Integrating a Large-scale, Reusable Lexicon with a Natural Language Generator

Hongyan Jing
Department of Computer Science
Columbia University
New York, NY 10027, USA
hjing@cs.columbia.edu

Yael Dahan Netzer
Department of Computer Science
Ben-Gurion University
Be'er-Sheva, 84105, Israel
yaeln@cs.bgu.ac.il

Michael Elhadad
Department of Computer Science
Ben-Gurion University
Be'er-Sheva, 84105, Israel
elhadad@cs.bgu.ac.il

Kathleen R. McKeown
Department of Computer Science
Columbia University
New York, NY 10027, USA
kathy@cs.columbia.edu

Abstract

This paper presents the integration of a large-scale, reusable lexicon for generation with the FUF/SURGE unification-based syntactic realizer. The lexicon was combined from multiple existing resources in a semi-automatic process. The integration is a multi-step unification process. This integration allows the reuse of lexical, syntactic, and semantic knowledge encoded in the lexicon in the development of lexical chooser module in a generation system. The lexicon also brings other benefits to a generation system: for example, the ability to generate many lexical and syntactic paraphrases and the ability to avoid non-grammatical output.

1 Introduction

Natural language generation requires lexical, syntactic, and semantic knowledge in order to produce meaningful and fluent output. Such knowledge is often hand-coded anew when a different application is developed. We present in this paper the integration of a large-scale, reusable lexicon with a natural language generator, FUF/SURGE (Elhadad, 1992; Robin, 1994); we show that by integrating the lexicon with FUF/SURGE as a tactical component, we can reuse the knowledge encoded in the lexicon and automate to some extent the development of the lexical realization component in a generation application.

The integration of the lexicon with FUF/SURGE also brings other benefits to generation, including the possibility to accept a semantic input at the level of WordNet synsets, the production of lexical and syntactic paraphrases, the prevention of non-grammatical output, reuse across applications, and wide coverage.

We present the process of integrating the lexicon with FUF/SURGE, including how to represent the lexicon in FUF format, how to unify input with the lexicon incrementally to generate more sophisticated and informative representations, and how to design an appropriate semantic input format so that the integration of the lexicon and FUF/SURGE can be done easily.

This paper is organized as follows. In Section 2, we explain why a reusable lexical chooser for generation needs to be developed. In Section 3, we present the large-scale, reusable lexicon which we combined from multiple resources, and illustrate its benefits to generation by examples. In Section 4, we describe the process of integrating the lexicon with FUF/SURGE, which includes four unification steps, with each step adding additional lexical or syntactic information. Other applications and comparison with related work are presented in Section 5. Finally, we conclude by discussing future work.

2 Building a reusable lexical chooser for generation

While reusable components have been widely used in generation applications, the concept of a "reusable lexical chooser" for generation remains novel.

There are two main reasons why such a lexical chooser has not been developed in the past:

1. In the overall architecture of a generator, the lexical chooser is an internal component that depends on the semantic representation and formalism and on the syntactic realizer used by the application.
2. The lexical chooser links conceptual elements to lexical items. Conceptual elements are by definition domain and application dependent (they are the primitive concepts used in an application knowledge base). These primitives are not easily ported from application to application.

The emergence of standard architectures for generators (RAGS, (Reiter, 1994)) and the possibility to use a standard syntactic realizer answer the first issue.

To address the second issue, one must realize that if the *whole* lexical chooser can not be made domain-independent, major parts can be made reusable. The main argument is that *lexical knowledge is modular*. Therefore, while choice of words is constrained by domain-specific conceptual knowledge (what information the sentences are to represent) on the one hand, it is also affected by several other dimensions:

- inter-lexical constraints: collocations among words
- pragmatic constraints: connotations of words
- stylistic constraints: familiarity of words
- syntactic constraints: government patterns of words, *e.g.*, thematic structure of verbs.

We show in this paper how the separation of the syntactic and conceptual interfaces of lexical item definitions allows us to reuse a large amount of lexical knowledge across applications.

3 The lexicon and its benefits to generation

3.1 A large-scale, reusable lexicon for generation

Natural Language generation starts from semantic concepts and then finds words to realize such semantic concepts. Most existing lexical resources, however, are indexed by words rather than by semantic concepts. Such resources, therefore, can not be used for generation directly. Moreover, generation needs different types of knowledge, which typically are encoded in different resources. However, the different representation formats used by these resources make it impossible to use them simultaneously in a single system.

