	$\hat{P}(t_i \mid w_{i-1} t_{i-2}^{i-1})$	$\hat{P}(w_i \mid w_{i-2}^{i-1} t_{i-1}^{i})$
	$\hat{P}(t_i \mid w_{i-2}^{i-1} t_{i-1})$	$\hat{P}(w_i \mid w_{i-1} t_{i-2}^{i})$
	$\hat{P}(t_i \mid t_{i-2}^{i-1})$	$\hat{P}(w_i \mid w_{i-1} t_{i-1}^{i})$
	$\hat{P}(t_i \mid w_{i-1} t_{i-1})$	$\hat{P}(w_i \mid w_{i-1} t_i)$
	$\hat{P}(t_i \mid t_{i-1})$	$\hat{P}(w_i \mid t_{i-1}^{i})$
lowest	$\hat{P}(t_i)$	$\hat{P}(w_i \mid t_i)$

Harper, 2002) to a trigram word-based LM, a joint part-of-speech class-based LM, a conditional trigram SuperARV LM, and the parser-based LMs of Chelba (2000), Xu et al. (2002), Roark (2001), and Charniak (2001). The joint probabilistic SuperARV LM achieved a lower perplexity than all of the models. Although the parser-based models benefitted from interpolation with the word-based trigram, the joint SuperARV LM did not. In fact, we found that since the SuperARV LM is highly correlated with the word-based LM, unless the word model is based on additional materials not used to train the SuperARV LM, then the SuperARV LM does not benefit from interpolation with that word-based model. The conditional version of the SuperARV LM is less correlated with the word-based LM and so benefited from interpolation; however, the interpolated model was unable to achieve the performance of the joint SuperARV LM. Similar results were found for the DARPA WSJ continuous speech recognition task (WSJ CSR) (Wang, 2003; Wang and Harper, 2002).

- The rescoring effectiveness of our SuperARV LM was compared to several other LMs on the 1993 20K open vocabulary DARPA Hub1 WSJ CSR lattices (denoted 93–20K) used in Chelba (2000). The word error rate (WER) was 12.83% for the SuperARV LM, 13.72% for the trigram word-based LM, 13.51% for the joint part-of-speech (POS) class-based LM, 12.89% for Chelba's parser-based LM (Chelba, 2000), and 12.3% for Xu and Chelba's model (Xu et al., 2002). The SuperARV LM was superior to the word-based and POS LMs and performed similarly to Chelba's parser-based model, but did not perform as well as Xu and Chelba's LM, which utilized substantially more context than our LM. We will discuss our full parser based LM in section 10.4.3.
- The tight integration of lexical category, lexical features, and structural dependency constraints is important to the SuperARV LM's performance. We found in experiments in which we relax the knowledge sources that the performance of the LM decreases (that is, perplexity and WER increase) (Wang, 2003; Wang and Harper, 2002). See the perplexity results in Table 10.5 in which the indicated types of information were relaxed, that is, modifiee constraints (-m), need roles (-n), lexical category (-c), lexical features (-f), and label (-L) in the role value assignment from the full SuperARV LM (denoted SARV in the table). Clearly, the fully constrained SuperARV LM achieved the lowest perplexity, suggesting that each of these knowledge sources provides important information to the model such that if any of it is relaxed, the model performs less well. It is interesting that

TABLE 10.5 Comparing perplexity results for each language model on the WSJ CSR test sets. The language models appear in decreasing order of perplexity.

language model	92–5K	93–5K	92–20K	93–20K
3gram	45.61	50.51	106.52	109.22
POS	44.21	30.26	98.79	96.64
SARV-fmn	45.01	27.42	96.23	93.16
SARV-fm	43.42	27.22	95.10	91.33
SARV-f	42.33	27.06	94.87	90.20
SARV-mn	40.38	26.96	90.23	89.54
SARV-n	35.02	26.08	87.32	88.04
SARV-L	28.76	25.71	82.45	84.82
SARV-m	26.86	25.58	80.24	83.12
SARV	**21.35**	**23.42**	**69.95**	**74.22**

ignoring lexical features produced an even greater increase in perplexity than eliminating the need roles and the modifiee constraints. Similar results were found when rescoring WSJ CSR lattices.

