Scalability of Redundancy Detection in Focused Document Collections

Dick Crouch, Cleo Condoravdi, Reinhard Stolle, Tracy King,
Valeria de Paiva, John Everett, Daniel Bobrow
rdc+@parc.com
PARC
3333 Coyote Hill Road
Palo Alto, CA, USA, 94304

Abstract

We describe the application of primarily symbolic methods to the task of detecting logical redundancies and inconsistencies between documents in a medium sized, domain focused collection (1000–40,000 documents). Initial investigations indicate good scalability prospects, especially for syntactic and semantic processing. The difficult and largely neglected task of mapping from linguistic/semantic representations to domain tailored knowledge representations is potentially more of a bottleneck.

1 Introduction

We describe aspects of a prototype system being developed for the detection of redundancies and inconsistencies in a medium sized, domain-focused document collection. This is being applied to Eureka, a database of around 40,000 technician-authored, free-text tips concerning the repair of photocopiers. (Fig. 1 illustrates two short and partially redundant tips.) The system is built on the hypothesis that deep symbolic analysis, mapping texts onto formal semantic/knowledge representations, is a necessary component for detecting redundancy and inconsistency between documents. While surface approaches (e.g., latent semantic indexing) do well at detecting overall similarity of subject matter, they offer limited purchase on the detection of essentially logical/inferential relations.

The system is currently transitioning from an initial phase where it was applied to 31 documents, hand-selected for their known similarities, and post-edited to correct egregious spelling and grammatical errors. It is now in the process of being scaled up to cover a randomly selected set of over 1300 documents that have only been post-edited to the extent of running a spelling checker over them. This paper focuses on some of issues arising from this effort: what scales well, and what is more problematic.

The principal message is that the scalability prospects of symbolic methods for natural language understanding are generally good. Symbolic parsing scales well, both in terms of coverage and in terms of dealing with the proliferation of ambiguity that arises out of broad coverage. Broad syntactic coverage enables broad semantic coverage, at the level of compositional/syntax-driven semantic interpretation. A more problematic area is the frequently neglected task of mapping linguistically motivated semantic representations onto inferentially tractable conceptual representations, on the basis of domain knowledge. But in all cases, scalability is fostered by the use of general purpose, theoretically motivated mechanisms.

The paper is organized as follows. Sec. 2 introduces some of the system components. Sec. 3 describes the nature of the redundancy detection task. Sec. 4 summarizes the use of statistical techniques to identify similar documents that can then be subjected to deeper analysis. Sec. 5 describes the syntactic and semantic processing, and discusses issues of linguistic scalability. Sec. 6 discusses some of the core research issues raised by the task of mapping from semantic to knowledge representation.

Tip 57	
Problem:	Left cover damage
Cause:	The left cover safety cable is breaking, allowing the left cover to pivot too far, breaking the cover.
Solution:	Remove the plastic sleeve from around the cable. COMMENTS: Cutting the plastic off of the cable makes the cable more flexible, which prevents cable breakage. Cable breakage is a major cause of damage to the left cover. PART NUMBERS: 2E30072, 12K1980

Tip 118	
Problem:	The current safety cable (12K1980) used in the 5100 Document Handler fails prematurely causing the Left Document Handler Cover (2E30072) to break.
Cause:	The plastic jacket made the cable too stiff. This causes stress to be concentrated on the cable ends, where it eventually fails.
Solution:	When the old safety cable (12K1980) fails, replace it with the new one (12K1981), which has the plastic jacket shortened.

Figure 1: Two Eureka tips with the same diagnosis of a problem, but different solutions.

2 System Components

The system uses a broad coverage, hand coded Lexical Functional Grammar of English (Butt et al., 1998) and the parser from the Xerox Linguistic Environment (XLE) (Maxwell and Kaplan, 1993). The sentences in a document are parsed and then semantically interpreted via a process of linear logic deduction ("Glue Semantics") (Dalrymple, 1999). The resulting semantic representations are mapped through to linguistically neutral knowledge representations (KR). The semantics-to-KR mapping is an inference driven process, drawing on a set of mapping rules and domain facts and ontology. As well as canonicalizing linguistic input to a standard underlying conceptual structure, the semantics-to-KR mapping addresses such issues as the domain-driven resolution of ambiguity. Finally conceptual structures for document pairs are compared for overlap and inconsistency at a number of levels: (i) entities and events referred to, (ii) entities and events linked by thematic roles, (iii) overall causal and narrative structure matched using the Structure Mapping Engine (Forbus et al., 1989).

