Extending a Surface Realizer to Generate Coherent Discourse

Eva Banik
The Open University
Milton Keynes, UK
e.banik@open.ac.uk

Abstract

We present a discourse-level Tree Adjoining Grammar which tightly integrates syntax and discourse levels, including a representation for discourse entities. We show that this technique makes it possible to extend an optimisation algorithm used in natural language generation (polarity filtering) to the discourse level. We implemented the grammar in a surface realizer and show that this technique can be used to reduce the search space by filtering out referentially incoherent solutions.

1 Introduction

A fundamental problem that microplanners and surface realizers face in natural language generation is how to restrict the search space of possible solutions. A traditional solution to this computational complexity problem is to divide the generation process into tractable sub-problems, each represented as a module in a pipeline, where every decision made by a module restricts the number of options available to others further down the line. Though such pipeline architectures are computationally efficient, they severely restrict the flexibility of the system and the quality of the generated output. Most systems with pipeline architectures generate relatively simple, domain-specific output. Systems that produce more complex linguistic constructions typically achieve this by adding more modules to the pipeline (e.g. a revision module (Robin, 1994) or aggregation (Shaw, 2002)). Since complex linguistic constructions often require interaction between modules, adding them to the repertoire of pipelined NLG systems becomes an engineering and programming task.

Integrated NLG systems have a simpler architecture because they do not need to model interactions between modules. However, they still face the problem of computational complexity that was originally solved by the pipeline model. Strategies that have been introduced to reduce the search space in integrated systems include greedy/incremental search algorithms (Stone et al., 2003), constructing a dependency graph for a flat semantic input and converting it into a derivation tree (Koller and Striegnitz, 2002), using planning algorithms (Appelt, 1985; Koller and Stone, 2007), polarity filtering (Kow, 2007) and using underspecified g-derivation trees (G-TAG, Danlos (2000)). Despite all these efforts, most systems still don't attempt to go above the sentence level or generate very complex sentences. In this paper we present a new technique for designing an integrated grammar for natural language generation. Using this technique it is possible to use linguistic constraints on referential coherence to automatically reduce the search space — which in turn makes it possible to generate longer and more coherent texts.

First we extend the grammar of a surface realizer to produce complex, multi-sentential output. Then we add a representation for discourse referents to the grammar, inspired by Centering Theory's notion of a backward looking center and preferred center. Having done this, we show that by integrating discourse-level representations into a syntactic grammar we can extend an optimization technique — polarity filtering (Kow, 2007; Gardent and Kow, 2006) — from syntactic realization to the discourse level.

2 The Problem of Referential Coherence

Referential coherence is the phenomenon which is responsible for the contrast in (1), in the sense that the example in (1b) is perceived to be more coherent than (1a).

(1) a Elixir is approved by the FDA. Viral skin disorders are relieved by

Aliprosan. Elixir is a white cream. Aliprosan is an ingredient of Elixir.

b Elixir is a white cream. Elixir is approved by the FDA. Elixir contains Aliprosan. Aliprosan relieves viral skin disorders.

Centering Theory (Grosz et al., 1995) is a frequently used framework for modeling referential coherence in discourse. It is based on the notion that for each utterance in a discourse there is a set of entities which are the *centers* of attention and which serve to link that utterance to other utterances in the same discourse segment. Entities mentioned by an utterance (the set of forward looking centers) form a partially ordered list called the Cf list where roughly, subjects are ranked highest, followed by objects, indirect objects and other arguments or adjuncts. The backward looking center of Un is said to be the most highly ranked element on the Cf list of Un-1 mentioned in the previous utterance.

Centering Theory has been adapted to NLG by Kibble (1999; 2001), and implemented in Kibble and Power (2004). Rather than using the notion of centering transitions as defined by Grosz et al. (1995), in these papers centering theory is redefined as constraints on **salience** and **cohesion**. These constraints state that there is a preference for consecutive utterances to keep the same center and that there is a preference for the center of Un to be realized as the highest ranked entity on the Cf list of Un. Kibble and Power (2004) show how these constraints can be used to drive text planning, sentence planning and pronominalization in an integrated fashion. Our approach is similar to Kibble and Power (2004) in that we don't use the concept of centering transitions. However, our method is more efficient in that Kibble and Power (2004) use centering transitions to rank the set of generated solutions (some of which are incoherent), whereas we encode centering constraints in elementary trees to reduce the search space of possible solutions *before* we start computing them.

3 GenI and Polarity Filtering

The grammar described in the next section was implemented in the GenI surface realizer (Kow, 2007), which uses a lexicalized feature-based Tree Adjoining Grammar to generate all possible paraphrases for a given flat semantic input. GenI implements an optimization technique called *polarity filtering* to constrain the effects of lexical ambiguity. The basic idea of polarity filtering is to associate elementary trees with a set of polarities. When these polarities don't 'cancel each other out', it means that it is not possible to combine the set of trees selected for a given input. This is a quick way to check whether the number of argument slots is the same as the number of potential arguments. For example, if the lexical selection consists of two trees for a given input, one of which provides an NP (`-NP`) and one of which expects two NPs (`-2NP`) then the sum of polarities will be `-NP` and therefore the generator will not attempt to combine the trees.