- We investigated the use of the SuperARV LM on the DARPA Hub4 (Broadcast News) CSR task (Wang, 2003; Wang et al., 2003). The challenge for building a SuperARV LM for this domain was that, in contrast to the DARPA WSJ CSR task, there was no corpus of CFG parse trees available for deriving CDG annotations for training. To circumvent this problem to some extent, we generated a set of training parse trees by using available probabilistic parsers (Charniak, 2000; Collins, 1999), and then converted the trees into CDG annotations. Fortunately, much of the Hub4 training data corresponds to planned speech that is similar to the material that appears in the WSJ PTB, and so we were able to train the parsers using that treebank. The language model training data sources for the SuperARV and baseline LMs consisted of two sets of Hub4 training data, that is, the 130 million (M) word loosely transcribed Broadcast News corpus for language model training and the 380,000 word closely transcribed material for acoustic training, and two sets of non-Hub4 training data, that is, the North American Business News (NABN) corpus and the Switchboard-I corpus (Stolcke et al., 2003); the former was chosen for additional business and politics coverage, and the latter for its conversational speech characteristics.
  A baseline word-based LM, with a vocabulary size of 48K, was constructed for the Hub4 task by interpolating the 5-gram word-based LMs for each Hub4 training set together with the trigram LMs for each non-Hub4 training set. Each LM was estimated by the SRILM toolkit (Stolcke, 2002) using the modified Kneser-Ney smoothing algorithm. A 4-gram SuperARV LM was trained using an automatically constructed CDG treebank for each data set, and parameter tuning was done on a heldout data set of 20% of the training data. Since spontaneous speech characteristics (such as disfluencies) are not well represented in the training data for the SuperARV LM, we also interpolated the SuperARV LM with the baseline LM.
  The language models were evaluated by rescoring n-best lists (of up to 2,000 hypotheses per utterance), generated by SRI's 1997 Broadcast News System (Sankar et al., 1998) for the 1996 Hub4 development test set using a log

linear combination of two types of acoustic models (crossword and within-word triphones), word insertion penalty, and language model. The WER was computed for the hypotheses with the highest combined scores. The WER for the SuperARV LM alone was 26.1% compared to 26.9% for the baseline LM. When the SuperARV and word-based LMs were interpolated, the WER dropped slightly to 26.0% (possibly because that spontaneous speech characteristics such as disfluencies were not accurately represented in the training data for the SuperARV LM). This result is exciting because these levels of performance were obtained using a potentially errorful method to automatically construct the needed treebanks for the language model training data. It should be noted that efforts to generate more consistent and accurate treebanks also contributed to LM performance (Wang, 2003; Wang et al., 2003).

- We also investigated the use of the SuperARV LM on the Hub5 CSR task (Wang, 2003; Wang et al., 2004) using an approach quite similar to that used for Hub4. To create training material for the SuperARV LM, we parsed the sentences in the LM training data using available probabilistic parsers (Charniak, 2000; Collins, 1999) to generate CFG parse trees and then transformed those trees to CDG parses.

  The baseline and SuperARV LMs were trained on the acoustic training transcripts (that is, Switchboard-1 corpus, the Credit Card corpus, the CallHome English corpus, transcribed Switchboard Cellular data released by LDC, and the Switchboard-2 data transcribed by CTRAN and released by BBN, for a total of 418 hours of speech), as well as the 1996 Broadcast News Hub4 LM training corpus (130M words). An additional 191M words of LM training data were retrieved from the Web through the Google search engine by searching for conversational n-grams extracted from the conversational telephone speech (CTS) transcripts (Bulyko et al., 2003). Furthermore, 102M words of data relevant to the topics of the Switchboard-2 and LDC Fisher data collections were selected from Google newsgroups, in an attempt to better match the unseen CTS test data drawn from those collections (Stolcke et al., 2003).

