An Overview of SURGE: a Reusable Comprehensive Syntactic Realization Component

Michael Elhadad
elhadad@cs.bgu.ac.il
Mathematics and Computer Science Dept.
Ben Gurion University in the Negev
Beer Sheva, 84105 Israel
Fax: +972-7 472-909

Jacques Robin
jr@di.ufpe.br
Departamento de Informática
Universidade Federal de Pernambuco
Av. Prof. Luiz Freire s/n Cidade Universitária
Recife, PE 50740-540 Brazil
Fax: +55-81 271-4925 (or 4281)

Abstract

This paper describes SURGE, a syntactic realization front-end for natural language generation systems. By gradually integrating complementary aspects of various linguistic theories within the computational framework of functional unification, SURGE has evolved to be one of the most comprehensive grammars of English for language generation available today. It has been successfully re-used in a variety of generators, with very different architectures and application domains.

1 Introduction

This paper is an overview of SURGE (Systemic Unification Realization Grammar of English) a syntactic realization front-end for natural language generation systems. Developed over the last seven years[1] it embeds one of the most comprehensive computational grammar of English for generation available to date. It has been successfully re-used in eight generators, that have little in common in terms of architecture. It has also been used for teaching natural language generation at several academic institutions.

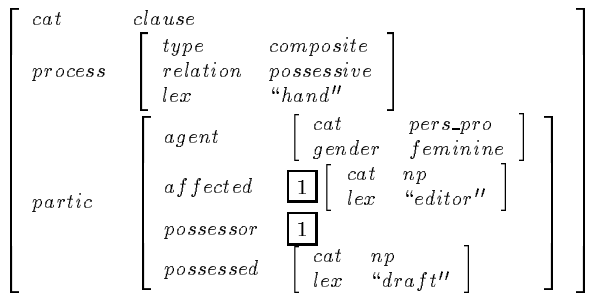
We first define the task of a stand-alone syntactic realization component within the overall generation process and provide criteria for its evaluation. We then briefly survey the computational formalism underlying the implementation of SURGE as well as the syntactic theories that it integrates. We then describe in more details the structure of the grammar and its coverage. Finally, we review the application domains across which SURGE has been re-used, compare it with other syntactic realization front-ends for generation and discuss its current limitations.

[1]The research presented in this paper started out while the authors were doing their PhD. at Columbia University, New York. We are both indebted to Kathleen McKeown for her guidance and support during those years.

2 Reusable Realization Component for NLG

Natural language generation has been traditionally divided into three successive tasks [30]: (1) content determination, (2) content organization, and (3) linguistic realization. The goal of a reusable realization component is to encapsulate the domain-independent part of this third task. The input to such component should thus be as high-level as possible without hindering portability. Independent efforts to define such an input [23] [21] [38] [25] have crystalized around a skeletal, partially lexicalized thematic tree specifying the semantic roles, open-class lexical items and top-level syntactic category of each constituents. An example SURGE input with the corresponding sentence is given in Fig. 1.

Input Specification (I_1):

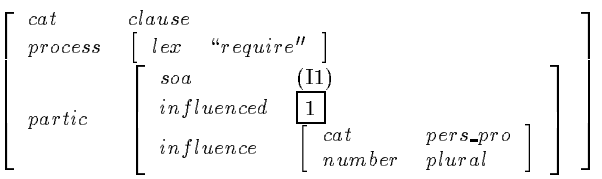


Output Sentence (S_1): "She hands the draft to the editor"

Figure 1: An example SURGE I/O

The Syntactic Realization Subtask The task of the realization component is to map such skeletal tree onto a natural language sentence. It involves the following sub-tasks:

1. *Map thematic structure onto syntactic roles*: *e.g.,* `agent`, `process`, `possessed` and `possessor` onto `subject`, `verb-group`, `direct-object` and `indirect-object` (respectively) in S_1.
2. *Control syntactic paraphrasing and alternations* [19]: *e.g.,* adding the `(dative-move yes)` feature to I_1 would result in the generation of the paraphrase (S_2): *"She hands the editor the draft"*.
3. *Prevent over-generation*: *e.g.,* fail when adding the same `(dative-move yes)` feature to an input similar to I_1 except that the `possessed` role is filled by `((cat pers-pro))` (for personal pronoun) to avoid the generation of (S_3) * *"She hands the editor it"*.
4. *Provide defaults for syntactic features*: *e.g.,* `definite` for the NPs of S_1.
5. *Propagate agreement features*, providing enough input to the morphology module: *e.g.,* after the `agent` and `process` thematic roles have been mapped to the `subject` and `verb-group` syntactic roles (respectively), propagate the default `(person third)` feature added to the `subject` filler to the `verb-group` filler; without such a propagation the morphology module would not be able to inflect the verb *"to hand"* as *"hands"* in S_1.
6. *Select closed-class words*: *e.g., "she", "the"* and *"to"* in S_1.
7. *Provide linear precedence constraints* among syntactic constituents: *e.g.,* `subject` > `verb-group` > `indirect-object` > `direct-object` once the default active voice has been chosen for S_1.
8. *Inflect open-class words* (morphological processing): *e.g.,* the verb *"to hand"* as *"hands"* in S_1.
9. *Linearize the syntactic tree into a string of inflected words* following the linear precedence constraints.
10. *Perform syntactic inference*: As an example of syntactic inference, suppose that I_1 is embedded within a clause as follows:



I_1 fills the `soa (state-of-affair)` role of "to require" and the `influenced` role is co-referent with the `affected` role of I_1. From this co-reference constraint and the subcategorization constraint of "to require" (provided in the lexicon), SURGE deduces that the `affected` role of the subordinated clause should be *controlled* [28] (Chap. 7) by the `influenced` role of the matrix clause and therefore be mapped onto the `subject` syntactic role. This inferred constraint in turn triggers the choice of the passive voice for the subordinated clause, resulting in (S_4): *"They required the draft to be handed by her to the editor."*

Evaluation Criteria A re-usable syntactic realization component (or grammar for short) for language generation should thrive to satisfy the following properties:

1. *Comprehensive coverage*: it should be able to generate a large set of valid syntactic forms and to output many alternate syntactic forms from a single input.
2. *Prevention of over-generation*: it should generate only grammatical outputs (the counter-part of the previous criterion).
3. *Encapsulation of syntactic knowledge*: it should accept inputs that can be prepared by the client program or user with as little knowledge of syntax as possible.
4. *Regular input*: it should require inputs whose structure is well-defined and regular, and thus easily producible by a recursive client program.
5. *Partially canned input*: it should accept inputs where canned phrases can co-exist with individual words [31], to be usable across the granularity gradient from simple template systems to fully compositional generation systems.
6. *Compact input*: it should provide appropriate defaults for all syntactic features, so that the input can contain only the few features that have non-default values.
7. *Extensibility*: it should be easy to add new syntactic constructs, and quickly verify their interaction with the rest of the grammar.
8. *Efficiency*: it should generate a sentence from the input specification as fast as possible.
9. *Robustness*: it should be able to generate grammatical fragments from ill-formed or incomplete inputs.
10. *User-friendliness*: it should be easy to use for computational linguists not familiar with the implementation details and easy to learn for users with little knowledge of linguistic theory.
11. *Versatility*: it should be usable in a wide variety of system architectures and application domains.

We come back to these criterion in Section 6 when we compare SURGE with other syntactic realization components.

3 The FUF/SURGE package

SURGE is implemented in the special-purpose programming language FUF [4] [5] and it is distributed as a package with a FUF interpreter. This interpreter has two components:

- The *functional unifier* that fleshes out the input skeletal tree with syntactic features from the grammar.
- The *linearizer* that inflects each word at the bottom of the fleshed out tree and print them out following the linear precedence constraints indicated in the tree.