A semantics-to-KR mapping is beneficial for our particular natural language understanding task, and probably for others as well. If understanding is characterized—at least in part—as the ability to reason and draw inferences, then a degree of understanding is packaged into the semantics-to-KR mapping. This allows for inferentially more lightweight and tractable procedures, such as structure mapping, to be applied to the resulting, canonicalized representations to detect instances of entailments and inconsistencies. An alternative to a semantics-to-KR mapping would be to reason directly on linguistic semantic representations, using domain knowledge as a background theory. We suspect this would be computationally prohibitive.

More generally, a semantics-to-KR mapping introduces a deeper level of semantic representation, more specific to a particular domain and a particular understanding task. This specificity should not be a barrier to scalability, since a large part of the mapping can be done in a domain and task independent way.

3 Redundancy Detection

The task of detecting redundancies and inconsistencies in a focused document collection has much to recommend it. From a practical point of view it is a useful application. The number of high-value, domain-specific, medium-sized focused document collections is proliferating on corporate intranets and elsewhere. However, these collections tend only to be valuable if their quality is maintained. Confronted by multiple documents saying the same thing, or worse, documents that contradict one another, many users will give up consulting the collection. Tools for identifying such redundancies and inconsistencies are needed, since detecting them by hand in a collection of more than a few thousand documents may be prohibitively expensive.

From a theoretical and engineering point of view, the task has nice properties. First, it is one where a degree of language understanding is required. Sophisticated, bag-of-words based techniques like latent semantic indexing can perform

a good job of identifying documents that are similar/concern the same subject matter. But these techniques offer little purchase on finer distinctions such as whether two documents on the same subject matter agree with or contradict one another. Some degree of understanding and inference is required for this.

Second, in terms of language understanding the task is a relatively low level one. In particular, it is not necessary to have everything working perfectly before any useful results can be produced. One can use incomplete analyses in a tool that filters out candidate sets of documents for final validation by a human editor; the better the filter, the less search required by the validator. In this context, incremental development of linguistic and knowledge based technology is feasible, and its impact measurable.

Third, the task concerns the understanding of natural language texts rather than (spoken) dialogues. This has advantages, but also poses certain challenges. On the positive side, processing does not have to be done in real time; turnaround times of minutes (or even hours) for a new document entering the collection are quite acceptable. On the negative side, the input tends to be much longer and to exhibit a much greater degree of ambiguity. Unlike a dialogue system, where clarification sub-dialogues can be used to resolve ambiguities before they proliferate, ambiguity management is a major theoretical and engineering issue.

4 Similarity Pre-Filtering

Similarity between documents is not the same thing as redundancy or inconsistency between them. Two documents can be similar and concern the same subject matter, and yet exhibit various relations to each other: (i) redundancy: the contents of one document entail/subsume those of the other; (ii) inconsistency: the documents contradict one another; (iii) independence: the documents neither entail nor contradict one another. Fig. 1 illustrates two documents that are redundant with regard to the diagnosis of a problem, but independent in suggesting alternative solutions. Nonetheless, detecting broad document similarity is a useful preliminary to detecting redundancy and inconsistency—these finer grained relations are unlikely to hold in the absence of similarity.

We have therefore investigated the utility of Probabilistic Latent Semantic Analysis (PLSA) (Hofmann, 2000) for the task of finding similar documents in our test corpus of 1321 tips. A matching algorithm that employs a word-based PLSA model produced a similarity ranking of the 871,860 pairwise comparisons of tips. Within the 100 top-ranked pairs, 88 pairs represent tips whose contents exhibit an interesting conceptual relation between each other. Furthermore, an independent manual sampling of the 1321 tips had revealed 17 pairs of conceptually related documents. Three of these 17 pairs are among the 100 top-ranked pairs; all of the 17 pairs appear in the top 7% of the ranking. A detailed description of the statistical model, the matching algorithm, and the experimental results can be found in (Brants and Stolle, 2002).