  We evaluated the SuperARV LM in the context of the SRI 2003 CTS system (Stolcke et al., 2003). Baseline language models of increasing orders were used for initial decoding and lattice generation, lattice expansion and rescoring, and finally n-best rescoring. The system employed computationally inexpensive decoding steps first to generate intermediate results that constrain the search space, followed by rescoring with more sophisticated acoustic and language models. A bigram class-conditioned mixture LM (Bulyko et al., 2003) was used during the first pass of acoustic decoding, and a trigram class-conditioned mixture LM was used for lattice expansion. A 4-gram class-conditioned mixture LM was employed for rescoring n-best hypotheses. For the baseline 2-gram, 3-gram, and 4-gram LMs, separate LMs were built for each data source; all source-specific LMs, with the exception of the web-topic LM, were then combined by class-conditioned interpolation (Bulyko et al., 2003), and the resulting mixture LM was interpolated with the topic-related LM using fixed weighting of $(0.8, 0.2)$. The interpolation of word n-gram was static and resulted in a single combined backoff n-gram model, as described in Stolcke (2002).

  To train the SuperARV LM, similarly to our procedure on the Hub4 Broadcast News CSR task, we parsed the sentences in the LM training data

to generate their CFG parse trees and then transformed the trees to CDG parses. Just as for the baseline LM, a separate SuperARV 4-gram LM was trained for each available source and the resulting source-specific LMs were then combined into a single model, with the weights obtained by minimizing the perplexity on a held-out development set. To enable efficient integration of the SuperARV LM into the recognition system, we generated an ARPA-style backoff LM based on the SuperARV word probability estimates. Note that the SuperARV language model, as a class-based LM, is theoretically able to estimate probabilities for any word sequence; however, to keep the generated word LM to a reasonable size, n-gram pruning similar to Stolcke (1998) is applied. The pruning threshold was tuned on a development set to achieve a satisfactory balance between LM size and perplexity. The word 4-gram SuperARV LM thus obtained was used in the n-best rescoring stages of our system. As an expedient to leverage the SuperARV LM in the earlier stages of the recognition system, we used the "LM rescoring" feature of the SRILM toolkit (Stolcke, 2002). We replaced bigram and trigram probability estimates in the baseline LMs (for the initial decoding and lattice expansion stages) with SuperARV LM probability estimates (backing off as needed for lack of full 4-gram word contexts). Following this replacement, backoff weights were recomputed to normalize the LM.

Comparing the SRI DARPA RT-03 Spring CTS system (which used the baseline LMs) with a modified system that uses the SuperARV LM on the DARPA RT-02 CTS evaluation set, we found that the use of the SuperARV LM provided a significant reduction in word error rate, reducing the final result on the complete RT-02 set (that is, tuning and test sets) by 1.6% (from 25.8% to 24.2%). Note that later stages benefited from both the improved LM and the better quality of adaptation hypotheses in earlier stages. By converting the SuperARV LM into the standard word n-gram form, no additional computational effort was incurred at recognition time, so the model could be used in all stages of a multipass CSR system, giving 6.2% relative WER reduction on a standard CTS recognition task.

The SCDG parser-based LM

Like other parser-based LMs, an SCDG parser LM estimates a string's probability by summing the probabilities of all CDG parses for the string produced by the parser. Given the set of partial CDG parses for a sentence prefix w_1^{i-1} is D_1^{i-1}, the probability $P(w_i|w_1^{i-1})$ is calculated as:

$$P_{\text{SCDG}}(w_i|w_1^{i-1}) = \frac{\sum_{d\in D_1^i} P(d)}{\sum_{d'\in D_1^{i-1}} P(d')} \tag{10.3}$$

where $P(d)$ is the probability of a partial CDG parse d for the sentence prefix w_1^i in question. In section 10.4.2, we described the implementation of the underlying SCDG parser, where we already discussed, in particular, how to calculate $P(d)$ in equation (10.3).