Underlying Formalism FUF is an extension of the original functional unification formalism put forward by Kay [14]. It is based on two powerful concepts: encoding knowledge in recursive sets of attribute value pairs called *Functional Descriptions* (FD) and uniformly manipulating these FDs through the operation of unification.

Both the input and the output of a FUF program are FDs, while the program itself is a meta-FD called a *Functional Grammar* (FG). An FG is an FD with disjunctions and control annotations. Control annotations are used in FUF for two distinct purposes:

- To control recursion on linguistic constituents: the skeletal tree of the input FD is fleshed out in top-down fashion by re-unifying each of its sub-constituent with the FG.
- To reduce backtracking when processing disjunctions.

The key extensions of FUF over Kay's original formalism are its wide range of control annotations [8] and its special constructs to define subsumption among atomic values [3]. The main advantages of FUF over PROLOG for natural language generation are:

- Partial knowledge assumption (a consequence of relying neither on predicate arity

nor on attribute order when encoding information in FDs).

- Default control flow adapted to generation (top-down breadth-first recursion on linguistic constituents) and possibility to fine-tune it (indexing on grammatical features and dependency-directed backtracking and goal freezing on uninstantiated features).
- Built-in linearizer and morphology component.

Linguistic Framework SURGE represents our own synthesis, within a single working system and computational framework, of the descriptive work of several (non-computational) linguists. We took inspiration principally:

- From [12] and [36] for the overall organization of the grammar and the core of the clause and nominal sub-grammars.
- From [9] and [20] for the semantic aspects of the clause.
- From [28] for the treatment of long-distance dependencies.
- From [29] for the many linguistic phenomena not mentioned in other works, yet encountered in many generation application domains.

Since many of these sources belong to the *systemic* linguistic school, SURGE is mostly a functional unification implementation of systemic grammar rules. In particular, the type of FD that it accepts as input specifies a "process" in the systemic sense: it can be an event, a relation, or state in addition to a "process" in its most common, aspectually restricted sense. The hierarchy of general process types defining the thematic structure of a clause (and the associated semantic class of its main verb) in the current implementation is compact and able to cover many clause structures. Yet, the argument structure and/or semantics of many English verbs do not fit neatly in any element of this hierarchy [19]. To overcome this difficulty, SURGE also includes *lexical processes* inspired by *lexicalist* grammars such as the Meaning-Text Theory [24] and HPSG [28].

A lexical process is a shallower and less semantic form of input, where the sub-categorization constraints and the mapping from the thematic roles to the oblique roles [28] are already specified (instead of being automatically computed by the grammar as is the case for general process types). The use of specific lexical processes to complement general process types is an example of the type of theory integration that we were forced to carry out during the development of SURGE. In the current state of linguistic research, with each theory focusing on a small set of linguistic phenomena, such an heterogeneous approach is the best practical strategy to provide broad coverage.

4 Organization and Coverage

At the top-level, SURGE is organized into sub-grammars, one for each syntactic category. Each sub-grammar encapsulates the relevant part of the grammar to access when recursively unifying an input sub-constituent of the corresponding category. For example, generating the sentence *"James buys the book"* involves successively accessing the sub-grammars for the clause, the verb group, the nominal group (twice) and the determiner sequence. Each sub-grammar is then divided into a set of *systems* (in the systemic sense), each one encapsulating an orthogonal set of decisions, constraints and features. The main top-level syntactic categories used in SURGE are: clause, nominal group (or NP), the determiner sequence, the verb group, the adjectival phrase and the PP. We now describe the systems and coverage currently implemented in SURGE for each of these categories.

4.1 The Clause

Following [12], the *thematic* roles accepted by SURGE in input clause specifications first divide into: *nuclear* and *satellite* roles. Nuclear roles, answer the questions "who/what was involved?" about the situation described by the clause. They include the *process* itself, generally

surfacing as the `verb`[2] and its associated *participants* surfacing as verb arguments. Satellite roles (also called *adverbials*) answer the questions "when/where/why/how did it happen?" and surface as the remaining clause complements.