These results strongly suggest that the determination of surface similarity can be a useful pre-filter, identifying those pairs of documents for which more detailed analysis and comparison may prove worthwhile.

5 Syntactic and Semantic Processing

5.1 Parsing

The grammar used for parsing has been developed as part of the ParGram project (Butt et al., 1998). It uses LFG as a formalism, producing c(onstituent)-structures (trees) and f(unctional)-structures (attribute value matrices). The f-structures encode predicate-argument relations and other grammatical information, e.g., number, tense. When parsing, the input string is first tokenized using finite state tools and then run through a finite state morphology before being processed by the grammar. The grammar consists of annotated phrase structure rules, which combine output from the morphology into words and then into phrases, and a set of lexicons. A successful parse produces one or more well-formed trees and corresponding f-structure(s).

The grammar has 323 rules with regular expression right-hand sides which compile into a collection of finite-state machines with a total of 5,132 states and 13,639 arcs. The grammar uses several lexicons and three guessers, one for items known to the morphology but not to the

syntax, one for morphologically unknown items, and one for domain specific part numbers, fault codes etc. As such, most common and proper nouns, adjectives, and adverbs have no explicit lexical entry. The main verb lexicon contains 9,652 verb stems and 23,525 subcategorization frame-verb stem entries; there are also lexicons for adjectives and nouns with subcategorization frames and for closed class items, e.g., pronouns, modals.

To increase robustness, the grammar has been augmented with a fragment grammar. This grammar parses the sentence as well-formed chunks specified by the grammar; these contain Ss, NPs, PPs, and VPs. Any string which cannot be parsed as a chunk is parsed as a token. The grammar has a fewest-chunk method for determining the correct parse; for example, if a string can be parsed as two NPs and a VP or as one NP and an S, the NP-S option is chosen. Additionally, the parser enters a skimming mode performing incomplete analysis of remaining constituents when the amount of time or memory devoted to a sentence exceeds a threshhold. Between them, the fragment grammar and skimming ensures that every sentence gets an analysis.

The XLE parser (Maxwell and Kaplan, 1993) is used to perform syntactic analysis of the documents relative to the grammar. This produces for each sentence the set of all possible analyses permitted by the grammar, applying a number of techniques to compute in typically polynomial time a potentially exponential number of different analyses. The complete set of parses is represented in a parse-forest data structure that is also typically polynomial in size. Efficient packing techniques for ambiguity management make it feasible to search for all possible analyses of long sentences (containing 40 or more words).

5.1.1 Grammar Coverage

A recent driving force behind the development of the grammar has been its application to parsing the UPenn Wall Street Journal corpus. Recent results (Riezler et al., 2002) indicate that the grammar parses all of WSJ Section 23 as unseen unlabeled data: 74.7% full parses, 25.3% fragment parses or skimmed parses. Accuracy is measured (roughly) in terms of how much of the predicate-argument structure in parsed f-structures matches that in gold standard hand-corrected annotations for 700 WSJ test sentences. This gives F-measures of: 76% (picking a parse at random), 84% (picking the best matching parse) and 79% (picking a parse via stochastic disambiguation).

Essentially the same grammar is used to parse the Eureka tips, though with the addition of (i) some extra lexical entries and regular expressions to capture idiosyncratic domain vocabulary such as fault codes and part numbers, and (ii) a handful of additional constructions, e.g., to account for a tendency to drop definite determiners. No attempt has yet been made to hand annotate a Eureka test set to evaluate coverage accuracy. For the extended set of documents now being used, the grammar yields around 30% fragment parses. The higher proportion of fragment parses indicates both the greater basic ungrammaticality of the Eureka corpus compared to the WSJ, as well as the difficulty of segmenting free form text into sentences. However, a good proportion of the fragment parses are still indicative of gaps in grammatical coverage. As an indication of processing time, for the two tips in Fig. 1 total parsing time (on a 2GHz processor with 2Gb RAM) was 9 seconds, with no fragmentary parses. For a larger sample of 3949 sentences, representing around 200 tips, total parsing time was 7479 seconds, with an average parsing time of 16 seconds for sentences of length 41–50 words, and overall an average ambiguity of around 30 parses per sentence. These processing times are not appropriate for crawling the web, but are quite acceptable for off-line processing of relatively slowly changing document collections.

