DSG/TAG: An appropriate grammatical formalism for flexible sentence generation

Raymond Kozlowski
Department of Computer and Information Sciences
University of Delaware
Newark, DE 19716, USA
kozlowsk@cis.udel.edu

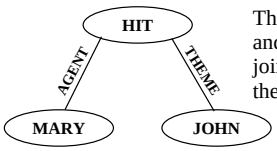
Abstract

We have developed a simple sentence generation architecture flexible enough to handle diverse lexical and grammatical forms of expression in a uniform manner. Central in our approach are lexico-grammatical resources that pair elementary semantic structures with their syntactic realization and all syntactic consequences of the choice. In this paper, we discuss one grammatical formalism that meets the significant demands put by the architecture.

1 Introduction

Sentence generation takes as input some semantic representation of the meaning to be conveyed and produces (one of) a set of grammatical sentences whose meaning matches the original input. The form of the input we use is a hierarchical predicate/argument structure such as that shown in Fig. 1[1]. A typical sentence generation system starts with the top predicate (here: HIT) and realizes it with a main verb (e.g. *hit*) which sets up a syntactic context into which the realizations of the other portions of the input are fit. In this case, with the active voice, the realizations of the agent and theme arguments of HIT, *Mary* and *John*, are placed in the subject and complement positions.

[1] Augmenting such an input with pragmatic information would be consistent with our architecture as long as the input remained hierarchical.



The input consists of the predicate HIT and two arguments MARY and JOHN joined with the predicate HIT using the thematic roles AGENT and THEME.

Figure 1: The semantic input for *Mary hit John*

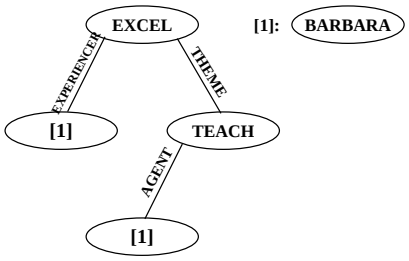


Figure 2: The semantic input underlying (1-3)

This simplistic view of sentence generation faces serious complications. A number of researchers have pointed out that the same semantic input may have a number of realizations that differ considerably in syntactic structure, both within a language and across languages ((Dorr, 1993); (Elhadad et al., 1997); (Stede, 1999); (Nicolov and Mellish, 2000); (Stone and Doran, 1997)). Suppose that the paraphrases in (1-3) come from the semantic input in Fig. 2.

(1) *Barbara excels at teaching.*
(2) *Barbara teaches well.*
(3) *Barbara is a good teacher.*

(1) is consistent with the simplistic view in that the top predicate EXCEL is realized by a verb (*excel*) which sets up an appropriate syntactic context into which the realizations of the other pieces can be fit. In (2-3), however, the predicate is realized by an adverb (*well*) and an

adjective (*good*). Since most generation systems need the main verb to set up the syntactic context of a clause, processing cannot start with an adverb or an adjective. Consequently, most systems that handle this case do so using exceptional processing (e.g. (Dorr, 1993)), special assumptions (e.g. (Elhadad et al., 1997)), or trying all possible predicates to start generation at (in (Nicolov and Mellish, 2000), such choice of a predicate imposes hierarchy on a non-hierarchical input), all to allow processing to start with the main verb.

We have developed a semantic head-driven generation methodology capable of handling such cases without special assumptions or processing. Processing is allowed to start with an adverb or an adjective which is able to set up an appropriate syntactic context into which other realizations can be fit (in the same way as into one set up by a main verb). We achieve this by a unique way of combining the realization of an argument with that of a predicate (regardless of its syntactic rank or category), as we shall see.

Our lexico-grammatical resources are designed around semantic units in order to have our generation methodology be driven by semantics. In order to keep the methodology simple, no independent reasoning about the syntax is performed. For this reason, a resource encapsulates the syntactic consequences of the lexico-grammatical unit.

This approach places significant demands on the grammatical formalism used to implement it. We find that one closely related to Tree-Adjoining Grammars (TAG, (Joshi, 1987)) and D-Tree Substitution Grammars (DSG, (Rambow et al., 2001)), which we call DSG/TAG, meets those demands well.

2 An overview of our architecture[2]

Like (Reiter and Dale, 2000), we view the generation process as consisting of 1) decomposing the semantic input into pieces, 2) finding lexico-grammatical resources to realize the pieces, and 3) putting the realizations of the pieces together in adherence to the rules of the language. We have designed our algorithm and lexico-grammatical resources in accordance with this view.

