‘If you’ve heard it, you can say it’ - Towards an Account of Expressibility

David D. McDonald
Raytheon BBN Technologies
Cambridge, MA USA
dmcdonald@bbn.com

Charles F. Greenbacker
University of Delaware
Newark, DE, USA
charlieg@cis.udel.edu

Abstract

We have begun a project to automatically create the lexico-syntactic resources for a microplanner as a side-effect of running a domain-specific language understanding system. The resources are parameterized synchronous TAG Derivation Trees. Since the KB is assembled from the information in the texts that these resources are abstracted from, it will decompose along those same lines when used for generation. As all possible ways of expressing each concept are pre-organized into general patterns known to be linguistically-valid (they were observed in natural text), we obtain an architectural account for expressibility.

1. Expressibility

People speak grammatically. They may stutter, restart, or make the occasional speech error, but all in all they are faithful to the grammar of the language dialects they use. One of the ways that a language generation system can account for this is through the use of grammar that defines all of the possible lexico-syntactic elements from which a text can be constructed and defines all their rules of composition, such as lexicalized Tree Adjoining Grammar (TAG). Without the ability to even formulate an ungrammatical text, such a generator provides an account for human grammaticality based on its architecture rather than its programmer.

We propose a similar kind of accounting for the problem of expressibility: one based on architecture rather than accident. *Expressibility*, as defined by Meteer (1992), is an issue for microplanners as they decide on which lexical and syntactic resources to employ. Not all of the options they might want to use are available in the language – they are not expressible. Consider the examples in Figure 1, adapted from Meteer 1992 pg. 50.

Expression	Construction (‘decide’)
“*quick decision*”	<result> + <quick>
“*decide quickly*”	<action> + <quick>
“*important decision*”	<result> + <important>
* “*decide importantly*”	<action> + <important>

Figure 1: Constraints on expressibility: To say that there was a decision and it was important, you are forced to use the noun form because there is no adverbial form for *important* as there is for *quick*

In this short paper, we discuss our approach to expressibility. We describe in detail our novel method centered on how to use parser observations to guide generator decisions, and we provide a snapshot of the current status of our system implementation.

2. Related Work

Natural language generation (NLG) systems must have some way of making sure that the messages they build are actually expressible. Template-based generators avoid problems with expressibility largely by anticipating all of the wording that will be needed and packaging it in chunks that are guaranteed to compose correctly. Becker (2006), for example, does this via fully lexicalized TAG trees.

Among more general-purpose generators, one approach to expressibility is to look ahead into the lexicon, avoiding constructions that are lexically incompatible. Look-ahead is expensive, however, and is only practical at small abstraction distances such as Shaw’s re-writing sentence planner (1998).

Meteer’s own approach to expressibility started by interposing another level of representation between the microplanner and the surface realizer, an ‘abstract syntactic representation’ in the sense of RAGS (Cahill et al. 1999), that employed functional relationships (head, argument, matrix, adjunct) over semantically typed, lexicalized constituents. This blocks **decide importantly* because 'important' only has a realization as a property and her composition rules prohibit using a property to modify an action ('decide'). Shifting the perspective from the action to its result allows the composition to go through.

We are in sympathy with this approach – a microplanner needs its own representational level to serve as a scratch pad (if using a revision-based approach) or just as a scaffold to hold intermediate results. However, Meteer's semantic and lexical constraints do require operating with fine-grain details. We believe that we can work with larger chunks that have already been vetted for expressibility because we've observed someone use them, either in writing or speech.

3. Method

Our approach is similar to that of Zhong & Stent (2005) in that we use the analysis of a corpus as the basis for creating the resources for the realization component. Several differences stand out. For one, we are working in specific domains rather than generic corpora like the WSJ. This enables the biggest difference: our analysis is performed by a completely accurate,[1] domain-specific NLU system ('parser')[2] based on a semantic grammar (McDonald 1993). It is reading for the benefit of a knowledge base, adding specific facts within instances of a highly structured, predefined prototypes. Such instances are used as the starting point for the generation process.

On the KB side, our present focus happens to be on hurricanes and the process they go through as they evolve. We have developed a semantic grammar for this domain, and it lets us analyze texts like these:[3]

(1) "*Later that day it made landfall near the Haitian town of Jacmel.*"

(2) "*... and remained at that intensity until landfall on the morning of September 1 near Cocodrie, Louisiana.*"

(3) "*By landfall on Monday morning …*"

Such texts tell us how people talk about hurricanes, specifically here about landfall events. They tell us what combinations of entities are reasonable to include within single clauses (intensity, time, location), and they tell us which particular realizations of the landfall concept have been used in which larger linguistic contexts. They also indicate what information can be left out under the discourse conditions defined by the larger texts they appear in.[4]

As different texts are read, we accumulate different realization forms for the same content. In example #1, landfall is expressed via the idiom *make landfall*, the time is given in an initial adverbial, and the location as a trailing adjunct. In #2, the landfall stands by itself as the head of a time-adverbial and the time and location are adjuncts off of it. This set of alternative phrasings provides the raw material for the microplanner to work with – a natural set of paraphrases.