To overcome these limitations, we built a large-scale, reusable lexicon for generation by combining multiple existing resources. The resources that are combined include:

- The WordNet Lexical Database (Miller et al., 1990). WordNet is the largest lexical database to date, consisting of over 120,000 unique words (version 1.6). It also encodes many types of lexical relations between words, including synonymy, antonymy, and many more.
- English Verb Classes and Alternations (EVCA) (Levin, 1993). It categorized 3,104 verbs into classes based on their syntactic properties and studied verb alternations. An alternation is a variation in the realization of verb arguments. For example, the alternation "there-insertion" transforms *A ship appeared on the horizon* to *There appeared a ship on the horizon*. A total of 80 alternations for 3,104 verbs were studied.
- The COMLEX syntax dictionary (Grishman et al., 1994). COMLEX contains syntactic information for over 38,000 English words.
- The Brown Corpus tagged with WordNet senses (Miller et al., 1993). We use this corpus for frequency measurement.

In combining these resources, we focused on verbs, since they play a more important role in deciding sentence structures. The combined lexicon includes rich lexical and syntactic knowledge for 5,676 verbs. It is indexed by WordNet synsets(which are at the semantic concept level) as required by the generation task. The knowledge in the lexicon includes:

- A complete list of subcategorizations for each sense of a verb.
- A large variety of alternations for each sense of a verb.
- Frequency of lexical items and verb subcategorizations in the tagged Brown corpus
- Rich lexical relations between words

The sample entry for the verb *"appear"* is shown in Figure 1. It shows that the verb *appear* has eight senses (the sense distinctions come from WordNet). For each sense, the lexicon lists all the applicable subcategorization for that particular sense of the verb. The subcategorizations are represented using the same format as in COMLEX. For each sense, the lexicon also lists applicable alternations, which we encoded based on the information in EVCA. In addition, for each subcategorization and alternation, the lexicon lists the semantic category constraints on verb arguments. In the figure, we omitted the frequency information derived from Brown Corpus and lexical relations (the lexical relations are encoded in WordNet).

The construction of the lexicon is semi-automatic. First, COMLEX and EVCA were merged, producing a list of syntactic subcategorizations and alternations for *each verb*. Distinctions in these syntactic restrictions according to each *sense* of a verb are achieved in the second stage, where WordNet is merged with the result of the first step. Finally, the corpus information is added, complementing the static resources with actual usage counts for each syntactic pattern. For a detailed description of the combination process, refer to (Jing and McKeown, 1998).

```
appear:
sense 1 give an impression
((PP-TO-INF-RS :PVAL ("to") :SO ((sb, —)))
(TO-INF-RS :SO ((sb, —)))
(NP-PRED-RS :SO ((sb, —)))
(ADJP-PRED-RS :SO ((sb, —) (sth, —)))))
sense 2 become visible
((PP-TO-INF-RS :PVAL ("to")
    :SO ((sb, —) (sth, —)))
...
(INTRANS THERE-V-SUBJ
         :ALT there-insertion
         :SO ((sb, —) (sth, —))))
...
sense 8 have an outward expression
((NP-PRED-RS :SO ((sth, —)))
(ADJP-PRED-RS :SO ((sb, —) (sth, —))))
```

Figure 1: Lexicon entry for the verb *appear*

3.2 The benefits of the lexicon

There are a number of benefits that this combined lexicon can bring to language generation.

First, the use of synsets as semantic tags can help map an application conceptual model to lexical items. Whenever application concepts are represented at the abstraction level of a WordNet synset, they can be directly accepted as input to the lexicon. By this way, the lexicon can actually lead to the generation of many *lexical paraphrases*. For example, {*look, seem, appear*} is a WordNet synset; it includes a list of words that can convey the semantic concept `''give an impression of''`. We can use synsets to find words that can lexicalize the semantic concepts in the semantic input. By choosing different words in a synset, we can therefore generate lexical paraphrases. For instance, using the above synset, the system can generate the following paraphrases:

> *"He seems happy."*
> *"He looks happy."*
> *"He appears happy."*

Secondly, the subcategorization information in the lexicon prevents generating a non-grammatical output. As shown in Figure 1, the lexicon lists applicable subcategorizations for each sense of a verb. It will not allow the generation of sentences like

> *"*He convinced me in his innocence"* (wrong preposition)
> *"*He convinced to go to the party"* (missing object)
> *"*The bread cuts"* (missing adverb (*e.g., "easily"*))
> *"*The book consists three parts"* (missing preposition)

In addition, alternation information can help generate *syntactic paraphrases*. For instance, using the "simple reciprocal intransitive" alternation, the system can generate the following syntactic paraphrases:

> *"Brenda agreed with Molly."*
> *"Brenda and Molly agreed."*
> *"Brenda and Molly agreed with each other."*

Finally, the corpus frequency information can help the lexical choice process. When multiple words can be used to realize a semantic concept, the system can use corpus frequency information in addition to other constraints to choose the most appropriate word.

The knowledge encoded in the lexicon is general, thus it can be used in different applications. The lexicon has wide coverage: the final lexicon consists of 5,676 verbs in total, over 14,100 senses (on average 2.5 senses/verb), and over 11,000 semantic concepts (synsets). It uses 147 patterns to represent the subcategorizations and includes 80 alternations.

To exploit the lexicon's many benefits, its format must be made compatible with the architecture of a generator. We have integrated the lexicon with the FUF/SURGE syntactic realizer to form a combined lexico-grammar.

4 Integration Process

In this section, we first explain how lexical choosers are interfaced with FUF/SURGE. We then describe step by step how the lexicon is integrated with FUF/SURGE and show that this integration process helps to automate the development of a lexical realization component.

4.1 FUF/SURGE and the lexical chooser

FUF (Elhadad, 1992) uses a functional unification formalism for generation. It unifies the input that a user provides with a grammar to generate sentences. SURGE (Elhadad and Robin, 1996) is a comprehensive English Grammar written in FUF. The role of a lexical realization component is to map a semantic representation drawn from the application domain to an input format acceptable by SURGE, adding necessary lexical and syntactic information during this process.

Figure 2 shows a sample semantic input (a), the lexicalization module that is used to map this semantic input to SURGE input (b), and the final SURGE input (c) — taken from a real application system(Passoneau et al., 1996). The functions of the lexicalization module include selecting words that can be used to realize the semantic concepts in the input, adding syntactic features, and mapping the arguments in the semantic input to the thematic roles in SURGE.

Sentence: *It has 24 activities, including 20 tasks and four decisions.*

$$\begin{bmatrix} \text{concept} & \text{total-node-count} \\ \text{args} & \begin{bmatrix} \text{theme} & \begin{bmatrix}\text{concept} & \text{process-flowgraph}\\ \text{ref} & \text{pronoun}\end{bmatrix} \\ \text{rheme} & \begin{bmatrix} \text{concept} & \text{elaboration} \\ \text{args} & \begin{bmatrix} \text{theme} & \begin{bmatrix} \text{concept} & \text{cardinality}\\ \text{args} & \begin{bmatrix}\text{theme} & [1] \\ \text{value} & [2]\end{bmatrix}\end{bmatrix} \\ \text{expansion} & \begin{bmatrix}\text{concept} & \text{subset-node-count}\\ \text{args} & \ldots\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

$$[1] = \begin{bmatrix}\text{concept} & \text{flownode}\\ \text{ref} & \text{full}\end{bmatrix}$$

$$[2] = \begin{bmatrix}\text{concept} & \text{cardinal}\\ \text{cardinal} & 24\\ \text{ref} & \text{full}\end{bmatrix}$$

(a) The semantic input (i.e., input of lexicalization module)

$$\begin{bmatrix} \text{concept} & \#\text{(under total-node-count)} \\ \text{proc} & \begin{bmatrix}\text{type} & \text{possessive}\end{bmatrix} \\ \text{partic} & \begin{bmatrix} \text{possessor} & \begin{bmatrix}\text{cat} & \text{pronoun}\end{bmatrix} \\ \text{possessed} & \begin{bmatrix} \text{cat} & \text{common} \\ \text{cardinal} & \begin{bmatrix}\text{value} & [1]\end{bmatrix} \\ \text{definite} & \text{no} \\ \text{head} & \begin{bmatrix}\text{lex} & \text{“activity”}\end{bmatrix} \\ \text{qualifier} & \begin{bmatrix} \text{cat} & \text{clause} \\ \text{mood} & \text{present-participle} \\ \text{proc} & \begin{bmatrix}\text{type} & \text{locative}\\ \text{lex} & \text{“include”}\end{bmatrix} \\ \text{partic} & \begin{bmatrix}\text{location} & \begin{bmatrix}\text{cat} & \text{np}\\ \ldots & \ldots\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