2.1 The algorithm

Our algorithm is a simple, recursive process:

1. given an unrealized input, find a lexico-grammatical resource that matches a piece containing the top predicate
2. recursively realize arguments and modifiers, as determined by the resource in step 1
3. combine the realizations in step 2 with the resource in step 1, as determined by it

2.2 Lexico-grammatical resources

The key to the simplicity of our algorithm lies in the lexico-grammatical resources which encapsulate information necessary to carry through generation. These consist of three parts:

- the semantic side: the portion of semantics realized by the resource (including the predicate and any arguments; this part is matched against the input semantics)
- the syntactic side: either word(s) in a syntactic configuration or a grammatical form without words, and syntactic consequences
- a mapping between semantic and syntactic constituents indicating which constituent on the semantic side is realized by which constituent on the syntactic side

A resource encapsulates all syntactic consequences of the lexico-grammatical choice, e.g. the resource for the verb *rain* includes the semantically-vacuous subject *it* and the resource for *excel at* includes a *PRO* in the subject position of the complement, as in (1).

Resources should be complete but elementary rather than composite, which allows the maximal use of compositionality in generation. This leads to the imperative and wh-question forms, which realize separate semantic or pragmatic content, constituting separate resources in our architecture. Note that a resource might not contain any words (e.g. the imperative).

[2]Details of our architecture are given in (Kozlowski, 2001) and (Kozlowski, 2002).

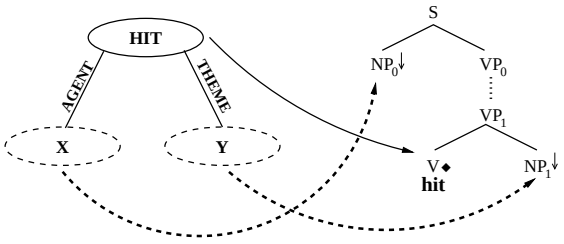


Figure 3: A resource for HIT realized by *hit*

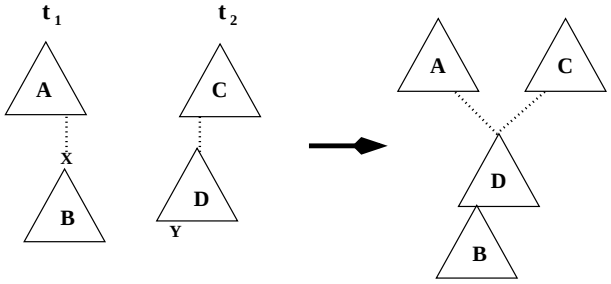


Figure 4: Substitution in DSG

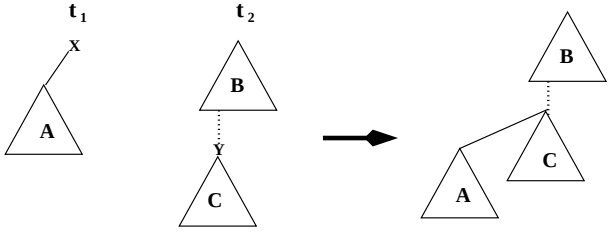


Figure 5: Sister-adjunction in DSG

3 The architecture using DSG/TAG

Now that we have briefly described our generation architecture, we are ready to discuss the implementation, including the selection of an appropriate grammatical formalism and examples of lexico-grammatical resources and the generation process.

3.1 The DSG/TAG formalism

Both the algorithm and the constraints on lexico-grammatical resources place significant demands on the grammatical formalism to be used in the implementation of the architecture. The formalism we use, which we call DSG/TAG, is similar to DSG but there are slight differences in the formal details[3]. Consider the elementary structure on the right-hand side of Fig. 3. This structure is *anchored* by the verb *hit*. It has two substitution nodes, $\mathtt{NP}_0$ and $\mathtt{NP}_1$ to which the realizations of arguments are to be substituted. Immediate domination is represented by solid lines. Dotted lines represent domination of length zero or more, where other syntactic material (e.g. modifiers) may end up. In regular TAG, such links must join nodes of the same category. In DSG, this requirement is dropped, a fact that is useful for us.

The DSG operation of *substitution* is designed for adding the realization of an argument to that of a predicate. It is more complicated than TAG substitution which involves the unification of the root of the former with a substitution node of the latter. Fig. 4 shows the general form of DSG substitution of structure $\mathtt{t}_1$ into structure $\mathtt{t}_2$. Node X is unified with node Y[4]. Notice that there is a non-determinism in the position of component A vis-à-vis component C. For our purposes, we make an additional restriction that component A ends up above component C[5]. Notice that since $\mathtt{t}_2$ ends up between components A and B, this entire operation is technically similar to the TAG operation of adjunction of $\mathtt{t}_2$ to $\mathtt{t}_1$.