The SCDG parser-based LM was evaluated on the 1993 20K open vocabulary DARPA Hub1 WSJ CSR task in order to compare to other parser-based LMs on this task (Chelba, 2000; Roark, 2001; Xu et al., 2002). The LM training data for this task was composed of the 1987–1989 files containing 37,243,300 words. Since this is a speech corpus, this material contains no punctuation or case information. All words

outside the provided vocabulary were mapped to $\langle UNK \rangle$. We evaluated all LMs on the 1993 20K open vocabulary DARPA WSJ CSR evaluation set (denoted 93–20K), which consists of 213 utterances and 3,446 words. For consistency, we rescored the same lattices as those used by Chelba (2000), Roark (2001), and Xu et al. (2002). Note that, for the parser-based LMs, we split the contractions in the word lattices so that they could be processed properly. The SCDG parser LM was trained using constraint dependency grammar parses for the complete (that is, 37+ M word) WSJ CSR LM training data, obtained using the CFG-to-CDG transformer (as in Wang and Harper, 2002 and Wang, 2003) to transform the CFG parses from the WSJ Penn Treebank (Marcus et al., 1993) and the BLLIP Treebank (Charniak et al., 2000), which together cover the entire LM training set.

We compared several variations of our SCDG parser LM to a variety of other LMs: the baseline trigram provided by LDC for the task, the joint probabilistic part-of-speech LM (Wang and Harper, 2002), the best SuperARV LM described in Wang and Harper (2002), Chelba's (2000) parser-based LM (trained on a 20M word subset of the WSJ CSR LM training data), Chelba's parser-based LM retrained using the entire training set (Wang and Harper, 2002), and Xu and Chelba's (Xu et al., 2002) revised parser-based LM (trained on a 20M word subset of the WSJ CSR LM training data due to time and space constraints on the model). All models were trained on the complete 37+ M word LM training data using the 20K open vocabulary, unless otherwise indicated. Note that we did not compare to Roark's model (2001) since it was trained on the much smaller WSJ PTB corpus. Note that (Xu et al., 2002) rescored 50-best lists generated from Chelba's (2000) lattices, whereas, Chelba (2000) performed lattice rescoring.

Models		**WER(SAC) (%)**	**Interp. Weight**
3gram		13.72(36.18)	N/A
POS		13.51(37.96)	1.0
SuperARV		12.83(43.86)	1.0
Loose SCDG LM	**(1): basic model**	12.79(44.25)	0.8
	(2): (1)+crossing dependencies	12.71(45.86)	0.8
	(3): (2)+distance and barrier model	12.59(48.33)	0.8
	(4): (3)+modifiee lexical features	12.44(49.02)	0.8
Tight SCDG LM using (4)		12.18(53.66)	0.9
Chelba (rerun)		12.89(41.66)	0.6
Chelba (2000)		13.0(–)	0.6
Xu and Chelba (2002)		12.3(–)	0.6
Lattice Accuracy		3.41 (68.86)	N/A

TABLE 10.6 Comparing WER and SAC (%) after rescoring lattices or n-best lists using each LM on the DARPA WSJ CSR 1993 20K open vocabulary evaluation test set. The lattice WER and SAC that define the best accuracy possible for a set of lattices given perfect knowledge is also provided.

Table 10.6 shows the word error rate (WER) and sentence accuracy (SAC) after rescoring with each LM. Note that Lattice WER/SAC shows the best possible accuracy for the set of lattices used given perfect knowledge. As can be seen from

the table, all the SCDG parser LMs outperformed the trigram, POS, SuperARV, and Chelba's parser-based LMs. As was found for the SCDG parser, model enrichment improved the performance of the loosely coupled SCDG parser-based LM. Also, the tightly coupled SCDG parser-based LM outperformed each of the loosely coupled SCDG LMs, which verifies our belief that this tight integration benefits both tagging and modifiee determination. Furthermore, the tightly coupled LM performed slightly better than Xu and Chelba's revised structured LM (Xu et al., 2002).