(b) The lexicalization module

$$\begin{bmatrix} \text{cat} & \text{clause} \\ \text{proc} & \begin{bmatrix}\text{type} & \text{possessive}\end{bmatrix} \\ \text{partic} & \begin{bmatrix} \text{possessor} & \begin{bmatrix}\text{cat} & \text{pronoun}\end{bmatrix} \\ \text{possessed} & \begin{bmatrix} \text{cat} & \text{common} \\ \text{cardinal} & \begin{bmatrix}\text{value} & 24\end{bmatrix} \\ \text{definite} & \text{no} \\ \text{head} & \begin{bmatrix}\text{lex} & \text{“activity”}\end{bmatrix} \\ \text{qualifier} & \begin{bmatrix} \text{cat} & \text{clause} \\ \text{mood} & \text{present-participle} \\ \text{proc} & \begin{bmatrix}\text{type} & \text{locative}\\ \text{lex} & \text{“include”}\end{bmatrix} \\ \text{partic} & \begin{bmatrix}\text{location} & \begin{bmatrix}\text{cat} & \text{np}\\ \ldots & \ldots\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

(c) The SURGE input (i.e., output of lexicalization module)

Figure 2: A sample lexicalization component

The development of the lexicalizer component was done by hand in the past. Furthermore, for each new application, a new lexicalizer component had to be written despite the fact that some lexical and syntactic information is repeatedly used in different applications. The integration process we describe, however, partially automates this process.

4.2 The integration steps

The integration of the lexicon with FUF/SURGE is done through incremental unification, using four unification steps as shown in Figure 3. Each step adds information to the semantic input, and at the end of the four unification steps, the semantic input has been mapped to the SURGE input format.

(1) The semantic input

Different generation systems usually use different representation formats for semantic input. Some systems use case roles ; some systems use flat attribute-value representation (Kukich et al., 1994). For the integrated lexicon and FUF/SURGE package to be easily pluggable in applications, we need to define a standard semantic input format. It should be designed in such a way that applications can easily adapt their particular semantic inputs to this standard format. It should also be easily mapped to the SURGE input format.

In this paper, we only consider the issue of semantic input format for the expression of the predicate-argument relation. Two questions need to be answered in the design of the standard semantic input format: one, how to represent semantic concepts; and two, how to represent the predicate-argument relation.

We use WordNet synsets to represent semantic concepts. The input can refer to synsets in several ways: either using a globally unique synset number[1] or by specifying a word and its sense number in WordNet.

The representation of verb arguments is a more complicated issue. Case roles are frequently used in generation systems to represent verb arguments in semantic inputs. For example, (Dorr et al., 1998) used 20 case roles in their lexical conceptual structure corresponding to underlying positions in a compositional lexical structure. (Langkilde and Knight, 1998) use a list of case roles in their interlingua representations.

We decided to use numbered arguments (similar to the DSyntR in MTT (Mel'cuk and Perstov, 1987)) instead of case roles. The difference between the two is not critical but the numbered argument approach avoids the need to commit the lexicon to a specific ontology and seems to be easier to learn[2].

Figure 4 shows a sample semantic input. For easy understanding, we refer to the semantic concepts using their definitions rather than numerical index numbers. There are two arguments in the input. The intended output sentence for this semantic input is *"A boat appeared on the horizon"* or its paraphrases.

(2) Lexical unification

In this step, we map the semantic concepts in the semantic input to concrete words. To do this, we use the synsets in WordNet. All the words in the same synset can be used to convey the same semantic concept. For the above example, the semantic concepts *"become visible"* and *"a small vessel for travel on water"* can be realized by the the verb *appear* and the noun *boat* respectively. This is the step that can produce lexical paraphrases. Note that when the system chooses a word, it also determines the particular sense number of the word, since a word as it belongs to a synset has a unique sense number in WordNet.