3.1 Derivation Trees as templates

As shown in Figure 3, to create resources for the microplanner, we start with the semantic analysis that the parser anchors to its referent when it instantiates the appropriate event type within the prototypical model of what hurricanes do, here a 'landfall event', noting the specific time and location. Following Bateman (e.g. 2007) and Meteer (1992), we work with typed, structured objects organized under a foundational ontology.[5] Figure 2 shows the current definition of the landfall class in a local notation for OWL Full.

```
(Class HurricaneLandfall
  (restrict hurricane - Hurricane)
  (restrict intensity - Saffir-Simpson)
  (restrict location - PhysEndurant)
  (restrict time - Date&Time))
```

Figure 2. The Landfall class

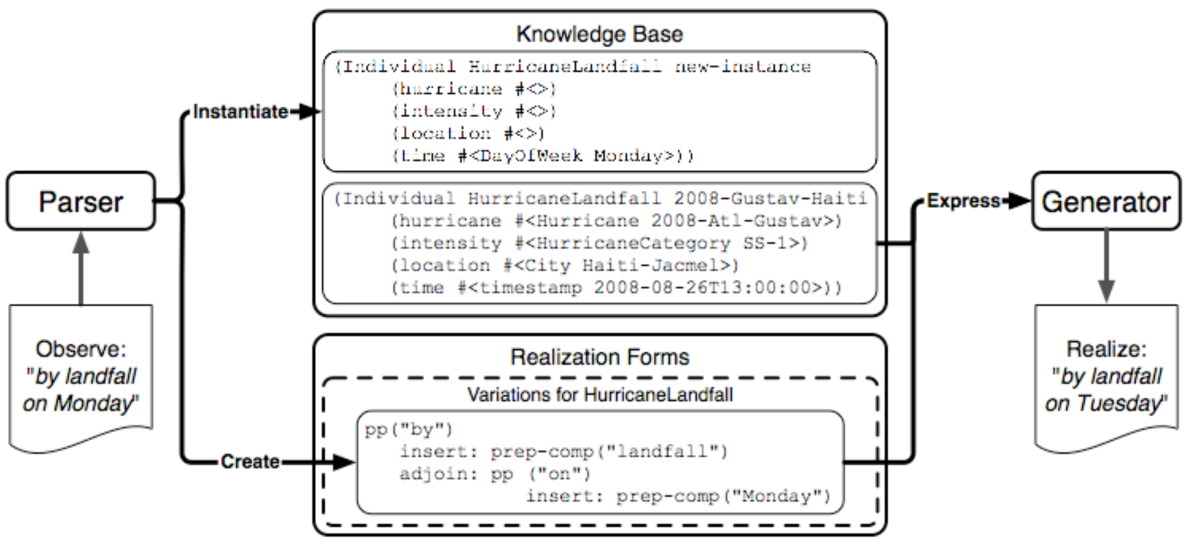
[1] Parse accuracy and correct word sense interpretation is only possible if the semantic domain under analysis is restricted by topic and sublanguage.

[2] Most systems referred to as "parsers" stop at a structural description. Ours stops at the level of a disambiguated conceptual model and is more integrated than most.

[3] #1 and 2 are from the Wikipedia article on Hurricane Gustav. #3 is from a New York Times article.

[4] For example, in #1 and #3 the precise date had been given already in earlier sentences.

[5] An extension of Dolce (Gangemi et al. 2002).


Knowledge Base
Instantiate
Parser
Observe: "by landfall on Monday"
Express
Generator
Realize: "by landfall on Tuesday"
Realization Forms
Variations for HurricaneLandfall
Create

```
(Individual HurricaneLandfall new-instance
     (hurricane #<>)
     (intensity #<>)
     (location #<>)
     (time #<DayOfWeek Monday>))
```

```
(Individual HurricaneLandfall 2008-Gustav-Haiti
     (hurricane #<Hurricane 2008-Atl-Gustav>)
     (intensity #<HurricaneCategory SS-1>)
     (location #<City Haiti-Jacmel>)
     (time #<timestamp 2008-08-26T13:00:00>))
```

```
pp("by")
   insert: prep-comp("landfall")
   adjoin: pp ("on")
             insert: prep-comp("Monday")
```

Figure 3. Overview

The semantic analysis recursively maps constituents' referents to properties of a class instance. Accompanying it is a syntactic analysis in the form of a TAG Derivation Tree[6] (DT) where each of its nodes (initial trees, insertions or adjunctions) points both to its lexical anchor and its specific correspondence in the domain model.