The DSG operation of *sister-adjunction* is designed for adding the realization of a modifier to that of a predicate. Fig. 5 shows the general form of sister-adjoining structure $\mathtt{t}_1$ into structure $\mathtt{t}_2$. Node X is unified with node Y.

3.2 Examples of resources

Fig. 3 shows an example of a simple resource, one for the predicate HIT realized by the verb *hit* in the active voice configuration. The semantic side on the left-hand side indicates that the predicate and the AGENT and THEME roles are realized by this resource. The agent and

[3]One difference stems from our desire, like (Candito and Kahane, 1998), to capture appropriate semantic dependencies in elementary structures.

[4]When component A is empty, the operation is the same as TAG substitution.

[5]We have not found instances where this constraint causes difficulties for our architecture. For instance, in the Kashmiri example in (Rambow et al., 1995), used to motivate the use of DSG, our architecture calls for a wh-questioning resource separate from one headed by the verb of the questioned clause.

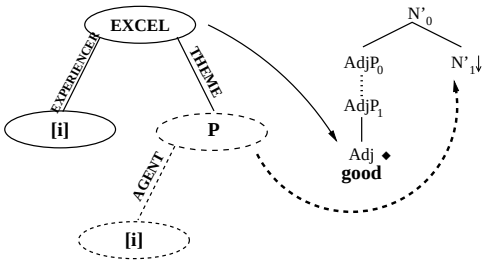


Figure 6: A resource for EXCEL realized by *good*

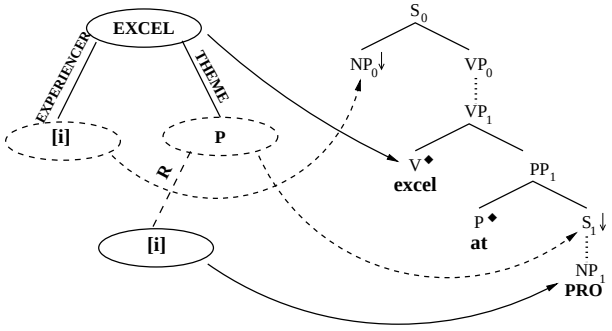


Figure 7: A resource for *excel at* realizing EXCEL

theme *arguments* are included as unrealized (indicated by the dashed ovals)[6]. These arguments are linked to the positions for their realizations on the syntactic side, the subject and complement ($\texttt{NP}_0$ and $\texttt{NP}_1$) of the clause anchored by *hit*. The resource also includes a link between HIT and its realization, the verb *hit*.

No assumptions are made about the syntactic rank or category of a predicate realization. Consider the resource for EXCEL realized by the adjective *good* in Fig. 6. Note the link between the uninstantiated theme on the semantic side and the position for its corresponding syntactic realization, the substitution node $\texttt{N}_1'$[7].

DSG/TAG allows the encapsulation of more syntactic consequences than regular TAG. Consider the resource for the predicate EXCEL realized by *excel at* in Fig. 7. One syntactic consequence is a *PRO* in the subject position of the complement of *excel at*. The flexibility of the DSG/TAG formalism allows the inclusion of the

[6]variables (X, Y, P, R) match anything.

[7]Also notice that the experiencer of EXCEL is considered realized by the *good* resource and coreferenced with an argument of the theme of EXCEL, which must be realized by a separate resource.

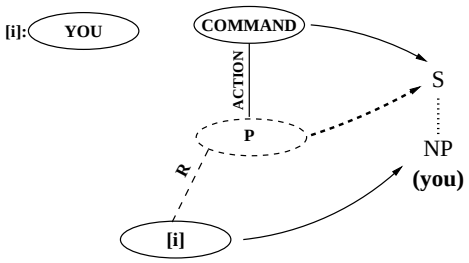


Figure 8: A resource for COMMAND

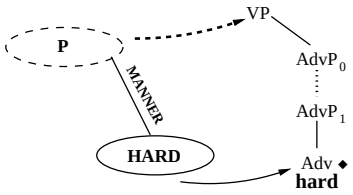


Figure 9: A modifier resource for MANNER HARD

PRO within the elementary structure for *excel at*. Note that the $\texttt{S}_1$ node (the position for the realization of the theme) dominates a node of different category, the NP *PRO*.

DSG/TAG allows a resource to contain more than one word, e.g. the idiom *kick the bucket* or the verb *rain* with the semantically-vacuous subject *it* as a syntactic consequence, or no words, e.g. the imperative (Fig. 8).