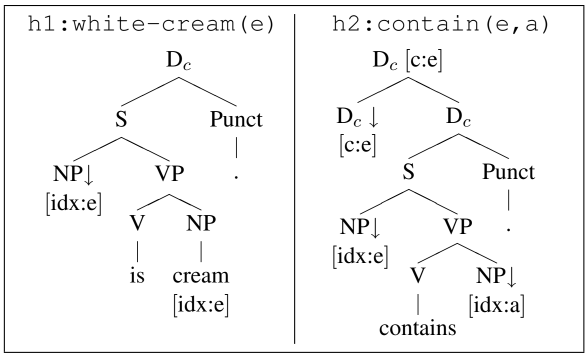


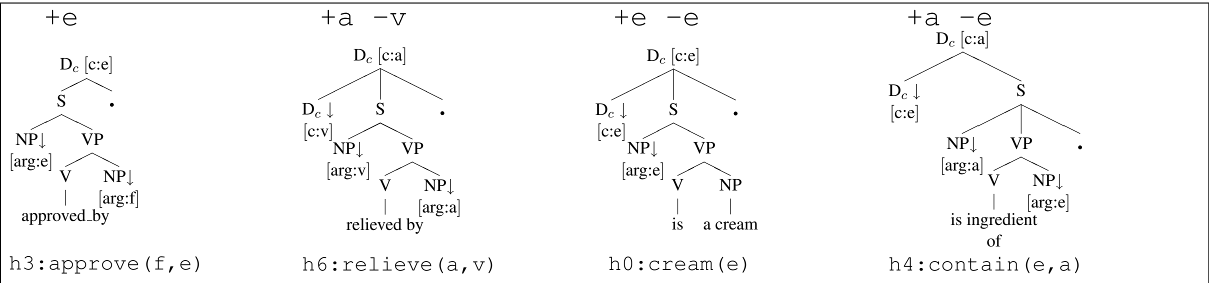
Figure 1: Elementary syntax/discourse trees

Values for polarities are defined as follows: every initial tree is assigned a `-cat` polarity for each substitution node of category *cat* and a `+cat` polarity if its root node is of category *cat*. Auxiliary trees are assigned a `-cat` polarity for each substitution node only.

Polarity filtering is a very powerful optimization technique, because it allows the generator to reduce the search space early on in the process, before it attempts to combine any trees.

4 An Integrated Syntax-Discourse Grammar

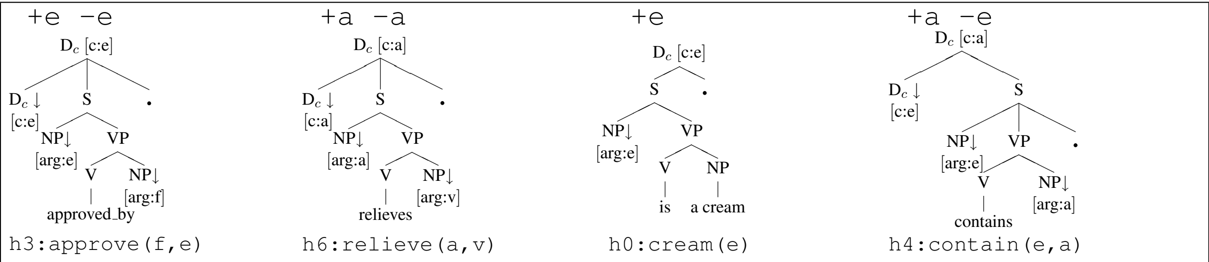
In order to generate mutisentential text, we first define a discourse-level Tree Adjoining Grammar. The trees in the grammar tightly integrate syntax and discourse representations in the sense that sentence-level elementary trees include one or more discourse-level nodes. The elementary trees in Fig. 1 illustrate what we mean by this: every lexical item that would normally project a sentence in a syntactic grammar (i.e., an S-rooted



+2a -v

Elixir is approved by the FDA. Viral skin disorders are relieved by Aliprosan. Elixir is a white cream. Aliprosan is an ingredient of Elixir.

Figure 2: Discourse-level polarities for (1a) sum up to `+2a -v`



+a

Elixir is a white cream. Elixir is approved by the FDA. Elixir contains Aliprosan. Aliprosan relieves viral skin disorders.

Figure 3: Discourse-level polarities for (1b) sum up to `+a`

tree) here projects a discourse clause (i.e., a D*c* rooted tree). Every predicate that projects a discourse clause is assigned two kinds of elementary trees: a discourse initial tree (Fig. 1a) and a discourse continuing tree (Fig. 1b), which takes the preceding discourse clause as an argument.

We model referential coherence by associating a discourse entity with every root- and substitution node of category D_c. A discourse entity on a root node is "exported" by the elementary tree to be the center of attention in the next sentence. This roughly corresponds to Centering Theory's notion of a forward looking center. A discourse entity on a substitution node is the entity expected by the sentence to have been the center of attention in the previous utterance, roughly corresponding to the notion of backward looking center in Centering Theory.