We represented all the synsets in Wordnet in FUF format. Each synset includes its numerical index number and the list of word senses included in the synsets. This lexical unification works for both nouns and verbs.

(3) Structural unification

After the system has chosen a verb (actually a particular sense of a verb), it uses that information as an index to unify with the subcategorization and alternations the particular verb sense has. This step adds additional syntactic information to the original input and has the capacity to produce syntactic paraphrases using alternation information.

(4) Constraints on the number of arguments

Next, we use the constraints that a subcategorization has on the number of arguments it requires to restrict unification with subcategorization patterns. We use 147 possible patterns. For example, the input in Figure 4 has two arguments. Although *INTRANS* (meaning intransitive) is listed as a possible subcategorization pattern for *"appear"* (see sense 2 in Figure 1), the input will fail to unify with it since *INTRANS* requires a single argument only. This prevents the generation of non-grammatical sentences. This step adds a feature which specifies the transitivity of the verb to FUF/SURGE input, selecting one from the lexicon when there is more than one possibility for the given verb.

[1] Since there are a huge number of synsets in WordNet, we will provide a searchable database of synsets so that users can look up a synset and its index number easily. For a particular application, users can adapt the synsets to their specific domain, such as removing non-relevant synsets, merging synsets, and relabeling the synsets for convenience, as discussed in (Jing, 1998).

[2] The difference between numbered arguments and labeled roles is similar to that between named semantic primitives and synsets in WordNet. Verb classes share the same definition of which argument is denoted by 1, 2 etc. if they share some syntactic properties as far as argument taking properties are concerned.

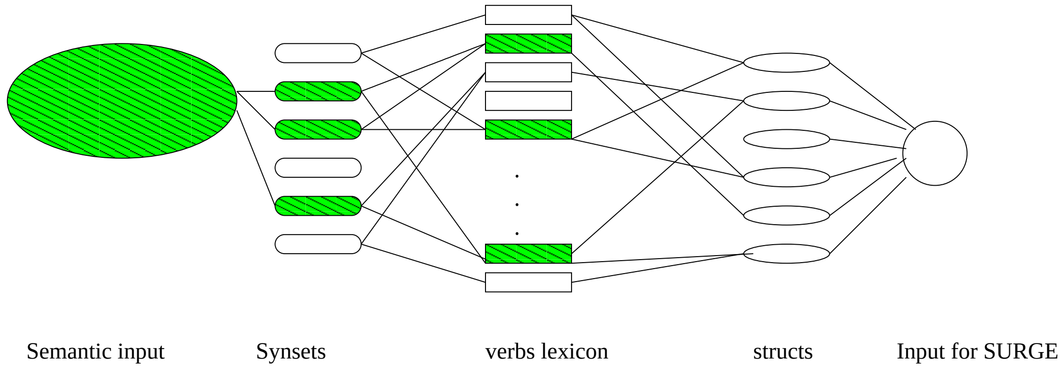

Figure 3: The integration process

$$\left[\begin{array}{ll} \text{relation} & \left[\begin{array}{ll}\text{concept} & \texttt{``become visible''}\end{array}\right] \\ \text{args} & \left[\begin{array}{ll} 1 & \left[\begin{array}{ll}\text{concept} & \texttt{``a small vessel for travel on water''}\end{array}\right] \\ 2 & \left[\begin{array}{ll}\text{concept} & \texttt{``the line at which the sky and Earth appear to meet''}\end{array}\right] \end{array}\right] \end{array}\right]$$

Figure 4: The semantic input using numbered arguments

(5) Mapping structures to SURGE input

In the last step, the subcategorization and alternations are mapped to SURGE input format. The mapping from subcategorizations to SURGE input was manually encoded in the lexicon for each one of the 147 patterns. This mapping information can be reused for all applications, which is more efficient than composing SURGE input in the lexicalization component of each different application. Figure 5 shows how the subcategorization NP-WITH-NP (*e.g.*, *The clown amused the children with his antics*) is mapped to the SURGE input format. This mapping mainly involves matching the numbered arguments in the semantic input to appropriate lexical roles and syntactic categories so that FUF/SURGE can generate them in the correct order.