Fig. 9 shows a modifier resource. The link between the uninstantiated predicate being modified and the position for its syntactic realization (the VP node) specifies how the modifier is to be combined with the predicate realization.

3.3 Generation examples

In discussing generation examples, we refer to a single instantiation of steps 1-3 of the algorithm as a *region*, the unwinding of the recursion as the *descent* process, and its closing off as the *ascent* process, after (McCoy et al., 1992).

3.3.1 A standard generation example

Consider the semantic input in Fig. 10 with (4) as one of its realizations.

(4) Mary hit John hard.

The top $\Gamma_{\texttt{HIT}}$ region realizes the top of the input. One resource that matches a piece of the input including the top predicate is *hit* in Fig. 3. The matching determines what piece of

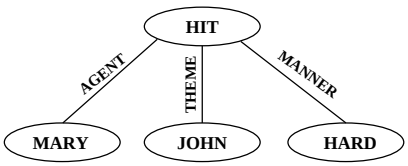


Figure 10: The semantic input underlying (4)

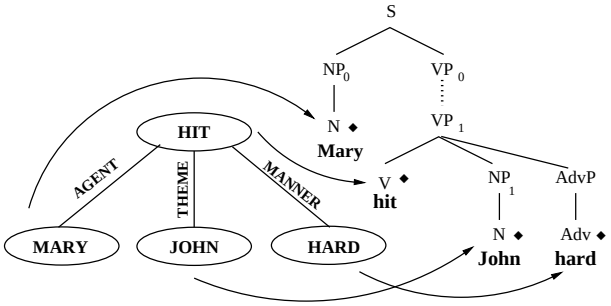


Figure 11: The result in region $\Gamma_{\texttt{HIT}}$

the input it realizes and the remaining pieces to be realized independently. The $\Gamma_{\texttt{HIT}}$ region spawns the $\Gamma_{\texttt{MARY}}$ and $\Gamma_{\texttt{JOHN}}$ argument subregions for the realization of the subtrees of the input rooted at MARY and JOHN, which produce the noun phrases *Mary* and *John* as the realizations. The remaining thematic role MANNER with the argument HARD is realized in a modifier region, which produces the modifier realization *hard*.

We turn now to the ascent process. In the $\Gamma_{\texttt{HIT}}$ region, step 3 of the algorithm involves the substitution of the noun phrases *Mary* and *John* to the subject and complement positions of the clause headed by *hit*. This comes from the mapping in the *hit* resource between the agent and the theme of HIT and the subject and complement of *hit*. The realization of the remaining thematic role MANNER with the argument HARD is sister-adjoined to the *hit* structure. The VP node of *hard* is unified with a node that realizes the predicate HIT. The mapping in the resource includes a link between HIT and the verb *hit*. Any node from this node up to the root of the realization (S) where *hard* can be sister-adjoined, may be used. There is only one such node here, $\texttt{VP}_1$. The result is shown in Fig. 11[8].

[8]We do not discuss further details of the process such as collapsing the domination links and inflection, achieved using syntactic features.

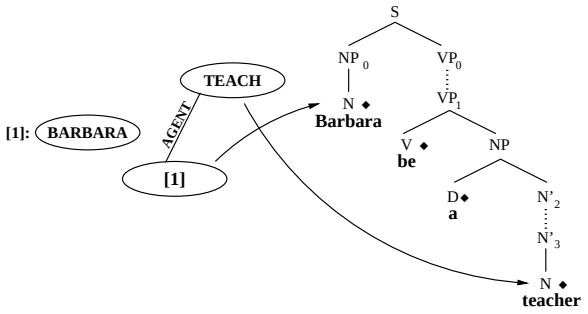


Figure 12: The result in region $\Gamma_{\texttt{TEACH}}$

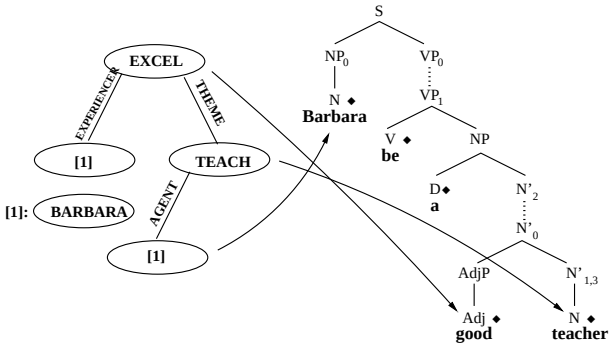


Figure 13: The result in region $\Gamma_{\texttt{EXCEL}}$

3.3.2 Non-verb predicate realization

We make no assumptions about the rank or category of the realization of a predicate. An adverb or an adjective can set up a syntactic context in exactly the same way as a main verb. Suppose the semantic input underlying (1-3) is as given in Fig. 2. The top $\Gamma_{\texttt{EXCEL}}$ region selects the *good* resource in Fig. 6 which matches a portion of the input including the top predicate. The $\Gamma_{\texttt{EXCEL}}$ region spawns the $\Gamma_{\texttt{TEACH}}$ subregion for the realization of the subtree of the input rooted at TEACH. This subtree is realized in the standard way, yielding the result in Fig. 12.