5.2 Semantic Interpretation

We have dwelt on grammatical issues, since broad and deep parsing is a prerequisite for successful semantic interpretation. It is useful to identify at least 4 levels of semantic representation, increasingly distant from syntactic output.

(1) Basic predicate argument structure: This is essentially what is provided by the f-structure output of parsing. Much of the semantic coverage in the system is thus directly parasitic on syntactic coverage.

(2) Scoped, context-independent logical

forms: F-structures do not provide the standard logical machinery of variables, connectives, quantifiers, etc., and the first phase of semantic interpretation does so. A Glue Semantics (Dalrymple, 1999) implementation is used. The results of parsing give rise to a set of lexical premises. The premises are lexical meanings associated with propositional linear logic formulas. Atomic propositions denote semantic resources corresponding to constituents found in parsing. A process of inference finds all possible ways of consuming these lexical premises to produce a single semantic resource for the sentence, generating a distinct meaning for the sentence as a by-product of each derivation.

Thus glue interpretation typically increases the degree of ambiguity, not only through generating alternative quantifier scoping, but also through other more subtle instances of modifier scope and attachment ambiguity. Ambiguity is managed via a two stage derivation technique, building on (Gupta and Lamping, 1998). This first leaves to one side all modifiers (corresponding to linear logic identities), and then packs all possible interpolations of modifiers into the final derivation. This gives rise to a packed semantic representation, similar to the packed syntactic representations that the XLE produces for f-structures. This packed structure makes it easy to identify and eliminate most spurious ambiguities, e.g., alternate but equivalent scopings of existential quantifiers or conjunctions. In fact, the semantic analysis is deliberately set up to minimize the number of non-spurious ambiguities. Thus plural NPs default to a collective reading corresponding to an existential quantification over a group object.

In addition to scoping NPs and modifiers, Glue interpretation also provides a number of semantic arguments not explicitly present in f-structures, e.g., Davidsonian event arguments for verbs; comparison class arguments for comparative and superlative constructions. This is also the appropriate place to provide additional arguments for different verb classes, e.g., event plus resultant state arguments for verbs of achievement; cause of change arguments in intransitive versions of verbs like *break* (capturing that *John broke the window* implies *the window broke*); object of change arguments in intransitive versions of verbs like *eat* (capturing that *John ate the meal* implies *John ate*). However, this has not yet been systematically pursued, and is a major task still to be performed in the scaling of semantic coverage.

(3) Contextually resolved logical forms: the Glue implementation does not currently attempt to resolve contextual dependencies, such as anaphoric references or the interpretation of compound nouns. There are for this. First, there are proposals for anaphora and ellipsis resolution as part of Glue interpretation, e.g., (Crouch, 1999). Essentially, this involves a model of context that makes contextual resources (modelled as linear logic propositions) available as premises to Glue derivations, and also the output of such derivations. However, this requires slightly extending the fragment of linear logic used for Glue interpretation (to include tensors: a form of resource sensitive conjunction). The consequences of this extension on the packing of semantic ambiguity are still being explored. Second, we have been investigating the combination of Glue interpretation with the use of discourse analysis (Polanyi and van den Berg, 1995) for anaphora resolution, but this is also still under exploration. Third, such things as compound noun resolution cannot readily be modelled as drawing on contextual resources made available by previous derivations. Rather, it requires a process of domain inference to spell out plausible relations between the nouns. At present, we therefore place hooks in semantic representations to mark where the semantics-to-KR mapping needs to resolve domain and contextual dependencies.