The final SURGE input for the sentence *"A boat appeared on the horizon"* is shown in Figure 6. Using the "THERE-INSERTION" alternation that the verb *"appear"* (sense 2) authorizes, the system can also generate the syntactic paraphrase *"There appeared a boat on the horizon"*. The SURGE input the system generates for *"There appeared a boat on the horizon"* is very different from that for *"A boat appeared on the horizon"*.

It is possible that for a given application some generated paraphrases are not appropriate. In this case, users can edit the synsets and the alternations to filter out the paraphrases they do not want.

The four unification steps are completely automatic. The system can send feedback upon failure

$$\left[\begin{array}{ll} \text{struct} & \text{np-with-np} \\ \text{relation} & \left[\begin{array}{ll}\text{word} & [1]\langle\ldots\rangle\end{array}\right] \\ \text{args} & \left[\begin{array}{ll} 1 & [2]\langle\ldots\rangle \\ 2 & [3]\langle\ldots\rangle \\ 3 & [4]\langle\ldots\rangle \end{array}\right] \\ \text{proc} & \left[\begin{array}{ll} \text{type} & \text{lexical} \\ \text{lex} & [1] \\ \text{subcat} & \left[\begin{array}{ll} 1 & \left[\begin{array}{ll}\text{cat} & \text{np}\end{array}\right] \\ 1 & [2] \\ 2 & \left[\begin{array}{ll}\text{cat} & \text{np}\end{array}\right] \\ 2 & [3] \\ 3 & \left[\begin{array}{ll} \text{cat} & \text{pp} \\ \text{prep} & \left[\begin{array}{ll}\text{lex} & \text{``with''}\end{array}\right] \\ \text{np} & [4] \end{array}\right] \end{array}\right] \end{array}\right] \\ \text{lex-roles} & \left[\begin{array}{ll} 1 & [2] \\ 2 & [3] \\ 3 & [4] \end{array}\right] \end{array}\right]$$

Figure 5: Mapping subcategorization "NP-WITH-NP" to SURGE input

of unification.

5 Related Work

The lexicon, after it is integrated with FUF/SURGE, can also be used for other tasks in language generation. For example, revision (Robin, 1994) is a technique for building semantic inputs incrementally. The revision process decides whether it is appropriate to attach a new constituent to the current semantic input, for example, by adding an

$$\begin{bmatrix} \text{relation} & \begin{bmatrix} \text{concept} & \texttt{``become visible''} \\ \text{word} & \textit{``appear''}^{a} \end{bmatrix} \\ \text{args} & \begin{bmatrix} 1 & \begin{bmatrix} \text{concept} & \texttt{``a small vessel for travel on water''} \\ \text{word} & \textit{``boat''}^{a} \end{bmatrix} \\ 2 & \begin{bmatrix} \text{concept} & \texttt{``the line at which the sky and Earth appear to meet} \\ \text{word} & \textit{``horizon''}^{a} \end{bmatrix} \end{bmatrix} \\ \text{struct} & \text{PP}^{b} \\ \text{arg1} & \begin{bmatrix} 1 & \text{given} \\ 2^{c} & \text{given} \end{bmatrix} \\ \text{cat} & \text{clause}^{c} \\ \text{lexical-roles} & \ldots^{d} \end{bmatrix}$$

[a]Enriched in first step
[b]Enriched in second step
[c]Enriched in third step
[d]Enriched in fourth step

Figure 6: SURGE input for “A boat appeared on the horizon”

object or an adverb. Such decisions are constrained by syntactic properties of verbs. The integrated lexicon is useful to verify these properties.

Nitrogen (Langkilde and Knight, 1998), a natural language generation system developed at ISI, also includes a large-scale lexicon to support the generation process. Given that Nitrogen and FUF/SURGE use very different methods for generation, the way that we integrate the lexicon with the generation system is also very different. Nitrogen combines symbolic rules with statistics learned from text corpora, while FUF/SURGE is based on Functional Unification Grammar. Other related work includes (Stede, 1998), which suggests a lexicon structure for multilingual generation in a knowledge-based generation system. The main idea is to handle multilingual generation in the same way as paraphrasing of the same language. Stede's work concerns mostly the lexical semantics of the transitivity alternations.