Because of the link between the theme of EXCEL and the $\texttt{N}_1'$ node of *good*, the realization of the subtree rooted at TEACH, *Barbara be a teacher* in Fig. 12, is to be substituted to the $\texttt{N}_1'$ node of *good*. For the substitution, we must determine a node in the realization in Fig. 12 to be unified with $\texttt{N}_1$. Note that the predicate TEACH is linked to its realization, the noun *teacher*. Any node from that node up to the root of the realization (the S node), substitutable to $\texttt{N}_1'$, may be

used. There is only one such node here, $\mathtt{N}'_3$. The result is shown in Fig. 13. Notice that *Barbara be a* ends up above the *good* resource.

3.3.3 An already-realized subtree

Some resources realize arguments of predicates not realized by the resources. Consider the resource in Fig. 7, which matches the top of the input in Fig. 2. Its semantic side contains an argument of the unrealized theme predicate of `EXCEL`, realized by the *PRO*. The matching process determines that this argument is the agent of `TEACH`. In the descent process, a region will eventually be spawned for its realization. That region will notice that the agent has already been realized (by the *PRO*) and will simply use the realization in subsequent processing.

4 Conclusions

We have discussed in this paper the appropriateness of the DSG/TAG formalism for a uniform and flexible sentence generation architecture. An example of the flexibility of our architecture is the ability to generate (5-6) from the same input in a uniform manner, by properly specifying the resources for the wh-question and the imperative forms, which DSG/TAG allows.

(5) *Who invented calculus?*

(6) *Identify the inventor of calculus!*

Some examples presented here illustrate that a grammar for generation is likely to be different from one appropriate for parsing. For example, since generation is driven by semantics, it is irrelevant whether a resource contains any words.

We have developed a fully-operational prototype of our generation architecture, capable of generating, among others, the examples presented here. Our future work includes the incorporation of collocations, syntactic constraints imposed by the realization of a predicate on the realization of its arguments (e.g. sentential complement of *be glad* vs. nominal complement of *welcome*), and a multiple indexation scheme for an efficient search for resources matching a portion of input including the top predicate.

References

Marie-Helene Candito and Sylvain Kahane. 1998. Defining DTG Derivations to Get Semantic Graphs. In *4th Int. Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*, Philadelphia.

Bonnie J. Dorr. 1993. Interlingual machine translation: a parametrized approach. *Artificial Intelligence*, 63.

Michael Elhadad, Kathleen McKeown, and Jacques Robin. 1997. Floating constraints in lexical choice. *Computational Intelligence.*

Aravind K. Joshi, 1987. *The Relevance of Tree Adjoining Grammar to Generation*, pages 233–252. Kluwer Academic, Dordrecht, The Netherlands.

Raymond Kozlowski. 2001. Utilizing the variety of lexico-grammatical resources in uni- and multilingual sentence generation. Ph.D. dissertation proposal. Department of Computer and Information Sciences. University of Delaware.

Raymond Kozlowski. 2002. Driving multilingual sentence generation with lexico-grammatical resources. In *2nd International Natural Language Generation Conference (INLG'02).* To appear.

Kathleen F. McCoy, K. Vijay-Shanker, and Gijoo Yang. 1992. A Functional Approach to Generation with TAG. In *30th Annual Meeting of the Association for Computational Linguistics.*

Nicolas Nicolov and Chris Mellish. 2000. PROTECTOR: Efficient Generation with Lexicalized Grammars. In *Recent Advances in Natural Language Processing.*

Owen Rambow, K. Vijay-Shanker, and David Weir. 1995. D-Tree Grammars. In *Association for Computational Linguistics*, volume 33.

Owen Rambow, K. Vijay-Shanker, and David Weir. 2001. D-Tree Substitution Grammars. *Computational Linguistics*, 27(1):87–122.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems.* Cambridge University Press.

Manfred Stede. 1999. *Lexical semantics and knowledge representation in multilingual text generation.* Kluwer Academic Publishers, Boston.

Matthew Stone and Christine Doran. 1997. Sentence Planning as Description Using Tree Adjoining Grammar. In *Association for Computational Linguistics.*