Semantically, participants are tightly associated with specific nodes in the process type hierarchy, while satellites are versatile roles compatible with virtually any process type. Syntactically, participants can be distinguished from satellites in that[3]: (1) they surface as `subject` for at least one syntactic alternation [19], (2) they can neither be moved around in the clause nor (3) omitted from the clause while preserving its grammaticality.

Following this sub-division of thematic roles, the clause sub-grammar is divided into four orthogonal systems:

- *Transitivity*, which handles mapping of nuclear thematic roles onto a default core syntactic structure for main assertive clauses.
- *Voice*, which handles departures from the default core syntactic structure triggered by the use of syntactic alternations (*e.g.,* passive or dative moves).
- *Mood*, which handles departures from the default core syntactic structure triggered by variations in terms speech acts (*e.g.,* interrogative or imperative clause) and syntactic functions (*e.g.,* matrix *vs.* subordinate clause).
- *Adverbial*, which handles mapping of satellite roles onto the peripheral syntactic structure.

The Transitivity System Following [9], we distinguish among *simple* (either event or relation) and *composite* processes (di-transitive, complex-transitive [29] (p.721) and causative constructs). For example, we view the clause *"She gives it to him"* as the conflation of two simple processes: (1) a simple event *"She gives it"* and (2) the resulting possessive relation *"It belongs to him"*. The description of composite processes is useful to account for alternations like dative move [19] (pp.45-49) when the relation is `possessive, locative` (pp.49-55), `creation` (pp.55-58), `causative` (pp.25-32) and others.

The subtasks of the transitivity system are: (1) to provide an interface to the semantic encoding used in the client program, (2) to determine which combinations of predicates are mergeable as composite processes, (3) to constrain syntactic alternations and (4) to constrain the syntactic realization of each participant. SURGE currently covers 21 simple process types and 15 composite process types, thus accepting 36 different nuclear thematic structures as input (*cf.* [32] for examples of each type).

The Mood System The subtasks of the mood system are: (1) to provide an interface to the specification of speech acts and interpersonal constraints in the client program (*e.g.,* imperative mood to express a request to a subordinate), (2) to account for hypotactic relations among clauses (*e.g.,* subordination, embedding) and (3) to constrain to use of abbreviated forms (*e.g.,* participle and verbless clauses). The mood of dependent clauses is often inferred by SURGE from its syntactic function in the matrix and the head verb of the matrix. SURGE currently covers 15 different moods (*cf.* [32] for examples of each mood).

The Adverbial System The main tasks of the adverbial system are: (1) to provide an interface to the semantic encoding used in the client program, (2) to determine the relative ordering of adverbials within the clause, (3) to restrict possible co-occurrences of adverbials, (4) to restrict the syntactic realization of adverbials and (5) to provide default closed-class words (*e.g.,* prepositions such as "for" or subordinating conjunctions such as "when").

The adverbial system encodes a more subtly constrained mapping from thematic structures to syntactic roles than the transitivity system.

[2] Or as `direct-object` in the case of clauses headed by support-verbs [11].

[3] While each of these three tests has a number of exceptions, taken together they are very reliable.

While nuclear roles surface as mutually exclusive core syntactic functions (*e.g,* each clause can only have one subject and one direct object), satellite roles surface as one of three peripheral syntactic functions, *predicate adjuncts*, *sentence adjuncts* and *disjuncts* following [29] (pp. 504-505), with potentially multiple instances of each. The adjuncts *vs.* disjuncts distinction is purely syntactic and accounts for general alternations (*e.g.,* only adjuncts can be clefted or appear in alternative questions). The distinction between predicate and sentence adjunct is semantic: the former modifies only the *process* of the clause, while the latter elaborates on the entire situation described by the clause. It has subtle repercussions in terms of syntactic alternations, possible positions and co-occurrence (*e.g.,* predicate adjuncts cannot be fronted). SURGE reflects this distinction in input where satellite roles can be either `predicate-modifier` which get mapped onto predicate adjuncts and `circumstancials` which get mapped onto either sentence adjuncts or disjuncts (depending on semantic label, syntactic category and lexical content).