6 Conclusion

We have presented in this paper the integration of a large-scale, reusable lexicon for generation with FUF/SURGE, a unification-based natural language generator. This integration makes it possible to reuse major parts of a lexical chooser, which is the component in a generation system that is responsible for mapping semantic inputs to surface generator inputs. We show that although the *whole* lexical chooser can not be made domain-independent, it is possible to reuse a large amount of lexical, syntactic, and semantic knowledge across applications.

In addition, the lexicon other benefits to a generation system, including the abilities to generate many lexical paraphrases automatically, generate syntactic paraphrases, avoid non-grammatical output, and choose the most frequently used word when there is more than one candidate words. Since the lexical, syntactic, and semantic knowledge encoded in the lexicon is general and the lexicon has a wide coverage, it can be reused for different applications.

In the future, we plan to validate the paraphrases the lexicon can generate by asking human subjects to read the generated paraphrases and judge whether they are acceptable. We would like to investigate ways that can systematically filter out paraphrases that are considered unacceptable. We are also interested in exploring the usage of this system in multilingual generation.

References

B. J. Dorr, N. Habash, and D. Traum. 1998. A thematic hierarchy for efficient generation from lexical-conceptual. Technical Report CS-TR-3934, Institute for Advanced Computer Studies, Department of Computer Science, University of Maryland, October.

M. Elhadad and J. Robin. 1996. An overview of SURGE: a re-usable comprehensive syntactic realization component. In *INLG'96*, Brighton, UK. (demonstration session).

M. Elhadad. 1992. *Using Argumentation to Control Lexical Choice: A Functional Unification-Based Approach.* Ph.D. thesis, Department of Computer Science, Columbia University.

R. Grishman, C. Macleod, and A. Meyers. 1994. COMLEX syntax: Building a computational lexicon. In *Proceedings of COLING'94*, Kyoto, Japan.

H. Jing and K. McKeown. 1998. Combining multiple, large-scale resources in a reusable lexicon for natural language generation. In *Proceedings* *of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics*, volume *1*, pages 607–613, Université de Montréal, Quebec, Canada, August.

H. Jing. 1998. Applying wordnet to natural language generation. In *Proceedings of COLING-ACL'98 workshop on the Usage of WordNet in Natural Language Processing Systems*, University of Montreal, Montreal, Canada, August.

K. Kukich, K. McKeown, J. Shaw, J. Robin, N. Morgan, and J. Phillips. 1994. 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.

I. Langkilde and K. Knight. 1998. The practical value of n-grams in generation. In *INLG'98*, pages 248–255, Niagara-on-the-Lake, Canada, August.

B. Levin. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press, Chicago, Illinois.

I.A. Mel'cuk and N.V. Perstov. 1987. *Surface-syntax of English, a formal model in the Meaning Text Theory*. Benjamins, Amsterdam/Philadelphia.

G. Miller, R. Beckwith C. Fellbaum, and D. Gross K. Miller. 1990. Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography (special issue)*, **3**(4):235–312.

G.A. Miller, C. Leacock, R. Tengi, and R.T. Bunker. 1993. A semantic concordance. Cognitive Science Laboratory, Princeton University.

R. Passoneau, K. Kukich, J. Robin, V. Hatzivassiloglou, L. Lefkowitz, and H. Jing. 1996. Generating summaries of workflow diagrams. In *Proceedings of the International Conference on Natural Language Processing and Industrial Applications (NLP-IA'96)*, Moncton, New Brunswick, Canada.

E. Reiter. 1994. Has a consensus nl generation architecture appeared, and is it psycholinguistically plausible? In *Proceedings of the Seventh International Workshop on Natural Language Generation (INLGW-1994)*, pages 163–170, Kennebunkport, Maine, USA. available from the cmp-lg archive as paper cmp-lg/9411032.

J. Robin. 1994. *Revision-Based Generation of Natural Language Summaries Providing Historical Background: Corpus-Based Analysis, Design, Implementation, and Evaluation*. Ph.D. thesis, Department of Computer Science, Columbia University. Also Technical Report CU-CS-034-94.

M. Stede. 1998. A generative perspective on verb alternations. *Computational Linguistics*, 24(3):401–430, September.