For example, the tree on the left in Fig. 1. exports the discourse entity representing its subject ('e') as its "forward looking center". The tree on the right in Fig. 1. is looking for a discourse entity called 'e' as its "backward looking center" and exports the same discourse entity as its "forward looking center". The combination of these two trees therefore yields a coherent discourse, which is expected to be continued with an utterance centered on 'e'.

5 Polarity Filtering on Discourse Entities

By treating discourse entities on D_c nodes as an additional polarity key we can apply the polarity filtering technique on the discourse level. This means we can filter out lexical selections that wouldn't lead to a coherent discourse the same way as those lexical selections are filtered out which won't lead to a syntactically well formed sentence. To give an example, given the semantic representation in Figure 4 potential realizations by a generator which is not aware of discourse coherence would include both of the examples in (1).

As an experiment, we generated the above example using the same input but two different grammars. In the first case we used a grammar which consists of discourse-level trees but no annotations for discourse entities. The realizer pro-

```
h0:white_cream(e)
h1:elixir(e)
h2:fda(f)
h3:approve(f e)
h4:contain(e a)
h5:aliprosan(a)
h6:relieve(a v)
h7:viral_skin_disorders(v)
```

Figure 4: Input for the sentences in (1)

duced 192 solutions, including many incoherent ones such as (1a). In the second case, we used a grammar with the same trees, but annotated with discourse referents. In this case the realizer produced only 16 solutions, all of which maintained referential coherence. In the first case, the grammar provided 128 ways to associate trees with the input (tree sets), and the 192 solutions included all possible sentence orders. Since for most trees in the grammar there are more than one ways to annotate them with discourse referents, in the second case the grammar contained more trees (differing only in their discourse referent asignments). In this case there were 1536 tree sets selected for the same input. Of these, 1320 were discarded by polarity filtering on discourse entities. Of the remaining 216 tree sets 200 were ruled out by feature unification when the trees were combined.

Figures 2 and 3 illustrate two sets of trees that were selected by the realizer, corresponding to the examples in (1). Discourse-level polarity filtering in this example (for the input in (4)) discards all tree sets whose polarities don't sum up to one of the discourse entities, i.e., `+e`, `+a`, `+f` or `+v`. The polarity of the tree set in Fig.2 is `+2a -v` so the tree set is discarded. For the tree set in Fig.3 the polarities sum up to `+e` and the realizer attempts to combine the trees, which in this case leads to a referentially coherent solution (1b).

The search space of the realizer can be further restricted by only allowing tree sets whose polarities sum up to a specific discourse entity. In this case the realizer will produce paragraphs where the center of attention in the last sentence is the discourse entity used for polarity filtering.

6 Conclusions

We have described a discourse-level extension of Tree Adjoining Grammar which tightly integrates syntax with discourse and includes a representation of discourse entities. We have shown that including discourse entities in the grammar of a surface realizer improves the coherence of the generated text and that these variables can also be used in a very efficient optimization technique, polarity filtering, to filter out referentially incoherent solutions.

References

D.E. Appelt. 1985. *Planning English sentences*. Cambridge University Press, Cambridge.

L. Danlos. 2000. G-TAG: A lexicalized formalism for text generation inspired by Tree Adjoining Grammar. In A. Abeille and O. Rambow, editors, *Tree Adjoining Grammars: Formalisms, linguistic analysis and processing*, pages 343–370. CSLI, Stanford, CA.

C. Gardent and E. Kow. 2006. Three reasons to adopt TAG-based surface realisation. In *Proceedings of TAG+8)*, Sydney/Australia.

B.J. Grosz, A.K. Joshi, and S Weinstein. 1995. Centering: a framework for modelling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

R. Kibble and R. Power. 2004. Optimizing referential coherence in text generation. *Computational Linguistics*, 30(4):401–416.

R. Kibble. 1999. Cb or not Cb? centering theory applied to NLG. In *ACL workshop on Discourse and Reference Structure*, pages 72–81.

R. Kibble. 2001. A reformulation of rule 2 of centering theory. *Comput. Linguist.*, 27(4):579–587.

A. Koller and M. Stone. 2007. Sentence generation as planning. In *Proceedings of ACL*.

A. Koller and K. Striegnitz. 2002. Generation as dependency parsing. In *Proceedings of ACL*.

E. Kow. 2007. *Surface realisation: ambiguity and determinism*. Ph.D. thesis, Universite de Henri Poincare - Nancy 1.

J. Robin. 1994. *Revision-based generation of Natural Language Summaries providing historical Background*. Ph.D. thesis, Columbia University.

J. Shaw. 2002. *Clause Aggregation: An approach to generating concise text*. Ph.D. thesis, Columbia University.

M. Stone, C. Doran, B. Webber, T. Bleam, and M. Palmer. 2003. Microplanning with communicative intentions: The SPUD system. *Computational Intelligence*, 19(4):311–381.