It should be noted that Xu et al. (2002) trained their LM on a subset of the WSJ CSR training data and rescore using 50-best lists derived from the same set of lattices. Consequently, to more directly compare our tightly coupled SCDG parser-based LM to this state-of-the-art LM, we retrained it on the smaller set of training data used by Xu et al. (2002) and then used the resulting LM to rescore 50-best lists. In this case, we obtained a WER of 12.5%, higher than the 12.3% reported by Xu and Chelba. However, when the tightly coupled SCDG parser LM was trained on the complete data set, we obtained a comparable WER of 12.3% given 50-best rescoring. In prior work on CDG induction (Harper and Wang, 2001), we found that CDGs tend to have a greater selectivity and lower generality than CFGs learned from the same corpus, which is further confirmed by these findings. Although it is possible to relax grammar constraints to increase generality, we have found that a selective SCDG parser LM trained on more data performs better than a more general SCDG model.

Table 10.6 also shows the interpolation weight λ for combining each corresponding model with the baseline trigram (optimized on the same development set). As discussed in section 10.4.3 and in Wang and Harper (2002), the POS and almost-parsing SuperARV LMs were not improved by interpolation because they already contain word cooccurrence knowledge in their models. By contrast, the parser-based LMs of Chelba (2000) and Xu et al. (2002) obtained a decrease in WER when interpolated with word n-gram LMs. Because of their focus on modeling syntactic knowledge with nonterminals, these parser-based LMs may not capture all the local word cooccurrence information available to the trigram. Our SCDG parser-based LM tightly integrates word and parse structure information; however, all the SCDG parser-based LMs, whether loosely or tightly coupled with SuperARV tagging, obtained an improved performance given interpolation with the trigram, possibly due to pruning. The lower interpolation weight of the loosely coupled SCDG parser-based LM indicates that it received a greater level of compensation from the word n-gram than its tightly coupled counterpart.

The SCDG parser-based LM tightly integrates multiple knowledge sources such as word identity, syntactic constraints, and lexical features. The model utilizes long-distance dependency information and subcategorization information to make word predictions. When evaluated on the WSJ CSR task, the model outperformed the almost-parsing SuperARV LM and produced a recognition accuracy comparable to or exceeding state-of-the-art parser-based LMs.

10.5 Conclusions and Future Directions

We have developed and investigated annotation and extraction approaches to learn CDGs from annotated corpora. From these investigations, we have gained insights into the contributions of different types of constraints to grammar quality and a better understanding concerning the inductive bias of a CDG learned from a corpus. Using these insights, we have developed high-quality statistical CDG-based parsing models

and language models built on the concept of a SuperARV, which is an abstraction of the joint assignment of dependencies for a word that provides a mechanism for lexicalizing CDG parsing rules.

The SuperARV framework was originally developed for read and planned speech, which tends to exhibit more standard grammatical structures. The models described in this chapter did not include any provisions for dealing with the special features of conversational speech, such as incomplete sentences and disfluencies, and future work will be aimed at modeling these features. Conversational speech raises many challenges to our models. Effective automatic identification of speech repairs is important for building higher quality treebanks automatically, for improving parsing accuracy, and for building higher quality language models for spontaneous speech. Most current parsers assume that sentence boundaries are given and parse at the sentence level; however, speech recognizers produce only words as output. Although recognizers do work on segments of speech, these rarely correspond to a sentence in text. Because this can cause serious problems for a parser-based language model, understanding the impact of segmentation on parser-based LMs is important to obtaining further improvements to these models (Harper et al., 2005).

Another interesting direction for future work is applying our models to languages other than English. Since CDG is the formal basis of our models, there is sufficient grammar power to model a variety of other languages. Foth et al. (2000), Heinecke et al. (1998), Menzel (1994), Menzel (1995), and Schröder et al. (2000) have used CDG to parse spoken and newswire German test sets. Building on the insights of this prior work, we should be able to construct an SCDG parser and language models for German. The availability of the Prague Dependency Treebank (Hajic, 1998) and the research conducted at the 1998 Johns Hopkins Summer Workshop (Hajic et al., 1998) makes Czech an attractive option. Another possible direction is to model Asian languages such as Chinese. Since there are few inflectional and grammatical markers in the Chinese language, an approach that combines syntactic and semantic knowledge is likely to be important for producing a high-quality model.