Here is an example output of Stage (3):

```
cable(cable220), plastic(plastic219),
def(cable220,?A), def(plastic219,?B),
nullAgent(nullAgent221), nullpro(nullAgent221,?C),
cut_off(cut218, nullAgent221, plastic219, cable220),
gerund_event(cut218),
cable(cable214), def(cable214,?D),
comparison_class(cclass215,?E),
prop_arg(prop211, more(flexible,cable214,cclass215)),
make(make_ev213, cut218, prop211),
cable(cable216),typedef(cable216,?F),
nn(cable216, breakage, ?G, breakage217),
cf(A1, prevent(prevent_ev212,make_ev213,breakage217)),
cf(A2, prevent(prevent_ev212, cut218, breakage217)),
cf(A3, prevent(prevent_ev212, cable214, breakage217))
```

This is the output, skolemized and with some details (e.g., tense) suppressed for space, for

the sentence *Cutting the plastic off of the cable makes the cable more flexible, preventing cable breakage.* Some salient points: (i) The variables ?A–?G represent contextual hooks. For example ?A, ?D and ?F may subsequently get instantiated to show that the skolem constants cable220, cable216 and cable214 are co-referring. (ii) Recursive structure is flattened by means of introducing constants like prop211 to denote propositional arguments to predicates. (iii) The last three lines represent the packing of three distinct syntactic analyses into one semantic representation. In parse A1 it is the make event that prevents breakage, in A2 the cutting, and in A3 the cable. (iv) The predicate "breakage" also illustrates a limitation, since there is nothing to link it explicitly to the verbal predicate "break." Currently, this canonicalization is done in the semantics-to-KR mapping: one of many instances where there is some latitude about where a task belongs. (v) The typing of skolem constants at this level does not currently make the distinctions argued for in (Condoravdi et al., 2001), according to which the breakage217 argument to "prevent" should denote a concept rather than an entity. The semantics-to-KR mapping fills this gap by construing the third argument to "prevent" as type denoting.

Provisional results show that of the 3949 sentences for which parsing figures were given, 3117 received a full semantic analysis for at least one possible parse, including some with fragmentary parses. By the time of the workshop we hope to report on a mechanism for pulling out fragmentary semantic analyses when a full analysis is not found. The packing of semantic representations across alternative parses is currently performed in a non-optimal way, leading to a turnaround time of about 4 hours.

(4) Inferentially tractable knowledge representation: The final level of semantic representation is discussed in the next section. At issue is mapping from a representation that reflects linguistic structure to one designed to support tractable inference and other tasks. As points (iv) and (v) above illustrate, there are gray areas where it is unclear whether some aspect of interpretation should be carried out at Stage (4) or before.

6 Semantics-to-KR Mapping

Our strategy for redundancy detection, as described in Sec. 3, is first to build a canonicalized representation of the contents of each document in the form of a set of assertions, and then to perform a pairwise comparison of the documents' canonicalized representations using the Structure Mapping Engine (Forbus et al., 1989). At the core of this process needs to be a principled technique for mapping from semantic representations to knowledge representations that encode the conceptual contents in a linguistically neutral fashion.

The goal of the semantics-to-KR mapping is to build a data structure whose interpretation is faithful to the output of the linguistic processing. However, the degree of completeness of the correspondence between the two structures can vary with the task for which the KR structures are being produced. In our application of redundancy detection, for example, the level of abstraction at which the conceptual content is represented is driven by the level of abstraction at which we wish to identify redundant content.

Both the input and the output of the semantics-to-KR mapping are formal representations with compositionally defined semantics. The compositionality of the linguistic semantics reflects *syntactic structure*. The compositional ingredients on the KR side, on the other hand, are motivated by the need to perform *automated reasoning* or *structure mapping*. As a result, the two types of structure do not lend themselves to a straightforward one-to-one mapping. This misalignment leads to one of the main difficulties in defining a principled way of mapping from the semantic representations to KR.

A further difference between the nature of the semantic representation and the nature of the conceptual representation is the way in which these data structures relate to knowledge that is external to the represented natural language text. The Glue semantics representations are largely "stand-alone" structures in that they are almost completely independent of the domain and largely independent of the context of discourse. The KR structures, on the other hand, interpret the text in a particular context, in a particular domain, and aiming at a particular

reasoning task.[1] Therefore, the semantics-to-KR mapping process is guided by an ontology and facts about the domain and about the world and by inferences drawn from this domain and world knowledge. Much can be filled in from these external knowledge sources that is only implicit in the language or missing altogether.