Integrating descriptions by [29], [34] [12], SURGE can currently map 13 predicate modifiers onto 26 syntactic realizations of predicate adjuncts and 34 circumstancials onto 32 syntactic realizations of sentence adjuncts and 36 syntactic realizations of disjuncts (*cf.* [32] for examples of each adverbial form).

4.2 Nominals

Nominals are an extremely versatile syntactic category, and except for limited cases (cf. [35], [18], [10]), no linguistic semantic classification of nominals has been provided. Consequently, while for clauses input can be provided in thematic form, for nominals it must be provided directly in terms of syntactic roles. The task of mapping domain-specific thematic relations to the syntactic slots in an NP is therefore left to the client program (*cf.* [6] for a discussion of this process).

Nominal Types SURGE distinguishes among the following nominal types: *common nouns, proper nouns, pronouns, measures, noun-compounds* and *partitives*. Noun-compounds can have a deep embedded structure. Measures have a very specific syntactic behavior (when used as classifiers the head noun of the measure does not take a number inflection, *e.g., "a 3 meter boat" vs. "the boat measures 3 meters"*).

NP Structure and Functions The specific syntactic roles accepted by SURGE for nominal description are (given in their partial order of precedence):
determiner-sequence, describer, classifier, head, and *qualifier*.

There are two types of pre-modifiers: describers, which can be also appear as subject complements (*e.g., "a blue book" = "the book is blue*) and classifiers which cannot [36], since they are an abbreviated form of an indirect relation [18] (*e.g., "a theory book"* $\neq$ *? "the book is theory"* but rather *"a theory book" = "the book is about theory"*). SURGE currently covers a total of 28 syntactic realizations for the 5 functions above (*cf.* [32] for examples of each).

Determiner Sequence The determiner-sequence is itself decomposed into the following elements:
`pre-determiner, determiner, ordinal, cardinal, quantifier`. This sub-grammar is special in that it is mainly a closed system, *i.e.,* all lexical differences must be determined by some configuration of syntactic features. The determiner sequence sub-grammar currently covers 24 features controlling 109 decision points (*cf.* [5] for examples).

4.3 The Verb Group

The verb group grammar decomposes in three major systems: *tense*, *polarity* and *modality*. SURGE implements the full 36 English tenses identified in [12] pp.198-207 It provides an interface to the client program is in terms Allen's temporal relations [2] (*e.g.,* to describe a past event, the client provides the feature `(tpattern (:et :before :st))`, specifying that the event time (et) precedes the speech time (st)).

5 Implemented Applications

SURGE has already been successfully re-used by at least[4] eight generators. The practical usability and versatility of SURGE is demonstrated by the fact that these systems differ widely in their application domains, their research emphasis and the strategy they use to generate the skeletal thematic trees that they pass on to SURGE.

COOK [33] generates stock market reports from semantic messages summarizing the daily fluctuations of several financial indexes. It focuses on the semi-automatic acquisition of a declarative lexicon including collocations and idioms statistically compiled from a large textual corpus. It composes a SURGE input by using FUF to unify this declarative lexicon with the input semantic message. It relies on a fixed, bottom-up control strategy driven by output syntactic function: first choose the verb arguments, then the verb and finally the adjuncts.

ADVISOR-II [5] generates query responses in an interactive student advising system. It focuses on presenting the same underlying content as argument towards different conclusions and on expressing the same piece of content across linguistic ranks (from complex clause down to the determiner sequence). It composes a SURGE input by using FUF to unify an input semantic network (enriched with argumentative constraints) with a declarative, word-based lexicon. It relies on a complex, input driven control strategy involving goal freezing and dependency-directed backtracking.