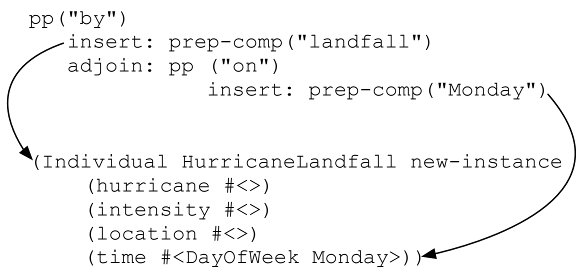
To create a reusable resource, we abstract away from the lexicalization in these DT/model-anchored pairs, and replace it with the corresponding model classes as determined by the restrictions on the properties. For example, the day of the week in #3, lexically given as *Monday morning* and then dereferenced to an object with the meaning '9/1/2008 before noon' is replaced in the resource with that object's type.

The result is a set of templates associated with the combination of types that corresponds to the participants in its source text – the more composed the type, the more insertions / adjunctions in the template derivation tree.

3.2 Synchronous TAGS

This combination of derived trees and model-levels classes and properties where the nodes of the two structures are linked is a *synchronous TAG* (ST). As observed by Shieber and Schabes (1991) who introduced this notion, "[STs] make the fine-grained correspondences between expressions of natural language and their meanings explicit by … node linking".

[6] The primary analysis is phrase structure in a chart, but since every rule in the grammar corresponds to either a lexicalized insertion or adjunction, the pattern of rule application is read out as a TAG derivation tree.



```
pp("by")
   insert: prep-comp("landfall")
   adjoin: pp ("on")
             insert: prep-comp("Monday")

(Individual HurricaneLandfall new-instance
     (hurricane #<>)
     (intensity #<>)
     (location #<>)
     (time #<DayOfWeek Monday>))
```

Figure 4. Synchronous TAG

In particular, they observe that STs solve an otherwise arbitrary problem of 'where does one start' when faced with a bag of content to be realized as a text. Our STs identify natural 'slices' of the content – those parts that have already been observed to have been realized together in a naturally occurring text.

Because we have the luxury to be creating the knowledge base of our hurricane model by the accretion of relationships among individually small chunks of information (a triple store), we can take synchronous TAGS a step further and allow them to dictate the permitted ways that information can be delimited within the KB for purposes of generation following the ideas in (Stone 2002).

If we can surmount the issues described below, this stricture – that one can only select for generation units of content of the types that have been observed to be used together (the model side of the STs) – is a clean architectural explanation of how it is that the generator's messages are always expressible.

4. State of Development

We are at an early stage in our work. Everything we have described is implemented, but only on a 'thin slice' to establish that our ideas were credible. There are many issues to work out as we 'bulk up' the system and begin to actually integrate it in a in 'tactical' microplanner and begin to actually do the style of macro-planning (determining the relevant portions of the domain model to use as content given the intent and affect) that our use of synchronous TAGS should allow. The most salient issues are how broadly we should generalize when we substitute domain types for lexicalizations in the templates, and what contextual information must be kept with the templates.

The type generalizations need to be broad enough to encompass as many substitutions as possible, while being strict enough to ensure that when the template is applied to those objects the realizations available to them permit them to be expressed in that linguistic context.[7]

The examples all have specific contexts in the sentences and recent discourse. Two of them (#2, #3) are using the landfall event as a time phrase. Can we move them and still retain the naturalness of the original (e.g. from sentence initial to sentence final), or does this sort of information need to be encoded?

Another issue is how to evaluate a system like this. Given the accuracy of the analysis, recreating the source text is trivial, so comparison to the source of the resources as a gold standard is meaningless. Some alternative must be found.

While we work out these issues, we are extending the NLU domain model and grammar to cover more cases and thence create more synchronized TAG templates. We then manually identify alternative domain content to app hly to them to in order to explore the space of realizations and identify unforeseen interactions.

Our short-term goals are to vastly increase the grammar coverage for our motivating examples and to hand over all microplanning decisions to the system itself. Long-term goals include broadening the coverage further still, to as open a domain as is feasible, as well as testing different macroplanners and applications with which to drive the entire process. Among several possibilities are automatic merged-and-modified summarization and a query-based discourse system.

[7] In our example, substituting different days and times is obvious (*by landfall on the afternoon of August 22*), but as we move away from that precise set of types (general-time-of-day + date) we see that what had been lexically fixed in the derivation tree (*by landfall* ***on***) has to shift: ... ***at*** *2:00 on August 22.*

5. Discussion

Because the phrasal patterns observed in the corpus act as templates guiding the generation process, and as the underlying NLU system and generator (McDonald 1993, Meteer et al. 1987) are mature and grounded in linguistic principles, our system combines template-based and theory-based approaches.