The remainder of this section discusses some salient aspects of our system, some examples of the misalignment described above, and some general interesting and difficult problems any such mapping would have to address.

In contrast to other work in this area, e.g., (Hobbs et al., 1993), reasoning is not directly done on linguistically based logical forms but on canonicalized, linguistically independent representations. Words are mapped to concepts from an independently specified ontology, whose terms are unambiguous. For instance, there is no concept bank_type, with bank-financial-institution_type and riverbank_type as specializations. The ontological hierarchy should reflect the specificity of concepts rather than accidental homonymy.

The canonicalization of the linguistic input to conceptual structure operates at the lexical level as well as at higher levels. At the lexical level, "sleeve" and "jacket" in our example map to the same concept covering_type. Similarly, verbal predicates, such as "break," and their corresponding nominalizations, such as "breakage," map to the same concept breaking-event_type. Canonicalization at a higher level is exemplified by mutually entailing sentences with distinct linguistic semantic representations but identical knowledge representations, such as *Cutting off the plastic makes the cable more flexible* and *Cutting off the plastic makes the cable less stiff*.

Ontological commitments are made in the semantics-to-KR mapping rather than in the linguistic semantics. We believe that this choice affords the system a higher degree of modularity. For instance, event skolems are interpreted as event-denoting terms and events are taken to be ontologically concrete particulars in the tradition of (Davidson, 1980) or (Vendler, 1967). As a consequence, event skolems in the linguistic semantic representation that are arguments of predicates denoting happenings correspond to event skolems in KR that denote either individual events or event types. For example, cut218, an argument of the cut_off predication in the semantics, corresponds to an instance of the concept cutting-event_type in the KR fragment below, while breakage217 corresponds to a subconcept of the concept breaking-event_type. On the other hand, purely relational predicates, such as "prevent," map to predicates rather than event types in KR and the corresponding event skolems do not appear in KR, e.g., prevent_ev212. However, we could choose to make different ontological commitments—for instance, taking event skolems to be situations as in (Schubert and Hwang, 2000) or names for fine-grained propositions as in (Hobbs, 1985)—and still use the same output of the linguistic semantics.

As an example, the following is the output of the semantics-to-KR mapping for the sentence whose semantic representation was given in the previous section.

```
(isa cable220 cable_type), (isa plastic219 plastic_type),
(isa cut218 cutting-event_type),
(changed-object cut218 cable220),
(removed-object cut218 plastic219),
(causes-decrease cut218 cable220 rigidity),
(prevents cut218 breakage217)
(equals breakage217
   (subtype_construct breaking-event_type
    (restrictions ?var (changed-object ?var cable220))))
```

This example illustrates that we adopt a neo-Davidsonian framework for knowledge representations. We thus have a separate assertion specifying the type of the event skolem and relate the event skolem with the other arguments via thematic relations, such as changed-object and removed-object. Event types, like all other types, are organized in a hierarchical ontology.

Use of a neo-Davidsonian framework is another ingredient of the canonicalization process, as it allows for mapping to the same concept regardless of the arity of the linguistic predicate. It is also better suited for our "match the overlap" approach, as it allows for incomplete information: not all arguments of the linguistic predicate need to be related to the event skolem via some thematic relation in order to obtain a well-formed representation.

Two further properties of the KR language

[1] As a consequence, when porting the system to a different domain, performing the required knowledge engineering is a considerably larger effort than adapting the infrastructure for syntactic and semantic processing.

are worth noting. (i) In addition to the basic ontological vocabulary, the KR language should be expressive enough to allow for various kinds of constructors to create expressions of various types dynamically. The KR fragment above, for instance, contains the term-creating constructor subtype_construct, essential for making reference to those breaking events whose changed-object is cable220.[2] (ii) Some of the vagueness and context-dependency present in the linguistic input should be mirrored in the KR language. In our example, the object of comparison is not explicitly specified but is hidden in a special KR predicate, causes-decrease, for which the object of comparison is relationally fixed (in this case, the degree of rigidity of cable220 prior to the occurrence of cut218).