COMET [22] generates natural language explanations coordinated with graphics in an multimedia tutorial and troubleshooting system for a military radio. It focuses on combining an wide variety of constraints on lexical choice, including the accompanying illustration, the user's vocabulary and previous discourse in addition to the usual syntax and domain semantics. It composes a SURGE input by using FUF to unify text plan fragments with a word-based lexicon. It relies on a co-routine control strategy allowing the lexicon to request partial re-planning of textual fragments when it reaches an impasse while combining constraints.

STREAK [32] generates newswire style summaries of basketball games from game statistics and related historical data. It focuses on usage of concise linguistic forms, generation of very complex sentences, high paraphrasing power and empirical evaluation of architecture and knowledge structures based on corpus analysis. It incrementally composes the SURGE input using a revision-based control strategy. It first composes a draft of this input by using FUF to unify a semantic network of obligatory facts with a phrase planner and then a lexicon. This draft is then incrementally revised (at times non-monotonically) to opportunistically incorporate as many optional or background fact that can fit under corpus-observed space and linguistic complexity limits.

PLANDOC [16] generates documentation of telephone network extension and upgrade proposals by planning engineers, from the trace of the simulation system that they use to come-up with their proposals. Its focuses on high paraphrasing power and on aggregation of related semantic messages into complex clauses. It composes SURGE inputs using a strategy similar to ADVISOR-II, but using constraints derived from the extended discourse instead of argumentative orientation.

FLOWDOC [27] generates executive summaries of workflow diagrams acquired and displayed using the SHOWBIZ [37] graphical business re-engineering tool. It focuses on pointing out characteristics of the workflow relevant to re-engineering but obscured by the diagram complexity. It composes SURGE inputs using a strategy similar to STREAK's initial draft building stage, except for the combination of compositional generation from a word-based lexicon of general workflow terms together with canned phrases for task-specific operations (entered by the user during workflow acquisition).

KNIGHT [17] generates paragraph-long explanations of scientific processes from a large knowledge base in biology. It focuses on the robustness of the generator when used in a vast domain and

[4] It is also currently used by several other systems not mentioned here either because they are still in preliminary stages or because we have little information about them .

the evaluation of the produced output. It composes SURGE inputs procedurally by filling FD templates using information extracted from the knowledge base.

Abella's system [1] generates visual scene descriptions from camera input in two domains: map-aided navigation and kidney X-rays analysis. Its strategy to compose SURGE inputs is entirely geared toward its very specific research focus: validating a fuzzy lexical semantic model of English spatial prepositions. It thus first performs visual reasoning to pick the prepositions best describing the main spatial relations among the input picture landmarks and then procedurally fills locative clause FD templates associated with each chosen preposition.

6 Related Work

Three other available systems provide reusable surface realization components: MUMBLE [26], NIGEL [21] and the systems developed at Cogentex (e.g., [13]). These three systems differ from SURGE in that they are all developed within a single linguistic theory (TAGs for MUMBLE, systemic functional for NIGEL and Meaning-Text Theory (MTT) [24] for Cogentex's systems), whereas SURGE integrates several ones. Each system also puts the emphasis on different dimensions resulting in different strengths and weaknesses.

MUMBLE's determinism (motivated on cognitive grounds) makes it very efficient. Another of its strengths is the regular input provided by Meteer's text-structure [25]. NIGEL's strengths are comprehensive coverage and encapsulation of syntactic knowledge through the inquiry mechanism. However, the procedural implementation of both these systems make them weak on extensibility and user-friendliness. In addition, NIGEL also has a tendency to over-generate. This last weakness has however recently being turned into a strength by the addition of a post-processor that filters outputs using a statistical language model [15]. This post-processor allows NIGEL's output to nicely degrades in the face of ill-formed or incomplete input, making it more robust than other systems.