10.6 Acknowledgments

We thank Srinivas Bangalore, Eugene Charniak, Ciprian Chelba, Jason Eisner, Yang Liu, Andreas Stolcke, and Peng Xu for valuable input to and support of the research described in this chapter. This research has been supported by Purdue Research Foundation, the National Science Foundation (NSF) under grant number 9980054-BCS, the Advanced Research and Development Activity (ARDA) under contract number MDA904-03-C-1788, and by the Defense Advanced Research Projects Agency (DARPA) under contract number MDA972-02-C-0038. Any opinions, findings, and conclusions expressed in this paper are those of the authors and do not necessarily reflect the views of Purdue, NSF, ARDA, or DARPA.

References

Allen, J. (1995). *Natural Language Understanding*. Benjamin/Cummings, 2^{nd} edition.

Bulyko, I., Ostendorf, M., and Stolcke, A. (2003). Getting more mileage from web text sources for conversational speech language modeling using class-dependent mixtures. In *Proceedings of the Joint Conference on Human Language Technology and North American Chapter of Association for Computational Linguistics (HLT-NAACL)*, volume 2, Edmonton, Alberta, Canada, pages 7–9.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proceedings of the First Annual Meeting of the North American Association for Computational Linguistics*.

Charniak, E. (2001). Immediate-head parsing for language models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Charniak, E., Blaheta, D., Ge, N., Hall, K., and Johnson, M. (2000). BLLIP WSJ Corpus. CD-ROM. Linguistics Data Consortium.

Chelba, C. (2000). *Exploiting Syntactic Structure for Natural Language Modeling*. PhD thesis, CLSP, The Johns Hopkins University.

Chen, S. F. and Goodman, J. T. (1998). An empirical study of smoothing techniques for language modeling. Technical report, Harvard University, Computer Science Group.

Collins, M. J. (1996). A new statistical parser based on bigram lexical dependencies. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 184–191.

Collins, M. J. (1999). *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania.

Eisner, J. M. (1996). An empirical comparison of probability models for dependency grammar. Technical report, University of Pennsylvania, CIS Department, Philadelphia PA 19104-6389.

Foth, K., Menzel, W., Pop, H. F., and Schrder, I. (2000). An experiment in incremental parsing using weighted constraints. In *Proceedings of the 18th International Conference on Computational Linguistics*, Saarbürcken, Germany, pages 1026–1030.

Hajic, J. (1998). Building a syntactically annotated corpus: The Prague Dependency Treekbank. In *Issues of Valency and Meaning (Festschrift for Jarmila Panevova)*, pages 106–132. Carolina, Charles University, Prague.

Hajic, J., Brill, E., Collins, M., Hladka, B., Jones, D., Kuo, C., Ramshaw, L., Schwartz, O., Tillmann, C., and Zeman, D. (1998). Core natural language processing technology applicable to multiple languages – Workshop '98. Technical report, Johns Hopkins University.

Harbusch, K. (1997). The realtion between tree-adjoining grammars and constraint dependency grammars. In *Proceedings of the Fifth Meeting on the Mathematics of Language*, pages 38–45.

Harper, M., Dorr, B., Hale, J., Roark, B., Shafran, I., Lease, M., Liu, Y., Snover, M., Yung, L., Krasnyanskaya, A., and Stewart, R. (2005). 2005 Johns Hopkins Summer Workshop final report on parsing and spoken structural event detection. Technical report, Johns Hopkins University.

Harper, M. P. and Helzerman, R. A. (1995a). Extensions to constraint dependency parsing for spoken language processing. *Computer Speech and Language*, 9:187–234.

Harper, M. P. and Helzerman, R. A. (1995b). Managing multiple knowledge sources in constraint-based parsing spoken language. *Fundamenta Informaticae*, 23(2,3,4):303–353.

Harper, M. P., Hockema, S. A., and White, C. M. (1999a). Enhanced constraint dependency grammar parsers. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing*.

Harper, M. P. and Wang, W. (2001). Approaches for learning constraint dependency grammar from corpora. In *Proceedings of the Grammar and Natural Language Processing Conference*, Montreal, Canada.