The final component in our system architecture is the identification of redundancies and inconsistencies between documents, based on the knowledge representations generated by the semantics-to-KR mapping. We use the Structure Mapping Engine (Forbus et al., 1989), a computational model of Gentner's Structure Mapping Theory (Gentner, 1983), to determine the structural overlap for pairs of documents. SME anchors its matching process in identical elements that occur in the same structural positions in both representations, and uses this to build a correspondence. SME seeks the largest possible mapping between two representations, subject to a systematicity constraint, which forces such mappings to be consistent in all their subparts, and a one-to-one mapping constraint. Finally, we reason about the overlaps and mismatches of structural fragments in order to identify redundancies and inconsistencies between documents.

References

T. Brants and R. Stolle. 2002. Finding similar documents in document collections. Submitted to *Using Semantics for Information Retrieval and Filtering: State of the Art and Future Research*, Workshop at LREC-2002.

M. Butt, T. King, M. Niño, and F. Segond. 1998. *A Grammar Writer's Cookbook*. CSLI Lecture Notes. CSLI, Stanford.

C. Condoravdi, R. Crouch, M. van den Berg, J. Everett, V. Paiva, R. Stolle, and D. Bobrow. 2001. Preventing existence. In C. Welty and B. Smith, editors, *Formal Ontology in Information Systems: Proc. FOIS-2001, Ogunquit, Maine*, pages 162–173. ACM Press, New York.

R. Crouch. 1999. Ellipsis and glue languages. In Shalom Lappin and Elabbas Benmamoun, editors, *Fragments: Studies in Ellipsis and Gapping*. Oxford University Press, Oxford.

M. Dalrymple, editor. 1999. *Semantics and Syntax in Lexical Functional Grammar: The Resource Logic Approach*. MIT Press, Cambridge, MA.

D. Davidson. 1980. The logical form of action sentences. In D. Davidson, editor, *Essays on actions and events*, pages 105–122. Clarendon Press, Oxford.

K. D. Forbus, B. Falkenhainer, and D. Gentner. 1989. The structure mapping engine: Algorithm and examples. *Artificial Intelligence*, 41(1):1–63.

D. Gentner. 1983. Structure mapping: A theoretical framework for analogy. *Cognitive Science*, 7:155–170.

V. Gupta and J. Lamping. 1998. Efficient linear logic meaning assembly. In *Proc. COLING-ACL-1998*, pages 464–470, Montreal.

J. R. Hobbs, M. Stickel, S. Appelt, and P. Martin. 1993. Interpretation as abduction. *Artificial Intelligence*, 63(1–2):69–142.

J. R. Hobbs. 1985. Ontological promiscuity. In *Proc. ACL-1985*, pages 61–69, Chicago, IL.

T. Hofmann. 2000. Probabilistic latent semantic indexing. In *Proc. SIGIR-1999*, pages 35–44, Berkeley, CA.

J. Maxwell and R. Kaplan. 1993. The interface between phrasal and functional constraints. *Computational Linguistics*, 19:571–589.

L. Polanyi and M. van den Berg. 1995. Discourse structure and discourse interpretation. In *Proc. 10th Amsterdam Colloquium*.

S. Riezler, R. Kaplan, T. King, M. Johnson, R. Crouch, and J. Maxwell. 2002. Parsing the Wall Street Journal using a lexical functional grammar and discriminative estimation techniques. To appear *ACL-2002*.

L. K. Schubert and C. H. Hwang. 2000. Episodic logic meets little red riding hood: A comprehensive, natural representation for language understanding. In L. M. Iwanska and S. C. Shapiro, editors, *Natural Language Processing and Knowledge Representation*. MIT Press.

Z. Vendler. 1967. Facts and events. In Z. Vendler, editor, *Linguistics in philosophy*, pages 122–146. Cornell University Press, Ithaca, NY.

[2] ?var is designated as a variable ranging over instances of breaking-event_type.