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

Jacques Robin
jr@di.ufpe.br
Departamento de Informática
Universidade Federal de Pernambuco
Recife, PE 50740-540 Brazil

1 Introduction

This paper describes a short demo providing 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 reused 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. We then briefly survey the computational formalism underlying the implementation of SURGE as well as the syntactic theories that it integrates. We then describe the structure of the grammar.

[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

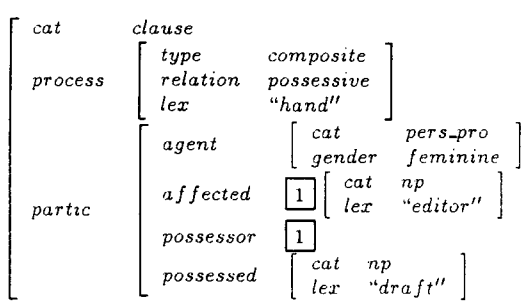
Natural language generation has been traditionally divided into three successive tasks: (1) content determination, (2) content organization, and (3) linguistic realization. The goal of a re-usable realization component is to encapsulate the domain-independent part of this third task. The input to such component should thus be as high-level as possible without hindering portability. Independent efforts to define such an input 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.

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 al-*

Input Specification (I_1):



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

Figure 1: An example SURGE I/O

ternations [6]: *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, 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.

3 The FUF/SURGE package

SURGE is implemented in the special-purpose programming language FUF [1] and it is distributed as a package with a FUF interpreter. This interpreter has two components: (1) the *functional unifier* that fleshes out the input skeletal tree with syntactic features from the grammar, and (2) 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.

FUF is an extension of the original functional unification formalism put forward by Kay [5]. 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: (1) to control recursion on linguistic constituents: the tree of the input FD is fleshed out in top-down fashion by re-unifying each of its sub-constituent with the FG, and (2) to reduce backtracking when processing disjunctions.

SURGE represents our own synthesis, within a single working system and computational framework, of the descriptive work of several (non-computational) lin-

guists. We took inspiration principally from [4] for the overall organization of the grammar and the core of the clause and nominal sub-grammars; [3] for the semantic aspects of the clause; [7] for the treatment of long-distance dependencies: and [8] 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. In particular, the type of FD that it accepts as input specifies a "process" in the systemic sense: it can be an event or a relation. 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 [6]. To overcome this difficulty, SURGE also includes *lexical processes* inspired by *lexicalist* grammars such as the Meaning-Text Theory and HPSG [7].

A lexical process is a shallower and less semantic form of input, where the subcategorization constraints and the mapping from the thematic roles to the oblique roles [7] 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, 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), determiner sequence, verb group, adjectival phrase and PP.

Following [4], 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** 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.

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

(1) *Transitivity*, which handles mapping of nuclear thematic roles onto a default core syntactic structure for main assertive clauses.

(2) *Voice*, which handles departures from the default core syntactic structure triggered by the use of syntactic alternations (*e.g.*, passive or dative moves).

(3) *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).
(4) *Adverbial*, which handles mapping of satellite roles onto the peripheral syntactic structure.

Nominals are an extremely versatile syntactic category, and except for limited cases, 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.

The verb group grammar decomposes in three major systems: *tense*, *polarity* and *modality*. SURGE implements the full 36 English tenses identified in [4] pp.198-207 It provides an interface to the client program is in terms Allen's temporal relations (*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 Current 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 [2]. 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 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] M. Elhadad. *Using argumentation to control lexical choice: a unification-based implementation.* PhD thesis, Computer Science Department, Columbia University, 1993.

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

[3] 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.

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

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

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

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

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