Harper, M. P., White, C. M., Helzerman, R. A., and Hockema, S. A. (1999b). Faster MUSE CSP arc consistency algorithms. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing*.

Harper, M. P., White, C. M., Wang, W., Johnson, M. T., and Helzerman, R. A. (2000). Effectiveness of corpus-induced dependency grammars for post-processing speech. In *Proceedings of the 1st Annual Meeting of the North American Association for Computational Linguistics*, pages 102–109.

Heeman, P. A. (1998). POS tagging versus classes in language modeling. In *Proceedings of the 6th Workshop on Very Large Corpora, Montreal*.

Heinecke, J., Kunze, J., Menzel, W., and Schröder, I. (1998). Eliminative parsing with graded constraints. In *Proceedings of the International Conference on Computational Linguistics and the Annual Meeting of the Association for Computational Linguistics*.

Helzerman, R. A. and Harper, M. P. (1992). Log time parsing on the MasPar MP-1. In *Proceedings of the Sixth International Conference on Parallel Processing*, volume 2, pages 209–217.

Helzerman, R. A. and Harper, M. P. (1996). MUSE CSP: An extension to the constraint satisfaction problem. *Journal of Artificial Intelligence Research*, 5:239–288.

Holan, T., Kuboň, V., Oliva, K., and Plátek, M. (1998). Two useful measures of word order complexity. In Polguère, A. and Kahane, S., eds., *Processing of Dependency-Based Grammars: Proceedings of the Workshop*, pages 21–28.

Järvinen, T. and Tapanainen, P. (1998). Towards an implementable dependency grammar. In Polguère, A. and Kahane, S., eds., *Processing of Dependency-Based Grammars: Proceedings of the Workshop*, pages 1–10.

Jelinek, F. (1990). Self-organized language modeling for speech recognition. In Waibel, A. and Lee, K.-F., eds., *Readings in Speech Recognition*. Morgan Kaufmann, San Mateo, CA.

Jelinek, F. (1997). *Statistical Methods For Speech Recognition*. MIT Press.

Joshi, A. K., Levy, L., and Takahashi, M. (1975). Tree adjunct grammars. *Journal of Computer and System Sciences*, 10:136–163.

Joshi, A. K. and Srinivas, B. (1994). Disambiguation of super parts of speech (or supertags): Almost parsing. In *Proceedings of the 1994 International Conference on Computational Linguistics*.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Maruyama, H. (1990a). Constraint Dependency Grammar. Technical Report #RT0044, IBM, Tokyo, Japan.

Maruyama, H. (1990b). Constraint Dependency Grammar and its weak generative capacity. *Computer Software*.

Maruyama, H. (1990c). Structural disambiguation with constraint propagation. In *The Proceedings of the Annual Meeting of Association for Computational Linguistics*, pages 31–38.

Menzel, W. (1994). Parsing of spoken language under time constraints. In *11th European Conference on Artificial Intelligence*, pages 560–564.

Menzel, W. (1995). Robust processing of natural language. In *Proceedings of the 19th Annual German Conference on Artificial Intelligence*.

Menzel, W. (1996). Constraint satisfaction for robust processing of spoken language. In *Proceedings of the European Conference on Artificial Intelligence Workshop on Non-Standard Constraint Processing*.

Niesler, T. R. and Woodland, P. C. (1996). Variable-length category-based N-gram language model. In *Proceedings of the International Conference of Acoustics, Speech, and Signal Processing*, pages 164–167.

Polguère, A. and Kahane, S., eds. (1998). *Processing of Dependency-Based Grammars: Proceedings of the Workshop*. Association For Computational Linguistics, New Brunswick, NJ.

Price, P. J., Fischer, W., Bernstein, J., and Pallett, D. (1988). A database for continuous speech recognition in a 1000-word domain. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 651–654.

Ratnaparkhi, A. (1999). Learning to parse natural language with maximum entropy models. *Machine Learning*, 34:151–175.

Roark, B. (2001). Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

Rosenfeld, R. (August 2000). Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 88:1270–1278.