MTT-based syntactic realization components are strong on extensibility (declarative PROLOG implementation) and comprehensive coverage. But the fact that they must work in tandem with an MTT-based lexicon (the Explanatory Combinatorial Dictionary) make them weak on versatility and encapsulation of syntactic knowledge. SURGE's strengths are comprehensive coverage, encapsulation of syntactic knowledge, the extensibility and user-friendliness of its purely declarative implementation and the versatility of its compact, and regular input where individual words and canned phrases can co-exist. Its main weakness, inefficiency for complex sentences, is currently being addressed by the development of a graph-based, re-implementation the FUF interpreter in C to replace the current list-based implementation in LISP. We expect SURGE's run times to improve by an order of magnitude with this new interpreter. A more systematic use of control annotations in the grammar could also improve run time though less dramatically.

7 Future Work

The development of SURGE itself continues, as prompted by the needs of new applications, and by our better understanding of the respective tasks of syntactic realization and lexical choice [7]. We are specifically working on (1) integrating a more systematic implementation of Levin's alternations within the grammar, (2) extending composite processes to include mental and verbal ones, (3) modifying the nominal grammar to support nominalizations and some forms of syntactic alternations (relying on [10]) and (4) improving the treatment of obligatory pronominalization and binding. As it stands, SURGE provides a comprehensive syntactic realization component, easy to integrate within a wide range of architectures for complete generation systems. It is available on the WWW at `http://www.cs.bgu.ac.il/surge/`.

References

[1] A. Abella. *From imagery to salience: locative expressions in context.* PhD thesis, Computer Science Department, Columbia Universtity, New York, NY, 1994.

[2] J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[3] M. Elhadad. Types in functional unification grammars. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, Detroit, MI, 1990. ACL.

[4] M. Elhadad. Fuf: The universal unifier - user manual, version 5.2. Technical Report CUCS-038-91, Columbia University, 1993.

[5] M. Elhadad. *Using argumentation to control lexical choice: a unification-based implementation.* PhD thesis, Computer Science Department, Columbia University, 1993.

[6] M. Elhadad. Lexical choice for complex noun phrases. *Submitted to Journal of Machine Translation*, 1996.

[7] M. Elhadad, K. McKeown, and J. Robin. Floatings constraints in lexical choice. *Computational Linguistics*, 1996. To appear.

[8] M. Elhadad and J. Robin. Controlling content realization with functional unification grammars. In R. Dale, H. Hovy, D. Roesner, and O. Stock, editors, *Aspects of automated natural language generation*, pages 89–104. Springer Verlag, 1992.

[9] R. Fawcett. The semantics of clause and verb for relational processes in english. In M. Halliday and R. Fawcett, editors, *New developments in systemic linguistics.* Frances Pinter, London and New York, 1987.

[10] P. Fries. *Tagmeme sequences in the English noun phrase.* Number 36 in Summer institute of linguistics publications in linguistics and related fields. Benjamin F. Elson for The Church Press Inc., Glendale, CA, 1970.

[11] M. Gross. Lexicon-grammar and the syntactic analysis of french. In *Proceedings of the 10th International Conference on Computational Linguistics*, pages 275–282. COLING, 1984.

[12] M. Halliday. *An introduction to functional grammar.* Edward Arnold, London, 1994. 2nd Edition.

[13] L. Iordanskaja, M. Kim, R. Kittredge, B. Lavoie, and A. Polguère. Generation of extended bilingual statistical reports. In *Proceedings of the 15th International Conference on Computational Linguistics.* COLING, 1994.

[14] M. Kay. Functional grammar. In *Proceedings of the 5th Annual Meeting of the Berkeley Linguistic Society*, 1979.

[15] K. Knight and V. Hatzivassiloglou. Two-levels, many-paths generation. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 252–260, Boston, MA, 1995. ACL.

[16] K. Kukich, K. McKeown, J. Shaw, J. Robin, N. Morgan, and J. Phillips. User-needs analysis and design methodology for an automated document generator. In A. Zampolli, N. Calzolari, and M. Palmer, editors, *Current Issues in Computational Linguistics: In Honour of Don Walker.* Kluwer Academic Press, Boston, 1994.