Van Deemter et al. (2005) outlined three criteria for judging template-driven applications against "standard" (non-template) NLG systems. (1) *Maintainability* is addressed by the fact that our templates aren't hand-made. To extend the set of available realization forms we expose the NLU system to more text. The subject domain has to be one that has already been modeled, but we are operating from the premise that a NLG component would only bother to speak about things that the system as a whole understands. (2) *Output quality and variabilit*y are determined by the corpus; using corpora containing high quality and varied constructions will enable similar output from the generator. (3) Most crucially, our parser and generator components are linguistically *well-founded*. Composition into our 'templates' is smoothly accommodated (extra modifiers, shifts in tense or aspect, application of transformations over the DT to form questions, relative clauses, dropped constituents under conjunction). The fully-articulated syntactic structure can be automatically annotated to facilitate prosody or to take information structure markup on the DT.

The closest system to ours may be Marciniak & Strube (2005) who also use an annotated corpus as a knowledge source for generation, getting their annotations via "a simple rule-based system tuned to the given types of text". As far as we can tell, they are more concerned with discourse while we focus on the integration with the underlying knowledge base and how that KB is extended over time.

Like them, we believe that one of the most promising aspects of this work going forward is that the use of a parser provides us with "self-labeling data" to draw on for statistical analysis. Such training material would reduce the effort required to adapt a generator to a new domain, while simultaneously improving its output.

Acknowledgments
This work was supported in part by the BBN POIROT project: DARPA IPTO contract FA865 0-06-C-7606.

References

John Bateman, Thora Tenbrink, and Scott Farrar. 2007. The Role of Conceptual and Linguistic Ontologies in Interpreting Spatial Discourse. *Discourse Processes*, 44(3):175–213.

Tilman Becker. 2006. Natural Language Generation with Fully Specified Templates. In W. Wahlster (Ed.), *SmartKom: Foundations of Multimodal Dialog Systems*, 401–410. Springer, Berlin Heidelberg.

Lynne Cahill, Christy Doran, Roger Evans, Chris Mellish, Daniel Paiva, Mike Reape, Donia Scott, & Neil Tipper. 1999. *Towards a Reference Architecture for Natural Language Generation Systems, The RAGS project*. ITRI technical report number ITRI-99-14, University of Brighton, March.

Aldo Gangemi, Nicola Guarino, Claudio Masolo, Alessandro Oltramari, & Luc Schneider. 2002. Sweetening Ontologies with DOLCE. In *Proceedings of the 13th International Conference on Knowledge Acquisition, Modeling and Management (EKAW)*, pages 166–181, Sigüenza, Spain, October 1–4.

Tomasz Marciniak & Michael Strube. 2005. Using an Annotated Corpus As a Knowledge Source For Language Generation. In *Proceedings of the Corpus Linguistics 2005 Workshop on Using Corpora for Natural Language Generation (UCNLG)*, pages 19–24, Birmingham, UK, July 14.

David McDonald. 2003. The Interplay of Syntactic and Semantic Node Labels in Partial Parsing, in the proceedings of the Third International Workshop on Parsing Technologies, August 10-13, 1993 Tilburg, The Netherlands, pp. 171-186; revised version in Bunt and Tomita (eds), Recent Advances in Parsing Technology, Kluwer Academic Publishers, pgs. 295-323.

Marie W. Meteer. 1992. *Expressibility and the Problem of Efficient Text Planning*. Pinter, London.

Marie Meteer, David McDonald, Scott Anderson, David Forster, Linda Gay, Alison Huettner & Penelope Sibun. 1987. Mumble-86: Design and Implementation, TR #87-87 Dept. Computer & Information Science, UMass., September 1987, 174 pgs.

James Shaw. 1998. Clause Aggregation Using Linguistic Knowledge. In *Proceedings of the 9th International Workshop on Natural Language Generation*, pages 138–147, Niagara-on-the-Lake, Ontario, August 5–7.

Stuart Shieber & Yves Schabes. 1991. Generation and synchronous tree-adjoining grammar. *Computational Intelligence*, 7(4):220–228.

Matthew Stone. 2003. Specifying Generation of Referring Expressions by Example. In *Proceedings of the AAAI Spring Symposium on Natural Language Generation in Spoken and Written Dialogue*, pages 133–140, Stanford, March.

Kees van Deemter, Emiel Krahmer, & Mariët Theune. 2005. Real versus Template-Based Natural Language Generation: A False Opposition? *Computational Linguistics*, 31(1):15–24.

Huvava Zhong & Amada Stent. 2005. Building Surface Realizers Automatically From Corpora. In *Proceedings of the Corpus Linguistics 2005 Workshop on Using Corpora for Natural Language Generation (UCNLG)*, pages 49–54, Birmingham, UK, July 14.