[17] J. Lester. *Generating natural language explanations from large-scale knowledge bases.* PhD thesis, Computer Science Department, Universtity of Texas at Austin, New York, NY, 1994.

[18] J. Levi. *The syntax and semantics of complex nominals.* Academic Press, 1978.

[19] B. Levin. *English verb classes and alternations: a preliminary investigation.* University of Chicago Press, 1993.

[20] J. Lyons. *Semantics*. Cambridge University Press, 1977.

[21] C. Matthiessen. The organization of the environment of a text-generation grammar. In G. Kempen, editor, *Natural Language Generation: New Results in Artificial Intellligence, Psychology and Linguistics*. Martinus Ninjhoff Publishers, 1987.

[22] K. McKeown, J. Robin, and M. Tanenblatt. Tailoring lexical choice to the user's vocabulary in multimedia explanation generation. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*. ACL, 1993.

[23] K. R. McKeown, M. Elhadad, Y. Fukumoto, J. Lim, C. Lombardi, J. Robin, and F. Smadja. Text generation in comet. In R. Dale, C. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation*, pages 103–140. Academic Press, 1990.

[24] I. Mel'cuk and N. Pertsov. *Surface-syntax of English, a formal model in the Meaning-Text Theory*. Benjamins, Amsterdam/Philadelphia, 1987.

[25] M. Meteer. *Expressibility: The problem of efficient text planning*. Pinter, London, 1992.

[26] M. Meteer, D. McDonald, S. Anderson, D. Forster, L. Gay, A. Huettner, and P. Sibun. Mumble-86: Design and implementation. Technical Report COINS 87-87, University of Massachussets at Amherst, Ahmerst, Ma., 1987.

[27] R. Passoneau, K. Kukich, J. Robin, and L. Lefkowitz. Generating executive summaries of workflow diagrams, 1996. Submitted to the 8th International Workshop on Natural Language Generation, Herstmonceux, UK.

[28] C. Pollard and I. A. Sag. *Head Driven Phrase Structure Grammar*. University of Chicago Press, Chicago, 1994.

[29] R. Quirk, S. Greenbaum, G. Leech, and J. Svartvik. *A comprehensive grammar of the English language*. Longman, 1985.

[30] E. Reiter. Has a consensus natural language generation architecture appeared and is it psycholinguistically plausible? In *Proceedings of the 7th International Workshop on Natural Language Generation*, pages 163–170, June 1994.

[31] E. Reiter. Nlg vs templates. In *Proceedings of the 5th European Workshop on Natural-Language Generation (ENLGW-95)*, Leiden, The Netherlands, 1995.

[32] J. Robin. Revision-based generation of natural language summaries providing historical background: corpus-based analysis, design, implementation and evaluation. Technical Report CU-CS-034-94, Computer Science Department, Columbia Universtity, New York, NY, 1994. PhD. Thesis.

[33] F. Smadja and K. McKeown. Using collocations for language generation. *Computational Intelligence*, 7(4):229–239, December 1991.

[34] S. Thompson and R. Longacre. Adverbial clauses. In T. Shopen, editor, *Complex constructions*, volume 2 of *Language typology and syntactic description*. Cambridge University Press, 1985.

[35] Z. Vendler. *Adjectives and Nominalizations*. Mouton, The Hague, Paris, 1968.

[36] T. Winograd. *Language as a cognitive process*. Addison-Wesley, 1983.

[37] K. Wittenburg. Visual language parsing: if i has a hammer ... In *Proceedings of the International Conference on Cooperative Multimodal Communication*, pages 17–33, Eindhoven, The Netherlands, 1995.

[38] G. Yang, K. McCoy, and K. Vijay-Shanker. From functional specification to syntactic structure: systemic grammar and tree adjoining grammars. *Computational Intelligence*, 7(4), December 1991.